Reformulating Theories of Action for Efficient Planning

D. Paul Benjamin

School of Computer and Information Science, Syracuse University
Center for Science & Technology, Syracuse, NY, 13244-4100 USA
benjamin@top.cis.syr.edu http://www.cis.syr.edu/people/benjamin

Abstract

Domain theories are used in a wide variety of fields of computer science as a means of representing properties of the domain under consideration. These fields include artificial intelligence, software engineering, VLSI design, cryptography, and distributed computing. In each case, the advantages of using theories include the precision of task specification and the ability to verify results. A great deal of effort has gone into the development of tools to make the use of theories easier. This effort has met with some success. However, a fundamental problem remains: the choice of symbolic formulation for a theory, including both the choice of features for describing the environment and the design of abstractions that encode the actions. This paper describes fundamental research on the algebraic structure of the representations of domain theories. The perspective of this work is to view a problem's state space as though it were physical space, and the actions in the state space as though they were physical motions. A domain theory should then state the laws of motion within the space. Following the analogy with physics, a representation is a coordinate system, and theories are transformed by changing coordinates. This permits symbolic computational techniques to be used to transform theories and find useful decompositions. A system has been implemented using Mathematica and GAP that performs these computations. The mathematical basis for this approach is given, and the computations are illustrated by examples.

1 Formulations of Theories of Action

Each theory can be represented in a large number of different ways that vary in their computational effectiveness. A good choice of symbols and a good choice of formulation using those symbols are absolutely necessary for effective symbolic computation. A simple example is given by the following three representations for the two-disk Towers of Hanoi.

Representation TOH1: Let the nine states of the 2-disk Towers of Hanoi be $\{ (b, s), 1 \le b \le 3, 1 \le s \le 3 \}$, where b and s are the numbers of the pegs the big and small disks are on, respectively. Let the two actions be:

X: $(b, s) \rightarrow (b, s\,(\mathrm{mod}(3)) + 1)$

Y: $(b, s) \rightarrow (b + 1\,(\mathrm{mod}(3)) + 1, s)$

X moves the small disk right one peg (wrapping around from peg 3 to peg 1), and Y moves the large disk one peg to the left (wrapping around from peg 1 to peg 3). X is always executable, but Y can be executed only in three states.

Representation TOH2: Let the states be the same as TOH1, and let the six possible actions be:

X1 = move the top disk from peg 1 to peg 2

Y1 = move the top disk from peg 1 to peg 3

X2 = move the top disk from peg 2 to peg 3

Y2 = move the top disk from peg 2 to peg 1

X3 = move the top disk from peg 3 to peg 1

Y3 = move the top disk from peg 3 to peg 2

Each of these actions is executable in four states.

Representation TOH3: Let the states be the same as TOH1, and let the two possible actions be:

X: $(b, s) \rightarrow (b, s\,(\mathrm{mod}(3)) + 1)$

Z: $(b, s) \rightarrow (b + 1\,(\mathrm{mod}(3)) + 1, s + 1\,(\mathrm{mod}(3)) + 1)$

X is as before, and Z is a macro-action that moves both disks left one peg.

Each of these representations can be implemented in terms of every other. For example, Z in TOH3 is implemented as a disjunction of sequences of actions from TOH1: Z = {XXY, XYX, YXX} or from TOH2: Z = {X1Y1X2, X2Y2X3, X3Y3X1}.

One of the most important properties of a representation for planning is the mutual interference of its actions. The actions in TOH3 are independent controls; X solves the position of the small disk, and Z solves the big disk. X does not changethe position of the big disk, and Z does not change the relative positions of the disks. The disks can be solved in either order. This representation captures the important property of the Towers of Hanoi task: the disks can be solved in any order. In representations TOH1 and TOH2, this property was obscured by details of the implementations of the disk moves.

Even for this simple puzzle there are many possible formulations. The three formulations given above differ in that the first indexes the moves according to the disk moved, whereas the second indexes the moves by the peg moved from (or to, for the inverse moves.) The third representation eliminates subgoal interference, which is present in the first two representations. One advantage of the first and third formulation is clear: they scales up to theories for more disks, because their actions will be included unchanged in those theories. New moves will be added for the new disks. However, the actions in the second formulation must be redefined as more disks are added, so this theory does not scale up. The result is that analysis and synthesis of the two-disk problem performed using the first or third formulation can be reused when larger problems are attempted, but analysis and synthesis from the second formulation must be discarded when larger problems are attempted. Furthermore, the properties of these two representations scale up to all Towers of Hanoi problems, e.g., the family of representations based on TOH3 have no subgoal interference. The third representation is clearly the best for efficient planning in this domain.

Computer science is not the first field to be faced with the problem of properly formulating theories. Throughout the history of science, it has always been desirable to formulate theories in as general a way as possible, so that important regularities are identified and separated from details particular to individual situations. In particular, physics has had a great deal of success in formulating theories of wide generality yet high predictive accuracy. In this paper, we will see that many of the mathematical structures employed in the statement of physical theories can be usefully generalized to the statement of abstract theories.

We will begin with a brief discussion of the properties of physical theories in the next two sections. The remainder of the paper discusses the appropriate mathematics for analyzing these properties, and gives examples of the analysis and reformulation of theories.

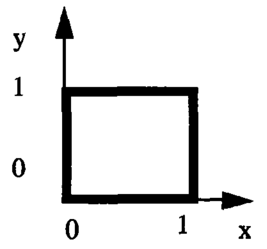
2 Invariants of Laws

The ability to formulate any law of nature depends on the fact that the predictions given by the law, together with certain initial conditions, will be the same no matter when or where the results of the predictions are observed. In physical theories, the fact that absolute time and location are never relevant is essential for the statement of laws; without this fact, general laws could not be stated, and the complexity of the world would eliminate the possibility of intelligent comprehension of the environment. This irrelevance is stated in terms of the invariance of laws under translation in time and space. Such invariance is so self-evident that it was not even stated clearly until less than a century ago. It was Einstein who recognized the importance of invariance in the formulation of physical law, and brought it to the forefront of physics. Before Einstein, it was natural to first formulate physical law and then derive the laws of invariance. Now, the reverse is true. As the eminent physicist Wigner states, "It is now natural for us to try to derive the laws of nature and to test their validity by means of the laws of invariance, ..."(Wigner, 1967, p.5). This is especially clear in the development of quantum mechanics.

Invariance is important not only in physics. As Dijkstra states, "Since the earliest days of proving the correctness of programs, predicates on the programs's state space have played a central role. This role became essential when non-deterministic systems were considered.... I know of only one satisfactory way of reasoning about such systems: to prove that none of the atomic actions falsifies a special predicate, the so-called 'global invariant'." (Dijkstra, 1985.) In other words, as the system moves in its state space, the global invariant is a law of motion. Dijkstra goes on to point out the central difficulty in the use of invariants: "That solves the problem in principle; in each particular case, however, we have to choose how to write down the global invariant. The choice of notation influences the ease with which we can show that, indeed, none of the atomic actions falsifies the global invariant." We need a mathematics of invariance to help us formulate invariants.

3 Symmetry

A symmetry is a mapping of an object or system to itself such that the result of the mapping is indistinguishable from the original. For example, the human body (idealized) has a left-right symmetry. The following square has a number of symmetries, including flipping it onto itself about the lines x = 1/2 or y = 1/2 or x = y, and rotating it ninety degrees either clockwise or counterclockwise:



$x = 0, \quad 0 \leq y \leq 1$

$x = 1, \quad 0 \leq y \leq 1$

$y = 0, \quad 0 \leq x \leq 1$

$y = 1, \quad 0 \leq x \leq 1$

Symmetries can exist in physical space or in state space. For each invariance, there is a corresponding set of symmetries, each of which maintains the invariant. For example, the invariance of physical law under translation in space corresponds to the symmetries of space under all translations. Also, the global invariant of a non-deterministic program corresponds to all permutations of the atomic actions; each permutation maintains the invariant. This correspondence holds in reverse, also. For each set of symmetries, there is a corresponding invariant.

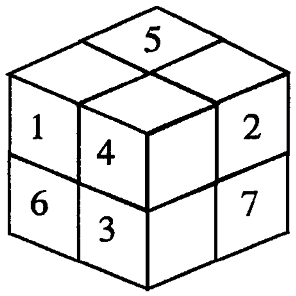
For example, the square above can be represented by the theory shown to its right. This theory has syntactic symmetries corresponding to the symmetries of the square, e.g. interchanging x and y gives the flip about the line x = y, and interchanging the first and second lines of the theory flips the square about the line x = 1/2.

Many of the important symmetries in physics are geometric. In other words, they are symmetries of the space in which motion takes place. By viewing a program as "moving" in its state space, we can take the same approach as physicists: formulate geometric symmetries of the space, and use them to derive invariants, thereby obtaining laws governing the use of the program.

4 The Mathematics of Symmetry: Group Theory

Given a global invariant, the corresponding set of transformations is closed under composition, and for every transformation, its inverse is also a transformation that preserves the invariant. The identity transformation is also always in the set. Thus, the set of transformations form a *group*. Group theory is the language of symmetries, and has assumed a central role in modern physics. Group theory can also be used to analyze the symmetries of a task and derive invariants, which are then used to synthesize a program. The following example is given in Benjamin (1994). (This paper will not provide any background in group theory. The reader is referred to any standard text.)

Let us begin by examining a simple task, the 2x2x2 Rubik's Cube with 180-degree twists (we use such a small example for brevity of presentation, but the techniques are generally applicable, as will be shown). Let the 8 cubicles (the fixed positions) in the 2x2x2 Cube be numbered in the following way (8 is the number of the hidden cubicle):



The goal configuration for the 2x2x2 Rubik's Cube with 180-degree twists.

Number the cubies (the movable, colored cubes) similarly, and let the goal be to get each cubie in the cubicle with the same number. For brevity of presentation, we will consider only 180° twists of the cube. Let f, r, and t denote 180° clockwise turns of the front, right, and top, respectively (cubie 8 is held fixed; Dorst (1989) shows that this is equivalent to factoring by the Euclidean group in three dimensions). Note that this cubie numbering is just a shorthand for labeling each cubie by its unique coloring. This holds true for the Cube with only 180° twists, as position determines orientation.

The actions for the Cube can be represented as a group, which is generated by the actions f, r, and t. We use group representation theory to represent f, r, and t as matrices:

$$f = \begin{bmatrix} 0&0&1&0&0&0&0 \\ 0&1&0&0&0&0&0 \\ 1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&1&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1 \end{bmatrix} \quad r = \begin{bmatrix} 1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1 \\ 0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&1&0&0&0 \end{bmatrix}$$

$$t = \begin{bmatrix} 0&1&0&0&0&0&0 \\ 1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1 \end{bmatrix}$$

These matrices are 7-dimensional, corresponding to the 7 unsolved cubies. We find eigenvectors of eigenvalue 1; these are the invariants. Any invariant of all the actions is irrelevant and can be projected away. To do this, we first change the coordinate system so that the invariant eigenvectors are axes, and then project to the noninvariant subspace, removing all irrelevant information at once. In this case, the eigenvectors are:

$$r: \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \text{ for } \lambda = 1, \text{ and } \begin{bmatrix}0\\-1\\1\end{bmatrix} \text{ for } \lambda = -1$$

$$f: \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \text{ for } \lambda = 1, \text{ and } \begin{bmatrix}-1\\0\\1\end{bmatrix} \text{ for } \lambda = -1$$

$$t: \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \text{ for } \lambda = 1, \text{ and } \begin{bmatrix}-1\\1\\0\end{bmatrix} \text{ for } \lambda = -1$$

and the common invariant eigenvector is: $\begin{bmatrix}1\\1\\1\end{bmatrix}$.

Note that we have shortened these eigenvectors to save space; they are actually 7-vectors, with additional zeroes. We then change the basis. The appropriate matrix is:

$$P = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & 0 & 0 & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & 0 & 0 & 0 \\ \frac{1}{\sqrt{3}} & 0 & 0 & \frac{-2}{\sqrt{6}} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & 0 & \frac{-1}{2} & \frac{-1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & 0 & \frac{-1}{2} & \frac{1}{2} & \frac{-1}{2} \\ 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & \frac{-1}{2} & \frac{-1}{2} \end{bmatrix}$$

yielding the new representations for r, f, and t:

$$r = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix} \quad f = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 & 0 & 0 \\ 0 & 0 & -\frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

$$t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This well-known procedure computes the irreducible invariants of a group. The irreducible factors of dimension 1, 1, 2, and 3 are found along the diagonals of the matrices. Projecting to these subspaces yields two subproblems:

$$r = \begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix} \quad f = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix} \quad t = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$r = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad f = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \quad t = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

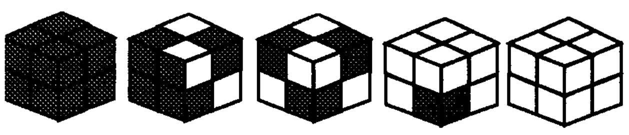
On cubelets 1, 2, and 3, the subgroup generated is {i. r. f. t. rt. tr}

On cubelets 4, 5, 6, and 7, the subgroup generated is: {i, r, f, t, rf, rt, fr, ft, tr, tf, rfr, rft, rtr, rtf, frt,ftr, ftf, trf, tfr, rfrt, rftr, rftf, rtrf, rtfr}.

Using each set of matrices as generators, we get two subgroups of actions, the second of which is a faithful representation of the whole group. The first subgroup moves cubies 1, 2, and 3, while holding 4, 5, 6, and 7 in position. The second subgroup moves cubies 4, 5, 6, and 7 while holding 1, 2, and 3 in their positions. We then repeat this procedure on the first set of actions to obtain a full set of prime factors of the group.

These factors can be assembled in different ways to form serial algorithms. There is more than one way to decompose this group, analogous to the different ways of listing the prime factors of a number. Five serial algorithms are obtained in this way. We examine one of them.

R = {i, frtr} {i, rtft} {i, t} {i, r, f}



One of { i,r,f } brings cubelet 3 into cubicle 3.

One of { i,t } brings cubelet 1 and 2 into their places.

One of { i, rtft } brings (4,6) and (5,7) in the proper planes (`the front face looks right').

One of { i, frtr } finishes the Cube.

The above figure is read right-to-left; shaded cubicles have been solved. "i" denotes the identity (null) action.

Each step in the algorithm brings a subset of cubies to its goal value. Subsequent steps hold that pattern of cubies invariant. In this way, a divide-and-conquer algorithm is synthesized. For example, the first step solves cubicle 3. Knowing the colors of the solved cubicle 8, we know the colors of cubicle 3 - it has the same color as the bottom of cubicle 8, and two new colors. There is only one such cubie, and it must be in one of three locations: in its goal position, or in cubicle 1 or 2. The system need only examine those 3 values to determine what action to take. Once cubicle 3 is solved, it need not be examined again. The system next solves cubicles 1 and 2; it need only examine either position to see if the proper cubie is there; if so, it does nothing, otherwise, it twists the top. Finally, the system uses the appropriate sequence of actions to solve the remaining four cubicles, by first examining the front face to see if it is a uniform color, and then examining the top or right face to see if it is of uniform color.

We have transformed the global 7-dimensional description of the Cube into a composition of local descriptions, each characterized by a set of symmetries. This decomposition has average cost of 5.17, whereas an optimal solution is of average length 2.46. This is the usual cost of planning by decomposition: the solution need not be optimal.

We see that there are two "subspaces" of cubies in this Cube: one consisting of cubicles 1,2,3 and 8, and the other of cubicles 4,5,6, and 7. For each subspace, there is a subprogram that interchanges the cubies in its cubicles while holding the cubies in the other subspace invariant, in exactly the same way that the X and Z actions in TOH3 are independent controls. We are thus justified in thinking of these two sets of cubicles as *independent* subspaces. Recognizing the symmetries that characterize such subspaces is essential for synthesizing such algorithms.

5 The Mathematics of Local Symmetries: Inverse Semigroups

The previous section illustrates how the analysis of global symmetries can be useful in synthesis. However, global symmetries, and in general global properties, are not sufficiently general. We must also examine *local symmetries*, which are symmetries between parts of a system, rather than of the whole system. Local symmetries correspond to local invariants, which are predicates that hold for some part of a system, but not necessarily for the whole system. An example is a loop invariant in a program, which holds during the execution of the loop, but not necessarily elsewhere. This is the sort of invariant more often encountered in computing.

To handle local symmetries, we use a more general theory than group theory: the theory of inverse semigroups. A semigroup is a closed, associative system, and an inverse semigroup has the additional property that every action is invertible. The difference between a group and an inverse semigroup is that the transformations in the group must be globally defined (they correspond to global symmetries) whereas those in the inverse semigroup can be defined on only part of the space, and are thus partial functions on the space. Inverse semigroups are thus more appropriate for reasoning about programming constructs, e.g., rules, which can be defined only on some variable bindings.

We want to understand the structure of the space of possible formulations for a given model. Appropriate mathematical tools for this work come from algebraic linguistics, which studies structural properties of languages and their transformations. A formulation corresponds to a presentation of a semigroup.

In physical theories, space-time is represented in terms of a coordinate system. Invariance under translation in time or space then becomes invariance under coordinate change. Inverse semigroups possess a similar notion of coordinate system, and we use this in the same way.

We consider a system formulation $M = (Q, A, \delta)$ to be a set A of actions defined as partial functions on a state space Q, together with a mapping $\delta: Q \times A \rightarrow Q$ that defines the state transitions. We are not concerned with the syntactic details of the encoding of the actions A, but rather with which actions should be labeled the same and should therefore be considered instances of a common abstraction. In other words, we are concerned with the algebraic structure of A. Without loss of generality, we restrict our attention to semigroups of partial 1-1 functions on the state space Q (Howie, 1976). This formalism is general, encompassing nondeterministic systems and concurrent systems.

To analyze the structure of such a semigroup of transformations, a usual step is to examine Green's relations (Lallement, 1979). Green's equivalence relations are defined as follows for any semigroup S:

$a \, R \, b$ iff $aS^1 = bS^1$ $\quad$ $a \, L \, b$ iff $S^1a = S^1b$

$a \, J \, b$ iff $S^1aS^1 = S^1bS^1$ $\quad$ $H = R \cap L$

$D = RL$

where S^1 denotes the monoid corresponding to S (S with an identity element 1 adjoined). Intuitively, we can think of these relations in the following way: aRb iff for any plan that begins with "a", there exists a plan beginning with "b" that yields the same behavior; aLb iff for any plan that ends with "a", there exists a plan ending with "b" that yields the same behavior; aHb indicates functional equivalence, in the sense that for any plan containing an "a" there is a plan containing "b" that yields the same behavior; two elements in different D-classes are functionally dissimilar, in that no plan containing either can exhibit the same behavior as any plan containing the other.

Green's relations organize the actions of a transformation semigroup according to their functional properties, and organize the states according to the behaviors that can be exhibited from them. Utilizing Green's relations, Lallement defines a local coordinate system:

Definition. Let D be a D-class of a semigroup, and let $H_{\lambda\rho}$ ($\lambda \in \Lambda, \rho \in P$) be the set of H-classes contained in D (indexed by their L-class and R-class). A coordinate system for D is a selection of a particular H-class H_0 contained in D, and of elements $q_{\lambda\rho}, q'_{\lambda\rho}, r_{\lambda\rho}, r'_{\lambda\rho} \in S^1$ with $\lambda \in \Lambda$, $\rho \in P$ such that the mappings $x \rightarrow q_{\lambda\rho} x r_{\lambda\rho}$ and $y \rightarrow q'_{\lambda\rho} y r'_{\lambda\rho}$ are bijections from H_0 to $H_{\lambda\rho}$ and from $H_{\lambda\rho}$ to H_0, respectively. A coordinate system for D is denoted by $[H_0;\{(q_{\lambda\rho}, q'_{\lambda\rho}, r_{\lambda\rho}, r'_{\lambda\rho}): \lambda \in \Lambda, \rho \in P\}]$.

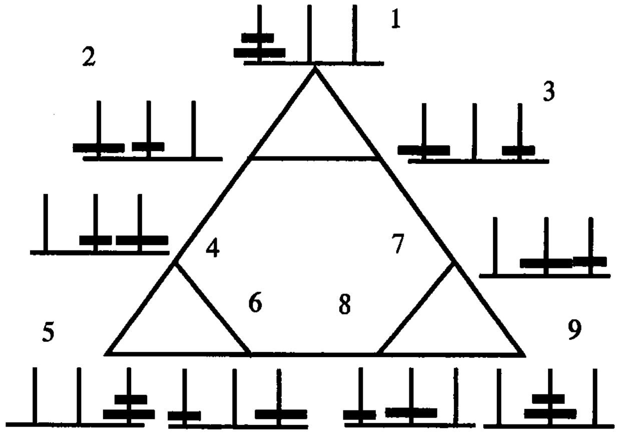
There may be more than one local coordinate system for a D-class. Each coordinate system gives a matrix representation in much the same way that a coordinate system in a vector space gives a matrix representation, permitting us to change coordinates by performing a similarity transformation (inner automorphism) in the usual way (the reader is referred to Lallement for details).

Each local coordinate system within the semigroup expresses a distinct syntactic labeling of a subsystem. Reformulation to improve computational properties is done by transformation of coordinates in the familiar manner (inner automorphisms). As we vary the coordinate system, each action varies through an equivalence class of sequences of actions - its reformulations.

Any property that is invariant under all local coordinate transformations is an abstract property; otherwise, it is an implementation property. In particular, the abstract syntactic formulations of inputs, which should not reflect implementation distinctions, can be determined by factoring a given semigroup by its group of local coordinate transformations. The quotient semigroup thus obtained is the abstract language implemented in the given system, and the system is decomposed into the product of the quotient and the kernel, thereby facilitating analysis and synthesis.

6 Example: The Towers of Hanoi

Let us number the nine states of the 2-disk Towers of Hanoi as follows:



Let us use representation TOH1. The two actions X and Y generate a semigroup of 31 distinct partial functions on the states. Green's relations for this semigroup are:

D0 $\boxed{0}$ $\quad$ D1 $\boxed{x, xx, xxx}$

D2

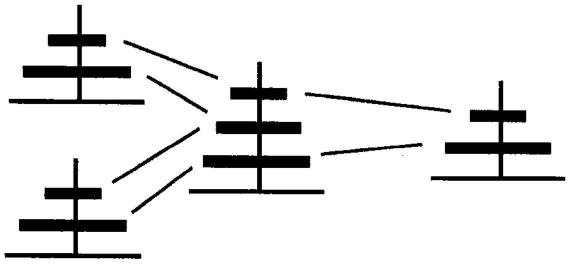
xyx xyxxyx xyxxyxxyx	xy xyxxy xyxxyxxy	xyxx xyxxyxx xyxxyxxyxx
xxyx xxyxxyx xxyxxyxxyx	xxy xxyxxy xxyxxyxxy	xxyxx xxyxxyxx xxyxxyxxyxx
yx yxxyx yxxyxxyx	y yxxy yxxyxxy	yxx yxxyxx yxxyxxyxx

There are three D classes, shown as the three separate large boxes. In each D class, the R classes are rows and the L classes are columns, and they intersect in the small boxes, which are H classes. Note that D0 and D1 consist of only one R class and one L class, and hence one H class. The idempotents are in bold type.

There are no nontrivial inner automorphisms of D0 and D1. The group of inner automorphisms of D2 is a cyclic group of order three. These coordinate transformations are global within D2, but are local in the semigroup. These inner automorphisms are calculated by the matrix techniques explained by Lallement [7]. A generator for this group is the automorphism that maps xyx to xxy, xxy to yxx, and yxx to xyx. Factoring D2 by this map gives:

```
Define z = case {
    little disk left of large disk:   xxy
    little disk on large disk:        xyx
    little disk right of large disk:  yxx }
where z is a new symbol.
```

This is representation TOH3. This representation applies not only to the two-disk problem, but to all Towers of Hanoi problems. This is seen by forming free products of the semigroup with itself, amalgamating the coordinate axes in all possible ways. There are three free products of this semigroup with itself that amalgamate coordinates: D2 can be identified with itself, D1 with itself, and D2 with D1 (by mapping xxy, xyx, and yxx to x). These correspond to identifying the moves of the larger disk of one copy of the semigroup with the moves of the larger disk in another copy, identifying the moves of the smaller disk with the moves of the smaller disk, and identifying the moves of the larger disk with the moves of the smaller disk, respectively. The result of this construction is the three-disk Towers of Hanoi, which is viewed as three copies of the two-disk task running concurrently.



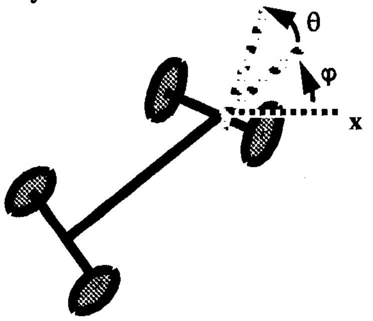
From the logical perspective, this can be viewed as composing three copies of a theory for the two-disk Towers of Hanoi, to yield a valid theory for the three-disk problem. Copies of the theory are joined by unifying variables between theories. Two copies joined at the middle disk will not suffice, as then the largest disk could be placed on the smallest disk. A third copy of the theory prohibits this. The purpose of computing a coordinate system is that the coordinate axes determine what to unify.

The interaction of these three smaller tasks yields a set of relations. For example, consider moving the middle disk one peg to the right. When considering this disk as the larger disk in the task consisting of the upper two disks, this move is xyxxy (in the state when all disks are on the left peg), yxxyx (when the smallest disk is to the right of the middle disk), or xxyxxyxx. On the other hand, when considering this disk as the smaller disk in the task consisting of the lower two disk, this move is x (in the other copy of the semigroup). This means that we add the relation xyxxy = yxxyx = xxyxxyxx to the presentation of the semigroup.

This method of composing larger tasks from smaller ones guarantees that this decomposition generalizes to any Towers of Hanoi problem with n disks, by considering a coordinate system for the three-disk task, and forming free products with amalgamation in the same manner as before. These free products with amalgamation will always reduce to free products with amalgamation of coordinate systems of the two-disk task, so that by induction the properties of the two-disk task determine the properties of all Towers of Hanoi tasks.

7 Example: A Robot Vehicle

This procedure has been applied to a number of more interesting examples, including one given by Nelson (1967). Consider a mobile robot or car moving on a smooth, 2-dimensional surface. The configuration space of the car is an open submanifold of $R^2 \times T^2$, parameterized by (x, y, φ, θ), where x and y are the Cartesian coordinates of the center of the front axle, φ is the angle of the car measured counterclockwise from the positive x-axis, and θ is the angle made by the front wheels with the car.



T is the torus generated as the angles vary between $-\pi$ and π, and θ is constrained to vary between $-\theta_{max}$ and θ_{max}, which gives the submanifold of $R^2 \times T^2$. The two control fields are:

$$\text{Steer} = \frac{\partial}{\partial\theta} \qquad \text{and}$$

$$\text{Drive} = \cos(\varphi + \theta)\frac{\partial}{\partial x} + \sin(\varphi + \theta)\frac{\partial}{\partial y} + \sin\theta\frac{\partial}{\partial\theta}.$$

Let S(t) and D(t) be the flows generated by Steer and Drive, respectively. Each of these flows constitutes a semigroup. The semigroup S of motions of the car is generated by these two semigroups, together with a set of commutation relations relating S and D, e.g., if the driver turns the wheels a certain angle then drives a certain distance, the car ends up where it started.

We apply the same reformulation procedure used on the previous example. The reformulation procedure chooses a small piece of the configuration space, such as a square Q large enough to permit the car to maneuver to reach every configuration in the square. Analyzing Green's relations for these flows in Q, the reformulation algorithm finds those sub-semigroups of maximal symmetry in Q. These are the circles in x-y space, the curves of constant θ that return to their starting point. The inner automorphisms of these spheres are the rotational symmetries, and factoring by this group gives the familiar decomposition of planning the car's motion into planning position and then planning rotation. Synthesis in the new representation is done by synthesizing the path in the x and y dimensions, e.g., avoiding obstacles, then lifting this path into the φ dimension by planning the necessary orientation changes, then lifting this path into the θ dimension by planning the necessary turns of the steering wheel. This decomposition applies to all surfaces that can be composed from the base case (the squares) which include all the usual roads, parking lots and garages, etc.

8 Related Work

Sacerdoti (1974), Knoblock et al. (1991), and Unruh & Rosenbloom (1989), among others, describe techniques for building an abstraction hierarchy. For example, in ABSTRIPS an ordering was imposed on the state-description predicates; bringing the predicates to their goal values in this order was viewed as top-down search in a hierarchy of abstract problem descriptions.

Niizuma & Kitahashi (1985) and Banerji & Ernst (1977) describe techniques for projecting the states to a quotient space. In this view, an equivalence relation is imposed on the states, and the equivalence classes are the states in the quotient space. The only actions retained in the new representation are those that move between equivalence classes.

Zimmer (1990) and Benjamin et al. (1990) describe ways for decomposing the actions. In this approach, the set of sequences of actions is decomposed into two sets: those that are most relevant (according to some criterion) for solving the problem, and those that are less relevant. This induces an equivalence relation on the set of states, as in the previously described approach; a difference is that sequences of actions (macros) are used, rather than actions. The decomposition procedure is then repeated on the less relevant actions.

A similar approach is taken by Subramanian (1987), who drops statements from a theory if the reduced theory can still derive the goal statement; the dropped statements are considered irrelevant. In these approaches, the state space is reduced by removing states that can no longer be reached by actions (statements) retained in the representation (theory). These approaches differ from the state projection approach mainly in the order in which states and actions are reformulated. In the state projection approach, a feature is chosen, inducing an equivalence relation that factors the states and decomposes the actions. In the action decomposition approach, the sequences of actions are decomposed according to some criterion, e.g., irrelevance, which induces an equivalence relation on the states.

Korf (1983) and Riddle (1986) describe methods for serializing the subgoals. Finding a set of serializable subgoals for a problem permits solution of the problem by solving each subgoal in order, which can be viewed as a hierarchy of abstract goals. Korf points out that this reduces the exponent of the search, possibly resulting in a big gain in efficiency.

However, none of these authors have addressed the problem in the general way described in this paper. The reformulation techniques used by the above authors merely elucidate the structure of a given representation, and thus possess a serious limitation: they must preserve the structure of the representation, and are thus limited in the reformulations that they can produce. Such techniques can only remove extraneous information to uncover existing structure in a given representation. If this structure is not appropriate for efficient problem-solving, then their techniques will be of little use.

For example, in ABSTRIPS (Sacerdoti, 1974) the relevant predicates must already exist in the initial representation, or numbers cannot be assigned to them. In Subramanian's work, if the theory is stated in such a way that the irrelevant information is distributed among all the statements of the theory, rather than concentrated in a subset of the statements, then it cannot be dropped without rendering the theory incapable of solving the task. TOH2 is an example of such a representation. Earlier work by Ernst & Goldstein(1982) achieved results by restricting the permissible strategies to a specific class.

In contrast, reformulation by factoring by the group of coordinate transformations is fully general, as it applies to any semigroup of transformations on a state space. The only projects that are similar in scope to the research described in this paper are those of Lowry (1987, 1987a) and van Baalen (1989, 1992), both of whom use reformulation techniques that can modify structure in a general way. Their research is not in disagreement with the ideas presented in this paper. However, the techniques and results of this research project are very distinct from Lowry's and van Baalen's. The use of semigroup coordinate systems gives a formal yet intuitive understanding of representations, and yields new insights, such as viewing the n-disk Towers of Hanoi as composed of concurrent copies of the 2-disk task, which leads to the extension of the abstract properties of the 2-disk task to the general case.

9 Summary

Subgoal decompositions are crucial for effective planning and learning, but often the useful decompositions are obscured by implementation detail. This work views a representation of a theory of actions as a language, and analyzes its structure to identify those properties that are invariant under reformulation. These properties include the subgoal decompositions.

The advantage of planning in a decomposed formulation is obvious: the size of the search space is no longer the product of the sizes of the search spaces for each component of the decomposition, but instead is the sum. A similar advantage holds for learning: once the formulation has been decomposed, each component can be learned independently of the others, thereby reducing the size of the hypothesis space to the sum of the sizes of the hypothesis spaces for each component. For example, learning the fastest way to change orientation is independent of learning the most direct path between two positions. It is possible for a problem-solver to learn different dimensions by different means, e.g. learning the path between two positions by being told and learning how to turn by exploration.

In addition, the fact that the decomposition of the base case scales up to large systems guarantees that anything learned in one part of the system applies everywhere. This follows because every large system in the class generated by the base case is locally everywhere isomorphic to the base case.

10 Acknowledgements

This research was supported in part by NSF grant 9509537, and in part by AFOSR grant F49620-93-C-0063.

References

Benjamin, D. Paul, (1994). Formulating Patterns in Problem Solving, *Annals of Mathematics and Artificial Intelligence*, special issue on Mathematics in Pattern Recognition, Vol. 9, No. III-IV.

Benjamin, D. Paul, (1992). Reformulating Path Planning Problems by Task-preserving Abstraction, *Journal of Robotics and Autonomous Systems*, 9, pp. 1-9.

Howie, J. M., (1976). An Introduction to Semigroup Theory, Academic Press.

Lallement, Gerard (1979). Semigroups and Combinatorial Applications, Wiley & Sons.

Nelson, Edward, (1967). Tensor Analysis, Mathematical Notes, Princeton University Press.

Petrich, Mario, (1984). Inverse Semigroups, John Wiley & Sons, Inc., New York.