

Connecting Perception and Action by Associating Symmetries in Vision and Language*

D. Paul Benjamin

School of Computer Science and Information Systems, Pace University
1 Pace Plaza, New York, NY, 10038 USA
benjamin@sol.pace.edu <http://sol.pace.edu/~benjamin>

Abstract

Symmetries are of fundamental importance in information processing. It is well understood how important visual symmetries are in the understanding of scenes. This paper explores the symmetries of languages of actions, and relates them to visual symmetries. Linguistic symmetry is explored in the context of problem solving. Algebraic techniques permit the representation and manipulation of linguistic symmetry, and the exploration of its connection to perceptual symmetry. It is shown that the invariants of the decomposition of a language of actions form a complete and independent set of perceptual features that naturally describe the search space.

1. Introduction

Visual symmetry is a fundamental concept widely used to categorize objects in computer vision. Linguistic symmetry is equally fundamental, e.g. in Galois theory, in which symmetry groups of equations determine whether equations are solvable or not. This paper briefly presents how these two forms of symmetry can be related in a robot architecture.

The next section briefly describes the formalism used to reason about and manipulate linguistic symmetry. The techniques presented in this section are then illustrated on two examples. Finally, the robot

architecture is briefly discussed.

2. The Algebraic Structure of Languages

We consider a language of actions $M = (Q, A, \delta)$ consisting of 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. Without loss of generality, we can restrict our attention to semigroups of partial 1-1 functions on the state space Q [5]. This formalism is very general. It encompasses nondeterministic systems and concurrent systems. In this short paper, deterministic examples are given due to space considerations. Two examples are provided to illustrate the method. The reader is referred to earlier papers for further details and examples [1], [2], [3], [4]

The description of a system for the synthesis of a plan for M differs from the description of M in an essential way: planning an action is reversible whether or not the action itself is reversible (assuming the synthesizing system can backtrack.) Thus, the process of synthesizing plans can be described by a theory whose actions form an appropriate inverse semigroup containing the original actions together with newly added inverses corresponding to backtracking.

To analyze the structure of such a semigroup of transformations, a usual step is to examine Green's relations [7]. Green's equivalence relations are defined as follows: given any semigroup S :

*This research is supported by NSF grant 9509537 and AFOSR grant F49620-93-C-0063.

semigroup of 31 distinct partial functions on the states. Green's relations for this semigroup are:

D2	D0	0	D1	x, xx, xxx
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:

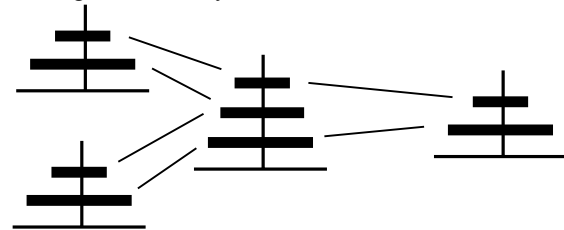
Define $z = \text{case } \{$
 little disk left of large disk: xyx
 little disk on large disk: xyx
 little disk right of large disk: yxx $\}$

where z is a new symbol.

x is as before, and z moves both disks left one peg. z is implemented as a disjunction of sequences of actions. x and z are independent controls; x solves the position of the small disk, and z solves the big disk. x does not change the position of the big disk, and z does not change the relative positions of the disks. The disks can be solved in either order, as these controls generate an abelian group. x and z are the unit vectors

of the axes of a coordinate system for this task. This new coordinate system captures an important property of the Towers of Hanoi task: the disks can be solved in either order. In the original theory, this property was obscured by details of the implementations of the disk moves.

This decomposition 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 result of computing a coordinate system is that the coordinate axes determine what to unify.

The unification of these three smaller theories 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 xxyxyxx. 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 $xyxyx = yxxyx = xxyxyxx$ to the presentation of the semigroup.

Adding all such relations transforms the original semigroup into a new semigroup. The reader may have noticed that the resulting semigroup is exactly the same as that obtained by factoring the two-disk theory by its group of local coordinate transformations. *This is always the case.* This can be seen by realizing that when the two-disk theory represents a subtask of a larger Towers of Hanoi task, the actions in the two-disk theory must be able to be formulated as actions in the larger theory. So, when considering all ways of composing two-disk theories to make larger Towers of Hanoi theories, we are considering all ways of formulating two-disk actions as n-disk actions, i.e., all ways of transforming coordinates within the two-disk theory.

Thus, 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.

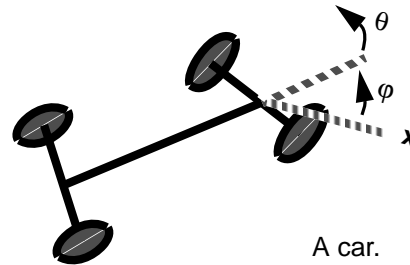
The features that are invariants of the components of the decomposition are the positions of the disks *relative to each other*. As these are the invariants of the decomposition they are the features that should be perceived. *The agent should represent the search space in terms of these invariants if it wishes to effectively solve the problem by subgoaling, so therefore the agent should see the problem environment in terms of these features.*

The importance of this linguistic transformation is

that it establishes the goal for the vision system of a robot attempting to solve this task: it must calculate the relative positions of the disks. Once this goal has been set, the robot can then bring to bear its knowledge about disks and pegs to guide the motions of the camera and to select appropriate methods for processing the visual data, e.g. detecting curved edges that are likely to be the edges of the disks, and ignoring information such as the color of the disks.

4. Example: A Car

Nelson [8] gives the following example of a car moving on a smooth, 2-dimensional surface. The configuration space of the car is an open submanifold of $\mathbb{R}^2 \times \mathbb{T}^2$, 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.



A car.

\mathbb{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 $\mathbb{R}^2 \times \mathbb{T}^2$. The two control fields are:

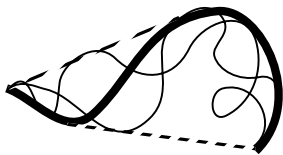
$$\text{Steer} = \frac{\partial}{\partial \theta} \quad \text{and}$$

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

These two fields do not span the entire four-dimensional configuration space, but as Nelson shows, the closure of these two fields under Lie products permits any motion to be approximated arbitrarily closely. 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.

Choose a small piece of this 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 , we see that each D -class corresponds to a class of curves with the same endpoints and convex hull. For example, in the next figure, the path drawn in bold is in the same D -class as all the other paths with the same endpoints that lie completely within its convex hull (dotted lines).



A D -class.

The reformulation algorithm finds those subsemigroups of maximal symmetry in Q . These are the circles in x - y space. These correspond to the D -classes of the curves of constant θ that return to their starting point. This structure is reminiscent of homotopy classes, except that curves of constant turning are used instead of curves of minimum length as the representatives of the classes. The inner automorphisms of these spheres are the rotational symmetries, and factoring by this group again gives the familiar decomposition of planning the car's motion into planning position and then planning rotation. The vector field for position changes is independent of ϕ and θ , and is simply \mathbb{R}^2 , and that for rotation consists of maneuvers that are independent of x and y , but utilize θ , in a manner similar to the discretized robot.

Steer, the control that modifies θ , was stated as an independent vector field, and thus planning turns of the steering wheel is a third component of the decomposition. This component was not present in the discretized mobile robot. 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. As before, this decomposition applies to all surfaces that can be composed from the base case (the squares.) The subsemigroups that are amalgamated are the circular neighborhoods within the squares.

The features that are the invariants of the components of the decomposition are the paths of constant curvature. It therefore makes sense for the vision system of an intelligent agent driving this car to detect curves of constant curvature and represent the world in terms of them. This provides the path planning system with the representation of the search space it needs to utilize the decomposition of path planning into position and orientation, and, as in the previous example, it determines the goal of the vision system.

5. An Architecture Based on Symmetry

A robot control architecture is currently under construction, using the SOAR cognitive architecture [6] as its basis, and the DARPA Image Understanding Environment to process visual data. The IUE has been selected because of its ability to create and process local coordinate systems. This architecture is being constructed for a Pioneer I robot equipped with camera and gripper.

SOAR is a cognitive architecture originally developed at CMU and undergoing continuing development at a number of locations, including the University of Michigan and the Information Sciences Institute. Knowledge in SOAR is represented as rules, which are organized into *problem spaces*. Each problem space contains the rules relevant to some aspect of the system's environment. In our system, some problem spaces contain rules describing the actions of the robot for particular tasks and subtasks; different coordinate systems (representations) are in different problem spaces. Other problem spaces contain rules governing the vision system, including rules about how to control the camera, rules for selecting software to process the visual data, and rules for the creation and modification of local coordinate systems in the visual data (which call the appropriate IUE functions.) The reformulation method is itself represented as rules in a problem space; the rules call external Mathematica functions to

perform the algebraic analysis. Newly created coordinate systems are placed in new problem spaces.

The basic problem-solving mechanism in SOAR is subgoaling: every time there is choice of two or more rules, SOAR creates a subgoal of deciding which to select, and brings the entire knowledge of the system to bear on solving this subgoal by selecting a problem space and beginning to search. This search can encounter situations in which two or more rules can fire, which in turn causes subgoals to be created, etc. When a rule is successfully chosen, the corresponding subgoal has been solved and the entire solution process is summarized in a single rule, called a *chunk*, which contains the general conditions necessary for that rule to be chosen. This rule is added to the system's rule set, so that in similar future situations the search can be avoided. In this way, SOAR learns.

The SOAR publications extensively document how this learning method speeds up the system's response time in a manner that accurately models the speedup of human subjects on the same tasks. Our research project is investigating the effectiveness of this method at learning how to perceive task environments.

6. Summary

Identification of the local invariants and symmetries of a language of actions yields a decomposition into components associated with independent control dimensions, thereby increasing the tractability of synthesis. The definition of local coordinate systems for semigroups yields a means of identifying these local symmetries. This analysis is expensive, but can be performed on small, simple systems whose properties are guaranteed to hold for an infinite class of tasks. The set of invariants of the decomposition form a complete set of independent features that describe the environment, and are the features the vision system should detect.

References

- [1] Benjamin, D. Paul, (1994), Formulating Patterns in Problem Solving, *Annals of Mathematics and AI*, **10**, pp.1-23.
- [2] Benjamin, D. Paul, (1992). Reformulating Path Planning Problems by Task-preserving Abstraction, *Journal of Robotics and Autonomous Systems*, **9**, pp. 1-9.
- [3] Benjamin, D. Paul, (1992a). Towards an Effective Theory of Reformulation, in *Proceedings of the Workshop on Change of Representation and Problem Reformulation*, Michael R. Lowry (ed.), NASA Ames Research Center Technical Report FIA-92-06, pp.13-27, April, 1992.
- [4] Benjamin, D. Paul, Alec Cameron, Leo Dorst, Madeleine Rosar, and Hsiang-Lung Wu, (1991), Integrating Perception with Problem Solving, *Proceedings of the AAAI Spring Symposium on Integrated Intelligent Architectures*, Stanford University, March, 1991.
- [5] Howie, J. M., (1976). An Introduction to Semigroup Theory, Academic Press.
- [6] Laird, J.E., and Rosenbloom, P.S. (1996) The evolution of the Soar cognitive architecture. In T. Mitchell (ed.) *Mind Matters*.
- [7] Lallement, Gerard (1979). *Semigroups and Combinatorial Applications*, Wiley & Sons.
- [8] Nelson, Edward, (1967). *Tensor Analysis*, Mathematical Notes, Princeton University Press.