

# A Decomposition Approach to Solving Distributed Constraint Satisfaction Problems

D. Paul Benjamin

School of Computer and Information Science, Syracuse University

Center for Science and Technology, Syracuse, NY 13244-4100

benjamin@top.cis.syr.edu

<http://www.cis.syr.edu/people/benjamin>

## ABSTRACT

This paper presents a decomposition method for distributed constraint satisfaction problems that is based on an algebraic analysis of the semigroup structure of a domain theory. This analysis identifies the local symmetries of the structure of the theory, and each local symmetry corresponds to a subproblem. To each symmetry corresponds a local invariant, which characterizes the subproblem. The structure of the semigroup of symmetries is used to compose solutions of the subproblems. This leads to a hierarchical decomposition of the original problem with minimal coordination cost. This method is currently being applied to a variety of tasks, including transportation scheduling, job-shop scheduling, and real-time planning in robotics.

## 1. INTRODUCTION

Distributed constraint satisfaction is an approach to solving large, complex problems in a tractable way by distributing the variables of a constraint satisfaction problem among a number of separate problem-solving agents. A number of interesting applications, including planning and scheduling problems in various domains, have been formulated as constraint satisfaction problems [8]. Unfortunately, it is often not clear how to decompose such problems appropriately; a bad distribution of the variables can lead to very poor performance, due to the additional cost of the agents' coordination. Minimization of the coordination cost is essential for efficient distributed problem solving. We present a decomposition method for distributed constraint satisfaction problems that is based on an algebraic analysis of the semigroup structure of a domain theory. This analysis identifies the local symmetries of the structure of the theory, and each local symmetry corresponds to a subproblem. To each symmetry corresponds a local invariant, which characterizes the subproblem. Each problem-solving agent can independently find a solution to its subproblem that does not violate the invariants of intersecting subproblems. The structure of the semigroup of symmetries is used to compose solutions of the subproblems. This leads to a hierarchical decomposition of the original problem with minimal

coordination cost. This method is currently being applied to a variety of tasks, including transportation scheduling, job-shop scheduling, and real-time planning in robotics.

We begin with a brief discussion of local properties. The subsequent section discusses the appropriate mathematics for analyzing these properties. The last two sections show how to compose state spaces so that local properties scale up, and present an example.

## 2. LOCAL PROPERTIES OF STATE SPACES

The important properties and insights into system behavior are to be found in the state space of the system, not the physical space. These two spaces have very different properties, especially in that points that are close in one space might be far apart in the other, e.g., in chess, moving one pawn one square leads to a new position that is close in physical description, but may be completely different in its properties, especially in terms of what moves can be made. As in physics, classification, analysis and synthesis are better performed in state space.

Resource limitations force a problem-solving agent to reason in terms of local properties and plans. Resource limitations within the agent's structure, e.g., size of memory or speed of memory retrieval, as well as limitations inherent in the universe, e.g., the inability to see the entire physical space at once, force an agent to limit its immediate attention to a small part of the system at any instant. Consideration of resource limitations leads us to conclude that the predicates we need to use to reason about the world often hold only locally in state space, not globally. A property may hold locally at each point in the space, in which case we say that it holds *locally everywhere*. A property that holds locally everywhere does not necessarily hold globally. Global truth is a much stronger situation. There are many cases of essential properties that hold locally everywhere, e.g., a space may be locally compact but not compact, or locally convex but not convex. A

representation may be locally accurate everywhere, but not globally accurate.

So we need to explore how the local properties of syntactic forms defined on a space affect analysis and synthesis in that space.

### 3. THE SEMIGROUP STRUCTURE OF SYSTEMS

Consideration of local properties and local representations of systems requires an appropriate set of tools for analyzing the local structure of languages of actions. Such a set of tools can be found in algebraic linguistics. In particular, we will examine systems as semigroups. However, there is an important difference between the goal of our use of these tools and the goals of linguistics. Here we are concerned with problem solving, so we will employ the mathematical tools to analyze how a single agent can represent a system and synthesize a control for it, rather than analyzing how an agent can communicate knowledge.

We consider a task theory  $M = (Q, A, \delta)$  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. 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, so we are concerned with the algebraic structure of  $A$ . A task is specified by a pair (i.g), where  $i: 1 \rightarrow Q$  maps a one-element set into  $Q$  identifying an initial state, and  $g: 1 \rightarrow Q$  identifies a desired state. Without loss of generality, we can restrict our attention to semigroups of partial 1-1 functions on the state space  $Q$  [6]. This formalism is extremely general. It encompasses nondeterministic systems and concurrent systems.

The description of a system for the synthesis of a plan (or control) for  $M$  differs from the description of  $M$  in at least one essential way: the process of 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. As this paper is primarily concerned with system synthesis, we will analyze only inverse semigroups in the remainder of this paper.

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 for any semigroup  $S$ :

$$aRb \text{ iff } aS^1 = bS^1 \quad aLb \text{ iff } S^1a = S^1b$$

$$aJb \text{ iff } S^1aS^1 = S^1bS^1 \quad D = RL \quad H = R \cap L$$

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. This allows us to define a basic local neighborhood of a semigroup:

*Definition.* Given a transformation semigroup  $S = (Q, A)$  and a point  $q$  in  $D$ , a  $D$ -class of  $S$  containing an action whose domain includes  $q$  is called a *basic local neighborhood* of  $q$ .

Each basic local neighborhood in state space consists of behaviorally similar states. If the agent's perceptual capabilities are not sufficiently precise to distinguish different neighborhoods, then it cannot plan and move effectively in the space. A state  $q$  may be in more than one basic local neighborhood. Clearly, every element of  $Q$  is in at least one basic local neighborhood. The set of all basic local neighborhoods forms a neighborhood base for a topology on  $Q$ ; however, this will not be pursued in this paper, so we will just refer to basic local neighborhoods as neighborhoods.

Utilizing Green's relations, Lallement defines a local coordinate system for a neighborhood:

*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  $q_{\lambda\rho}, q^{-1}_{\lambda\rho}, r_{\lambda\rho}, r^{-1}_{\lambda\rho} \in S^1$  with  $\lambda \in \Lambda, \rho \in P$  such that the mappings  $x \rightarrow q_{\lambda\rho}xr_{\lambda\rho}$  and

$y \rightarrow r^{-1}\lambda\rho y q^{-1}\lambda\rho$  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  $[H_0; \{(q\lambda\rho, q^{-1}\lambda\rho, r\lambda\rho, r^{-1}\lambda\rho): \lambda \in \Lambda, \rho \in P\}]$ .

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 within a local neighborhood by performing a similarity transformation (inner automorphism) in the usual way (the reader is referred to [7] for details).

Each local coordinate system within the semigroup expresses a distinct syntactic labeling of a subsystem, as we can choose a point in the neighborhood to be the ‘‘origin’’, and label points in the neighborhood according to the actions that map the origin to them.

#### 4. SYSTEM COMPOSITION AND DECOMPOSITION: HOW PROPERTIES SCALE UP

Analysis and synthesis in small systems or in small localities of large systems is relatively well understood and tractable, e.g., by the techniques of control theory; however, it is often the case that global analysis or synthesis are required. This means that we need to know how the properties of a small part of a large system are related to the global properties of the system. In other words, we need to know how the properties scale up.

Properties that hold locally everywhere are not obtained from global properties by specialization, and global properties are not obtained from those that hold locally everywhere by generalization. For example, a space that is locally Euclidean need not be globally Euclidean. Instead, global properties are obtained from properties that hold locally everywhere by *composition*. Properties that hold locally everywhere are obtained from global properties by *decomposition* of the global description. Decomposition is the process of transforming a description of a system into a composition of descriptions of smaller systems, so as to express global properties as arising from the interaction of local properties. A good decomposition makes both analysis and synthesis much cheaper, because they can be performed on small subsystems and their results composed to apply to the whole system.

Numerous methods of composition have been proposed. Specific forms of composition work well in a

number of individual areas, e.g., composition of program modules, distributed processes, or program specifications. Each such form of composition specifies how certain properties of a particular class of systems scale up. But the quest for a general composition operator has not been so successful. The most widely used general composition operator for systems is the wreath product.

The wreath product is a construction essentially the same as the cascade of machines (see [7] for the definition of wreath product, and [11] for the definition of cascade) and has been used extensively in previous attempts to automate hierarchical synthesis by finding a wreath product decomposition of a given system e.g., [12], [5]. This technique decomposes a semigroup into the wreath product of a set of groups and a small set of ‘‘reset’’ semigroups. We will not examine the definition of wreath product in detail because we will not use it.

Unfortunately, application of wreath product decompositions to general semigroups can be very expensive computationally. Even the wreath product decomposition of small semigroups may require a large number of groups. The minimal number of groups required for a wreath product decomposition of a given semigroup is called the *group complexity* of that semigroup. It is not known whether the group complexity is decidable. This makes it very difficult to design good algorithms for finding such decompositions.

Also, this form of decomposition is not sufficiently precise for synthesis. For a given transformation semigroup, we obtain a wreath product that properly *contains* the semigroup. In practice, the wreath product is usually far larger than the original semigroup. It cannot be guaranteed (and practice shows that it is rarely the case) that paths synthesized in the component semigroups of the wreath product decomposition can be composed to form a valid path in the original semigroup.

For these reasons, we avoid the wreath product, and instead turn again to our intuition that the mathematics used by physicists to formulate theories of systems in the physical universe is appropriate for use by other problem solving agents to formulate such theories.

The physical model that we choose to emulate is the manifold, which is defined by a number of conditions, most of which we will not recite here as only the

following is of immediate relevance:<sup>1</sup>

For every neighborhood  $N_i$  of the manifold there exists a homeomorphism  $f_i$  into the unit sphere of an  $n$ -dimensional Euclidean space  $E^n$ , for some  $n$ . If  $N_1$  and  $N_2$  are two distinct neighborhoods with a non-empty intersection, then  $f_1(f_2^{-1})$  is a homeomorphism between  $f_2(N_2)$  and  $f_1(N_1)$ .

We emulate this condition to construct a composition operator in the following way.

Definition. Given two semigroups  $S_1$  and  $S_2$ , neighborhoods  $N_1$  and  $N_2$ , and an isomorphism  $\phi$ :

$N_1 \rightarrow N_2$ , we define the composition  $S_1 \mathbf{am}_\phi S_2$  to be the amalgam of  $S_1$  and  $S_2$  with core  $N_1$  ([13], p.61). In other words,  $S_1 \mathbf{am}_\phi S_2$  consists of the set

$\langle \mathbf{Math?} \rangle$  with semigroup multiplication defined by:

$xy$  if  $x, y \in S_1$  or  $x, y \in S_2 - N_2$  (the product is that of the appropriate semigroup),

$xf_1(y)$  if  $x \in S_2 - N_2$  and  $y \in N_1$  (the product is that of  $S_2$ ),

$f_2(x)y$  if  $y \in S_2 - N_2$  and  $x \in N_1$  (the product is that of  $S_2$ ),

0 otherwise.

The class of inverse semigroups has the strong amalgamation property, which means that the application of this composition operation to two inverse semigroups can be embedded in an inverse semigroup in such a way that the images of  $S_1$  and  $S_2$  under the embedding intersect only in the image of the core. For details see [13] Chapter XIII. To simplify the presentation, we ignore the embedding and simply treat  $S_1 \mathbf{am}_\phi S_2$  as an inverse semigroup.

This definition expresses the idea that composed systems are connected through a shared subsystem. This idea is fundamentally different from the usual idea that systems are composed by communication lines connected to I/O ports. The structure to be amalgamated is the group of symmetries (automorphisms) of the conjugate hull of the semigroup. The details of this construction have been partly described in [3] and [4].

### 5.EXAMPLE: THE N QUEENS

The  $n$ -Queens task is to place  $n$  queens on an  $n \times n$  chessboard so that no two queens attack each other. We are posed the task of enumerating all solutions for a given value of  $n$ .

A direct way to construct a theory of this domain that specifies the legal solutions is to specify a solution as satisfying a predicate that is the conjunction of the goal conditions: no-two-queens-on-same-row & no-two-queens-on-same-column & no-two-queens-on-same-up-diagonal & no-two-queens-on-same-down-diagonal. In [10], it is shown how this domain theory can be combined with a theory of global search to derive a correct high-level program. However, even with efficient heuristics, the search space grows exponentially with the size of the problem. For example, in the 4-queens problem, there is no solution if we place the first queen in the upper left-hand corner or lower-left hand corner, but the system cannot know this until it has searched all ways of placing the remaining queens. This exponential growth in the number of possibilities is the well-known Achilles' heel of problem solving, as it precludes the efficient solution of very large problems. Global search cannot be used to enumerate the solutions to the million-queens problem.

Effective search heuristics are known for finding a single solution to a problem of that magnitude, but they cannot efficiently enumerate all solutions. This prevents consideration of the space of solutions, and in particular prevents optimal problem solving. More seriously, the heuristic approach requires engineering a new heuristic for each problem class. This does not even partially solve the designer's problem of finding an efficient algorithm; it merely transforms it into the problem of finding an efficient heuristic.

What is needed is a method for finding good decompositions of the problem. A good decomposition of the problem has at least the following two properties: it splits the problem into components such that solutions within the components compose to form solutions to the whole problem, and it applies to all problems in the class.

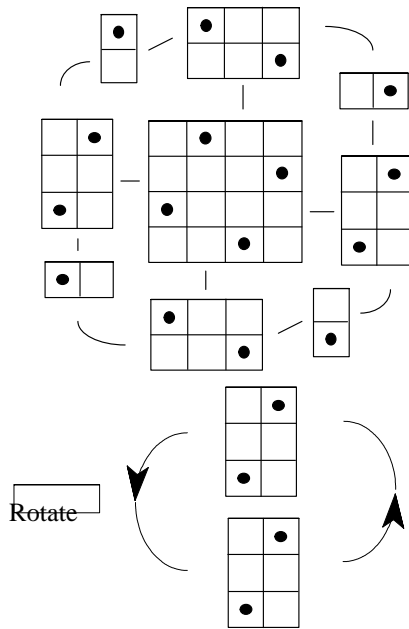
The two forms of decomposition usually implemented are first-rest decomposition and half-half decomposition. In first-rest decomposition, the first part of the problem description is solved and its solution is

<sup>1</sup>In general a transformation semigroup will not satisfy all the conditions for a manifold using the given definition of neighborhood. In particular, using  $D$ -classes as neighborhoods, for each pair of points there is not necessarily an open set containing one point and not the other.

combined with the solution of the rest (this constituting a recursive call to first-rest decomposition.) In half-half decomposition, the problem description is split roughly in half, and solutions of each half are combined (again leading to recursive calls.) Mergesort is a typical example of half-half decomposition, while bubble sort is a first-rest decomposition.

These forms of decomposition are not generally applicable. Neither of these forms of decomposition applies to the n-queens problem. In first-rest decomposition, we would place one queen, then place the remaining n-1 queens. But as we saw above, if we place the first queen in the wrong place (a corner), the 1-queen solution will not compose with any solution to the n-1 queens to give a valid n-queen solution. Similarly, half-half decomposition would find any solution for placing 2 queens, and compose it with another 2-queen solution (which could be a copy of itself.) This need not be a solution to the 4-queen problem.

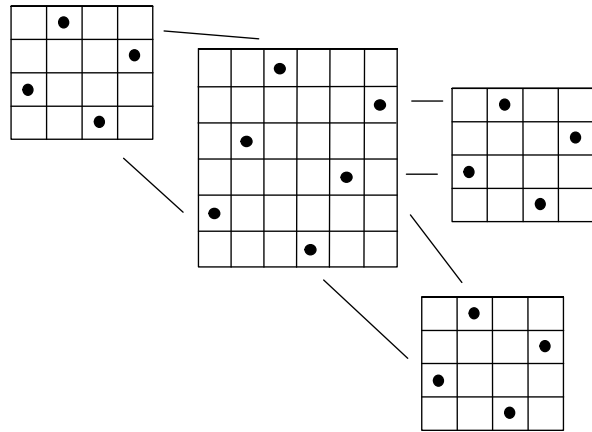
The next figure shows the hierarchy of local symmetries for the solution to the 4-queens problem.



The top-level local symmetry is that of the 3x4 subproblems, but the four 3x4 subproblems in the 4x4 are mapped to each other only by the cyclic global symmetry of order 4. The first new local symmetries arise from considering the 2x3 subproblem that contains 2 queens. The four instances of this subproblem are of course also mapped to each other by the cyclic global symmetry, but also map to themselves

by a local symmetry of order 2. The next figure shows these four instances, together with the 1x2 subproblems that arise from intersecting the 2x3 subproblems:

We construct larger instances of the n-queens problem from copies of the 4x4 solution, amalgamating the local symmetries. The 6-queens solution can be constructed from four copies of the 4-queens solution by amalgamating 2x3 subproblems:



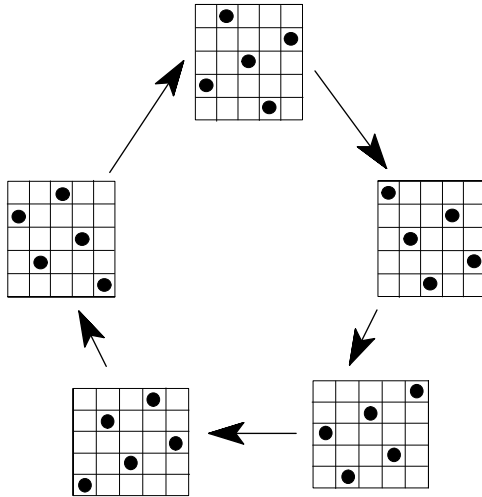
The 6-queens solution can be viewed as three 4-queens solutions in this way.

Stated another way, the construction of the 6-queens solution from multiple 4-queens solutions can be thought of as follows: take 2 copies of the 4-queens solution and amalgamate a 2x3 subproblem in one copy with a 2x3 in the other. There are 6 distinct queens. The four queens in the first copy are guaranteed to be on their own rows, columns, and diagonals, and the four queens in the second copy are similarly guaranteed to satisfy the solution conditions. But we are not guaranteed that the two non-amalgamated queens in one copy are on distinct rows, columns, or diagonals from the two non-amalgamated queens in the other copy. It is necessary that those four queens also satisfy the solution conditions, i.e., we need them to be a 4-queens solution by themselves. Therefore, composing three 4-queens solutions in this way is both necessary and sufficient to give a 6-queens solution, which implies that all 6-queens solutions are constructed in this way.

Space limitations prevent a complete presentation of the structure of this task, but we will look at the symmetries of the 5-queens solutions, and see how they appear in larger solutions.

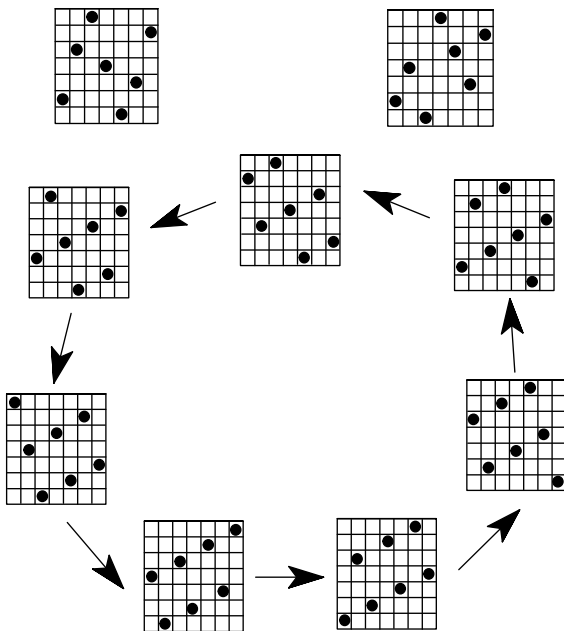
Five 5-queens solutions appear below. They have an obvious four-fold symmetry, which is obtained by

rotating 90 degrees. This maps the lower four solutions cyclically, and maps the upper solution to itself.



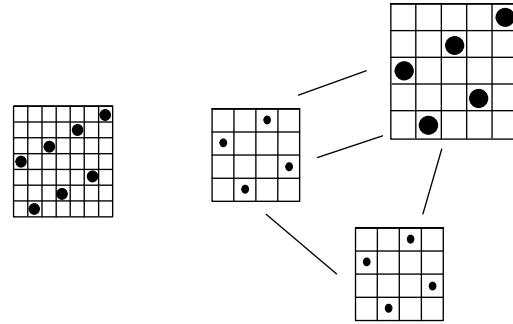
There is also the flip symmetry, which gives the other five solutions.

It is not as obvious that there is a five-fold symmetry of these solutions. Cut the left column off a solution and paste it onto the right side, and the solutions are mapped as shown by the arrows in the diagram. Thus, the symmetry group of the 5-queens solutions is the product of cyclic groups of order 2, 4, and 5, and there is really only one basic 5-queen solution pattern, from which we obtain all the other solutions.



These two solution patterns combine to give the seven-queens solutions. For example, the 7-queen

pattern in the next diagram is composed of a 5-queen and two 4-queens as shown at right.



The next diagram shows the basic patterns of all 7-queen solutions. These three distinct patterns (the two single patterns and the seven-fold symmetric pattern) give the three ways of combining a single copy of a 5-queen solution with 4-queen solutions. Thus, we can construct n-queens solutions from smaller solutions. It is not necessary to search a space containing non-solutions. Instead, analysis of the structure of the base case reveals both the decompositions of the base case and the inductive definition of the class of problems with the same structure. Ideally, a system for software design should have the capability to perform this analysis, to synthesize programs that construct the solution space directly in this fashion whenever possible.

## 6.SUMMARY

Identification of the local invariants and symmetries of a constraint satisfaction task yields the identification of a set of implementation-independent invariants that characterize the task, which can be used to derive a hierarchical decomposition of the task into independent components that can be solved by separate problem-solving agents.

## 7.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] Formulating Systems, *Proceedings of the 28th IEEE Southeastern Symposium on System Theory*, IEEE Computer Society Press, April, 1996.

- [4] Formulating Patterns in Problem Solving, *Annals of Mathematics and AI*, **10**, pp.1-23, 1994.
- [5] Topolskiy, N.G., Description and Machine Decompositional Synthesis of Digital Automata with a Large Number of States, *Tekh. Kibern. (USSR)* Vol. 17, No. 3, pp.136-47, 1979, Translated in: *Eng. Cybern. (USA)* Vol. 17, No.3, pp 95-105.
- [6] Howie, J. M., (1976). An Introduction to Semigroup Theory, Academic Press.
- [7] Lallement, Gerard (1979). Semigroups and Combinatorial Applications, Wiley & Sons.
- [8] Mackworth, A. K. (1987). Constraint Satisfaction, in Shapiro, S.C., ed., *Encyclopedia of Artificial Intelligence*. Wiley, pp. 205-211.
- [9] Wigner, P. (1967) Symmetries and Reflections, Indiana University Press.
- [10] Smith, Douglas R., (1990). KIDS: A Semi-Automatic Program Development System, *IEEE Transactions on Software Engineering*, Vol. 16, No. 9, Special Issue on Formal Methods, pp.1024-1043, September, 1990.
- [11] Bobrow, Leonard S., and Arbib, Michael A., (1974). Discrete Mathematics, Saunders, 1974.
- [12] Halatsis, C., Sigala, M., and Philokyprou, G., Polylinear Decomposition of Synchronous Sequential Machines, *IEEE Transactions on Computers*, Vol. C-27, No. 12, pp.1144-52, 1978.
- [13] Petrich, Mario, (1984). Inverse Semigroups, John Wiley & Sons, Inc., New York.
- [14] Stoker, J. J., (1969). Differential Geometry, Wiley-Interscience.