# Inductive versus Recursive Partitioning of Progress Graphs

*David Cape* dacvdc@umr.edu
*Bruce McMillin* ff@umr.edu
*Benjamin Passer* bwp279@umr.edu
*Mayur Thakur* thakurk@umr.edu

July 16, 2007

**Abstract**

In this paper we consider two-dimensional progress graphs and compare the complexity of the reduced state transition systems obtained by two different partitioning heuristics based on dihomotopy. We show experimentally that a recursive reduction algorithm could improve the results of Goubault et al.

*Keywords*: model-checking, LTL, dihomotopy, deadlock, trace

# 1 Introduction

A problem for model checking is the explosion of the number of states as the complexity of the concurrent program being analyzed grows. Non-determinism in the interleaving of events is to blame, but topological methods may provide some hope for mitigating the problem, especially for deadlock detection. Dijkstra is credited with the idea of a progress graph with axes corresponding to the processes, and in his dissertation and other subsequent work [GR02, GH05], Goubault and others have advanced the concept of using "higher-dimensional automata" and "dihomotopy" to simplify the analysis of progress graphs by decomposing them into components. The larger the components, the fewer components there will be and the greater the reduction of the state-space explosion.

The methods of Goubault and others may not be optimally efficient in this regard, though, and in this paper, we propose a recursive approach instead of an inductive one to further reduce the state-space explosion. We have found experimentally that the inductive method produces a number of regions (components) which is approximately quadratic in the number of semaphores for the progress graphs being considered, but the recursive approach lowers the exponent to approximately one. If these algorithms can be exploited for general use in model checking LTL formulae over traces [BL01], then we expect a significant benefit to model checking methods from this improvement, because there are exponentially many paths to be checked.

A heuristic which often works in research is that if a problem seems too difficult, then perhaps a slightly easier similar problem should be considered. This was the strategy for this research. First, we considered the problem of forbidden points in a modified progress graph and discovered a recursive algorithm: then we modified the algorithm to handle forbidden rectangles. We believe that it will generalize to higher-dimensional progress graphs having more than two processes and many forbidden cuboids.

# 2 Preliminaries

## 2.1 Two Dimensions

**Definition 2.1.** *Progress Graph (space) with Forbidden Points. A progress graph with forbidden points is a product of two closed intervals of real numbers (one for each process of a concurrent program) minus isolated forbidden points in the interior of the graph (space).*

**Definition 2.2.** *Progress Graph (space) with Forbidden Rectangles. A progress graph with forbidden rectangles is a product of two closed intervals of real numbers (one for each process of a concurrent program) minus the interior of forbidden rectangles which have their interior and boundary in the interior of the graph and have edges parallel to the coordinate axes. See Figure 1.*

Note: in the following text, the term progress graph refers to either of the above two types except when one or the other is clearly specifically intended.

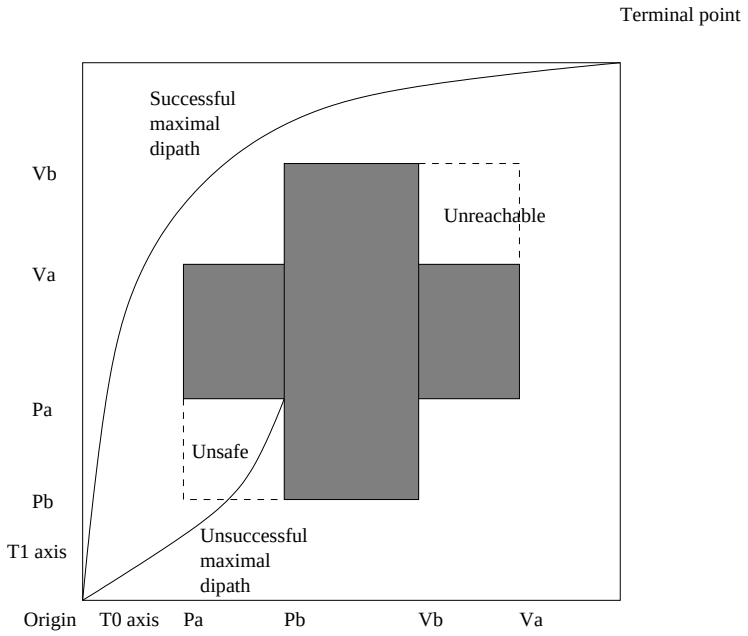


Figure 1: This is the standard two-semaphore Swiss flag example. Two processes $T_0 = \{Pa; Pb; Vb; Va; \}$ and $T_1 = \{Pb; Pa; Va; Vb; \}$ comprise a concurrent program. The horizontal rectangle corresponds to $a$ and the vertical rectangle corresponds to $b$: they are forbidden.

**Definition 2.3.** *Dipath. A dipath in a progress graph is a continuous function from the interval [0,1] into the progress graph which has the property of being non-decreasing in each coordinate. (We will mostly be interested in dipaths which map 0 to the minimal point (lower left corner) of the progress graph.)*

**Definition 2.4.** *Maximal Dipath. A maximal dipath in a progress graph is a dipath which cannot be extended backwards or forward in the progress graph.*

**Definition 2.5.** *Successful Dipath. A successful dipath in a progress graph is a dipath which ends at the upper right corner of the progress graph.*

**Definition 2.6.** *Valid Dipath. A valid dipath in a progress graph is a dipath which begins at a reachable point, meaning that there exists a dipath that begins at the lower left corner of the progress graph and ends at that point.*

**Lemma 2.7.** *In a progress graph with forbidden points, every dipath is a valid dipath.*

**Proof of Lemma 2.7** This lemma is equivalent to the proposition that all points in a progress graph with forbidden points are reachable. Because the forbidden points must be isolated, there is a finite number of them. Because it is also clear that the boundaries are reachable, as forbidden points must only be in the

interior of the progress graph, consider an interior point $(x, y)$ of the progress graph. Let $V$ be the set of forbidden points of the form $(x_i, y_i)$ where $x_i < x$. If $V$ is empty, there are no forbidden points strictly to the left of $(x, y)$, so a direct dipath from the lower left corner to $(x, y)$ exists. If $V$ is non-empty, let $r$ be the minimum value of $x - x_i$.This guarantees that there are no forbidden points with a horizontal coordinate in the interval $(x - r, x)$. Thus, a dipath that starts at the lower left corner (assumed to be the origin for simplicity), travels horizontally until $(x - r/2, 0)$, travels vertically until $(x - r/2, y)$, and then travels horizontally until $(x, y)$ is a dipath from the lower left corner to $(x, y)$. □

**Lemma 2.8.** *In a progress graph with forbidden points, every maximal dipath is a successful dipath which begins at the lower left corner of the progress graph.*

**Proof of Lemma 2.8** The endpoints of any dipath in a progress graph with forbidden points are a nonzero distance away from all forbidden points, as the endpoints are not allowed to be forbidden points. Consider the closest forbidden point $(x, y)$ to $p(1)$ for an unsuccessful dipath $p$. Because $p(1)$ is a positive distance $r_1$ from $(x, y)$ and a positive distance $r_2$ from the upper right corner, any extension of $p$ of length half of the minimum of $r_1$ and $r_2$ in the direction of the upper right corner will not intersect any forbidden point, so $p$ was not maximal. Similarly, dipaths can be extended backwards toward the lower left corner of the progress graph. □

**Definition 2.9.** *Dihomotopy/Dihomotopy Equivalence. A dihomotopy between dipaths $f$ and $g$ is a continuous function from the the unit square to the progress graph which restricts to $f$ on the bottom edge, restricts to $g$ on the top edge, and takes the left and right edges to the initial and final points (respectively) of $f$ and $g$, which must coincide at their endpoints. It is also required that each horizontal cross-section between $f$ and $g$ be a dipath. If a dihomotopy exists between $f$ and $g$, then they are said to be dihomotopically equivalent.*

**Definition 2.10.** *Type. Let $p$ be a maximal dipath in a progress graph, and let $S$ be an ordered set of forbidden points of the progress graph. Then the type of $p$ relative to $S$ describes how the dipath "passes by" the points of $S$. For example, if the dipath goes to the left and then above a particular point of $S$, then for that point, the type data is $(1, 0)$. The 1 in the first coordinate signifies that it went above (not below) the point and the 0 in the second coordinate signifies that it went to the left (not to the right).*

Note that the only possible type data for a point of S is $(0, 1)$ or $(1, 0)$ because of the nondecreasing properties of the dipath. This definition may seem redundant, therefore, but it has been written for ease of use when generalized to codimension two forbidden sets in higher dimensional progress graphs.

**Lemma 2.11.** *Let $p$ be a maximal dipath in a progress graph with forbidden points, and let $(x, y)$ be a forbidden point of the progress graph. Then $p$ goes above $(x, y)$ iff it goes to the left of $(x, y)$.*

**Proof of Lemma 2.11** Because $p$ is a maximal dipath in a progress graph with forbidden points, by Lemma 2.8, $p$ is a successful dipath which begins at the lower left corner (assumed to be the origin for simplicity). If $p$ goes above a forbidden point $(x, y)$, then $p$ goes through a point $(x, y_0)$ with $y_0 > y$. Let $t_0 \in (0, 1)$ be such that $p(t_0) = (x, y_0)$. Because $p$ is a dipath, it is continuous, so its projection $\pi_2(p(t))$ is also continuous. Because $\pi_2(p(0)) = 0$ and $\pi_2(p(t_0)) = y_0$, by the Intermediate Value Therorem there exists a $t_1 \in (0, t_0)$ such that $\pi_2(p(t_1)) = y$. That is, there is a $t_1 \in (0, t_0)$ such that $p(t_1) = (x_1, y)$ for some $x_1$. Now, $p$ is a dipath, so $x_1 \leq x$ because $t_1 < t_0$. However, $x_1 \neq x$ because $(x, y)$ is a forbidden point, meaning that $p$ goes to the left of $(x, y)$. The proof of the converse is trivial. □

**Definition 2.12.** *Admissible Type. A type is admissible if there is a dipath in a progress graph which has the given type. Otherwise, it is called inadmissible.*

**Definition 2.13.** *Transition Digraph. We define a transition digraph to be a digraph obtained from a progress graph by partitioning it into regions (nodes) and boundaries between them (directed edges).*

**Definition 2.14.** *Path in a Transition Digraph. A path in a transition digraph is a path in the usual sense of digraph. Maximality and successfulness are defined as in the case of dipaths.*

**Definition 2.15.** *Maximal Region in a partitioned progress graph. A maximal region in a partitioned progress graph is a component/region of the partition which corresponds to a maximal node in the associated transition digraph, meaning that there are no directed edges coming out of that node. This can represent an unsafe region.*

## 2.2 Three and Higher Dimensions

All of the concepts of the previous subsection can be generalized to higher dimensions. Forbidden points are replaced by forbidden sets of codimension two which extend from end to end in all free directions. That is, only two coordinates are specified as forbidden. The analogue of forbidden rectangles is the product of a rectangle in two dimensions with the entire intervals in all other directions. The only interesting feature of the generalization is that the forbidden sets can "extend in different directions (with respect to each other)".

Dihomotopy, however, remains a two-dimensional property, for it captures the concept of a continuous deformation via a one-parameter family of dipaths. Induction is the key to generalizing the proofs. For example, see the next proof.

**Theorem 2.16.** *In a progress graph with codimension two forbidden sets, every dipath is valid; equivalently, every point is reachable.*

**Proof of Theorem 2.16** □

The definition of type is modified to include "don't care" symbols (*) in the coordinates for which the forbidden codimension two sets extend from end to end of the progress graph. For example, a dipath $p$ in a three-dimensional progress graph with only one forbidden line segment specified by (0.5, *, 0.25) could have a type described by (0,*,1) if $p$ crosses the plane $x = 0.5$ before it crosses the plane $z = 0.25$, or described by (1,*,0) otherwise.

# 3 Progress Graphs

## 3.1 Forbidden Points

### 3.1.1 Descriptions

*Inductive Reduction Algorithm.* We describe an adaptation of an algorithm developed by Goubault et al. Given a progress graph with forbidden points, partition the progress graph by drawing horizontal and vertical line segments from each forbidden point to the boundaries of the progress graph. See Figure 2. This divides the progress graph into rectangles, which are called components. We form a transition digraph which has these components as nodes/vertices, and the boundaries (line segments) between them as directed edges (where crossing a boundary represents a transition). Certain squares in the digraph commute in the sense that traversing one or the other path from the initial point of the square to the terminal point is not distinguishable from the point of view of dihomotopy. Such commutative squares have been called "relations" in the literature.

*Recursive Reduction Algorithm.* First, let $V$ be the set of forbidden points. Next, let $B_0$ be the entire progress graph, and let $B_1$ be the smallest rectangle inside $B_0$ which contains all of the points of $V_0 = V$ and contains the upper right corner of the progress graph. The left and bottom sides of $B_1$ will contain at least one point of $V_0$, either in the interior or at the corner. Let $V_0'$ be the set of points of $V_0$ which are on the left or bottom sides of $B_1$, and let $V_1 = V_0 - V_0'$. Repeat this procedure to define $B_l$, $V_l$, and $V_l'$ for each $l \geq 0$ until $V$ is exhausted. Let the closed sets $N_0 = cl(B_0 - B_1), N_1 = cl(B_1 - B_2), \ldots$, and call $\{N_0, N_1, \ldots\}$ the initial partition of the progress graph.

Next, we will partition each $N_l$ according to the positions of the points of $V'_{(l-1)}$ and $V_l'$. For example, see Figure 2, which represents a two-dimensional case. The basic idea is that we draw L-shaped regions and then partition them with line segments.

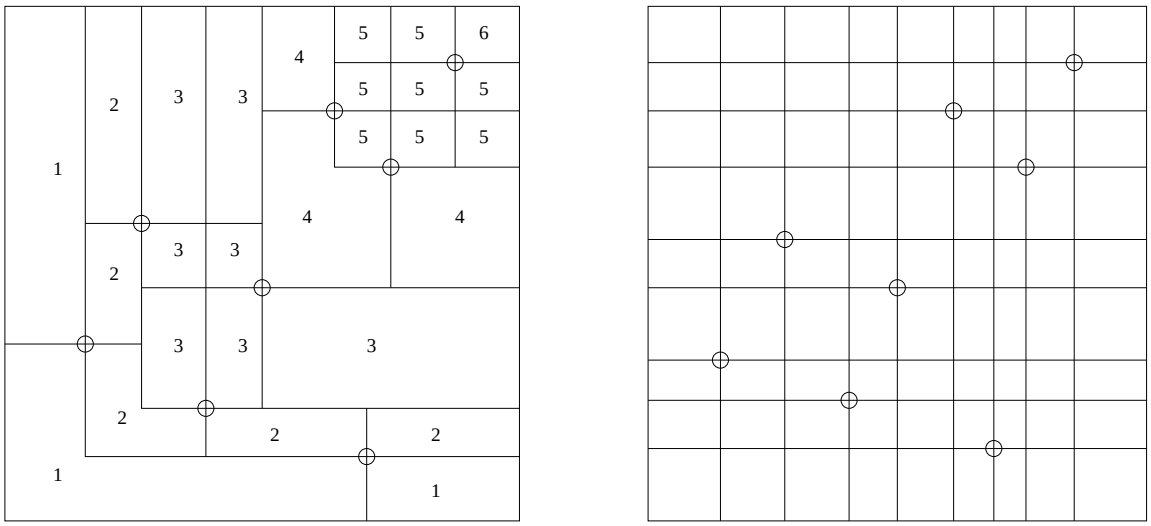


Figure 2: Small circles represent forbidden points. Numbers correspond to L-shaped regions in the initial partition. Notice the recursive partitioning in the progress graph on the left.

Since each point of $V_l'$ is on at least one face of $B_{l+1}$, one can define the direction(s) normal to the faces that contain a given point of $V_l'$. For each such direction, extend line segments to the left or down (outward) from the inner boundary of $N_l$ to the outer boundary of $N_l$ (remember that $N_l$ is the complement of a L-shaped region). Also extend line segments to the right or up (inward) through $N_l$ from the points of $V'_{(l-1)}$. This procedure also produces a transition digraph.

### 3.1.2 Correctness

*Note: we shall assume that dipaths cross partition boundaries transversally.*

**Theorem 3.1.** *If $f$ and $g$ are maximal dipaths in a progress graph with forbidden points that have the same type relative to the set of forbidden points, then they are dihomotopically equivalent (there is a dihomotopy between $f$ and $g$).*

**Proof of Theorem 3.1** Because $f$ and $g$ are maximal dipaths in a progress graph with forbidden points, by Lemma 2.8, they coincide at their endpoints. Construct a dihomotopy between $f$ and $g$ as follows. Define $h_s(t) = h(s,t) = sf(t) + (1-s)g(t)$. We must show that for any $s \in [0,1]$, $h_s$ is a dipath: that is, that it is nondecreasing in each coordinate and avoids forbidden points. Because $f$ and $g$ are nondecreasing, and for all $s \in [0,1]$, $s \geq 0$ and $1 - s \geq 0$, $h_s$ is also nondecreasing in each coordinate, as it is a nonnegative linear combination of $f$ and $g$. In order to show that $h_s$ avoids forbidden points, consider that $h_0$ and $h_1$ must have the same type relative to the set of forbidden points, as they are simply the functions $g$ and $f$, respectively. Begin by reparameterizing $f$ and $g$ so for all $t \in [0,1]$, $\pi_1(f(t)) = \pi_1(g(t))$. Suppose that there exist some $s, t \in (0,1)$ such that $h_s(t) = (x,y)$, a forbidden point. Now, $h_s$ is simply a weighted average of $h_0$ and $h_1$, so we have $h_0(t) = (x, y_0)$ and $h_1(t) = (x, y_1)$, so either $y_0 \leq y \leq y_1$ or the reverse, which implies that either $f$ and $g$ have different types or at least one of them passes through the forbidden point. By contraposition, $h_s(t)$ avoids forbidden points. □

**Corollary 3.2.** *Contrapositive of above theorem. If $f$ and $g$ are maximal dipaths in a progress graph with forbidden points from different dihomotopy equivalence classes, then they have different types.*

**Theorem 3.3.** *If $p$ is a path in a transition digraph obtained from a progress graph with forbidden points by*

*using the Recursive Reduction Algorithm, and $f$ and $g$ are maximal dipaths that both map to $p$, then $f$ and $g$ have the same type.*

**Proof of Theorem 3.3** Suppose that two dipaths $f$ and $g$ having different types map to the same path $p$. Consider the L-shaped regions. They inherit a natural total order increasing from the lower left to the upper right. Moreover, there is a total order on the regions within each L-shaped region from top edge to right edge. As $f$ passes through the L-shaped regions in order, its type is determined by which regions it passes through. Every forbidden point is at the intersection of two perpendicular line segments, one of which is the boundary between two L-shaped regions. It is impossible for $g$ to map to the same path in the transition digraph as $f$, since the dipaths must "choose" where to cross the boundary: either to the left or right of the forbidden point in question, or above or below it. □

**Definition 3.4.** *Type of a path in a transition digraph (obtained from a progress graph by using the Recursive Reduction Algorithm). If $p$ is a path in a transition digraph obtained from a progress graph by using the Recursive Reduction Algorithm, define the type of $p$ to be the type of any maximal dipath that maps to $p$.*

**Corollary 3.5.** *If $f$ and $g$ are maximal dipaths in a progress graph with forbidden points from different dihomotopy equivalence classes, then they map to paths having different types in the transition digraph obtained from the Recursive Reduction Algorithm.*

There exists an inadmissible maximal path through the transition digraph obtained from a progress graph under the Recursive partitioning. For example, see Figure 3.

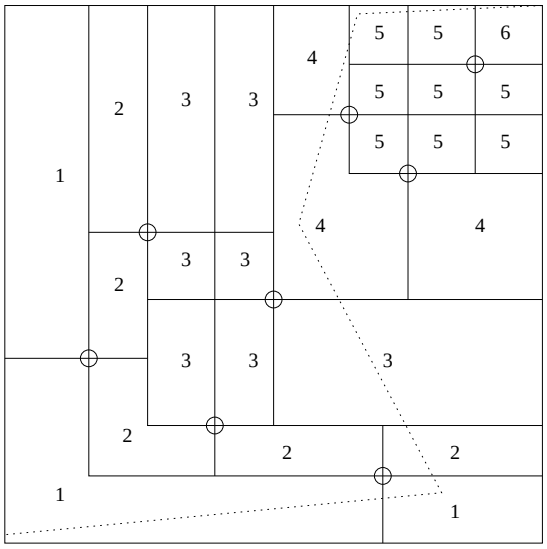


Figure 3: The path from the lower left to upper right corners does not have an admissible type; i.e., it does not come from a dipath.

## 3.2 Forbidden Rectangles

### 3.2.1 Descriptions

*Inductive Reduction Algorithm. [GH05]* There is a correspondence between semaphores and forbidden rectangles with minimal (lower left) corner corresponding to locking and maximal (upper right) corner corresponding to it unlocking by each process of the progress graph. Take the rectangles one at a time, and draw from the minimal point a horizontal ray back to the left boundary of the progress graph, and draw a vertical

ray down to the bottom boundary of the progress graph. Similarly, draw horizontal and vertical rays from the maximal point of the forbidden rectangle to the right and top boundaries of the progress graph. Do so for each rectangle.

An optimization is possible which is described in Goubault and Raussen's paper [GR02], but this is the essential idea of the inductive algorithm. The optimization is that it is not necessary to extend boundary partitions to the boundary of the progress graph when they intersect another forbidden region first.

When there are only two semaphores, one can observe three representative examples illustrated in Figure 4, which show how the inductive algorithm should have approximately quadratic complexity.

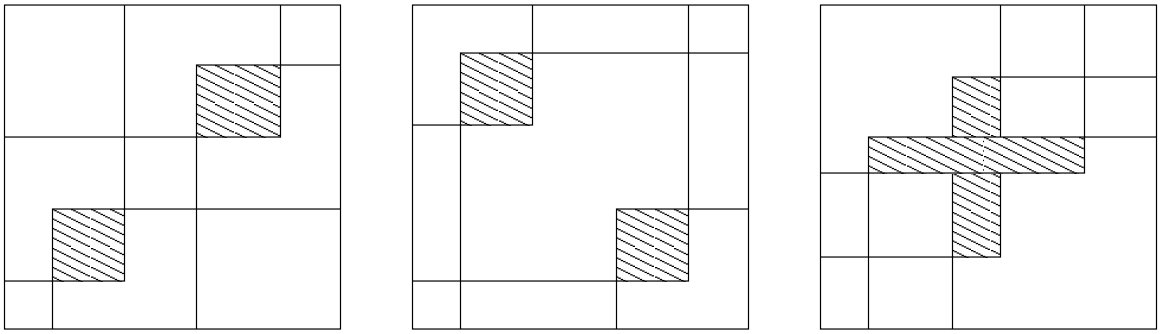

Figure 4: In the first two progress graphs, there are $(2+1)^2 = 9$ regions; in the third, there are 10 regions, but one is unreachable. In general we expect approximately $(m+1)^2$ regions.

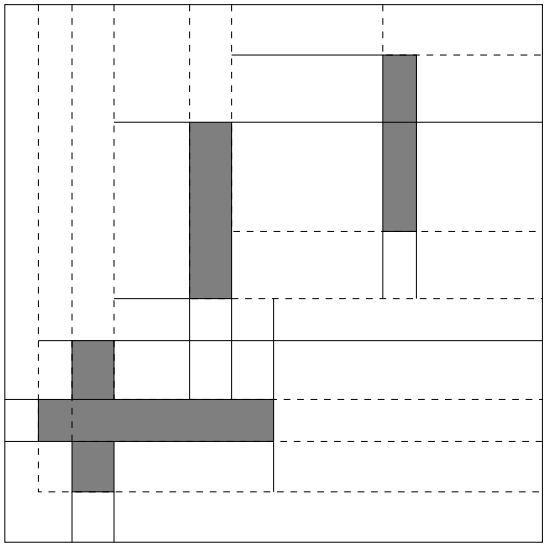

Figure 5: Shaded regions are forbidden. Dashed lines are for the initial partition, and solid line segments complete the partitioning.

*Recursive Reduction Algorithm.* First, collect all of the corners of all of the forbidden rectangles $C_j$ into a set $V = \cup_{j=1}^{m} V_j$. then follow the procedure for forbidden points, except that a slight modification is required because some of the L-shaped regions obtained may be crossed by forbidden rectangles. To accomodate this possibility, simply add new vertices to $V$ wherever a forbidden region crosses the boundary of an L-shaped

region. The basic idea is still that we draw L-shaped regions and then partition them with line segments. See Figures 5 and 6.

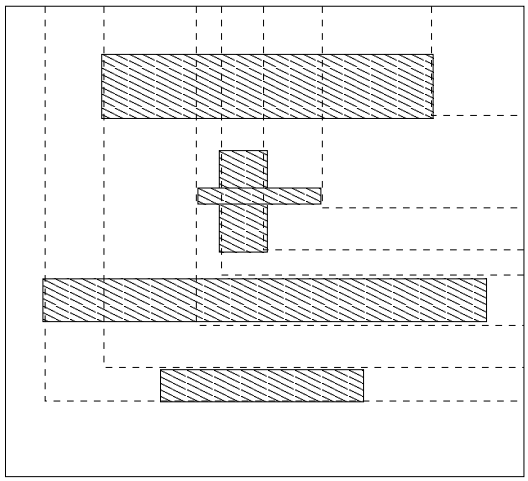

Figure 6: The long horizontal forbidden region at the top crosses through L-shaped regions showing the need to add vertices to protect forbidden regions.

### 3.2.2 Correctness

Correctness arguments are based on those for forbidden points. Along with the properties of the section on progress graphs with forbidden points, we have two additional properties. These arise from consideration of unsafe and unreachable regions. By taking all four corners of each forbidden rectangle to be forbidden as well, we obtain enough information to define types. The modification of adding vertices to $V$ as we form smaller L-shaped regions ensures that these regions are not crossed by forbidden rectangles without a separation of components in the final partition. The set of paths (with admissible types) of the transition digraph obtained does in fact contain representatives of each dihomotopy class of dipaths. See Theorems 3.6 and 3.7 below for the precise definition of correctness.

**Theorem 3.6.** *If $f$ is a maximal dipath in a progress graph, then it maps to a maximal path in the transition digraph obtained by the Recursive Reduction Algorithm.*

**Proof of Theorem 3.6** Suppose that $f$ is a maximal dipath in a a progress graph with forbidden rectangles (the case of forbidden points is clear). If $f$ is valid and succesful, then it clearly maps to a maximal path in the transition digraph, so consider the other two (non-exclusive) possibilities. If $f$ is not successful, but is maximal, then it must reach a point of deadlock $p$. Because $f$ must approach $p$ to an arbitrary proximity, there is only one region $R$ which contains points on $f$ for parameter values that are arbitrarily close to 1, so $R$ must be shown to be maximal. In order for deadlock to occur at $p$, it must be on the boundary of two forbidden rectangles, one above it and one to its right. Let $v_0$ and $v_1$ be the lower left corners of these forbidden rectangles, as illustrated in Figure 7.

Consider the rectangle $C$ formed with $v_0$ as the upper left corner and $v_1$ as the lower right corner (and with $p$ as its upper right corner). We claim that $R$ is contained inside $C$, so it is maximal. Why – because all regions of the partition (including $R$) are either rectangular, in which case it is clear, or L-shaped regions of the partition. But $R$ also contains $p$, so it must be rectangular. The possibility of $f$ being invalid can be handled by similar arguments. □

**Theorem 3.7.** *There is an inverse mapping from maximal paths in a transition digraph obtained from the recursive algorithm that have an admissible type to maximal dipaths in the original progress graph. We do*

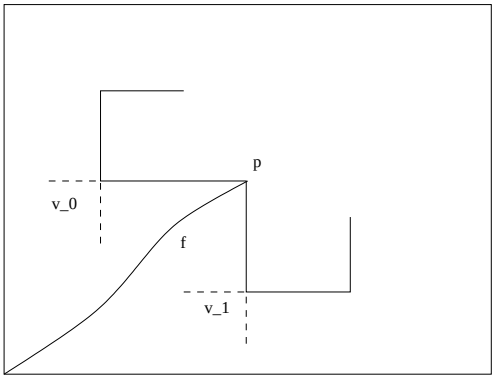


Figure 7: $f$ must go between $v_0$ and $v_1$ to a maximal region. Since every vertex $v_i$ of every forbidden region is a point of intersection of two perpendicular line segments (of the partition) passing through $v_i$, $f$ ends in a maximal region no matter what other partitioning is done by the algorithm.

*not claim that the mapping is unique or one-to-one (but we can obtain a bijection between admissible type classes of paths and dihomotopy classes of dipaths).*

**Proof of Theorem 3.7** Because we are considering admissible types only, the only thing to prove is that the mapping obtained from the definition of admissible can take maximal paths to maximal dipaths. All maximal regions of the recursive partition are rectangular, so any dipath which maps to a maximal path can be extended to its maximal corner, and therefore can be made maximal. □

We use all four corners of each forbidden rectangle for the recursive algorithm, yet it does not affect the complexity significantly. The inductive algorithm only uses the minimal and maximal corners.

## 3.3 Forbidden Codimension Two Sets in $N$ Dimensions

### 3.3.1 Descriptions

*Inductive Reduction Algorithm.* The algorithm of Goubault et al. for forbidden cuboids has been described using "pre-cubical sets" in an earlier paper [GH05]. Its modification to handle simple codimension two forbidden sets as in earlier sections is straightforward.

*Recursive Reduction Algorithm.* We assume that the progress graph extends from $(0, 0, \ldots, 0)$ to $(1, 1, \ldots, 1)$ for simplicity. We will use coordinates $(x_0, x_1, \ldots, x_{N-1})$ corresponding to the processes.

In order to obtain a partition which has the advantages of being basically recursive, but not unduly cumbersome, we consider the $N$ processes in pairs. Since each forbidden set corresponds to a forbidden point in two particular coordinates, we can do the partitioning for two-dimensional planes as before, and then extend them through the $N-2$ orthogonal directions. So, there are $N$ choose 2 recursive partitions to be done and then combined appropriately (meaning that two points are considered to be in the same region of the $N$-dimensional progress graph if their projections are in the same regions in all possible 2-dimensional planes.

### 3.3.2 Correctness

**Theorem 3.8.** *If $f$ and $g$ are maximal dipaths in a progress graph with forbidden codimension two sets that have the same type relative to all of the forbidden sets, then they are dihomotopically equivalent (there is a dihomotopy between $f$ and $g$).*

**Proof of Theorem 3.8** Sketch. Because $f$ and $g$ are maximal dipaths in a progress graph with forbidden codimension two sets, by an analogue of Lemma 2.8, they coincide at their endpoints. Construct a

dihomotopy between $f$ and $g$ as follows.

First, we approximate $f$ and $g$ by dihomotopically equivalent rectangular paths (piecewise linear in the directions of the axes). The final step is a little more difficult: to show that rectangular paths with the same type are also dihomotopically equivalent.

To approximate $f$, let $d$ be the minimum over points of $f$ of the Manhattan/Hamming/Taxi-cab distance from $f$ to any boundary or forbidden point. We will define a rectangular dipath $j$ which is so close to $f$ that it must be dihomotopically equivalent to $f$ via an easy dihomotopy map. Partition [0,1] by $0 = a_0 \leq a_1 \leq \ldots \leq a_m = 1$ such that for all $i$, the distance from $f(a_{i-1})$ to $f(a_i)$ is less than $d$. Next, let $r_i = (a_i - a_{i-1})/N$, and define

$$j(a_{i-1}) = f(a_{i-1}),$$
$$j(a_{i-1} + r_i) = (\pi_1(f(a_i)), \pi_2(f(a_{i-1})), \pi_3(f(a_{i-1})), \ldots, \pi_N(f(a_{i-1}))),$$
$$j(a_{i-1} + 2r_i) = (\pi_1(f(a_i)), \pi_2(f(a_i)), \pi_3(f(a_{i-1})), \ldots, \pi_N(f(a_{i-1}))),$$
$$\ldots,$$
$$j(a_i) = f(a_i).$$

We can connect these points linearly, and then construct the weighted average dihomotopy between $f$ and $j$ as we have done in previous sections.

To construct a dihomotopy between $j$ and the corresponding rectangular dipath for $g$ (which we shall call $k$), let $t_{il}$, respectively $s_{il}$, be the turning points of $j$, respectively $k$, where $i$ ranges from 0 to $m-1$, and $l$ ranges from 0 to $N-1$. The required dihomotopy is fairly easy to construct (see below) if one assumes for all $i$ and all $l$, that $\pi_l(t_{il}) \leq \pi_l(s_{il}$. but if this is not the case then we can simply insert extra intervals where no distance is traveled by $j$ and reindex $t$ and $s$ to make it so.

Assuming that the intervals have been reindexed as desired, we attach rectangles to span the gap between $j$ and $k$. We in effect widen $j$ in the direction of the $0^{th}$ axis toward $k$. We claim that the entire gap between $j$ and $k$ can be spanned by a union of rectangles. An example will help to clarify this.

Let us consider a four-dimensional progress graph with two dipaths and suppose that the turning points of $j$ are $(1/2, 0, 0, 0)$, $(1/2, 1/2, 0, 0)$, $(1/2, 1/2, 1/2, 0)$, $(1/2, 1/2, 1/2, 1/2)$, $(2, 1/2, 1/2, 1/2)$, $(2, 2, 1/2, 1/2)$, and $(2, 2, 2, 1/2)$, and the turning points of $k$ are $(1, 0, 0, 0)$, $(1, 1, 0, 0)$, $(1, 1, 1, 0)$, $(2, 1, 1, 1)$, $(2, 2, 1, 1, )$, and $(2, 2, 2, 1)$. Then the procedure we have described thus far attaches a rectangle from $(1/2, 0, 0, 0)$ to $(1, 1/2, 0, 0)$, a rectangle from $(1/2, 1/2, 0, 0)$ to $(1, 1/2, 1/2, 0)$, and from $(1/2, 1/2, 1/2, 0)$ to $(1, 1/2, 1/2, 1/2)$ first. At this point, we can use the induction.

It should be clear, but it seems cumbersome to write out the piecewise linear dihomotopy which this construction yields,but one can observe due to the geometry that it avoid forbidden points. Why? Because given any two axes, the corresponding forbidden points are entirely outside the projection of the image of the dihomotopy (since the two dipaths have the same type). □

**Corollary 3.9.** *Contrapositive of above theorem. If $f$ and $g$ are maximal dipaths in a progress graph with forbidden codimension two sets from different dihomotopy equivalence classes, then they have different types.*

**Theorem 3.10.** *If $p$ is a path in a transition digraph obtained from a progress graph with forbidden codimension two sets by using the generalized Recursive Reduction Algorithm, and $f$ and $g$ are maximal dipaths that both map to $p$, then $f$ and $g$ have the same type.*

**Proof of Theorem 3.10** Straightforward: this has been proved for dimension two, and the generalized algorithm uses the two-dimensional one. If $f$ and $g$ have different types, then their projections to some two-dimensional space yields a progress graph where they also have different types. Then by Theorem 3.3, they cannot both map to $p$. □

**Definition 3.11.** *Type of a path in a transition digraph (obtained from a progress graph by using the Recursive Reduction Algorithm). If $p$ is a path in a transition digraph obtained from a progress graph by using the Recursive Reduction Algorithm, define the type of $p$ to be the type of any maximal dipath that maps to $p$.*

**Corollary 3.12.** *If $f$ and $g$ are maximal dipaths in a progress graph with forbidden codimension two sets from different dihomotopy equivalence classes, then they map to paths having different types in the transition digraph obtained from the generalized Recursive Reduction Algorithm.*

# 4 Experimental Results

## 4.1 Description

This section describes the results from running an implementation of the Recursive Reduction Algorithm on uniformly distributed random progress graphs and comparing it to the results of an implementation of the Inductive Reduction Algorithm. We did the comparison for progress graphs with forbidden rectangles.

First, we generated a set of $m$ (the number of semaphores) balanced parentheses. Then this was converted to a process file by labeling the parentheses in matching pairs. We did this to create 100 different processes. 30 progress graphs were constructed by randomly taking pairs of processes. We attempted at first to satisfy the condition that there be no nested forbidden rectangles but found this to cumbersome, so nested forbidden rectangles were permitted. This is how we obtained 30 trials for each distinct number of semaphores from 1 to 15.

The counting of paths was accomplished by considering all nodes of the transition digraph which had no parent. These were potential starting points for maximal dipaths. A depth-first-type search was conducted recursively using the rule: number of paths from parent equals sum of number of paths from its children. Terminal nodes (without children) were assigned the value of 1. A dynamic programming technique reduced the time complexity of this task.

## 4.2 Analysis

### 4.2.1 Results

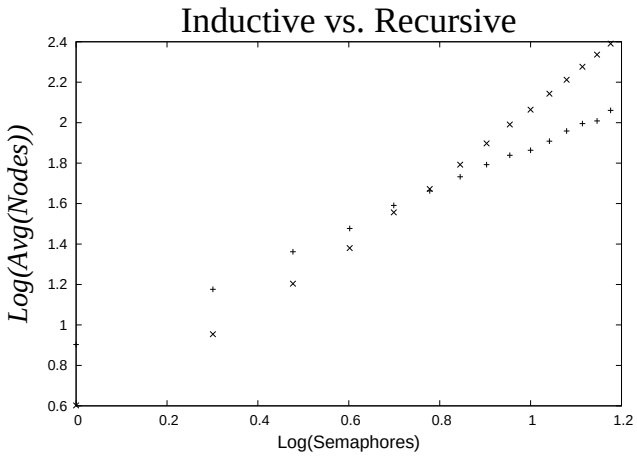


Figure 8: The inductive algorithm's data for forbidden rectangles has a steeper slope than that for the recursive algorithm.

One observes in Figures 8 and 9 that the recursive algorithm outperforms the inductive algorithm in terms of the number of nodes, first of all. Moreover, the recursive algorithm outperforms the inductive one in terms of the number of edges also. There is an even more pronounced difference when we consider paths. See Figure 10.

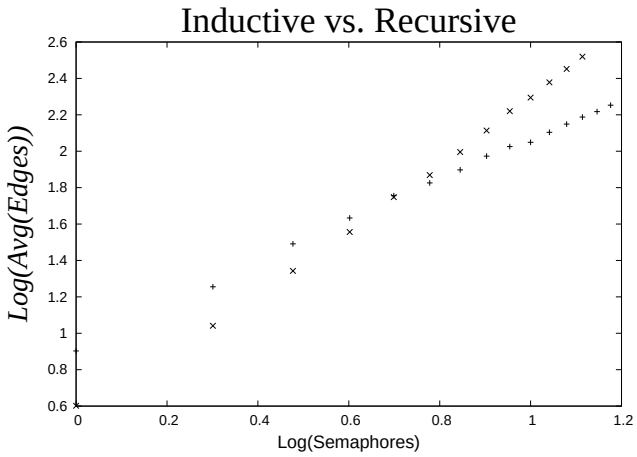


Figure 9: The inductive algorithm's data for forbidden rectangles has a steeper slope than that for the recursive algorithm.

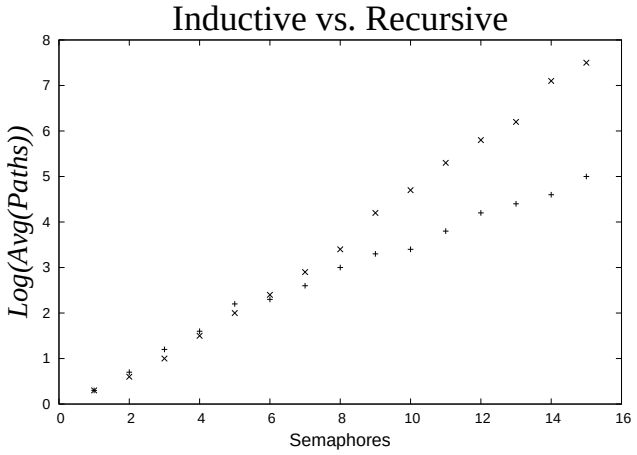


Figure 10: The top (respectively, bottom) plot is for the inductive (recursive) algorithm.

Regression analysis on the nodes and edges vs. semaphores data for forbidden rectangles has shown that the complexity is approximately $\Theta(m^2)$ for the inductive algorithm and approximately $\Theta(m)$ for the recursive algorithm. There is an exponential reduction in the number of paths per semaphore as well.

#### 4.2.2 Discussion

For forbidden points, we expect the recursive algorithm to produce an approximately linear number of regions per semaphore, and the inductive algorithm to have a (greater) complexity of $\Theta(m^2)$.

The reason for the prediction of $\Theta(m)$ for the Recursive Reduction Algorithm is that each boundary between L-shaped regions "consumes" at least two vertices, and in the best case, there will be at most a bounded number of regions within the partitioned L-shaped region.

The reason for the prediction of $\Theta(m^2)$ is that one may view each forbidden rectangle as producing a vertical and a horizontal boundary, since line segments are extended from the minimal and maximal corners in both directions. This should yield a set of line segments near each edge of the progress graph of cardinality

$m + 1$, so the Inductive Reduction Algorithm generates approximately an $(m + 1)$ x $(m + 1)$ grid.

# 5 Future Work and Conclusion

We believe that the recursive algorithm can be extended to higher-dimensional progress graphs simply by using hyperplanes instead of line segments. We intend to implement a higher-dimensional version of the recursive algorithm and obtain complexity data for the Dining Philosopher problem, as Goubault and Haucourt [GH05] have done for their inductive algorithm. This will involve exploration of the reduced state transition system obtained by the recursive algorithm. We describe briefly now how to get *admissible paths* from the paths of the transition graph. One has to keep historical information as the graph is explored to ensure that the signature remains admissible. This can be done by maintaining a current minimum possible value for the progress (coordinate) of each process (axis).

One must weigh the benefit of the reduced state-space against its complexity; yet we have found evidence that a recursive approach is superior to an inductive one in terms of the number of regions in the transition graph. Analysis of the cost of the reduction will be done for the higher-dimensional implementation.

# References


[BL01] Benedikt Bollig and Martin Leucker. Deciding LTL over Mazurkiewicz traces. In *TIME*, pages 189–197, 2001.

[GH05] Eric Goubault and Emmanuel Haucourt. A practical application of geometric semantics to static analysis of concurrent programs. In *CONCUR*, pages 503–517, 2005.

[GR02] Eric Goubault and Martin Raussen. Dihomotopy as a tool in state space analysis. In *Latin American Theoretical INformatics*, pages 16–37, 2002.