PPEM–86–04

The Use of Shape Grammars in Processor Embeddings*

Duane A. Bailey
Janice E. Cuny

COINS Technical Report A–86–23
July 1986

Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts 01003

*The Parallel Programming Environments Project at the University of Massachusetts is supported by the Office of Naval Research, contract N000014-84-K-0647. Duane Bailey was also supported by an American Electronics Association ComputerVision fellowship.

Abstract

Programming for parallel architectures often includes the specification of a *program embedding* in which logical processes and their interconnections are mapped onto physical processor elements and their connecting links, respectively. Few parallel programming environments provide assistance in performing this embedding. We are investigating the use of shape grammar descriptions of embeddings. We report on a specific grammar that we have developed for generating embeddings of arbitrarily large, complete binary trees in square processor arrays. We also discuss generalizing these techniques to produce efficient embeddings of other regular logical structures.

1. Introduction

Within the next decade, parallel architectures composed of a thousand or more processing elements will be commercially available. As the number of processors grows, however, programming them will become an increasingly cumbersome task. This is especially apparent in the activity of *program embedding* in which logical processes and their interconnections are mapped onto physical processing arrays. Aspects of this mapping problem have received significant amounts of attention, yet current parallel programming environments have been able to provide only limited support. We are investigating the use of grammar-based graph descriptions in the development of a strategy for program embedding. We report here on the characteristics of a specific embedding that we have found for mapping arbitrarily deep, complete binary trees in square grid processor arrays and on the potential for generalizing these techniques to other regular logical structures.

In the next section we describe the tree embedding in terms of specialized shape grammars. Section 3 contains a proof-of-correctness for our shape grammar and identifies salient points of similar proofs for grammars used in other embeddings. In the last section, we draw some conclusions on the advantages of this program embedding strategy and its extension to other regular structures.

2. A Shape Grammar for Embedding Trees

Previous embeddings of binary trees into rectangular arrays have been of two types: hand embeddings and recursive embeddings. Hand embeddings[7] have optimal processor utilization but they will not be feasible for the huge numbers of processing elements that we anticipate in future machines. Recursive embeddings[3]-[4] can be automated but they have so far failed to achieve the efficiency of hand embeddings. The Hyper-H embedding, for example, is a useful method for laying out the tree for VLSI but its processor utilization is not optimal. Our strategy, based on specialized *shape grammars*[5], automatically generates optimal embeddings for arbitrarily large trees – even in the case where the number of logical nodes exceeds the number of available processors.

In this section we describe a technique for performing tree embeddings for the CHiP architecture[7]. A CHiP machine (Figure 1) consists of a $2^n \times 2^n$ array of processors with local memory. *Corridors* of switches are located horizontally and vertically between rows of processor elements. We require that each switch have degree 8 and a crossover level of 3, allowing it to host three simultaneous communication paths.

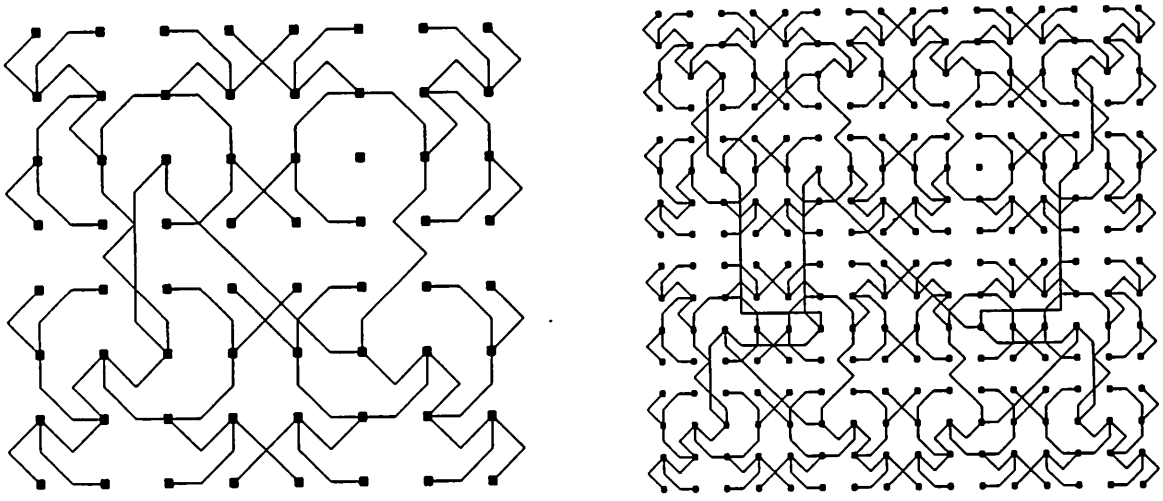


Figure 1: Tree embeddings in the CHiP processor array. Squares are processors and lines are channels. Switches are not shown.

Our tree embedding technique uses a two-phase shape grammar[1] to determine the node assignments and switch settings. In the first phase, processes are embedded into the processor array. For this we use a type of parallel shape grammar, called a *template grammar*, in which derivations are constrained to rewrite the entire sentential shape at each step; that is, the left-hand sides of the set of productions applied must be a minimal set covering the sentential form. In the second phase, communication channels are routed. For this we use a type of sequential shape grammar, called a *ruler grammar*, in which scaling transformations are not allowed.

The template grammar for the first phase of our tree embedding is shown in Figure 2. The letters that appear on the shapes are not actually part of the shapes – they are simply coordinate labels; nonterminal shapes contain an arrow, while terminal shapes do not. A sample derivation is shown in Figure 3. Because the entire shape must be rewritten in every step, all successful derivations in this grammar have the same form: first odd numbered productions are applied for an arbitrary number of steps and then even numbered productions are applied for a single step. The very first production divides

[1] A shape grammar operates on shapes in much the same way that a conventional grammar operates on strings: the portion of a sentential form that matches the left-hand side of a production is replaced by its right-hand side. Unlike string grammars, however, the productions of a shape grammar may undergo arbitrary Euclidean transformations before matching [5].

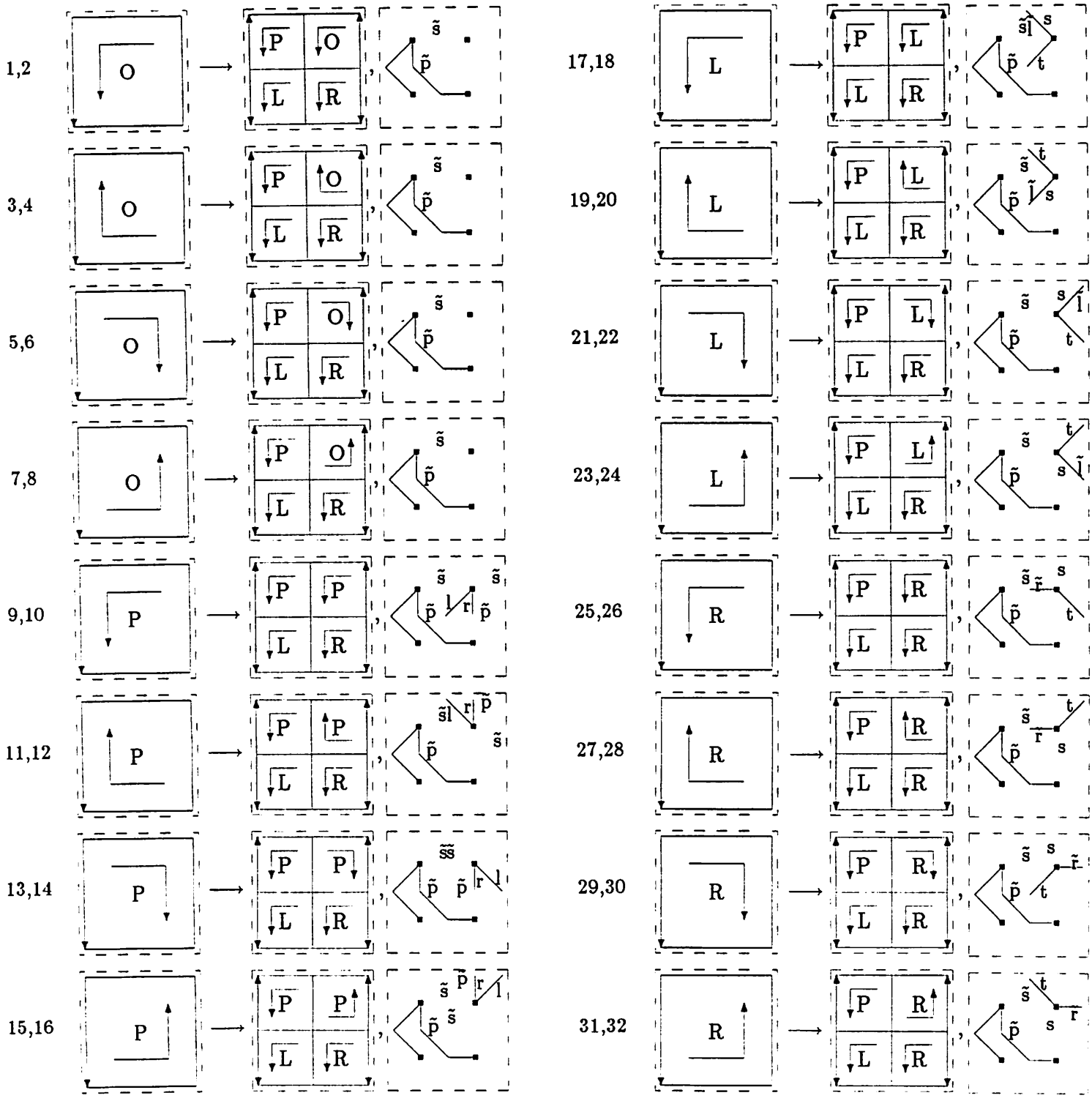


Figure 2: The process embedding grammar. Note that the start shape (left side of production 1) may be scaled before the derivation begins.

the processor array into quadrants. The root of the tree will be located in the quadrant labeled P and its left and right children will be located in the quadrants labeled L and R respectively. The single unassigned 'orphan processor' will be located in the upper right quadrant, labeled O. To embed the remainder of the tree, the array is recursively subdivided into quadrants which will host lower level subtrees. In the final step, processor nodes and 'buds' of communication channels are laid down, all of the nonterminal shapes are removed, and we are left with a start shape for the second phase of the grammar.

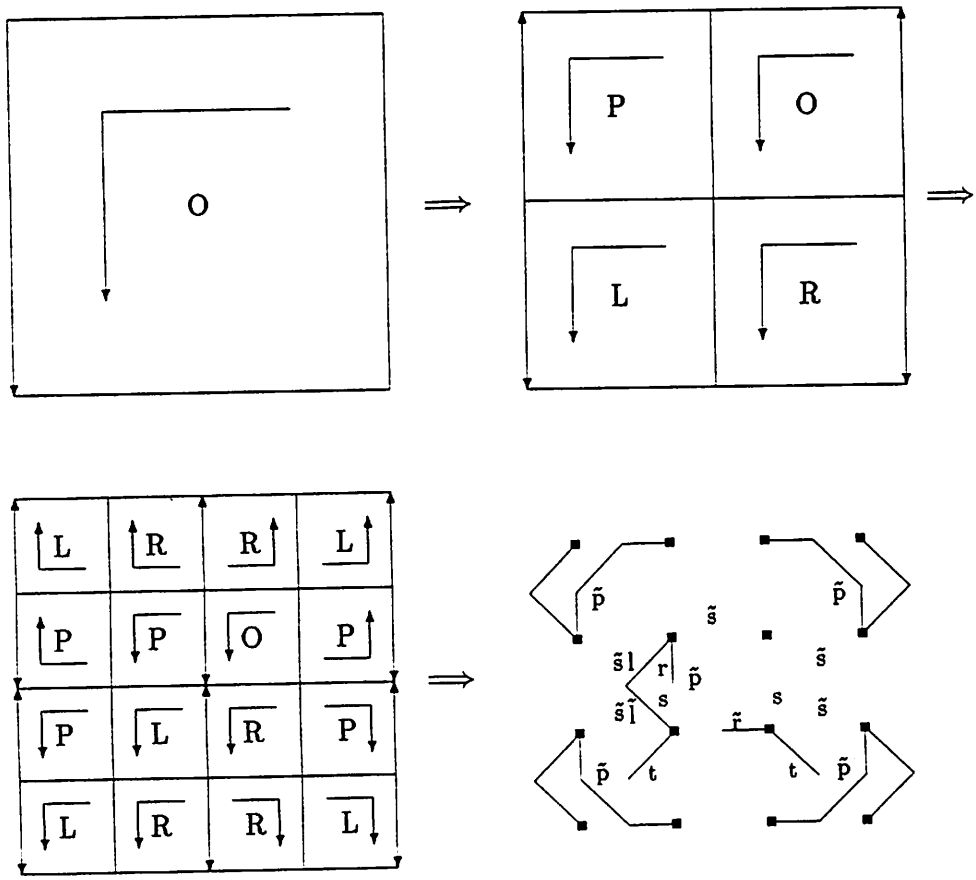


Figure 3: Derivation of a terminal shape by the process embedding phase. The result is a start shape for the channel embedding phase.

A ruler grammar for the channel embedding phase is given in Figure 4. In this case, the letters are again used for labels but their positioning is significant. The grammar works by

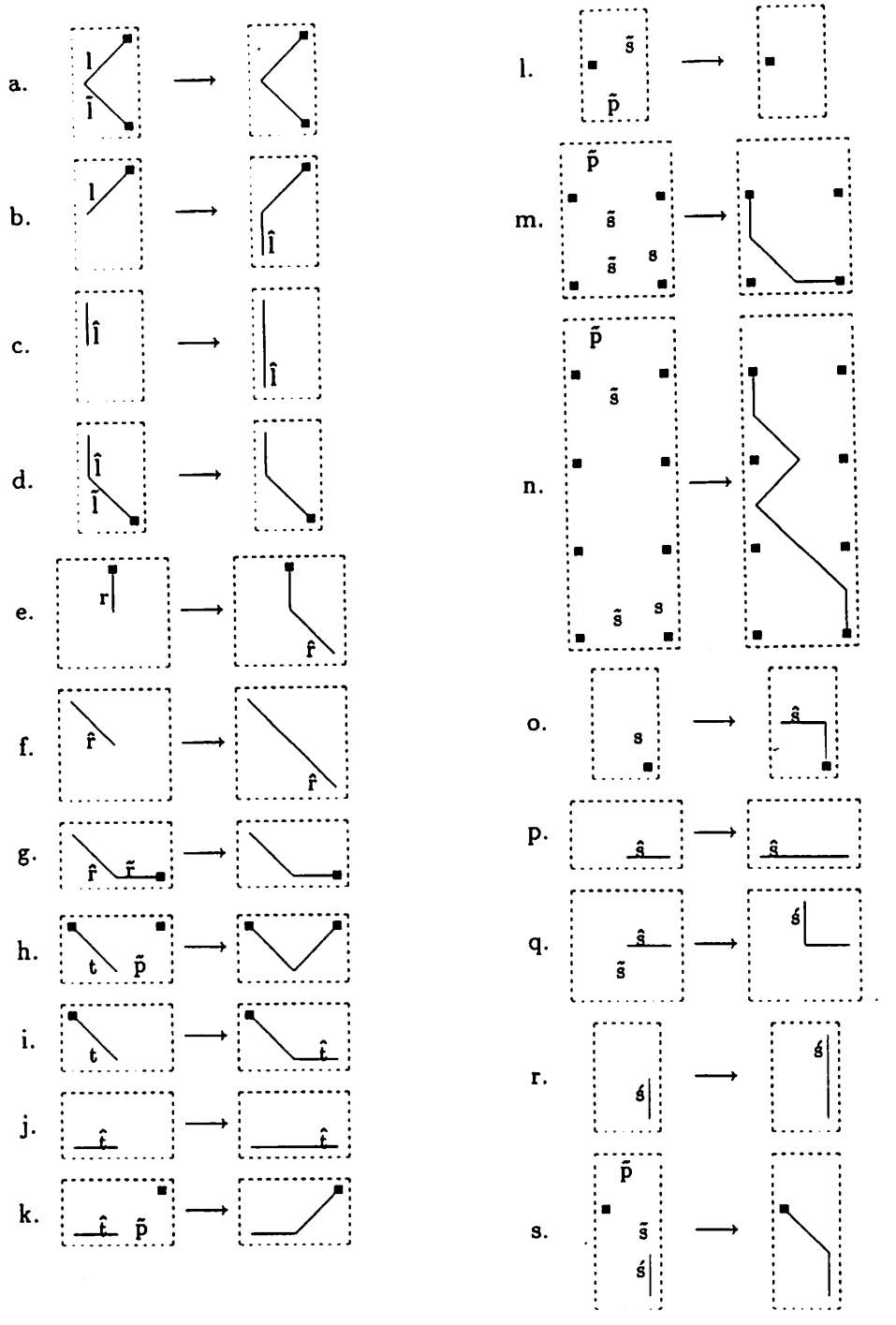


Figure 4: The channel embedding grammar. Any shape accepted by the first grammar is a valid start shape for this sequential shape grammar. Productions *a–d*, for example, draw connections from the root to its left child.

'growing' the channels from the buds left by first phase in such a way that no two channels share a common wire. Figure 5 shows several steps in a sample derivation. Derivations which erroneously create channels that are not part of our embedding will fail to rewrite all nonterminal markers. Final embeddings produced by our grammar for a 63 node tree in an 8×8 grid and a 255 node tree in a 16×16 grid are shown in Figure 1. It should be noted that the quadrants of the 16×16 embedding are extensions of the 8×8 embedding: the smaller 8×8 embedding is found in quadrant III of the large embedding, and can be seen in the other three quadrants flipped horizontally (quadrant II) or vertically (quadrant IV) or both (quadrant I).

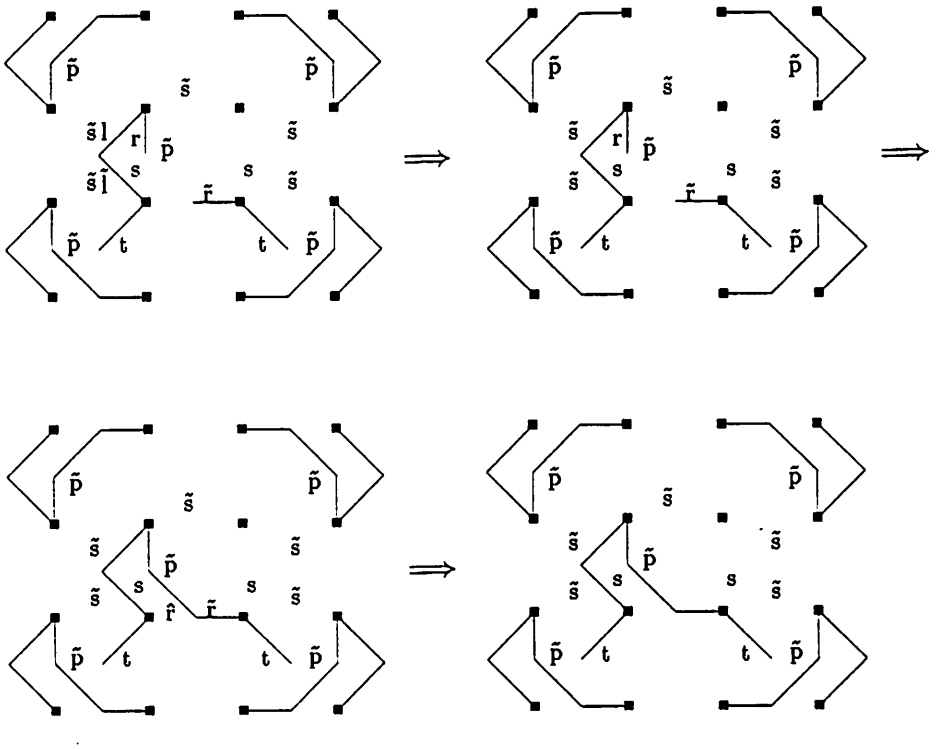


Figure 5: Routing of channels from a left and right son to their parent. The start shape is the final shape of Figure 3.

In the next section we investigate the proof-of-correctness. This proof is interesting to us, not only for the obvious result, but also because it suggests certain properties of shape grammars that make proofs-of-correctness less difficult.

3. The Proof of Correctness

In this section we present a proof demonstrating that the above grammar generates exactly the language of tree embeddings. The proof is not unlike proofs-of-correctness for string grammars: we first demonstrate the grammar generates exactly the embedding shapes, and then concern ourselves with showing that no non-embedding shape is generated.

We begin with a number of definitions that are specific to the proof, aided by Figure 6.

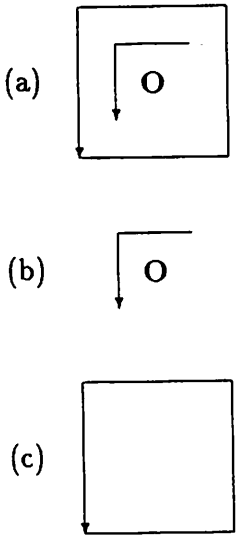


Figure 6: Anatomy of an lhs-shape: (a) shows the canonically oriented start shape, (b) shows its interior and (c) shows the exterior.

Definition 1 (initial definitions)

- *A <u>lhs-shape</u> (left-hand side shape) is any shape forming the left side of a production of the process embedding grammar. (The shape of Figure 6a is, in fact, the start shape, and the lhs of production one.)*
- *The <u>interior</u> of a lhs-shape consists of the bent-arrow and letter. The <u>exterior</u> is the oriented box surrounding the interior.*

- *The canonical orientation of a lhs-shape is the orientation that fixes the exterior's arrow pointing down and to the left. A production is in canonical orientation if it's left side is canonically oriented. A canonical derivation is a derivation that begins with a canonically oriented start shape.*

□

We now make some initial observations on the nature of the two-phase embedding grammar – especially with respect to the sentential forms generated.

Observation 2 (initial observations)

1. *The only lhs-shape that can cover lhs-shape s is s itself. Furthermore, if S is a set of lhs-shapes, S can only cover lhs-shapes that are members of S.*
2. *Each production set used in the derivation of a terminal shape is composed of either terminal or nonterminal productions – not both. Should a production set involve both a terminal and nonterminal production the result of its application is a shape with terminal and nonterminal components. The next derivation step, however, cannot be made up of productions whose left sides cover the terminal portions of the shape since terminal shapes of the process embedding phase cannot be composed from nonterminal shapes.*
3. *Derivations of process embedding shapes are deterministic. Since lhs-shapes are mentioned by exactly one terminal and one nonterminal production, production sets are determined solely by the shape to be rewritten, and the choice to extend or terminate a derivation. Thus, if nonterminal shapes s_1 and s_2 are derived from the same start shape, the derivation of one is a prefix for the derivation of the other.*
4. *Given any particular subshape, each routing production can be applied in at most one particular orientation; there is no ambiguity in the application of routing productions.*
5. *The set of lhs-shapes is closed under horizontal and vertical flips of the interior and rewriting of the marker from the set $\{O, P, L, R\}$.*

□

The proof itself begins with a series of characterizations of the shapes produced at various stages of a derivation, leading to Lemma 8 which states that all of the sentential forms produced are 'foldable'. This property is the key to the first half of the proof where we demonstrate that the grammar generates the desired embeddings.

In any embedding of a $(4^n - 1)$-node binary tree in a $2^n \times 2^n$ square grid, some processor must be left unassigned. This processor is termed the orphan processor, and its position is marked by the $\downarrow\overline{O}$ in sentential forms of the first phase. Formally, we have

Definition 3 (orphan definition) *The orphan position in a nonterminal process embedding shape is the position assigned the $\overline{\downarrow O}$ marker; the orphan processor of a terminal process embedding shape is the processor generated in the final step of the derivation at the orphan position.* □

While the grammar is formally defined to have the start shape depicted in Figure 6, very similar shapes can be generated from other lhs-shapes. Consider, for example, the derivation shown in Figure 7a; it is the same as the canonical derivation shown in Figure 7b, except in its orientation and the labeling of the position corresponding to the orphan. For any derivation, there is a similar canonical derivation. If we define a shape-rewriting function ϕ that maps shapes in the canonical derivation to their similar counterparts, we can express their relation formally with the aid of Figure 8.

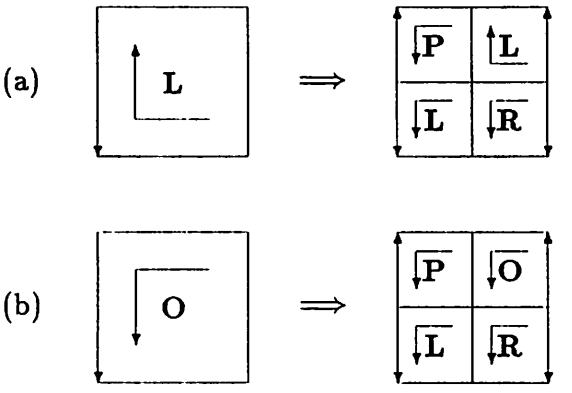


Figure 7: Similar canonical derivations. The similar canonical derivation for the derivation shown in (a) is depicted in (b). Quadrant I of the right side of (a) is the distinguished position since it corresponds to the orphan position of the right side of derivation (b).

Lemma 4 (similar derivation lemma) *Given nonterminal shape $D'_n(s')$ generated by an n-step derivation D'_n from lhs-shape s', there exists a similar canonical derivation D_n from the start shape s, which can be transformed into D'_n using a function $\phi_{s,s'}$ which rewrites the $\overline{\downarrow O}$ marker to the interior of s'.*

Proof: For derivations involving a single production, the lemma is true by examination.

Suppose the induction hypothesis holds true for $n-1$ step derivations. Let D_n and D'_n be the n step derivations from s and s'. Observation 2, part 3, indicates that these are

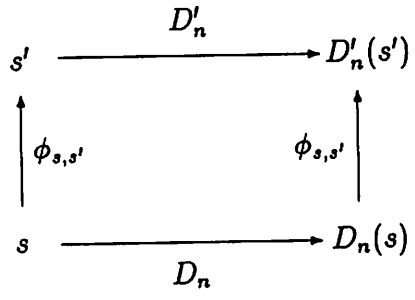


Figure 8: Some canonical derivation D_n can be made equivalent to D'_n by the use of a simple shape rewriting function $\phi_{s,s'}$.

extensions of D_{n-1} and D'_{n-1}, respectively. Since the induction hypothesis states that $\phi_{s,s'}$ rewrites D_{n-1} to D'_{n-1}, we need only show the final production set of D_n can be rewritten to the final production set D'_n, using $\phi_{s,s'}$.

Since $\phi_{s,s'}$ rewrites the interior of s to the interior of s', the only differences between canonically oriented shapes $D_{n-1}(s)$ and $D'_{n-1}(s')$ are in the orphan position. Since the final production set's left sides must cover the shape $D_{n-1}(s)$, the interior of s must be mentioned by the left side of a production. Likewise, the final step of D'_n must also cover the shape $D'_{n-1}(s')$. Here, a production whose left side consists of the lhs-shape s' is used. The two nonterminal productions whose left sides are s and s' are themselves made similar by $\phi_{s,s'}$. Since $D_{n-1}(s)$ and $D'_{n-1}(s')$ are the same elsewhere, it is clear to see the final production set is rewritten as stated, the induction hypothesis is proved and the lemma stands. □

Lemma 4 is important because it provides us with a basic tool for understanding the structure of sentential forms that are generated by differing lhs-shapes.

The treatment of the orphan position in derivations is important. Many statements about the orphan position, in fact, are independent of the initial shape of the derivation. The ϕ-image of this position is important enough to warrant the following definition.

Definition 5 (distinguished position definition) *The distinguished position is that position in a sentential form which, the similar canonical derivation from the start shape, corresponds to the orphan position. The distinguished processor is the processor generated by the final step of the process embedding phase from the distinguished position.* □

We now see that these images can be generated independently in each of the four quadrants.

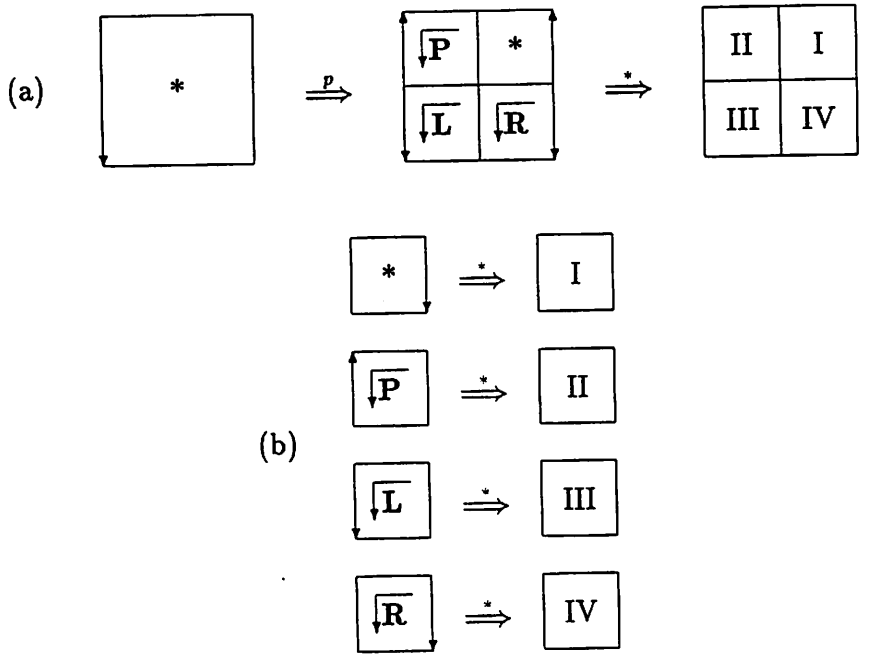


Figure 9: The derivation in (a), beginning with production p, can be seen to be the union of four shorter derivations from the various lhs-shapes mentioned in the right side of p as shown in (b).

Lemma 6 (derivation decomposition lemma) *Any derivation of a process embedding shape starting with nonterminal production p may be considered the union of four derivations from the four lhs-shapes found in the quadrants of the right side of p (see Figure 9).*

Proof: From Observation 2, part 1, no lhs-shape can be made up of portions of two or more lhs-shapes, so in order to cover the right side of production p four productions must be applied, each having a left side matching a quadrant. These productions are applied independently and in parallel. This independence is carried through the derivation, inductively. Now, for each derivation step after the first, we construct four new production sets consisting of elements of the derivation's production set that are applicable to respective quadrants. Since each quadrant of the right side of p is a lhs-shape, these production sets describe four independent, valid derivations of the quadrants final embedding shape. □

This brings us to the important property of foldability. An embedding is foldable if all four of its quadrants can be identified with the 'napkin-folding' shown in Figure 10. We give a precise definition, below.

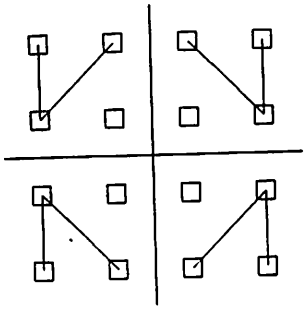


Figure 10: A foldable shape. Each quadrant can be made equivalent to any of the others by combinations of vertical and horizontal flips.

Definition 7 (folding equivalence definition) *The quadrants of a shape are folding-equivalent if quadrant I is related to the other three under the following transformations:*

- *quadrant II is a vertical flip of quadrant I,*
- *quadrant III is a vertical and horizontal flip (half rotation) of quadrant I and*
- *quadrant IV is a horizontal flip of quadrant I.*

A sentential form in the embedding grammar is foldable if the quadrants, less the embeddings in their distinguished processors (or positions), are folding-equivalent. □

Since derivations from lhs-shapes are related by the transitivity of similar derivations (Lemma 4) an important statement can be made about the regularity of the sentential forms. Namely, we may think of derivations of sentential forms as being a single derivation generating foldable shapes, or we may decompose a portion of the derivation into four *folding-equivalent* derivations – each from a quadrant of the right side of the first production.

Lemma 8 (equivalent derivation lemma) *The folding property is characteristic of two specific types of sentential forms of the two-phase grammar:*

1. *Disregarding distinguished processors (or positions) and their channels, process embedding shapes are foldable.*

2. *Suppose S is a sentential form derived from a process embedding shape p by productions that preserved the folding symmetry. Then any production applicable to one quadrant of S (less its distinguished processor and channels) is equally applicable in the other three quadrants. Together, these productions serve to rewrite S in a manner that preserves its foldability.*

Proof: We prove these independently, noting that if a nonterminal embedding shape is rewritten using terminal productions, they preserve the foldability of the shape.

1. Suppose the process embedding shape, s, is 2×2. Each node is a distinguished processor of its respective quadrant and s is trivially foldable.

   For larger shapes, realize that the right side of any production is foldable. If, using Lemma 6, we think of the derivation as the tiling of four derivations from the quadrants of the right side of the first production, Lemma 4 can be used to show the four shapes resulting from these four parallel derivations differ only in orientation and assignment of the distinguished processor. Since the transformations that canonically orient the quadrants of the right side of production p are equivalent 'modulo' the flips of folding-equivalence, the composite shape is foldable.

2. Suppose a sentential form in the channel embedding phase is folding equivalent. Any production avoiding the distinguished processor in one quadrant is equally applicable in similar contexts found in other quadrants. Since the orientation of the application of each routing production is unambiguous (by Observation 2, part 4), the application of a production in one quadrant, is folding-equivalent to its application in the other three other quadrants. Thus, if all four productions are immediately applied, the resulting shape is foldable.

□

The last section of the lemma underscores the importance of similarly embedded channels: as long as the channels of each quadrant are routed in the same fashion, the shape will maintain is foldability. As we will see shortly, our embedding process will take advantage of this fact when we consider the recursive embedding of a binary tree.

We concentrate first, however, on the actions of the channel embedding productions – productions that are applied during the second phase of the grammar. If we can guarantee that the method of routing channels always connects source processors to destination processors on one side, these channels cannot 'bend backwards' and run amok. This desirable characteristic of channel embedding production sets is described immediately.

Definition 9 (unidirectional definition) *The reachability set of a set of channel embedding productions R from processor P, is the set of processors that are potentially connected to P by channels embedded by R. A set of channel embedding productions, R, is unidirectional with respect to a set of processor shapes if and only if for each processor P the reachability set of R is collinear with, and bounded by P on one side.* □

Each production of the channel embedding grammar is responsible for routing some portion of one of four logical *wire* types, shown in Figure 11. We may group most of the routing productions that embed a common wire into sets labeled $wireset_l$, $wireset_r$, $wireset_t$, $wireset_{s_1}$, and $wireset_{s_2}$. These production sets, we suggest, will route the wires as indicated. We turn our attention to identifying the characteristics of these sets.

Lemma 10 (channel properties lemma) *The productions, potential derivations and reachability sets for each wire embedding set are as follows:*

wire set	rules	derivations	$\Delta rows$	$\Delta cols$
$wireset_l$	*a-d*	*a*, bc^*d	n	0
$wireset_r$	*e-g*	ef^*g	n	n
$wireset_t$	*h-k*	*h*, ij^*k	0	n
—	*l*	*l*	0	0
—	*m*	*m*	-1	-1
—	*n*	*n*	-3	-1
$wireset_{s_1}$	*o-p*	op^*	0	$-n$
$wireset_{s_2}$	*q-s*	qr^*s	$-n$	0

Furthermore, the production sets $wireset_l$, $wireset_r$, $wireset_t$, $wireset_{s_1}$ and $wireset_{s_2}$ are unidirectional with respect to processor shapes in sentential forms generated by the grammar.

Proof: We first demonstrate that $wireset_l$ is composed of exactly the productions $\{a-d\}$. Productions *a* and *b* are the only productions that mention marker *l*. Since production *b* generates $\hat{l}$ the closure of this set must include productions *c* and *d*. Since no production outside the set $\{a-d\}$ mentions the markers *l*, $\hat{l}$ or $\tilde{l}$, the closure is complete.

These same productions must be responsible for the embedding of root-to-left-child wires: Clearly one means of erasing an *l* and $\tilde{l}$ is to use the *a*, if applicable. Otherwise, one must first erase the *l* mark - changing it to the $\hat{l}$ mark - with production *b*, apply any number of *c* productions (which preserve $\hat{l}$ marks) and erase both of the $\hat{l}$ and $\tilde{l}$ marks with a final *d* production.

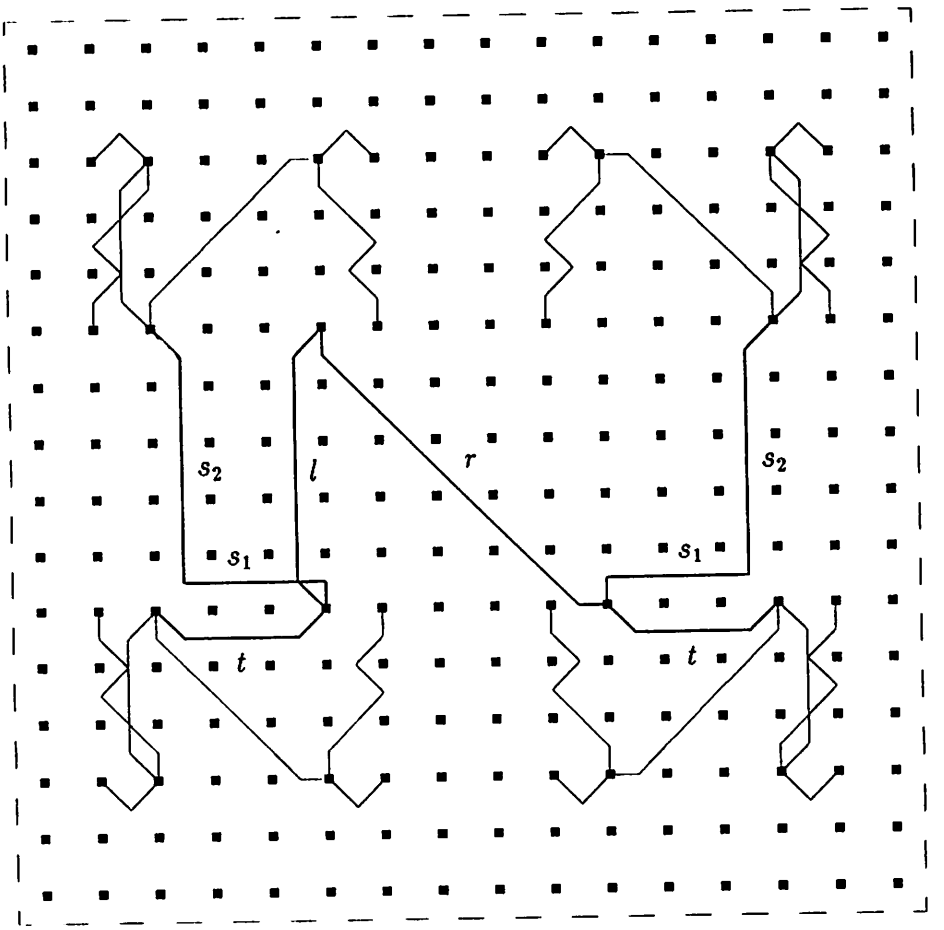


Figure 11: The layout for the six wires (thick, labeled) in a 16×16 processor grid. Thin lines show the six embedding wires of the previous level. Note that the zig-zag lines in the 8×8 embedding are special case embeddings using production n.

We need only show that the reachability set of $wireset_l$ consists of those processors found below the processor mentioned in the initial shape, in the same column. Observe the a production connects with the processor immediately below it. For $bc * d$ derivations, note that c preserves the current column, and that b and d are column-wise inverses of each other. Since each of the productions performs one switch hop in the south direction, these productions will only embed wires to southerly processors. The fact that this production set is unidirectional falls out immediately from the collinearity of the reachability set, and the fact that the reachability set of the processors consists of only southerly processors.

Proofs for sets $wireset_r$, $wireset_t$, $wireset_{s_1}$ and $wireset_{s_2}$ are similar. □

At this point, we are ready to use the folding properties to complete the first half of our proof with Lemmas 11–14. A 'six-wire embedding' embedding routes six logical channels at each level of the embedding in the following way (see Figure 11). If the embedding is larger than 4×4 each quadrant is completely wired independently using a six-wire subembedding. The root is then embedded at the distinguished processor of quadrant II while its two children are embedded in the distinguished processors of quadrants III and IV. Channels are routed downward and to the lower right from the root to form links to the children. These children are connected to the grandchildren of the root, located in each of the four quadrants the following fashion: if the grandchild is located on the same row as its parent, a horizontal wire is used as the connecting channel, otherwise an 'L' shaped wire extends first horizontally, then vertically to the upper quadrant. Short distance cases are embedded more directly with special case routing productions, as seen in the subembeddings of Figure 11.

Ignoring, for the moment, the constraints on the channels of the CHiP machine, the next lemma shows our grammar potentially generates the six-wire embedding with the connectivity of a complete binary tree.

Lemma 11 (potential connectedness lemma) *The two-phase grammar potentially generates the six-wire embedding which assigns processes to all but the orphan processor with the connectedness of a complete binary tree. Furthermore, the root is embedded horizontally opposite the orphan processor.*

Proof: If this shape is 2×2 the resulting shape is immediately a complete binary tree of depth one. Its root processor is located directly across from the orphan.

If the embedding shape is larger, we may assume (by Lemma 8, part 1) that each quadrant is equivalent – up to its distinguished processor. The inductive hypothesis suggests that six-wire embeddings of smaller arrays have the connectedness of a complete binary tree. In particular, each of the four quadrants may be wired individually, or as a whole

(using Lemma 8, part 2) as part of this embedding. Thus each quadrant has a similar complete binary tree, rooted across from its distinguished processor.

Observe Figure 11. Since the process embedding shape is foldable, and the six-wire subembeddings of each quadrant preserve this, the root and distinguished processors in quadrants I and II share columns with the root and distinguished processors in quadrants IV and III respectively. In addition, the distinguished processors in quadrants I and IV share rows with the distinguished processors in quadrants II and III respectively, and the distinguished processor of quadrant II is diagonally opposite the distinguished processor in quadrant IV. If the root is embedded in quadrant II, the left and right children located in quadrants III and IV can be connected with wire embedding production sets $wireset_l$ and $wireset_r$ respectively. Furthermore, since each root processor is in the same row as its distinguished processor, $wireset_t$ can be used to embed the wire connecting children and grandchildren found in the lower quadrants. For embeddings of size 4×4 production m can be applied to the left and right sides (flipped appropriately) to connect the upper grandchildren. Production n connects children and upper grandchildren of 8×8 processor embeddings. For larger embeddings, we connect children with upper grandchildren by first using wire embedding set $wireset_{s_1}$ to route the wire toward the lower grandchild and then using $wireset_{s_2}$ to route the wire upward along the column to the upper grandchild. Our statement that a channel embedding production set 'can be used' makes the assumption that the channel buds of the respective processors are oriented correctly with respect to each other. This may be easily verified – remembering that buds are also effected by the vertical and horizontal flips of the folding relation.

Important to note is the fact that no wire embedded uses the context of the orphan processor – thus similar derivations from embedding shapes starting with different lhs-shapes are equivalent up to the distinguished processor. In particular this embedding of a $2^n \times 2^n$ array may be used as an embedding of a quadrant of a $2^{n+1} \times 2^{n+1}$ embedding.

Since the root is connected to four grandchildren, each of which is a complete binary tree, the entire embedding must be a complete binary tree. A final production, l, serves to erase the markers left on the root of the processor embedding – the only two remaining. □

While it seems logical that the wires can be routed, we have no guarantee that there are no wire embedding conflicts in switch corridors. To show these free of multiplexing, we identify specific corridors as 'reserved corridors'. If these corridors are not used, we can be assured that each subembedding's reserved corridors leave room to embed the six wires of larger embeddings. Figure 12, and the following definition more formally delineate the reserved corridors.

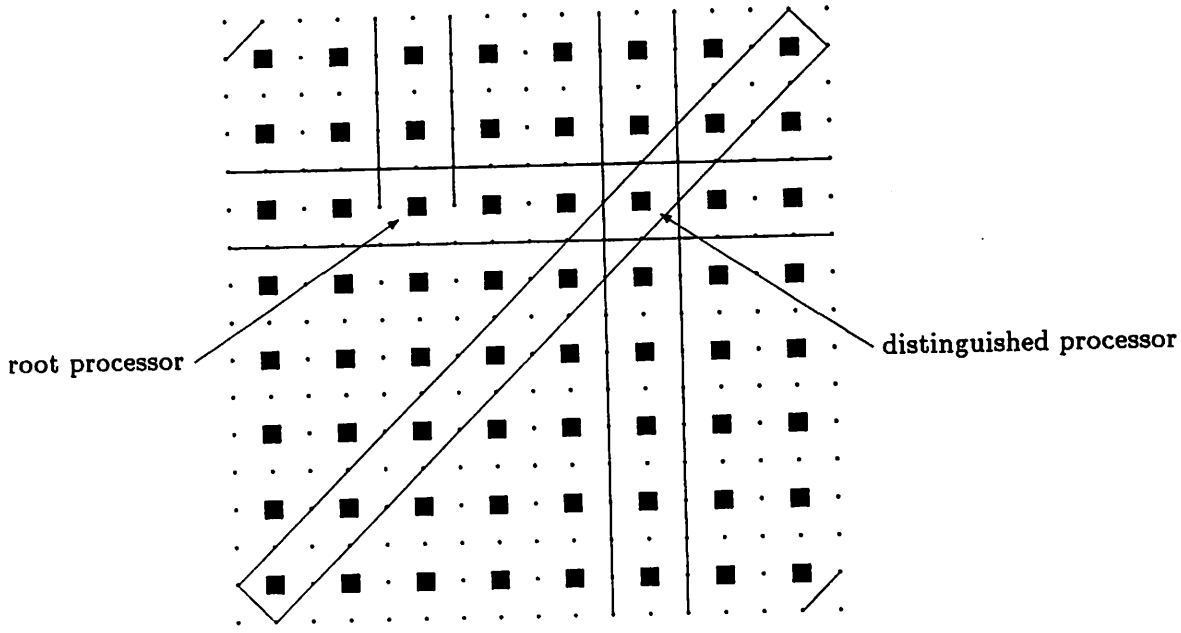


Figure 12: The reserved corridors of an 8 × 8 embedding.

Definition 12 (reserved corridors definition) *Reserved corridors consist of the following switch connection corridors:*

- *horizontal corridors immediately above and below the distinguished processor,*
- *vertical corridors immediately to the left and right of the distinguished processor,*
- *major diagonal corridors immediately above and below the distinguished processor,*
- *minimal diagonals that clip each corner of the array and*
- *the vertical corridors to the left and right of and above the root processor.*

□

Each six-wire embedding, can now be shown to avoid the reserved corridors.

Lemma 13 (reserved corridor lemma) *Each six-wire embedding has the property that no reserved corridor is used.*

Proof: We use a proof by induction on the size of the embedding. For a 2×2 embedding the lemma is true by observation.

Suppose the hypothesis is true for $2^{n-1} \times 2^{n-1}$ embeddings. As in the previous lemma, we consider an embedding of a $2^n \times 2^n$ array based on the embeddings of four $2^{n-1} \times 2^{n-1}$ quadrant embeddings, oriented as described in Lemma 8 part 1. We now consider the $2^n \times 2^n$ shape prior to the embedding of the six wires. This partially embedded shape is folding equivalent by Lemma 8, part 2 – if each subquadrant's wiring is identical and interleaved. Since the reserved columns of quadrants I and IV align (see Figure 13), they are available in the partially embedded shape. Similarly, the rows of quadrants I and II, the maximal diagonals of quadrants I and III and the corner-clipping diagonals can be seen to be reserved corridors and the reserved verticals above the root are simply portions of the reserved corridors about the distinguished processor in quadrant II. We need only show that the addition of the six wires that complete the embedding fails to use these corridors.

The connection between root and left child is located on the left half, and is vertical, below the root. It must, therefore, miss each of the reserved corridors of the $2^n \times 2^n$ shape. Since the diagonal from the root to right child extends northwest to southeast, it cannot impinge on the reserved corridors. The horizontal wires embedded by the $wireset_t$ productions are located on the lower half of the array, avoiding the horizontal reserved corridors which are above, and thus avoids using the reserved corridors. A similar argument can be made for the two wires embedded by the $wireset_{s_1}$ which parallel the $wireset_t$ wires in the lower half. The production set $wireset_{s_2}$ also is applied to both sides of the $2^n \times 2^n$ embedding when $n > 3$. The left wire certainly is independent of the reserved corridors, while the right wire runs along the inner vertical corridor between the right grandchildren. For $n > 3$ this corridor does not abut the orphan processor of the $2^n \times 2^n$ embedding (which is the distinguished processor of quadrant I) – the wire, therefore, fails to use the reserved corridors. Production m embeds child-to-upper-grandchild wires on both sides of a 4×4 array, while n embeds similar wires for 8×8 embeddings. It is easily observed that these wires fail to route channels over reserved corridors of their respective shapes.

Since none of the six wires embedded impinges on the reserved corridors, the induction hypothesis holds, and the lemma is true. □

We now demonstrate that the six-wire embeddings are generated by the grammar.

Theorem 14 (existence theorem) *The six-wire embedding shapes are generated by the grammar.*

Proof: Lemma 11 indicates that the six-wire embedding, if valid, will have the connectedness of a complete binary tree. Lemma 13 shows this embedding has the property that

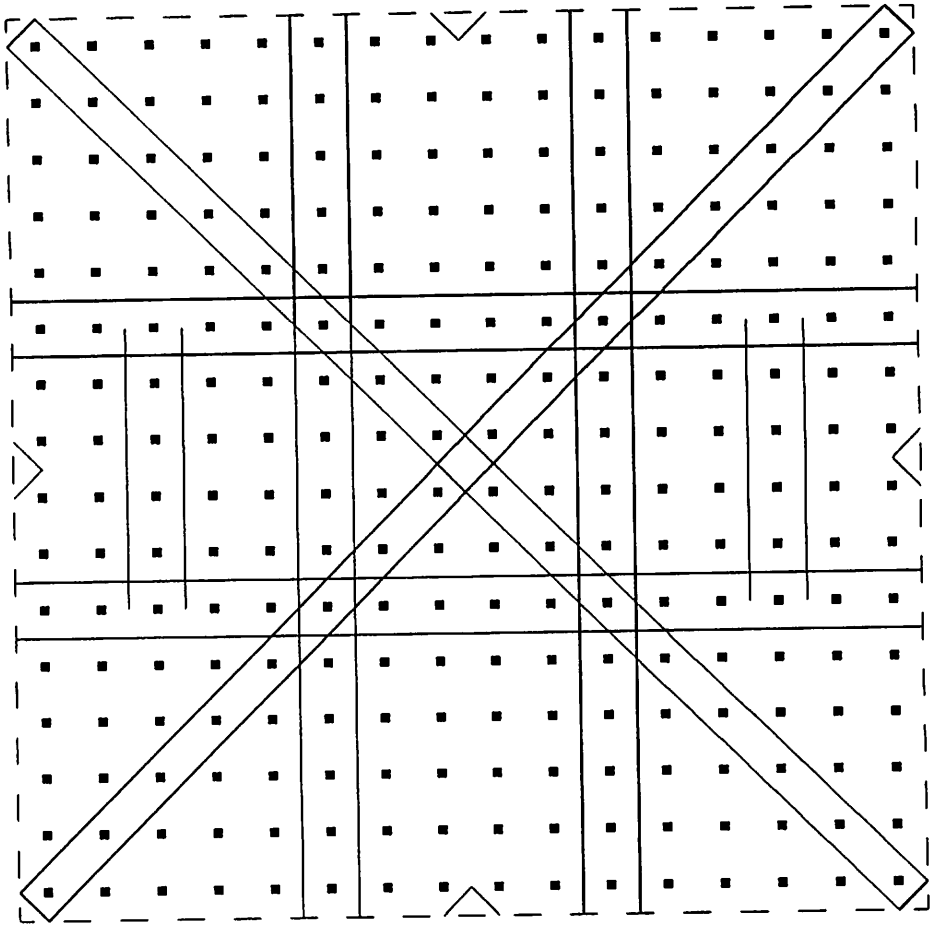


Figure 13: The alignment of the four quadrants' reserved corridors. The thick lines indicate the reserved corridors of the 16 × 16 embedding, which were portions of reserved corridors in smaller embeddings. Comparison with Figure 11 shows how the embedded wires are routed along reserved corridors.

it reserves corridors in final embedding shapes. Once again, an inductive proof demonstrates that the six wires added to the $2^n \times 2^n$ embedding use the reserved corridors of the $2^{n-1} \times 2^{n-1}$ embedding found in each quadrant, thus avoiding interference from previously embedded wires. This suggests each stage of the six-wire embedding is valid, leading to a final tree embedding. □

We have thus complete the first half of our proof. We now consider the task of showing that the grammar does not generate any other embeddings. First, we describe a property of collinear processors, connected by a common channel embedding production set.

Definition 15 (nice-property definition) *Let P be a set of processors and R be a channel embedding production set that partitions P into two sets, $source(P)$ and $destination(P)$ and induces a bijection, $f : source(P) \rightarrow destination(P)$ such that f identifies the two processors that support a channel routed by R. Then P is said to have the nice-property with respect to R if each of the following conditions hold:*

1. *R is unidirectional with respect to P, and*
2. *The channels routed by R fail to overlap each other.*

□

If we number the collinear processors, P, in the natural way, it becomes clear that condition two is equivalent to saying that for each pair s, s' in $source(P)$ with $s < s'$, the inequality $s < s' < f(s)$ never holds.

If a set of collinear processors is connected with a routing production set, we can demonstrate there is no way to re-wire these processors with that channel embedding production set.

Lemma 16 (unique connectedness lemma) *A set of processors P, with the nice-property with respect to routing production set R, is uniquely connectable by productions in R.*

Proof: Suppose that P is a set of processors with the nice-property. We can partition P into $source(P)$ and $destination(P)$ such that $source(P)$ contains processors that initiate channels and $destination(P)$ contains those that accept channels. As a result of the nice-property $source(P)$ and $destination(P)$ are disjoint and there exists a bijection $f : source(P) \rightarrow destination(P)$ which identifies the two processors that support a channel.

An immutable characteristic of the processor orientation determined by the grammar is the direction of each processor's channel partner. We identify this with the two characteristic functions $exit : source(P) \rightarrow \{+, -\}$ and $enter : destination(P) \rightarrow \{+, -\}$. If a source processor's index is less than its partner, *exit* takes on the value $+$; elsewhere it

is $-$. Similarly, *enter* has the value $+$ for every destination processor whose source has a greater index, and $-$ for all others. Note that any valid routing bijection (in particular, f) must respect the characteristic functions *exit* and *enter*.

Suppose there is another bijective function with the nice-property, $f' : source(P) \rightarrow destination(P)$. Let $S^* = \{s | f(s) \neq f'(s)\}$ and $D^* = \{d | f^{-1}(d) \neq f'^{-1}(d)\}$, the sets composed of preimages and images of mappings f and f' where they differ. Since f and f' are bijections, when restricted to S^* they are also bijections onto D^*.

The set S^* has either a maximal member with $exit = +$ or a minimal member with $exit = -$. Without loss of generality, identify the maximal $s_M \in S^*$ such that $exit(s_M) = +$. The channels routed by R fail to overlap, since P has the nice-property with respect to R, thus $d_M = f(s_M)$ is precisely the maximal member of D^* with $enter = -$. Note that $d = f'(s_M)$ is not d_M, and yet $enter(d) = -$. This suggests that either $s_M < d < d_M$, which violates the hypothesis on overlapping, or $d < s_M$ which indicates that $exit(s_M) = -$, which is, of course, impossible. The theorem is true as stated. □

Since each production set is responsible for eventually erasing marks on source and destination processors that cannot be erased by other production sets, the channel must be routed if the embedding is to be accepted by the embedding grammar. If we can show each row, column and diagonal has the nice-property with respect to each routing production set that is unidirectional along that row, column or diagonal, then any embedding shape accepted has a unique wiring.

Lemma 17 (nice-property existence lemma) *Any wire set that is unidirectional along a row, column or diagonal of an embedding shape, also has the nice-property with respect to that row, column or diagonal.*

Proof: We know, by Lemma 10 that each of the sets $wireset_l$, $wireset_r$, $wireset_t$, $wireset_{s_1}$, and $wireset_{s_2}$ are unidirectional with respect to their start shapes. To show that a given row, column or diagonal has the nice property, we must show that, given a process set the set may be broken into two sets with the property that they may be connected by a bijective function. The bijection we will actually depend on will be induced by the six-wire embedding. We prove the lemma for columns used by $wireset_l$ – proofs of other cases are similar in nature.

We examine the columns of the process embedding with respect to P and L labeled processors and note that the six-wire embedding induces the appropriate bijection. Once again, it is an inductive proof. In a 2×2 embedding column one has one P and one L process, which are connected. Here, $source = \{(1,1)\}$ and $destination = \{(2,1)\}$. Column two has neither of the processor types and vacuously supports the lemma.

Now suppose the lemma holds for a $2^{n-1} \times 2^{n-1}$ embedding. Then we may consider a $2^n \times 2^n$ embedding as the tiling of four $2^{n-1} \times 2^{n-1}$ embeddings, whose distinguished processors are labeled with O, P, L and R. Now, for columns not containing a distinguished processor, no P-L wire spans the two quadrants, and the *source* and *destination* sets of the $2^n \times 2^n$ embedding simply consist of the union of the *source* and *destination* sets of the component quadrants. Two columns contain distinguished processors. The right column, like the previously mentioned columns, does not have a $wireset_l$ wire spanning quadrants I and IV, thus *source* and *destination* sets are similarly defined. The remaining distinguished processor column has a single $wireset_l$ wire spanning quadrants. This wire fails to overlap others embedded by $wireset_l$ since this wire is laid along reserved corridors. If *source* is defined to be the source processors of the appropriate columns of the two $2^{n-1} \times 2^{n-1}$ embeddings, augmented with the distinguished processor of quadrant II, and *destination* is defined as the *destination* sets of the same quadrants augmented with the distinguished processor of quadrant III, the bijection is formed. This was all that was necessary to guarantee the nice-property.

Similar arguments show rows, columns and diagonals have the nice-property with respect to $wireset_r$, $wireset_t$, $wireset_{s_1}$ and $wireset_{s_2}$ routing production sets. □

We have, now, the tools for demonstrating the correctness of the tree embedding grammar.

Theorem 18 (correctness theorem) *The grammar generates exactly the language of six-wire embeddings.*

Proof: The six-wire embedding has the nice-property with respect to rows, columns and diagonals, and thus each row, column and diagonal must therefore be uniquely connectable. If some other embedding existed, the connectivity of some row, column or diagonal would have different connectivity than that imposed by the six-wire embedding – which was discounted by Lemmas 16 and 17. The only shapes generable from the grammar are the six-wire embedding shapes. □

4. Conclusions

We have shown, in the previous sections, that shape grammars are capable of generating embeddings of binary trees in square grids. The motivation for this work has been to find more efficient embeddings of binary trees in CHiP-like processors. The grammar, in fact, is an improvement on previous embeddings, as we shall see in the next subsection. In addition, we shall discuss the possibility of generalizing this technique for other regular structures.

Processor Utilization in the Tree Embedding

When the number of logical processes exceeds the number of physical processors, it is necessary to contract the logical structure to fit the array. The contraction must map sets of processes to processors while preserving their connectivity. Fishburn and Finkel[2] have suggested using *quotient maps* to generate equivalence classes of processes which may be multitasked on a processor. If the cardinality of each equivalence set is the same, the contraction is said to be *computationally uniform*. A quotient map also induces a set of equivalence classes on the logical communication channels and, if every physical channel emulates the same number of logical channels, it is said to be *exchange uniform*. If a contraction is exchange and computationally uniform, it is said to be *totally uniform*.

Berman and Snyder[1] have suggested the quotient map shown in Figure 14 in which the depth of the tree has been reduced by one, the left and right subtrees of the root have been identified, and the root has been grouped with its two children. Further contractions would be accomplished by iterating this procedure. Repeated coallessing of the root, however, makes this contraction unattractive: it is not computationally uniform because the equivalence class of the root must always contain nearly twice the number of processes found in any other class.

Quotient maps for grids, however, contract more uniformly. Figure 15 details such a contraction. If the grid is folded like a napkin – horizontally and vertically – groups of four nodes are identified to form a smaller grid of equivalence classes. This contraction is totally uniform. It might be expected, therefore, that embeddings of trees that take advantage of grid contractions may yield more uniform distribution of computation and communication. We have found this to be true.

Any embedding of a contracted binary tree in a square grid can off-load extra processes from the root to the unused processor, provided it is possible to route a channel between them. Our embedding assures that the orphan processor is always located horizontally opposite the root, easily accessible to such a channel. We call a complete binary tree in which an extra process has been attached to the root in this way an *augmented tree*.

Extending our grammar to augmented trees in the obvious way produces an embedding with quadrants that are identical under vertical and horizontal flips and can thus be folded into a contraction. This contraction identifies nodes and channels in much same way as Berman's quotient map except in its use of the extra processor. It off-loads nearly half the processes that would otherwise be located at the root, making it exchange uniform and as computationally uniform as possible. Our strategy of first mapping the tree onto a large *logical* processor grid, and then contracting it to an augmented tree embedding of the correct size is conceptually distinct from the contract-then-map paradigm and yields a more uniform distribution of the computation and communication.

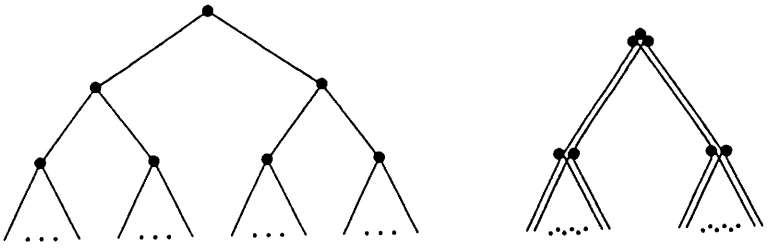


Figure 14: A tree contraction. The left and right subtrees are folded together to form a shorter tree.

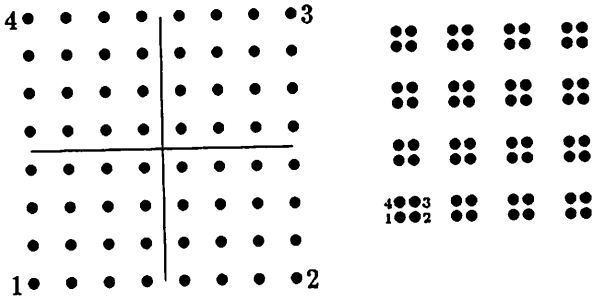
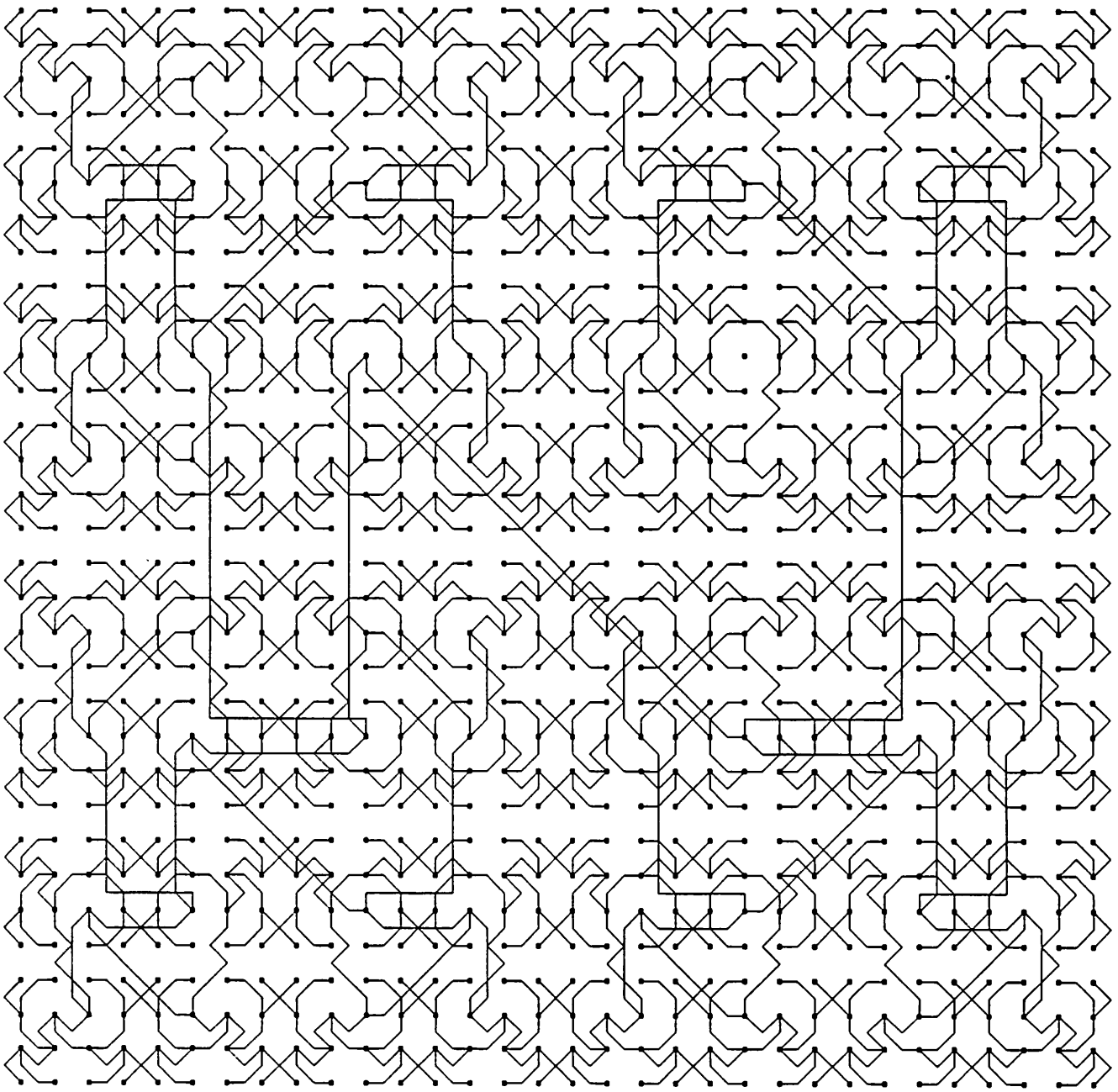


Figure 15: Quotient map for square grids. The array is folded like a napkin; all corners are mapped to the same node.

The Potential for Generalization

Because shape-related information is often important to the embedding process, we are investigating the use of shape grammars for describing layouts of other regular process structures. We expect that the strategy used in producing the tree embeddings – first recursively assign processes to localities and then route their interconnections – will work in creating layouts for other regular structures. We are now concentrating on the development of tools to automate some aspects of these grammatical descriptions in the hopes of making them more practical.

Proofs-of-correctness for shape grammars are more difficult than those for string grammars. We have, however, identified two useful concepts that may be used in generating shape grammars for other embeddings: reserved corridors and foldability. Reserved corridors assure that embeddings of arbitrary size are possible; while they are not necessary, they are sufficient. Foldability identifies the role of recursion in the shape grammar. In addition, it permits contractions induced by quotient maps of square grids, resulting in a uniform distribution of tasks and communication. We expect that this embed-then-contract approach will also be useful in producing uniform maps of other recursively described structures and we are investigating the extent to which it too can be automated.



REFERENCES

[1] Francine Berman and Lawrence Snyder, "On Mapping Parallel Algorithms into Parallel Architectures," Proceedings of the 1984 International Conference on on Parallel Processing, pp. 307-309 (1984).

[2] John P. Fishburn and Raphael A. Finkel, "Quotient Networks," *IEEE Transactions on Computers,* Volume C-31, Number 4, pp. 288-295 (1982).

[3] Haim Mizrahi and Israel Koren, "Evaluating the Cost Effectiveness of Switches in Processor Architectures," Proceedings of the 1985 International Conference on Parallel Processing, pp. 480-487.

[4] C. Mead and M. Rem, "Cost and Performance of VLSI Computing Systems," *IEEE Journal of Solid State Circuits,* Volume SC-14, Number 2 (1979).

[5] George Stiny, *Pictorial and Formal Aspects of Shape and Shape Grammars,* Birkhaüser Verlag, Basel und Studtgart, 1975, 414 Pages.

[6] Lawrence Snyder, "Parallel programming and the Poker Programming Environment," *Computer,* 17(7), pp. 27-37 (1984).

[7] Lawrence Snyder, "Introduction to the Configurable, Highly Parallel Computer," *Computer,* 15(1), pp. 47-57 (1982).