# Sensors, Patches, Pores, and Channels:
Progress on Universally Programmable Intelligent Matter

UPIM Report 5

Technical Report UT-CS-03-513

Bruce J. MacLennan*

Department of Computer Science
University of Tennessee, Knoxville
www.cs.utk.edu/~mclennan

December 29, 2003

**Abstract**

This report presents several systematic techniques for constructing complex nanostructures. First, we define a general "patch format," which allows membranes to be hierarchically and iteratively assembled from smaller elements. The usefulness of the technique is extended by demonstrating how such patches may be joined and transformed in various ways. Several membrane architectures, previously developed, are re-derived in patch format, demonstrating its utility. Similar approaches are applied to the assembly of nanotubes. Patch techniques are applied to the construction of channels in membranes, first to the synthesis of a simple (always open) pore, second to the use of sensor molecules, and third to their combination in a simple, open-once channel. Finally, the ideas of patch assembly are applied to the synthesis of non-unit mesh membranes.

## 1 Patches

Large membranes will not always be homogeneous in structure; often they will contain pores and active channels of various sorts embedded in a matrix. One way of

*This research is supported by Nanoscale Exploratory Research grant CCR-0210094 from the National Science Foundation. It has been facilitated by a grant from the University of Tennessee, Knoxville, Center for Information Technology Research. 

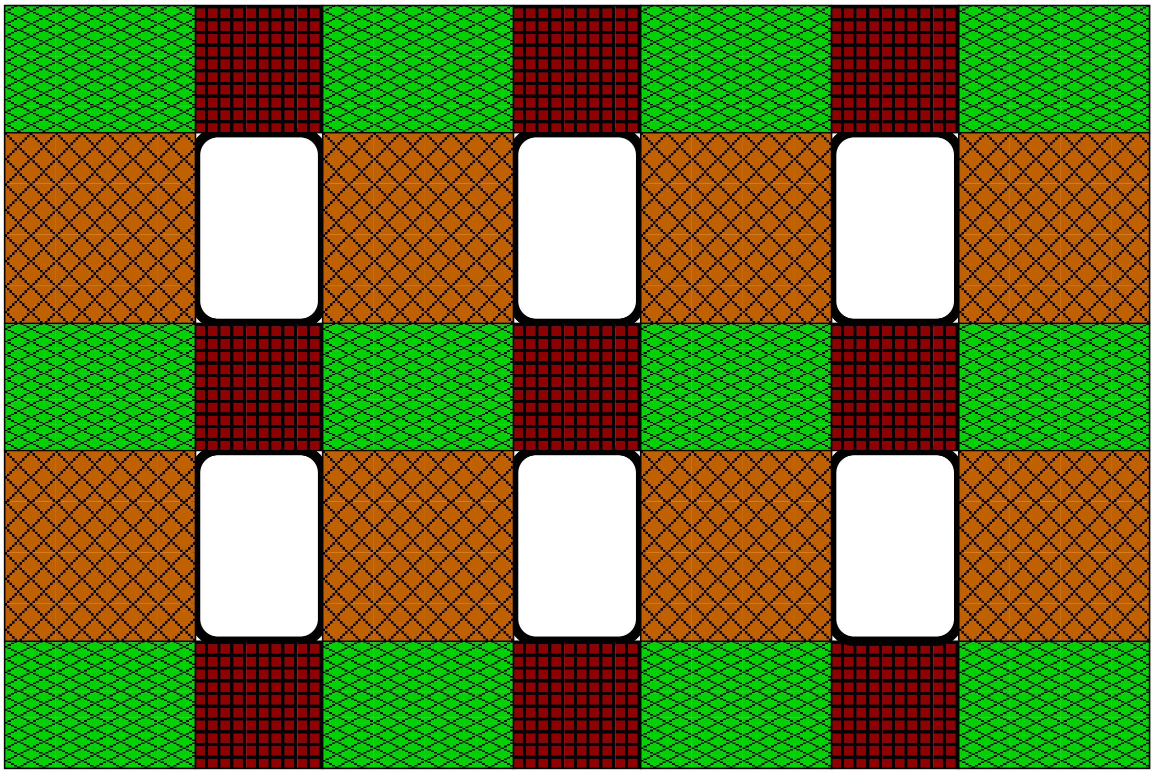

Figure 1: Heterogeneous membrane constructed from patches. Colors and patterns represent rectangular patches of differing architectures, including *pores*, which are patches with open interiors. This figure shows how complex, heterogeneous membranes may be synthesized hierarchically and iteratively from simpler, homogeneous rectangular patches.

assembling such a structure is by combining rectangular patches, much as in a patchwork quilt (Fig. 1). To facilitate this we have defined a uniform interface for such patches.

## 1.1 Definition

A combinatory program $P$ constructs an $m \times n$ patch if it has the following *patch synthesizer* "signature" $(m, n \geq 1)$:

$$PFY_1 \cdots Y_n X_1 \cdots X_m \Longrightarrow FV_1 \cdots V_m U_1 \cdots U_n. \tag{1}$$

$F$ is any combinatory operator (especially another patch synthesizer). $X_1, \ldots, X_m$ will be horizontal connections from the patch to the right (or to terminal groups if this is the right-most patch); similarly $Y_1, \ldots, Y_n$ are vertical connections to the patch below or to terminal groups. Whatever rectangular structure is created by $P$ (e.g., a cross-linked or hexagonal grid), $V_1, \cdots V_m$ represents the horizontal connections from the patch on its left side, and $U_1, \ldots, U_n$, the vertical connections from the patch above it. The $V_k$ and $U_j$ connections are passed to $F$, which is typically another patch synthesizer. See Fig. 2.

If this patch is the top-left-most of the membrane, so that additional additional structure will not be attached to the $V_k$ and $U_j$, then $F$ can be any inert group $\mathsf{N}$. This will leave a permanent border around the left and top margins. Similarly, if this is the bottom- and/or right-most patch, then $Y_j$ and/or $X_k$ will be terminal groups. The replicating and sharing varieties of $\hat{\mathsf{W}}$ can be used to link to identical or shared terminal groups $X$ and $Y$, as described in an earlier report (Sec. 1.2 [Mac02a]). For example, to share a single $Y$ on the lower margin and a single $X$ on the right margin, use $\check{\mathsf{W}}^{m-1}(\check{\mathsf{W}}^{n-1}(PF)Y)X$, since

$$\begin{aligned} \check{\mathsf{W}}^{m-1}(\check{\mathsf{W}}^{n-1}(PF)Y)X &\Longrightarrow \check{\mathsf{W}}^{n-1}(PF)YX^{(m-1)} \cdots X^{(0)} \\ &\Longrightarrow PFY^{(n-1)} \cdots Y^{(0)} X^{(m-1)} \cdots X^{(0)}. \end{aligned}$$

To have multiple copies, simply use $\mathsf{W}$ instead of $\check{\mathsf{W}}$.

## 1.2 Joining Patches

Any such patch synthesizer may be joined either horizontally or vertically with another patch synthesizer of compatible dimensions, to yield a patch synthesizer combining the two (Fig. 3). For example, if $P$ is an $m \times n$ patch synthesizer and $Q$ is an $m \times n'$ patch synthesizer, then $\mathsf{B}^{n+1}QP$ is an $m \times (n + n')$ patch synthesizer with $Q$ to the right of $P$. To see this, suppose

$$PFY_1 \cdots Y_n X_1 \cdots X_m \Longrightarrow FV_1 \cdots V_m U_1 \cdots U_n, \tag{2}$$
$$QFY'_1 \cdots Y'_{n'} X'_1 \cdots X'_m \Longrightarrow FV'_1 \cdots V'_m U'_1 \cdots U'_{n'}. \tag{3}$$

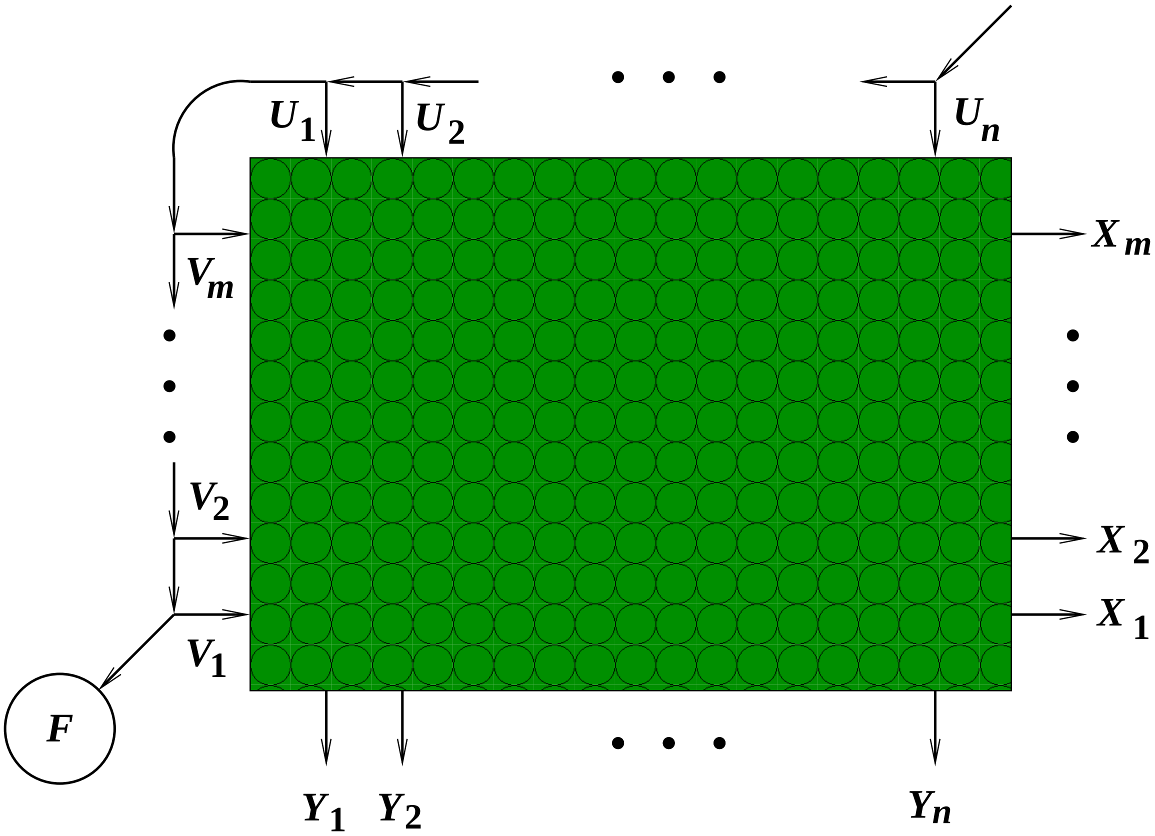


Figure 2: Patch synthesizer format. The $Y_j$ represent connections to the next patch below, or to terminal groups if this patch is bottom-most in a membrane. Similarly, the $X_k$ represent connections to the next patch to the right, or to terminal groups if this patch is right-most in a membrane. The $U_j$ and $V_k$ represent connection into the top and left borders, respectively, from other patches. The $U_j$ and $V_k$ are parameters passed to $F$, which may use them in the synthesis of other patches to the left of this patch or above it.

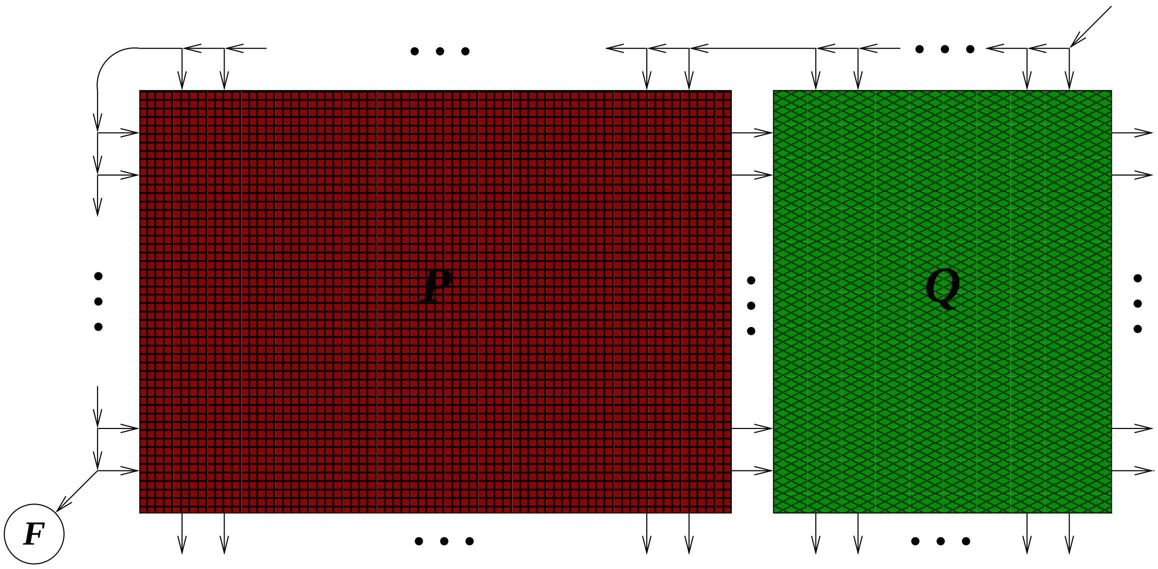


Figure 3: Horizontal join of two patches. In this figure $P$ and $Q$ denote the patches constructed by two patch synthesizers (which also may be called $P$ and $Q$). After patch $Q$ is synthesized, patch $P$ is constructed on the connections into the left-hand border of $Q$. $F$ is passed connections into the left and top borders of the $(P, Q)$ composite patch so that it can construct additional patches on either of these edges. Thus the result of the horizontal join, $\mathsf{joinh}_n QP$, is a composite patch synthesizer, fitting the patch synthesizer format (Fig. 2), which can enter into larger composite patches.

To have the resulting patches connected horizontally, with $P$ to the left of $Q$, we set $V'_k = X_k$, $k = 1, \ldots, m$. Now let $R = \mathsf{B}^{n+1}QP$ and observe:

$$\begin{aligned} RFY_1 \cdots Y_n Y'_1 \cdots Y'_{n'} X'_1 \cdots X'_m &\Longrightarrow \mathsf{B}^{n+1}QPFY_1 \cdots Y_n Y'_1 \cdots Y'_{n'} X'_1 \cdots X'_m \\ &\Longrightarrow Q(PFY_1 \cdots Y_n) Y'_1 \cdots Y'_{n'} X'_1 \cdots X'_m \\ &\Longrightarrow PFY_1 \cdots Y_n V'_1 \cdots V'_m U'_1 \cdots U'_{n'} \\ &= PFY_1 \cdots Y_n X_1 \cdots X_m U'_1 \cdots U'_{n'} \\ &\Longrightarrow FV_1 \cdots V_m U_1 \cdots U_n U'_1 \cdots U'_{n'}. \end{aligned}$$

Therefore we have the following simple definition of $\mathsf{joinh}$ so that $\mathsf{joinh}_n QP$ is a synthesizer that joins $Q$ to the right of $P$ ($n$ wide):

**Definition 1** (joinh)

$$\mathsf{joinh}_n = \mathsf{B}^{n+1}.$$

Similarly, if $P$ is $m \times n$ and $Q$ is $m' \times n$, then $P \circ \mathsf{B}^m Q$ is the $(m + m') \times n$ patch synthesizer with $P$ below $Q$ (Fig. 4). To see this, let

$$PFY_1 \cdots Y_n X_1 \cdots X_m \Longrightarrow FV_1 \cdots V_m U_1 \cdots U_n, \tag{4}$$

$$QFY'_1 \cdots Y'_n X'_1 \cdots X'_{m'} \Longrightarrow FV'_1 \cdots V'_{m'} U'_1 \cdots U'_n. \tag{5}$$

To attach $Q$ above $P$ we set $Y'_k = U_k$, $k = 1, \ldots, n$. Suppose $R = P \circ \mathsf{B}^m Q$ and observe:

$$\begin{aligned} RFY_1 \cdots Y_n X_1 \cdots X_m X'_1 \cdots X'_{m'} &\Longrightarrow (P \circ \mathsf{B}^m Q) FY_1 \cdots Y_n X_1 \cdots X_m X'_1 \cdots X'_{m'} \\ &\Longrightarrow P(\mathsf{B}^m QF) Y_1 \cdots Y_n X_1 \cdots X_m X'_1 \cdots X'_{m'} \\ &\Longrightarrow \mathsf{B}^m QFV_1 \cdots V_m U_1 \cdots U_n X'_1 \cdots X'_{m'} \\ &\Longrightarrow Q(FV_1 \cdots V_m) U_1 \cdots U_n X'_1 \cdots X'_{m'} \\ &= Q(FV_1 \cdots V_m) Y'_1 \cdots Y'_n X'_1 \cdots X'_{m'} \\ &\Longrightarrow FV_1 \cdots V_m V'_1 \cdots V'_{m'} U'_1 \cdots U'_n. \end{aligned}$$

Therefore we have the following definition of $\mathsf{joinv}$, which has the effect that $\mathsf{joinv}_m PQ$ is a synthesizer that joins $P$ ($m$ in height) below $Q$:

**Definition 2** (joinv)

$$\mathsf{joinv}_m = \mathsf{B}(\mathsf{CBB}^m)\mathsf{B}.$$

To see the correctness of this definition, observe:

$$\begin{aligned} \mathsf{joinv}_m PQ &\Longrightarrow \mathsf{B}(\mathsf{CBB}^m)\mathsf{B}PQ \\ &\Longrightarrow \mathsf{CBB}^m(\mathsf{B}P)Q \\ &\Longrightarrow \mathsf{B}(\mathsf{B}P)\mathsf{B}^m Q \\ &\Longrightarrow \mathsf{B}P(\mathsf{B}^m Q) \\ &\Longrightarrow P \circ \mathsf{B}^m Q. \end{aligned}$$

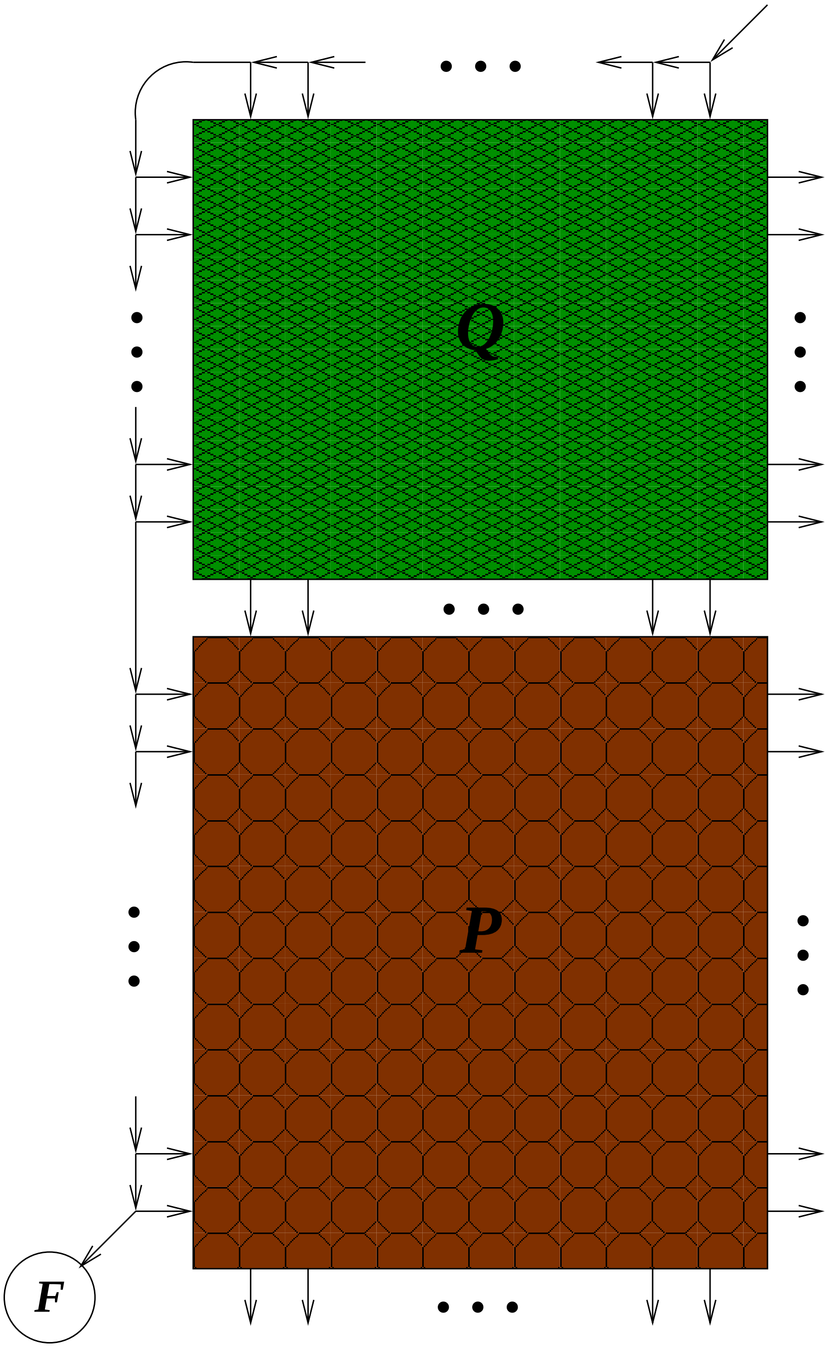


Figure 4: Vertical join of two patches. The upper patch $Q$ is constructed on the connections into the upper border of the lower patch $P$. The result, $\mathsf{joinv}_m PQ$, is in patch synthesizer format (Fig. 2) so that it can be used in the synthesis of larger composite patches.

Rectangular patch synthesizers can be iteratively assembled, both horizontally and vertically, to hierarchically construct large, heterogeneous membranes. For example, if $P$ is a $k \times l$ patch synthesizer, and $Q$ is a $k \times l'$ patch synthesizer, then $(\mathsf{joinh}_l P)^n Q$ appends $n$ copies of $P$ on the right of $Q$. Similarly, if $P$ is $k \times l$ and $Q$ is $k' \times l$, then $(\mathsf{joinv}_k P)^m Q$ places $m$ copies of $P$ below $Q$.

To illustrate, suppose $P$ and $Q$ both synthesize $k \times k$ patches of different architectures (e.g., one could be a membrane, the other a pore). Since $\mathsf{joinh}_k QP$ appends $Q$ to the right of $P$, clearly, then, $\mathsf{joinv}_k(\mathsf{joinh}_k PQ)(\mathsf{joinh}_k QP)$ is a synthesizer for a $2 \times 2$ checkerboard, which we may denote $S_2 = \begin{pmatrix} P & Q \\ Q & P \end{pmatrix}$. Continuing, $R_n = (\mathsf{joinh}_{2k} S_2)^{n-1} S_2$ is a row, $n$ in length, of such $2 \times 2$ squares. Finally, $(\mathsf{joinv}_{2k} R_n)^{n-1} R_n$ stacks $n$ of these double rows, producing a checkerboard, $2n$ on a side, of alternating $P$ and $Q$ (each $k \times k$).

Based on the foregoing examples, we may define two convenient operators for the iterative assembly of membranes. As we have seen, if $S$ is a $k \times l$ membrane, then $(\mathsf{joinh}_l S)^{n-1} S$ will make a horizontal row of $n$ copies of $S$. The size of this program is $\mathcal{O}(n|S|)$, but it can be reduced to $\mathcal{O}(n + |S|)$ as follows:

$$\begin{aligned} (\mathsf{joinh}_l S)^{n-1} S &\Longleftarrow \mathsf{Z}_{n-1}(\mathsf{joinh}_l S) S \\ &\Longleftarrow \mathsf{B}\mathsf{Z}_{n-1}\mathsf{joinh}_l SS \\ &\Longleftarrow \mathsf{W}_{(2)}\mathsf{B}\mathsf{Z}_{n-1}\mathsf{joinh}_l S. \end{aligned}$$

Therefore we define $\mathsf{iterh}$ so that $\mathsf{iterh}_{l,n} S$, which makes a horizontal row of $n$ copies of the $l$-wide membrane $S$:

**Definition 3** (iterh)

$$\mathsf{iterh}_{l,n} = \mathsf{W}_{(2)}\mathsf{B}\mathsf{Z}_{n-1}\mathsf{joinh}_l.$$

Similarly, we may define $\mathsf{iterv}$ so that $\mathsf{iterv}_{k,m} S$, which makes a vertical stack of $m$ copies of the $k$-high membrane $S$:

**Definition 4** (iterv)

$$\mathsf{iterv}_{k,m} = \mathsf{W}_{(2)}\mathsf{B}\mathsf{Z}_{m-1}\mathsf{joinv}_k.$$

With these operators, our checkerboard example may be expressed: $\mathsf{iterv}_{2k,n}(\mathsf{iterh}_{2k,n} S_2)$.

In this way we can build upon a basic library of elementary membrane patches, pores, and other nanostructural units in order to construct large, heterogeneous membranes. Some of these elementary patches will be discussed in Sec. 1.4 and the following sections.

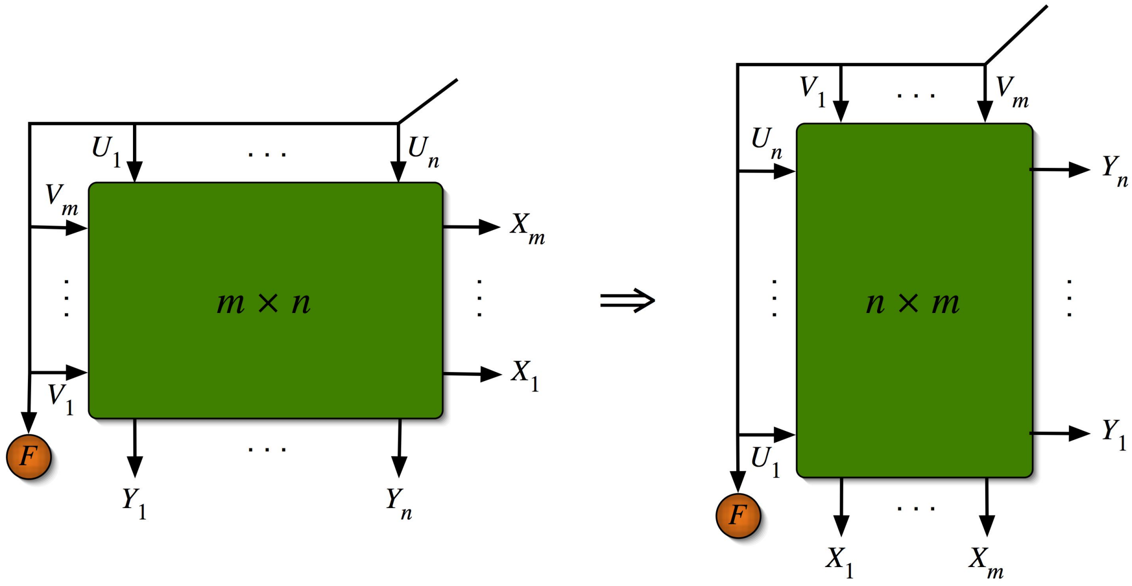


Figure 5: Flip-reversal of a patch synthesizer. An $m \times n$ patch is reversed and flipped to yield an $n \times m$ patch by $\mathsf{flipr}_{m,n}$. The assembled structure will be correspondingly flip-reversed.

## 1.3 Patch Manipulation

Patches may be manipulated in various ways by using combinators to rearrange their input and output connections. For example, Fig. 5 illustrates a "flip-reversal" of a patch synthesizer, accomplished by flipr. If $m \times n$ patch $P$ has the signature

$$PFY_1 \cdots Y_n X_1 \cdots X_m \Longrightarrow FV_1 \cdots V_m U_1 \cdots U_n, \tag{6}$$

then we want $\mathsf{flipr}_{m,n} P$ to be an $n \times m$ patch with the effect

$$\mathsf{flipr}_{m,n} P X_1 \cdots X_m Y_1 \cdots Y_n \Longrightarrow FU_1 \cdots U_n V_1 \cdots V_m,$$

with the $U_j$ and $X_k$ defined as in the original patch (Eq. 6). We work backward as follows, making use of the fact (Sec. 2 [Mac02b]) that $(\mathsf{C}_{[m+n-1]})^k$ performs a right rotation of $m+n$ arguments by $k$ positions:

$$\begin{aligned}
\mathsf{flipr}_{m,n} PFX_1 \cdots X_m Y_1 \cdots Y_n &\Longrightarrow FU_1 \cdots U_n V_1 \cdots V_m \quad \text{(desired)} \\
&\Longleftarrow \mathsf{C}^n_{[m+n-1]} FV_1 \cdots V_m U_1 \cdots U_n \\
&\Longleftarrow P(\mathsf{C}^n_{[m+n-1]} F) Y_1 \cdots Y_n X_1 \cdots X_m \\
&\Longleftarrow \mathsf{C}^n_{[m+n-1]}(P(\mathsf{C}^n_{[m+n-1]} F)) X_1 \cdots X_m Y_1 \cdots Y_n.
\end{aligned}$$

Thus we have determined $\mathsf{flipr}_{m,n} PF = \mathsf{C}^n_{[m+n-1]}(P(\mathsf{C}^n_{[m+n-1]} F))$. To factor $P$ and $F$ out of the definition we continue working backward.

$$\mathsf{flipr}_{m,n} PF \quad = \quad \mathsf{C}^n_{[m+n-1]}(P(\mathsf{C}^n_{[m+n-1]} F))$$

$$
\begin{aligned}
&\Longleftarrow \quad \mathsf{BC}^n_{[m+n-1]}P(\mathsf{C}^n_{[m+n-1]}F) \\
&\Longleftarrow \quad \mathsf{B}(\mathsf{BC}^n_{[m+n-1]}P)\mathsf{C}^n_{[m+n-1]}F \\
&\Longleftarrow \quad \mathsf{CBC}^n_{[m+n-1]}(\mathsf{BC}^n_{[m+n-1]}P)F \\
&\Longleftarrow \quad (\mathsf{CBC}^n_{[m+n-1]} \circ \mathsf{BC}^n_{[m+n-1]})PF.
\end{aligned}
$$

Hence we have

$$\mathsf{flipr}_{m,n} = \mathsf{CBC}^n_{[m+n-1]} \circ \mathsf{BC}^n_{[m+n-1]}.$$

Since $|\mathsf{C}^n_{[m+n-1]}| \in \mathcal{O}[n(m+n)]$ we may prefer to use $\mathsf{Z}_n\mathsf{C}_{[m+n-1]}$, which is $\mathcal{O}(m+n)$, as in the following

**Definition 5** (flipr)

$$\mathsf{flipr}_{m,n} = \mathsf{CB}(\mathsf{Z}_n\mathsf{C}_{[m+n-1]}) \circ \mathsf{B}(\mathsf{Z}_n\mathsf{C}_{[m+n-1]}).$$

Another useful patch manipulation is to reverse the order of some or all of the connections to the patch. Before considering such patch manipulations we address the general problem of reversing a function's arguments. Therefore we will define rev so that $\mathsf{rev}_n F X_n X_{n-1} \cdots X_2 X_1 = F X_1 X_2 \cdots X_{n-1} X_n$. We present the definition and then show its correctness.

**Definition 6** (rev)

$$
\begin{aligned}
\mathsf{rev}_1 &= \mathsf{C}^{[0]} = \mathsf{I}, \\
\mathsf{rev}_{n+1} &= \mathsf{C}^{[n]} \circ \mathsf{rev}_n, \quad n \geq 1.
\end{aligned}
$$

That is, $\mathsf{rev}_n = \mathsf{C}^{[n-1]} \circ \mathsf{C}^{[n-2]} \circ \cdots \circ \mathsf{C}^{[0]}$. Unfortunately $|\mathsf{rev}_n| \in \mathcal{O}(n^2)$.

**Theorem 1** *For $n \geq 1$,*

$$\mathsf{rev}_n F X_n X_{n-1} \cdots X_2 X_1 = F X_1 X_2 \cdots X_{n-1} X_n.$$

**Proof:** The proof is by induction, and the $n = 1$ case is obvious. Therefore, for $n \geq 1$,

$$
\begin{aligned}
\mathsf{rev}_{n+1} F X_{n+1} X_n \cdots X_1 &\Longrightarrow (\mathsf{C}^{[n]} \circ \mathsf{rev}_n) F X_{n+1} X_n \cdots X_1 \\
&\Longrightarrow \mathsf{C}^{[n]}(\mathsf{rev}_n F) X_{n+1} X_n \cdots X_1 \\
&\Longrightarrow \mathsf{rev}_n F X_n \cdots X_1 X_{n+1} \\
&\Longrightarrow F X_1 \cdots X_n X_{n+1}.
\end{aligned}
$$

■

We could have equally well defined $\mathsf{rev}_n = \mathsf{C}_{[0]} \circ \mathsf{C}_{[1]} \circ \cdots \circ \mathsf{C}_{[n-1]}$.

Now if patch synthesizer $P$ has the usual signature $(PFY_1 \cdots Y_n X_1 \cdots X_m \Longrightarrow FV_1 \cdots V_m U_1 \cdots U_n)$, then $\mathsf{rev}_n \circ P$ reverses the order of the connections along the lower border:

$$\begin{aligned}
(\mathsf{rev}_n \circ P) F Y_n \cdots Y_1 X_1 \cdots X_m &\Longrightarrow \mathsf{rev}_n (PF) Y_n \cdots Y_1 X_1 \cdots X_m \\
&\Longrightarrow PFY_1 \cdots Y_n X_1 \cdots X_m \\
&\Longrightarrow FV_1 \cdots V_m U_1 \cdots U_n.
\end{aligned}$$

Since, typically, each $Y_j$ is connected to the corresponding $U_j$ in a patch, such a reversal may lead to many links crossing in the membrane, constricting the structure, so reversal of the $Y_j$ is perhaps more useful if combined with a reversal of the $U_j$ connections. The latter is accomplished by $P \circ \mathsf{B}^m \mathsf{rev}_n$; to see this, observe:

$$\begin{aligned}
(P \circ \mathsf{B}^m \mathsf{rev}_n) F Y_1 \cdots Y_n X_1 \cdots X_m &\Longrightarrow P(\mathsf{B}^m \mathsf{rev}_n F) Y_1 \cdots Y_n X_1 \cdots X_m \\
&\Longrightarrow \mathsf{B}^m \mathsf{rev}_n F V_1 \cdots V_m U_1 \cdots U_n \\
&\Longrightarrow FV_1 \cdots V_m U_n \cdots U_1.
\end{aligned}$$

The latter is because $\mathsf{B}^m G = G_{(m)}$ defers the application of a regular combinator $G$ by $m$ argument positions; hence, $\mathsf{B}^m \mathsf{rev}_n$ skips the $V_1 \cdots V_m$ before reversing. Therefore, to reverse both the $Y_j$ and $U_j$ connections, we may use $\mathsf{rev}_n \circ P \circ \mathsf{B}^m \mathsf{rev}_n$:

$$\begin{aligned}
(\mathsf{rev}_n \circ P \circ \mathsf{B}^m \mathsf{rev}_n) F Y_n \cdots Y_1 X_1 \cdots X_m &\Longrightarrow P(\mathsf{B}^m \mathsf{rev}_n F) Y_1 \cdots Y_n X_1 \cdots X_m \\
&\Longrightarrow \mathsf{B}^m \mathsf{rev}_n F V_1 \cdots V_m U_1 \cdots U_n \\
&\Longrightarrow FV_1 \cdots V_m U_n \cdots U_1.
\end{aligned}$$

Thus we may define a "horizontal flip" operation:

**Definition 7** ($\mathsf{fliph}$)

$$\mathsf{fliph}_{m,n} = \mathsf{Brev}_n \circ \mathsf{CB}(\mathsf{B}^m \mathsf{rev}_n).$$

To see the correctness of this definition, observe:

$$\begin{aligned}
(\mathsf{Brev}_n \circ \mathsf{CB}(\mathsf{B}^m \mathsf{rev}_n)) P &\Longrightarrow \mathsf{Brev}_n (\mathsf{CB}(\mathsf{B}^m \mathsf{rev}_n) P) \\
&\Longrightarrow \mathsf{rev}_n \circ (\mathsf{CB}(\mathsf{B}^m \mathsf{rev}_n) P) \\
&\Longrightarrow \mathsf{rev}_n \circ (\mathsf{B} P (\mathsf{B}^m \mathsf{rev}_n)) \\
&\Longrightarrow \mathsf{rev}_n \circ (P \circ \mathsf{B}^m \mathsf{rev}_n).
\end{aligned}$$

[In general, $F \circ P \circ G$ can be written $(\mathsf{B}F \circ \mathsf{CB}G)P$ or $(\mathsf{CB}G \circ \mathsf{B}F)P$.]

By analogous reasoning, we can see that $\mathsf{B}^n \mathsf{rev}_m \circ P$ reverses the $X_k$ connections:

$$(\mathsf{B}^n \mathsf{rev}_m \circ P) F Y_1 \cdots Y_n X_m \cdots X_1 \Longrightarrow FV_1 \cdots V_m U_1 \cdots U_n.$$

The $V_k$ connections are reversed by $P \circ \mathsf{rev}_m$:

$$(P \circ \mathsf{rev}_m) Y_1 \cdots Y_n X_1 \cdots X_m \Longrightarrow FV_m \cdots V_1 U_1 \cdots U_n.$$

Obviously, $\mathsf{B}^n \mathsf{rev}_m \circ P \circ \mathsf{rev}_m$ reverses both the $X_k$ and $V_k$ connections. Hence we have

**Definition 8** ($\mathsf{flipv}$)

$$\mathsf{flipv}_{m,n} = \mathsf{B}(\mathsf{B}^n\mathsf{rev}_m) \circ \mathsf{CBrev}_m.$$

To reverse *all* the connections, use $\mathsf{flipv}_{m,n} \circ \mathsf{fliph}_{m,n}$ (which commute):

$$(\mathsf{flipv}_{m,n} \circ \mathsf{fliph}_{m,n})FY_n \cdots Y_1 X_m \cdots X_1 \Longrightarrow FV_m \cdots V_1 U_n \cdots U_1$$

Note that this operation is different from $\mathsf{rev}_{m+n} \circ P \circ \mathsf{rev}_{m+n}$, which flips the patch across its main diagonal (making it $n \times m$):

$$\begin{aligned}(\mathsf{rev}_{m+n} \circ P \circ \mathsf{rev}_{m+n})FX_m \cdots X_1 Y_n \cdots Y_1 &\Longrightarrow P(\mathsf{rev}_{m+n}F)Y_1 \cdots Y_n X_1 \cdots X_m \\ &\Longrightarrow \mathsf{rev}_{m+n}FV_1 \cdots V_m U_1 \cdots U_n \\ &\Longrightarrow FU_n \cdots U_1 V_m \cdots V_1.\end{aligned}$$

This is also useful, and so we give it a name:

**Definition 9** ($\mathsf{flipd}$)

$$\mathsf{flipd}_{m,n} = \mathsf{Brev}_{m+n} \circ \mathsf{CBrev}_{m+n}.$$

The $\mathsf{flipr}$ operation is equivalent to reversing the horizontal and vertical connections and flipping the patch diagonally. That is,

$$\mathsf{flipr}_{m,n} = \mathsf{fliph}_{m,n} \circ \mathsf{flipv}_{m,n} \circ \mathsf{flipd}_{m,n},$$

although this would be an exceptionally inefficient way to compute it! Obviously, all of these reversals and flips commute in simple ways.

## 1.4 Particular Membranes in Patch Format

### 1.4.1 Cross-linked Membrane

It is especially easy to construct a cross-linked membrane in patch format; the patch synthesizer for an $m \times n$ membrane is simply $\check{\mathsf{S}}_n^{[m]}$. To show this, we will use the notation of Sec. 1 [Mac02a], and prove

**Theorem 2**

$$\check{\mathsf{S}}_n^{[m]}FY_1 \cdots Y_n X_1 \cdots X_m \Longrightarrow FX_1^{(n)} \cdots X_m^{(n)}(Y_1 X_1^{(n-1)} \cdots X_m^{(n-1)}) \cdots (Y_n X_1^{(0)} \cdots X_m^{(0)}).$$

**Proof:** The proof is by induction on $m$. For $m = 0$,

$$\check{\mathsf{S}}_n^{[0]}FY_1 \cdots Y_n \Longrightarrow \mathsf{I}FY_1 \cdots Y_n \Longrightarrow FY_1 \cdots Y_n.$$

For $m \geq 0$,

$$\begin{aligned}
\check{\mathsf{S}}_n^{[m+1]} F Y_1 \cdots Y_n X_1 \cdots X_{m+1} &\Longrightarrow (\check{\mathsf{S}}_n \circ \mathsf{B}\check{\mathsf{S}}_n^{[m]}) F Y_1 \cdots Y_n X_1 \cdots X_{m+1} \\
&\Longrightarrow \check{\mathsf{S}}_n (\mathsf{B}\check{\mathsf{S}}_n^{[m]} F) Y_1 \cdots Y_n X_1 X_2 \cdots X_{m+1} \\
&\Longrightarrow \mathsf{B}\check{\mathsf{S}}_n^{[m]} F X_1^{(n)} (Y_1 X_1^{(n-1)}) \cdots (Y_n X_1^{(0)}) X_2 \cdots X_{m+1} \\
&\Longrightarrow \check{\mathsf{S}}_n^{[m]} (F X_1^{(n)}) (Y_1 X_1^{(n-1)}) \cdots (Y_n X_1^{(0)}) X_2 \cdots X_{m+1} \\
&\Longrightarrow F X_1^{(n)} X_2^{(n)} \cdots X_{m+1}^{(n)} (Y_1 X_1^{(n-1)} X_2^{(n-1)} \cdots X_{m+1}^{(n-1)}) \\
&\qquad \cdots (Y_n X_1^{(0)} X_2^{(0)} \cdots X_{m+1}^{(0)}).
\end{aligned}$$

■

The size of the cross-linked patch synthesizer is (Sec. 16 [Mac02c]):

$$|\check{\mathsf{S}}_n^{[m]}| = 18mn - 17, \text{ for } m, n > 0.$$

This program is $\mathcal{O}(mn)$, but, much as we did in Sec. 1.3 [Mac02a], we can write

$$\check{\mathsf{S}}_n^{[m]} = \check{\Phi}_{n+1}^m \mathsf{I} = \mathsf{Z}_m \check{\Phi}_{n+1} \mathsf{I}, \tag{7}$$

which has size in $\mathcal{O}(m+n)$. Therefore we can define the cross-linked membrane patch synthesizer:

**Definition 10** (xpatch)

$$\mathsf{xpatch}_{m,n} = \mathsf{Z}_m \check{\Phi}_{n+1} \mathsf{I}.$$

### 1.4.2 Hexagonal Membrane

A patch synthesizer for hexagonal membranes could be designed along the same lines as the original hexagonal membrane (Sec. 3 [Mac02a]). However, there is a simpler approach, for we can define a single "double row" in patch synthesizer format, and iterate it to construct large hexagonal membranes; this illustrates the usefulness of patches. We will define $\mathsf{phrow}_n$ to synthesize a hexagonal double row of width $n$ in patch format, as shown in Fig. 6. In terms of the notation of this figure, we want

$$\mathsf{phrow}_n F Y_1 \cdots Y_n X \Longrightarrow F V U_1 \cdots U_n.$$

To derive the definition, we use a convenient means of creating combinatory functions, which depends on a property of *regular* combinators [Mac02c, p. 6]. If $F_1, \ldots, F_N$ are regular, then the composition $F_1 \circ F_2 \circ \cdots F_N$ will sequentially manipulate the argument list according to the operators $F_1, F_2, \ldots, F_N$, *in that order* (Sec. 2 [Mac02c]). Therefore, we can start with the given input arguments, $F Y_1 \cdots Y_n X$, and then choose combinatory operators to transform it into the required output $F V U_1 \cdots U_n$.

Here is the derivation. We begin with the inputs to the patch synthesizer:

$$F Y_1 Y_2 \cdots Y_{n-1} Y_n X.$$

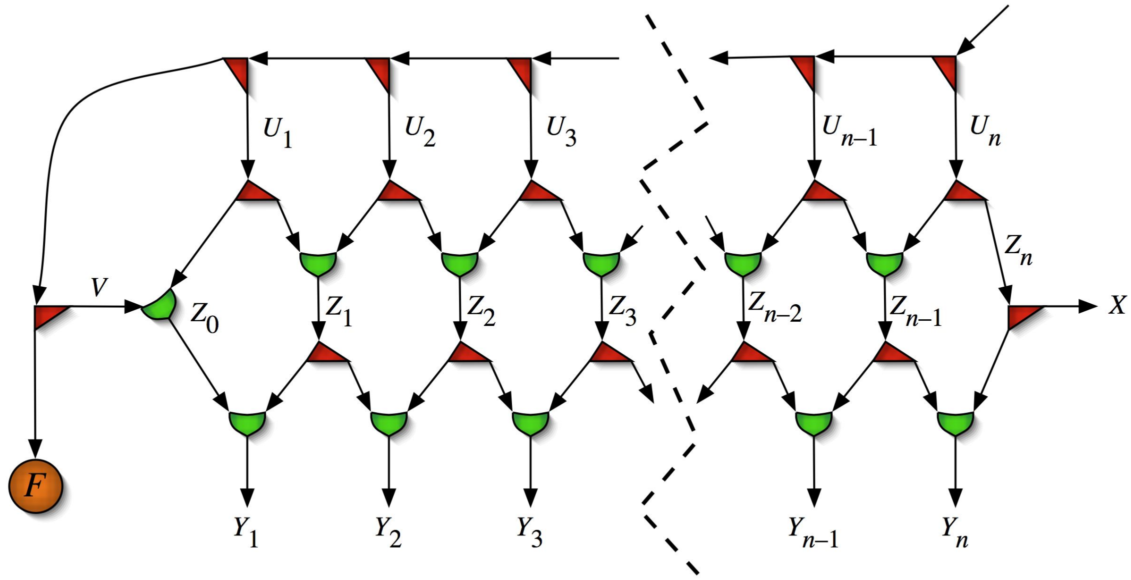


Figure 6: Double row of hexagonal membrane in patch synthesizer format. Red triangles are $\mathsf{A}$ (application) nodes (short leg = operator, long leg = operand), and green cup-shapes are $\mathsf{V}$ (sharing) nodes.

Referring to Fig. 6 to see where we are going, we apply $\check{\mathsf{W}}_{[n]}$ to share the $Y_j$:

$$FY_1'Y_1Y_2'Y_2\cdots Y_{n-1}'Y_{n-1}Y_n'Y_nX.$$

The alternating pairs after $FY_1'$ are grouped into applications by $\mathsf{BB}^{[n]}$:

$$FY_1'(Y_1Y_2')\cdots(Y_{n-1}Y_n')(Y_nX).$$

For convenience, we rewrite this structure using the $Z_j$ defined in the figure ($Z_0 = Y_1'$, $Z_n = Y_nX$, $Z_j = Y_jY_{j+1}'$, $j = 1, \ldots, n-1$):

$$FZ_0Z_1\cdots Z_{n-1}Z_n.$$

Again we use $\check{\mathsf{W}}_{[n]}$ to share the first $n$ arguments:

$$FZ_0'Z_0Z_1'Z_1\cdots Z_{n-1}'Z_{n-1}Z_n.$$

Since $V = Z_0'$ we may rewrite this:

$$FVZ_0Z_1'Z_1\cdots Z_{n-1}'Z_{n-1}Z_n.$$

Alternating pairs after $FV$ are again combined into applications by $\mathsf{BB}^{[n]}$:

$$FV(Z_0Z_1')(Z_1Z_2')\cdots(Z_{n-2}Z_{n-1}')(Z_{n-1}Z_n).$$

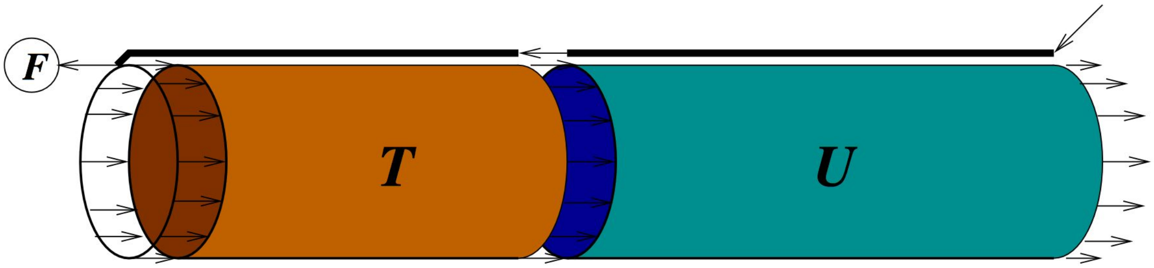


Figure 7: Joining tubes in patch format. If $T$ and $U$ are patch synthesizers for nanotubes of the same circumference, then $U \circ T$ is a patch synthesizer for the concatenation shown in the figure. This operation may be iterated to combine repeating units into nanotubes of arbitrary length.

Now, since $U_j = (Z_{j-1}Z'_j)$, $j = 1, \ldots, n$, we see that we have the desired result:

$$FVU_1U_2 \cdots U_{n-1}U_n.$$

Putting all these steps together, we conclude that the required transformation is achieved by $\check{\mathsf{W}}_{[n]} \circ \mathsf{BB}^{[n]} \circ \check{\mathsf{W}}_{[n]} \circ \mathsf{BB}^{[n]}$. This leads us to

**Definition 11** (phrow)

$$\mathsf{phrow}_n = (\check{\mathsf{W}}_{[n]} \circ \mathsf{BB}^{[n]})^2.$$

It is then simple to iterate this row to create a patch synthesizer for an $m \times n$ hexagonal membrane:

**Definition 12** (hpatch)

$$\mathsf{hpatch}_{m,n} = \mathsf{iterv}_{1,m}\mathsf{phrow}_n.$$

It would have been possible to construct the cross-linked membrane patch synthesizer xpatch by a similar iteration of a single cross-linked row, but Def. 10 is already sufficiently simple $(\mathsf{Z}_m \check{\Phi}_{n+1} \mathsf{I})$.

## 2 Nanotube Patches

Nanotubes can also be synthesized in patch format to allow end-to-end connection (Fig. 7). To accomplish this we first define $\mathsf{ptubep}_{m,n}$ to synthesize a patchable cross-linked tube of circumference $m$ and length $n$. That is, we require

$$\begin{aligned} \mathsf{ptubep}_{m,n} F X_1 \cdots X_m \quad \Longrightarrow \quad & F X_1^{(n)} \cdots X_m^{(n)} y_1^{(1)} \cdots y_n^{(1)} \\ & \text{where } y_k \equiv (y_k^{(0)} X_1^{(n-k)} \cdots X_m^{(n-k)}). \end{aligned}$$

This may be accomplished in much the same way as in the original cross-linked nanotube (Sec. 2 [Mac02a]) by using the same auxiliary function $G$ defined there:

$$G = \mathsf{B}^m \check{\mathsf{Y}}(\mathsf{C}_{[m]}\mathsf{I}). \tag{8}$$

Working backward,

$$\begin{aligned} \mathsf{ptubep}_{m,n} F X_1 \cdots X_m &\Longleftarrow \check{\mathsf{S}}_n^{[m]} F \overbrace{G \cdots G}^{n} X_1 \ldots X_m \\ &\Longleftarrow \mathsf{W}^{n-1}(\check{\mathsf{S}}_n^{[m]} F) G X_1 \ldots X_m. \end{aligned}$$

Therefore,

$$\begin{aligned} \mathsf{ptubep}_{m,n} F &= \mathsf{W}^{n-1}(\check{\mathsf{S}}_n^{[m]} F) G \\ &\Longleftarrow (\mathsf{W}^{n-1} \circ \check{\mathsf{S}}_n^{[m]}) F G \\ &\Longleftarrow \mathsf{C}(\mathsf{W}^{n-1} \circ \check{\mathsf{S}}_n^{[m]}) G F. \end{aligned}$$

Thus, substituting the definition of $G$ (Eq. 8), we have the following

**Definition 13** (ptubep)

$$\mathsf{ptubep}_{m,n} = \mathsf{C}(\mathsf{W}^{n-1} \circ \check{\mathsf{S}}_n^{[m]})(\mathsf{B}^m \check{\mathsf{Y}}(\mathsf{C}_{[m]}\mathsf{I})).$$

Unfortunately, $|\check{\mathsf{S}}_n^{[m]}| \in \mathcal{O}(mn)$, and so $|\mathsf{ptubep}_{m,n}| \in \mathcal{O}(mn)$, but we can apply Eq. 7 to get an $\mathcal{O}(m+n)$ program:

**Definition 14** (ptube)

$$\mathsf{ptube}_{m,n} = \mathsf{C}(\mathsf{W}^{n-1} \circ \mathsf{Z}_m \check{\Phi}_{n+1}\mathsf{I})(\mathsf{B}^m \check{\mathsf{Y}}(\mathsf{C}_{[m]}\mathsf{I})).$$

If $T$ is a patchable tube synthesizer of length $n$ and $U$ is one of length $n'$, both of the same circumference $m$, then $U \circ T$ is a patch synthesizer that appends $U$ to the right of $T$. To see this, suppose $T = \mathsf{ptube}_{m,n}$ and $U = \mathsf{ptube}_{m,n'}$. Observe:

$$\begin{aligned} (U \circ T) F X_1 \cdots X_m &\Longrightarrow U(TF) X_1 \cdots X_m \\ &\Longrightarrow TFX_1^{(n)} \cdots X_m^{(n)} y_1^{(1)} \cdots y_{n'}^{(1)} \\ &\qquad \text{where } y_k \equiv (y_k^{(0)} X_1^{(n-k)} \cdots X_m^{(n-k)}) \\ &\Longrightarrow FX_1^{(n+n')} \cdots X_m^{(n+n')} z_1^{(1)} \cdots z_n^{(1)} y_1^{(1)} \cdots y_{n'}^{(1)} \\ &\qquad \text{where } z_k \equiv (z_k^{(0)} X_1^{(n+n'-k)} \cdots X_m^{(n+n'-k)}). \end{aligned}$$

Hence $U \circ T$ appends $U$ to the right of $T$.

This operation is easily iterated, for $T^k$ is $k$ replicates of $T$ connected end-to-end (and thus of length $kn$). This operation can also be expressed $\mathsf{Z}_k T$. If, as in the case of ptube, the size of the synthesizer $T$ is $\mathcal{O}(m+n)$, then the size of $\mathsf{Z}_k T$ is $\mathcal{O}(k+m+n)$.

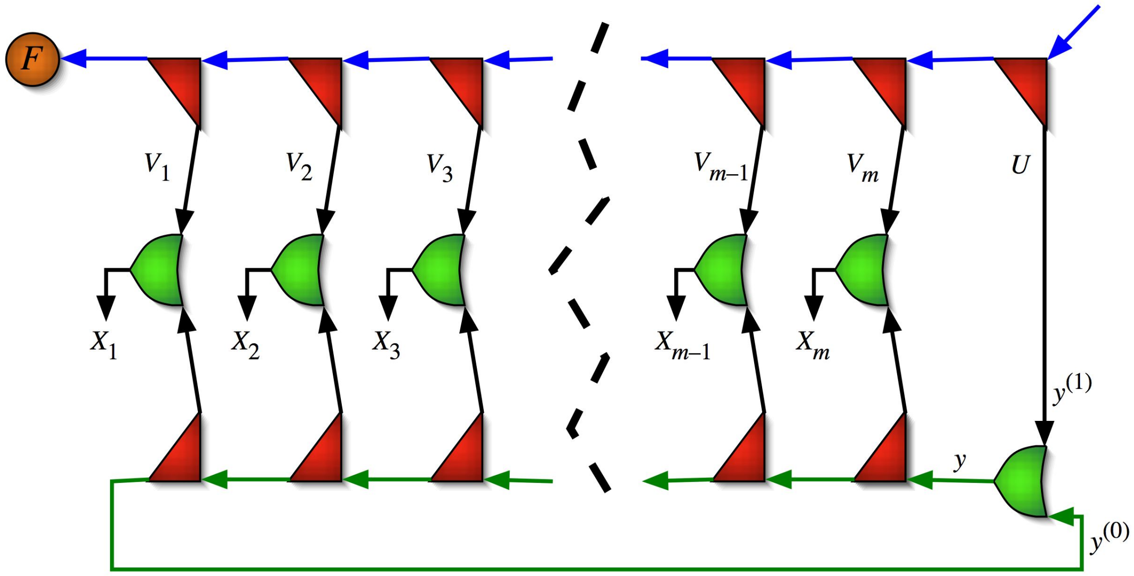


Figure 8: One rib (ring) of a cross-linked nanotube in patchable format. The figure has been linearized for the sake of clarity. The green arrows represent the rib (ring) itself. The $X_k$ are the links to the staves (linear chains) of the nanotube to the right of this rib. The $V_k$ are the connections to be passed to $F$ so that additional ribs may be synthesized on the left; the arguments to $F$ are shown by the blue arrows. $U$ links the rib to the "backbone" of the nanotube. As usual, red triangles are A (application) nodes (short leg = operator, long leg = operand), and green shapes are V (sharing) nodes. The structure of the cross-linked nanotube is described in an earlier report [Mac02a].

The iteration of patchable tubes suggests that nanotubes may be synthesized by iteration of a single rib (ring), as the hexagonal membrane was synthesized by iterating double rows (Sec. 1.4.2 above). As usual, we use $G$ (Eq. 8) to construct the rib:

$$GX_1^{(0)} \cdots X_m^{(0)} \Longrightarrow y^{(1)} \quad \text{where } y \equiv (y^{(0)} X_1^{(0)} \cdots X_m^{(0)}).$$

We will define $\mathsf{prib}_m$ as a patch synthesizer for a single rib of circumference $m$. Referring to Fig. 8, we can see that it should accomplish:

$$\mathsf{prib}_m F X_1 \cdots X_m \Longrightarrow F V_1 \cdots V_m U, \tag{9}$$

where $V_k = X_k^{(1)}$ and $U = y^{(1)}$. Working backwards,

$$\begin{aligned} FX_1^{(1)} \cdots X_m^{(1)} y^{(1)} &\Longleftarrow FX_1^{(1)} \cdots X_m^{(1)} (GX_1^{(0)} \cdots X_m^{(0)}) \\ &\Longleftarrow \check{\mathsf{S}}^{[m]} FGX_1 \cdots X_m. \end{aligned}$$

In the last line, we have made use of properties of $\check{\mathsf{S}}$ (Sec. 15 [Mac02c]). Thus we have determined that $\mathsf{prib}_m F = \check{\mathsf{S}}^{[m]} FG$. (This is related to the use of $\check{\mathsf{S}}_n^{[m]}$ to construct an $m \times n$ cross-linked membrane: Sec. 1.4.1 above.) Since $\mathsf{C}\check{\mathsf{S}}^{[m]} GF = \check{\mathsf{S}}^{[m]} FG$, we have, after substitution for $G$,

**Definition 15** ($\mathsf{prib}$)

$$\mathsf{prib}_m = \mathsf{C}\check{\mathsf{S}}^{[m]} (\mathsf{B}^m \check{\mathsf{Y}} (\mathsf{C}_{[m]} \mathsf{I})).$$

To construct a patchable nanotube of length $n$ we can now write $(\mathsf{prib}_m)^n$ or, since this is $\mathcal{O}(mn)$, we can use $\mathsf{Z}_n \mathsf{prib}_m$, which is only $\mathcal{O}(m+n)$.

Due to the directedness of the links used in combinatory molecular synthesis, nanotubes will have a *chirality* (a left- or right-handedness). (This can be seen clearly in Fig. 8 [Mac02a].) Therefore, it can be useful to be able to reverse the chirality of a nanotube patch. Since the staves (linear chains of $\mathsf{V}$ nodes) of the nanotube correspond to the horizonal chains ("woof") of a membrane patch, the chiral reversal of nanotube $T$ ($m$ in circumference) is accomplished by $\mathsf{flipv}_{m,0} T$. We illustrate this for the case of $\mathsf{prib}$ (Eq. 9):

$$\begin{aligned} \mathsf{flipv}_{m,0} \mathsf{prib}_m F X_m \cdots X_1 &\Longrightarrow (\mathsf{rev}_m \circ \mathsf{prib}_m \circ \mathsf{rev}_m) F X_m \cdots X_1 \\ &\Longrightarrow \mathsf{rev}_m (\mathsf{prib}_m (\mathsf{rev}_m F)) X_m \cdots X_1 \\ &\Longrightarrow \mathsf{prib}_m (\mathsf{rev}_m F) X_1 \cdots X_m \\ &\Longrightarrow \mathsf{rev}_m F V_1 \cdots V_m U \\ &\Longrightarrow F V_m \cdots V_1 U \end{aligned}$$

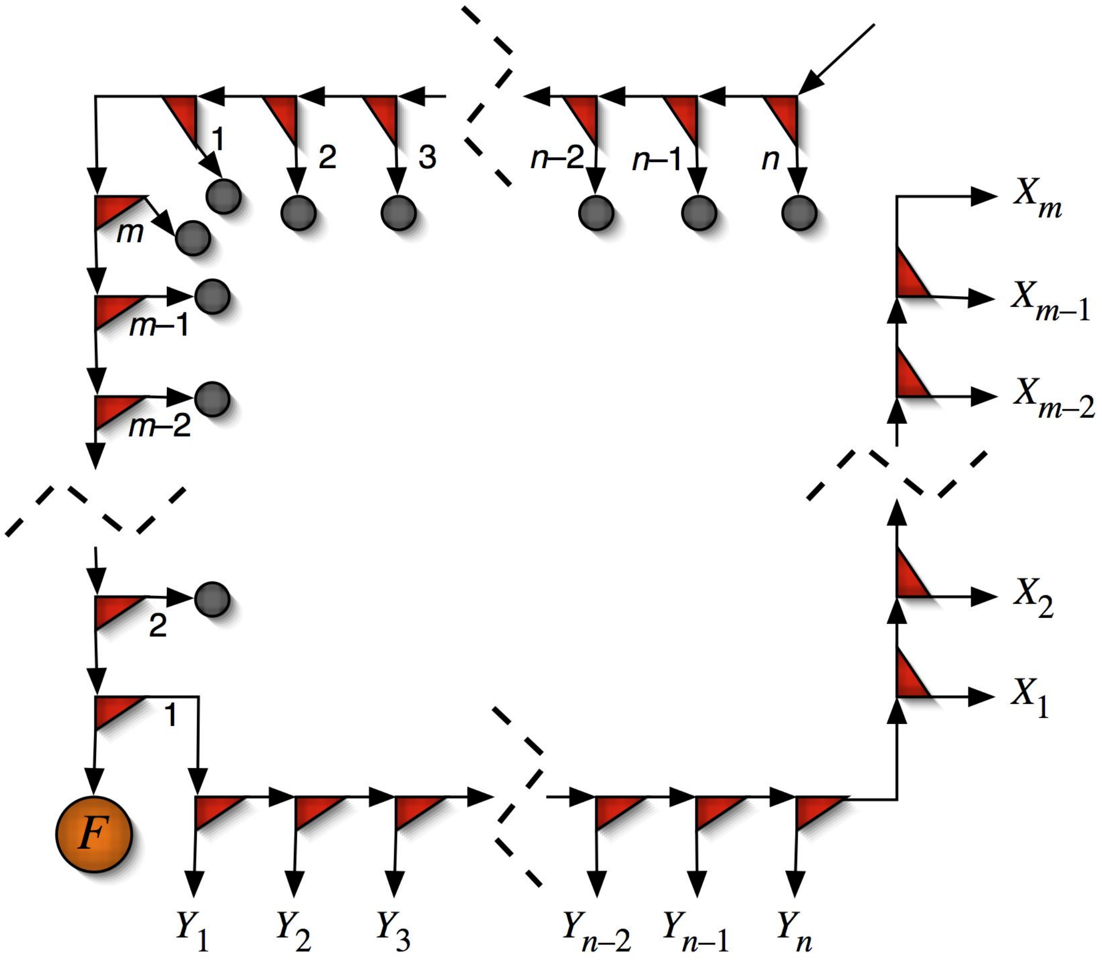


Figure 9: Simple pore in patch synthesizer format. This $m \times n$ pore is synthesized by $\mathsf{spore}_{m,n}$. The dark circles may be any desired terminal groups. Fig. 10 depicts such a pore synthesized in a cross-linked membrane.

# 3 Pores

A rectangular *pore* is simply a patch in which the interior is an open space. These pores can be combined with other patches to create membranes with pores of a given size and distribution (all in terms of the fundamental units, of course; see Sec. 6 below). Pores can be included in the surfaces of nanotubes as well.

Consider the simple pore shown in patch synthesizer format in Fig. 9. Its simple and repetitive structure makes it somewhat difficult to express accurately, so we will introduce some specialized notation for the purpose. Consider first the right-hand margin of the pore; it has the structure:

$$(X_1(X_2(X_3\cdots(X_{m-2}(X_{m-1}X_m))\cdots))).$$

It may be described more precisely by the following ad hoc notation:

$$\begin{aligned}\Xi^m_m &= X_m,\\ \Xi^m_k &= (X_k\Xi^m_{k+1}),\ \text{for } 1\le k<m.\end{aligned}$$

Now consider the bottom margin of the pore; it has the structure:

$$(F(Y_1(Y_2(Y_3\cdots(Y_{n-2}(Y_{n-1}(Y_n\Xi^m_1)))\cdots)))).$$

This may be described more precisely by use of the following ad hoc notation ($m,n\ge 1$):

$$\begin{aligned}\Upsilon^n_{n+1} &= \Xi^m_1,\\ \Upsilon^n_j &= (Y_j\Upsilon^n_{j+1}),\ \text{for } 1\le j\le n.\end{aligned}$$

The complete pore is, then,

$$(F\Upsilon^n_1\underbrace{\mathsf{NN}\cdots\mathsf{N}}_{m+n-1}).$$

Here $\mathsf{N}$ is any inert group used as terminators on the left-hand and upper margins of the pore. Any groups might be so used, and in addition it is a simple modification to use different groups for the left-hand and upper margins.

By means of our notation, we can express the intended operation of the pore synthesizer:

$$\mathsf{spore}_{m,n}FY_1\cdots Y_nX_1\cdots X_m\Longrightarrow F\Upsilon^n_1\underbrace{\mathsf{NN}\cdots\mathsf{N}}_{m+n-1}.$$

We can proceed as we did before (p. 13) by the stepwise application of regular operators to transform the argument structure. Given the input structure,

$$FY_1\cdots Y_nX_1\cdots X_m,$$

we use the left-reduction $\mathsf{B}_{[m+n-1]}$ (Sec. 2 [Mac02b]) to group the $Y_j$ and $X_k$ (of which there are $m+n$) to the right:

$$F(Y_1(Y_2 \cdots (Y_n(X_1(X_2 \cdots (X_{m-1}X_m) \cdots))) \cdots)) = F\Upsilon_1^n.$$

Next we must append the desired terminal group ($\mathsf{N}$ here), which we will do with a simple operator $A_{\mathsf{N}}$, to be defined shortly:

$$F\Upsilon_1^n\mathsf{N}.$$

We want $m+n-2$ additional copies of the terminal group, which is accomplished by $\mathsf{B}\hat{\mathsf{W}}^{m+n-2}$ (that is, $\hat{\mathsf{W}}^{m+n-2}$ deferred by one term), where $\hat{\mathsf{W}} = \mathsf{W}$ if we want separate copies of the terminal group (as shown in Fig. 9), which is the usual case, or $\hat{\mathsf{W}} = \check{\mathsf{W}}$ if we want one copy to be shared, which is less useful, since it will pinch up the pore. In either case we obtain:

$$F\Upsilon_1^n \underbrace{\mathsf{NN}\cdots\mathsf{N}}_{m+n-1},$$

which is the desired result of $\mathsf{spore}_{m,n}$. Thus, the synthesis of the pore is accomplished by $\mathsf{B}_{[m+n-1]} \circ A_{\mathsf{N}} \circ \mathsf{B}\hat{\mathsf{W}}^{m+n-2}$, where it remains to define $A_{\mathsf{N}}$. We work backward from its intended effect (i.e., $A_X$ appends $X$ after the two following arguments):

$$\begin{aligned} A_X F Y &\Longrightarrow FYX \\ &\Longleftarrow \mathsf{I}FYX \\ &\Longleftarrow \mathsf{C}^{[2]}\mathsf{I}XFY. \end{aligned}$$

Therefore $A_X = \mathsf{C}^{[2]}\mathsf{I}X$. We can now state

**Definition 16** ($\mathsf{spore}$)

$$\mathsf{spore}_{m,n} = \mathsf{B}_{[m+n-1]} \circ \mathsf{C}^{[2]}\mathsf{IN} \circ \mathsf{B}\hat{\mathsf{W}}^{m+n-2}, \text{ for } m,n \geq 1.$$

Again, in this definition, $\mathsf{N}$ may be replaced by any desired terminal group. Figure 10 depicts a simple pore (generated by $\mathsf{spore}_{6,9}$) embedded in a cross-linked membrane. As in our previous report [Mac02a], the red color of the warp (vertical) lines indicates that they are chains of $\mathsf{A}$ (application) nodes, while the green color of the woof (horizontal) lines indicates that they are made of $\mathsf{V}$ (sharing) nodes. The only exception is the green line forming the bottom and right margins of the pore, which comprises $\mathsf{A}$ nodes around these two borders.

# 4 Sensors

Channels open or close under control of *sensor molecules*, which can respond to conditions, such as electromagnetic radiation or the presence of chemical species of

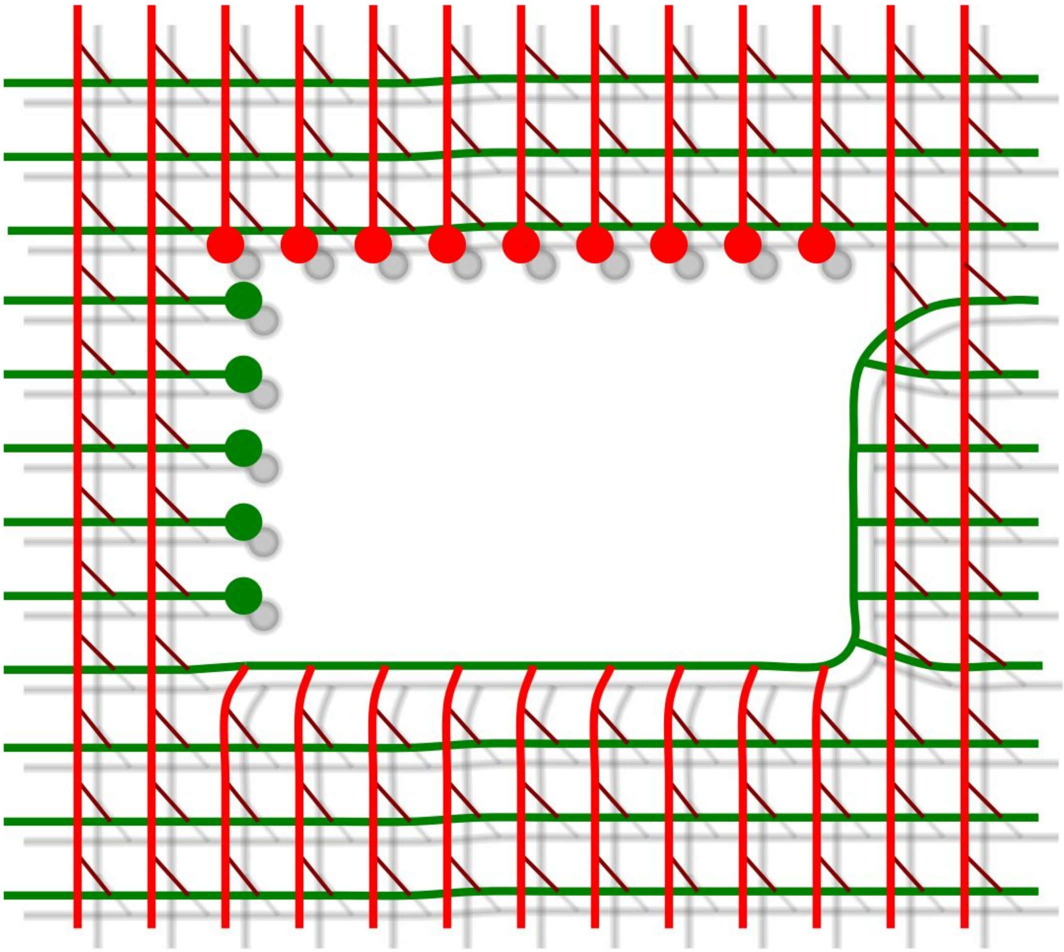

Figure 10: A simple pore embedded in a cross-linked membrane. This is a $6 \times 9$ pore generated by $\mathsf{spore}_{6,9}$.

interest. A *triggered* (as opposed to *sampled* or *polled*) sensor is one in which the occurrence of a condition triggers computation. Formally, the triggering condition is considered a required reactant for the substitution. Each sensor molecule will be unique, but a simple interface standard is desirable.

This is most simply accomplished by synthesizing an (otherwise inert) molecular group, which we denote $\mathsf{K}_{-\lambda}$, that responds to condition $\lambda$ by reconfiguring into a $\mathsf{K}$ combinator (or a group recognized by the $\mathsf{K}$ reaction). Formally, this is described by the reaction equation:

$$\mathsf{K}_{-\lambda} + \lambda \longrightarrow \mathsf{K} + \text{other reaction products}.$$

We call this a $\mathsf{K}$-*active sensor*; obviously an $\mathsf{S}$-active sensor could be defined analogously.

$\mathsf{K}$-active sensors can be used for a kind of triggered conditional execution. For example, an expression of the form $\mathsf{K}_{-\lambda}FXY$ will be irreducible so long as it is not exposed to condition $\lambda$. When $\lambda$ is present, however, the following reduction is enabled:

$$\mathsf{K}_{-\lambda}FXY + \lambda \Longrightarrow \mathsf{K}FXY \Longrightarrow FY,$$

which enables $FY$, the application of function $F$ to argument $Y$, to proceed. More explicitly, the reaction is[1]

$$U\mathsf{A}_3\mathsf{K}_{-\lambda}FXY + \lambda + 2\mathsf{DQ} \longrightarrow U\mathsf{A}FY + \mathsf{DAQ}X + \mathsf{DAKQ} + \text{other reaction products}.$$

As we will see (Sec. 5.2 below), the deletion of structure $X$, resulting from $\mathsf{DAQ}X$, can also have useful effects. It should also be observed that in the "conditional" expression $\mathsf{K}_{-\lambda}FXY$, computation (substitution reactions) may be taking place in any or all of the substructures $F$, $X$, and $Y$, either before or after the triggering of the sensor; this is a consequence of the order-independence of combinatory computing. What the untriggered sensor blocks is the application of function $F$ to its argument $Y$.

Due to the order-independence of combinatory computing, triggered sensors must be handled carefully. For example, structures containing $\mathsf{K}_{-\lambda}$ can be replicated, but the replication operation will require an adequate supply of $\mathsf{K}_{-\lambda}$ molecules. The supply can be diminished by the presence of $\lambda$, which will degrade the $\mathsf{K}_{-\lambda}$ molecules into $\mathsf{K}$s. Similarly, if a structure $\mathcal{S}(\mathsf{K}_{-\lambda})$, which contains sensors, is exposed to $\lambda$ before it is replicated, then it is the altered (already triggered) structure $\mathcal{S}(\mathsf{K})$ that will be replicated, rather than the untriggered. The practical effect of these problems is that all self-assembly of structures containing triggered sensors must be allowed to proceed to completion under conditions in which the triggering situations are impossible. Once the structure is complete (including all required replication of the sensors), it can be placed in its operational environment, in which the triggering condition may occur.

[1] This equation makes use of the revised $\mathsf{K}$ reaction, $U\mathsf{A}_2\mathsf{K}XY + 2\mathsf{DQ} \longrightarrow UX + \mathsf{DAQ}Y + \mathsf{DAKQ}$, which corrects that originally described [Mac02c, Mac02d].

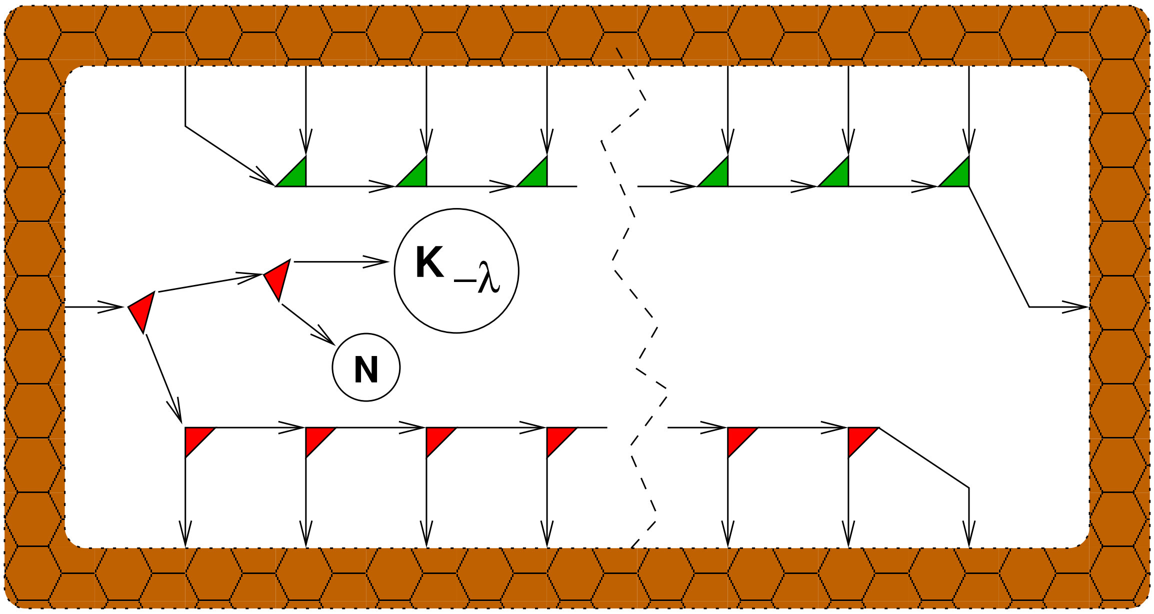


Figure 11: Active region of simple open-once channel in quiescent state. The larger circle represents the sensor $\mathsf{K}_{-\lambda}$, which has the effect of a $\mathsf{K}$ combinator when triggered by environmental condition $\lambda$. The smaller circle is an $\mathsf{N}$ (inert) group. Red triangles are $\mathsf{A}$ (application) nodes, and green triangles are $\mathsf{V}$ (sharing) nodes. The colored border represents the surrounding membrane (e.g., a hexagonal membrane).

*Sampled* or *polled* sensors respond to a condition only when they are asked to do so. We have not yet investigated this kind of sensor, since it is generally incompatible the order-independent computing of combinatory logic. However, they may be useful in some applications, including the construction of channels that open and close repeatedly.

# 5 Channels

## 5.1 Introduction

Given a sensor, "one-shot" channels — which open and stay open, or close and stay closed — are easy to implement. In the former case, the sensor triggers the dissolution of the interior of its patch (perhaps using the deletion operator $\mathsf{D}$ to disassemble it). In the latter case, the sensor triggers a synthesis process to fill in a pore. Reusable channels (which open and close repeatedly) are more complicated, since, in order to reset themselves, they need a supply of sensor molecules that are protected from being triggered before they are used.

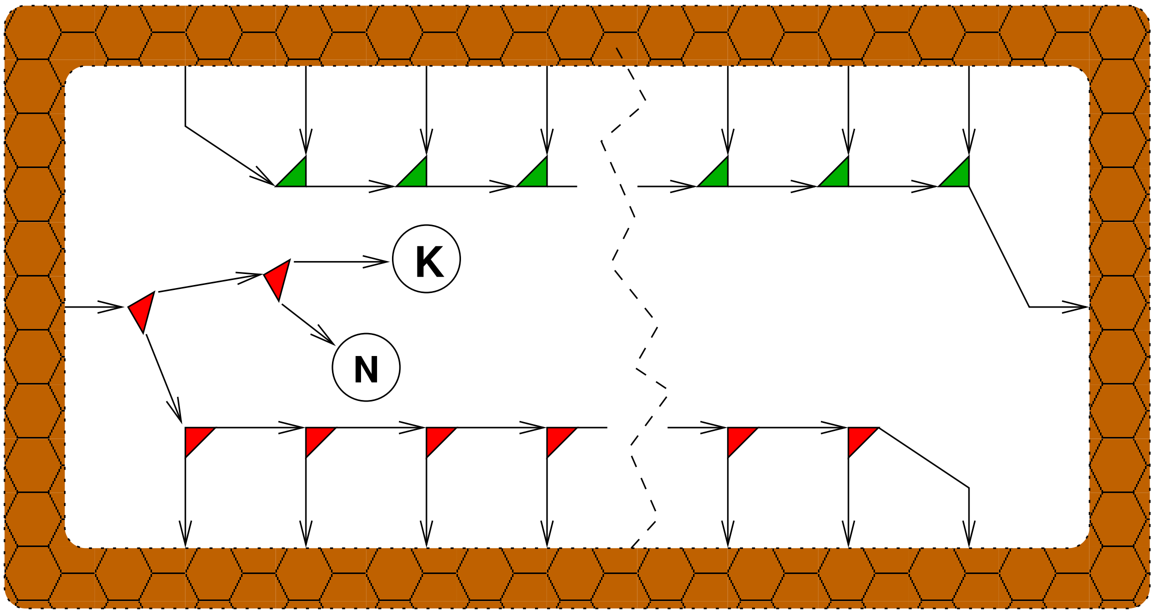


Figure 12: Active region of open-once channel in triggered state. The sensor group $\mathsf{K}_{-\lambda}$ has reconfigured into a $\mathsf{K}$ combinator, resulting in the structure $\mathsf{KN}U$, where $U$ denotes the horizontal chain of red ($\mathsf{A}$) nodes.

### 5.2 Open-once Channel

To illustrate the assembly and operation of channels, we will describe a *simple open-once channel*, that is, a channel that, when subjected to a triggering environmental stimulus, opens a space or pore in a membrane. The active region of such a channel, that is, the part that responds to the environmental situation, is shown in its quiescent state in Fig. 11. This active region is along the upper border of the channel; that is, when the channel opens, a portion of the membrane below the active region will be removed. (If this is confusing, refer to Fig. 16.)

It can be seen in Fig. 11 that the membrane above the channel, whose lower border is formed by the green $\mathsf{V}$ (sharing) nodes, is disconnected from the active region of the channel, comprising the red $\mathsf{A}$ (application) nodes and the others ($\mathsf{N}$, $\mathsf{K}_{-\lambda}$). Thus, even in its quiescent state, there is a gap in the membrane. Depending on the size of the channel, the stiffness of the membrane, and other factors, this may allow some leakage through the membrane. It is in fact possible to construct a channel with these two margins "knitted" together, but it is more complicated than the *simple* open-once channel described here.

When environmental condition $\lambda$ occurs, the sensor group $\mathsf{K}_{-\lambda}$ reconfigures so that it is recognized as a $\mathsf{K}$ combinator; this situation is depicted in Fig. 12. Letting $U$ denote the horizontal chain of red ($\mathsf{A}$) nodes, we now have the combinatory expression $\mathsf{KN}U$, which reduces to $\mathsf{N}$; this is depicted in Fig. 13. As a consequence, $U$ is a reaction

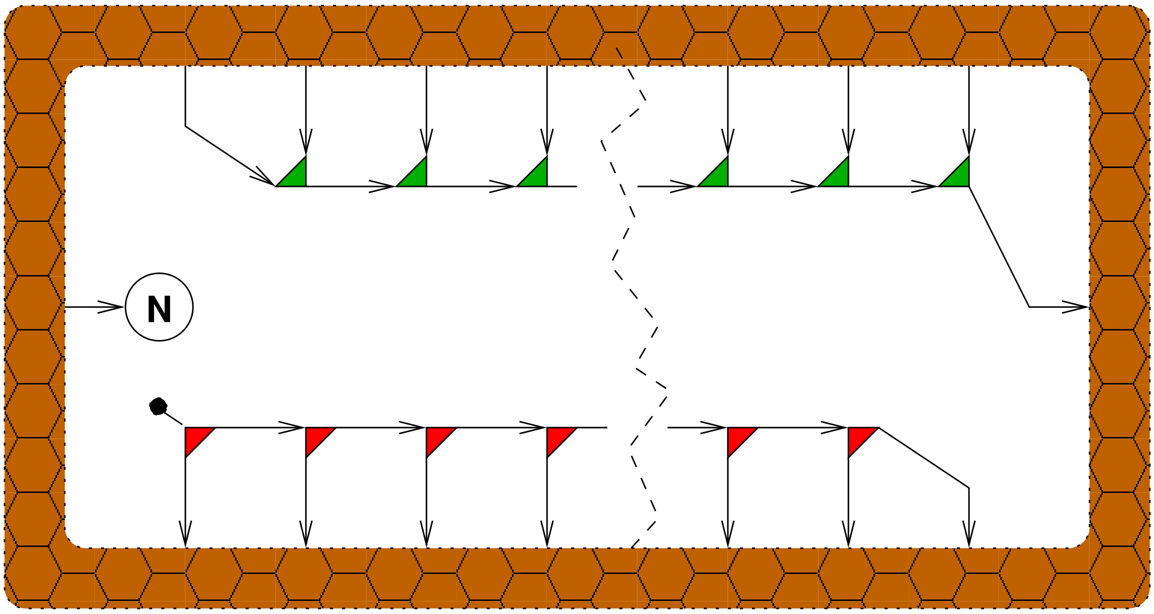


Figure 13: Disconnection of upper border. The combinator substitution reaction $\mathsf{KN}U \Longrightarrow \mathsf{N}$ results in $U$, the horizontal chain of red ($\mathsf{A}$) nodes, being designated as a reaction waste product. The small black circle is a $\mathsf{D}$ (delete) operator.

waste product, and so it is marked by a $\mathsf{D}$ (delete) operator (represented by the small black circle in Fig. 13). (The detailed mechanisms for disassembly and deletion of reaction waste products are discussed in a prior report [Mac02d].)

The disassembly and deletion process, triggered by the $\mathsf{D}$ operator, will disconnect all the $\mathsf{A}$ nodes in $U$ chain (the horizontal red chain in Fig. 13), which will in turn trigger disassembly and deletion of part of the membrane below it; this is depicted in Fig. 14. The shape and extent of the region deleted depends on the structure of the membrane (for the particular case of a hexagonal membrane, see Fig. 16).

To synthesize such a channel, the upper border is constructed in patch synthesizer format, as shown in Fig. 15. The required effect of the synthesizer is:

$$\mathsf{sopen}_{\lambda,n} F Y_1 \cdots Y_n X \Longrightarrow F(\mathsf{K}_{-\lambda}\mathsf{N}(Y_1(Y_2 \cdots (Y_{n-1} Y_n) \cdots))) X^{(n-1)} \cdots X^{(0)}.$$

Again, we apply regular operators to sequentially modify the argument structure. We begin with

$$F Y_1 Y_2 \cdots Y_{n-1} Y_n X.$$

The $F$ and its $n$ arguments $Y_j$ are grouped to the right by $\mathsf{B}_{[n-1]}$:

$$F(Y_1(Y_2 \cdots (Y_{n-1} Y_n) \cdots))X.$$

Since $\mathsf{CB}GFU \Longrightarrow \mathsf{B}FGU \Longrightarrow F(GU)$, we apply $\mathsf{CB}(\mathsf{K}_{-\lambda}\mathsf{N})$ to produce:

$$F(\mathsf{K}_{-\lambda}\mathsf{N}(Y_1(Y_2 \cdots (Y_{n-1} Y_n) \cdots)))X.$$

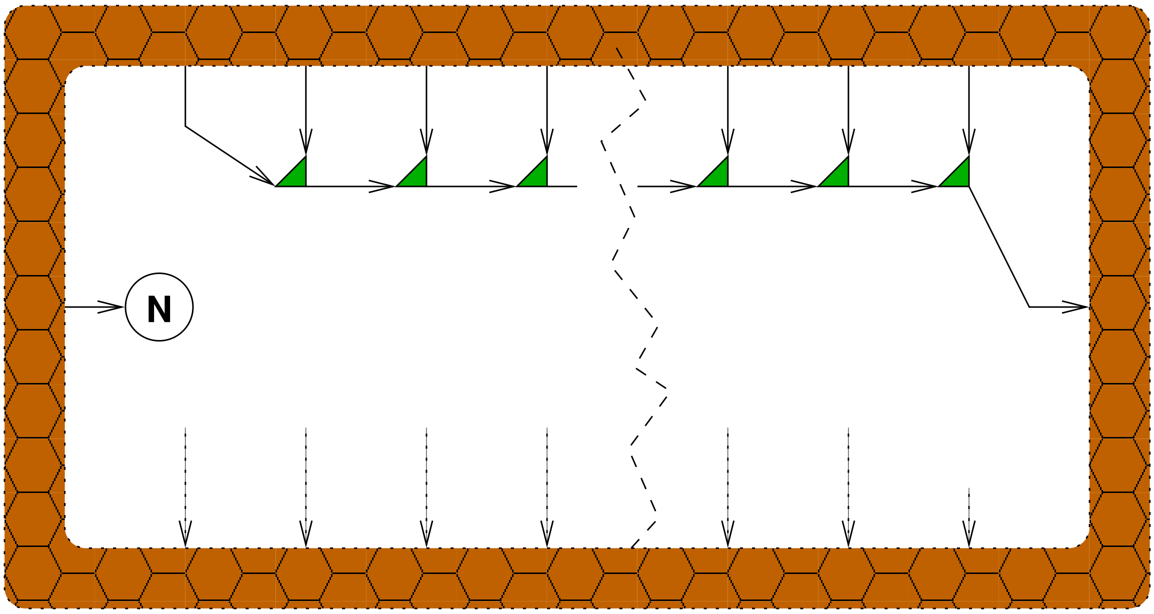


Figure 14: Deletion of upper border triggers opening of channel. Recursive deletion of the $\mathsf{A}$ nodes of the upper border (red chain in Fig. 13) triggers deletion of the part of the membrane that depended on it.

The connection to the right, $X$, is shared $n$ times by $\mathsf{B}\check{\mathsf{W}}^{n-1}$ to yield:

$$F(\mathsf{K}_{-\lambda}\mathsf{N}(Y_1(Y_2\cdots(Y_{n-1}Y_n)\cdots)))X^{(n-1)}\cdots X^{(0)}.$$

Hence we have derived

**Definition 17** (sopen)

$$\mathsf{sopen}_{\lambda,n} = \mathsf{B}_{[n-1]} \circ \mathsf{CB}(\mathsf{K}_{-\lambda}\mathsf{N}) \circ \mathsf{B}\check{\mathsf{W}}^{n-1}.$$

As indicated, the simple channel, when triggered, disconnects the vertical connections $Y_1, \ldots, Y_n$. Depending on the patch below the channel, deletion of the $Y_1, \ldots, Y_n$ will cause the deletion of nodes below them, opening a space in the membrane. For example, Fig. 16 shows how the triggered channel opens a triangular region in a hexagonal membrane. It is apparent that $\mathsf{sopen}_{\lambda,n}$ will open an equilateral triangle, $n$ on a side. Therefore a complete hexagonal channel, comprising the trigger region and hexagonal membrane sufficient for the opened space, can be synthesized by $\mathsf{joinv}_n\mathsf{hpatch}_{n,n}\mathsf{sopen}_{\lambda,n}$. Therefore we define a synthesizer for a hexagonal channel opened by $\lambda$:

**Definition 18** (hsopen)

$$\mathsf{hsopen}_{\lambda,n} = \mathsf{joinv}_n\mathsf{hpatch}_{n,n}\mathsf{sopen}_{\lambda,n}.$$

Of course, such a channel can be combined with membrane patches of any type, not just with other hexagonal patches.

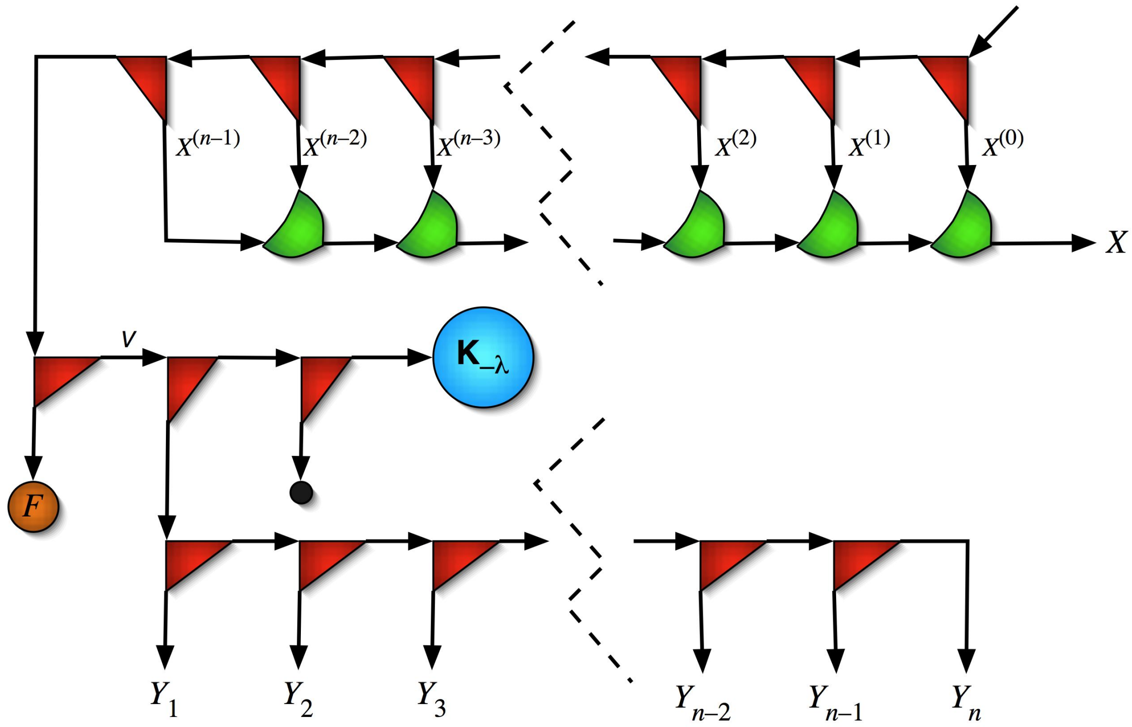


Figure 15: Simple, open-once channel in patch synthesizer format. Such a channel is constructed by $\mathsf{sopen}_{\lambda,n}$. The small, dark circle represents an N (inert) group. Note the orientation of the red A (application) nodes: the shorter leg is the operator link, the longer is the operand link.

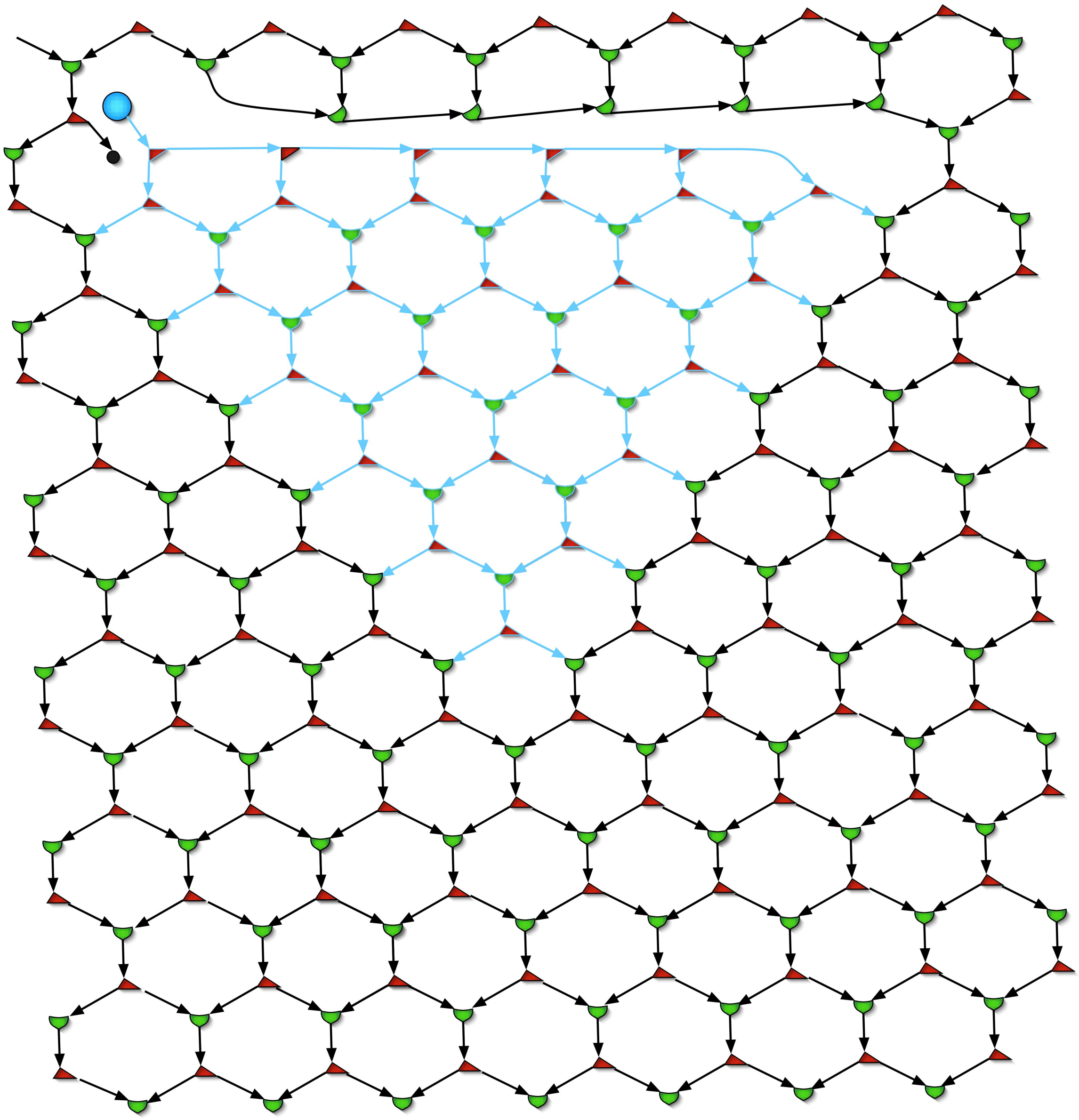

Figure 16: Links and nodes deleted from hexagonal grid when simple channel opens. This figure depicts a simple channel embedded in a hexagonal membrane, with the channel in the same state as in Fig. 14, when the upper border of the channel border is ready to be deleted. The large blue circle represent the D (deletion) operator, and the blue links and blue outlined nodes are those that will be deleted. Thus the channel opens an equilateral triangle in the hexagonal membrane. Since the channel is $\mathsf{sopen}_{\lambda,6}$, the triangular space is 6 on a side.

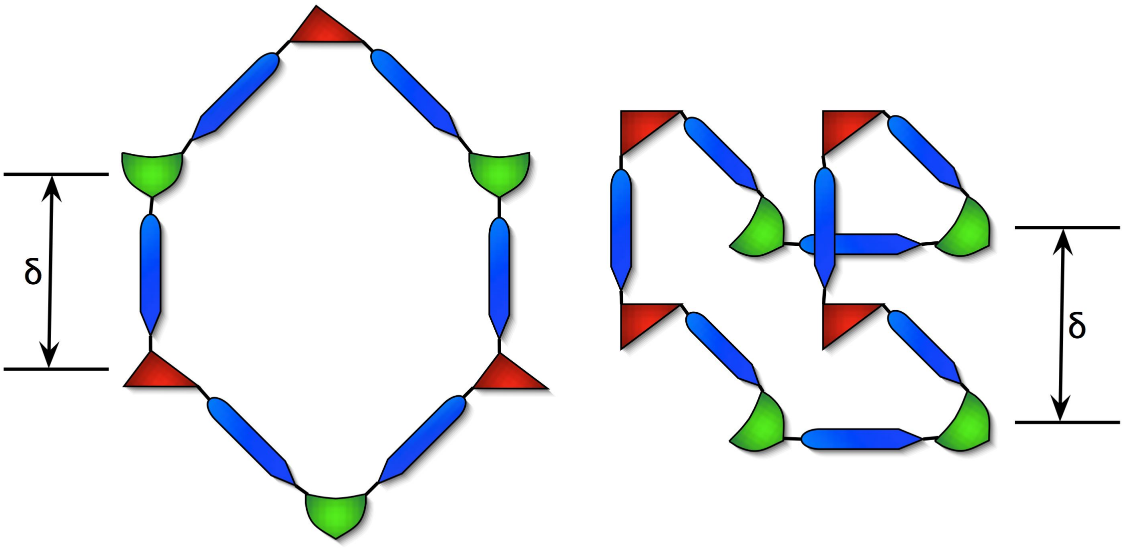


Figure 17: Incremental lengths of combinatory nanostructures. The lengths of the linking groups (blue) and the diameters of the A (application) nodes (red) and of the V (sharing) nodes (green) together define the minimal *incremental length* $\delta$ of possible combinatory nanostructures. The figure shows the smallest or *unit mesh* for the hexagonal and cross-linked membranes.

# 6 Non-unit Meshes

The membranes and nanotubes previously described are said to have a *unit mesh*, that is, the dimensions of the basic (square or hexagonal) cells are determined by the size of the primitive groups (A, V) and the links between them (Fig. 17). Thus, the sizes of these groups define the minimal incremental lengths of possible combinatory nanostructures, the *incremental length* being the average distance between the units of a repeating structure. Therefore, we cannot define structures with a mesh smaller than the minimal incremental length, but we can define structures with larger meshes. There are several ways to accomplish this.

First, it is relatively straight-forward to modify the preceding definitions to have larger mesh-dimensions (multiples of the cells). In addition, various pendant groups can be incorporated into the structure. This approach will not be addressed in this report.

Second, it can be seen that a $k \times k$ pore is similar to a cell in a mesh-$k$ membrane, and, with care, they can be used for this purpose. The problem is that pores don't really have connected borders on their left or top sides (see Fig. 9 and remember that the patch backbone is deleted when the pore is embedded in a membrane). Therefore, when two pores are joined, the border between them is missing or incomplete (e.g., a dangling chain of N nodes). However, borders can be attached by adding $k \times 1$ or $1 \times k$ membrane patches to the edges. The resulting cells can then be iterated horizontally and vertically to construct a membrane, which may be adequate for its purpose, but is not purely a mesh-$k$ membrane, since the cells have $k$ connections (rather than 1) on each side.

Third, this use of pores suggests a way of constructing a large-mesh cell, called a *spanning patch*, that is more general. We will develop a patch structured as shown in Fig. 18. In terms of its connections, it has the general structure of a $2 \times 2$ cross-linked membrane. However, it incorporates four functions (i.e., combinatory structures that compute during patch assembly), which control the structure of the four borders; $H_1$ and $H_2$ control the spacing along the lower and upper cell boundaries; $P_1$ and $P_2$ control the spacing on the left and right boundaries. By use of these four functions quite general structures may be created. To keep the roles of the four functions clear, we use the following notation for the spanning patch synthesizer:

$$\mathsf{xspan}\left[\begin{array}{c} H_2 \\ P_1 \quad P_2 \\ H_1 \end{array}\right].$$

The definition of the spanning patch is derived by the sequential composition of regular operators. We begin with the patch parameters:

$$FY_1Y_2X_1X_2.$$

Applying $\mathsf{C}^{[3]}$ rotates the parameters left:

$$FY_2X_1X_2Y_1.$$

Next $\mathsf{CI}P_2$ places $P_2$ after $F$:

$$FP_2Y_2X_1X_2Y_1.$$

$\mathsf{B}$ groups it with its argument:

$$F(P_2Y_2)X_1X_2Y_1.$$

Next $\check{\mathsf{S}}^{[2]}$ shares the $X_k$ and makes one copy of each an argument of $P_2Y_2$:

$$FX_1X_2(P_2Y_2X_1'X_2')Y_1. \tag{10}$$

Note (Fig. 18) that $U_2 = P_2Y_2X_1'X_2'$, so we have:

$$FX_1X_2U_2Y_1.$$

Next we use $\mathsf{C}_{[2]}\mathsf{I}H_1H_2$ to place the $H_k$ after $F$, ready to be applied to the corresponding $X_k$:

$$FH_1H_2X_1X_2U_2Y_1.$$

$\Psi$ applies the $H_k$ to the corresponding $X_k$:

$$F(H_1X_1)(H_2X_2)U_2Y_1.$$

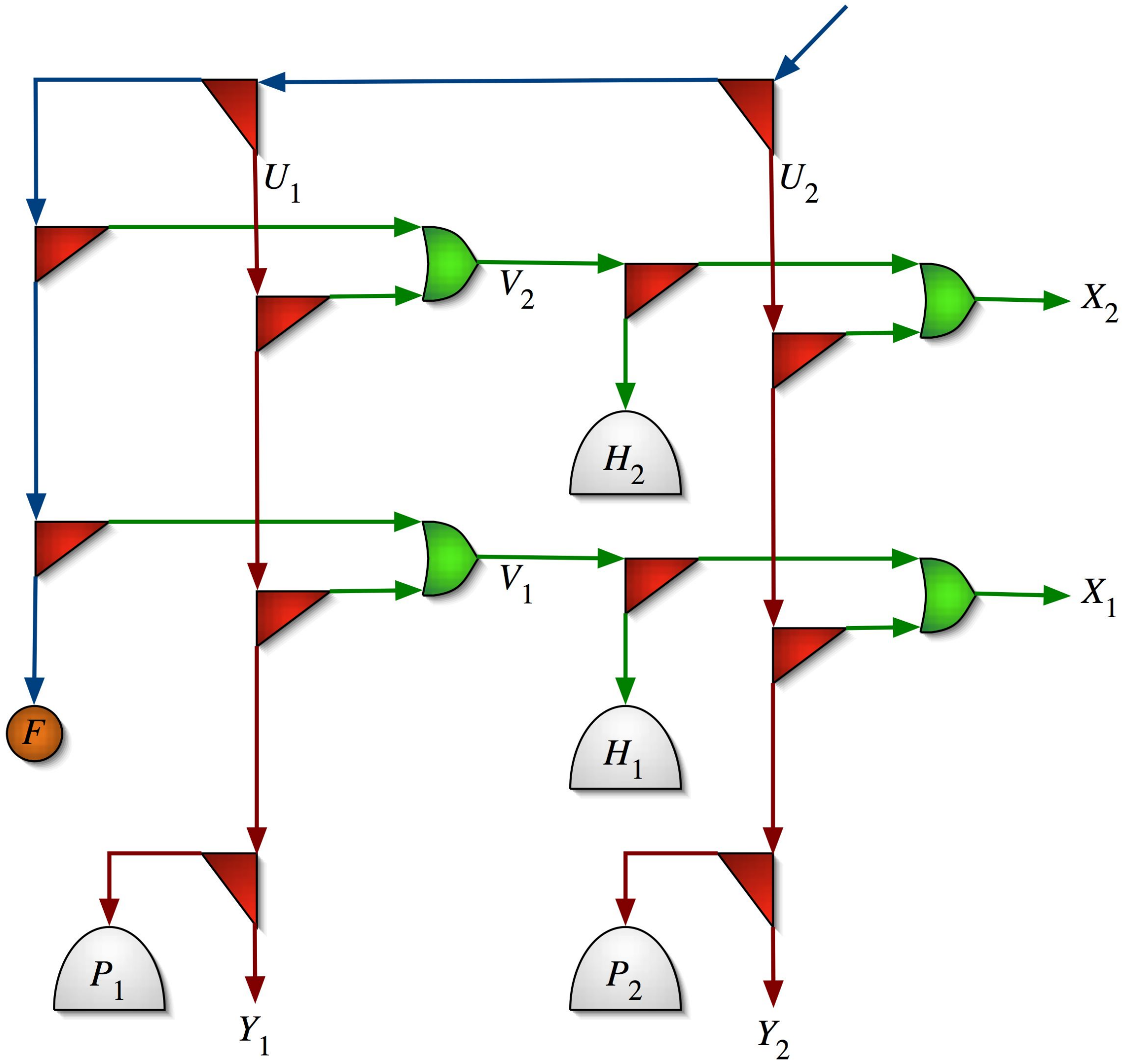


Figure 18: Cross-linked spanning patch. The *horizontal spanning functions* $H_k$ are combinatory trees defining functions that can create chains of arbitrary length between the corresponding $V_k$ and $X_k$, thus making the patch as wide as desired. Similarly, the *perpendicular spanning functions* $P_j$ are functions that can operate to create a chain between the corresponding $U_j$ and $Y_j$. The horizontal $\mathsf{V}$-chains (the woof) are shown in green; the vertical $\mathsf{A}$ chains (warp) are in red; blue indicates the backbone of the patch synthesizer. Such a spanning patch is created by $\mathsf{xspan}\begin{bmatrix} & H_2 & \\ P_1 & & P_2 \\ & H_1 & \end{bmatrix}$.

Note that $V_k = H_k X_k$, so this is equivalent to:

$$FV_1V_2U_2Y_1.$$

Now we need to work on $Y_1$, so we use $C_{[3]}$ to rotate it left circular to $F$:

$$FY_1V_1V_2U_2.$$

As before, we must place $P_1$ after $F$ and group it with $Y_1$, which is the double step $\mathsf{C}\mathsf{I}P_1 \circ \mathsf{B}$:

$$F(P_1Y_1)V_1V_2U_2.$$

Now $\check{\mathsf{S}}^{[2]}$ duplicates the $V_k$ and applies $P_1Y_1$ to one set of them:

$$FV_1V_2(P_1Y_1V_1'V_2')U_2. \tag{11}$$

But the parenthesized expression is just $U_1$, so we have

$$FV_1V_2U_1U_2,$$

which is the required result. Combining the individual steps in sequential composition, we have the (surprisingly complex)

**Definition 19** ($\mathsf{xspan}$)

$$\mathsf{xspan}\begin{bmatrix} & H_2 & \\ P_1 & & P_2 \\ & H_1 & \end{bmatrix} = \mathsf{C}^{[3]} \circ \mathsf{C}\mathsf{I}P_2 \circ \check{\mathsf{S}}^{[2]} \circ \mathsf{C}_{[2]}\mathsf{I}H_1H_2 \circ \Psi \circ \mathsf{C}_{[3]} \circ \mathsf{C}\mathsf{I}P_1 \circ \check{\mathsf{S}}^{[2]}.$$

When more convenient, we also write $\mathsf{xspan}\{P_1, P_2; H_1, H_2\}$ for this synthesizer.

Most commonly the horizontal and perpendicular spanning functions (the $H_k$ and $P_j$) will be used to generate chains of some terminal group $T$, such as shown in Fig. 19; therefore we derive combinatory expressions to generate such chains.

We begin with the horizontal spanning functions, for which $V_k = H_k X_k$ (Fig. 18). We want to define $\mathsf{spanh}_m T$ so that $V_k = \mathsf{spanh}_m T X_k$ is a horizontal chain of the form shown in Fig. 19(a), which can be written (dropping $k$ subscripts):

$$V = \underbrace{T(T(T \cdots (T(T}_{m\ T\text{'s}}\, X)) \cdots)).$$

This is accomplished by

**Definition 20** ($\mathsf{spanh}$)

$$\mathsf{spanh}_m = \mathsf{W}^{m-1}\mathsf{B}_{[m-1]}.$$

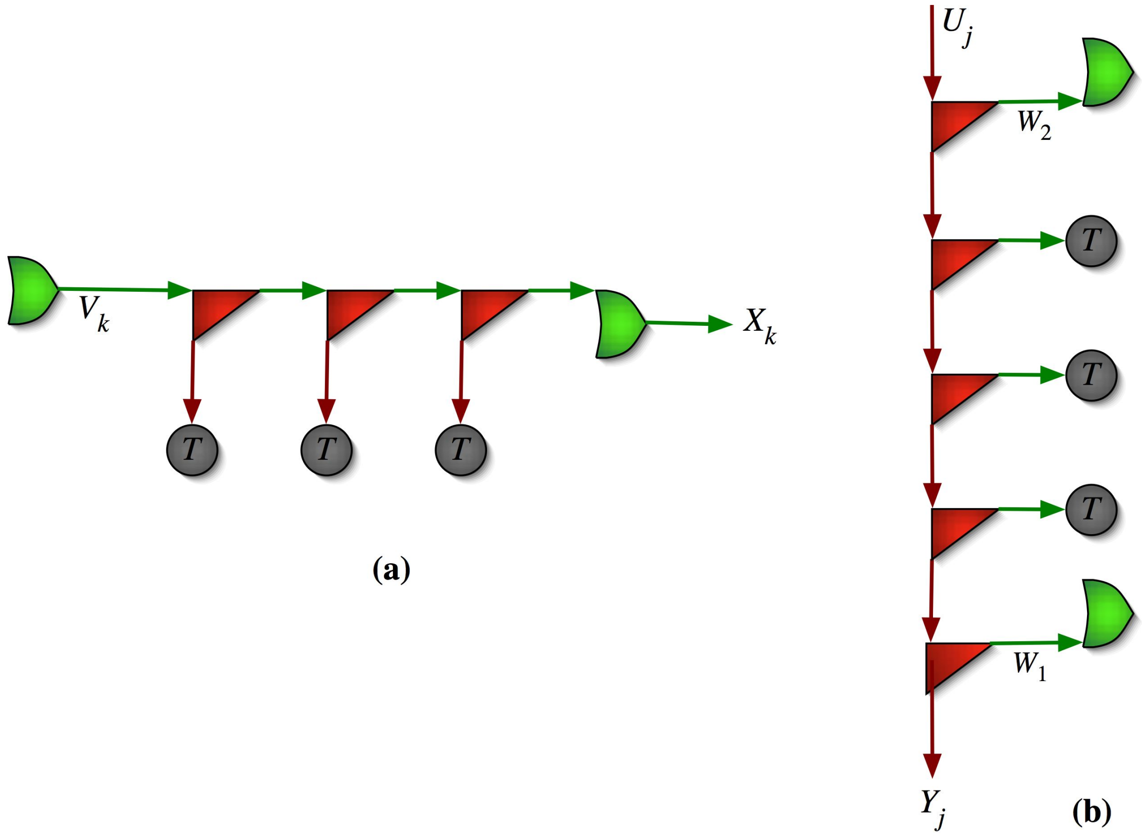


Figure 19: Typical horizontal (a) and perpendicular (b) spanning functions for use with the cross-linked spanning patch. These examples show chains of three links assembled by the functions $\mathsf{spanh}_3 T$ and $\mathsf{spanv}_3 T$. In the vertical chain, the $W_k$ are $V_k$ if $j = 1$ and $X_k$ if $j = 2$ (cf. Fig. 18).

To see this, observe

$$\begin{aligned}
\mathsf{spanh}_m TX &\Longrightarrow \mathsf{W}^{m-1}\mathsf{B}_{[m-1]}TX \\
&\Longrightarrow \mathsf{B}_{[m-1]}\overbrace{TT\cdots T}^{m}X \\
&\Longrightarrow T(T(T\cdots(T(TX))\cdots)).
\end{aligned}$$

For the vertical spanning functions we want (Eqs. 10, 11) $U_1 = P_1Y_1V_1'V_2'$ and $U_2 = P_2Y_2X_1'X_2'$; in general we may write $U = P_jY_jW_1W_2$. The chain to be created by $\mathsf{spanv}_n$ is illustrated in Fig. 19(b), which can be written:

$$\mathsf{spanv}_n TY_jW_1W_2 \Longrightarrow Y_j(W_1\underbrace{(T(T\cdots(T(T}_{n\ T\text{'s}}W_2))\cdots))).$$

This is accomplished by

**Definition 21** ($\mathsf{spanv}$)

$$\mathsf{spanv}_n = \mathsf{W}^{n-1}(\mathsf{C}^2_{[n+1]}\mathsf{B}_{[n+1]}).$$

Observe:

$$\begin{aligned}
\mathsf{spanv}_n TY_jW_1W_2 &\Longrightarrow \mathsf{W}^{n-1}(\mathsf{C}^2_{[n+1]}\mathsf{B}_{[n+1]})TY_jW_1W_2 \\
&\Longrightarrow \mathsf{C}^2_{[n+1]}\mathsf{B}_{[n+1]}\overbrace{TT\cdots T}^{n}Y_jW_1W_2 \\
&\Longrightarrow \mathsf{B}_{[n+1]}Y_jW_1TT\cdots TW_2 \\
&\Longrightarrow Y_j(W_1(T(T\cdots(TW_2)\cdots))).
\end{aligned}$$

With these definitions we can synthesize a single cell of size $k \times l$, with terminal groups $T$ on the vertical borders and $U$ on the horizontal borders, by:

$$\mathsf{xspan}\begin{bmatrix} & \mathsf{spanh}_l U & \\ \mathsf{spanv}_k T & & \mathsf{spanv}_k T \\ & \mathsf{spanh}_l U & \end{bmatrix}.$$

This cell can be iterated vertically and horizontally (by using $\mathsf{iterv}$ and $\mathsf{iterh}$) to assemble a membrane of any desired size.

# References

[Mac02a] Bruce J. MacLennan. Membranes and nanotubes: Progress on universally programmable intelligent matter — UPIM report 4. Technical Report CS-02-495, Dept. of Computer Science, University of Tennessee, Knoxville, 2002. Available at http://www.cs.utk.edu/~library/TechReports/ 2002/ut-cs-02-495.ps.

[Mac02b] Bruce J. MacLennan. Molecular combinator reference manual. Technical report, Dept. of Computer Science, University of Tennessee, Knoxville, 2002. Latest edition available at http://www.cs.utk.edu/~mclennan/UPIM/CombRef.ps.

[Mac02c] Bruce J. MacLennan. Molecular combinator reference manual — UPIM report 2. Technical Report CS-02-489, Dept. of Computer Science, University of Tennessee, Knoxville, 2002. Available at http://www.cs.utk.edu/~library/TechReports/ 2002/ut-cs-02-489.ps.

[Mac02d] Bruce J. MacLennan. Replication, sharing, deletion, lists, and numerals: Progress on universally programmable intelligent matter — UPIM report 3. Technical Report CS-02-493, Dept. of Computer Science, University of Tennessee, Knoxville, 2002. Available at http://www.cs.utk.edu/~library/TechReports/ 2002/ut-cs-02-493.ps.