Supporting Information

S7 TEXT. PSEUDOCODE FOR THE STABLE MOTIF CONTROL ALGORITHM AND THE STABLE MOTIF BLOCKING ALGORITHM

For the pseudocodes we assume that one starts with a target attractor \( A \), the logical functions \( F = (f_1, f_2, \ldots, f_N) \) for the logical network model of interest, and the stable motif succession diagram for the logical network model of interest (see Fig. 2). A stable motif succession diagram can be represented as a directed graph \( G_{\text{diag}} = (V_{\text{diag}}, E_{\text{diag}}) \) together with a dictionary \( L \). The nodes \( V_{\text{diag}} = (v_{\text{diag},1}, v_{\text{diag},2}, \ldots, v_{\text{diag},n}) \) denote either stable motifs \( M_i \) (if the node has at least one outgoing edge) or attractors \( A_i \) (if the node has no outgoing edges). The dictionary \( L \) stores the type of object (stable motif or attractor) each node in \( V_{\text{diag}} \) denotes. Each edge in \( E_{\text{diag}} \) connects a stable motif with the stable motifs or attractors that can be obtained from the reduced network associated to it; if network reduction leads to a simplified network with at least one stable motif, then the edges points from the stable motif being considered to the stable motifs of the simplified network, otherwise, the edges point towards an attractor. It should be noted that stable motifs/attractors may be assigned to more than one node in \( V_{\text{diag}} \). For example, in Fig. 2 there are three nodes that denote the motif \{ \( A = 0 \) \}, and two nodes that denote the attractor \( A_2 \).

A. Pseudocode for the stable motif control algorithm

Step 1: Identify the sequences of stable motifs that lead to \( A \). These can be obtained from the stable motif succession diagram (see Fig. 2) by choosing the attractor of interest in the right-most part and selecting all of the attractor’s predecessors in the succession diagram. The stable motif diagram is represented by the directed graph \( G_{\text{diag}} = (V_{\text{diag}}, E_{\text{diag}}) \) together with the list \( L \).

Algorithm 1: GETSEQUENCES\( (G, L, A) \)

```
comment: Sequences, SequencesLeft, and NewSequences are sets.
         \( S \) is a sequence (ordered list).
Sequences ← empty set
SequencesLeft ← empty set
for each \( v \in \) sink nodes of \( G \)
do  
    \{comment: \( L(v) \) gives the motif or attractor denoted by \( v \).
    if \( L(v) \) equals \( A \)
    then 
        \( S \) ← empty sequence
        add \( v \) to the beginning of \( S \)
        add \( S \) to SequencesLeft
    \}
repeat
NewSequences ← empty set
for each \( S \in \) SequencesLeft
    \( v \) ← first item of \( S \)
    if \( v \) has input nodes
    do  
        \{for each \( v' \in \) input nodes of \( v \)
        then 
            \( S' \) ← copy \( S \)
            add \( v' \) to the beginning of \( S' \)
            add \( S' \) to NewSequences 
        \}
    else add \( S \) to Sequences
    remove \( S \) from SequencesLeft
for each \( S' \in \) NewSequences
    do add \( S' \) to SequencesLeft
until NewSequences is empty
return (Sequences)
```
Step 2: Shorten each sequence $S \in \text{Sequences}$ by identifying the minimum number of motifs in $S$ required for reaching $A$ and removing the remaining motifs from the sequence. This minimum number of motifs can be identified from the stable motif succession diagram (Fig. 2); they are the motifs after which all consequent motif choices lead to the same attractor $A$.

Algorithm 2: $\text{SHORTENSEQUENCES1}(G, L, A, \text{Sequences})$

\begin{verbatim}
comment: $\text{ShortenedSequences1}$ is a set.
    $\mathcal{S}'$ is a sequence (ordered list).
    $\text{pathFound}$ is a Boolean variable

$\text{ShortenedSequences1} \leftarrow \text{empty set}$

for each $S \in \text{Sequences}$
    $S' \leftarrow \text{copy } S$
    for each $v \in S$ in reverse order
        $\text{pathFound} \leftarrow \text{false}$
        for $v' \in \text{sink nodes of } G$
            comment: $L(v')$ gives the motif or attractor denoted by $v'$.
            if $L(v')$ is not $A$
                then if there exists a directed path from $v$ to $v'$
                    then $\text{pathFound} \leftarrow \text{true}$
                        exit for loop
            else remove $v$ from $S'$
        if $\text{ShortenedSequences1}$ does not contain $S'$
            then add $S'$ to $\text{ShortenedSequences1}$
    return $(\text{ShortenedSequences1})$
\end{verbatim}
Step 3: For each stable motif state \( M = (\sigma_{m_1} = b_{m_1}, \sigma_{m_2} = b_{m_2}, \ldots, \sigma_{m_l} = b_{m_l}) \) corresponding to node \( v \), find the subsets of stable motif’s states \( O = \{M_i\}, M_i \subseteq M \) that, when fixed in the logical model, are enough to force the state of the whole motif into \( M \). At worst, there will only be one subset, which will equal the whole stable motif state \( M \). If any of these subsets is fully contained in another subset, remove the larger of the subsets. In each stable motif sequence \( S = (M_1, \ldots, M_L) \), substitute every stable motif \( M_j \) with the subsets of the stable motif states obtained, that is, \( S = (O_1, \ldots, O_L) \).

**Algorithm 3: SequencesWithMotifControlSets(ShortenedSequences1, SequenceDictionary, F, L)**

- **comment:** \( F = (f_1, f_2, \ldots, f_N) \) contains the Boolean functions of the logical model.
- \( O \) and \( Subsequence \) are sequences (ordered lists).

\[
\text{ShortenedSequences2} \leftarrow \text{empty set}
\]

for each \( S \in \text{ShortenedSequences1} \) do

- \( \text{index} \leftarrow 0 \)
- \( S' \leftarrow \text{sequence assigned to } S \text{ in SequenceDictionary} \)
- \( S'' \leftarrow \text{empty sequence} \)
- \( F' \leftarrow \text{copy } F \)

for each \( v \in S \) do

- \( \text{Subsequence} \leftarrow \text{empty sequence} \)
- for \( i \leftarrow \text{index} \) to length of list \( S' - 1 \) do
    - \( v' \leftarrow \text{get element of } S' \text{ in position } i \)
    - if \( v' \) equals \( v \) then
        - \( \text{index} \leftarrow i + 1 \)
        - exit for loop
    - add \( v' \) to the end of \( \text{Subsequence} \)

- \( \text{comment:} \ \text{DownstreamEffect}(L(v'), F') \) is described in Algorithm 4.
- \( \text{DownstreamEffect}(L(v'), F') \) evaluates the states of motif \( L(v') \) into \( F' \).
- If any \( f \in F' \) becomes a constant Boolean function after the evaluation, it evaluates the resulting Boolean state of the node corresponding to \( f \) in every \( F' \). This is done iteratively until no new constant Boolean functions are found, at which point the resulting \( F' \) is returned.

for each \( v' \in \text{Subsequence} \) do

- \( F' \leftarrow \text{DownstreamEffect}(L(v'), F') \)

- \( \text{comment:} \ \text{MotifControlSet}(L(v), F') \) is described in Algorithm 5.
- \( \text{MotifControlSet}(L(v), F') \) finds the subsets of stable motif’s states of \( L(v) \) that, when fixed, are enough to force the state of the whole motif into \( L(v) \).

- \( O \leftarrow \text{MotifControlSet}(L(v), F') \)
- add \( O \) to end of \( S'' \)
- \( F' \leftarrow \text{DownstreamEffect}(L(v), F') \)

add \( S'' \) to \( \text{ShortenedSequences2} \)

return \( (\text{ShortenedSequences2}) \)
Algorithm 4: DownstreamEffect($\mathcal{M}, F'$)

comment: DownstreamEffect($\mathcal{M}, F'$) evaluates the states of motif $\mathcal{M}$ into $F'$. If any $f \in F'$ becomes a constant Boolean function after the evaluation, it evaluates the resulting Boolean state of the node corresponding to $f$ in every $F'$. This is done iteratively until no new constant Boolean functions are found, at which point the resulting $F'$ is returned.

$M'$ and $M''$ are sets containing nodes in the logical model together with a Boolean variable with their state.

$F''$ is a sequence (ordered lists) of Boolean functions.

$M' \leftarrow \text{empty set}; M'' \leftarrow \text{copy } M; F'' \leftarrow \text{copy } F'$

repeat
  for each $f \in F''$
    do 
      if $f$ is not a constant Boolean function
        then $f \leftarrow f$ with the states in $M'$ evaluated on it
      if $f$ is a constant Boolean function
        then $\sigma \leftarrow \text{node in the logical model whose function is } f \text{ and the value of } f \text{ as its state.}$
        add $\sigma$ to $M''$
  $M' \leftarrow \text{copy } M''$
  $M'' \leftarrow \text{empty set}$
until $M'$ is empty
return $(F'')$

Algorithm 5: MotifControlSet($\mathcal{M}, F'$)

comment: MotifControlSet($\mathcal{M}, F'$) finds the subsets of stable motif’s states of $\mathcal{M}$ that, when fixed, are enough to force the state of the whole motif into $\mathcal{M}$.

$F'$ and $F''$ are sequences (ordered lists) of Boolean functions.

$F'$ are the logical functions of the nodes in the model whose states are specified in $\mathcal{M}$.

$O$ is a sequence (ordered list).

isMotifControlSet and validSubset are Boolean variables.

$O \leftarrow \text{empty sequence}$

for subsetSize $\leftarrow 1$ to length of list $\mathcal{M} - 1$
  for each $M \in \text{subsets of size subsetSize in } \mathcal{M}$
    do 
      validSubset $\leftarrow$ true
      for each $M' \in O$
        do 
          if $M'$ is a subset of $M$
            then validSubset $\leftarrow$ false
          if not validSubset
            then exit for loop
      if $O$ is empty
        then add $\mathcal{M}$ to $O$
    if not validSubset
      then exit for loop
  isMotifControlSet $\leftarrow$ true
  $F'' \leftarrow \text{DownstreamEffect}(\mathcal{M}, F')$
  for each $f \in F''$
    do 
      if $f$ is not a constant Boolean function
        then isMotifControlSet $\leftarrow$ false
      if isMotifControlSet
        then add $M$ to $O$
  if isMotifControlSet
    then add $\mathcal{M}$ to $O$
return $(O)$
Step 4: For each sequence $S = (O_1, \ldots, O_L)$ create a set of states $\mathcal{C}$ by choosing one of the subsets of stable motif states $M_{k_j}$ in each $O_j$ and taking their union, that is, $\mathcal{C} = M_{k_1} \cup \cdots \cup M_{k_L}$, $M_{k_j} \in O_j$. The network control set for attractor $A$ is the set of states $A = \{C_i\}$ obtained from all possible combinations of $M_{k_j}$'s for every sequence $S$. To avoid any redundancy, we additionally prune $\mathcal{C}$ of duplicates and remove the states $C_i$ which are supersets of any of the other states $C_j$ (i.e. $C_j \subset C_i$).

Algorithm 6: StableMotifControlSets(ShortenedSequences2)

```
comment: ControlSets, ControlSet, and $M$ are sets
          $O$ is a sequence (ordered list).
          $L$ and $index$ are integers.
          $countArray$ and $countArrayMax$ are arrays of integers.
ControlSets ← empty set
for each $S \in$ ShortenedSequences2
  $L$ ← length of list $S$
  comment: $countArray$ and $countArrayMax$ keep track of the combinations of motifs
            in $S$ that we have tried and that we have left.
  $countArray$ ← array of integers of length $L$
  $countArrayMax$ ← array of integers of length $L$
  for $i$ ← 0 to $L - 1$
    do $O$ ← get element of $S$ in position $i$
        $countArrayMax[i]$ ← length of list $O$
        $countArray[i]$ ← 0
  repeat
    $ControlSet$ ← empty set
    for $i$ ← 0 to $L - 1$
      do $O$ ← get element of $S$ in position $i$
         $M$ ← get element of $O$ in position $countArray[i]$
         for each $\sigma \in M$
           do add $\sigma$ to $ControlSet$
           add $ControlSets$ to $ControlSets$
    comment: $index$ gets increased whenever $countArray[index]$ reaches its
             max value, $countArrayMax[index]$.
    $index$ ← 0
    repeat
      $increasedIndex$ ← false
      $countArray[index]$ ← $countArray[index]$ + 1
      if $countArray[index]$ equals $countArrayMax[index]$
        then $countArray[index]$ ← 0
        $index$ ← $index + 1$
        $increasedIndex$ ← true
      if $index$ equals $L$
        then exit repeat loop
    until not $increasedIndex$
  until $index$ equals $L$
return ($ControlSets$)
```
Algorithm 7: PruneControlSets(ControlSets)

comment: PrunedControlSets is a set
PrunedControlSets ← copy ControlSets
for each ControlSet ∈ ControlSets
    for each ControlSet' ∈ ControlSets
        do |
            if ControlSet' is not ControlSet
                then |
                    if ControlSet' is a subset of ControlSet
                        then |
                            remove ControlSet from PrunedControlSets
                            exit for loop
        return (PrunedControlSets)

B. Pseudocode for the stable motif blocking algorithm

Step 1: Identify the sequences of stable motifs that lead to A. This step is the same as the first step in the stable motif control algorithm (Algorithm 1), and can be obtained from the stable motif succession diagram (Fig. 2).

Step 2: Take each stable motif’s state $M_i$ in the sequences obtained in the previous step (Sequences). Create a new set $M_A$ with all of these stable motif states, $M_A = \{M_i\}$.

Algorithm 8: MotifStates(Sequences, L)

comment: $M_A$ and $M$ are sets.
$M_A ←$ empty set
for each $S ∈$ Sequences
    do |
        for each $v ∈ S$ s.t. $v$ is not a sink node
            do |
                comment: $M$ stores the states of the motif $L(v)$.
                $M ← L(v)$
                add $M$ to $M_A$
    return ($M_A$)

Step 3: Take each node state $σ_j ⊂ M_i$ of the stable motif’s states $M_i$ in $M_A$. Create a new set $B_A$ with the negation of each node state, $B_A = \{σ_j\}$. The node states in $B_A$ and any combination of them are identified as potential interventions to block attractor $A$.

Algorithm 9: StableMotifBlocking($M_A$)

comment: $B_A$ is a set.
$B_A ←$ empty set
for each $σ ∈ M_A$
    do |
        comment: $σ'$ is a node in the logical model together with a Boolean variable with its state.
        $σ' ←$ reverse node state of $σ$
        add $σ'$ to $B_A$
return ($B_A$)