Integrating prior knowledge in Automatic Network Reconstruction

M. Favre  W. Marwan  A. Wagler

Laboratoire d’Informatique, de Modélisation et d’Optimisation des Systèmes (LIMOS)
UMR CNRS 6158 – Université Blaise Pascal, Clermont-Ferrand, France

Magdeburg Center for Systems Biology (MaCS), Otto-von-Guericke-Universität, Magdeburg, Germany

BioPPN 2014

TUNIS

June 23, 2014
Outline

1 Petri Nets and Extensions

2 Reconstruction Approach (Sketch)

3 Minimality Aspect

4 Integrating Prior Knowledge

5 Summary and Conclusions
Outline

1. Petri Nets and Extensions
2. Reconstruction Approach (Sketch)
3. Minimality Aspect
4. Integrating Prior Knowledge
5. Summary and Conclusions
Petri nets: The networks

**Standard networks** \( \mathcal{P} = (P, T, A, w) \)

- \( P \) set of involved **components** ("places" \( \bigcirc \)), (molecules, receptors, genes)
- \( T \) set of involved **reactions** ("transitions" \( \square \)), (reactions, activations,...)
- \( A \subseteq (P \times T) \cup (T \times P) \) set of directed **links** ("arcs" \( \rightarrow \)),
- arc weights \( w \) as stoichiometric coefficients.

**Extended networks** \( \mathcal{P} = (P, T, (A \cup A_R \cup A_I), w) \)

- \( A_R \subseteq P \times T \) **read-arcs**
- \( A_I \subseteq P \times T \) **inhibitor-arcs**
States

- Each place $p \in P$ can be marked with an integral number $x_p$ of tokens.
- A state can be represented as a vector $x \in \mathbb{N}^{|P|}$ with entries $x_p$ for all $p \in P$.

\[
\begin{bmatrix}
2 \\
0 \\
0 \\
1 \\
1 \\
0
\end{bmatrix}
\]

Switching transitions

- In a standard network, $t \in T$ is enabled at a state $x$ if
  $x_p \geq w(p, t)$ for all $p$ with $(p, t) \in A$, and
  switching $t$ yields a new state $x'$ with $x'_p = x_p + w(p, t)$ $\forall (p, t), (t, p) \in A$.
- In an extended network, $t \in T$ is enabled at a state $x$ if in addition
  $x_p \geq w(p, t)$ for all $p$ with $(p, t) \in A_R$, and,
  $x_p < w(p, t)$ for all $p$ with $(p, t) \in A_I$. 
Dynamic processes and reachability

**Dynamic processes** correspond to sequences of system states, obtained by sequences of transition switches.

Starting from an initial state $x^0$, explore the **dynamic behavior** of the system by
- consecutively switching enabled transitions,
- analyzing the reachability of certain system states.

Can we control transition switches and, thus, the dynamic behavior?
Dynamic processes and reachability

Dynamic processes correspond to sequences of system states, obtained by sequences of transition switches.

Starting from an initial state $x^0$, explore the **dynamic behavior** of the system by
- consecutively switching enabled transitions,
- analyzing the reachability of certain system states.

Can we control transition switches and, thus, the dynamic behavior?
Prediction of the dynamic behavior

Priority relations for transitions (Marwan, W. & Weismantel 2008)

Let $G$ be a network and $\mathcal{O}$ a priority relation on its transitions.

- If there are two or more transitions enabled at a state, this transition with highest priority will be switched.
- This allows to predict the dynamic behavior of the system (instead of listing all potentially possible switching sequences).

Example. Consider the network $G$ with $\mathcal{O} = \{ t_2 > t_3 \}$. 

![Diagram of network transitions with priority relations applied]
Outline

1. Petri Nets and Extensions
2. Reconstruction Approach (Sketch)
3. Minimality Aspect
4. Integrating Prior Knowledge
5. Summary and Conclusions
The Input

The input \((P, I_P, \mathcal{X}')\) consists of

- a set \(P\) of studied components
- the set \(I_P\) of all known \(P\)-invariants
- experimental time-series data \(\mathcal{X}' = \{\mathcal{X}'(x^1, x^k) : x^1 \in \mathcal{X}'_{ini}, x^k \in \mathcal{X}'_{term}\}\) with
  - initial states \(\mathcal{X}'_{ini} \subseteq \mathcal{X}'\), terminal states \(\mathcal{X}'_{term} \subseteq \mathcal{X}'\)
  - time series \(\mathcal{X}'(x^1, x^k) = (x^0; x^1, \ldots, x^j, \ldots, x^k)\)

Required properties of \(\mathcal{X}'\)

- **reproducibility**: for each \(x^j \in \mathcal{X}'\) there is a unique observed successor state \(\text{succ}_{\mathcal{X}'}(x^j) = x^{j+1} \in \mathcal{X}'\)
  (can be ensured by pre-processing (W., Wegener 2013))

- **monotonicity**: for each pair \((x^j, x^{j+1}) \in \mathcal{X}'\), the values of the elements do not oscillate in the possible intermediate states between \(x^j\) and \(x^{j+1}\)
  (depends on the chosen time-points (Durzinsky, W., Weismantel 2008))
The Input

The input \((P, \mathcal{I}_P, \mathcal{X}')\) consists of

- a set \(P\) of studied components
- the set \(\mathcal{I}_P\) of all known \(P\)-invariants
- experimental time-series data \(\mathcal{X}' = \{\mathcal{X}'(x^1, x^k) : x^1 \in \mathcal{X}_{ini}', x^k \in \mathcal{X}_{term}'\}\) with
  - initial states \(\mathcal{X}_{ini}' \subseteq \mathcal{X}'\), terminal states \(\mathcal{X}_{term}' \subseteq \mathcal{X}'\)
  - time series \(\mathcal{X}'(x^1, x^k) = (x^0; x^1, \ldots, x^j, \ldots, x^k)\)

Required properties of \(\mathcal{X}'\)

- **reproducibility**: for each \(x^j \in \mathcal{X}'\) there is a unique observed successor state \(\text{succ}_{\mathcal{X}'}(x^j) = x^{j+1} \in \mathcal{X}'\)
  (can be ensured by pre-processing (W., Wegener 2013))

- **monotonicity**: for each pair \((x^j, x^{j+1}) \in \mathcal{X}'\), the values of the elements do not oscillate in the possible intermediate states between \(x^j\) and \(x^{j+1}\)
  (depends on the chosen time-points (Durzinsky, W., Weismantel 2008))
Example: Light-induced sporulation of *Physarum polycephalum*

Experimental biological data from Starostzik and Marwan (1995)

\[ P = \{ FR, R, P_{FR}, P_{R}, G, Spo \}, \quad X'(x^1, x^4) = (x^0, x^1, x^2, x^3, x^4), \quad X'_{\text{ini}} = \{ x^1, x^5, x^6 \}, \]

\[ \mathcal{I}_P = \{ P_{FR}, P_{R} \}, \quad X'(x^5, x^0) = (x^2, x^5, x^0), \quad X'_{\text{term}} = \{ x^4, x^0, x^8 \} \]

serve as input for the algorithm, we present all observed states schematically:
Time series \((x^0, x^1, \ldots, x^k)\) are sequences of measured system states.

A network **fits** the given data \(\mathcal{X}'\) if it can **reproduce** the observations:

For given \(\mathcal{X}'\), a network \(\mathcal{P} = (P, T, (A \cup A_R \cup A_I), w)\) with priorities \(O\) is **\(\mathcal{X}'\)-deterministic** if it contains and correctly selects transitions to reach, from every observed state \(x^j \in \mathcal{X}'\), its observed successor \(x^{j+1} \in \mathcal{X}'\).

**Remark:** Finding all networks fitting the data means to create all **minimal** ones (that do not reproduce the data anymore after removing any element).
Time series \((x^0, x^1, \ldots, x^k)\) are sequences of measured system states.

A network **fits** the given data \(\mathcal{X}'\) if it can **reproduce** the observations:

For given \(\mathcal{X}'\), a network \(P = (P, T, (A \cup A_R \cup A_I), w)\) with priorities \(O\) is **\(\mathcal{X}'\)-deterministic** if it contains and correctly selects transitions to reach, from every observed state \(x^j \in \mathcal{X}'\), its observed successor \(x^{j+1} \in \mathcal{X}'\).

**Remark:** Finding all networks fitting the data means to create all **minimal** ones (that do not reproduce the data anymore after removing any element).
Step 1: Representation of observed responses

- Time series \((x^0; x^1 \ldots, x^k)\) may not consist of consecutive system states, as between \((x^j, x^{j+1}) \in \mathcal{X}''\), there can be non-observed intermediate states.
- To reach \(x^{j+1}\) from \(x^j\), we then need a sequence of transitions \(t_i, \ldots, t_k\) whose update vectors satisfy \(x^{j+1} - x^j = r_{t_i} + \ldots + r_{t_k}\).

Theorem (Durzinsky, W., Weismantel 2008)

To represent the difference \(d = x^{j+1} - x^j\), use only sign-compatible vectors from

\[
\text{Box}(d) = \left\{ r \in \mathbb{Z}^{|P|} : \begin{array}{l}
0 \leq r_p \leq d_p \quad \text{if } d_p > 0 \\
d_p \leq r_p \leq 0 \quad \text{if } d_p < 0 \\
r_p = 0 \quad \text{if } d_p = 0 \\
\sum_{p \in P'} r_p = 0 \quad \forall P' \in \mathcal{I}_P
\end{array} \right\} \setminus \{0\}.
\]

For each \(d = x^{j+1} - x^j\), determine all solutions \(\lambda \in \Lambda(d)\) with

\[
d = \sum_{r^t \in \text{Box}(d)} \lambda_t r^t, \quad \lambda_t \in \mathbb{Z}_+
\]

and let \(\mathcal{R}(d, \lambda) = \{r^t \in \text{Box}(d) : \lambda_t \neq 0\}\) be the set of used update vectors.
Step 1: Representation of observed responses

- Time series \( (x^0; x^1 \ldots, x^k) \) may not consist of consecutive system states, as between \((x^j, x^{j+1}) \in \mathcal{X}'\), there can be non-observed intermediate states.
- To reach \( x^{j+1} \) from \( x^j \), we then need a sequence of transitions \( t_i, \ldots, t_k \) whose update vectors satisfy \( x^{j+1} - x^j = r^{t_i} + \ldots + r^{t_k} \).

**Theorem (Durzinsky, W., Weismantel 2008)**

To represent the difference \( d = x^{j+1} - x^j \), use only sign-compatible vectors from

\[
\text{Box}(d) = \left\{ \mathbf{r} \in \mathbb{Z}^{|P|} : \begin{cases} 
0 \leq r_p \leq d_p & \text{if } d_p > 0 \\
d_p \leq r_p \leq 0 & \text{if } d_p < 0 \\
r_p = 0 & \text{if } d_p = 0 \\
\sum_{p \in P'} r_p = 0 & \forall P' \in \mathcal{I}_P
\end{cases} \right\} \setminus \{0\}.
\]

For each \( d = x^{j+1} - x^j \), determine all solutions \( \lambda \in \Lambda(d) \) with

\[
d = \sum_{r^t \in \text{Box}(d)} \lambda_t r^t, \quad \lambda_t \in \mathbb{Z}_+
\]

and let \( \mathcal{R}(d, \lambda) = \{ r^t \in \text{Box}(d) : \lambda_t \neq 0 \} \) be the set of used update vectors.
Step 1: Representation of observed responses

- Time series \((x^0; x^1 \ldots, x^k)\) may not consist of consecutive system states, as between \((x^j, x^{j+1}) \in \mathcal{X}'\), there can be non-observed intermediate states.
- To reach \(x^{j+1}\) from \(x^j\), we then need a sequence of transitions \(t_i, \ldots, t_k\) whose update vectors satisfy \(x^{j+1} - x^j = r^{t_i} + \ldots + r^{t_k}\).

**Theorem (Durzinsky, W., Weismantel 2008)**

To represent the difference \(d = x^{j+1} - x^j\), use only sign-compatibel vectors from

\[
\text{Box}(d) = \left\{ r \in \mathbb{Z}^{|P|} : \begin{array}{l}
 0 \leq r_p \leq d_p & \text{if } d_p > 0 \\
d_p \leq r_p \leq 0 & \text{if } d_p < 0 \\
r_p = 0 & \text{if } d_p = 0 \\
\sum_{p \in P'} r_p = 0 & \forall P' \in \mathcal{I}_P
\end{array} \right\} \setminus \{0\}.
\]

For each \(d = x^{j+1} - x^j\), determine all solutions \(\lambda \in \Lambda(d)\) with

\[
d = \sum_{r^t \in \text{Box}(d)} \lambda_t r^t, \quad \lambda_t \in \mathbb{Z}_+
\]

and let \(\mathcal{R}(d, \lambda) = \{r^t \in \text{Box}(d) : \lambda_t \neq 0\}\) be the set of used update vectors.
Step 1: Representation of observed responses (cont.)

- Every permutation \( \pi = (r_{t_1}, \ldots, r_{t_m}) \) of the elements of a set \( \mathcal{R}(d^j, \lambda) \) yields a sequence of intermediate states \( x^j = y^1, y^2, \ldots, y^m, y^{m+1} = x^{j+1} \) with

\[
\sigma_\pi, \lambda (x^j, d^j) = ((y^1, r_{t_1}), (y^2, r_{t_2}), \ldots, (y^m, r_{t_m})). 
\]

- Every sequence \( \sigma \) respects monotonicity and induces a priority relation \( \mathcal{O}_\sigma \).

Example

For the running example we obtain as sequences

\[
\begin{align*}
x^0 &\xrightarrow{r_{4.2}} x^2, &\quad x^0 &\xrightarrow{d^4} x^4, \\
x^1 &\xleftarrow{r_{4.1}} x^0, &\quad x^1 &\xleftarrow{r_{1.2}} x^2, \\
x^2 &\xrightarrow{r_{4.1}} x^5 &\quad x^5 &\xleftarrow{r_{4.2}} x^2, \\
x^3 &\xrightarrow{d^3 = d^6} x^8 &\quad x^7 &\xleftarrow{r_{4.2}} x^6, \\
x^6 &\xrightarrow{r_{4.1}} x^3 &\quad x^6 &\xrightarrow{r_{4.2}} x^3, \\
x^9 &\xrightarrow{r_{1.1}} x^0 &\quad x^9 &\xrightarrow{d^1} x^0, \\
x^{10} &\xleftarrow{r_{1.2}} x^9 &\quad x^{10} &\xleftarrow{d^4 = d^5} x^3, \\
x^{11} &\xrightarrow{R^1} x^3 &\quad x^{11} &\xrightarrow{d^3} x^4
\end{align*}
\]

with \( x^9 = (1, 0, 0, 1, 0, 0)^T \), \( x^{10} = (0, 1, 1, 0, 1, 0)^T \) and \( x^{11} = (0, 1, 1, 0, 0, 0)^T \).
Step 2: Detecting priority conflicts between sequences

Sequences and their conflicts

Sequences $\sigma$ and $\sigma'$ are in **priority conflict** if there are update vectors $r^t \neq r^{t'}$ and intermediate states $y, y'$ such that both $t, t'$ are enabled at $y, y'$ but $(y, r^t) \in \sigma$ and $(y', r^{t'}) \in \sigma'$ (since this implies $t > t'$ in $O_\sigma$ but $t' > t$ in $O_{\sigma'}$).

**weak priority conflict (WPC)**

if $y \neq y'$, the priority conflict can be resolved by adding appropriate control-arcs

**strong priority conflict (SPC)**

if $y = y'$, the priority conflict cannot be resolved by adding appropriate control-arcs

Note: we have a SPC between the trivial sequence $\sigma(x^k, 0)$ for any terminal state $x^k$ and any sequence $\sigma$ containing $x^k$ as intermediate state.

Example: $\sigma_{\pi_1, \chi^2}(x^1, d^1) = ((x^1, r^{1.1}), (x^0, r^{1.2}))$ and $\sigma(x^0, 0)$ are in SPC.

Task: select one sequence per $d^j$ s.t. no SPCs occur between them.
Step 2: Detecting priority conflicts between sequences

**Sequences and their conflicts**

Sequences $\sigma$ and $\sigma'$ are in **priority conflict** if there are update vectors $r^t \neq r^{t'}$ and intermediate states $y, y'$ such that both $t, t'$ are enabled at $y, y'$ but $(y, r^t) \in \sigma$ and $(y', r^{t'}) \in \sigma'$ (since this implies $t > t'$ in $\mathcal{O}_\sigma$ but $t' > t$ in $\mathcal{O}_{\sigma'}$).

- **weak priority conflict (WPC)**
  - if $y \neq y'$, the priority conflict can be resolved by adding appropriate control-arcs

- **strong priority conflict (SPC)**
  - if $y = y'$, the priority conflict cannot be resolved by adding appropriate control-arcs

**Note:** we have a SPC between the trivial sequence $\sigma(x^k, 0)$ for any terminal state $x^k$ and any sequence $\sigma$ containing $x^k$ as intermediate state.

**Example:** $\sigma_{\pi_1, \lambda^2}(x^1, d^1) = ((x^1, r^{1.1}), (x^0, r^{1.2}))$ and $\sigma(x^0, 0)$ are in SPC.

**Task:** select one sequence per $d'$ s.t. no SPCs occur between them.
Step 2: Detecting priority conflicts between sequences

Priority conflict graph \( G = (V_D \cup V_{\text{term}}, E_D \cup E_W \cup E_S) \)

The nodes correspond to sequences:
- \( V_D \): the sequence \( \sigma_{\pi,\lambda}(x^j, d^j) \) for all permutations \( \pi \) of \( R(d^j, \lambda) \)
- \( V_{\text{term}} \): the trivial sequence \( \sigma(x^k, 0) \) for all terminal states \( x^k \)

The edges correspond to their priority conflicts:
- \( E_D \): all SPCs between two sequences \( \sigma, \sigma' \) from the same difference vector
- \( E_S \): all SPCs between sequences \( \sigma, \sigma' \) from distinct difference vectors
- \( E_W \): all WPCs between sequences \( \sigma, \sigma' \) from distinct difference vectors

In \( G \), there are 8 possible selections \( S \) avoiding SPCs.
Step 2: Detecting priority conflicts between sequences

Priority conflict graph $G = (V_D \cup V_{term}, E_D \cup E_W \cup E_S)$

- The nodes correspond to sequences:
  - $V_D$: the sequence $\sigma_\pi, \lambda(x^j, d^j)$ for all permutations $\pi$ of $R(d^j, \lambda)$
  - $V_{term}$: the trivial sequence $\sigma(x^k, 0)$ for all terminal states $x^k$

- The edges correspond to their priority conflicts:
  - $E_D$: all SPCs between two sequences $\sigma, \sigma'$ from the same difference vector
  - $E_S$: all SPCs between sequences $\sigma, \sigma'$ from distinct difference vectors
  - $E_W$: all WPCs between sequences $\sigma, \sigma'$ from distinct difference vectors

In $G$, there are 8 possible selections $S$ avoiding SPCs.
Step 3: Constructing $\mathcal{X}'$-deterministic Petri nets

Composition of standard networks

For every selected set $S$, we obtain a standard network $\mathcal{P}_S$ by deriving transitions from the update vectors in the sets $\mathcal{R}(\mathcal{d}^j, \mathcal{\lambda})$ of the selected sequences $\sigma \in S$.

Example

In solution $S_1$ selecting all canonical sequences $\sigma_1(x^j, \mathcal{d}^j)$, there are three WPCs:

- between $\sigma_1(x^2, \mathcal{d}^2)$ and $\sigma(x^0, 0)$ since $\mathcal{d}^2, 0$ are enabled at $x^2, x^0$
- between $\sigma_1(x^3, \mathcal{d}^3)$ and $\sigma(x^0, 0)$ since $\mathcal{d}^3, 0$ are enabled at $x^3, x^0$
- between $\sigma_1(x^7, \mathcal{d}^3)$ and $\sigma(x^0, 0)$ since $\mathcal{d}^3, 0$ are enabled at $x^7, x^0$
Step 3: Constructing $\mathcal{X}'$-deterministic Petri nets (cont.)

A WPC$(\sigma, \sigma')$ due to $(y, r^t) \in \sigma$ and $(y', r^{t'}) \in \sigma'$ can be resolved by

- either disabling $t$ at state $y'$ (and adding priority $t > t'$),
- or disabling $t'$ at state $y$ (and adding priority $t < t'$).

This can be done by inserting control-arcs from $\text{CA}(\sigma, \sigma')$ which partitions into a subset $\text{CA}_{t > t'}(\sigma, \sigma')$ containing

- a read-arc $(p, t)$ with weight $w(p, t) > y'_p$ for all $p$ with $y_p > y'_p$,
- an inhibitor-arc $(p, t)$ with $w(p, t) < y_p$ for all $p$ with $y_p < y'_p$,

and a subset $\text{CA}_{t < t'}(\sigma, \sigma')$ defined analogously.

Remark: If one of $y, y'$ is a terminal state, say $y'$, one of the alternatives is not possible, then $t$ has to be disabled at $y'$ and $t > t' = 0$ holds automatically.

- Inserting one control-arc from each $\text{CA}(\sigma, \sigma')$ resolves all WPCs.
- The obtained extended Petri nets can be made $\mathcal{X}'$-deterministic by adding the requested priorities $t > t'$ or $t < t'$. 
Step 3: Constructing $\mathcal{X}'$-deterministic Petri nets (cont.)

A WPC($\sigma, \sigma'$) due to $(y, r^t) \in \sigma$ and $(y', r^{t'}) \in \sigma'$ can be resolved by

- either disabling $t$ at state $y'$ (and adding priority $t > t'$),
- or disabling $t'$ at state $y$ (and adding priority $t < t'$).

This can be done by inserting control-arcs from $\text{CA}(\sigma, \sigma')$ which partitions into a subset $\text{CA}_{t > t'}(\sigma, \sigma')$ containing

- a read-arc $(p, t)$ with weight $w(p, t) > y'_p$ for all $p$ with $y_p > y'_p$,
- an inhibitor-arc $(p, t)$ with $w(p, t) < y_p$ for all $p$ with $y_p < y'_p$,

and a subset $\text{CA}_{t < t'}(\sigma, \sigma')$ defined analogously.

**Remark:** If one of $y, y'$ is a terminal state, say $y'$, one of the alternatives is not possible, then $t$ has to be disabled at $y'$ and $t > t' = 0$ holds automatically.

- Inserting one control-arc from each $\text{CA}(\sigma, \sigma')$ resolves all WPCs.
- The obtained extended Petri nets can be made $\mathcal{X}'$-deterministic by adding the requested priorities $t > t'$ or $t < t'$. 

A. Wagler
June 23, 2014 17 / 29
Step 3: Constructing $\mathcal{X}'$-deterministic Petri nets (cont.)

A WPC($\sigma, \sigma'$) due to $(y, r^t) \in \sigma$ and $(y', r^{t'}) \in \sigma'$ can be resolved by

- either disabling $t$ at state $y'$ (and adding priority $t > t'$),
- or disabling $t'$ at state $y$ (and adding priority $t < t'$).

This can be done by inserting control-arcs from $\text{CA}((\sigma, \sigma'))$ which partitions into a subset $\text{CA}_{t > t'}((\sigma, \sigma'))$ containing

- a read-arc $(p, t)$ with weight $w(p, t) > y'_p$ for all $p$ with $y_p > y'_p$, 
- an inhibitor-arc $(p, t)$ with $w(p, t) < y_p$ for all $p$ with $y_p < y'_p$,

and a subset $\text{CA}_{t < t'}((\sigma, \sigma'))$ defined analogously.

**Remark:** If one of $y, y'$ is a terminal state, say $y'$, one of the alternatives is not possible, then $t$ has to be disabled at $y'$ and $t > t' = 0$ holds automatically.

- Inserting one control-arc from each $\text{CA}((\sigma, \sigma'))$ resolves all WPCs.
- The obtained extended Petri nets can be made $\mathcal{X}'$-deterministic by adding the requested priorities $t > t'$ or $t < t'$.
Outline

1. Petri Nets and Extensions
2. Reconstruction Approach (Sketch)
3. Minimality Aspect
4. Integrating Prior Knowledge
5. Summary and Conclusions
Inserting minimal sets of control-arcs

In a standard network $\mathcal{P}_S$, we have to add control-arcs that resolve all WPCs:

- Hereby, inserting one control-arc from $\text{CA}(\sigma, \sigma')$ resolves $\text{WPC}(\sigma, \sigma')$, but possibly also another WPC in $\mathcal{P}_S$.

- Let $A_S$ be the union of all the control-arcs from the sets $\text{CA}(\sigma, \sigma')$ for all $\text{WPC}(\sigma, \sigma')$ in $\mathcal{P}_S$.

- To resolve the WPCs in $\mathcal{P}_S$, we have to find subsets $A'$ of $A_S$ hitting each $\text{CA}(\sigma, \sigma')$.

- Non-minimal hitting sets yield extended Peri nets with unnecessary control-arcs and, thus, being not minimal.

**Theorem**

All minimal extended Petri nets based on $\mathcal{P}_S$ can be obtained by computing all minimal hitting sets in $A_S$. 

A. Wagler

June 23, 2014 19 / 29
Inserting minimal sets of control-arcs

In a standard network $\mathcal{P}_S$, we have to add control-arcs that resolve all WPCs:

- Hereby, inserting one control-arc from $\text{CA}(\sigma, \sigma')$ resolves $\text{WPC}(\sigma, \sigma')$, but possibly also another WPC in $\mathcal{P}_S$.

- Let $\mathcal{A}_S$ be the union of all the control-arcs from the sets $\text{CA}(\sigma, \sigma')$ for all $\text{WPC}(\sigma, \sigma')$ in $\mathcal{P}_S$.

- To resolve the WPCs in $\mathcal{P}_S$, we have to find subsets $\mathcal{A}'$ of $\mathcal{A}_S$ hitting each $\text{CA}(\sigma, \sigma')$.

- Non-minimal hitting sets yield extended Petri nets with unnecessary control-arcs and, thus, being not minimal.

**Theorem**

All **minimal** extended Petri nets based on $\mathcal{P}_S$ can be obtained by computing all **minimal** hitting sets in $\mathcal{A}_S$. 
Example

For the standard network $\mathcal{P}_{S_1}$ with $T_1 = \{d^1, d^2, d^3 = d^6, d^4 = d^5\}$, we have:

<table>
<thead>
<tr>
<th></th>
<th>$(P_{FR}, d^2) \in A_L$</th>
<th>$(P_R, d^2) \in A_R$</th>
<th>$(P_{FR}, d^3) \in A_I$</th>
<th>$(P_R, d^3) \in A_R$</th>
<th>$(G, d^3) \in A_R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPC1</td>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WPC2</td>
<td></td>
<td></td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>WPC3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\times$</td>
</tr>
</tbody>
</table>

Two minimal hitting sets result in two minimal extended Petri nets based on $\mathcal{P}_{S_1}$:

The total number of minimal $\chi'$-deterministic extended Petri nets is 66.
Indecomposability of difference vectors

In Step 1 *Representing the observed responses*, the set $\text{Box}(d^j)$ of update vectors to refine the sequence between $(x^j, x^{j+1}) \in \mathcal{X}'$ can be restricted if

- $d^j$ exactly corresponds to a well-known biochemical reaction (including the correct stoichiometry) or to a well-known mechanism (that a certain trigger is detected by a suitable receptor),
- experiments have shown that subsets of the input components of $d^j$ do not lead to the observed response,
- $d^j$ is treated as black box-like reaction where only input and output matter, but not the intermediate mechanism due to the chosen level of detail.

**Indecomposability of difference vectors**

Exclude the corresponding response $d^j \in D$ from decomposition, just define:

$$\text{Box}(d^j) := \{d^j\}$$
Indecomposability of difference vectors

In Step 1 *Representing the observed responses*, the set $\text{Box}(d^j)$ of update vectors to refine the sequence between $(x^j, x^{j+1}) \in \mathcal{X}'$ can be restricted if

- $d^j$ exactly corresponds to a well-known biochemical reaction (including the correct stoichiometry) or to a well-known mechanism (that a certain trigger is detected by a suitable receptor),
- experiments have shown that subsets of the input components of $d^j$ do not lead to the observed response,
- $d^j$ is treated as black box-like reaction where only input and output matter, but not the intermediate mechanism due to the chosen level of detail.

Excluding the corresponding response $d^j \in D$ from decomposition, just define:

$$\text{Box}(d^j) := \{d^j\}$$
Example

Since the light-dependent reactions of the photoreceptor are so much faster than the subsequent processes, the difference vectors $d^1$, $d^4$ and $d^5$ describing the photoconversions will not be decomposed due to the chosen level of detail.

Thus, only solutions based on $P_{S_1}$ with $T_1 = \{d^1, d^2, d^3 = d^6, d^4 = d^5\}$ remain. Accordingly, the total number of solutions reduces from 66 to 2.
Treating equal difference vectors in the same way

If two difference vectors $d^i, d^j \in D$ are equal, we have $\text{Box}(d^i) = \text{Box}(d^j)$ and $\Lambda(d^i) = \Lambda(d^j)$. In Step 2 *Detecting priority conflicts between sequences*, it is natural to require that $d^i = d^j$ are decomposed in the same way: using the same

- $\lambda \in \Lambda(d^i) = \Lambda(d^j)$ means that the same subset of molecules involved in a reaction will not interact according to different mechanisms,

- permutation $\pi$ of the elements in $R(d^i, \lambda) = R(d^j, \lambda)$ means that the order in which the reactions are applied reflects the relative rates of the reactions in $R(d^i, \lambda) = R(d^j, \lambda)$, which leads to the same priorities within the resulting sequences.

We call $\sigma_{\pi, \lambda}(x^i, d^i)$ and $\sigma'_{\pi, \lambda}(x^j, d^j)$ twin sequences if $d^i = d^j$ and the same $\lambda, \pi$ has been used.

Treating equal difference vectors in the same way

Force that twin sequences are always selected together by adding strong priority conflicts in $G$ between all other sequences stemming from a pair $d^i = d^j \in D$. 
Treating equal difference vectors in the same way

If two difference vectors $d^i, d^j \in D$ are equal, we have $\Box(d^i) = \Box(d^j)$ and $\Lambda(d^i) = \Lambda(d^j)$. In Step 2 Detecting priority conflicts between sequences, it is natural to require that $d^i = d^j$ are decomposed in the same way: using the same

- $\lambda \in \Lambda(d^i) = \Lambda(d^j)$ means that the same subset of molecules involved in a reaction will not interact according to different mechanisms,

- permutation $\pi$ of the elements in $R(d^i, \lambda) = R(d^j, \lambda)$ means that the order in which the reactions are applied reflects the relative rates of the reactions in $R(d^i, \lambda) = R(d^j, \lambda)$, which leads to the same priorities within the resulting sequences.

We call $\sigma^i_{\pi, \lambda}(x^i, d^i)$ and $\sigma^j_{\pi, \lambda}(x^j, d^j)$ twin sequences if $d^i = d^j$ and the same $\lambda$, $\pi$ has been used.

Treating equal difference vectors in the same way

Force that twin sequences are always selected together by adding strong priority conflicts in $G$ between all other sequences stemming from a pair $d^i = d^j \in D$. 

Treating equal difference vectors in the same way: Example

Example

In our running example, we have $d^3 = d^6$ and $d^4 = d^5$. The latter vectors can be decomposed in different ways, among the resulting sequences we have $\sigma_1(x^5, d^4)$, $\sigma_1(x^6, d^5)$ and $\sigma_3(x^5, d^4)$, $\sigma_3(x^6, d^5)$ as pairs of twin sequences.

Forcing to select these pairs together rules out 4 of the previously possible 8 selections. Accordingly, the total number of solutions reduces from 66 to 18.
Integrating knowledge on relative reaction rates

In Step 3 *Constructing $\mathcal{X}'$-deterministic Petri nets*, to resolve a $\text{WPC}(\sigma, \sigma')$ between $\sigma, \sigma'$ involving update vectors $r^t \neq r^{t'}$ and intermediate states $y \neq y'$, either transition $t$ has to be disabled at $y'$ or transition $t'$ at $y$, while the decision between $t$ and $t'$ on the other state can be handled by a priority.

Prior knowledge about the **relative reaction rates** of $t$ and $t'$ can help to decide whether $t > t'$ or $t < t'$ better reflects the reality:

- If $y'$ is a **terminal state** and $\sigma' = \sigma(y', 0)$ its trivial sequence, then $t > 0$ holds automatically at $y$, but $t$ has to be disabled at $y'$ using control-arcs from $\text{CA}_{t>0}(\sigma, \sigma')$ whereas $\text{CA}_{t<0}(\sigma, \sigma')$ is empty.

- For a $\text{WPC}(\sigma, \sigma')$ where the **time-scale** of the corresponding experimental observations clearly differs, deduce the correct priority $t > t'$ or $t < t'$ and reduce $\text{CA}(\sigma, \sigma')$ accordingly either to $\text{CA}_{t>t'}(\sigma, \sigma')$ or to $\text{CA}_{t'>t}(\sigma, \sigma')$. 
In Step 3 *Constructing $\mathcal{X}'$-deterministic Petri nets*, to resolve a $\text{WPC}(\sigma, \sigma')$ between $\sigma, \sigma'$ involving update vectors $r^t \neq r^{t'}$ and intermediate states $y \neq y'$, either transition $t$ has to be disabled at $y'$ or transition $t'$ at $y$, while the decision between $t$ and $t'$ on the other state can be handled by a priority.

Prior knowledge about the relative reaction rates of $t$ and $t'$ can help to decide whether $t > t'$ or $t < t'$ better reflects the reality:

- **If** $y'$ is a **terminal state** and $\sigma' = \sigma(y', 0)$ its trivial sequence, then $t > 0$ holds automatically at $y$, but $t$ has to be disabled at $y'$ using control-arcs from $\text{CA}_{t>0}(\sigma, \sigma')$ whereas $\text{CA}_{t<0}(\sigma, \sigma')$ is empty.

- For a $\text{WPC}(\sigma, \sigma')$ where the **time-scale** of the corresponding experimental observations clearly differs, deduce the correct priority $t > t'$ or $t < t'$ and reduce $\text{CA}(\sigma, \sigma')$ accordingly either to $\text{CA}_{t>t'}(\sigma, \sigma')$ or to $\text{CA}_{t'>t}(\sigma, \sigma')$. 

*Integrating knowledge on relative reaction rates*
Example

In our running example, we have different time-scales for the experimental observations as

- \( d^1 \) and \( d^4 = d^5 \) need only milliseconds to occur,
- \( d^2 \) needs about 1 hour to occur, and
- \( d^3 = d^6 \) need at least 10 hours

which implies \( d^1, d^4, d^5 > d^2 > d^3, d^6 \).

Consequently, we can reduce the sets \( CA(\sigma, \sigma') \) of four WPCs as follows:

- for WPC4 between \( \sigma_3(x^1, d^1), \sigma(x^7, d^3) \) to \( CA_{r1.2 > d^3}(WPC4) \);
- for WPC7 between \( \sigma_3(x^5, d^4), \sigma(x^2, d^2) \) to \( CA_{r4.2 > d^2}(WPC7) \);
- for WPC8 between \( \sigma_3(x^5, d^4), \sigma(x^3, d^3) \) to \( CA_{r4.2 > d^3}(WPC8) \);
- for WPC9 between \( \sigma_3(x^6, d^4), \sigma(x^3, d^3) \) to \( CA_{r4.2 > d^3}(WPC9) \).

Accordingly, the number of solutions decreases from 66 to 36.
1 Petri Nets and Extensions

2 Reconstruction Approach (Sketch)

3 Minimality Aspect

4 Integrating Prior Knowledge

5 Summary and Conclusions
Summary and Conclusions

- ANR aims at reconstructing all $\mathcal{X}'$-deterministic extended Petri nets that fit given experimental data $\mathcal{X}'$ which typically results in large solution sets.

- To keep this solution set reasonably small while still guaranteeing its completeness, we firstly generate only **minimal** solutions.

- To integrate **prior knowledge** in the reconstruction procedure to make the “right decisions” in some intermediate steps, we extend the input from $(P, I_P, \mathcal{X}')$ to $(P, I_P, \mathcal{X}', D_{in}, D_{eq}, O_D)$ where
  - $D_{in}$ contains all indecomposable difference vectors;
  - $D_{eq}$ contains all pairs of equal difference vectors to be treated in the same way;
  - $O_D$ contains relative reaction rates between difference vectors;
which results in substantial reductions of solution alternatives.

**Impact of ANR**
Models computed by an exact reconstruction approach have predictive ability due to the completeness of the solution set guaranteed by mathematical **proofs**.
Summary and Conclusions

- ANR aims at reconstructing all $\mathcal{X}'$-deterministic extended Petri nets that fit given experimental data $\mathcal{X}'$ which typically results in large solution sets.

- To keep this solution set reasonably small while still guaranteeing its completeness, we firstly generate only minimal solutions.

- To integrate prior knowledge in the reconstruction procedure to make the “right decisions” in some intermediate steps, we extend the input from $(P, I_P, \mathcal{X}')$ to $(P, I_P, \mathcal{X}', D_{in}, D_{eq}, O_D)$ where
  - $D_{in}$ contains all indecomposable difference vectors;
  - $D_{eq}$ contains all pairs of equal difference vectors to be treated in the same way;
  - $O_D$ contains relative reaction rates between difference vectors;

  which results in substantial reductions of solution alternatives.

Impact of ANR

Models computed by an exact reconstruction approach have predictive ability due to the completeness of the solution set guaranteed by mathematical proofs.