Handling Dynamic Changes in Petri Net Models of Workflow Processes

Third Annual Progress Seminar
By
Ahana Pradhan
(113050039)

Working under the guidance of
Prof. Rushikesh K. Joshi

Department of Computer Science & Engineering
Indian Institute of Technology Bombay
Powai, Mumbai-400076, India
Dynamic Migration of Workflows

Dynamic instance migration needs to be facilitated for workflows in order to reflect real-world changes in automated processes.

**Consistency model:**
Equivalence mapping from current state of old workflow to the migrating state of the new workflow.
Dynamic Evolution of Workflows

Reimbursement Workflow in an Academic Institute

Old process:

New process:
Given a marking in the old net (running instance), goal is to obtain a marking in the new net (migrated instance)
History equivalence (Compliance) [Ellis et al. COCS’95, Rinderle et al. ER’08]

History: t1, t2, t3
Delete-purged Compliance [Rinderle et al. ER’08]

Delete-purged History: t1, t3
History-based Consistency

Loop-purged Compliance  [Rinderle et al. ER’08, Sun et al., IST’09]

Common Reduced History: t1, t3
Valid transfer [Van der Aalst, ISF 01]

Marking-based Consistency

Marking \{ p2, p5 \}
Notable Existing Solution Approaches

Change region [Van der Aalst, ISF’01; Sun et al., IST’09]

State Space [Agostini et al., CSCW’00]
Outline

1. Algorithm for Trace equivalence token transportation
2. Lookahead Trace based consistency models
3. Conclusion
4. Future works
Yo-Yo Algorithm
**Consistency**
preservation of history (done tasks in old $\leftrightarrow$ done tasks in new)

**Validity**
reachability of marking in the new net

**Inconsistent!**
Missing token in parallel branch

**Invalid!**

**Correct**

Done task ty

Done task tx
Yo-Yo Approach

Token transportation by: Folding, transport, Unfolding

Old Net:

New Net:

Pre-computed transportation
Yo-Yo Approach: Folding

Old Net:

New Net:
Folding: Original Nets

Old Net:

New Net:

Pre-computed transportation
Folding: Step 1

Old Net:

New Net:
Folding: Step 2

Old Net:

New Net:
Folding: Step 3

Old Net:

New Net:
Transport: Step 1

New Net:

transport

Old Net:
Unfolding and Transport: Step 2

Old Net:

New Net:

transport
Unfolding and Transport: Step 3

Old Net:

New Net:

transport
Unfolding: Step 4

Old Net:

New Net:

No transport required
<table>
<thead>
<tr>
<th>Transportation between which two patterns</th>
<th>Peer patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>When such hand-in-hand folding of nets are possible</td>
<td>Yo-Yo compatibility</td>
</tr>
<tr>
<td>Which pattern to fold when</td>
<td>Folding order, obtained from Derivation Trees</td>
</tr>
<tr>
<td>What all pre-computed transportations cover the scope</td>
<td>Token transportation Catalog</td>
</tr>
<tr>
<td>Pattern</td>
<td>Specification</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
</tr>
<tr>
<td>SEQ:</td>
<td>tx ty</td>
</tr>
<tr>
<td>AND:</td>
<td>(tx) (ty)</td>
</tr>
<tr>
<td>XOR:</td>
<td>[tx] [ty]</td>
</tr>
</tbody>
</table>

silent transitions model gateway logic
Input nets

*Composition of primitive patterns*: sequence or nesting

\[
\begin{align*}
\text{Start} & \rightarrow \text{SEQ} \\
\text{SEQ} & \rightarrow \text{SEQ} \; t \; \text{SEQ} \; t \; \text{SEQ} \mid \text{SEQ} \; \text{AND} \; \text{SEQ} \mid \text{SEQ} \; \text{XOR} \; \text{SEQ} \mid e \\
\text{AND} & \rightarrow ( \text{SEQ} \; t \; \text{SEQ} ) \; ( \text{SEQ} \; t \; \text{SEQ} ) \\
\text{XOR} & \rightarrow [ \text{SEQ} \; t \; \text{SEQ} ] \; [ \text{SEQ} \; t \; \text{SEQ} ]
\end{align*}
\]

Example derivation

\[
\begin{align*}
\text{Start} & \rightarrow \text{SEQ} \rightarrow \text{SEQ} \; t_1 \; \text{SEQ} \; t_8 \; \text{SEQ} \rightarrow t_1 \; \text{AND} \; t_8 \\
& \rightarrow t_1 \; ( \text{SEQ} \; t_2 \; \text{SEQ} ) \; ( \text{SEQ} \; t_7 \; \text{SEQ} ) \; t_8 \\
& \rightarrow t_1 \; ( t_2 \; \text{SEQ} \; \text{AND} \; \text{SEQ} ) \; ( \text{SEQ} \; \text{AND} \; \text{SEQ} \; t_7 ) \; t_8 \\
& \rightarrow t_1 \; ( t_2 \; ( \text{SEQ} \; t_3 \; \text{SEQ} ) \; ( \text{SEQ} \; t_4 \; \text{SEQ} ) ) \; ( ( \text{SEQ} \; t_5 \; \text{SEQ} ) \; ( \text{SEQ} \; t_6 \; \text{SEQ} ) \; t_7 ) \; t_8 \\
& \rightarrow t_1 \; ( t_2 \; ( t_3 \; ( t_4 ) ) \; ( ( t_5 ) \; ( t_6 ) \; t_7 ) \; t_8
\end{align*}
\]
Input nets

Folding steps follow such order of derivation..
**Derivation Trees**

**Primitive Block**

- $p_1 \xrightarrow{tx} p_2 \xrightarrow{ty} p_3$
- $p_1 \xrightarrow{q_1 \, tx} p_2 \xrightarrow{ty} p_3$
- $p_1 \xrightarrow{q_3 \, ty} p_2 \xrightarrow{tx} p_3$

**Derivation Tree**

- $p_1 \xrightarrow{tx} p_2 \xrightarrow{ty} p_3$
- $p_1 \xrightarrow{q_1 \, tx} p_2 \xrightarrow{ty} p_3$
- $p_1 \xrightarrow{q_3 \, ty} p_2 \xrightarrow{tx} p_3$

**Grammar Non-terminals**

- $\bigcirc$ - SEQ
- $\bigtriangleup$ - AND
- $\bigotimes$ - XOR

**Triplets:**

Left-right positioning w.r.t. parent does not matter.
Derivation Trees
**Colored Derivation Trees**

<table>
<thead>
<tr>
<th>Node Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leaf/Non-leaf</td>
<td>Unmarked folded/unfolded place</td>
</tr>
<tr>
<td>Leaf</td>
<td>marked place in net</td>
</tr>
<tr>
<td>Non-leaf</td>
<td>abstraction of null-executed subnet</td>
</tr>
<tr>
<td>Non-leaf</td>
<td>abstraction of subnets where at least one labeled transition has been fired</td>
</tr>
</tbody>
</table>

**Red node:**
Color parent red

**Black node:**
Check if any transition Sibling has color at right, If yes, color parent red; Else color parent black
Pattern Alterations

Old Net:

New Net:
Peer Patterns
Yo-Yo compatibility

Both can generate the same sequence $t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 \rightarrow \textit{Folding order exists}$

$\text{Yield of } r_1 = t_1 \{ t_2 \{ t_3, t_4 \}, \{ t_5, t_6 \}, t_7 \} t_8$

$\text{Yield of } r_2 = t_1 t_2 t_3 t_4 \{ t_5, t_6 \} t_7 t_8$
Folding order

P1

P2

P3

P4

P1' = P1 - P1' + P3 - P3' + P4 - P4'

P2' = P2 - P2' + P3 - P3' + P4 - P4'

P3' = P1 - P1' + P2 - P2' + P3 - P3' + P4 - P4'

P4' = P1 - P1' + P2 - P2' + P3 - P3' + P4 - P4'
Pre-computed Token Transportation

Compatible yields
s1 s2 tx s3 s4 ty ...

s3 = s4 = ε

s3 = ε

(leaf)
Token Transportation Catalog
1. Color old tree
2. \(<p-q> \text{ be } 1^{st} \text{ peer patterns} \) to appear in folding order \(F\)
3. Color transfer between \(p, q\)
4. for each next \(<p-q> \text{ in } F\),
   if \(q\) has colored root,
   if \(p\) is colored,
   color transfer between \(p, q\)
   else
   \(\text{localPropagation}(q)\)
1. Color old tree
2. <p-q> be 1st peer patterns to appear in folding order F
3. Color transfer between p, q
4. for each next <p-q> in F,
   if q has colored root,
     if p is colored,
       color transfer between p, q
   else
     localPropagation(q)
Yo-Yo Algorithm

1. Color old tree
2. \(<p-q>\) be 1st peer patterns to appear in folding order \(F\)
3. Color transfer between \(p, q\)
4. for each next \(<p-q>\) in \(F\),
   if \(q\) has colored root,
   if \(p\) is colored,
   color transfer between \(p, q\)
else
   localPropagation(q)
1. Color old tree
2. Let (p-q) be the 1st peer patterns to appear in folding order F
3. Color transfer between p, q
4. For each next (p-q) in F,
   - If q has colored root,
     - If p is colored, color transfer between p, q
     - Else localPropagation(q)
Yo-Yo Algorithm

1. Color old tree
2. Let $p-q$ be the 1st peer patterns to appear in folding order $F$
3. Color transfer between $p$, $q$
4. For each next $p-q$ in $F$,
   - If $q$ has colored root, $false$
   - If $p$ is colored, color transfer between $p$, $q$
   - Else, localPropagation($q$)
Yo-Yo Algorithm

Max. no. of Transportation Steps = no. of patterns (linear time complexity)
1. Color old tree
2. \(<p-q>\) be 1st peer patterns to appear in folding order \(F\)
3. Color transfer between \(p\), \(q\)
4. for each next \(<p-q>\) in \(F\),
   if \(q\) has colored root,
   if \(p\) is colored,
   color transfer between \(p\), \(q\)
   else
   \(\text{localPropagation}(q)\)

Red root \(\rightarrow\) color rightmost child
1. Color old tree
2. \(<p-q>\) be 1st peer patterns to appear in folding order \(F\)
3. Color transfer between \(p, q\)
4. for each next \(<p-q>\) in \(F\),
   if \(q\) has colored root,
   if \(p\) is colored,
   color transfer between \(p, q\)
else
   localPropagation(q)
**Correctness**

**Catalog Completeness:**
Token transportation catalog is complete w.r.t. the 6 change patterns

**Lemma 1:**
For two Yo-Yo compatible derivation trees, consistent coloring between
The top peer patterns guaranties consistent coloring between their immediate child peer patterns

**Lemma 2:**
Lemma 1 can be repeated for all parent-child peer pairs across two Yo-Yo compatible derivation trees
### Correctness: Catalog Completeness

<table>
<thead>
<tr>
<th>Type of Node</th>
<th>Marking Status</th>
<th>Execution Status</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folded</td>
<td>Unmarked</td>
<td>Null/full-executed</td>
<td>Uncolored</td>
</tr>
<tr>
<td>Unfolded</td>
<td>Unmarked</td>
<td>NA</td>
<td>Uncolored</td>
</tr>
<tr>
<td>Folded</td>
<td>Marked</td>
<td>Null executed</td>
<td>Black</td>
</tr>
<tr>
<td>Unfolded</td>
<td>Marked</td>
<td>NA</td>
<td>Black</td>
</tr>
<tr>
<td>Folded</td>
<td>Marked</td>
<td>Partially executed</td>
<td>Red</td>
</tr>
<tr>
<td>Folded</td>
<td>Marked</td>
<td>Full executed</td>
<td>Red</td>
</tr>
</tbody>
</table>

**6 situations!**

1 marking, 2 x 4 x 4 = 32 situations
2 x 2 = 4 colorings
## Correctness: Catalog Completeness

<table>
<thead>
<tr>
<th>Pattern</th>
<th># valid markings</th>
<th># actual situations</th>
<th># colorings In derivation trees</th>
<th># non-migratable colorings</th>
<th># colorings where node type changes mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQ</td>
<td>3</td>
<td>28</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AND</td>
<td>6</td>
<td>420</td>
<td>20</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>XOR</td>
<td>6</td>
<td>116</td>
<td>12</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**Yield is**

\[
s_1 \{ s_2 \text{ tx } s_3, s_4 \text{ ty } s_5 \} s_6
\]

SEQ:
\[
s_1 s_2 \text{ tx } s_3 s_4 \text{ ty } s_5 s_6
\]

XOR:
\[
s_1 s_2 \text{ tx } s_3 s_6 \text{ or } s_1 s_4 \text{ ty } s_5 s_6
\]

**37 colorings in catalog**

**e.g. non-migratable**

**e.g. node type changes mapping**

**3rd APS**
Correctness: Lemma 1

Roots of two derivation trees are yield compatible. Consistent color transfer between the top patterns $P$ and $P'$ $\Rightarrow$ consistency ensured between their child peers $Q$ and $Q'$

<table>
<thead>
<tr>
<th></th>
<th>Root of $Q$</th>
<th>Root of $Q'$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Red</td>
<td>Red/uncolored</td>
</tr>
<tr>
<td></td>
<td>Black</td>
<td>Black/uncolored</td>
</tr>
<tr>
<td></td>
<td>uncolored</td>
<td>uncolored</td>
</tr>
</tbody>
</table>

Possible to refine root colors of $Q$ and $Q'$ consistently

Same relative Positions of $Q$ and $Q'$ w.r.t. $P$ and $P'$
Correctness: Lemma 2

Preservation of yield compatibility through folding order
Lookahead Consistency Models
# Lookahead Trace based Consistency

<table>
<thead>
<tr>
<th>Consistency Model Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong Lookahead</td>
<td>same lookahead trace sets of consistent marking</td>
</tr>
<tr>
<td>Accommodative Lookahead</td>
<td>old lookahead trace set preserved in new</td>
</tr>
<tr>
<td>Weak Lookahead</td>
<td>at least one old lookahead trace preserved in new</td>
</tr>
</tbody>
</table>

Lookahead trace: t2, t3

### Diagram

- **Strong**
- **Accommodative**
- **Weak**
Milk-products packaging:

Quality inspect → Polythene pack → Sealing → Label → Transport

Polythene pack, sealing, label, transport

Chocolate packaging:

Devan → Store → Polythene pack → Quality inspect → Sealing → Label → Transport

Polythene pack, sealing, label, transport

Dry fruit packaging:

Devan → Store → Quality inspect → Weigh & Insert → Cardboard pack → Polythene pack → Sealing → Label → Transport

Polythene pack, sealing, label, transport; Cardboard pack, sealing, ...
Accommodative lookahead

**Academic program:**

- **sem1:** admitted
- **sem2:** completed
- **sem3:** completed
- **sem4:** completed
- **degree completed**

**Home university Semester:**

- **orientation**
- **X**

\( X = \text{core credit courses for sem1, 2, 3} \)
\( X = \text{project for sem4} \)

**Orientation, reg., X, ob. grades**

**Foreign university Semester:**

- **Join**
- **orientation**
- **registration**
- **X**
- **Y**
- **obtain grades**
- **Leave**

\( X = \text{core credit courses} \)
\( Y = \text{core credit courses + electives} \)

**Orientation, reg., X, ob. Grades; Orientation, reg., Y, ob. Grades**
Weak lookahead

Old Curriculum:

New Curriculum:

gr1, gr2, gr3, sup. alloc., project, report; ...

gr1, gr2+backlog, gr3, sup. alloc., project, report; ...
Algo 1: Computing weak lookahead marking

1. Acyclic nets
2. No duplicate transitions
Algo 1: Computing weak lookahead marking

\{t_1t_3, t_2t_3\}
Algo 1: Computing weak lookahead marking

\{t1t3, t2t3\}
Algo 1: Computing weak lookahead marking

\{t_1 t_3, t_2 t_3\}
Traces = \{ t_1t_3, t_2t_3 \}  \quad \text{lookahead traces}
L = \{ t_1t_3, t_2t_3 \}  \quad \text{preserved lookahead traces}
S = \{ \{p_1'\}, \{p_2'\} \}  \quad \text{weak lookahead consistent marking}
Algo 2: Accept/Reject Branching

L = Polythene pack, sealing, label, transport
Algo 2: Accept/Reject Branching

L = Polythene pack, sealing, label, transport
Algo 2: Accept/Reject Branching

L = Polythene pack, sealing, label, transport

\( P_{\text{XOR}} = \{ p \} \)
Algo 2: Accept/Reject Branching

\[ L = \text{Polythene pack, sealing, label, transport} \]
\[ P_{\text{XOR}} = \{ p \} \]
\[ T_{\text{potential}} = \{ \text{cardboard pack, polythene pack} \} \]
Algo 2: Accept/Reject Branching

\[ L = \text{Polythene pack, sealing, label, transport} \]
\[ P_{\text{XOR}} = \{ p \} \]
\[ T_{\text{potential}} = \{ \text{cardboard pack, polythene pack} \} \]
\[ T_{\text{block}} = \{ \text{cardboard pack} \} \]
L ≠ { } → weak
+ |S| = 1, L = Traces → accommodative
+ $T_{\text{block}} = \{\}$ → strong
S = { } → no lookahead

|s| > 1 → no single marking can fire all preserved lookahead traces

Traces of lookahead traces
L preserved lookahead traces
S weak consistent markings
$T_{\text{block}}$ contradictory head-transitions
Practical Example: Resource Acquisition

Departmental Process

- Acquisition initiated
- Prepare acquisition list
- Estimate budget
- Ready to send for approval
- Apply to dept. office for fund
- Funds applied
- Recv. rejection
- Arranged fund from project
- Negotiate price
- Payment
- Drop acquisition
- Recv. delivery & invoice
- Acquisition complete
- Record acquisition details & catalog
- Finish

Central Library Process

- Acquisition initiated
- Prepare acquisition list
- Estimate budget
- Ready to send for approval
- Apply to acad. office for fund
- Funds applied
- Recv. approval
- Funds available
- Price & license agreed
- Negotiate price
- Sign license
- Payment
- License obtained
- Price agreed
- License recorded
- Create license record
- Payment complete
- Record delivery & invoice
- Acquisition complete
- Drop acquisition
- Record acquisition details & catalog
- Finish
Practical Example: Resource Acquisition

Departmental Process Instance

Re-engineered Process

1. Acquisition initiated
2. Prepare acquisition list
3. Estimate budget
4. Funds available
5. Price agreed
6. Payment
7. Funds applied
8. Recv. approval
9. Negotiate price
10. Funds available
11. Sign license
12. Price & license agreed
13. Payment
14. Funds available
15. Recv. order
16. Create license record
17. Acquisition complete
18. Activate e-resource
19. Acquisition complete
20. Record acquisition details & catalog
21. Finish
Practical Example: Resource Acquisition

(Algo 1)
L = negotiate price, Payment, Recv. delivery & invoice, Record details

Departmental Process Instance

Migrated Instance


**Practical Example: Resource Acquisition**

**Departmental Process Instance**

- **(Algo 1)**
  - \( L = \) negotiate price, Payment,Recv. delivery & invoice,Record details

**Migrated Instance**

- **(Algo 2)**
  - \( T_{block} = \) Negotiate price & license, Activate e-resource
Practical Example: Resource Acquisition

**Departmental Process Instance**

1. **acquisition initiated**
2. **prepare acquisition list**
3. **list ready**
4. **ready to send for approval**
5. **funds applied**
6. **arrange fund from project**
7. **arrange fund from IRCC**
8. **send to dept. office for fund**
9. **recv. approval**
10. **payment**
11. **negotiate price**
12. **price agreed**
13. **recv. delivery & invoice**
14. **record acquire details & catalog**
15. **fini**

**Algorithm 1 (Algo 1)**

$L = \text{negotiate price, Payment, Recv. delivery & invoice, Record details}$

**Algorithm 2 (Algo 2)**

$T_{block} = \text{Negotiate price \\ \\ Activate e-resource}$

**Inferences:**
- Accommodative
- Lookahead
- Consistency by schema
New approach to the token transportation problem by Catalog based modular solution by YoYo algorithm.

Embedding history in the catalog results in history equivalent solutions without computing them in runtime.

Novel approach of derivation tree and its coloring for representing net, markings along with the hierarchy of composition.

Structural analysis pushed to the schema level and linear runtime complexity for token transportation at instance level for trace equivalent migration.

Developed lookahead trace based consistency models with varying flexibility.

Demonstrated dynamic migration scenarios requiring future-based consistency notion, in contrast to trace based models.

Algorithms for computing lookahead consistent markings, and inferences regarding the class of consistency.

Support vs. enforcement of lookahead trace executions; Practical migration situation requiring lookahead enforcement.
Future Works

Consistency Models and Change Regions

Extending the scope of Yo-Yo Algorithm

Dynamic instance migration in distributed execution environment
Publications & Paper Presentations

1. [Full paper] Lookahead Consistency Models for Dynamic Migration of Workflow Processes

2. [Full paper] Catalog-based Token Transportation in Acyclic Block-Structured WF-nets


4. [Short paper] Token transportation in Petri net models of workflow patterns
THANK YOU