Extended Coloured Petri Nets with Structured Tokens – Formal Method for Distributed Systems

Khaoula Al Ali, Wolfgang Fengler, Bernd Däne, Alexander Pacholik

Faculty of Computer Science and Automation, Computer Architecture and Embedded Systems Group

http://www.tu-ilmenau.de/en/ra/
Outline

• Motivation
• Related work
  – Analysis methods
  – High level Petri nets
• High Coloured Petri Nets with Structured Tokens (HCPN-ST)
• Application example / case study
  – Modeling
  – Transformation & analysis
  – Results
• Conclusion & future work
Motivation – Distributed Systems

• Distributed protocols & algorithms are challenging for validation & verification (e.g. P2P systems)
  – Time modeling (naturals, intervals, statistic distribution)
  – Addressing schemes (relative, absolute, modulo)
  – Statistic parameters
    ➔ Difficult verifiable models
    ➔ Some effects are hard to express
      (abstract structures, addressing modes)

➔ Case-Study:
  optimized Petri net class for distributed protocols & algorithms
Motivation – Distributed Systems

• Distributed protocols & algorithms are challenging for validation & verification (e.g. P2P systems)
  – Time modeling (naturals, intervals, statistic distribution)
  – Addressing schemes (relative, absolute, modulo)
  – Statistic parameters
  ➔ Difficult verifiable models
  ➔ Some effects are hard to express
    (abstract structures, addressing modes)

➔ Case-Study:
  optimized Petri net class for
distributed protocols & algorithms
Motivation – Distributed Systems

- Distributed protocols & algorithms are challenging for validation & verification (e.g. P2P systems)
  - Time modeling (naturals, intervals, statistic distribution)
  - Addressing schemes (relative, absolute, modulo)
  - Statistic parameters
    - Difficult verifiable models
    - Some effects are hard to express
      (abstract structures, addressing modes)

- Case-Study:
  optimized Petri net class for distributed protocols & algorithms
Types of formal techniques:

1. Model only main mechanisms & apply exhaustive analysis
   - Petri nets
   - State charts

2. Model all mechanisms in detail & apply simulation, test sequences
   - Estelle
   - Lotos
   - SDL
Related work I – Formal techniques

Types of formal techniques:

1. Model only main mechanisms & apply exhaustive analysis
   - **Petri nets**
   - State charts

2. Model all mechanisms in detail & apply simulation, test sequences
   - Estelle
   - Lotos
   - SDL
Related work II – Petri net types

- **Petri nets (PN)**
  - Flexible method
  - Wide modeling & analysis support
  - Various extensions
  - But: large scale nets become unreadable

- **Coloured Petri nets (CPN) [Jensen92]**
  - Abstraction of (similar) structures
  - But: explicit colour transformation rules

- **Predicate/Transition nets (Pr/T-nets)[GenLau81]**
  - Implicitly defined colours (based on types)
  - Predicate based transformation rules
  (each representing sets of colour transformation rules)

Related work II – Petri net types

• **Petri nets (PN)**
  – Flexible method
  – Wide modeling & analysis support
  – Various extensions
  – But: large scale nets become unreadable

• **Coloured Petri nets (CPN) [Jensen92]**
  – Abstraction of (similar) structures
  – But: explicit colour transformation rules

• **Predicate/Transition nets (Pr/T-nets)[GenLau81]**
  – Implicitly defined colours (based on types)
  – Predicate based transformation rules
    (each representing sets of colour transformation rules)

Specifics of distributed systems modeling

Communication protocols:

- Dynamic environment
- Heterogeneous system
- Large node count
- Distributed algorithms
- Assignment to node addresses
- Exchange of messages
- Time modeling
Specifics of distributed systems modeling

Communication protocols:

- Dynamic environment
- Heterogeneous system
- Large node count
- Distributed algorithms
- Assignment to node addresses
- Exchange of messages
- Time modeling

- CPN unsuitable
- Pr/T-net extension required

- **HCPN-ST** extension based on:
  - Coloured Petri nets [Jensen92]
  - Predicate/Transition nets [GenLau81]
High Coloured Petri Nets with Structured Tokens (HCPN-ST)

Tuple \( HPCN-ST = (\Sigma, P, T, F, C, V, K, m_0, dt) \)

- \( \Sigma \): finite set of types & colour sets
- \( P, T \): sets of places, transitions \( P \cap T = \emptyset \)
- \( F_v \subseteq (P \times T) \), prearcs
- \( F_n \subseteq (T \times P) \), postarcs
- \( C \): colour function
- \( V \): arc expression
  \( \forall a \in F : type(V(a)) = boolean \) type
  and \( type( var(V(a))) \subseteq colour \) sets
- \( K \): capacity function
- \( m_0 \): initialization function
- \( dt \): time function
  \( \forall t \in T : dt(t) = n; n \in N \)
HCPN-ST: Example colour set

Tuple HCPN-ST:  \[ HPCN-ST = (\Sigma, P, T, F, C, V, K, m_0, dt) \]

**Type & colour set:**  \[ \Sigma = (I, Z, A, Plan, Wanderer) \]

1. **Elementary colour set**
   - \( I \): set of indices
   - \( Z \): set of pointers
   - \( Nodeaddress \): set of addresses
   - \( \{e\} \): empty position
   - \( \{r\} \): random position

   • Definition of message elements
   \[ A := \{a \mid a \in (Nodeaddress \cup e \cup r)\} \]

2. **Structured colour set**
   (protocol specific)
   - \( Plan := A \times A \times \cdots \times A \)
     \( i \) times
   - \( Wanderer := Z \times X \times Plan \)
   \[ X := \{x \mid x \in N^{>2}\} \]
   length

   • Used for construction of tokens representing different types of messages
Firing rule is expressed by pre- & postarc expressions:

- Prearc expressions (pre conditions)
- Postarc expressions (post conditions, modifications)

• Prearc & postarc expressions built upon:
  - Elementary colour
  - Structured colour
  - Logic expression

Colour transformations (token modifications) defined by postarc expressions.
Firing rule is expressed by pre- & postarc expressions:
- Prearc expressions (pre conditions)
- Postarc expressions (post conditions, modifications)

- Prearc & postarc expressions built upon:
  - Elementary colour
  - Structured colour
  - Logic expression

\[
\begin{align*}
\text{wm}_1, \text{wm}_2, \text{wm}_3: \text{Wanderer} \\
\text{Colour transformations (token modifications) defined by postarc expressions.}
\end{align*}
\]
Firing rule is expressed by pre- & postarc expressions:
- Prearc expressions (pre conditions)
- Postarc expressions (post conditions, modifications)

- Prearc & postarc expressions built upon:
  - Elementary colour
  - Structured colour
  - Logic expression

\[
\text{wm}_1, \text{wm}_2, \text{wm}_3: \text{Wanderer}
\]

Colour transformations (token modifications) defined by postarc expressions.
Firing rule is expressed by pre- & postarc expressions:

- Prearc expressions (pre conditions)
- Postarc expressions (post conditions, modifications)

• Prearc & postarc expressions built upon:
  - Elementary colour
  - Structured colour
  - Logic expression

\[ \text{wm}_1, \text{wm}_2, \text{wm}_3: \text{Wanderer} \]

\[ \text{wm}(\text{pos}_{z+1} = e) \rightarrow \text{wm}(z:=z+1 \land \text{pos}_{z} := n4) \]

Colour transformations (token modifications) defined by postarc expressions.
Firing rule is expressed by pre- & postarc expressions:

- Prearc expressions (pre conditions)
- Postarc expressions (post conditions, modifications)

- Prearc & postarc expressions built upon:
  - Elementary colour
  - Structured colour
  - Logic expression

\[\text{wm}_1, \text{wm}_2, \text{wm}_3: \text{Wanderer}\]

\[\text{wm}(\text{pos}_{z+1} = e) \rightarrow \text{wm}(z := z + 1 \land \text{pos}_z := n4)\]

\[\text{wm}_1, \text{wm}_2, \text{wm}_3: \text{Wanderer}\]

- Colour transformations (token modifications) defined by postarc expressions.
Application Example: Cluster generation by overlapping cycles

HCPN-ST example: Cluster generation by overlapping cycles
HCPN-ST example: Cluster generation by overlapping cycles

- emptynode
- wanderer
- net
- killedwr
- nodetoremove
- wrstart
- nodetoadd
- treatmentwr
- fullwr
- forward
- wrtoadd

n1
n2
nn

n1
n2
nn
**HCPN-ST example:** *Cluster generation by overlapping cycles*

```
HCPN-ST example: Cluster generation by overlapping cycles
```

- `emptynode`
- `wanderer`
- `net`
- `nettime`
- `nodetoremotetimer`
- `wr`
- `nodetoadddtimer`
- `kill`
- `nodetoremove`
- `Wrstart`
- `nodetoadd`
- `normforward`
- `random`
- `noremove`
- `addtoplan`
- `noadd`
- `addnno`
- `removefromplan`
- `treatmentwr`
- `next`
- `wrtoadd`
- `fullwr`
- `forward`
HCPN-ST example: *Cluster generation by overlapping cycles*
HCPN-ST example: Cluster generation by overlapping cycles

Cluster generation by overlapping cycles.
HCPN-ST example: Cluster generation by overlapping cycles

1. **Cluster Generation**
   - **Node Removal**: \( n1 \to \text{nodetoremoventimer} \)
   - **Node Addition**: \( n2 \to \text{nodetoaddtimer} \)
   - **Forwarding**: \( fm \to \text{treatmentwr} \)
   - **Next State**: \( n1 \to \text{next} \)

2. **Timed Transition**
   - \( td < t_{avg} - \Delta t \) \( \rightarrow \) kill
   - \( td \geq t_{avg} + \Delta t \) \( \rightarrow \) nodetoremove
   - \( \Delta t = 2\Delta t \) \( \rightarrow \) noadd

3. **Kill State**
   - \( wm(z = 1 \land \text{pos}_z = fm) \) \( \rightarrow \) kill

4. **Addition State**
   - \( \text{addtoplan} \) \( \rightarrow \) \( \text{addnoo} \)

5. **Forwarding**
   - \( \text{nettime} \to \text{net} \)
   - \( \text{wrtoadd} \)

6. **Example Model**
   - \( \text{HCPN-ST example: Cluster generation by overlapping cycles} \)
   - \( \text{HCPN-ST example model} \)

7. **Formulas**
   - \( (z > 1) \) \( \rightarrow \) \( \text{wm} \)
   - \( (z = x) \) \( \rightarrow \) \( \text{wm} \)
   - \( (z \neq x) \) \( \rightarrow \) \( \text{wm} \)
   - \( (\text{pos}_z = e \land \text{pos}_z \neq 0) \) \( \rightarrow \) \( \text{wm} \)
   - \( (\text{pos}_z = e \land \text{pos}_z \neq 1) \) \( \rightarrow \) \( \text{wm} \)
   - \( (z = 0) \) \( \rightarrow \) \( \text{wm} \)
   - \( (z = 1) \) \( \rightarrow \) \( \text{wm} \)
   - \( (z = z + 1) \) \( \rightarrow \) \( \text{wm} \)
   - \( t_d = \text{ttry} \)
   - \( \text{wr}(1, x, fm, (\text{pos2..posx}) = \text{random}(e, r)) \)

8. **Final States**
   - \( \text{fullwr} \)
   - \( \text{killedwr} \)
   - \( \text{wrgenerated} \)
Transformation: HCPN-ST → CPN

- Transformation of firing rules
- Transformation of token colours

HCPN-ST Model

CPN Model

Peneca Chromos

CPN Simulation

CPN Analysis

INA Tool

Petri net class (HCPN-ST) verifiable (analyzable + simulatable)

• Validation
• Statistics
• Parameters

• Verification of Petri net properties:
  - Liveliness
  - Boundedness
  - Reachability

# First Results

**CPN-Model:**
- up to 48 firing modes per transition
- up to 192 colours per place

**Simulation (Peneca Chromos):**
1. Conflict free
   (statically and dynamically)
2. Sound simulation
   (no deadlock during exhaustive test)
3. Behaviour as expected

**Analysis (INA):**
1. Liveliness:
   weakly live
2. Boundedness:
   net is bounded and safe (1-bounded)
3. Reachability:
   limited state space, no undesired states
Conclusions & Future work

Summary
✓ Problem oriented Petri net class developed
✓ General approach demonstrated, using a complex example
✓ Analysis by INA after transformation to CPN

Outlook
❖ More & detailed examples & results
❖ Tool support (Pr/T-nets + custom types + time & probabilities)
❖ Extended analysis & verification support
❖ Quantitative statistic results
❖ Comparison against other simulation approaches (e.g. MLDesigner)
Conclusions & Future work

Summary
✓ Problem oriented Petri net class developed
✓ General approach demonstrated, using a complex example
✓ Analysis by INA after transformation to CPN

Outlook
❖ More & detailed examples & results
❖ Tool support (Pr/T-nets + custom types + time & probabilities)
❖ Extended analysis & verification support
❖ Quantitative statistic results
❖ Comparison against other simulation approaches (e.g. MLDesigner)
Questions?