Structural Translation From Time Petri Nets to Timed Automata

Franck Cassez and Olivier H. Roux

IRCCyN/CNRS
BP 92101
1 rue de la Noë
44321 Nantes cedex 3
France

Automated Verification of Critical Systems (AVoCS’04)
4 September 2004, London (UK)

Contents

1. **Context & Related Work**
2. **Time Petri Nets & Timed Automata**
3. **Translation: TPN to TA**
4. **Conclusion**
Context

Petri Nets with time

- **Timed** Petri Nets ([Ramchandani, 1974]) – sharp timing constraints
  
P-Timed PN = T-Timed PN

- **Time** Petri Nets (TPN) ([Merlin, 1974]) – interval timing constraints
  
T-TPN ≠ P-TPN
Timed PN ⊆ T-TPN and in P-TPN
TPN ⊆ Time Stream Petri Nets ([Diaz & Senac, 1994])
Context

Petri Nets with time

- **Timed Petri Nets** ([Ramchandani, 1974]) – sharp timing constraints
  
  \[ \text{P-Timed PN} = \text{T-Timed PN} \]

- **T-Time Petri Nets (TPN)** ([Merlin, 1974]) – interval timing constraints
  
  \[ \text{T-TPN} \neq \text{P-TPN} \]

  Timed PN \(\subseteq\) T-TPN and in P-TPN

  TPN \(\subseteq\) Time Stream Petri Nets ([Diaz & Senac, 1994])

---

(c) IRCCyN/CNRS

From Time Petri Nets to Timed Automata
Context

Main Results & Tools for T-TPNs [Berthomieu & Diaz, 1991]

- Boundedness for TPNs undecidable
- Reachability for bounded TPNs decidable
- Tools: computation of the state class graph (SCG)
  - Tina [Berthomieu, 2003]
    Computes the SCG, untimed CTL* model-checking
  - Roméo [Gardey et al., 2003]
    Computes the SCG, Region Graph, Reachability
Context

Main Results & Tools for T-TPNs [Berthomieu & Diaz, 1991]

- Boundedness for TPNs undecidable
- Reachability for bounded TPNs decidable
- Tools: computation of the state class graph (SCG)
  - Tina [Berthomieu, 2003]
    Computes the SCG, untimed CTL* model-checking
  - Roméo [Gardey et al., 2003]
    Computes the SCG, Region Graph, Reachability

Timed Automata [Alur & Dill, 1994]
Finite Automata extended with real-valued clocks
Context

■ Main Results & Tools for T-TPNs [Berthomieu & Diaz, 1991]
  • Boundedness for TPNs undecidable
  • Reachability for bounded TPNs decidable
  • Tools: computation of the state class graph (SCG)
    - Tina [Berthomieu, 2003]
      Computes the SCG, untimed CTL* model-checking
    - Roméo [Gardey et al., 2003]
      Computes the SCG, Region Graph, Reachability

■ Main Results & Tools for Timed Automata ([Alur & Dill, 1994]):
  • Reachability + Timed CTL model-checking decidable
  • Tools:
    - Uppaal [Pettersson & Larsen, 2000]
    - Kronos [Yovine, 1997]
    - Cmc [Laroussinie et al, 1998]
Related Work

- From 1-safe TPN to TA [Sifakis & Yovine, 1996]
- From bounded TPN to TA [Sava, 2001]
  No correctness proof (equivalence of the semantics ?)
- From TPN to TA [Lime & Roux, 2003]
  correctness proof (timed bisimilarity)
  Enriched SCG = TA \implies \text{heavy computation}
  Needs a dedicated tool ([Gardey et al., 2003])
Related Work

Previous approaches:

- Either restricted to 1-safe TPN
- No formal correctness proof of the translation
- Or need to compute the state space of the TPN
Related Work

Previous approaches:
- Either restricted to 1-safe TPN
- No formal correctness proof of the translation
- Or need to compute the state space of the TPN

Our aim:
- **Structural** translation (no heavy computation)
- **Correctness proof** of the translation (behavioural equivalence)
Related Work

Previous approaches:
- Either restricted to 1-safe TPN
- No formal correctness proof of the translation
- Or need to compute the state space of the TPN

Our aim:
- Structural translation (no heavy computation)
- Correctness proof of the translation (behavioural equivalence)

Results:
- Structural translation
- Applies to non safe TPNs
- Correctness proof of the translation (behavioural equivalence)
- Model-checking of TCTL for bounded T-TPN
- Allows to use efficient tools for analysis of TA
1. Context & Related Work
2. Time Petri Nets & Timed Automata
3. Translation: TPN to TA
4. Conclusion
Initially: $P_0 = P_2 = 1$ at $\delta = 0$

$(P_0 P_2, 0)$
Initially: \( P_0 = P_2 = 1 \) at \( \delta = 0 \)

\( \delta \in [1, 4] \): \( T_0 \) enabled; fire \( T_0 \) at \( \delta = 3.7 \)

\[
(P_0 P_2, 0) \xrightarrow{3.7} (P_0 P_2, 3.7) \xrightarrow{T_0} (P_1 P_2, 3.7)
\]
Initially: $P_0 = P_2 = 1$ at $\delta = 0$

$\delta \in [1, 4]$: $T_0$ enabled; fire $T_0$ at $\delta = 3.7$

“untimed” $T_1$ is enabled $\implies$ clock for $T_1$ starts

$$(P_0 P_2, 0) \xrightarrow{3.7} (P_0 P_2, 3.7) \xrightarrow{T_0} (P_1 P_2, 3.7)$$
Initially: \( P_0 = P_2 = 1 \) at \( \delta = 0 \)

\( \delta \in [1, 4] \): \( T_0 \) enabled; fire \( T_0 \) at \( \delta = 3.7 \)

“untimed” \( T_1 \) is enabled \( \implies \) clock for \( T_1 \) starts

after 3 t.u. “timed” \( T_1 \) enabled and must fire before 5 t.u.

\[
(P_0P_2, 0) \xrightarrow{3.7} (P_0P_2, 3.7) \xrightarrow{T_0} (P_1P_2, 3.7) \xrightarrow{3 \leq t \leq 5} (P_1P_2, 3.7 + t)
\]
Initially: $P_0 = P_2 = 1$ at $\delta = 0$

$\delta \in [1, 4]$: $T_0$ enabled; fire $T_0$ at $\delta = 3.7$

“untimed” $T_1$ is enabled $\implies$ clock for $T_1$ starts

after 3 t.u. “timed” $T_1$ enabled and must fire before 5 t.u.

fire $T_1$ and time-elapsing

$$(P_0P_2, 0) \xrightarrow{3.7} (P_0P_2, 3.7) \xrightarrow{T_0} (P_1P_2, 3.7) \xrightarrow{3\leq t\leq 5} (P_1P_2, 3.7 + t)$$

$$\xrightarrow{T_1} (\emptyset, 3.7 + t) \xrightarrow{t'\geq 0} (\emptyset, 3.7 + t + t')$$
Time Petri Nets – Semantics

$\mathcal{T}$ a TPN

Semantics of $\mathcal{T} = [\mathcal{T}] = \text{sequence of alternating}$
- Discrete step
- Time step

$[\mathcal{T}] = \text{Timed Transition System (TTS)}$
Finite structure + real-valued clocks
Timed Automata

Alur & Dill, 1994

Finite structure + real-valued clocks

Invariant - Label - Guard - Reset
Finite structure + real-valued clocks

Invariant - Label - Guard - Reset

\((0, x = 0)\)
Timed Automata \[\text{Alur & Dill, 1994}\]

Finite structure + real-valued clocks

Invariant - Label - Guard - Reset

\[(0, x = 0) \xrightarrow{1.65} (0, x = 1.65)\]
Finite structure + real-valued clocks

Invariant - Label - Guard - Reset

\[(0, x = 0) \xrightarrow{1.65} (0, x = 1.65) \xrightarrow{a} (1, x = 0) \xrightarrow{t \geq 0} (1, x = t)\]
Timed Automata \cite{Alur&Dill:94}

Finite structure + real-valued clocks

Invariant - Label - Guard - Reset

\[
(0, x = 0) \xrightarrow{1.65} (0, x = 1.65) \xrightarrow{a} (1, x = 0) \xrightarrow{t \geq 0} (1, x = t)
\]

+ (arrays of) integer variables
Timed Automata (TA) + bounded integer variables

Semantics of a TA = \([\mathcal{A}]\) = sequence of alternating
  - Discrete step
  - Time step

Semantics: \([\mathcal{A}]\) = Timed Transition System (TTS)
From TPNs to TA

- Given: $\mathcal{T}$ a TPN with $n$ transitions and $m$ places
- $p$: array of integers; $p[i] = \text{tokens in place } i, i \in [1..m]$
- $A_i$ a TA with one clock associated to transition $T_i$
- $SU$ a cyclic supervisor (4 states) – computes new marking

- Synchronization: $\mathcal{A} = (SU \mid A_1 \mid \cdots \mid A_n)$
  $\mathcal{A}$ has $n$ clocks
- Discrete step in $[\mathcal{T}] = 4$ discrete steps in $[\mathcal{A}]$
From TPNs to TA

- **Given:** $\mathcal{T}$ a TPN with $n$ transitions and $m$ places
- **$p$:** array of integers; $p[i] = \text{tokens in place } i$, $i \in [1..m]$
- **Synchronization:** $\mathcal{A} = (SU | A_1 | \cdots | A_n)$
  - $\mathcal{A}$ has $n$ clocks
- **Discrete step in** $[\mathcal{T}] = 4$ discrete steps in $[\mathcal{A}]$

**Results**

1. **Theorem 3.2:** $[\mathcal{A}]$ and $[\mathcal{T}]$ are timed bisimilar
2. $\mathcal{T}$ is bounded iff $p$ is bounded
3. $\mathcal{T}$ has $k$ reachable markings $\implies \mathcal{A}$ has $\leq 4 \cdot k \cdot n$ discrete states
From TPNs to TA

■ Results

1. Theorem 3.2: $[\mathcal{A}]$ and $[\mathcal{T}]$ are timed bisimilar
2. $\mathcal{T}$ is bounded iff $p$ is bounded
3. $\mathcal{T}$ has $k$ reachable markings $\implies \mathcal{A}$ has $\leq 4 \cdot k \cdot n$ discrete states

■ Consequences

- Quantitative logic TCTL decidable on bounded TPNs
- Efficient translation to Uppaal: Roméo [Gardey et al., 2003]
From TPNs to TA

Results
1. Theorem 3.2: \([A]\) and \([T]\) are timed bisimilar
2. \(T\) is bounded iff \(p\) is bounded
3. \(T\) has \(k\) reachable markings \(\implies A\) has \(\leq 4 \cdot k \cdot n\) discrete states

Consequences
- Quantitative logic TCTL decidable on bounded TPNs
- Efficient translation to Uppaal: Roméo [Gardey et al., 2003]

Number of clocks?
- Useful clocks: only for enabled transitions
- Active clock reduction
- Use of active clocks feature in Uppaal
Contents

1. Context & Related Work
2. Time Petri Nets & Timed Automata
3. Translation: TPN to TA
4. Conclusion
Conclusion & Future Work

**Summary**

- **Formal semantics** for TPNs
- **Structural** translation to TA
  - Correctness proof
  - Model-checking TCTL

**Future Work**

- **Use on real case studies** (with Uppaal)
- **Expressiveness**: from TA to TPNs?
References (1)


Automaton for one Transition

(a) The automaton $A_i$ for transition $t_i$

$$\begin{align*}
p &\geq q_{t_i} \\
\text{?update} &
\end{align*}$$

$$\begin{align*}
\alpha(t_i) &\leq x_i \leq \beta(t_i) \\
\text{?pre(i)} &
\end{align*}$$

$$\begin{align*}
p &:= p - q_{t_i} \\
\text{Firing} &
\end{align*}$$

$$\begin{align*}
p &< q_{t_i} \\
\text{?update} &
\end{align*}$$

$$\begin{align*}
 x_i &:= 0 \\
\text{?update} &
\end{align*}$$

$$\begin{align*}
p &\geq q_{t_i} \\
\text{?update} &
\end{align*}$$

$$\begin{align*}
p &:= p + t_i q_{t_i} \\
\text{p} &< q_{t_i} \\
\text{?update} &
\end{align*}$$
Automaton for the Supervisor

(b) Supervisor $SU$

From Time Petri Nets to Timed Automata
About Active Clocks

\[
\begin{align*}
p & \geq t_i \\
\text{?update} & \\
\alpha(t_i) & \leq x_i \leq \beta(t_i) \\
\text{?pre} & \\
p & := p - t_i \\
\text{Firing} & \\
p & < t_i \\
\text{?update} & \\
x_i & := 0 \\
p & \geq t_i \\
\text{?update} & \\
\bar{t} & \\
p & < t_i \\
\text{?update} & \\
p & := p + t_i \\
\end{align*}
\]