On the decidability of model checking LTL fragments in monotonic extensions of Petri nets

María Martos-Salgado and Fernando Rosa-Velardo

Universidad Complutense de Madrid

PROLE 2013
Madrid, 19th September 2013
Feature models

A feature model is a representation of the products of some line of products in terms of features. Feature models are represented by feature trees. The relationship between a parent feature and its child features may be:
Our running example... Feature models

Feature models

A feature model is a representation of the products of some line of products in terms of features. Feature models are represented by feature trees. The relationship between a parent feature and its child features may be:

Mandatory

![Mandatory feature](image)
Feature models

A feature model is a representation of the products of some line of products in terms of features. Feature models are represented by feature trees. The relationship between a parent feature and its child features may be:

Mandatory

![Mandatory feature](image)

Optional

![Optional feature](image)
Our running example... Feature models

Feature models

A feature model is a representation of the products of some line of products in terms of features. Feature models are represented by feature trees. The relationship between a parent feature and its child features may be:

Mandatory

Optional

Or
Our running example... Feature models

Feature models

A feature model is a representation of the products of some line of products in terms of features. Feature models are represented by feature trees. The relationship between a parent feature and its child features may be:

- Mandatory
- Optional
- Alternative
Place/Transition nets. Building our car

![Diagram of Place/Transition nets for building a car](image)

- Create
- Mand.
- Opt.
- Or.
- Alt.
Place/Transition nets. Building our car
Place/Transition nets. Building our car

M. Martos, F. Rosa (UCM)
Place/Transition nets. Building our car

![Diagram of a Petri net with transitions and places](image)
Place/Transition nets. Building our car

![Diagram of Place/Transition nets]

M. Martos, F. Rosa (UCM)

Model checking extensions of Petri nets

PROLE 2013, 19/09/13
Model Checking

Model checking is a technique for automatic formal verification of systems. Given:

- A formal description of the system we want to check. Ex: A system modelled by a Petri net.
- A property \( P \) to check. Ex: A property expressed in some temporal logic.

A model checking algorithm tells us if \( P \) holds at the system specification or not.

Ex. We want to check if our model satisfies the formula:

\[
F(cov(\{p_1\}) \land (cov(\{p_3\}) \lor cov(\{p_4\})) \land ((cov(\{p_5\}) \land \neg cov(\{p_6\})) \lor (cov(\{p_6\}) \land \neg cov(\{p_5\})))) \land \bigwedge_{t \in T} \neg en(t)
\]
Reset nets

![Diagram of reset nets]

- Mand.
- Opt.
- Or.
- Alt.

Node labeled 'Create' connects to:
- 3
- 5
Reset nets
Reset nets
Reset nets

Create


PROLE 2013, 19/09/13
Reset nets

[Diagram of reset nets with nodes labeled Mand., Opt., Or., and Alt. connected to a main node labeled Create, with transitions numbered 3 and 5.]
\(\nu\)-Petri nets
ν-Petri nets
\( \nu \)-Petri nets

\[
\text{New} \xrightarrow{\nu} a_b \\
\text{Start}
\]

\( \nu \)-Petri nets

![Diagram of \( \nu \)-Petri nets]

- **New** \( \nu \) to \( b \)
- \( x \) from \( b \) to **Start**
- **Start** to \( a \), \( a \), \( a \), and \( a \) boxes
  - **Mand.**
  - **Opt.**
  - **Or.**
  - **Alt.**

- Connections and \( x \) labels within the boxes.
\( \nu \)-Petri nets

```
| New | \( \nu \) |
```

```
\[ \text{Start} \]
```

```
| \( x \) | \( x \) | \( x \) | \( x \) |
```

```
| a | a | a |
```

```
| \( x \) | \( x \) | \( x \) | \( x \) |
```

```
```

M. Martos, F. Rosa (UCM)
\( \nu \)-Petri nets
ν-Petri nets
(Linear) Temporal logics are the formalisms we use to express the properties we want to check. Their basic components are:

- Atomic formulae:

- Temporal operators:
Temporal logics

(Linear) Temporal logics are the formalisms we use to express the properties we want to check. Their basic components are:

- **Atomic formulae:**
  - $cov(m)$, where $m$ is a marking: $cov(m)$ holds in $\pi$ if the first marking in $\pi$ covers $m$.

- **Temporal operators:**
Temporal logics

(Linear) Temporal logics are the formalisms we use to express the properties we want to check. Their basic components are:

- **Atomic formulae:**
  - $cov(m)$, where $m$ is a marking: $cov(m)$ holds in $\pi$ if the first marking in $\pi$ covers $m$.
  - $first(t)$, where $t$ is a transition: $first(t)$ holds in $\pi$ if the first transition fired in $\pi$ is $t$.

- **Temporal operators:**
(Linear) Temporal logics are the formalisms we use to express the properties we want to check. Their basic components are:

- **Atomic formulae:**
  - $\textit{cov}(m)$, where $m$ is a marking: $\textit{cov}(m)$ holds in $\pi$ if the first marking in $\pi$ covers $m$.
  - $\textit{first}(t)$, where $t$ is a transition: $\textit{first}(t)$ holds in $\pi$ if the first transition fired in $\pi$ is $t$.

- **Temporal operators:**
  - $S, \pi \models \textbf{X} \varphi$ (next) holds if the property $\varphi$ holds in the state that follows $s$ in $\pi$. 
(Linear) Temporal logics are the formalisms we use to express the properties we want to check. Their basic components are:

- **Atomic formulae:**
  - $\text{cov}(m)$, where $m$ is a marking: \( \text{cov}(m) \) holds in \( \pi \) if the first marking in \( \pi \) covers \( m \).
  - $\text{first}(t)$, where \( t \) is a transition: \( \text{first}(t) \) holds in \( \pi \) if the first transition fired in \( \pi \) is \( t \).

- **Temporal operators:**
  - $\mathcal{S}, \pi \models \textbf{X}\varphi$ (next) holds if the property \( \varphi \) holds in the state that follows \( s \) in \( \pi \).
  - $\mathcal{S}, \pi \models \textbf{F}\varphi$ (eventually) holds if the property \( \varphi \) holds in some state of \( \pi \).
Temporal logics

(Linear) Temporal logics are the formalisms we use to express the properties we want to check. Their basic components are:

- **Atomic formulae:**
  - $\text{cov}(m)$, where $m$ is a marking: $\text{cov}(m)$ holds in $\pi$ if the first marking in $\pi$ covers $m$.
  - $\text{first}(t)$, where $t$ is a transition: $\text{first}(t)$ holds in $\pi$ if the first transition fired in $\pi$ is $t$.

- **Temporal operators:**
  - $S, \pi \models X \varphi$ (next) holds if the property $\varphi$ holds in the state that follows $s$ in $\pi$.
  - $S, \pi \models F \varphi$ (eventually) holds if the property $\varphi$ holds in some state of $\pi$.
  - $S, \pi \models \varphi U \psi$ (until) holds if there is a state of the path $\pi$ such that $\psi$ holds in that state, and $\varphi$ holds at every preceding state on the path.
(Linear) Temporal logics are the formalisms we use to express the properties we want to check. Their basic components are:

- **Atomic formulae:**
  - $\text{cov}(m)$, where $m$ is a marking: $\text{cov}(m)$ holds in $\pi$ if the first marking in $\pi$ covers $m$.
  - $\text{first}(t)$, where $t$ is a transition: $\text{first}(t)$ holds in $\pi$ if the first transition fired in $\pi$ is $t$.

- **Temporal operators:**
  - $\mathcal{S}, \pi \models \mathbf{X}\varphi$ (next) holds if the property $\varphi$ holds in the state that follows $s$ in $\pi$.
  - $\mathcal{S}, \pi \models \mathbf{F}\varphi$ (eventually) holds if the property $\varphi$ holds in some state of $\pi$.
  - $\mathcal{S}, \pi \models \varphi \mathbf{U}\psi$ (until) holds if there is a state of the path $\pi$ such that $\psi$ holds in that state, and $\varphi$ holds at every preceding state on the path.
  - $\mathcal{S}, \pi \models \mathbf{G}\varphi$ (globally) holds if $\varphi$ holds in every state of $\pi$. 

M. Martos, F. Rosa (UCM)
The fragments we consider

**LTL**

An LTL formula is either an atomic formula or a formula of the form $\neg \varphi$, $\varphi \land \psi$, $\varphi \lor \psi$, $X\varphi$, $F\varphi$, or $\varphi U\psi$, where $\varphi$ and $\psi$ are LTL formulae.
The fragments we consider

**LTL**

An LTL formula is either an atomic formula or a formula of the form $\neg \varphi$, $\varphi \land \psi$, $\varphi \lor \psi$, $X\varphi$, $F\varphi$, or $\varphi U \psi$, where $\varphi$ and $\psi$ are LTL formulae.

The fragments of LTL we consider are:

- $LTL_f$, the fragment of LTL that uses only *first* as basic predicate,
- $L(F)$, the fragment of LTL in which negation is only applied to basic predicates (not to operators), and the operators are $X$, $F$, $\land$ and $\lor$,
- $L(GF)$, the fragment of LTL in which the only allowed composed operator is $GF$, the operators are $F$, $\lor$ and $\land$ and negation is only applied to basic predicates.
Model checking Place/Transition nets

- $LTL$ (with both $first$ and $cov$). $\times$

- $LTL_f$ (with $cov$ only). $\checkmark$

- $\mathcal{L}(F)$ (with existential interpretation). $\checkmark$

- $\mathcal{L}(GF)$ (with existential interpretation). $\checkmark$
Outline

1. Introduction
2. Model Checking Reset nets
3. Model Checking $\nu$-Petri nets
4. A decidable fragment
5. Conclusions and Future Work
Model Checking $LTL_f$ is undecidable for reset nets

Undecidability of $LTL_f$

- We reduce model checking $LTL_f$ for lossy inhibitor nets, which is undecidable \(^a\), to the same problem for reset nets.
- Given a inhibitor net, we define a reset net by replacing its zero tests by reset arcs.
- We prove that there is a surjective homomorphism between the runs of both nets that preserves the sequence of labels of runs.
- The only atomic predicate in $LTL_f$ is $first$ so $N \models \varphi$ iff $N' \models \varphi$.

Model Checking $LTL_f$ is undecidable for reset nets

Undecidability of $LTL_f$

- We reduce model checking $LTL_f$ for lossy inhibitor nets, which is undecidable, to the same problem for reset nets.
- Given a inhibitor net, we define a reset net by replacing its zero tests by reset arcs.
- We prove that there is a surjective homomorphism between the runs of both nets that preserves the sequence of labels of runs.
- The only atomic predicate in $LTL_f$ is $\textit{first}$ so $N \models \varphi$ iff $N' \models \varphi$.

---

Model Checking $\mathcal{L}(\text{GF})$ is undecidable for reset nets

### Undecidability of $\mathcal{L}(\text{GF})$

- **GFcov($M$)** is a formula in $\mathcal{L}(\text{GF})$.
- **GFcov($M$)** expresses the repeated coverability problem.
- Repeated coverability problem is undecidable for reset nets

---

Model Checking $\mathcal{L}(\text{GF})$ is undecidable for reset nets

Undecidability of $\mathcal{L}(\text{GF})$

- $\text{GF} \text{cov}(M)$ is a formula in $\mathcal{L}(\text{GF})$.
- $\text{GF} \text{cov}(M)$ expresses the repeated coverability problem.
- Repeated coverability problem is undecidable for reset nets \(^a\)

Model Checking $\mathcal{L}(\mathbf{F})$ is undecidable for reset nets

**Undecidability of $\mathcal{L}(\mathbf{F})$**

- We reduce reachability, which is undecidable for reset nets\(^a\), to model checking some formula in $\mathcal{L}(\mathbf{F})$.
- Given a reset net, and a marking $m$, we can compute the set $M$ of the least markings greater than $M$.
- $m$ is reachable iff there is a marking $m'$ which covers $m$, iff the formula $\mathbf{F}(\text{cov}(m) \land \bigwedge_{\tilde{m} \in M} \neg \text{cov}(\tilde{m}))$ is satisfied.

Model Checking $L(F)$ is undecidable for reset nets

Undecidability of $L(F)$

- We reduce reachability, which is undecidable for reset nets\(^a\), to model checking some formula in $L(F)$.
- Given a reset net, and a marking $m$, we can compute the set $M$ of the least markings greater than $M$.
- $m$ is reachable iff there is a marking $m'$ which covers $m$, iff the formula $F(cov(m) \land \bigwedge_{\tilde{m} \in M} \neg cov(\tilde{m}))$ is satisfied.

---

Outline

1. Introduction
2. Model Checking Reset nets
3. Model Checking $\nu$-Petri nets
4. A decidable fragment
5. Conclusions and Future Work
For each place $p$ in the reset net, we add a place $p'$ which contains a token of a name which represent the name of the “real” tokens in $p$. 

M. Martos, F. Rosa (UCM)  Model checking extensions of Petri nets  PROLE 2013, 19/09/13 16 / 24
Representative tokens...

For each place $p$ in the reset net, we add a place $p'$ which contains a token of a name which represent the name of the “real” tokens in $p$. 

\[ \begin{align*}
  & p \rightarrow t \rightarrow q \\
  & r \rightarrow a \\
  \end{align*} \]

\[ \begin{align*}
  & p \rightarrow t \rightarrow q \\
  & x_p \rightarrow a \rightarrow x_r \\
  & b \rightarrow c \rightarrow x_q \\
  \end{align*} \]
Some corollaries

Model checking $LTL_f$ is undecidable for $\nu$-PN.
Because the previous simulation preserves all the behavioral properties.

Repeated coverability is undecidable for $\nu$-PN.
Because repeated coverability is undecidable for reset nets and the simulation preserves it.

Model checking $\mathcal{L}(\text{GF})$ is undecidable for $\nu$-PN.
Because repeated coverability is undecidable for reset nets.
Some corollaries

Model checking $LTL_f$ is undecidable for $\nu$-PN. 
Because the previous simulation preserves all the behavioral properties.

Repeated coverability is undecidable for $\nu$-PN. 
Because repeated coverability is undecidable for $\nu$-PN.

Model checking $\mathcal{L}(GF)$ is undecidable for $\nu$-PN.
Because repeated coverability is undecidable for $\nu$-PN.
Outline

1. Introduction
2. Model Checking Reset nets
3. Model Checking $\nu$-Petri nets
4. A decidable fragment
5. Conclusions and Future Work
Definition of $F_{cov}$

We call $F_{cov}$ to the fragment of $L(F)$ in which negation is not allowed. 

- In this logic we can express bounded repeated coverability.
- $F_{cov}$ cannot express properties like $\neg cov(M)$. In particular, the formula $F(cov(M) \land \bigwedge_{\tilde{m} \in M} \neg cov(\tilde{m}))$ (reachability).
- We consider existential interpretation, so that a formula is satisfied if some maximal run satisfies it.
Model checking $F_{cov}$ is decidable for reset nets

Decidability of $F_{cov}$

- If $\phi$ is a boolean combination of formulae of the form $cov(m)$, it is trivial to decide whether $\phi$ is satisfied because multiset inclusion is decidable.

- We proceed by induction on the nesting of operators $F$ in the formula of $F_{cov}$ we want to verify $\phi$. 
Model checking $F_{cov}$ is decidable for reset nets

Decidability of $F_{cov}$

- If $\phi$ is a boolean combination of formulae of the form $cov(m)$, it is trivial to decide whether $\phi$ is satisfied because multiset inclusion is decidable.

- We proceed by induction on the nesting of operators $F$ in the formula of $F_{cov}$ we want to verify $\phi$. 
Outline

1. Introduction
2. Model Checking Reset nets
3. Model Checking $\nu$-Petri nets
4. A decidable fragment
5. Conclusions and Future Work
Conclusions

<table>
<thead>
<tr>
<th></th>
<th>P/T</th>
<th>Reset</th>
<th>$\nu$-PN</th>
</tr>
</thead>
<tbody>
<tr>
<td>$LTL$</td>
<td>$+$ 1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$LTL_f$</td>
<td>$+$ 1</td>
<td>- 2</td>
<td>-</td>
</tr>
<tr>
<td>$\mathcal{L}(GF)$</td>
<td>$+$ 3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\mathcal{L}(F)$</td>
<td>$+$ 4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$F_{cov}$</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>$\forall F_{cov}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Future Work

A lot of work to do...

- Define more expressive logics for which the model checking problem is decidable.
- Define logics with atomic predicates that are more specific for the particular model.
- Find a logic which distinguishes between reset nets and $\nu$-Petri nets.
- Perform a finer complexity analysis.
Thank You!!