Nested-Units Petri Nets

Hubert Garavel
Inria Grenoble – LIG
and Saarland University

http://convecs.inria.fr
Outline

- Introduction
- The NUPN model
- The unit-safeness property
- Some expressiveness results
- The place-fusion abstraction
- Optimized encoding of markings
- Software support for NUPNs
- Conclusion
Three controversial equations in concurrency theory
Controversial equation #1

(for all a, b, c : actions) \( a.(b + c) = a.b + a.c \)?

- If the answer is **yes**
  - linear-time semantics

- If the answer is **no**
  - branching-time semantics

from: **R. van Glabbeek** and **F. Vaandrager.**
*Petri Net Models for Algebraic Theories of Concurrency* (PARLE, 1987)
Controversial equation #2

(forall a, b : actions)  \(a || b = a.b + b.a\)

- If the answer is yes
  - interleaving semantics
- If the answer is no
  - true concurrency
  - Petri nets can distinguish
  - (Mazurkiewicz traces and Winskel event structures can too)

A 3\textsuperscript{rd} controversial equation...

\[(\forall a, b, c) \ (a \cdot b) ||_b (b \cdot c) = (a \cdot b \cdot c) ||_b b \ ?\]

see also: G. Boudol, I. Castellani, M. Hennessy, A. Kiehn

A theory of processes with localities (Form. Asp. Comp. 1994)

- Interleaving semantics:
  - they are the same (i.e., a \cdot b \cdot c)

- Petri nets:
  - they are also the same
  - no way to indicate that a and c are not on the same side
  - Petri nets preserve \textit{concurrency}, not \textit{locality}
How to model locality and hierarchy?

- Places that belong to the same sequential process are enclosed into "units."
- Units can be recursively nested at an arbitrary depth.

\[(a.b) \parallel_b (b.c) \neq (a.b.c) \parallel_b b\]
The NUPN model
\( (NUPN = \text{Nested-Unit Petri Nets}) \)
NUPN definition

- Extension of elementary nets (all arc weights = 1)
- NUPN = 8-tuple \((P, T, F, M_0, U, u_0, \subseteq, \text{unit})\)
  - Elements 1-4 of this tuple are standard

**Definition 1.** A (marked) Nested-Unit Petri Net (acronym: NUPN) is a 8-tuple \((P, T, F, M_0, U, u_0, \subseteq, \text{unit})\) where:

1. \(P\) is a finite, non-empty set; the elements of \(P\) are called places.
2. \(T\) is a finite set such that \(P \cap T = \emptyset\); the elements of \(T\) are called transitions.
3. \(F\) is a subset of \((P \times T) \cup (T \times P)\); the elements of \(F\) are called arcs.
4. \(M_0\) is a subset of \(P\); \(M_0\) is called the initial marking.
NUPN definition

- NUPN = 8-tuple \((P, T, F, M_0, U, u_0, \sqsubseteq, \text{unit})\)

  - Elements 5-8 of these tuples are novel:
    - (5,6,7): tree of units + (8): mapping: place → unit

5. \(U\) is a finite, non-empty set such that \(U \cap T = U \cap P = \emptyset\); the elements of \(U\) are called units.
6. \(u_0\) is an element of \(U\); \(u_0\) is called the root unit.
7. \(\sqsubseteq\) is a binary relation over \(U\) such that \((U, \sqsubseteq)\) is a tree with a single root \(u_0\), where \((\forall u_1, u_2 \in U)\) \(u_1 \sqsubseteq u_2 \overset{\text{def}}{=} u_2 \sqsubseteq u_1\); thus, \(\sqsubseteq\) is reflexive, antisymmetric, transitive, and \(u_0\) is the greatest element of \(U\) for this relation; intuitively, \(u_1 \sqsubseteq u_2\) expresses that unit \(u_1\) is transitively nested in or equal to unit \(u_2\).
8. \text{unit} is a function \(P \rightarrow U\) such that \((\forall u \in U \setminus \{u_0\})\) \((\exists p \in P)\) \(\text{unit}(p) = u\); intuitively, \(\text{unit}(p) = u\) expresses that unit \(u\) directly contains place \(p\).
Analogy with known data structures

- **File systems**
  - unit \(\rightarrow\) directory
  - place \(\rightarrow\) file
  Directories can be recursively nested at arbitrary depth
  Each directory may (or not) contain files

- **XML documents**
  - unit \(\rightarrow\) element
  - place \(\rightarrow\) attribute
  (contrary to XML, the order of elements is not significant)
Units are not boxes...

- A NUPN units encapsulates **places only**
  This is different from "boxes" (or "subnets") that encapsulate places, transitions, and arcs

- Another key difference is parallel composition:
  - 2 boxes in parallel → 1 box
  - 2 units in parallel → 3 units
Execution rules ("token game")

- The usual firing rules of Petri nets are unchanged.
- Units are totally orthogonal to transitions.
- Yet, units allow markings to be structured:

\[
\text{places}(u) \overset{\text{def}}{=} \{ p \in P \mid \text{unit}(p) = u \} \quad \tilde{U} \overset{\text{def}}{=} \{ u \in U \mid \text{places}(u) \neq \emptyset \}
\]

**Proposition 1.** Let \((P, T, F, M_0, U, u_0, \sqsubseteq, \text{unit})\) be a NUPN. The family of sets \(\text{places}(u)\), where \(u \in \tilde{U}\), is a partition of \(P\).

\[
M \triangleright u \overset{\text{def}}{=} M \cap \text{places}(u)
\]

**Proposition 2.** Let \((P, T, F, M_0, U, u_0, \sqsubseteq, \text{unit})\) be a NUPN. Any marking \(M\) can be expressed as \(M = (M \triangleright u_1) \uplus \ldots \uplus (M \triangleright u_n)\), where \(u_1, \ldots, u_n\) are the units of \(\tilde{U}\), and where \(\uplus\) denotes the disjoint set union.
The unit-safeness property
Unit-safeness property

Disjonction of two units

\[
\text{disjoint}(u_1, u_2) \overset{\text{def}}{=} (u_1 \not\subseteq u_2) \land (u_2 \not\subseteq u_1)
\]

characterizes pairs of units neither equal nor nested one in the other.

Unit safeness of a marking

**Definition 5.** Let \((P, T, F, M_0, U, u_0, \sqsubseteq, \text{unit})\) be a NUPN. A marking \(M \subseteq P\) is said to be unit safe iff it satisfies the predicate defined as follows: \(\text{unit-safe}(M) \overset{\text{def}}{=} (\forall p_1, p_2 \in M) \ (p_1 \neq p_2) \Rightarrow \text{disjoint}(\text{unit}(p_1), \text{unit}(p_2))\); that is, all places of a unit-safe marking are contained in disjoint units.

Unit safeness of a NUPN

**Definition 6.** Let \(N = (P, T, F, M_0, U, u_0, \sqsubseteq, \text{unit})\) be a NUPN. \(N\) is said to be unit safe iff it is safe and all its reachable markings are unit safe.

Note: Using P/T nets rather than elementary nets, the safeness condition (i.e., contact freeness) would not be needed to ensure that strict-firing and weak-firing rules coincide.
Unit safeness $\Rightarrow$ local mutual exclusion

Proposition 3. Let $(P, T, F, M_0, U, u_0, \subseteq, \text{unit})$ be a NUPN. For each marking $M$ and unit $u$, unit-safe $(M) \Rightarrow \text{card}(M \triangleright u) \leq 1$; that is, a unit-safe marking cannot contain two different local places of the same unit.

- In each unit, local places are mutually exclusive
- In terms of linear algebra:

$$\sum_{p \in \text{places}} (u) x_p \leq 1$$

- So, unit safeness implies safeness (in fact, from the definition)
- These are not S-invariants, but inequalities
  - because a given unit may lose its token
Unit safeness $\Rightarrow$ hierarchical mutual exclusion

Proposition 4. Let $(P, T, F, M_0, U, u_0, \sqsubseteq, \text{unit})$ be a NUPN. For each marking $M$ and units $(u, u')$, one has: unit-safe $(M) \land (M \triangleright u \neq \emptyset) \land (u' \sqsubseteq u \lor u \sqsubseteq u') \Rightarrow (M \triangleright u' = \emptyset)$; that is, if a unit-safe marking contains a local place of some unit $u$, it contains no local place of any ancestor or descendant unit $u'$ of $u$.

- Parent and children units are mutually exclusive
  - If a parent has a token, children have no token
  - If a child has a token, parents have no token
Linear-algebraic characterization

- Unit-safeness $\iff$ system of linear inequalities

**Proposition 6.** Let $(P, T, F, M_0, U, u_0, \subseteq, \text{unit})$ be a safe NUPN. $N$ is unit safe iff any reachable marking $M$ satisfies the following system of inequalities:

$$(\forall u \in \widehat{U}) \ (\forall u' \in \widehat{U} \mid u \subseteq u') \ \sum_{p \in \text{places}(u) \cup \text{places}(u')} x_p \leq 1 \quad (I_{u,u'})$$

where each variable $x_p$ is equal to 1 if place $p$ belongs to $M$, or 0 otherwise.

- Again, these are inequalities, not S-invariants
Some expressiveness results
How restrictive is unit safeness?

- Unit safeness is an (optional) property of NUPNs.
- Unit-safe NUPNs are well-adapted to encode:
  - (nested) **co-begin/co-end** programming schemes
  - **process calculi** terms (without recursion through parallel composition)
- Unit-safe NUPNs can also express:
  - all safe elementary nets
  - all nets having a state-machine decomposition

This is shown by translation to unit-safe NUPNs.
Elementary safe net → unit-safe NUPN

**Proposition 8.** Let \((P, T, F, M_0)\) be any ordinary, safe \(P/T\) net (i.e., a safe elementary net). There exists at least one 4-tuple \((U, u_0, \sqsubseteq, \text{unit})\) such that \((P, T, F, M_0, U, u_0, \sqsubseteq, \text{unit})\) is a unit-safe NUPN.

- NUPNs generalize safe elementary nets
- \(N\) places → \(N+1\) units
  - \(N\) units, one single place in each unit
  - one root unit having no local place
State-machine net $\rightarrow$ unit-safe NUPN

Proposition 9. Let $(P, T, F, M_0)$ be any ordinary $P/T$ net possessing a state-machine decomposition. There exists at least one 4-tuple $(U, u_0, \sqsubseteq, \text{unit})$ such that $(P, T, F, M_0, U, u_0, \sqsubseteq, \text{unit})$ is a unit-safe NUPN.

- NUPNs generalize state machines
- $N$ state machines $\rightarrow$ $N+1$ units
  - $N$ units, one per state machine
  - one root unit having no local place
The place-fusion abstraction
Place-fusion abstraction

■ Idea:
  ▶ merge all places of each unit into a single place
  ▶ perform reachability exploration on this abstracted net

■ Advantages:
  ▶ complexity reduction when units have many places
  ▶ useful to determine concurrent units  [Garavel-Serwe-06]

■ Place-fusion abstraction:
  ▶ preserves the NUPN property
  ▶ but does not preserve safeness, nor unit safeness
Optimized encodings for markings
Gains due to safeness / unit safeness

- **For safe nets:** markings can be encoded with one bit per place (rather than one integer per place)
- **For unit-safe nets:** further reductions are possible
  - **local reductions** (in each unit)
    
    N places in a unit $\Rightarrow$ N+1 local states
    
    $\lceil \log_2 (N+1) \rceil$ or $\lceil \log_2 (N) + 1 \rceil$ bits
  
  - **hierarchical reductions** (between parent/children units)
    "vertical" overlapping between:
    
    — the bits encoding the N places of a unit
    
    — the bits encoding all sub-units of this unit
Statistical results

- 5 encoding schemes compared on > 3500 NUPNs
- Best encoding: **local + hierarchical** reductions applied recursively on the tree of nested units
- Number of bits reduced by more than 60%

<table>
<thead>
<tr>
<th>scheme</th>
<th>overlapping</th>
<th>number of bits or Boolean variables</th>
<th>average size</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>no</td>
<td>$\sum_{i=1}^{n} N_i$ (i.e., $N$)</td>
<td>100.00%</td>
</tr>
<tr>
<td>(b)</td>
<td>no</td>
<td>$\sum_{i=1}^{n} \lceil \log_2(N_i + 1) \rceil$</td>
<td>40.52%</td>
</tr>
<tr>
<td>(c)</td>
<td>no</td>
<td>$\sum_{i=1}^{n} \lceil \log_2(N_i) \rceil + 1$</td>
<td>46.44%</td>
</tr>
<tr>
<td>(b)</td>
<td>yes</td>
<td>$\nu(u_0)$ with leaf$(u_j) \Rightarrow \nu(u_j) = \lceil \log_2(N_j + 1) \rceil$</td>
<td>39.35%</td>
</tr>
<tr>
<td>(c)</td>
<td>yes</td>
<td>$\nu(u_0)$ with leaf$(u_j) \Rightarrow \nu(u_j) = \lceil \log_2(N_j) \rceil + 1$</td>
<td>44.94%</td>
</tr>
</tbody>
</table>
H-W-B codes

- A useful metrics to measure NUPN complexity
- Metrics: a triple of integers, noted H-W-B
  - H is the **height** of the tree of nested units
    (the root unit does not count if it has no local place)
  - \( W \) is the **width** of the tree of nested units, i.e., the number of leaf units
    (if the NUPN is unit safe, \( W \) gives an upper bound on the number of tokens present in reachable markings)
  - B is the **number of bits** needed to represent markings using the best recursive encoding
- If \( B = \text{number of places} \), the code is noted --B (H=1, W=B)
Software support for NUPNs
The "nupn" file format

Textual format used by CADP tools

Concise, human-readable, easy to read and parse

!creator caesar
!unit_safe
places #5 0...4
initial place 0
units #3 0...2
root unit 0
U0 #1 0...0 #2 1 2
U1 #2 1...2 #0
U2 #2 3...4 #0
transitions #3 0...2
T0 #1 0 #2 1 3
T1 #1 1 #1 2
T2 #1 3 #1 4

The NUPN was created by the CAESAR tool.
The creator tool warrants that unit-safeness holds.
There are 5 places numbered from 0 to 4.
The initial marking contains only place 0.
There are 3 units numbered from 0 to 2.
The root unit is unit 0.
Unit 0 contains 1 place (0) and 2 sub-units (1, 2).
Unit 1 contains 2 places (1, 2) and no sub-unit.
Unit 2 contains 2 places (3, 4) and no sub-unit.
There are 3 transitions numbered from 0 to 2.
Trans. 0 has 1 input place (0) and 2 output places (1, 3).
Trans. 1 has 1 input place (1) and 1 output place (2).
Trans. 2 has 1 input place (3) and 1 output place (4).
The NUPN extension for PNML

- PNML: ISO standard for Petri nets (2011)
- A NUPN-specific extension of PNML has been defined for the Model Checking Contest

```
<toolspecific tool="nupn" version="1.1">
  <size places="5" transitions="3" arcs="7"/>
  <structure units="3" root="u0" safe="true">
    <unit id="u0">
      <places>p0</places>
      <subunits>u1 u2</subunits>
    </unit>
    <unit id="u1">
      <places>p1 p2</places>
      <subunits/>
    </unit>
    <unit id="u2">
      <places>p3 p4</places>
      <subunits/>
    </unit>
  </structure>
</toolspecific>
```

Where to find NUPN examples?

- MCC (Model Checking Contest)
  

In total: 147 NUPNs out of 628 P/T nets (23%)
Where to find NUPN examples?

VLPN (Very Large Petri Nets)  
(in preparation)

http://cadp.inria.fr/resources/vlpn

350 realistic benchmarks collected from diverse sources: CHP, EXP, Fiacre, LOTOS, LNT, applied pi-calculus, etc.

- Group 1: nets containing redundant units
- Group 2: nets containing disconnected places or transitions
- Group 3: unsafe nets
- Group 4: nets having one single unit  
  code: 1-1-B
- Group 5: unstructured nets  
  code: - - B
- Group 6: communicating automata  
  code: 1-W-B, with W ≥ 2
- Group 7: pseudo-communicating automata  
  code: 2-W-B
- Group 8: genuine NUPNs (concurrency + hierarchy)  
  code: H-W-B, with W ≥ 3
How to produce NUPNs?

- From "flat" Petri nets:
  - PNML2NUPN (Lom Messan Hillah, Paris)
  - Translation PNML → NUPN (applies Prop. 8)

- From networks of communicating automata:
  - EXP.OPEN (Frédéric Lang, Grenoble)
  - Translation EXP networks → NUPN (applies Prop. 9)

- From process calculi:
  - CAESAR (Hubert Garavel, Grenoble)
  - Translation LOTOS → NUPN (more involved!)
How to analyze NUPNs?

- **CAESAR.BDD** (Hubert Garavel, Grenoble)
  - syntax /static semantics checks on ".nupn" files
  - structural and behavioural properties using BDDs
  - translation NUPN → PNML

- **CAESAR.SDD** (Alexandre Hamez, Toulouse)

- **GreatSPN** (Elvio Amparore, Torino)

- **ITS-TOOLS** (Yann Thierry-Mieg, Paris)

- **LoLA** (Karsten Wolf & Torsten Liebke, Rostock)

- **LTSmin** (Jeroen Meijer & Jaco van de Pol, Twente)

- **PNMC** (Alexandre Hamez, Toulouse)
10 competing tools
4 tools supporting NUPNs
  - they won all golden medals
  - they won 73% of medals

Conclusion
Benefits of NUPNs

- They store more information than other models:
  - LTS: no concurrency – no locality – no hierarchy
  - Petri nets: concurrency – no locality – no hierarchy
  - NUPN: concurrency + locality + hierarchy

- NUPNs are easy to produce from process calculi, high-level nets, communicating automata, etc.

- NUPNs allow significant savings in state-space generation (60% less bits/Boolean variables)

- NUPNs smoothly integrate with existing tools: no major software rewrite needed
Challenging open issues

- Dedicated algorithms exploiting NUPN structure
  - to efficiently decide if a NUPN is unit-safe
  - to compute behavioural properties: deadlocks, etc.
  - to enhance partial-order / stubborn-set reductions

- Conversion of "flat" Petri nets to "optimal" NUPNs
  - "hierarchical" decomposition into state machines
  - goal: less units, more places per unit, maximal nesting

- NUPNs extended to support multiple tokens
  - relax unit-safeness constraint $\Rightarrow$ new flow relations
  - useful to encode process calculi with parallel recursion