Compact and Efficiently Verifiable Models for Concurrent Systems

Hernán Ponce de León

(joint work with Andrey Mokhov)

Séminaire Spécification-Vérification LIPN, Université Paris 13, March 12, 2015
Motivation

Controller

One-Hot ParSeq controller

Program

```plaintext
int x = 1

Thread 1:
local b = x;
x = 5

Thread 2:
local c = x;
```

Scenarios

Execution paths

```plaintext
x:=5
b:=1
c:=1

b:=1

x:=5

c:=5
b:=5
```
1 Compact Representation of Sets of Partial Orders
   - Labeled Event Structures
   - Conditional Partial Order Graphs
   - Conditional Labeled Event Structures

2 Parameterized Structures
   - Complexity
   - Transformations
Definition (Labeled Event Structure)

A LES is 4-tuple $\mathcal{E} = (E, \leq, \#, \lambda)$ such that

- $E$ is a set of events,
- $\leq \subseteq E \times E$ is a partial order (called causality) satisfying the property of finite causes, i.e.
  \[ \forall e \in E : |\{ e' \in E \mid e' \leq e\}| < \infty, \]
- $\# \subseteq E \times E$ is an irreflexive symmetric relation (called conflict) satisfying the property of conflict heredity, i.e.
  \[ \forall e, e', e'' \in E : e \# e' \land e' \leq e'' \Rightarrow e \# e'', \]
- $\lambda : E \rightarrow L$ is a labeling mapping.
A computation state of an event structure is called a configuration.

**Definition (Configuration)**

A set of events $C \subseteq E$ such that

- $C$ is causally closed: $e \in C \Rightarrow \forall e' \leq e : e' \in C$, and
- $C$ is conflict-free: $\forall e, e' \in C : \neg(e \neq e')$.

Maximal (w.r.t $\subseteq$) configurations of $\mathcal{E}$ are denoted by $\Omega(\mathcal{E})$. 

---

Diagram:

```
    a
   /\  
  e b
 / \ /  
c  d  
  \  
  d'  
```

---

```
    a
   /\  
  a  b
   /  
   c
```

---

```
    a
   /\  
  b  
 /  
 c  d
```
Another Example

Program

```
int x = 1

Thread 1:  Thread 2:  Thread 3:
local b = x;  x = 5  local c = x;
```

Execution paths

```
x := 5  b := 1  c := 1
    ↓    ↓    ↓
    x := 5  x := 5  x := 5
b := 5  c := 5  x := 5
    ↓    ↓    ↓
    b := 5  c := 5  b := 5
```

Hernán Ponce de León
Joint behavior of several LESs can be modeled with an initial choice:

![Diagram showing the joint behavior of multiple LESs](image)

**Synthesis:** each LPO is a LES with \( \# = \emptyset \)
Some LESs are more compact than others

Merge $e_1, e_2$ iff:

- $e_1 \neq e_2$
- $\lambda(e_1) = \lambda(e_2)$
- $\text{past}(e_1) = \text{past}(e_2)$
merging a’s

merging b’s

merging c’s and d’s
A CPOG is a 5-tuple \( G = (V, A, X, \phi, \rho) \), where

- \( V \) is a set of vertices,
- \( A \) is a set of arcs between them,
- \( X \) is a set of operational variables (an opcode is an assignment \((x_1, x_2, \ldots, x_{|X|}) \in \{0, 1\}^{|X|}\) of these variables),
- \( \rho \) restricts the possible opcodes,
- \( \phi \) assigns a Boolean condition to every vertex and arc.

Conditions allow to ”switch off” vertices/arcs.
A CPOG is *well-formed* if the projection over any opcode allowed by $\rho$ generates an acyclic graph.

The set of scenarios of a CPOG is denoted by $P(G)$.
The complexity of a CPOG is the total count of literals used in all the conditions: \( \sum_{e \in V \cup E} |\phi(e)| \)

Complexity: 12
The complexity of a CPOG is the total count of literals used in all the conditions: $\sum_{e \in V \cup E} |\phi(e)|$

Complexity: 12

- $(\bar{x} \lor \bar{y})$ needs to be computed only once $\rightarrow$ Complexity: 8
The complexity of a CPOG is the total count of literals used in all the conditions: \( \sum_{e \in V \cup E} |\phi(e)| \)

- \((\bar{x} \lor \bar{y})\) needs to be computed only once → Complexity: 8
- Can we do better? \((\bar{x} \lor \bar{y}) = \bar{x} \land \bar{y}\)
Synthesis: linear combination \( f_1 \cdot G_1 + f_2 \cdot G_2 + f_3 \cdot G_3 + f_4 \cdot G_4 \) with orthogonal \( f_k \) (algebra of graphs)

\[
\begin{align*}
\phi_b &\triangleq f_1 + f_2 + f_3 = \bar{x} \lor \bar{y} \\
\phi_{b \rightarrow d} &\triangleq f_1 + f_2 = \bar{x}
\end{align*}
\]
There is redundancy in LESs: each action is defined by its past and not by its label over $\lambda$. 

\[ \text{Definition (Conditional Labeled Event Structure)} \]
\[ A \text{ CLES is a tuple } H = (E, \rightarrow, #, \lambda, X, \phi, \rho) \text{ where } E \text{ is a set of events; } \rightarrow \text{ is a flow relation; } # \text{ is the conflict relation; } \lambda \text{ is a labeling function; } X \text{ is a set of variables; } \phi \text{ assigns Boolean conditions to } E, \rightarrow \text{ and } #; \text{ and } \rho \text{ is the restriction function.} \] 
A CLES is called well formed if any opcode allowed by the restriction function generates a LES.
There is redundancy in LESs: each action is defined by its past and not by its label over $\lambda$.

**Definition (Conditional Labeled Event Structure)**

A CLES is a tuple $H = (E, \rightarrow, \#, \lambda, X, \phi, \rho)$ where $E$ is a set of events; $\rightarrow$ is a flow relation; $\#$ is the conflict relation; $\lambda$ is a labeling function; $X$ is a set of variables; $\phi$ assigns Boolean conditions to $E, \rightarrow$ and $\#$; and $\rho$ is the restriction function.
There is redundancy in LESs: each action is defined by its past and not by its label over $\lambda$.

**Definition (Conditional Labeled Event Structure)**

A CLES is a tuple $H = (E, \rightarrow, \#, \lambda, X, \phi, \rho)$ where $E$ is a set of events; $\rightarrow$ is a flow relation; $\#$ is the conflict relation; $\lambda$ is a labeling function; $X$ is a set of variables; $\phi$ assigns Boolean conditions to $E, \rightarrow$ and $\#$; and $\rho$ is the restriction function.

A CLES is called *well formed* if any opcode allowed by the restriction function generates a LES.
CLESSs generalize CPOGs and LESs:

- $\# = \emptyset \rightarrow \text{CPOG}$
- $\rightarrow$ is acyclic and $\phi = 1 \rightarrow \text{LES}$
1 Compact Representation of Sets of Partial Orders
   - Labeled Event Structures
   - Conditional Partial Order Graphs
   - Conditional Labeled Event Structures

2 Parameterized Structures
   - Complexity
   - Transformations
CLESs, CPOGs and LES allow to represent set of LPOs and support conditional elements.
CLESs, CPOGs and LES allow to represent set of LPOs and support conditional elements.

**Definition**

A mathematical structure defined on a set of elements $S$ is called a *parameterised structure* if the elements are labeled with Boolean conditions $\phi : S \rightarrow \Phi$, where $\Phi \subseteq X \rightarrow \{0, 1\}$.

- CLES: $S = events, arcs, \#$
- CPOG: $S = vertices, arcs$
- LES: $S = events, arcs, \#$ with $\phi(S) = 1$
Predicates in $\Phi$ can be computed by a Boolean circuit.

The decoding complexity of a predicate set is the number of variables and gates in the smallest circuit computing it.
**Complexity (2)**

**Definition**

*Complexity* of a parameterised structure with predicate set $\Phi$ on a set of elements $S$ is the decoding complexity of $\Phi$ + the number of elements on $S$.

The complexity of the CPOG is 17 (2 variables + 4 gates + 5 vertices + 6 arcs).
## Comparison of Parameterized Structures

| Name                          | Scenarios | Complexity |
|-------------------------------|-----------|------------|
|                               |           | CPOG | LES  | CLES |
| Phase encoder                 | 24        | 24   | 158  | 24   |
|                               | 120       | 35   | 825  | 35   |
|                               | 720       | 48   | 5001 | 48   |
| Decision tree                 | 8         | 44   | 36   | 36   |
|                               | 16        | 97   | 76   | 76   |
|                               | 32        | 195  | 156  | 156  |
| Trees of phase encoders       | 16        | 70   | 108  | 58   |
|                               | 29        | 43   | 158  | 41   |
|                               | 32        | 136  | 220  | 114  |
## Comparison of Parameterized Structures (2)

| Name             | Scenarios | CPOG | LES | CLES |
|------------------|-----------|------|-----|------|
| ARM Cortex M0    | 5         | 26   | 28  | 26   |
|                  | 6         | 27   | 35  | 27   |
|                  | 7         | 26   | 38  | 26   |
|                  | 8         | 28   | 43  | 28   |
|                  | 9         | 28   | 46  | 28   |
|                  | 10        | 29   | 46  | 29   |
|                  | 11        | 30   | 50  | 30   |
| Intel 8051       | 5         | 34   | 48  | 34   |
|                  | 6         | 35   | 52  | 35   |
|                  | 7         | 36   | 56  | 36   |
|                  | 8         | 37   | 56  | 37   |
|                  | 9         | 46   | 71  | 46   |
|                  | 10        | 47   | 81  | 47   |
|                  | 11        | 51   | 90  | 51   |
A LES is an acyclic CLES with $\phi = 1$
If we remove conflicts we obtain an acyclic CPOG. How to preserve this information?
If we remove conflicts we obtain an acyclic CPOG. How to preserve this information? **adding conditions!**
If we remove conflicts we obtain an acyclic CPOG. How to preserve this information? **adding conditions!**

Conditions replace conflicts:

- **only one event must be selected in each conflict**
- **an event can be selected only if their predecessors are selected**
- **configurations should be maximal**
If we remove conflicts we obtain an acyclic CPOG. How to preserve this information? **adding conditions!**

\[
\rho = (\bigwedge_{e \# f} \neg \phi_e \lor \neg \phi_f)(\bigwedge_{e \leq f} \phi_f \Rightarrow \phi_e)(\bigwedge_{e \in E} \phi_e \lor \bigvee_{e \# f} \phi_f)
\]
If we remove conflicts we obtain an acyclic CPOG. How to preserve this information? adding conditions!

\[
\rho = \left( \bigwedge_{e \neq f} \neg \phi_e \lor \neg \phi_f \right) \left( \bigwedge_{e \leq f} \phi_f \Rightarrow \phi_e \right) \left( \bigwedge_{e \in E} \phi_e \lor \bigvee_{e \neq f} \phi_f \right)
\]
Several events are labeled equally: we can fold the graph back
Several events are labeled equally: we can fold the graph back.
Merged vertices represent several events: $\lor$ their conditions.
Several events are labeled equally: we can fold the graph back.
Merged vertices represent several events: $\lor$ their conditions.
Theorem

Given an LES $\mathcal{E} = (E, \leq, \#, \lambda)$, the algorithm constructs a CPOG $G = (V, A, X, \phi, \rho)$ such that $P(G) = \Omega(\mathcal{E})$.

Proof: Let $H$ be the intermediate CLES, we show that projections of $H$ coincide with maximal configurations of $\mathcal{E}$ and $P(H) = P(G)$.
Let $\psi$ be a valid opcode

\[
\iff
\text{events of } H|_{\psi} \text{ are conflict free, causally closed and maximal}
\iff
\text{events of } H|_{\psi} \text{ form a maximal configuration of the CLES}
\iff
\text{events of } H|_{\psi} \text{ form a maximal configuration of the LES}
\]
- \( P(G) \subseteq P(H) \): trivial; no behavior is added

- \( P(H) \subseteq P(G) \):
  - since \( \lambda(e) = \lambda(e_1) \) (or \( \lambda(e) = \lambda(e_2) \)), labels are preserved
  - since folding = \( \forall e' \in E : e' \leq e \Leftrightarrow e' \leq e_1 \lor e' \leq e_2 \), dependencies are preserved
Complexity

**Theorem**

Given an LES $\mathcal{E} = (E, \leq, \#, \lambda)$, the algorithm constructs a CPOG $G = (V, A, X, \phi, \rho)$ of complexity $\Theta(|\mathcal{E}|)$.

**Proof:**

- $|V| \leq |E|$
- $e \rightarrow f \iff e_1 \leq f \lor e_2 \leq f$, then $|A| \leq |\leq|$
- Label each event with one variable, then $|X| = |E|$ and $\phi_{e \rightarrow f} = \phi_e \land \phi_f$ (|$\leq|$ gates)
- For $\rho$:
  1. 1 gate for each conflict
  2. 2 gates for dependencies
  3. $2|\#| + |E|$ gates for maximality
Overall idea: unfold the CPOG keeping conditions that will be replaced by conflicts.
Overall idea: **unfold** the CPOG keeping **conditions** that will be replaced by **conflicts**.

Possible extensions need to be computed at each step; find a set of predecessor events $P \subseteq E$ such that

- the vertex is active;
- its predecessors and their corresponding arcs are active;
- non predecessor are either not active or its corresponding arc is not active;
- the instance of the vertex is different to any other

\[
\phi_a \land \left( \bigwedge_{\substack{e_b \in P \\ b \rightarrow a \in A}} \phi_{e_b} \land \phi_{b \rightarrow a} \right) \left( \bigwedge_{\substack{e_b \in E \setminus P \\ b \rightarrow a \in A}} \neg \phi_{e_b} \lor \neg \phi_{b \rightarrow a} \right) \left( \bigwedge_{e_a \in E} \neg \phi_{e_a} \right)
\]
Transformations: from CPOGs to LESs

Initially only $a$ is a possible extension; for $b$ the encoding includes $\neg \phi_{a \rightarrow b} = 0$. 
When \( E = \{ e_a \} \), \( e \) and \( b \) are extensions with \((x \lor y)\) and \((\overline{x} \land \overline{y})\).
When $E = \{e_a, e_b, e_e\}$, $c$ and $d$ are extensions with $\bar{y}$ and $\bar{x}$. 
When \( E = \{ e_a, e_b, e_{c_1}, e_{d_1}, e_e \} \), \( c \) and \( d \) are extensions again with \( \overline{x} \land y \) and \( \overline{y} \land x \).
Transformations: from CPOGs to LESs

When $E = \{e_a, e_b, e_{c_1}, e_{c_2}, e_{d_1}, e_{d_2}, e_e\}$ the formula reduces to false; there are not possible extensions.
Mutual exclusive conditions are replaced by conflicts; $e_b \# e_e$ is added.
Mutual exclusive conditions are replaced by conflicts; $e_b \# e_e$ is added.
Theorem

Let $G = (V, A, X, \phi, \rho)$ be a well-formed CPOG and $\mathcal{E} = (E, \leq, \#, \lambda)$ the LES obtained by the unfolding procedure, then $\Omega(\mathcal{E}) = P(G)$.

Idea of the proof: unfolding + conditions = CLES; projections of the CLES coincide with configurations of the LES and scenarios of the CPOG.

Theorem

Given a CPOG and a prefix of its unfolding, deciding if an instance of a vertex is a possible extension is NP-hard.
LESs and CPOGs represent compactly sets of partial orders.
CLESs have the advantages of both.
New metric to compare the formalisms.
Two transformation methods without explicit enumeration of all the scenarios.

Future work:
Comparison against merged processes
Validation on larger examples (data mining, multithreaded programs)