PSL vs. CCSL

Régis Gascon Julien DeAntoni Frédéric Mallet

INRIA Sophia-Antipolis - EPI AOSTE

Université Nice Sophia-Antipolis

SAFA Workshop - October 6th, 2010

High level specification

- System design needs descriptions at different abstraction level.
- MARTE: UML profile for real time systems.
- MARTE/CCSL can be used for high level specification.
- CCSL observers allow checking an implementation (VHDL) conforms the specification. [André, Mallet, & DeAntoni '10]
- PSL assertions can also be used to express required properties on the system (lower level).
  - model checking,
  - equivalence checking.

Why comparing PSL and CCSL?

- CCSL and PSL are used in similar roles though designated for different purposes.
- Clock Constraint Specification Language
  - constructs from net theory and synchronous languages.
  - timed and causal patterns common in embedded systems.
- Property Specification Language
  - textual language for temporal logic.
  - sugaring operators to raise the abstraction level.

Agenda

- Definitions:
  - Clock Constraint Specification Language,
  - Property Specification Language.
- Preliminary comparisons.
- Translations (fragments).
- Conclusion.

Clock Constraint Specification Language (CCSL)

- CCSL is based on *clocks* (sequence of event occurrences).
- CCSL specification define constraints between the occurrences of clocks.
- CCSL relations
  - $c_1 \boxed{\subset} c_2$ (subclocking),
  - $c_1 \boxed{\sharp} c_2$ (exclusion),
  - $c_1 \boxed{\prec} c_2$ (precedence), $c_1 \boxed{\preceq} c_2$ (weak precedence),
  - $c_1 \boxed{\sim} c_2$ (alternance).

Clock Constraint Specification Language (CCSL)

- CCSL is based on *clocks* (sequence of event occurrences).
- CCSL specification define constraints between the occurrences of clocks.
- CCSL relations
- CCSL expressions ($\rightsquigarrow$ building new clocks)
  - $c_1 \boxed{+} c_2$ (union), $c_1 \boxed{*} c_2$ (intersection), $c_1 \boxed{-} c_2$ (difference),
  - $c_1 \boxed{\blacktriangledown} bw$ (filtering),
  - $c_1 \boxed{\wedge} c_2$ (greatest lower bound) $c_1 \boxed{\vee} c_2$ (lowest upper bound),
  - etc...

Semantics (examples)

Models are of the form $\sigma : \mathbb{N} \longrightarrow 2^{\mathrm{VAR}}$.

- $c_1 \boxed{\subset} c_2$ (subclocking)

c_2	$\top$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\ldots$
c_1	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\top$	$\ldots$

Semantics (examples)

Models are of the form $\sigma : \mathbb{N} \rightarrow 2^{\mathrm{VAR}}$.

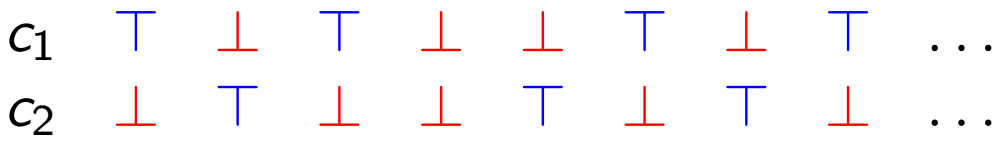
- $c_1 \boxed{\subset} c_2$ (subclocking)
- $c_1 \boxed{\preceq} c_2$ (weak precedence)

c_1	$\top$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\bot$	$\bot$	$\ldots$
	1	2	3	3	4	5	5	5	
c_2	$\bot$	$\top$	$\bot$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\ldots$
	0	1	1	2	3	3	4	5	

Semantics (examples)

Models are of the form $\sigma : \mathbb{N} \longrightarrow 2^{\mathrm{VAR}}$.

- $c_1 \boxed{\subset} c_2$ (subclocking)
- $c_1 \boxed{\preceq} c_2$ (weak precedence)
- $c_1 \boxed{\sim} c_2$ (alternance) $\rightsquigarrow$ bounded precedence



Semantics (examples)

Models are of the form $\sigma : \mathbb{N} \rightarrow 2^{\mathrm{VAR}}$.

- $c_1 \boxed{\subset} c_2$ (subclocking)
- $c_1 \boxed{\preceq} c_2$ (weak precedence)
- $c_1 \boxed{\sim} c_2$ (alternance)

- $c \boxed{\triangleq} c_1 \boxed{+} c_2$ (union)

c_1	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\top$	$\ldots$
c_2	$\bot$	$\top$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\ldots$
c	$\bot$	$\top$	$\top$	$\bot$	$\top$	$\bot$	$\top$	$\top$	$\ldots$

Semantics (examples)

Models are of the form $\sigma : \mathbb{N} \longrightarrow 2^{\mathrm{VAR}}$.

- $c_1 \boxed{\subset} c_2$ (subclocking)
- $c_1 \boxed{\preceq} c_2$ (weak precedence)
- $c_1 \boxed{\sim} c_2$ (alternance)

- $c \boxed{\triangleq} c_1 \boxed{+} c_2$ (union)
- $c \boxed{\triangleq} c_1 \boxed{\wedge} c_2$ (glb)

c_1	$\top$	$\top$	$\bot$	$\bot$	$\top$	$\top$	$\bot$	$\top$	...
	1	2	2	2	3	4	4	5	...
c_2	$\bot$	$\top$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	...
	0	1	2	3	4	4	4	4	...
c	$\bot$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\bot$	$\bot$	...
	0	1	2	2	3	4	4	4	

Clock Constraint Specification Language (CCSL)

- CCSL is based on *clocks* (sequence of event occurrences).
- CCSL specification define constraints between the occurrences of clocks.
- CCSL relations
- CCSL expressions ($\rightsquigarrow$ building new clocks)
- CCSL specification
  - Clocks
  - + Definitions: $c \boxed{\stackrel{\Delta}{=}} < \mathit{Expression} >$
  - + Relations

  seen as a conjunction of constraints.

Property Specification Language (PSL)

- IEEE standard for hardware assertion
- LTL + regular expressions + sugaring operators
- Same expressiveness than LTL.
- Here, we consider the following kernel:
  - Boolean formula:

$$b ::= p \mid \neg b \mid b \wedge b \mid b \vee b$$

  - Regular expressions

$$r ::= b \mid r \cdot r \mid r \cup r \mid r \cap r \mid r^*$$

  - PSL properties

$$\phi ::= r \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg \phi \mid \mathsf{X}\phi \mid \phi \mathsf{U} \phi$$

PSL semantics

Models are also of the form $\sigma : \mathbb{N} \rightarrow 2^{\mathrm{VAR}}$

- $\sigma, i \models p$ iff $p \in \sigma(i)$.
- Boolean cases and regular expressions are standard.
- $\sigma, i \models \mathrm{X}\phi$ iff $\sigma, i+1 \models \phi$ (ϕ holds at the next position.)

$$p \quad \bot \quad \top \quad \ldots \quad \models \mathrm{X}p$$

- $\sigma, i \models \phi \mathrm{U} \psi$ iff there exist $j \geq i$ such that
  - $\sigma, j \models \psi$ and
  - for all $i \leq k < j$, $\sigma, k \models \phi$.

  (ϕ holds until a position where ψ holds).

$$\begin{array}{cccccc} p & \top & \top & \top & \ldots & \\ q & \bot & \bot & \top & \ldots & \end{array} \models p \mathrm{U} q$$

PSL semantics

Models are also of the form $\sigma : \mathbb{N} \rightarrow 2^{\mathrm{VAR}}$

- $\sigma, i \models p$ iff $p \in \sigma(i)$.
- Boolean cases and regular expressions are standard.
- $\mathsf{F}\phi \equiv \top \mathsf{U} \phi$ (ϕ eventually holds.)

$$p \quad \bot \quad \bot \quad \ldots \quad \bot \quad \top \quad \ldots \quad \models \mathsf{F}p$$

- $\mathsf{G}\phi \equiv \neg \mathsf{F} \neg \phi$ (ϕ always holds.)

$$p \quad \top \quad \top \quad \top \quad \ldots \quad \models \mathsf{G}p$$

Preliminary Comparisons

- CCSL and PSL share common models,
- but none of these languages is included in the other.

- CCSL is designed to express safety:
  "Something bad never happens" (Gϕ in PSL).

- PSL cannot express precedence, glb and lub
  (needs unbounded counter).

Comparable fragments

What are the subsets that can be translated ?

We choose to bound some CCSL operators:

- Precedence: $c_1 \boxed{\preceq_n} c_2$ means
  - c_1 precedes c_2,
  - but the advance of c_1 never exceeds n.
- Glb, Lub (bound on the relative advance).

On the other hand, we consider the safety fragment of PSL.

Lemma

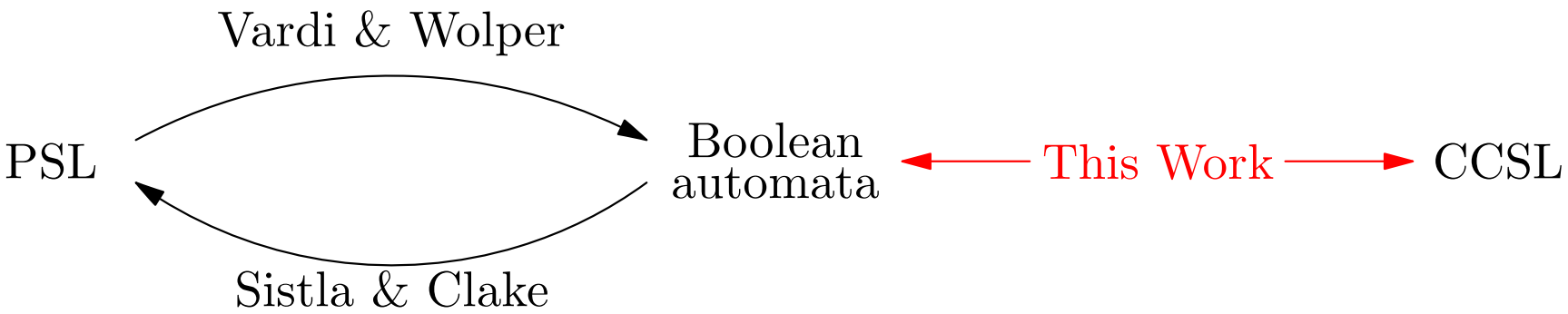
Every bounded CCSL specification can be encoded in PSL.

Lemma

Every safety PSL formula can be encoded in CCSL.

Boolean Automata

- We use an automaton based approach.
- Boolean automaton: automaton whose transitions are labeled by Boolean formulas.



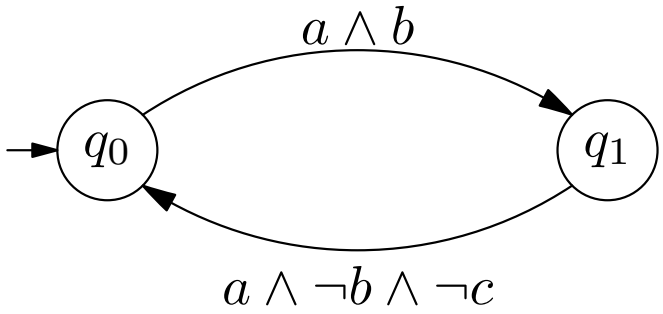
Boolean automaton $\longrightarrow$ CCSL (example)

This kind of translation is limited by the fact that CCSL cannot express liveness (Fϕ in temporal logic).

Lemma

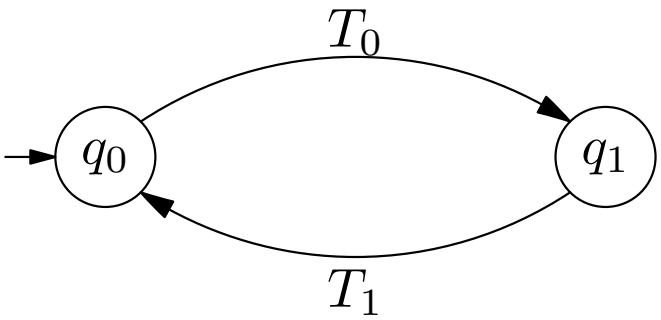
If every state is accepting, a Boolean automaton can be simulated by a CCSL specification.

Boolean automaton $\longrightarrow$ CCSL (example)



- Transitions:
  - $T_0 \boxed{\triangleq} a \boxed{*} b$
  - $T_1 \boxed{\triangleq} a \boxed{-} (b \boxed{+} c)$
- Global clock and initialization:
  - $Glob \boxed{\triangleq} q_0 \boxed{+} q_1 \boxed{+} a \boxed{+} b \boxed{+} c$
  - $Init \boxed{\triangleq} Glob \boxed{\blacktriangledown} 10^{\omega}$

Boolean automaton $\longrightarrow$ CCSL (example)



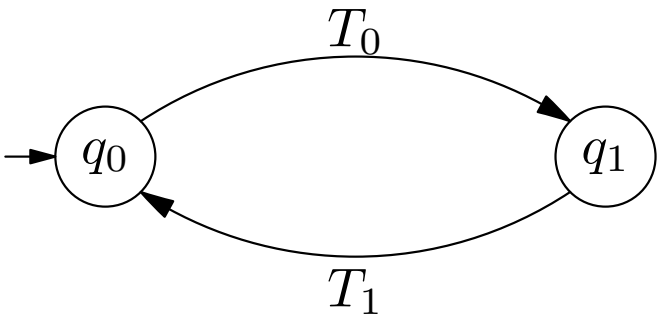
- At each instant either a state or a transition occurs:

$$Glob \boxed{=} (q_0 \boxed{+} q_1) \boxed{+} T_0 \boxed{+} T_1$$

- Two states cannot occurs at the same time

$$q_0 \boxed{\sharp} q_1$$

Boolean automaton $\longrightarrow$ CCSL (example)



- Each states alternate with its output:
  - $q_0 \boxed{\sim} T_0$
  - $q_1 \boxed{\sim} T_1$
- and with its input
  - $T_1 \cup \mathit{Init} \boxed{\sim} q_0$
  - $T_0 \boxed{\sim} q_1$

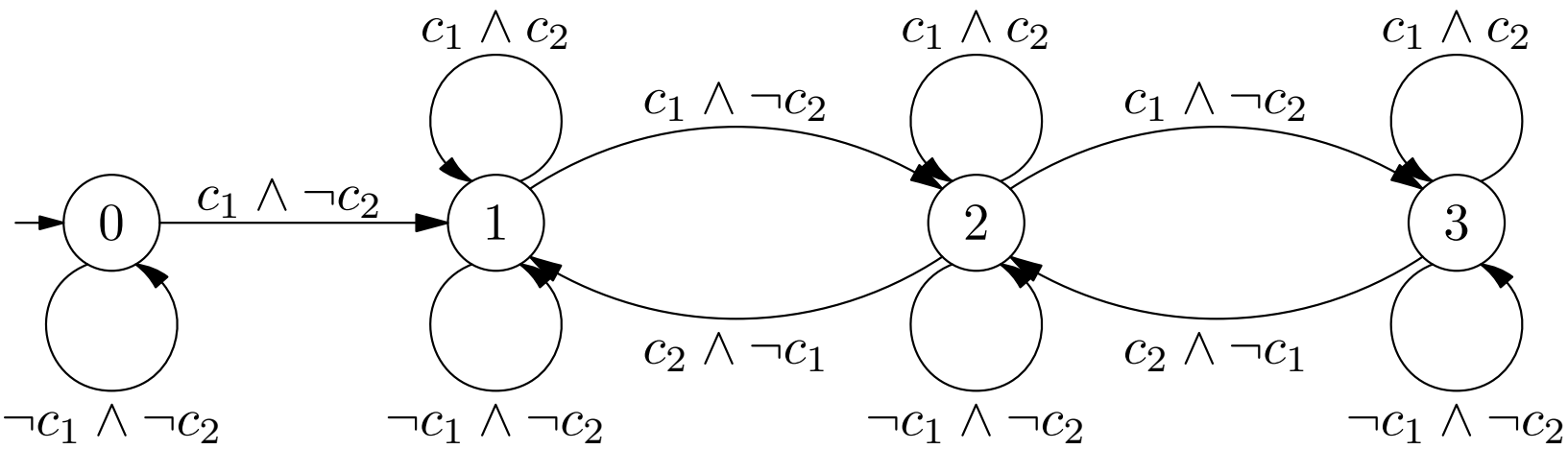
Boolean automaton ← CCSL (example)

Boolean automata cannot encode unbounded counters.

Lemma

Every CCSL specification in the bounded fragment can be encoded by a Boolean automaton.

Example: $c_1 \boxed{\preceq_3} c_2$



Main results

Lemma

The safety fragment of PSL can be simulated in CCSL.

Indeed, the translation of safety PSL to Boolean automata is such that every state is accepting.

Lemma

The bounded fragment of CCSL can be simulated in PSL.

By intermediate translation to Boolean automata or directly.

Concluding remarks

- Contributions:
  - Identification of differences between CCSL and PSL.
  - Translations for large fragments of these languages.

- Sometimes the context naturally bounds precedence (see e.g. alternance).

- PSL misses data (integer, strings) to efficiently encode CCSL (even the bounded fragment).

Future work

- Adding temporal modalities to a CCSL like language.
- Checking that a CCSL specification induce a finite state system.
- Defining CCSL libraries:
  - for PSL subsets,
  - for state machines.

Questions ?