

SAFE CCSL SPECIFICATION: A SUFFICIENT CONDITION

Frédéric Mallet¹, Jean-Vivien Millo¹, Robert de Simone²

¹Univ. Nice Sophia Antipolis, CNRS I3S, INRIA

²INRIA Sophia-Antipolis

CCSL AND TIMESQUARE


✗ CCSL: Clock Constraint Specification Language

```
*Alternates.extendedCCSL ✕
/*
 * CCSL specification for Defer Test
 * @author: Julien Deantoni
 * date : Wed feb 9th 2011
 */
ClockConstraintSystem MySpec {
  imports {
    import "csl:kernel" as kernelLib ;
    import "csl:lib" as CCSL;
    import "my.uml" as model;
  }
  entryBlock main

  Block main {
    Clock c1 -> evt1("model->p1")
    Clock c2 -> evt2("model->TheClass")
    Relation r1[Alternates](AlternatesLeftClock -> c1, AlternatesRightClock -> c2 )
    Expression expression_0=Intersection( Clock1-> ,Clock2-> )

  }
}
```

Marte profile

 <http://timesquare.inria.fr>

OUTLINE

- ✘ Motivation and overview
- ✘ Expressing the sufficient condition
- ✘ Conclusion

MOTIVATION

- ✗ Let us consider a clock **c** in a specification **spec**
- ✗ Liveness(**c**): **c** can tick/occur infinitely often.

- ✗ Model checking (LTL/CTL)
 - + State-based representation of the specification
 - ✗ State-based representation of each constraint[1]
 - ✗ A composition operator[1]
 - ✗ Have a finite state space

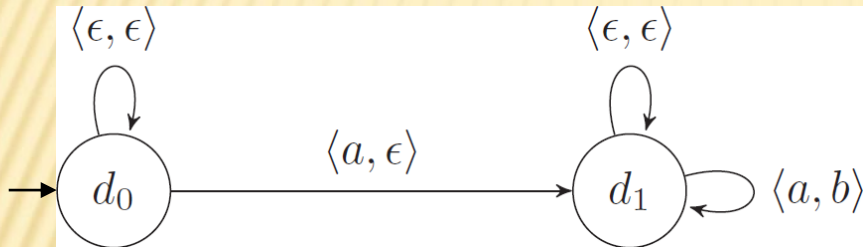


FINITE AND INFINITE CCSL CONSTRAINTS

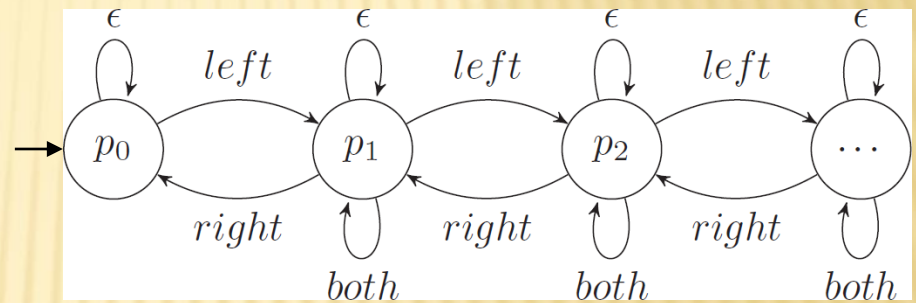
- ✗ Some constraints are finite
 - + coincidence, exclusion, subset, delay, or, and
- ✗ Some constraints are infinite
 - + Precedence, max, min
- ✗ Some products of **infinite** constraints are **finite**
- ✗ Ex: alternate

FINITE AND INFINITE CCSL CONSTRAINTS

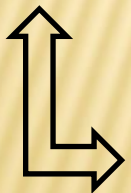
$b = a$ delayed by 1



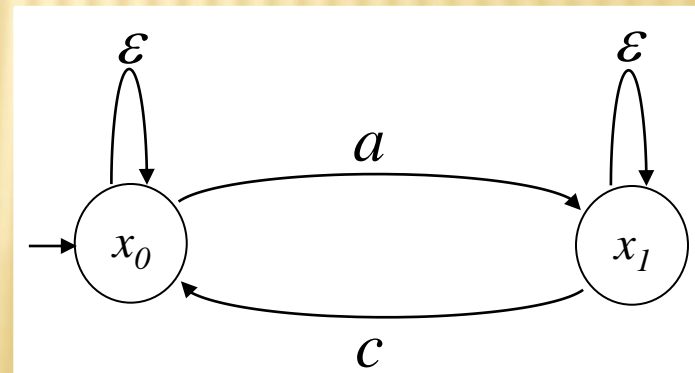
left precedes right



a alternates with c



a precedes c
 $b = a$ delayed by 1
 c precedes b

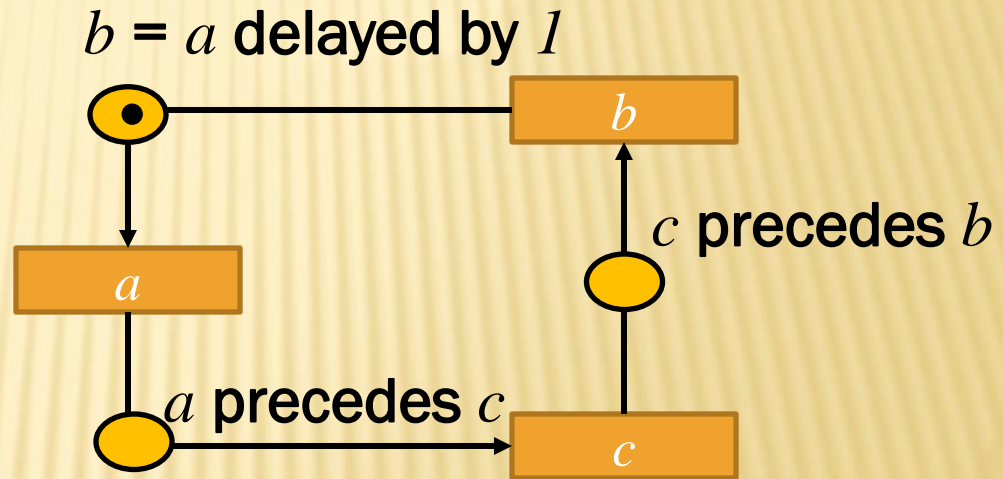


BUILDING THE STATE SPACE

- ✗ Problem: the composition operation terminates only when the state space is finite.
- ✗ Strategy: detect boundedness before composing.
- ✗ Solution: 1/ build a MG of precedence
2/ check for strong connections

EXAMPLE: ALTERNATES

a precedes *c*
b = *a* delayed by 1
c precedes *b*

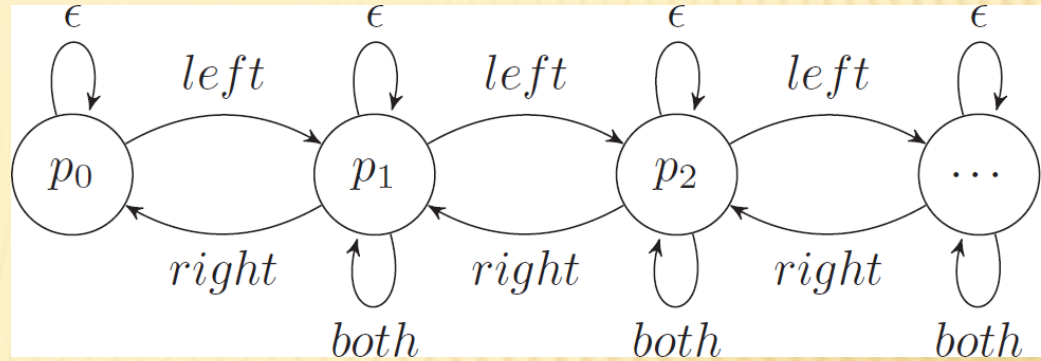


OUTLINE

- ✘ Motivation and overview
- ✘ Expressing the sufficient condition
- ✘ Conclusion

STATE SPACE

left precedes right

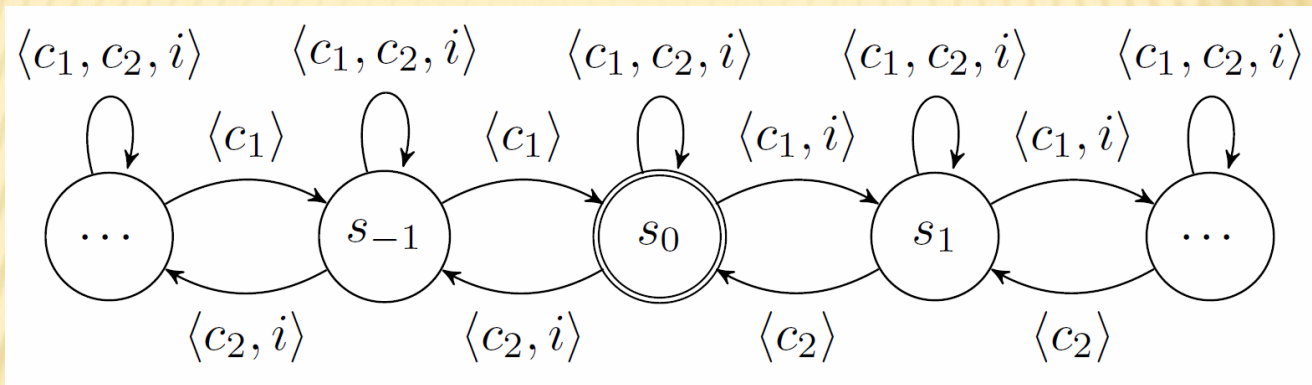


- ✘ The state space is based on $\delta = X_{\text{left}} - X_{\text{right}}$ counters
- ✘ δ 's are integer representations of precedencies
- ✘ Precedencies are induced by every constraint
 - + ...and capture its state space
- ✘ A precedence is equivalent to a place in a MG



INFINITE STATE CONSTRAINTS

✘ $i = \text{inf}(c_1, c_2)$: i ticks with the first of (c_1, c_2)

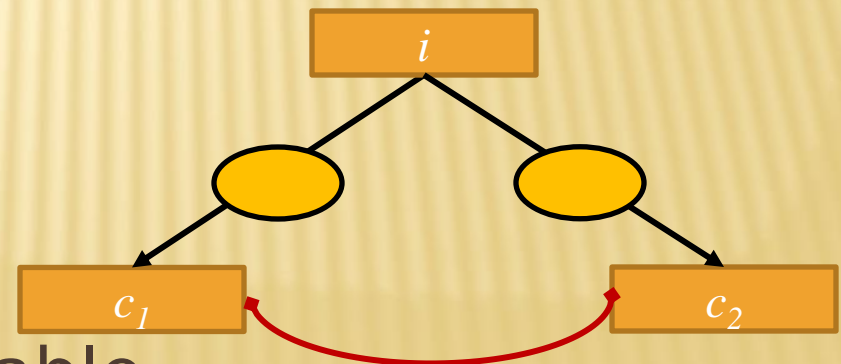


Here,
 $\delta = X_{c_1} - X_{c_2}$

✘ Corresponding precedence relation:

+ i precedes c_1

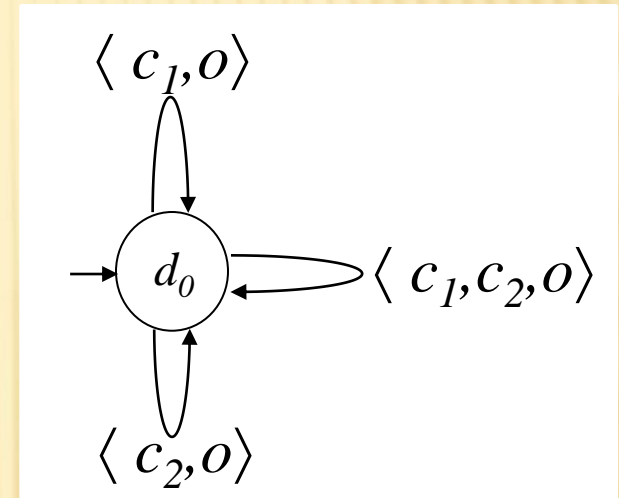
+ i precedes c_2



✘ c_1 and c_2 are synchronizable

FINITE STATE CONSTRAINTS

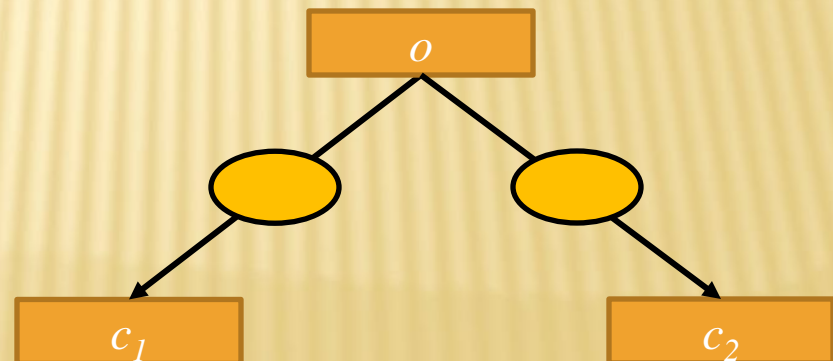
✗ $o = c_1$ or c_2



✗ Corresponding precedence relation:

+ o precedes c_1

+ o precedes c_2



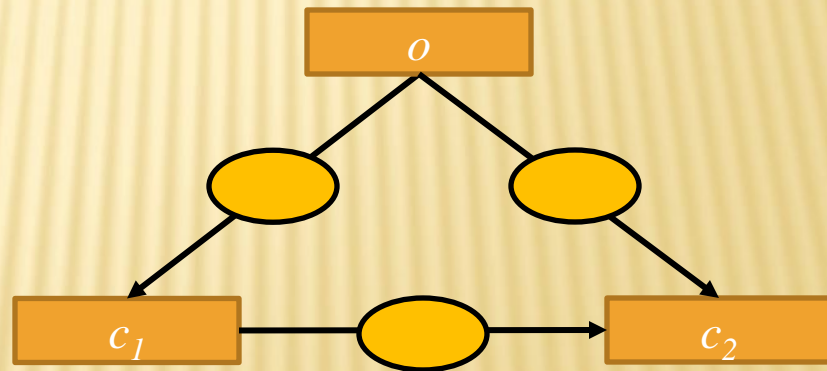
MG ABSTRACTION OF A CCSL SPEC

- ✗ Every clock \rightarrow A transition
- ✗ Every constraint \rightarrow precedence(s) \rightarrow place(s)

Clock c_1, c_2, o

c_1 precedes c_2

$o = c_1$ or c_2



BOUNDEDNESS CONDITION

- ✗ For every synchronizable relation
 - + The two transitions belong to the same SCC
 - + In a MG, all transitions of a SCC have the same asymptotic rate [Commoner et al. 1971]

Clock a, b, c, i

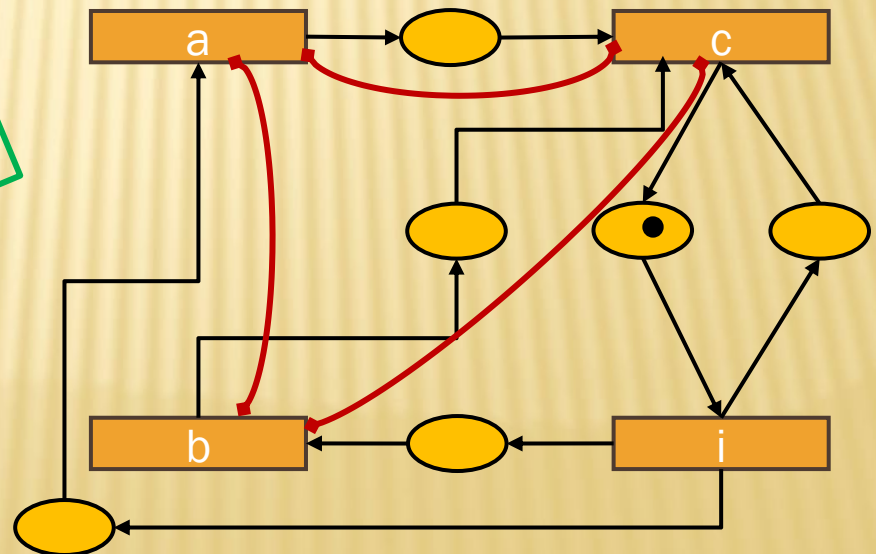
$i = \inf(a, b)$

a precedes c

b precedes c

c alternates with i

Safe!



OUTLINE

- ✘ Motivation and overview
- ✘ Expressing the sufficient condition
- ✘ Conclusion

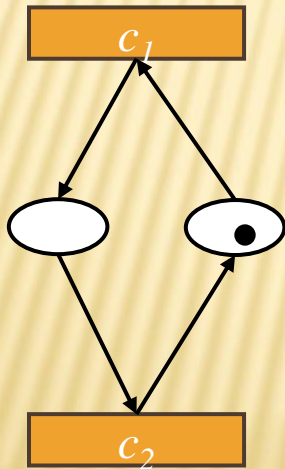
CONCLUSION

- ✗ We present a sufficient condition to detect bounded/safe CCSL specifications
 - + We use a MG abstraction of CCSL
 - + The condition is probably also necessary
 - ✗ but it is not proved.
 - + “Clock death” has not been considered

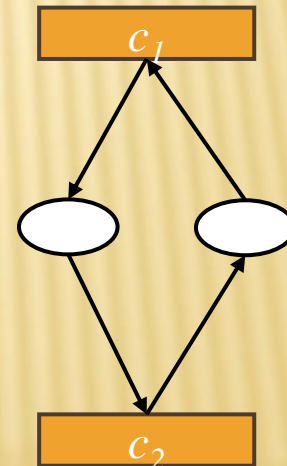
FUTURE WORK

✘ Universal deadlock detection:

c_1 alternates with c_2



c_1 precedes c_2
 c_2 precedes c_1



QUESTIONS?

- ✘ MARTE profile: <http://www.omgmarte.org>
- ✘ Timesquare: <http://timesquare.inria.fr>
- ✘ [1] F. Mallet and J-V. Millo, Boundedness issues in CCSL specifications, ICFEM 2013
- ✘ [2] Commoner, Holt, Even, Pnueli; Marked Directed Graphs, 1971