Building Synchronous DataFlow graphs with UML & MARTE/CCSL

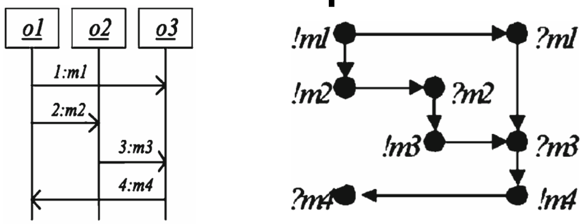
F. Mallet, J. DeAntoni, C. André, R. de Simone

Aoste - INRIA/I3S

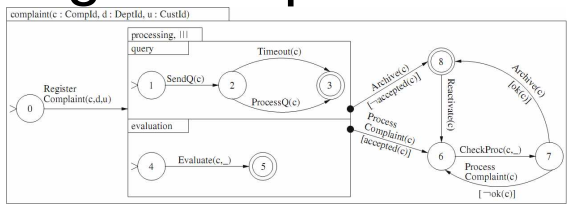
Université de Nice Sophia Antipolis

UML & Formal methods

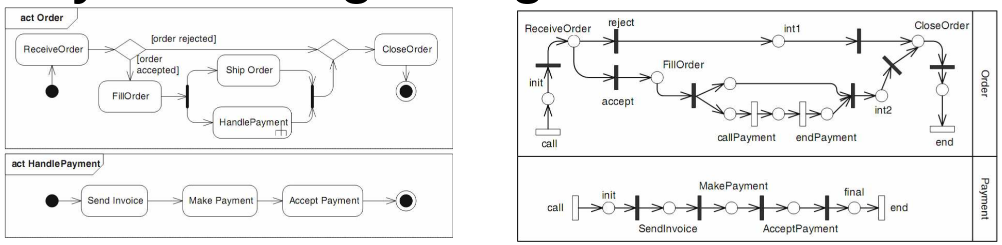
- Ambiguity and structural properties of basic sequence diagrams
  - Interactions + trace



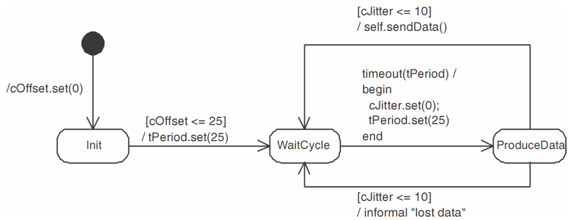
- Extending statecharts with process algebra operators
  - Untimed StateMachines + CSP



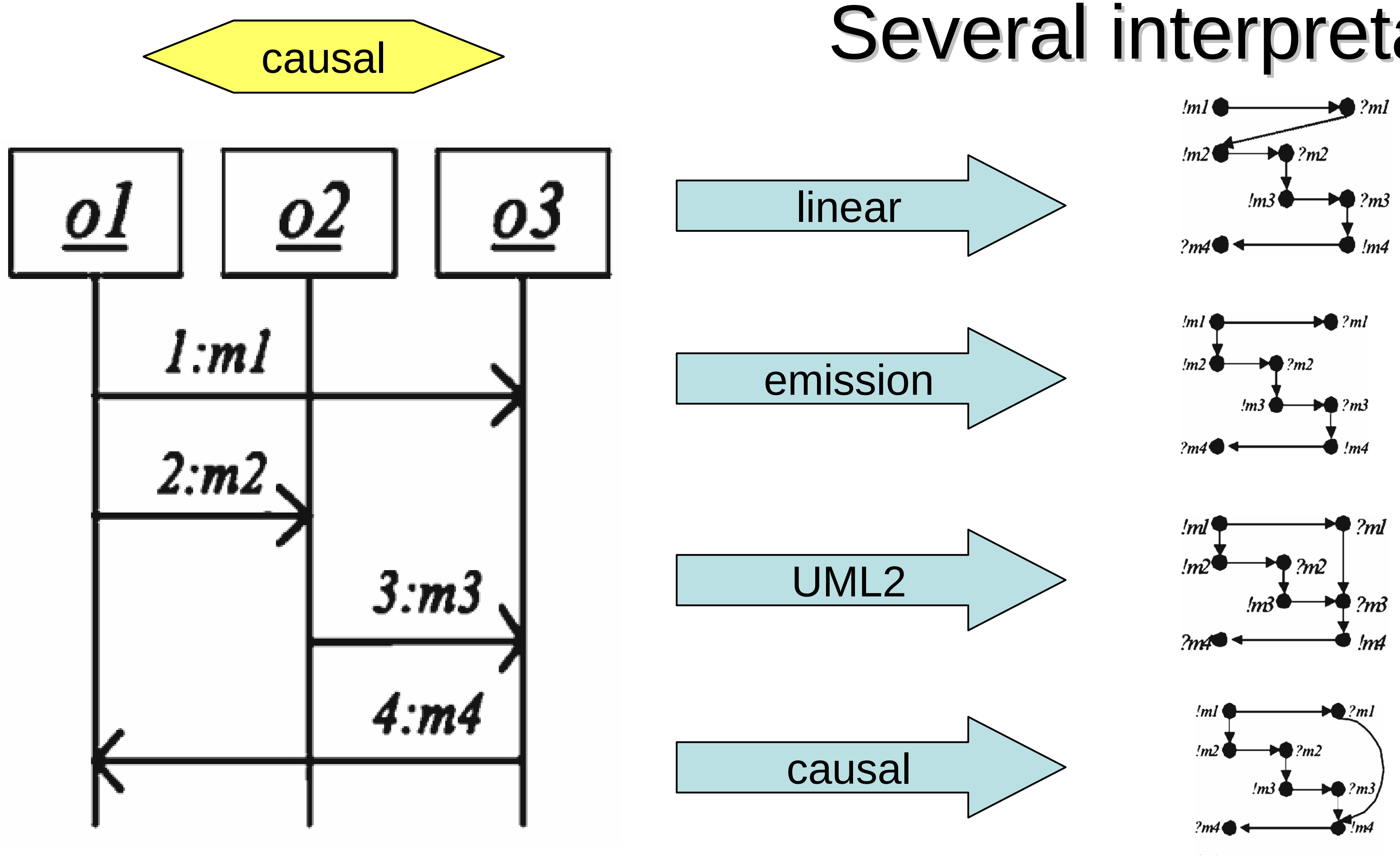
- UML Behavioral consistency checking using instantiable Petri Nets
  - Activities + PN



- Timing analysis and validation with UML
  - StateMachines + Timed automata

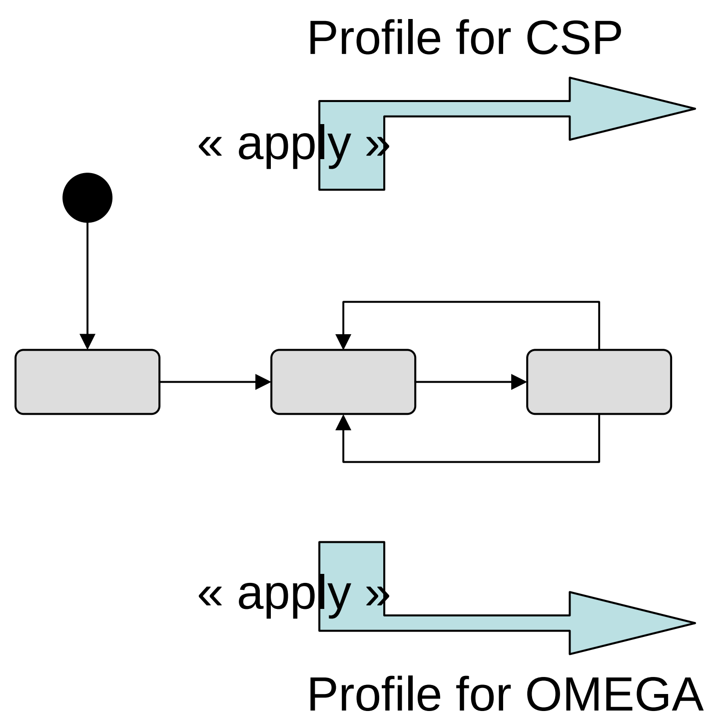


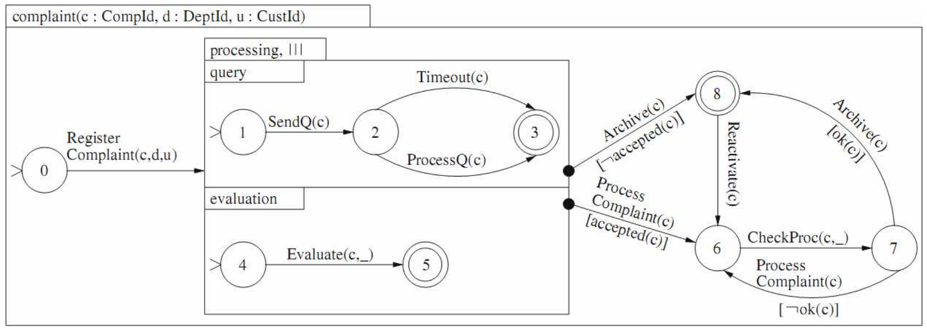
Several interpretations

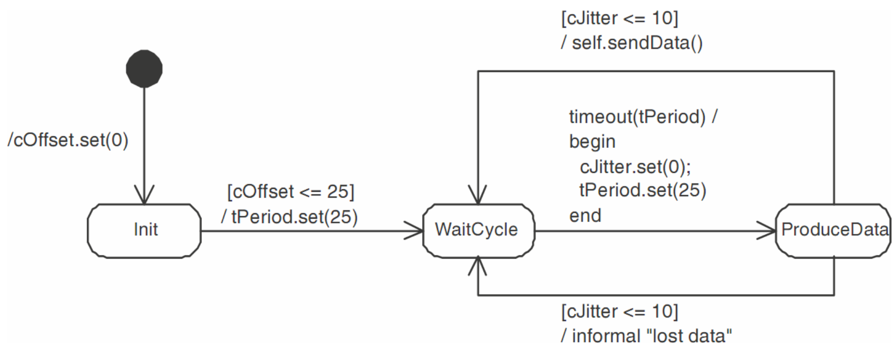


- Compare their constraining power : level of concurrency
- Do we want to choose between all these ?
  - Use UML as a framework for combining all of these semantics
  - Apply directors (like in Ptolemy) to choose the suitable semantics

Profiles to give the semantics







- How to combine these two diagrams ?
  - Put them next to each others ?

Proposition

- What
  - Explicit execution semantics within the model

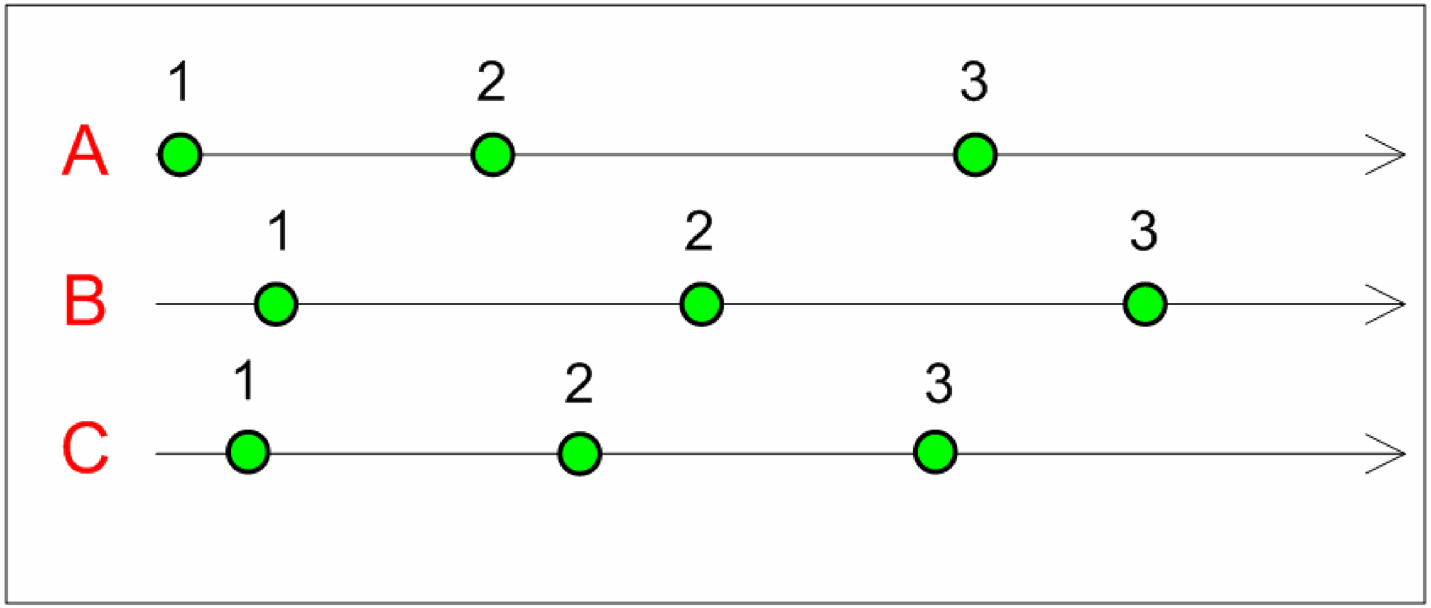
- How
  - Annotate the meta-model
  - Execution semantics defined with MARTE/CCSL

- **UML Profile for MARTE and CCSL**
  - **M**odeling and **A**nalysis of **R**eal-**T**ime and **E**mbedded systems
    - Time model => Timed Causality Semantics to UML models
  - CCSL: MARTE Companion **Modeling Language**
    - Apply to any (EMF) model => UML or not

- Example
  - UML (activity/state machine): **S**ynchronous **D**ata**F**low graphs
  - CCSL Library for SDF

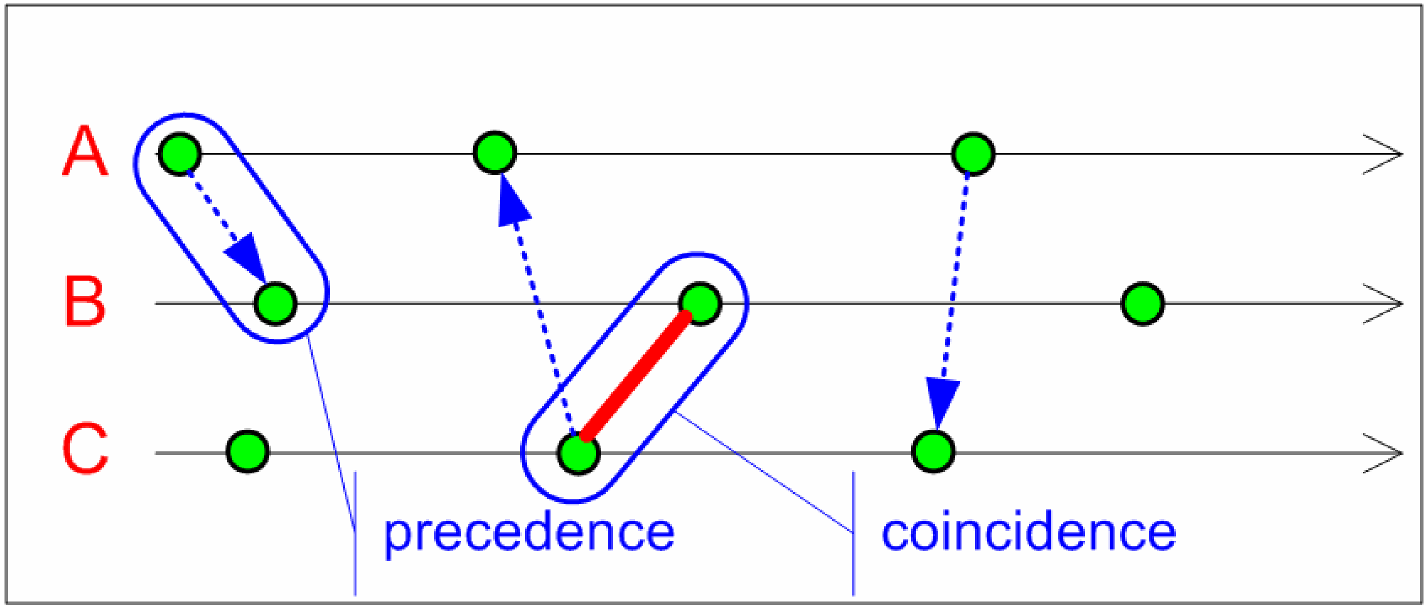
CCSL – Polychronous systems

- Clock Model
  - Clock C=$\langle \mathcal{I}, \prec \rangle$, infinite ordered set of instants
    - Discrete-time clocks: $\mathcal{I}$ is discrete and indexed by $\mathbb{N}^*$



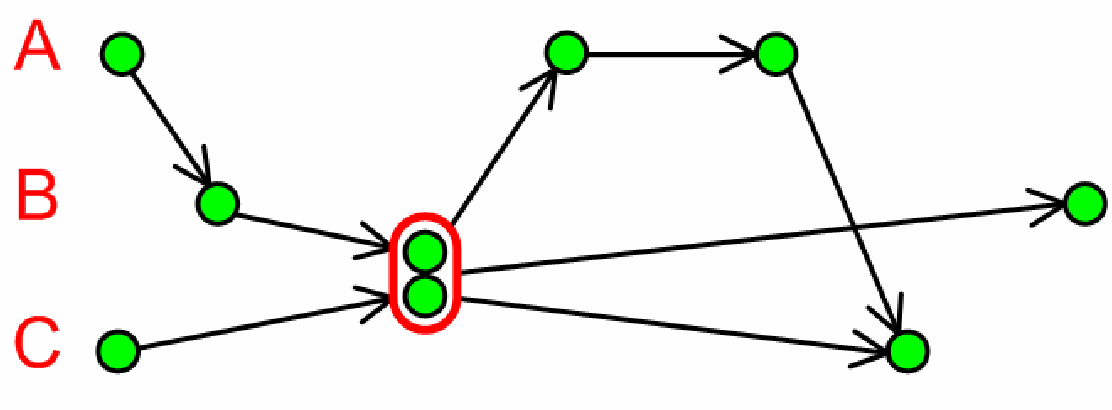
CCSL – Polychronous systems

- Clock Model
  - Clock C=$\langle \mathcal{I}, \prec \rangle$, infinite ordered set of instants
    - Discrete-time clocks: $\mathcal{I}$ is discrete and indexed by $\mathbb{N}^*$
  - Instant relations: coincidence, (strict) precedence, exclusion
  - Clock relations:
    - infinitely many instant relations according to predefined patterns (periodicity, alternation, sampling, …)



CCSL – Polychronous systems

- Clock Model (static)
  - Clock C=$\langle \mathcal{I}, \prec \rangle$, infinite ordered set of instants
  - Instant/clock relations = **constraints**
- Time system (dynamic)
  - Clocks = set of boolean variables
  - Constraints = set of boolean equations => SAT problem



Event structures [Winskel]

Occurrence nets [Petri]

TaggedSystems [LSV]

CCSL clock constraint - precedence

Precedence

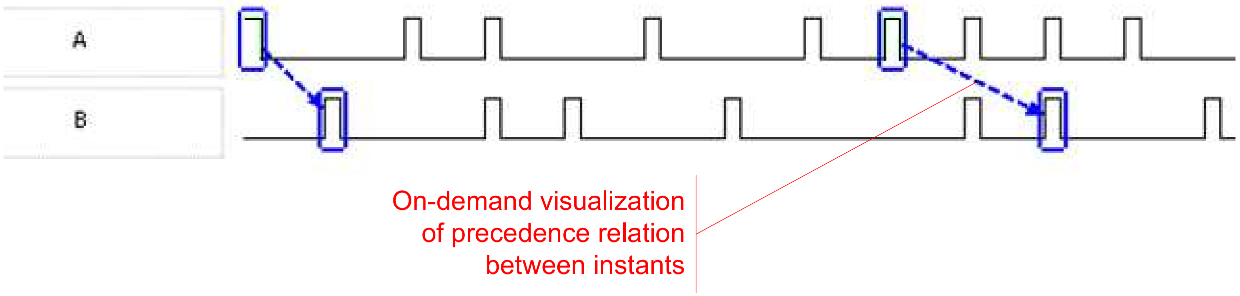
A precedes B (strict form) written as $A \prec B$

Semantics $(\forall k \in \mathbb{N}^*)\ A[k] \prec B[k]$

$$\frac{\beta \triangleq (\chi(A) = \chi(B))}{[\![A \prec B]\!] = (\beta \Rightarrow \neg B)}$$

Logical representation

Simulation



Use: causal dependency or asynchronous communication

CCSL clock constraint - Synchrony

Synchrony

A = B written as $A \boxed{=} B$

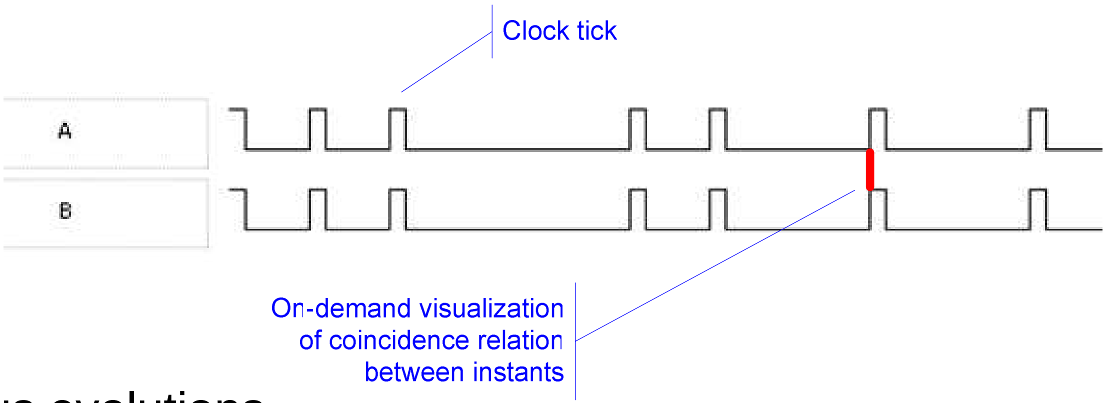
Semantics

$$\left(\forall k \in \mathbb{N}^{*}\right) A[k] \equiv B[k]$$

$$[\![A \boxed{=} B]\!] = \left([\![A]\!] = [\![B]\!]\right)$$

Logical representation

Simulation



Use: synchronous evolutions

CCSL clock constraint– filtering

Filtering

B = A filteredBy w written as $B \equiv A \blacktriangledown w$ where $w \in \mathbb{B}^{\omega}$ (infinite) Binary Word

Semantics

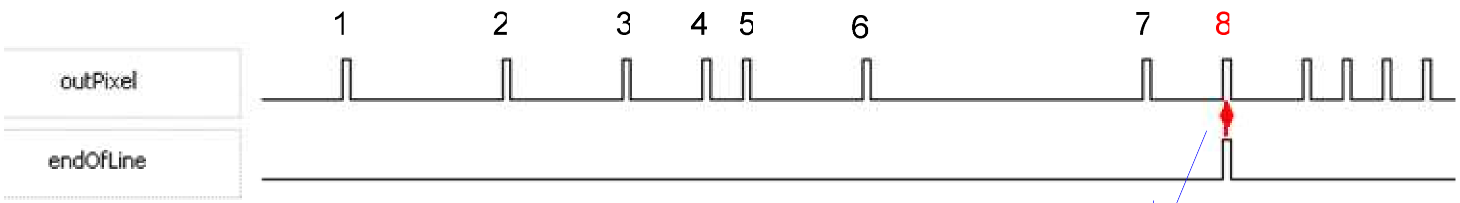
$$(\forall k \in \mathbb{N}^{*})\ B[k] \equiv A[w \uparrow k]$$

where $w \uparrow k$ is the index of the k^{th} 1 in w

$$\frac{\beta \ \square\ (w = 1.v)}{\square A \blacktriangledown w \square = (\beta \wedge \square A \square)}$$

Logical representation

Simulation



$$endOfLine \equiv outPixel \blacktriangledown (0^{7}.1)^{\omega}$$

On-demand visualization of coincidence relation between instants

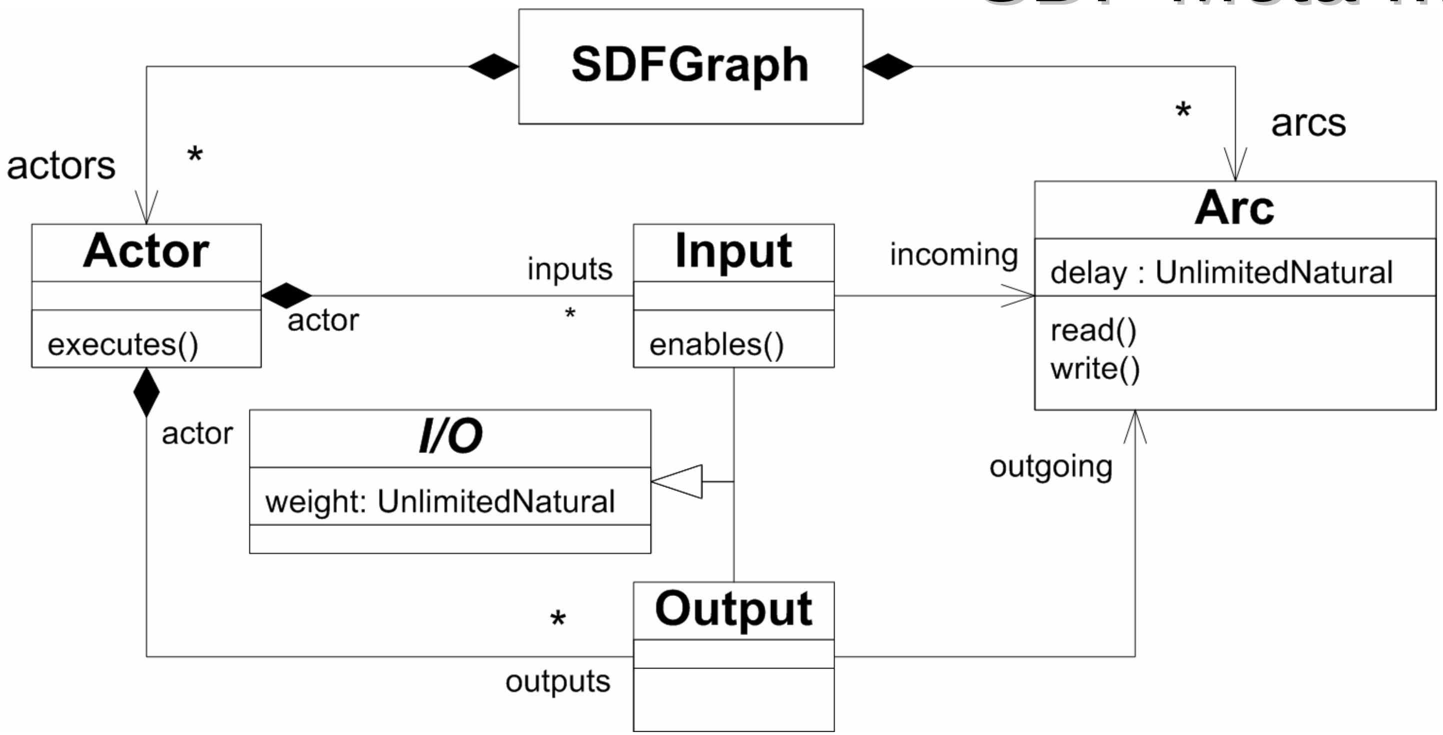
Use: a special case of synchrony (on selected instants)

Synchronous Data Flow (SDF)

- Data Flow graphs
  - Directed graphs
  - Nodes = functions/computations
  - Arcs = data path

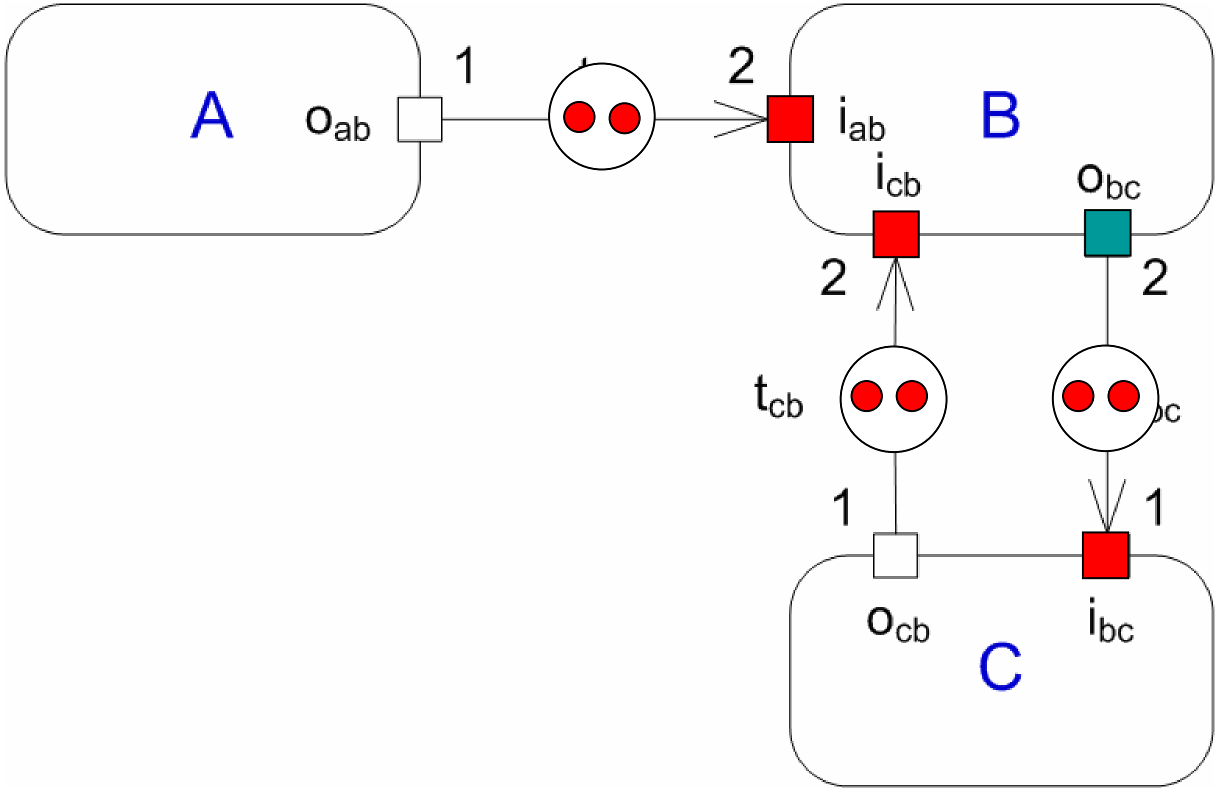
- Synchronous Data Flow [E.A. Lee, 1987]
  - Static number of data samples consumed/produced by each node
  - Well-suited for multi-rate **DSP algorithms** with continuous stream of data
  - Reduction of Kahn-Process Networks to allow **static scheduling** and ease parallelization
  - Equivalent to Computation Graphs [Karp & Miller, 1966]
  - Popular due to Ptolemy developed in Berkeley

SDF Meta-model



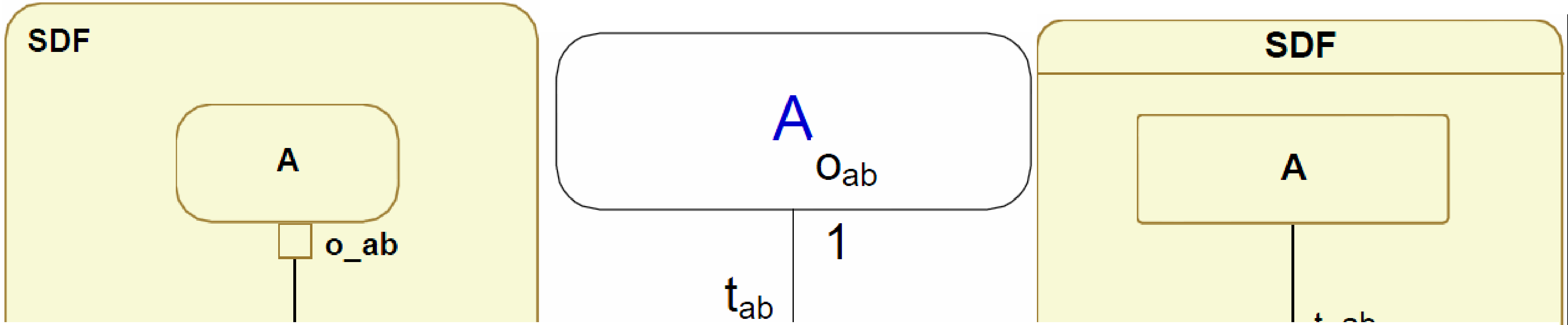
- Nodes are called actors
- Arcs have a delay
- Input/Output have a weight
  - Number of data samples consumed/produced

SDF Example



- Equivalent to a Marked-Event Graph
  - Conflict-free Petri Net
  - Static scheduling: **A A B A A B**
    C C

How to model SDF graphs in UML ?

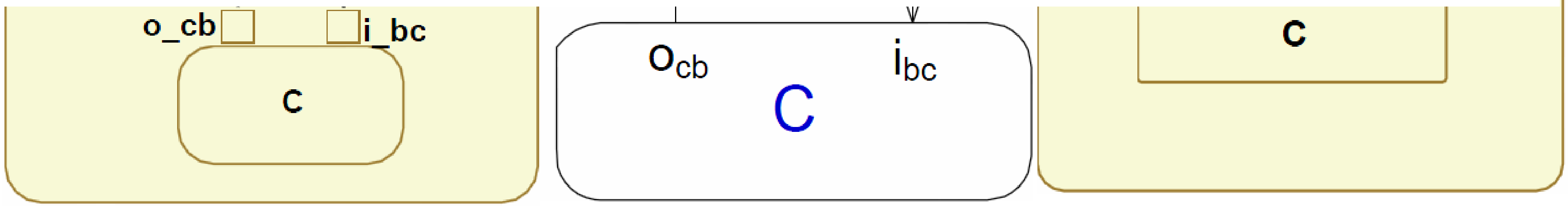


Where is the semantics ?

Is that compatible with the UML semantics ?

CCSL makes the semantics explicit …

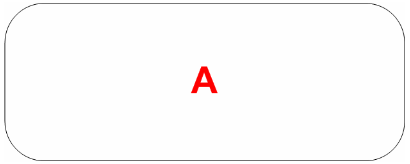
… within the model



CCSL Library for SDF (1/2)

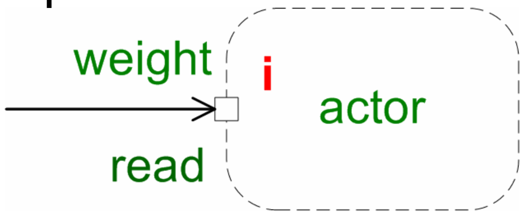
SDF

- Actor A

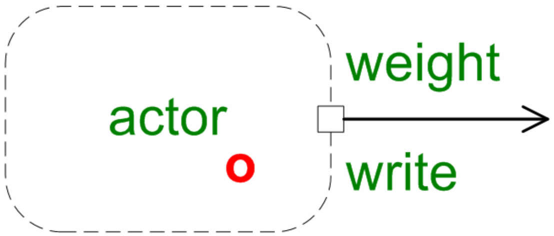


- Token T

- Input i



- Output o



CCSL

- Clock A;
- Clock write, read;

$$\text{def } token(\text{clock } write,\ \text{clock } read,\ \text{int } delay) \triangleq write \boxed{\prec} (read \text{ delayedFor } delay)$$

$$\text{def } input(\text{clock } actor,\ \text{clock } read,\ \text{int } weight) \triangleq (read \text{ by } weight) \boxed{\prec} actor$$

$$\text{def } output(\text{clock } actor,\ \text{clock } write,\ \text{int } weight) \triangleq actor \boxed{=} \left(write \text{ filteredBy } (1.0^{weight-1})^{\omega}\right)$$

CCSL Library for SDF (2/2)

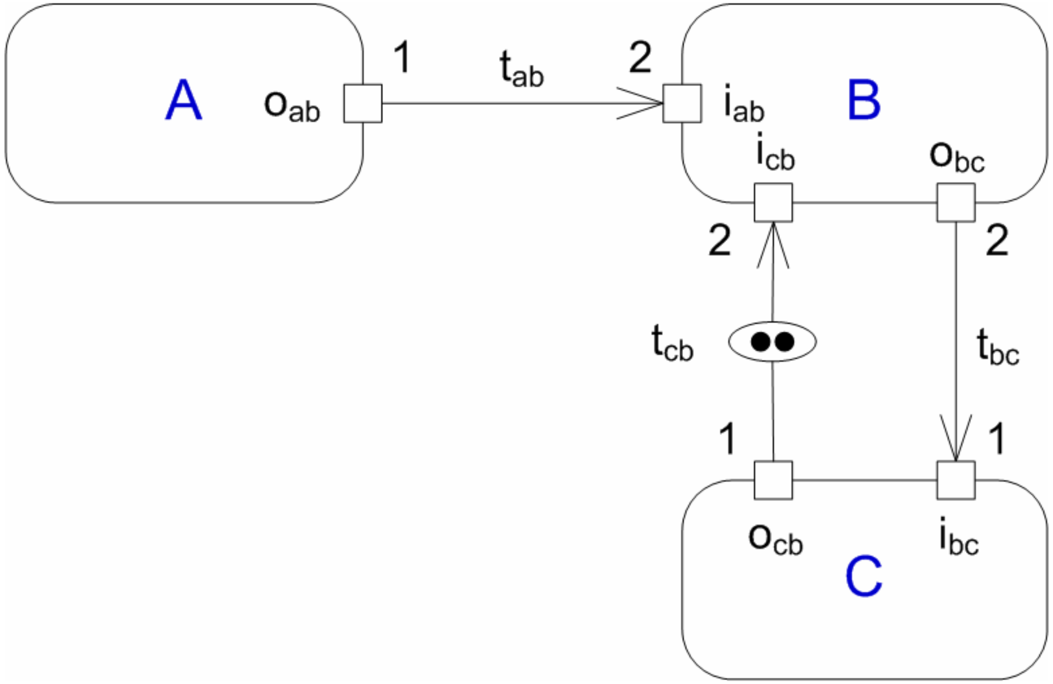
- SDF

- CCSL

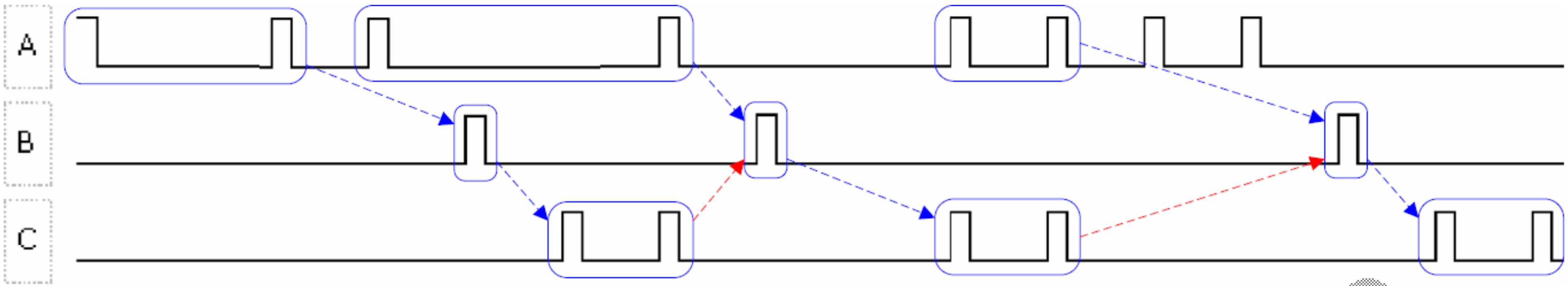
def arc(int $delay$, clock $source$, int out, clock $target$, int in) $\triangleq$

$output(source, write, out); token(write, read, delay); input(target, read, in)$

Example



$$\mathcal{S} = arc(0, A, 1, B, 2) \mid arc(0, B, 2, C, 1) \mid arc(2, C, 1, B, 2)$$



Conclusion

- (UML) Models must come with
  - A meta-model to describe the structural/composition rules
  - An explicit execution semantics
- CCSL can be used for describing
  - Temporal patterns
  - Causal relationships
- MARTE: attach CCSL specifications to UML models
- TimeSquare can
  - execute CCSL specifications
  - Animate DI2 models in Papyrus
  - http://www.inria.fr/sophia/aoste/time_square/