ULB UNIVERSITÉ LIBRE DE BRUXELLES

Almost ASAP Semantics : From Timed Models to Timed Implementations

M. De Wulf, L. Doyen, J.-F. Raskin

University of Brussels

Centre Fédéré en Vérification

Motivations

- Embedded Controllers
  - … are difficult to develop (concurrency, real-time, continuous environment, …).
  - … are safety critical.

➡ Use model-based development :

Timed Automata and
Reachability Analysis

Model-based development

- Make a model of the environment:
  Env
- Make clear the control objective:
  Bad
- Make a model of the control strategy:
  ControllerModel
- Verify:
  Does Env || ControllerModel avoid Bad ?
- Good, but after ?

Goal

- Transfer of verified properties from models to code.
- Type of models we consider:
  - Controllers specified as timed automata

Problems

- Timed automata are (in general) not implementable (in a formal sense)...

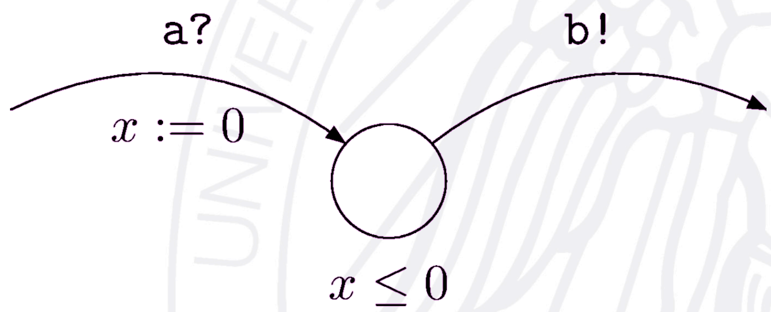
  Why ?

  - Zenoness : 0, 0.5, 0.75, 0.875, ...
  - No minimal bound between two transitions : 0,0.5,1,1.75,2,2.875,3,...
  - And more ... (robustness)

More Problems

- One can specify instantaneous response but not implement it.

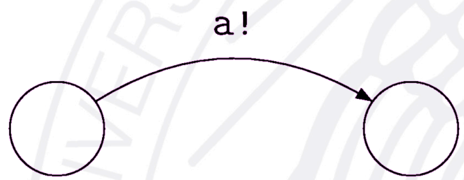
Not implementable



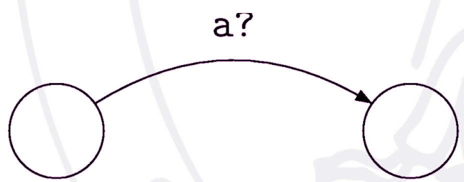
More Problems

- Instantaneous synchronisation between environment and controller is not implementable.

Environment



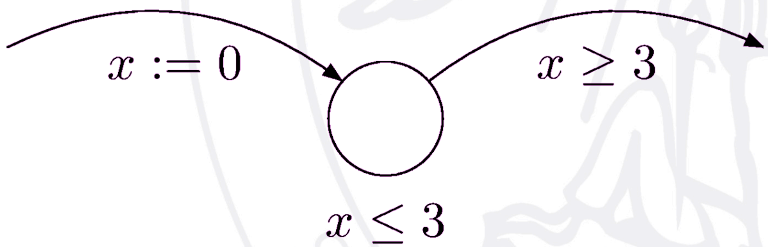
Classical controller
Not implementable



More Problems

- Models use continuous clocks and implementation uses digital clocks with finite precision

Classical controller
Not implementable



Problems : Summary

- My controller stragegy may be correct because :
  - it is zeno;
  - it acts faster and faster;
  - it reacts instantaneously to events and timeouts (synchrony hypothesis);
  - it uses infinitely precise clocks.

A possible solution...

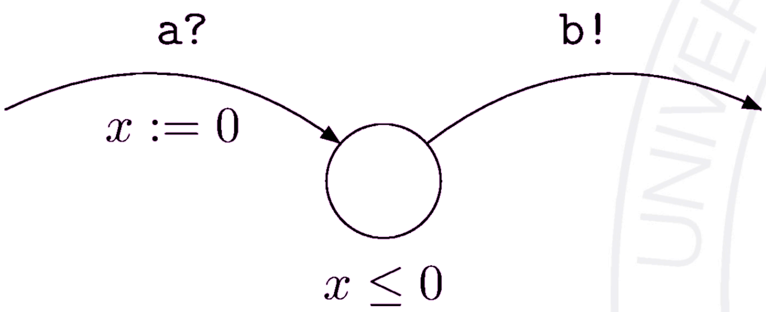
- Give an alternative semantics to timed automata : Almost ASAP semantics.
  - enabled transitions of the controller become urgent only after Δ time units;
  - events from the environment are received by the controller within Δ time units;
  - guards are enlarged by Δ.

  where Δ is a parameter

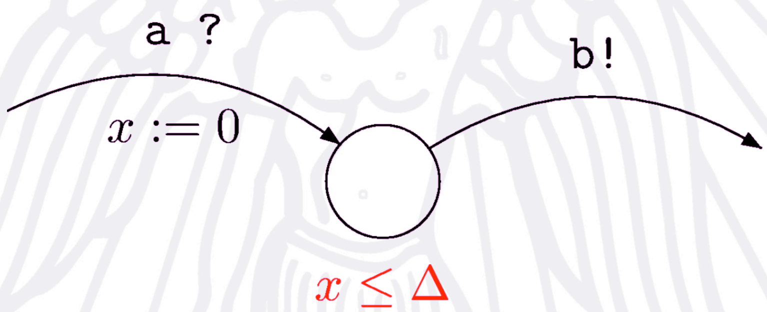
Intuition

- One can specify instantaneous response but not implement it.

Not implementable



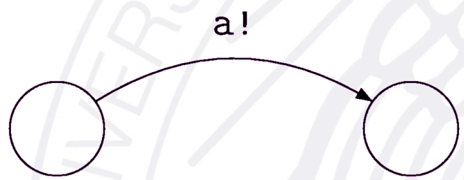
Solution : allow some delay



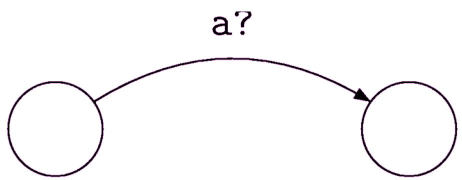
More Intuition

- Instantaneous synchronisation between environment and controller is not implementable.

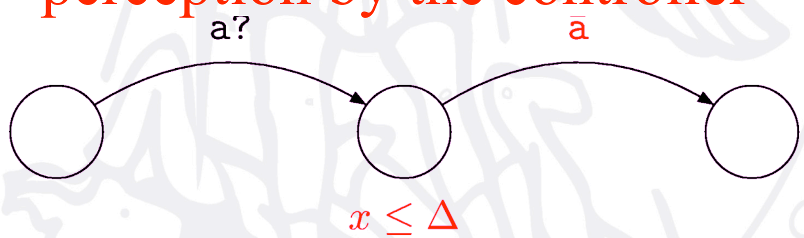
Environment



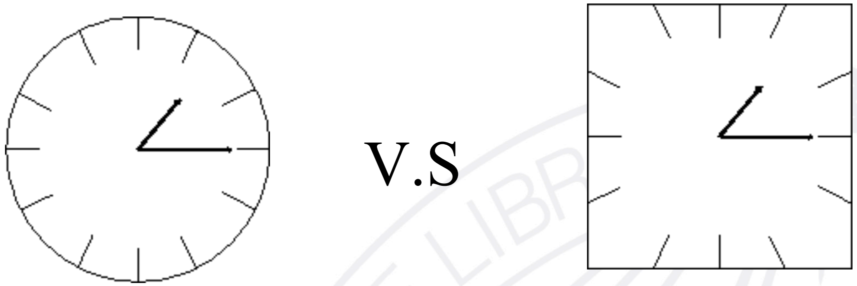
Classical controller
Not implementable



Solution :
Uncouple event from perception by the controller

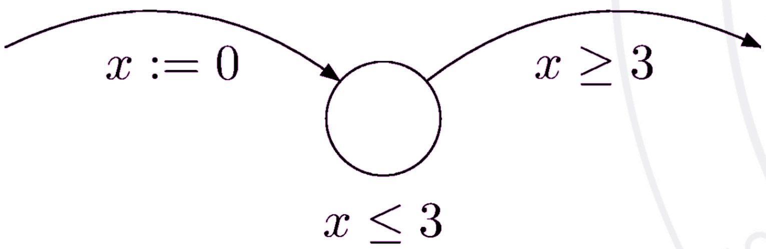


More Intuition

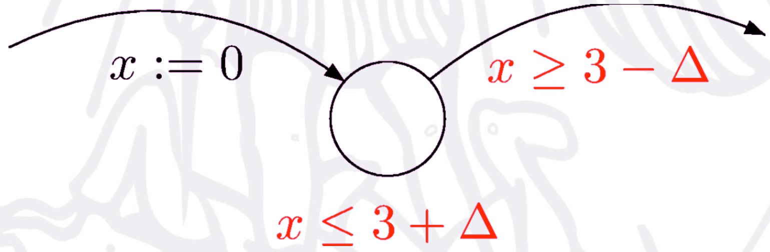


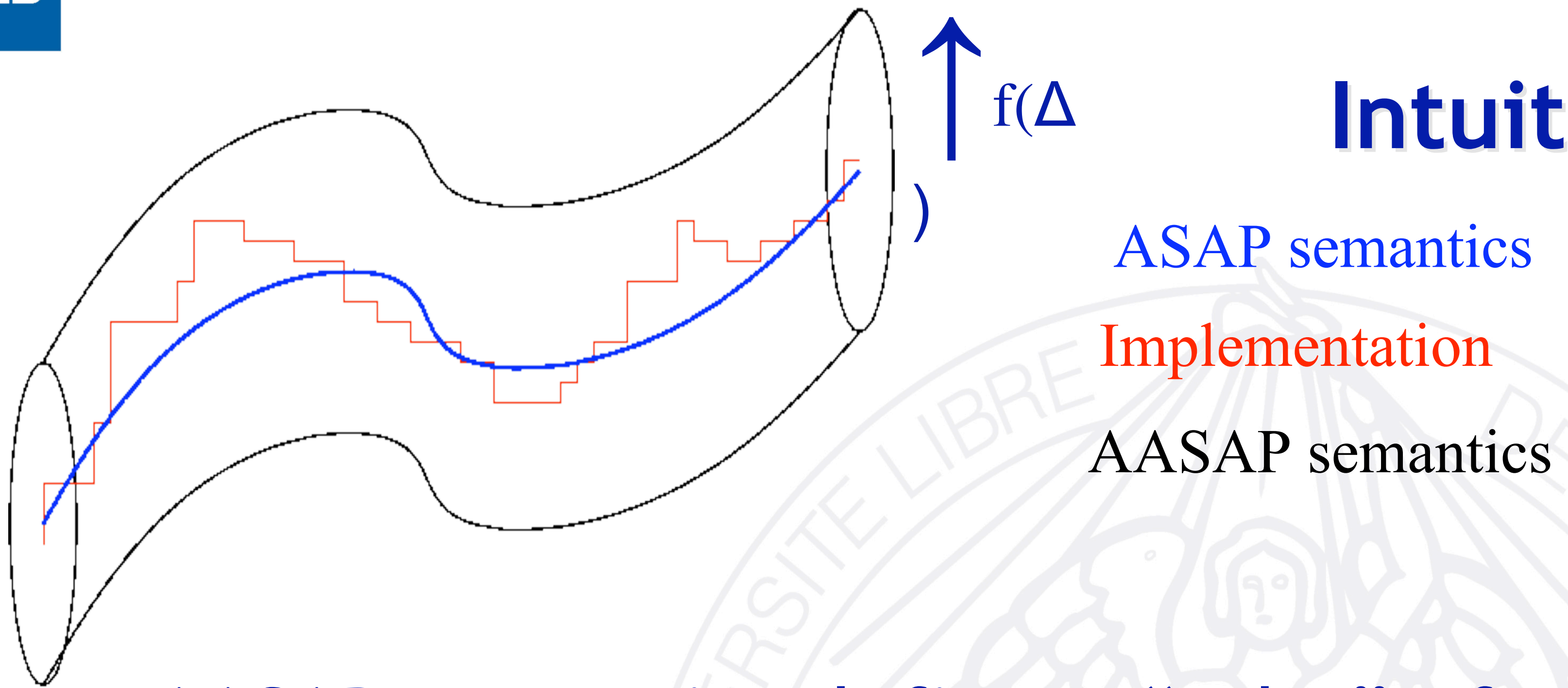
- Models use continuous clocks and implementation uses digital clocks with finite precision

Classical controller
Not implementable



Solution :
Slightly relax the constraints





Intuition

- AASAP semantics define a “tube” of strategies instead of a unique strategy in the ASAP semantics.
- This tube can be refined into an implementation while preserving safety properties

Verification

- The question that we ask when we make verification is no more:

  Does Env || ControllerMod avoid Bad ?

- But:

  for which values of Δ,

  does Env || ControllerMod(Δ) avoid Bad ?

Proof of "implementability" ?

- We define an "implementation semantics" based on:

Read System Clock
Update Sensor Values
Check all transitions and fire one if possible

- The timed behaviour of this scheme is determined by two values :
  - Time length of a loop : Δ_L
  - Time between two clock ticks : Δ_P

Proof of "implementability" ?

Theorem:

For any timed controller, its AASAP semantics simulates (in the formal sense) its implementation semantics, provided that :

$$\Delta > 2\Delta_L + 4\Delta_P$$

In this case, the implementation is guaranteed to preserve verified properties of the model, that is:

Environment || ControllerMod(Δ) avoid Bad

implies

Environment || ControllerImpl(Δ_L,Δ_P) avoid Bad

More Properties of the AASAP Semantics

- Faster is better !

For any Δ_1, Δ_2 such that $\Delta_1 > \Delta_2$:

If

Environment || ControllerMod(Δ_1) avoid Bad

then

Environment || ControllerMod(Δ_2) avoid Bad

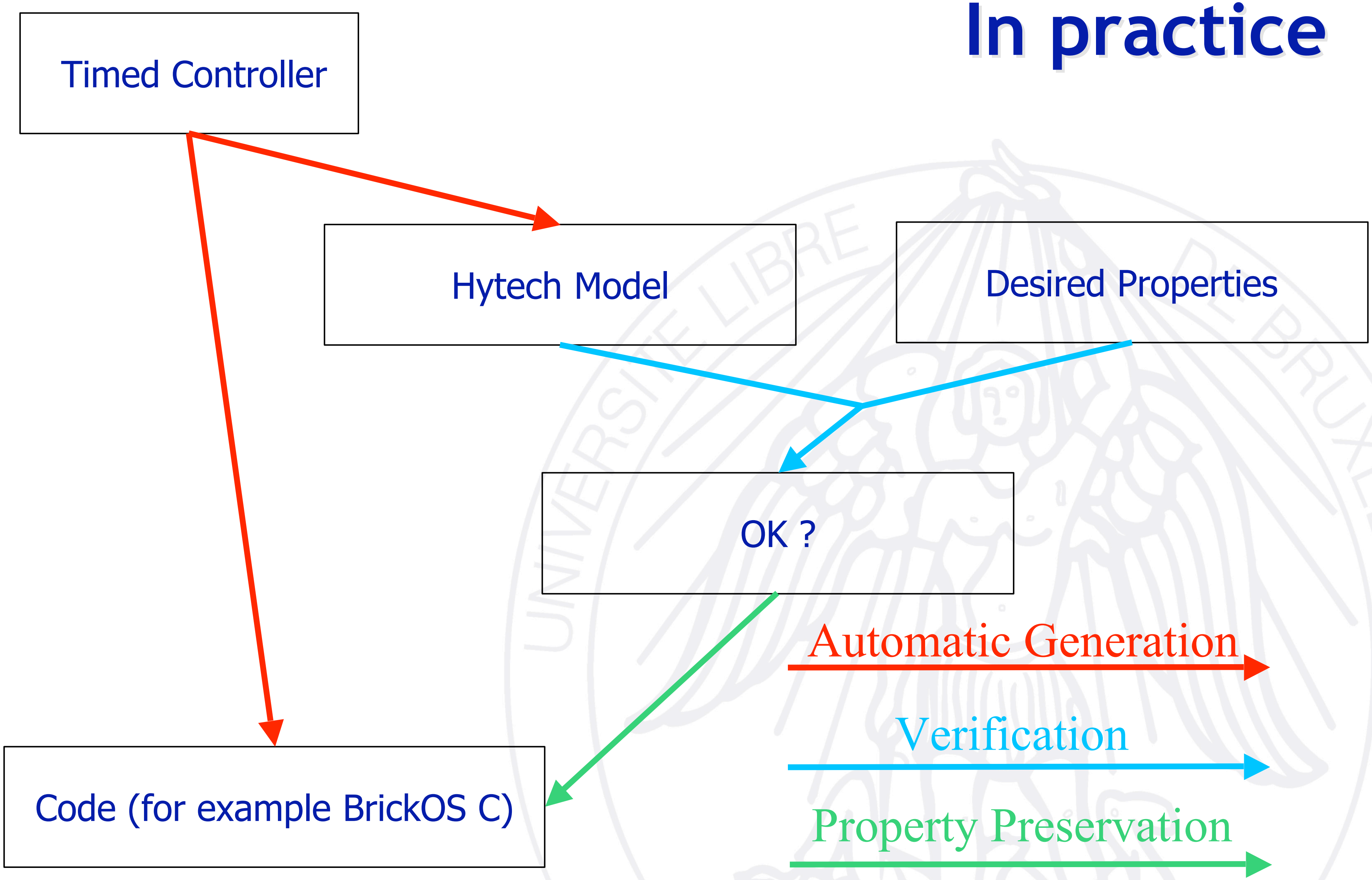
More Properties of the AASAP Semantics

- If $\Delta>0$, we get for free a proof that strategies:
  - are nonzeno
  - are such that transitions do not need to be taken faster and faster
- If only $\Delta=0$ guarantees some reachability property, then the control strategy is not implementable

In practice ?

- The AASAP semantics can be coded into a parametric timed automata with only one parameter $\Delta \in \mathbb{Q}$.
- Unfortunately, the reachability problem for that class of timed automata is undecidable… Direct corollary of [CHR02].
- Hytech implements a semi-decision procedure for that problem.

In practice



Timed Controller
Hytech Model
Desired Properties
OK ?
Code (for example BrickOS C)
Automatic Generation
Verification
Property Preservation

Conclusion

- Almost ASAP semantics is :
  - implementable
  - guarantees correct code and not only correct idealized model
  - maybe decidable (ongoing work).