Systematic Implementation of Real-Time Models

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

Université Libre de Bruxelles
Centre Fédéré en Vérification

Model-based Development for Controllers

- Make a model of the environment

  Environment
- Make clear the control objective:

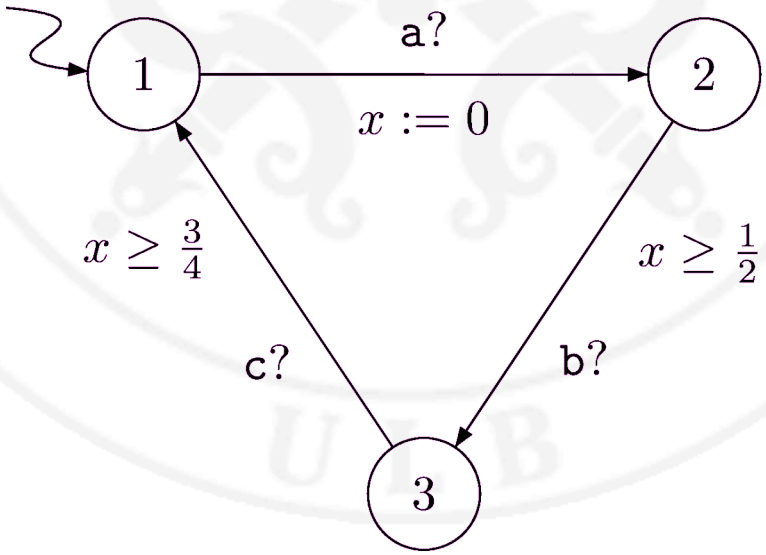
  Bad
- Make a model of your control strategy:

  ControllerMod
- Verify :

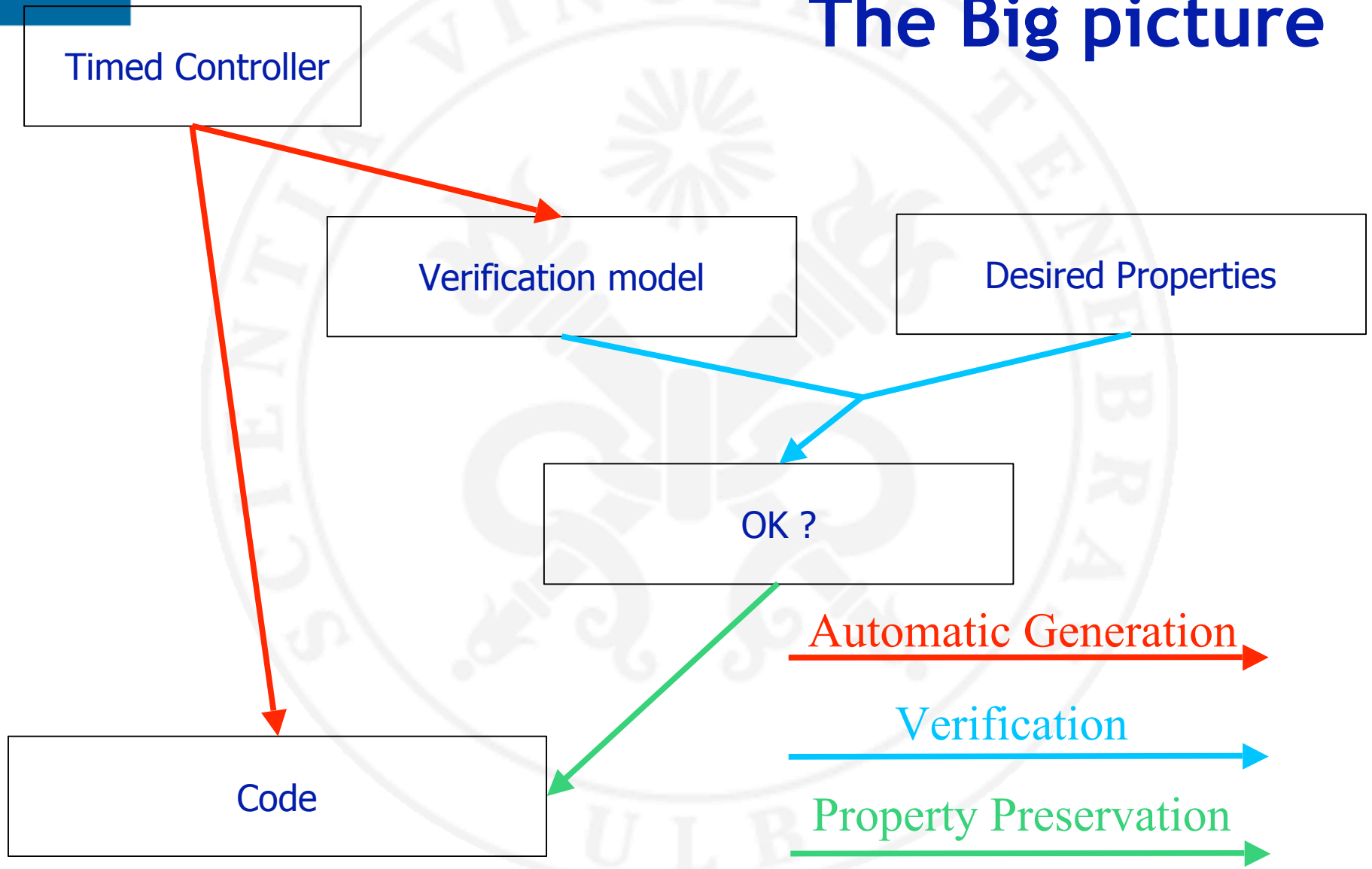
  Does Environment || ControllerMod avoid Bad?
- Good, but after ?

Goal

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



The Big picture



Outline of the talk

- Problems to implement T.A.
- AASAP semantics
- Verification in practice
- Case Study
- Code Generation
- Conclusion

Problem

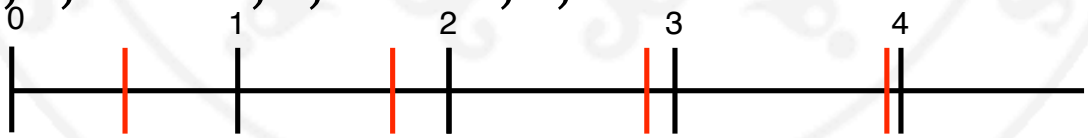
- Timed automata are (in general) not implementable (in a formal sense)...

  Why ?

  - Zenoness : 1/2, 3/4, 7/8, ...

  - No minimal bound between two transitions :

    1/2,1,1+3/4,2,2+7/8,3,...

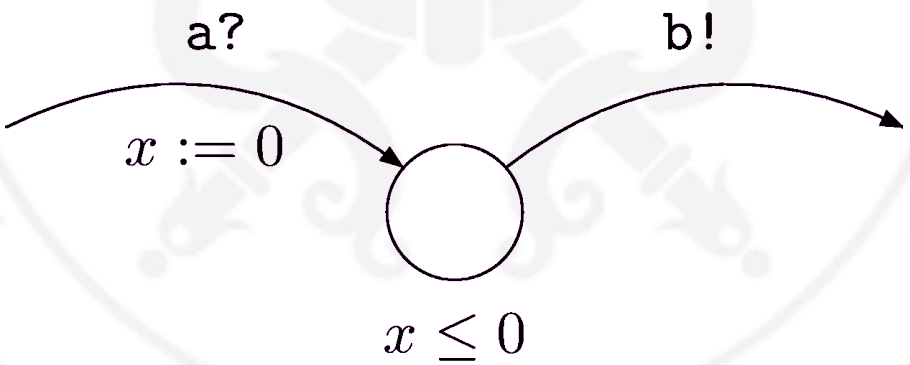


  - And more ...

More...

- One can specify instantaneous responses but not implement them.

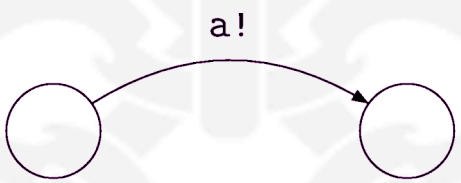
Not implementable



More...

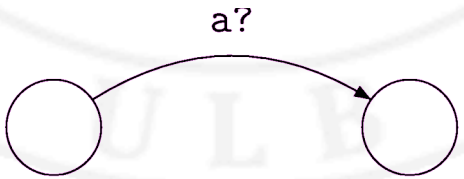
- Instantaneous synchronisations between environment and controller are not implementable.

Environment



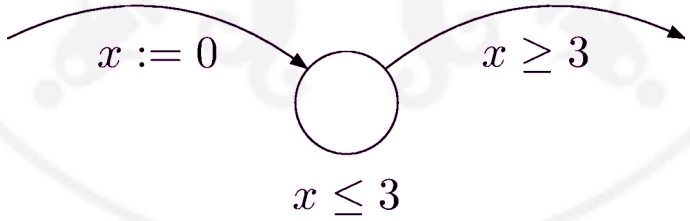
Not implementable

Controller



More...

- Models use continuous clocks and implementations use digital clocks with finite precision



Not implementable

Problems : Summary

- My controller stragegy may be correct because
  - … it is zeno…
  - … it acts faster and faster?
  - … it reacts instanteously to events, timeouts,…? (synchrony hypothesis)
  - … it uses infinitely precise clocks?

Outline of the talk

- Problems to implement T.A.
- AASAP semantics
- Verification in practice
- Case Study
- Code Generation
- Conclusion

A possible solution...

- Give an alternative semantics to timed automata :

  Almost ASAP semantics.

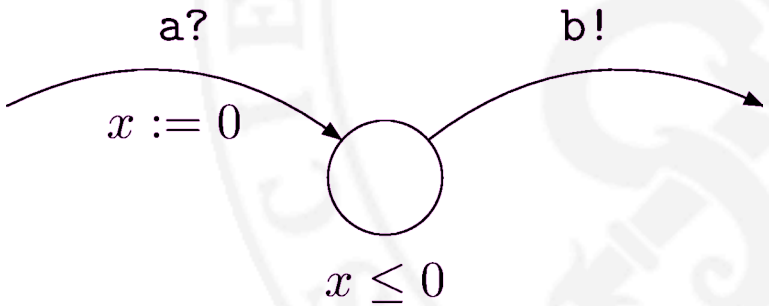
  Semantics parameterized by Δ in Q^+

  - 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;
  - truth values of guards are enlarged by $f(\Delta)$

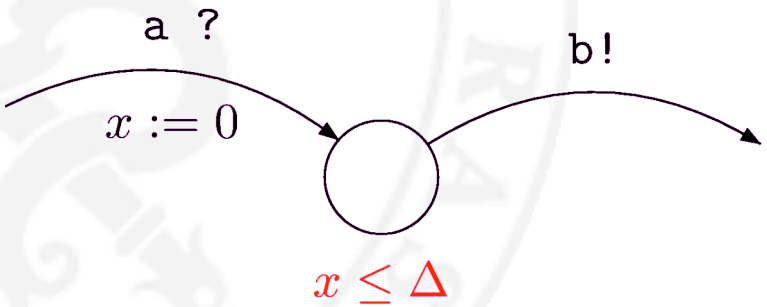
Intuition...

- One can specify instantaneous responses but not implement them.

Not implementable



Solution : allow some delay

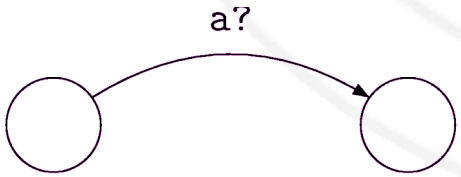


Intuition...

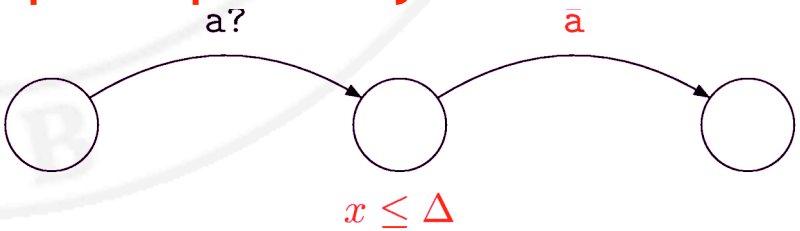
Instantaneous synchronizations between environment and controller are not implementable.

Environment

Not implementable

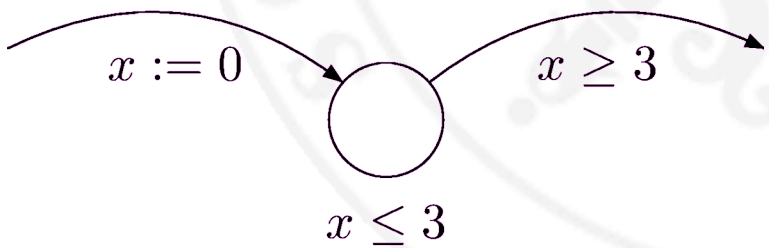


Solution :
Uncouple event from perception by the controller

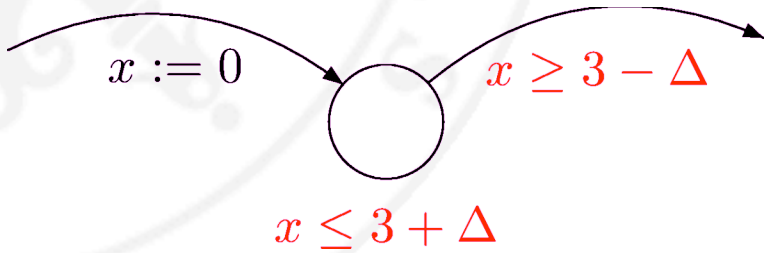


Intuition...

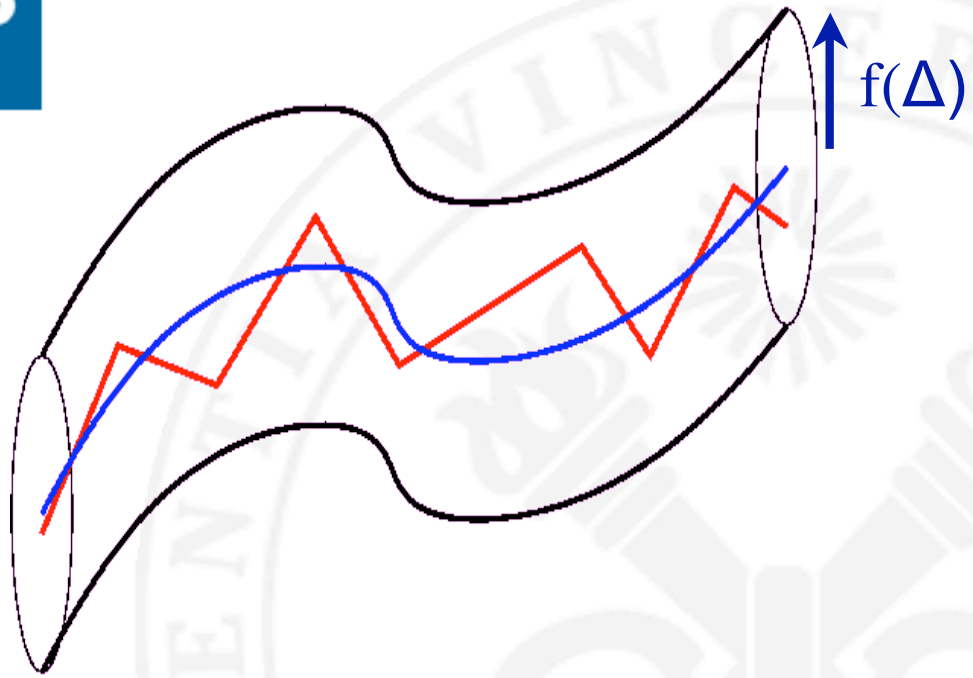
Models use continuous clocks and implementations use digital clocks with finite precision



Not implementable



Solution :
Slightly relax the constraints



Intuition

ASAP semantics

Implementation

AASAP semantics

- AASAP semantics defines 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 verified on the AASAP semantics

Outline of the talk

- Problems to implement T.A.
- AASAP semantics
- Verification in practice
- Case Study
- Code Generation
- Conclusion

Verification

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

  Does Environment || ControllerMod avoid Bad ?

- But:

  For which values of Δ

  does Environment || ControllerMod(Δ) avoid Bad ?

Verification in practice ?

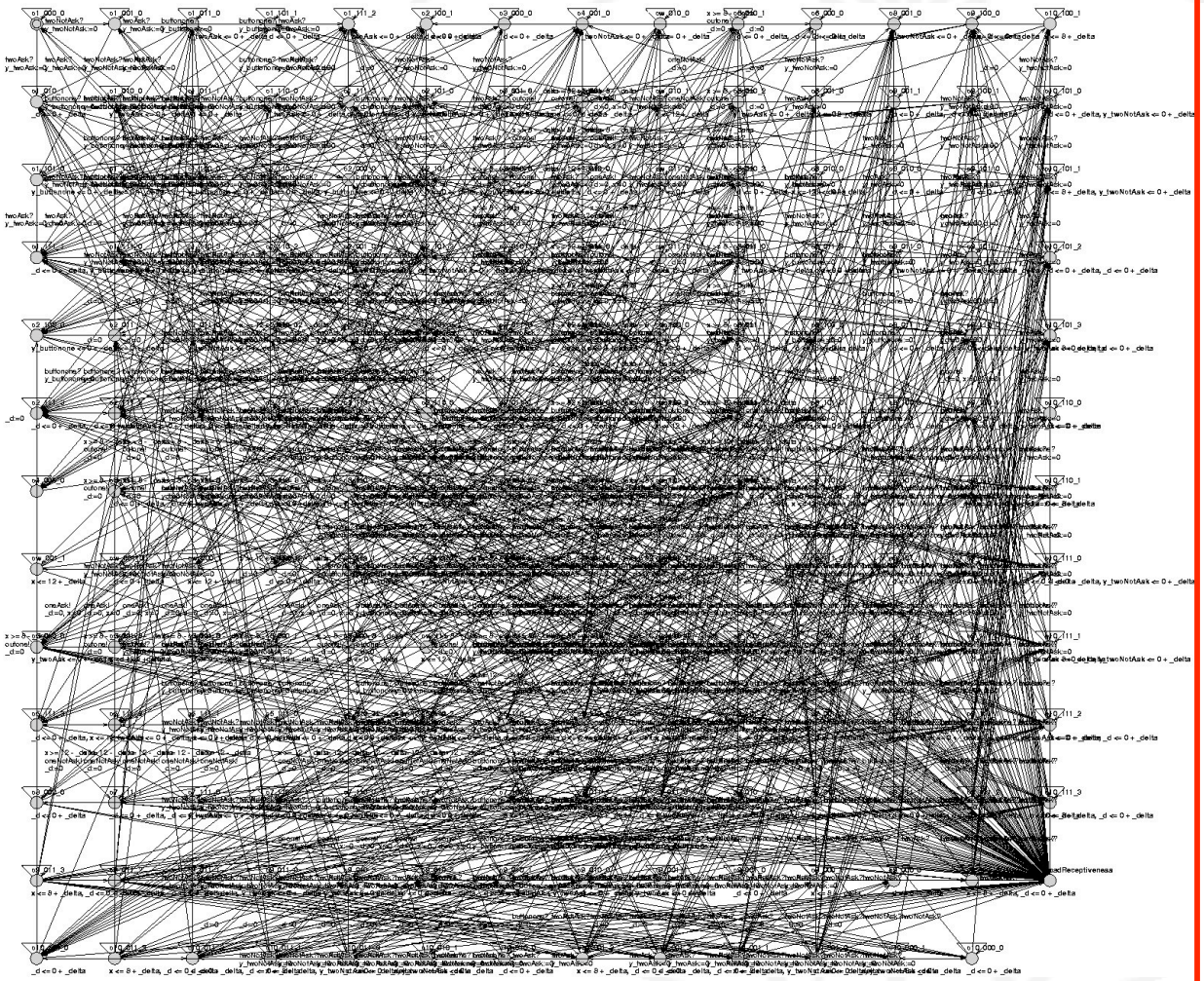
- The AASAP semantics can be coded into a parametric timed automata with only one clock compared to the parameter Δ in Q^+ ...
- ... But the initial coding we proposed in [DDR04] multiplied the number of locations by $2^{|\text{input labels}|}$!
- So, how to allow verification of large examples ?

Compositional Construction

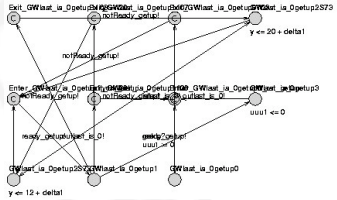
- A model-checker like Uppaal constructs state space on the fly
- So, compositional construction : parallel composition of small automatas (called widgets) to encode AASAP semantics
- Number of locations polynomial in the size of the original model

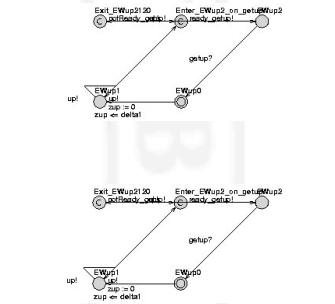
ULB

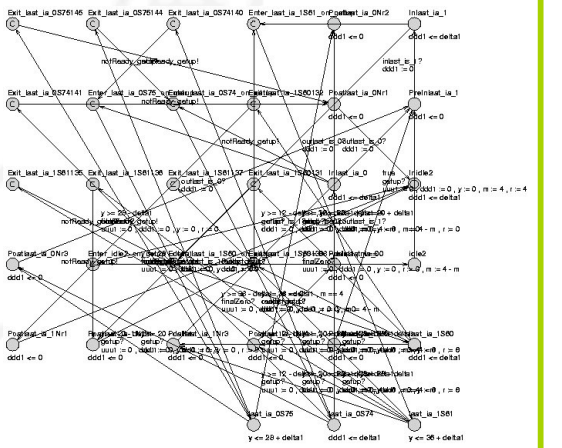
Non-Compositional



Compositional







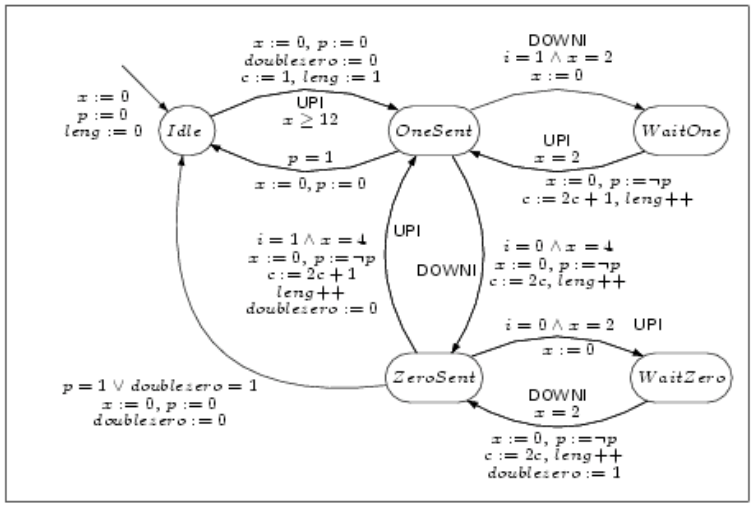
Outline

- Problems to implement T.A.
- AASAP semantics
- Verification in practice
- Case Study
- Code Generation
- Conclusion

A case study

The Philips Audio Control Protocol

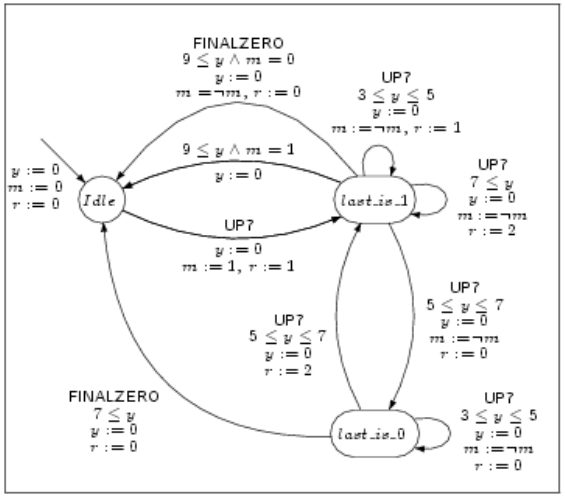
Sender

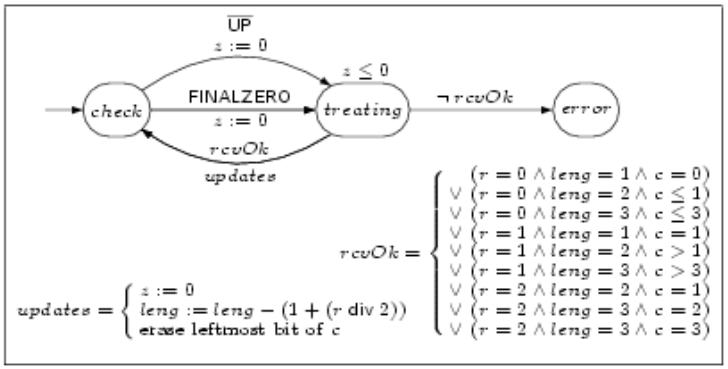


10011100

Timed Manchester encoding

Receiver





Observer

Properties /requirements for the protocol

- the receiver knows the length of a time slot but ignores when it begins;
- the receiver ignores length of the current bit string;
- only UP signals can be perceived reliably;
- S/R uses (unsync.) digital clocks : there will be imprecision in sending and perceived receiving times;
- Sensors are polled every time slice : discrepancy between occurence of UP events and detection.

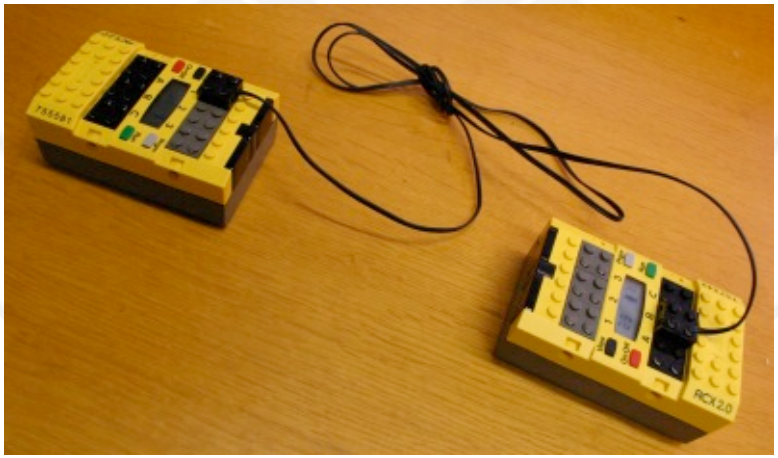
... 3 first items should be dealt with by the logic of the protocol, 2 last items are related to robustness of the protocol : the AASAP-semantics deals with it.

Application of the methodology

- Model (idealized) sender and receiver using the synchrony hypothesis
- Check for robustness :

  for which Δ :

  does Sender(Δ) || Receiver(Δ) || Observer avoid Bad ?
- Generate correct code for RCX



Outline of the talk

- Problems to implement T.A.
- AASAP semantics
- Verification in practice
- Case Study
- Code Generation
- Conclusion

Code Generation

- We simply annotate the transitions of the model with simple C code instructions
  - For example :
    -in the model of the sender, the next bit to be sent is choosen non-deterministically :
    r=S[i]; i++;
    -in the model of the receiver, on a transition decoding a one :
    i++; R[i]=0;

Outline

- Problems to implement T.A.
- AASAP semantics
- Verification in practice
- Case Study
- Code Generation
- Conclusion

Conclusion

- Almost ASAP semantics :
  - is implementable!
  - is verifiable, even for non-trivial case studies!
  - guarantees correct code and not only correct idealized model !
  - is tool-supported !

Further Informations

- [DDR04] M. De Wulf, L. Doyen, J.-F. Raskin. Almost ASAP Semantics: From Timed Model to Timed Implementation. LNCS 2993, HSCC 2004.
- Journal Version to appear in Formal Aspects of Computing
- http://www.ulb.ac.be/di/ssd/madewulf/aasap/

Thank You

Proof of "implementability" ?

[DDR04]

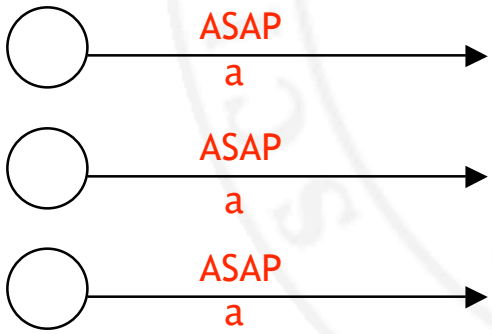
- We define an "implementation semantics" based on:

  Read System Clock
  Read 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
- We prove that this semantics is simulated (in the formal sense) by the AASAP-semantics if $3\ D_L + 4\ D_P < D$

Compositional Construction

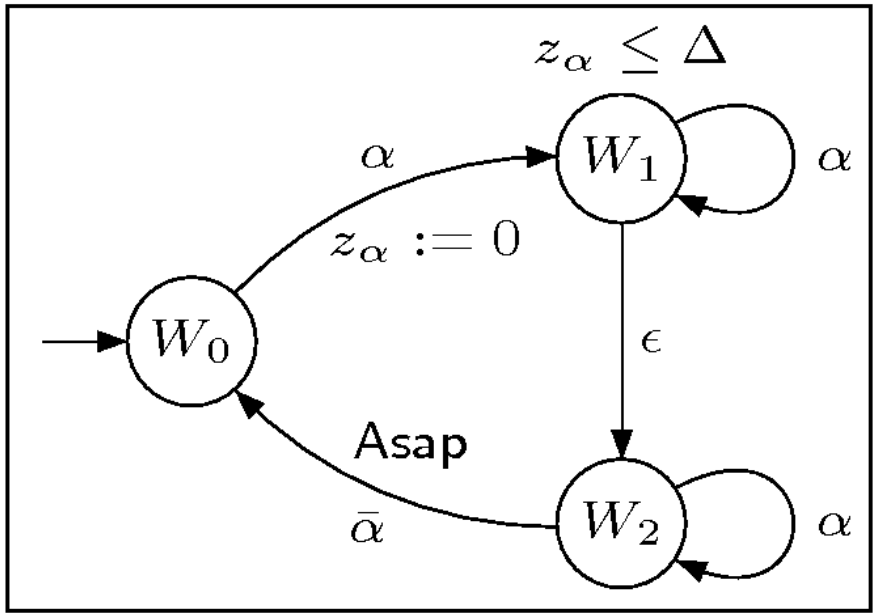
To specify urgency compositionally : use of the ASAP flag on transitions



Only urgent if the 3 automata are in a location with an ASAP transition

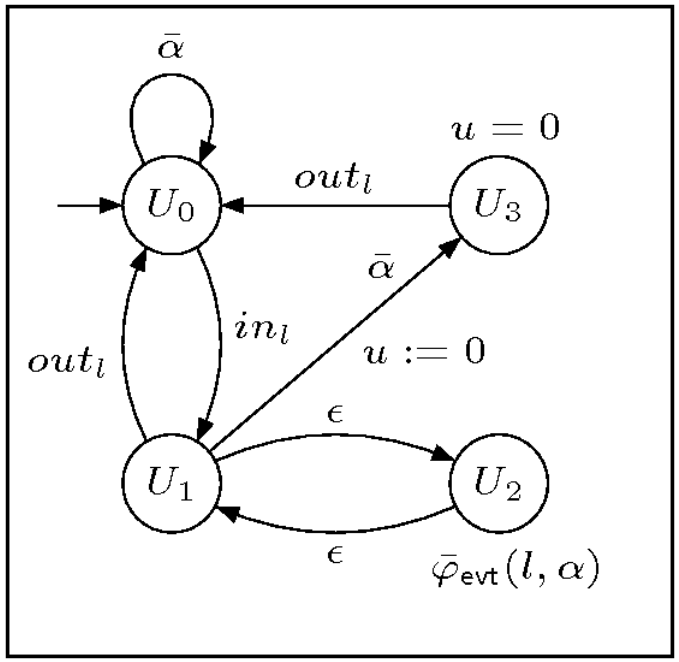
Widget 1 : Event-Watcher

- Record event a;
- Wait at most D unit of time …
- … before making the viewing of a urgent.



Widget 2 : Guard-Watcher

- Makes enabled transition in location l urgent D units of time after they became enabled.



If a transition is enabled when x> 3
It becomes urgent when x> 3 + D