Integration of Time Issues into Component-Based Applications

Saudrais Sébastien – Olivier Barais
–Noël Plouzeau

Irisa project Triskell

Goals

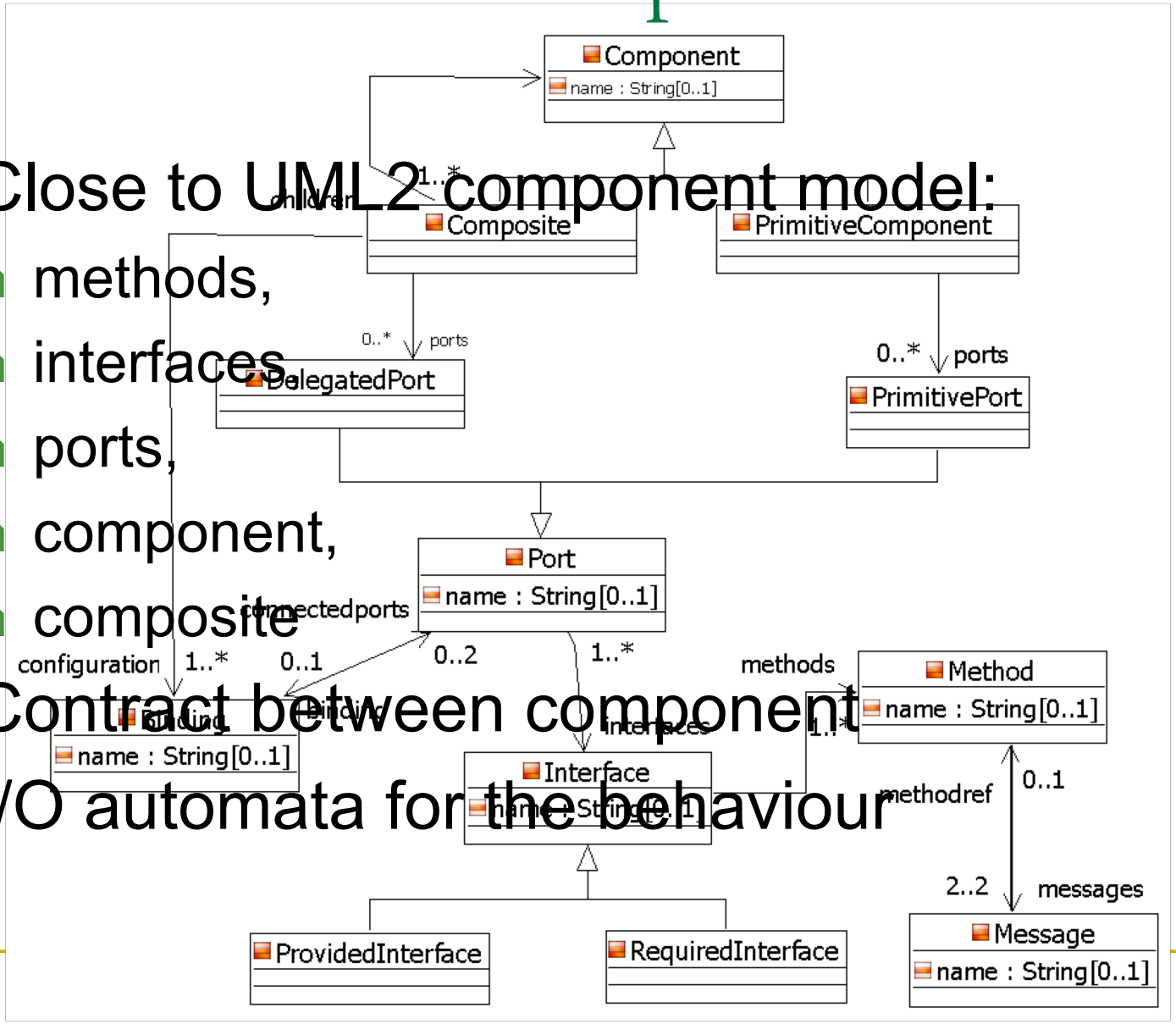
- Add time into components
  - QoS
  - Response time
- Keep functionalities
- Separation of concerns
  - Time vs functionalities
- Composition validation

Agenda

- Meta-model of components
  - Behaviour
  - Contract
- Add Time
  - In behaviour
  - In contracts
  - Composition
- Future works

Meta-model of components

- Close to UML2 component model:
  - methods,
  - interfaces,
  - ports,
  - component,
  - composite
- Contract between component
- I/O automata for the behaviour



Behaviour

- Process Algebra
  - Send-receive
  - A send (receipt) must have an acknowledge
- I/O automata for the verification

Contract

- Ports compatibility
- 4 levels of contract [Beugnard99]:
  - Syntactic (IDL)
  - Behavioural (pre-post conditions)
  - Synchronisation (services dependencies)
  - QoS
- More and more negotiable

Agenda

- Meta-model of components
  - Behaviour
  - Contract
- Add Time
  - In behaviour
  - In contracts
  - Composition
- Future works

Add time

- Where to add :
  - Behaviour (what is provided by the component)
  - Contract (what is required by the component)
- Validation of assembly
  - Validation of existing contracts
  - Check the QoS contracts

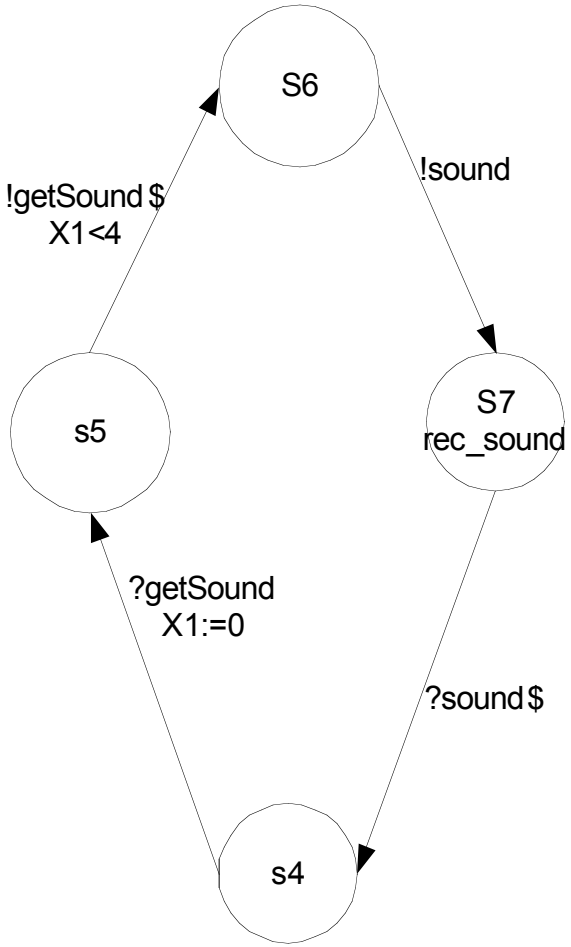
Add time into behaviour

- Formalism :
  - With time
  - Close to I/O automata
- Timed Automata
  - Automata with clocks
  - Transition with timed guard
- Definition of pattern
  - Execution time, delay, period

Timed automata

- A timed automaton is defined by:
  - S :set of locality
  - L : labels
  - X : set of clocks
  - T : transition relation
    - $T \subseteq SxLx2^{C}x\ h(C)xS$
    - 2^{C} : set of clocks to initialise
    - h(C) : clocks constraints
  - P : set of properties into localities

Timed Automaton



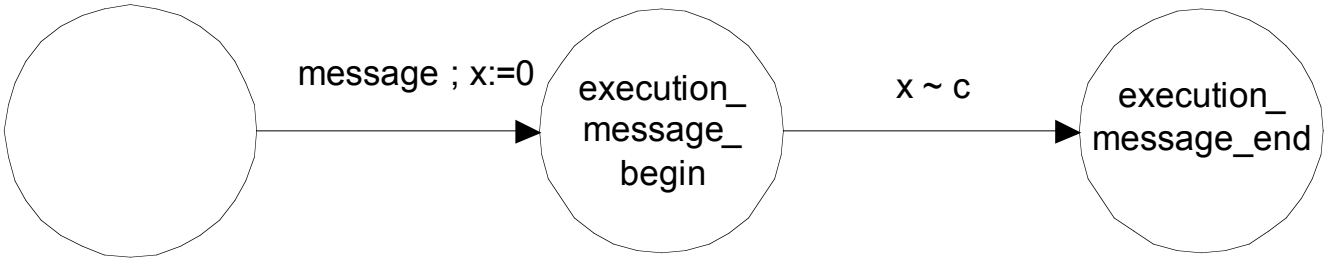
TA1

Time pattern

- A time property has:
  - A service call, message
  - A guard
  - Properties in locality
  - How to apply it
- TA with parameters

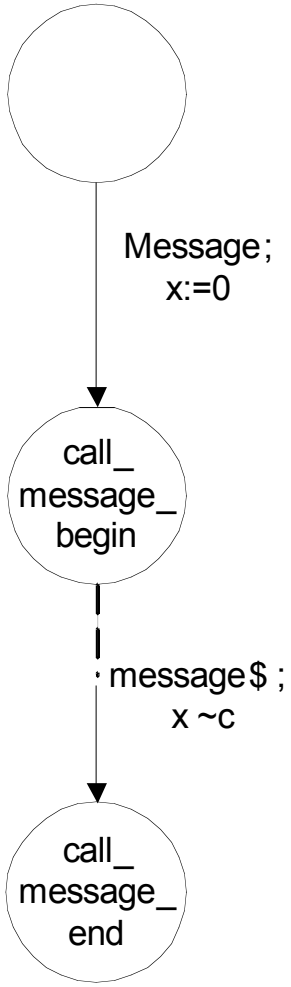
Execution time pattern

- Execution time after a message

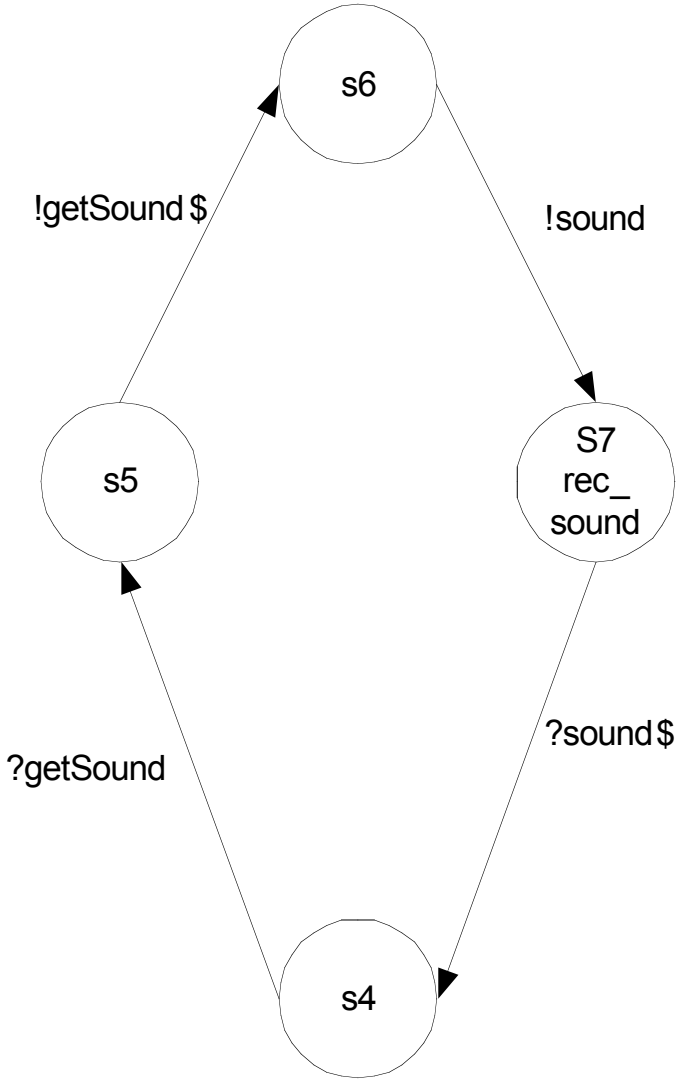


ET

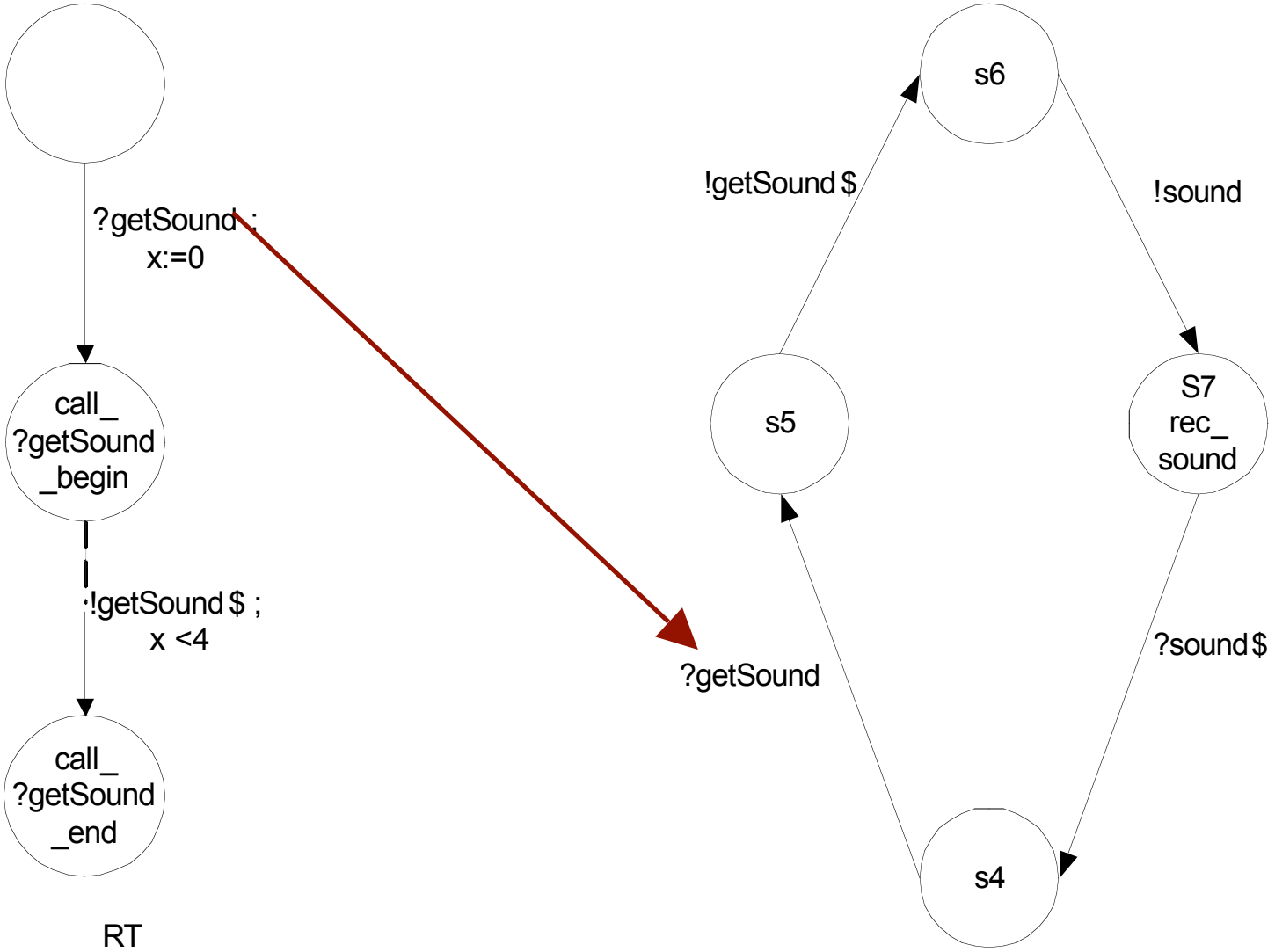
Applying pattern (1): response time



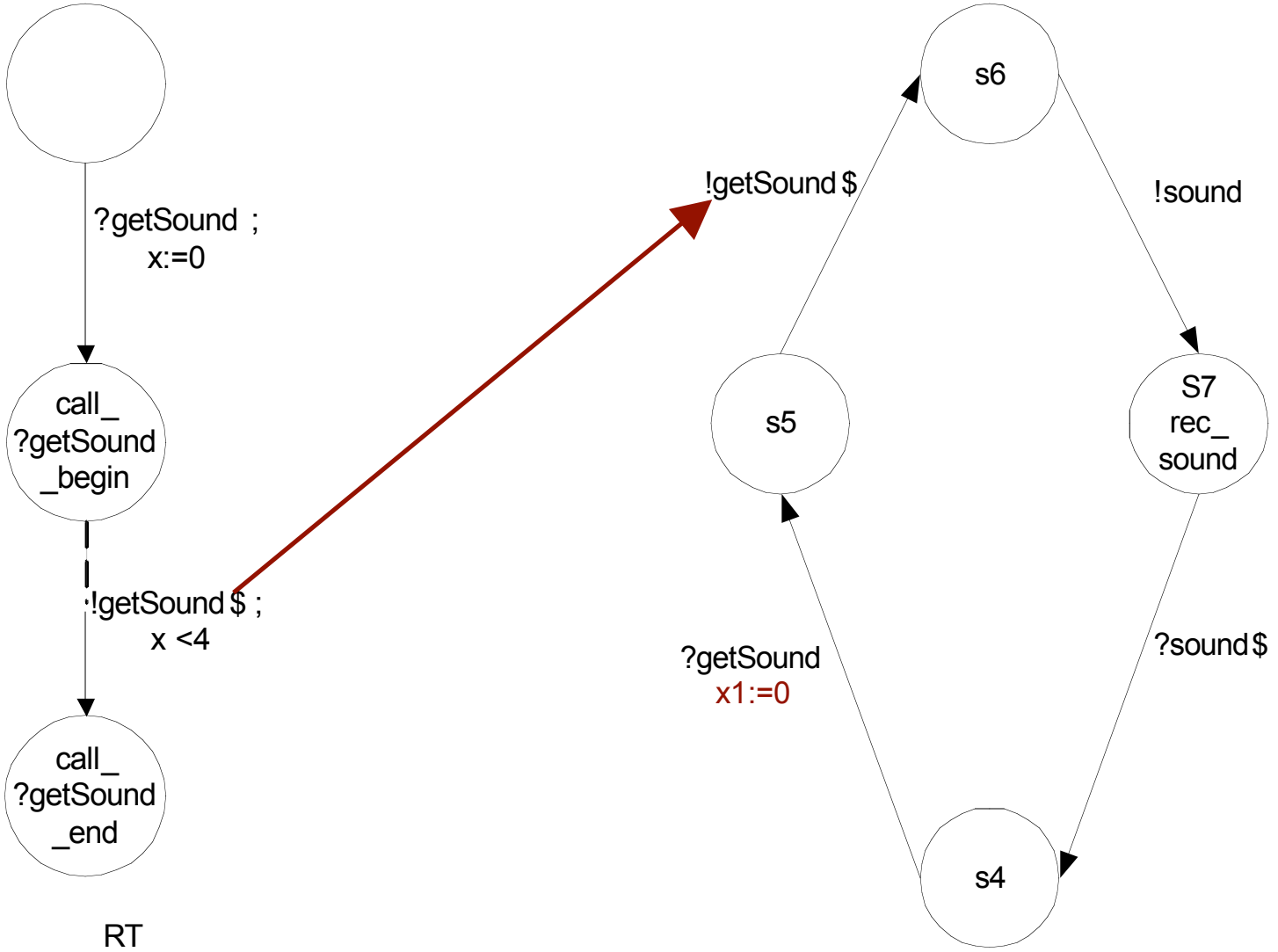
RT



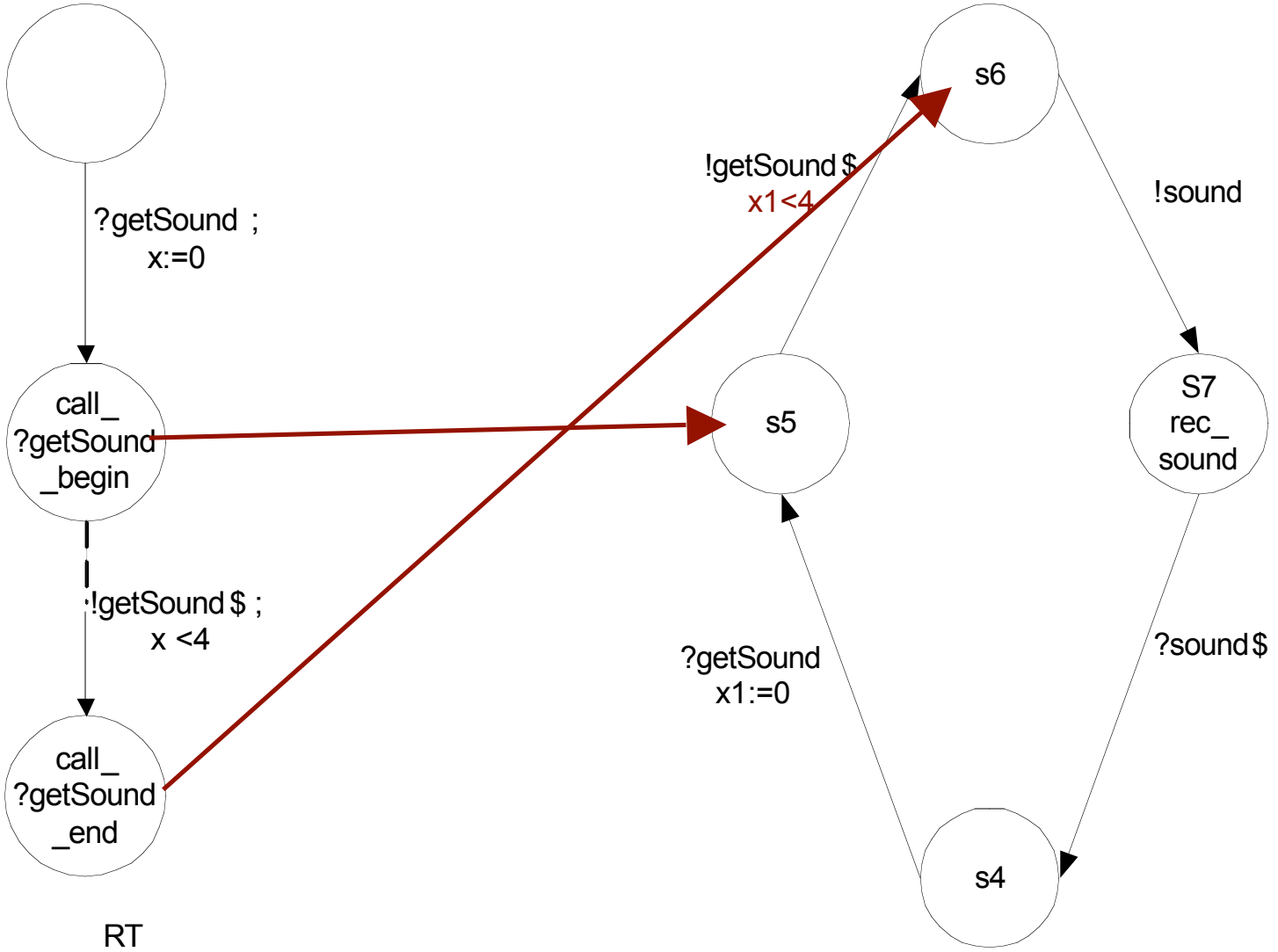
Applying pattern (1): response time



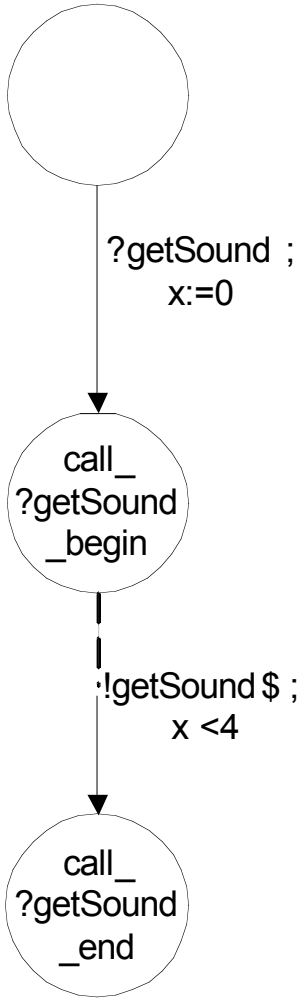
Applying pattern (1): response time



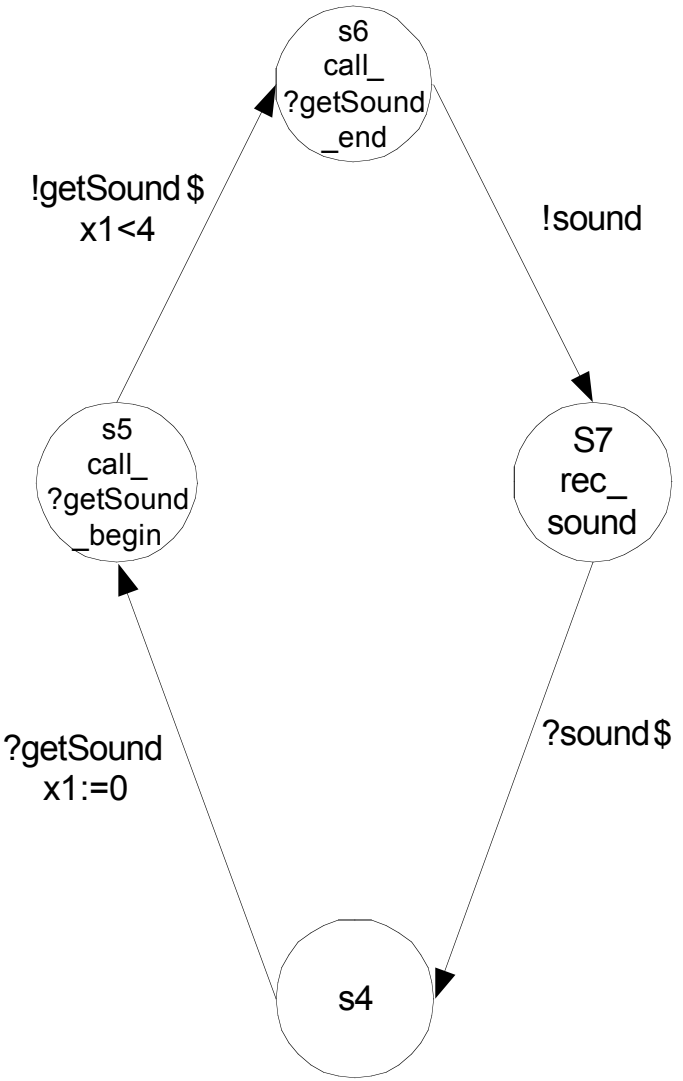
Applying pattern (1): response time



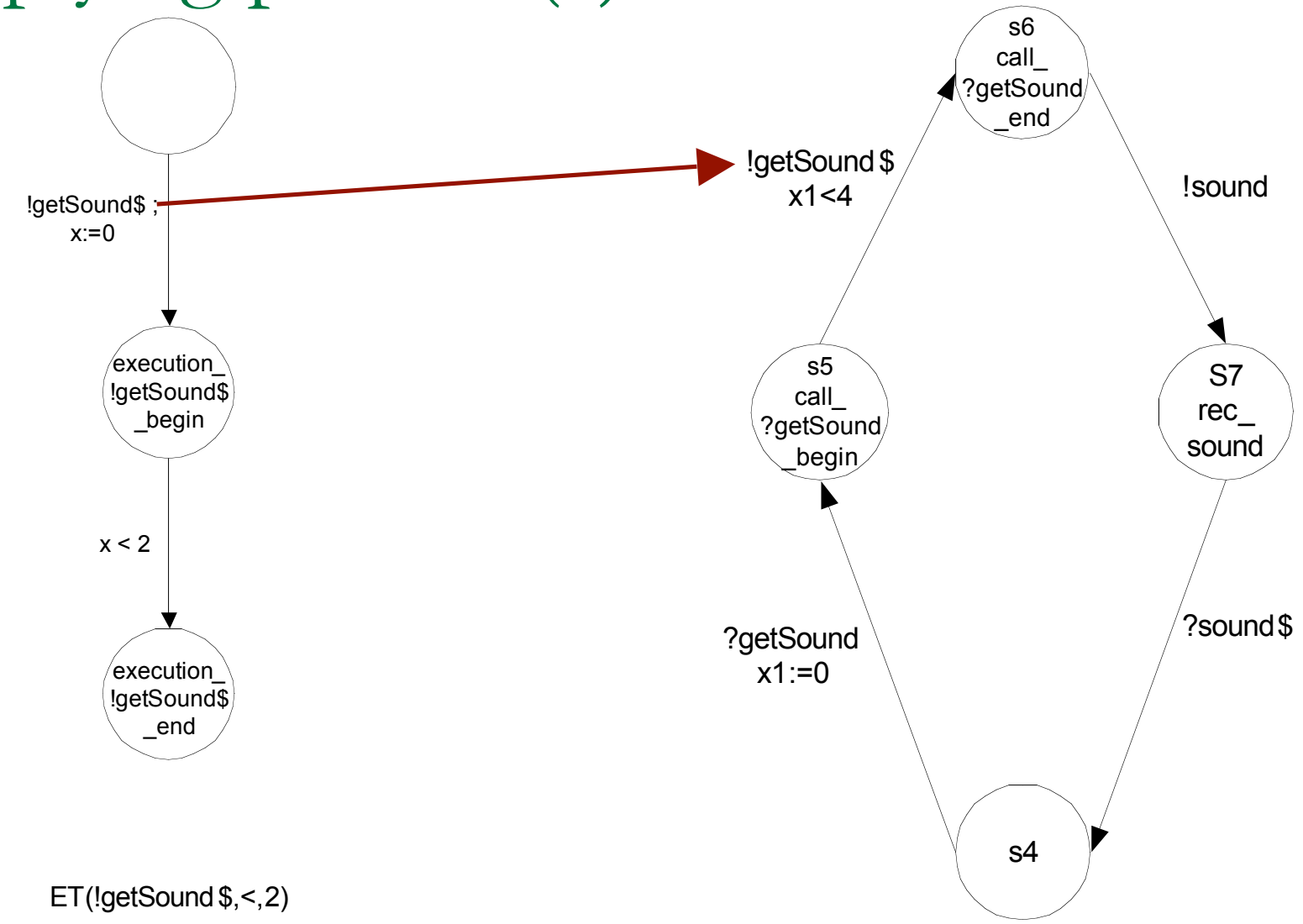
Applying pattern (1): response time



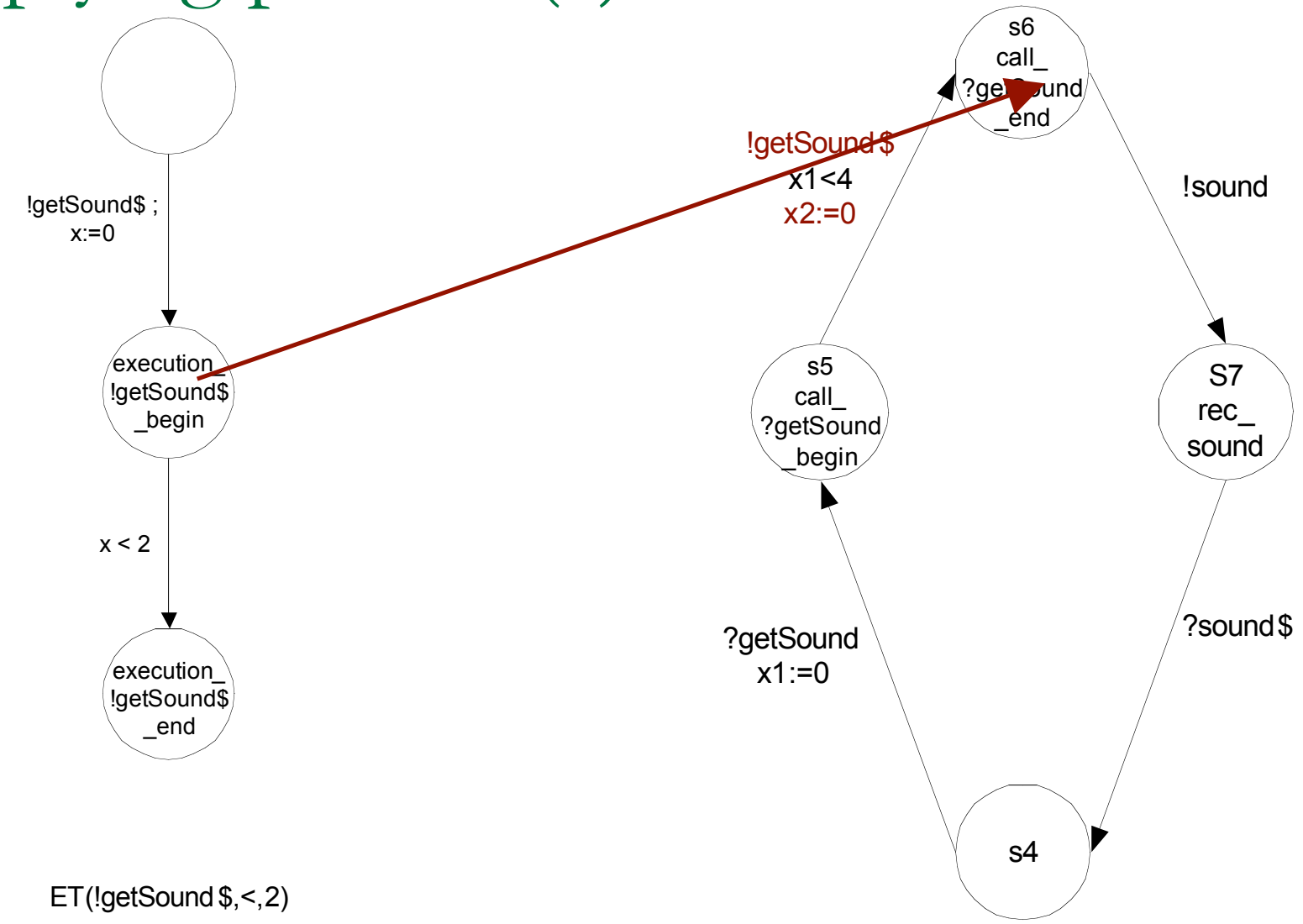
RT



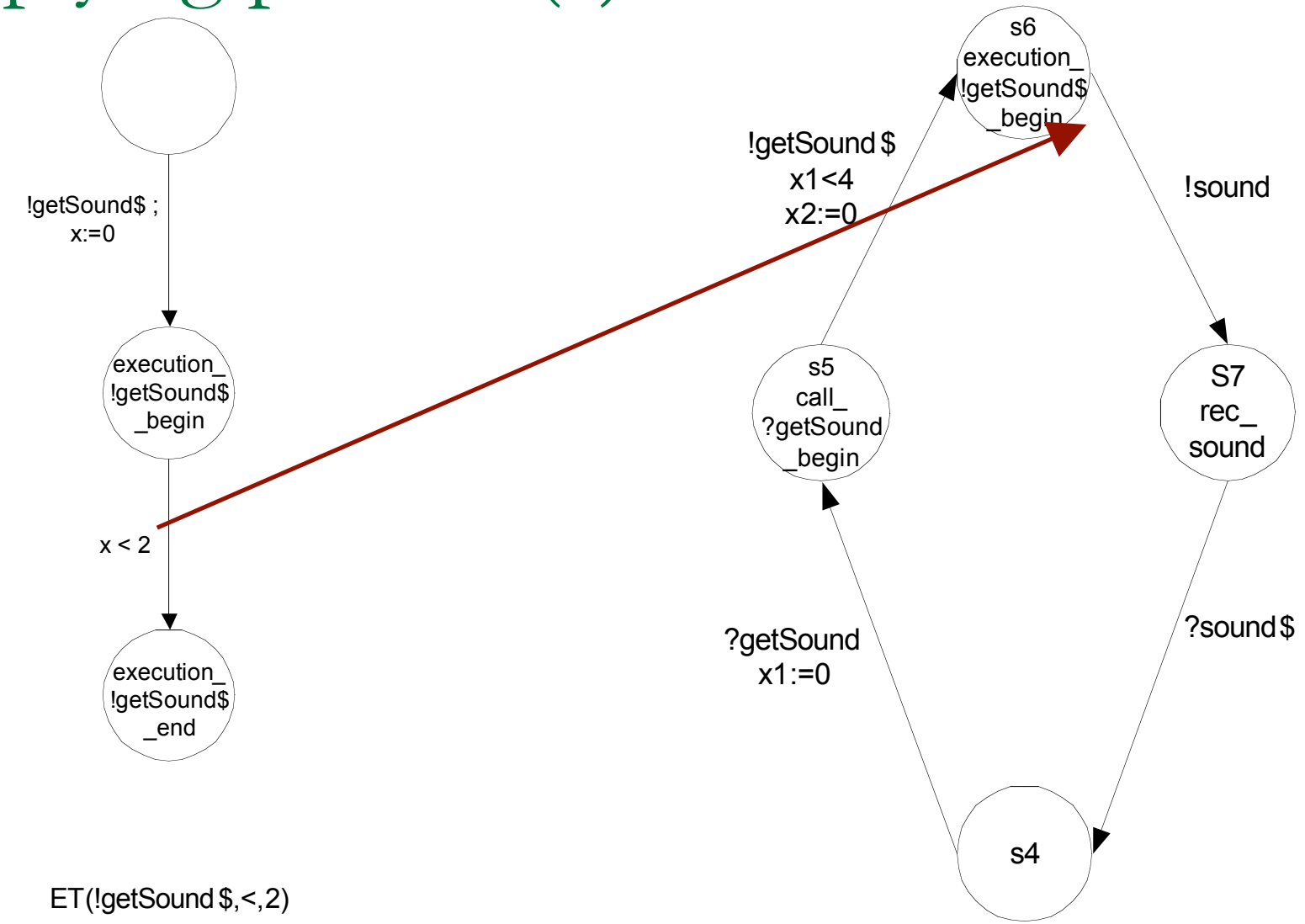
Applying pattern (2): execution time



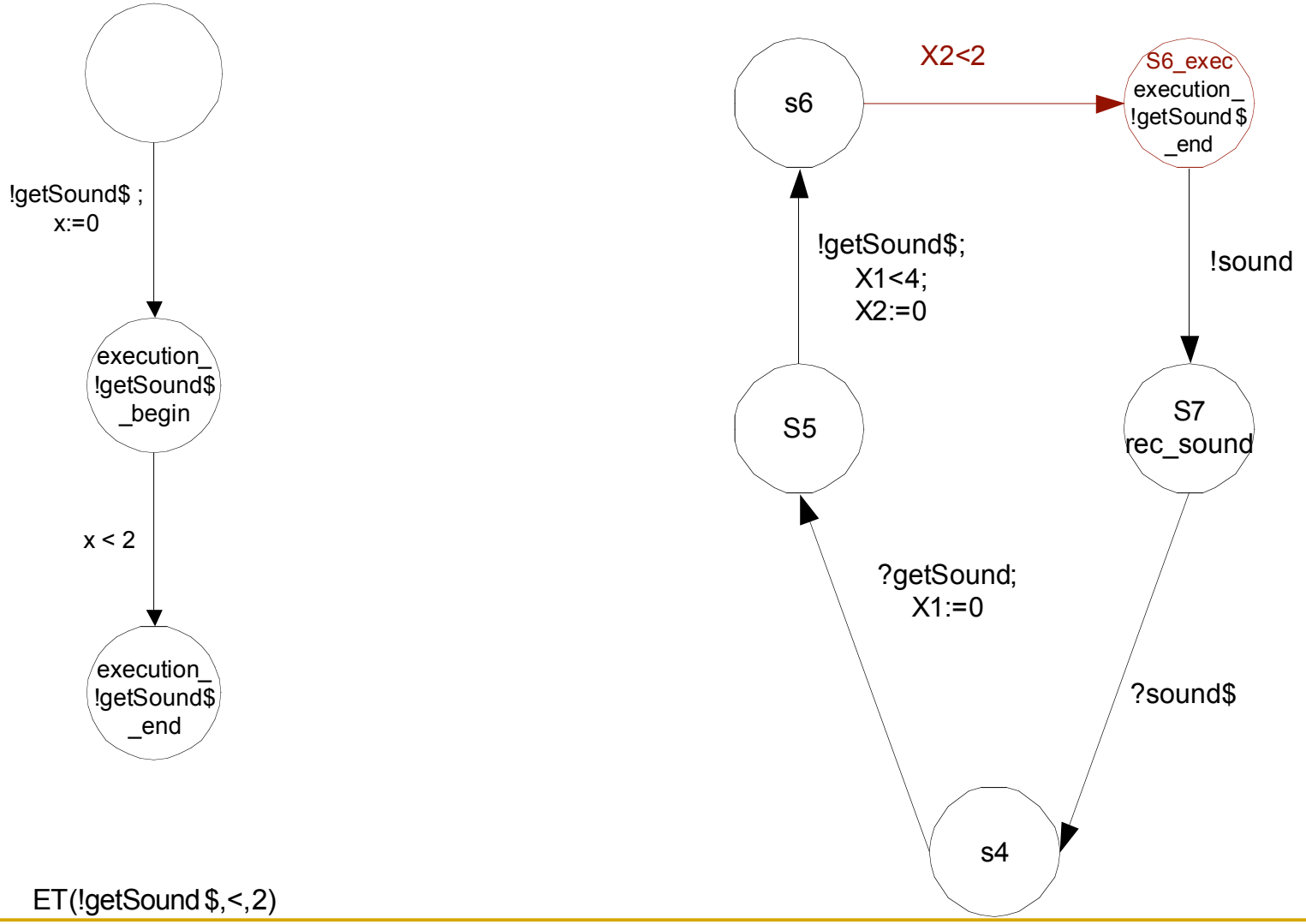
Applying pattern (2): execution time



Applying pattern (2): execution time

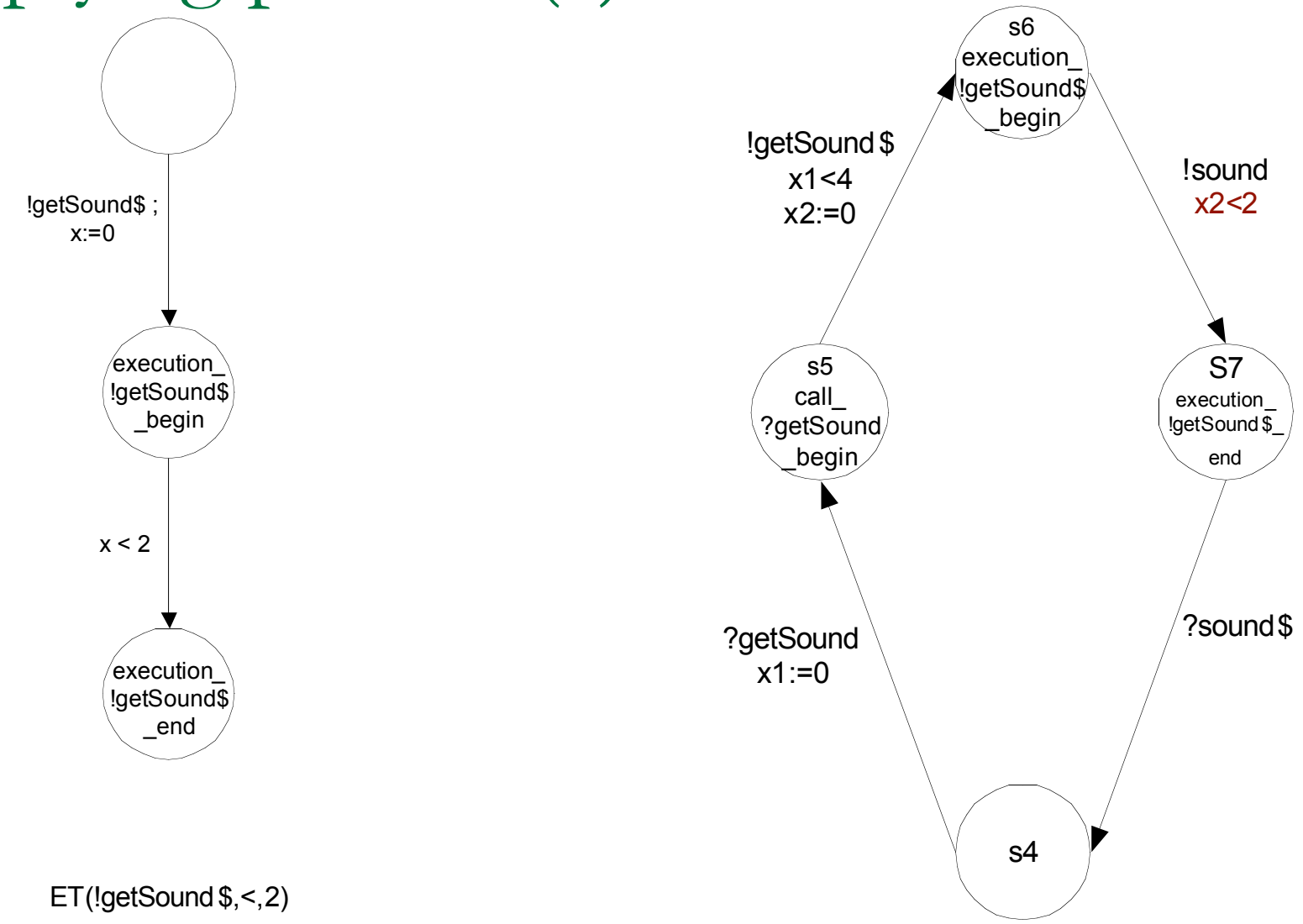


Applying pattern (2): execution time



ET(!getSound$,<,2)

Applying pattern (2): execution time



Add time into contracts

- 4th level contract
  - Attached to a required interface
- Timed temporal logic :TCTL
  - CTL with time quantifier
  - $\exists\Diamond p$ with time : $\exists\Diamond_{<5} p$
- Use patterns
  - Period of c of the property p : $\forall\Box(\forall\Diamond_{\sim c} p)$
  - c units of time between 2 properties : $p1 \rightarrow \forall\Box(\forall\Diamond_{\sim c} p2)$

Assembly Verification

- The component behaviour must satisfy the contracts
- Level 4 : TA against a TCTL formula
- Use of model-checker Kronos

Conclusions et Future works

- Time properties into components
- Separation of concerns
- Full implemented in Kermeta and Sintaks : Meta-Model, Operations and contracts

- Future works:
  - Time as an aspect (Need of a pointcut language)
  - Introduce time into other formalisms