Verification of timed UML models

Susanne GRAF

Iulian OBER, Ileana OBER

VERIMAG

www-verimag.imag.fr/~ober/IFx

- the problem
- semantics of objects with automata
- verifying objects with observers
- time dependent properties
- toolset

the problem

Model-based verification in UML

Which kind of verification?

- – model debugging – simulation
- – checking correctness properties – model-checking

which tool ?

Design choice: reuse existing state-of-the-art automata-based validation tools

- IF (http://www-verimag.imag.fr/~async/IF/)

Semantics of UML with time in terms of automata

Provide a means to express properties in UML

Verification of properties: use existing tools

which UML ?

Which language constructs?

- UML 1.4 – the operational part (true OO models, not just state-charts)
  - classes with operations, attributes, associations, generalization, state-charts; basic data types
- defining an action language (compat. to UML1.4 A.S.)
- fixing a semantics for communication & concurrency
  - active/passive objects, activity groups, run-to-completion
  - interactions: primitive/triggered operations, asynchronous signals

Which real-time ?

- a profile supporting imperative and declarative (constraint-based) specification of timing

Expressing requirements (properties) ?

- constraints – invariants (time related)
- observer objects (a lightweight UML extension)

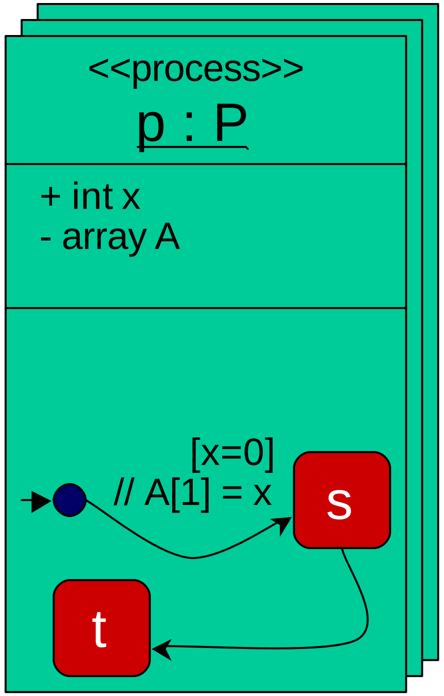
semantics in terms of automata

Why automata ?

– existing model-checking techniques

Which automata ?

– communicating extended timed automata : IF



- processes
  - agents running in parallel
  - own data
  - behavior described by state machine + actions

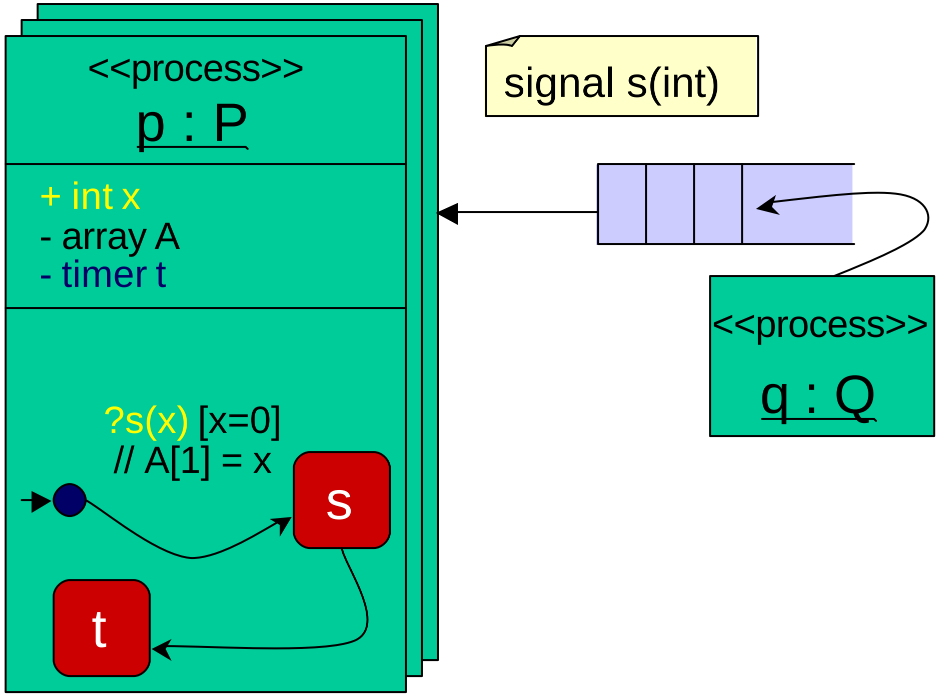
semantics in terms of automata

Why automata ?

- existing model checking techniques

Which automata ?

- communicating extended timed automata : IF



- parallel composition
  - asynchronous (interleaving)
- communication
  - asynchronous via buffers
  - shared memory
- dynamic priorities

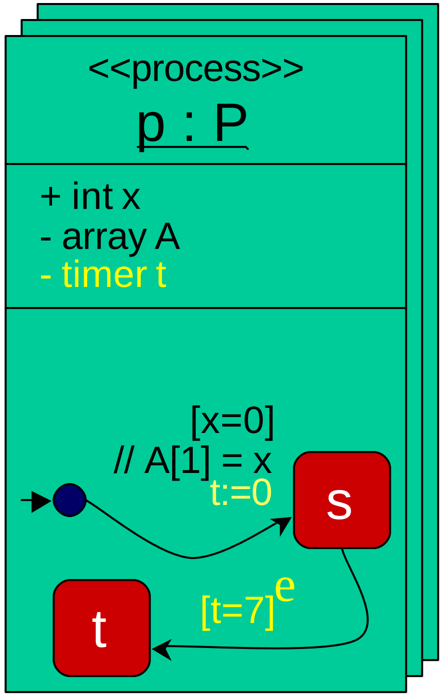
semantics in terms of automata

Why automata ?

– existing model-checking techniques

Which automata ?

– communicating extended timed automata : IF



- time model: timed automata with urgency
  - time passes in states and transitions are events
  - clocks measure duration and can be set and tested
  - urgency determines when transitions must be taken

representing objects

- structure
  - UML class → IF process
  - attributes & associations → variables
  - inheritance : replication of structural features
- behavior
  - state machines, actions → syntactic translation (almost)
  - operation calls X::m(x,y,…)

⇒ one IF process for every invocation of X::m

process X::m(x, y, …)

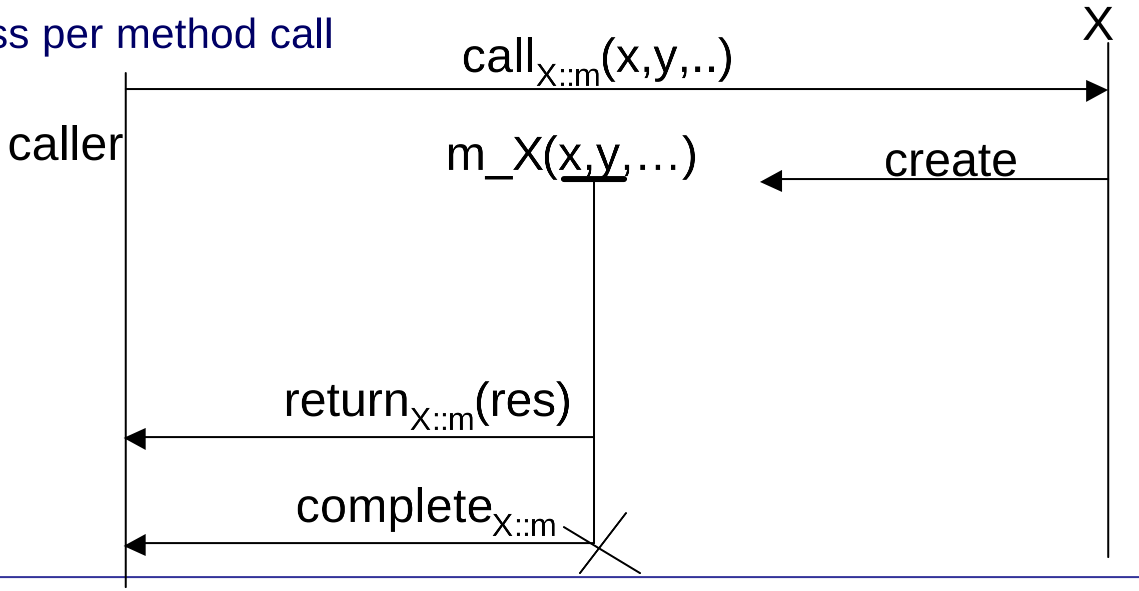
- lives message execution, implements the method behavior
- encapsulates the "stack frame" variables

⇒ predefined signals

$\text{call}_{X::m}$, $\text{return}_{X::m}$, $\text{complete}_{X::m}$

representing objects

- structure
  - UML object → IF process
  - attributes & associations → variables
  - inheritance : replication of structural features
- behavior
  - state machines, actions → syntactic translation (almost)
  - operation calls X::m(x,y,…) :
    - one IF process per method call


X
$\text{call}_{X::m}(x,y,..)$
caller
m_X(x,y,…)
create
$\text{return}_{X::m}(\text{res})$
$\text{complete}_{X::m}$

polymorphism, concurrency…

polymorphism ⇒ dynamic binding resolved with signals

- the object state machine decides the version of a method with which it responds to a call$_{X::m}$

concurrency ⇒ activity group management

- each active object has an associated group manager
- it handles/dispatches external calls for objects of the group
- keeps track of the running object

run-to-completion

- implemented with dynamic priority rules
- e.g. : $\forall x,y.\ (x.manager = y) \Rightarrow x < y$

Verification

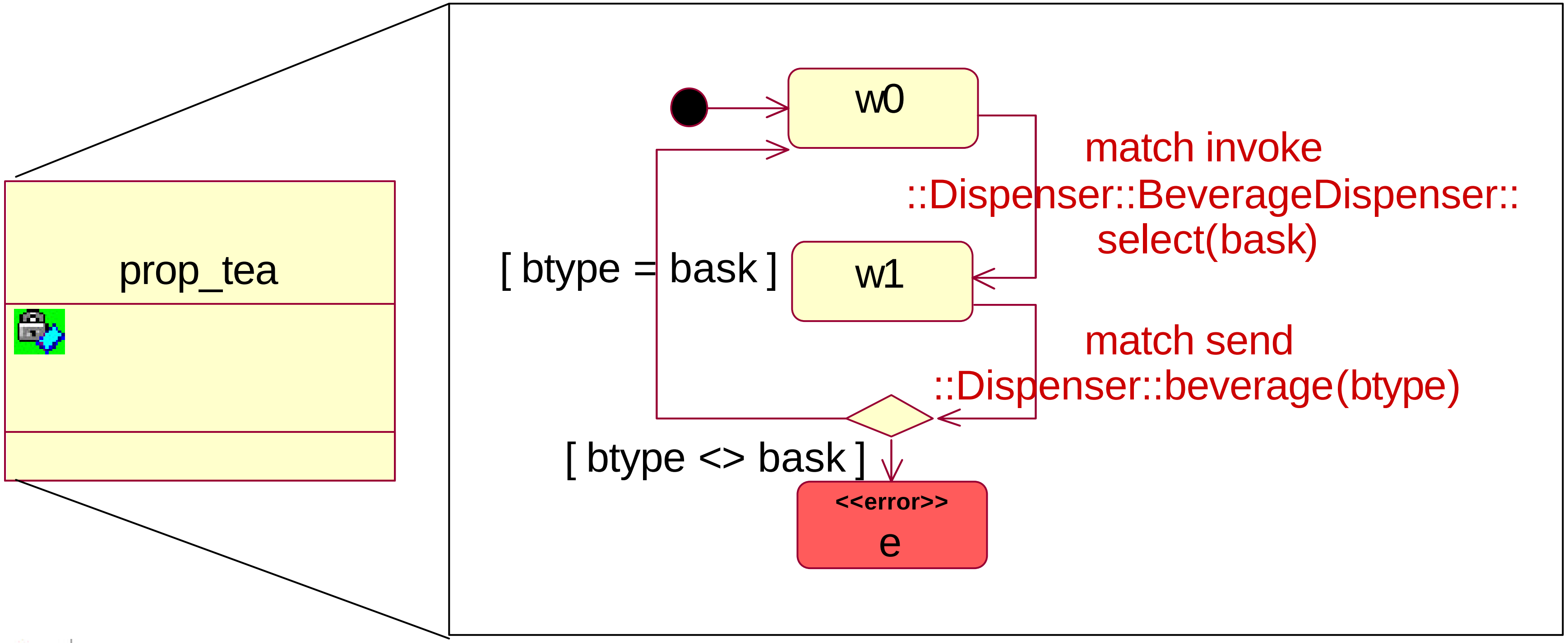
Main issue: how to express properties in UML ?

- generic properties: deadlocks, … (tool features)
- *time constraints*
- behavioural & timed properties: *observers*

Verification itself: use the existing tools

UML observer objects

- special objects monitoring the system state / events
  - synchronize with state changes at the semantic level (events)



observing events and states

- observable events (= state changes)
  - for operations: invoke, receive, accept, invokereturn, …
  - for signals: send, receive, accept
  - for actions: start, end
  - for states: entry, exit
- observable state
  - all entities reachable by navigation from already known entities (e.g. obtained from events)

semantics of real-time

- the OMEGA real time profile
  - imperative specifications: clocks, timers
  - declarative specifications: constraints on durations

- semantics: translation to timed automata primitives
  - Clocks and timers: straight forward
  - Events:
    - transition label + guard
    - attribute + clock, set at event occurrence
  - Durations: clock values or differences of clock values
  - Constraints:
    - time guards + urgency
    - observer

the toolset

Design choice

interconnectivity with most CASE tools : XMI

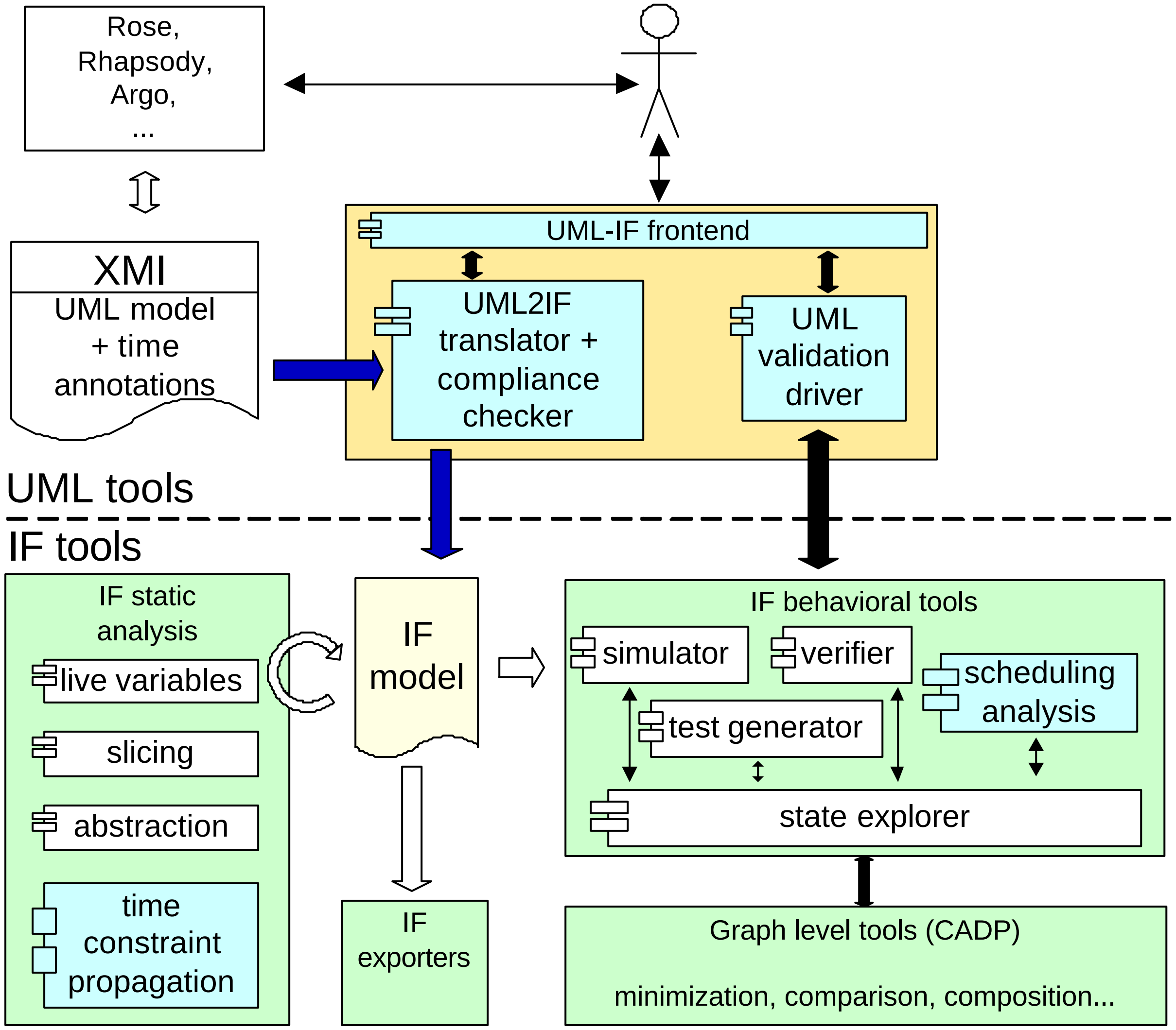
Model debugging

- step-by-step execution, state inspection
- scenario rewind/replay/save…
- control of non-determinism & time

Verification of properties: existing techniques

- State of the art: static analyse, on-the-fly verification,…
- Representation of time:
  - Symbolic representation of “zones”
  - Discrete time steps

toolset architecture



resources

OMEGA :

http://www-omega.imag.if

UML tools :

http://www-verimag.imag.fr/~ober/IFx

IF toolbox :

http://www-verimag.imag.fr/~async/IF