

# Timed Games

## UPPAAL-TIGA

Alexandre David  
1.2.05

## Overview

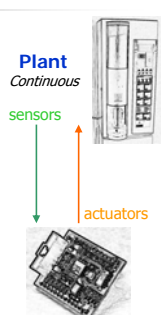
- Timed Games.
  - Algorithm (CONCUR'05).
  - Strategies.
  - Code generation.
  - Architecture of UPPAAL-TIGA.
  - Interactive game.
- Timed Games with Partial Observability.
  - Algorithm (ATVA'07).

13-03-2008 Alexandre David, TOV'08 2

## Why Timed Games?

- Real-time systems:
  - Systems where correctness depends on the logical order of events and on their **timings!**
  - ... in addition to correct computation.
- Real Time Model-checking:
  - Model the environment + the tasks.
  - Model  $\models \varphi$ ? *Automated proof.*

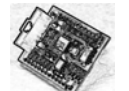
Plant  
*Continuous*



sensors

actuators

Controller Program  
*Discrete*

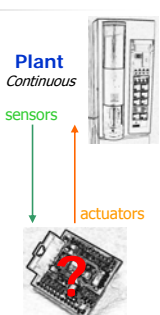


13-03-2008 Alexandre David, TOV'08 3

## Why Timed Games?

- Controller synthesis:
  - Model the environment + what a controller *can* do.
  - Generate the controller so that **controller  $\models \varphi$ !**  
Generate the *right* code automatically.
  - 2-player timed game: environment moves vs. controller moves.  
⇒ **Timed Game Automata.**

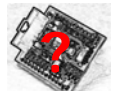
Plant  
*Continuous*



sensors

actuators

Controller Program  
*Discrete*



13-03-2008 Alexandre David, TOV'08 4

## Controller Synthesis/TGA

- Given
  - System moves  $S$ ,
  - Controller moves  $C$ ,
  - and a property  $\varphi$ ,
- find
  - a strategy  $S_c$  s.t.  $S_c || S \models \varphi$ ,
  - or prove there is no such strategy.

13-03-2008 Alexandre David, TOV'08 5

## Timed Game Automata

- Introduced by Maler, Pnueli, Sifakis [Maler & al. '95].
- The controller continuously observes the system (all delays & moves are observable).
- The controller can
  - wait (delay action),
  - take a controllable move, or
  - prevent delay by taking a controllable move.

13-03-2008 Alexandre David, TOV'08 6

### Timed Game Automata

- Timed automata with controllable and uncontrollable transitions.
- Reachability & safety games.
  - control:  $A \leftrightarrow \text{TGA.goal}$
  - control:  $A[\ ]$  not TGA.L4
- Memoryless strategy:
  - state  $\rightarrow$  action.

13-03-2008 Alexandre David, TOV'08 7

### TGA – Let's Play!

- control:  $A \leftrightarrow \text{TGA.goal}$

Strategy

- $x < 1 : \lambda$   
 $x = 1 : c$
- $x < 2 : \lambda$   
 $x \geq 2 : c$
- $x \leq 1 : c$
- $x < 1 : \lambda$   
 $x = 1 : c$

Note: This is *one* strategy. There are other solutions.

13-03-2008 Alexandre David, TOV'08 8

### Results

- [Maler & al. '95, De Alfaro & al. '01] There is a **symbolic iterative algorithm** to compute the set  $W^*$  of winning states for timed games.
- [Henziger & Kopke '99] **Safety** and **reachability** control are **EXPTIME-complete**.
- [Cassez & al. '05] Efficient **on-the-fly algorithm** for safety & reachability games.

13-03-2008 Alexandre David, TOV'08 9

### Algorithm

- On-the-fly forward** algorithm with a **backward** fix-point computation of the winning/losing sets.
  - Use all the features of UPPAAL in forward.
  - Possible to mix forward & backward exploration.
- Solved by Liu & Smolka 1998 for untimed games.
- Extended symbolic version at CONCUR'05.

13-03-2008 Alexandre David, TOV'08 10

**Initialization:**

$Passed \leftarrow \{S_0\}$  where  $S_0 = \{(l_0, \vec{0})\}'$ ;  
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$ ;  
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$ ;  
 $Depend[S_0] \leftarrow \emptyset$ ;

**Main:**

**while**  $(Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0])$  **do**  
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$ ;  
**if**  $S' \notin Passed$  **then**  
 $Passed \leftarrow Passed \cup \{S'\}$ ;  
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$ ;  
 $Win[S'] \leftarrow S' \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$ ;  
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = Post_\alpha(S')'\}$ ;  
**if**  $Win[S'] \neq \emptyset$  **then**  $Waiting \leftarrow Waiting \setminus \{e\}$ ;  
**else** (\* reevaluate \*)  
 $Win^* \leftarrow Pred_\lambda(Win[S]) \cup \bigcup_{S' \in T} Pred_\alpha(Win[S'])$ ;  
 $\cup_{S' \in T} Pred_\alpha(T \setminus Win[S']) \cap S$ ;  
**if**  $(Win[S] \subsetneq Win^*)$  **then**  
 $Waiting \leftarrow Waiting \cup Depend[S]$ ;  $Win[S] \leftarrow Win^*$ ;  
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$ ;  
**endif**  
**endif**  
**endwhile**

13-03-2008 Alexandre David, TOV'08

### Backward Propagation

Note: This is not a strategy, it's only the set of winning states.

13-03-2008 Alexandre David, TOV'08 12

## Backward Propagation

Predecessors of G avoiding B?

13-03-2008 Alexandre David, TOV'08 13

## pred<sub>t</sub> – From Federation to Zone

$$pred_t(\bigcup_i G_i, \bigcup_j B_j) = \bigcup_i \bigcap_j pred_t(G_i, B_j)$$

$$pred_t(G, B) = (G^\downarrow \setminus B^\downarrow) \cap ((G \cap B^\downarrow) \setminus B)^\downarrow$$

13-03-2008 Alexandre David, TOV'08 14

## Query Language (1)

- Reachability properties:
  - control:  $A[ p \cup q ]$
  - control:  $A \lt \! \! \rightarrow q \Leftrightarrow \text{control: } A[ \text{true} \cup q ]$
- Safety properties:
  - control:  $A[ p \cap W q ]$
  - control:  $A[ ] p \Leftrightarrow \text{control: } A[ p \cap W \text{false} ]$
- Tuning:
  - change search ordering,
  - add back-propagation of winning+losing states.

*Back-propagate winning states & BFS+DFS.*

*Back-propagate losing states & BFS-BFS.*

13-03-2008 Alexandre David, TOV'08 15

## Query Language (2)

- Time-optimality
  - control<sub>t\*</sub>(u,g):  $A[ p \cup q ]$ 
    - u is an upper-bound to prune the search, act like an invariant but on the path = expression on the current state.
    - g is the time to the goal from the current state (a lower-bound in fact), also used to prune the search. States with  $t+g > u$  are pruned.
- Cooperative strategies.
  - $E \lt \! \! \rightarrow \text{control: } \varphi$
  - Property satisfied iff  $\varphi$  is reachable but the obtained strategy is maximal.

13-03-2008 Alexandre David, TOV'08 16

## Cooperative Strategies

- State-space is partitioned between states from which there is a strategy and those from which there is no strategy.
- Cooperative strategy suggests moves from the opponent that would "help" the controller.
- Being used in testing.

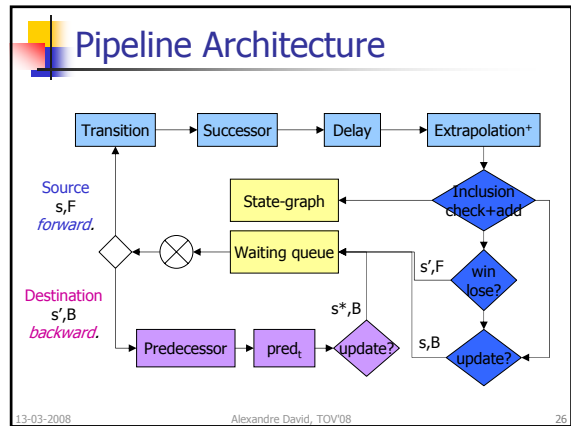
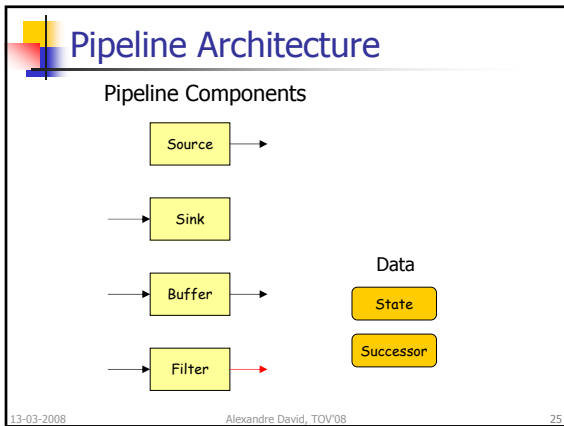
13-03-2008 Alexandre David, TOV'08 17

## Strategies?

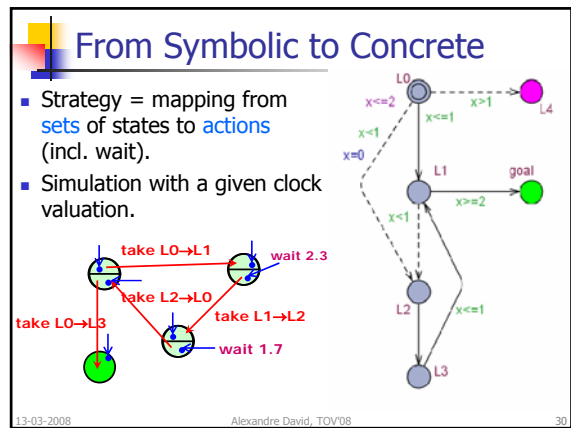
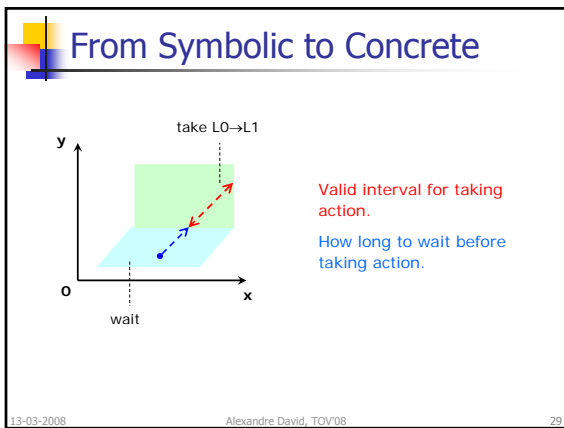
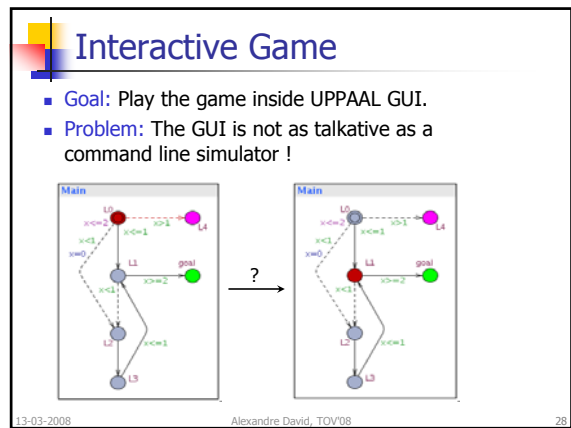
- The algorithm computes sets of winning and losing states, not strategies.
- Strategies are computed on top:
  - Take actions that lead to winning states (reachability).
  - Take actions to avoid losing states (safety).
  - Partition** states with actions to guarantee progress.
  - This is done on-the-fly and the obtained strategy depends on the exploration order.

13-03-2008 Alexandre David, TOV'08 18





- ## Interactive Game
- How to play a (timed) strategy against the user?
    - Concrete simulator.
    - Actions depend on the point in time.
    - Allowed delays depend on the actions.
    - The GUI has limited feedback for showing *counter-actions*.
- 13-03-2008 Alexandre David, TOV'08 27



## Interactive Game – GUI

- Avoid "your action has been countered": Restrict selection w.r.t. the strategy.
- What is a "selectable action" for the user?
  - His own transition – if can take it before TIGA
  - The choice of TIGA
- the other actions are not selectable

13-03-2008 Alexandre David, TOV'08 31

## Timed Games with Partial Observability

- Previous: Perfect information.
  - Not always suitable for controllers.
- Partial observation.
  - States or events, here states.
  - Distinguish states w.r.t. observations.
  - Strategy keeps track of states w.r.t. observations.
  - Observations = predicates over states.

13-03-2008 Alexandre David, TOV'08 32

## Results

- Discrete event systems
  - [Kupferman & Vardi '99, Reif '84, Arnold & al. '03]. Game given as modal logic formula: Full-observation as hard as partial observation.
  - [Chatterjee & al. '06, De Wulf & al. '06]. Game given as explicit graph: Full-observation **PTIME**, partial observation **EXPTIME**.
- Timed systems, game given as a TA
  - [Cassez & al. '07] Efficient on-the-fly algorithm, **EXPTIME**.

13-03-2008 Alexandre David, TOV'08 33

## State Based Full Observation

- 2-player reachability game, controllable + uncontrollable actions.
- Full observation: in  $l_2$  do  $c_1$ , in  $l_3$  do  $c_2$ .

13-03-2008 Alexandre David, TOV'08 34

## State Based Partial Observation

- Partition the state-space  $l_2=l_3$ .
- Can't win here.

13-03-2008 Alexandre David, TOV'08 35

## State Based Partial Observation

Winning Strategy:

- after: ● play  $c_1$
- after: ●● play  $c_1$
- after: ●●● play  $c_2$

The controller can observe each state's change

13-03-2008 Alexandre David, TOV'08 36

Franck Cassez ATVA'07

## Observation For Timed Systems

In Continuous Timed Systems, "next state" is reached by:

- ▶ either a **discrete** step
- ▶ or a **continuous** time-step

▶ Possible Observations:

- each 1/2 t.u.:
- each 1/4 t.u.:
- as it wishes:

2000 times within 1 t.u.

▶ the controller **cannot observe** each state's change

Issue: When does the controller observe the system ?

13-03-2008 Alexandre David, TOV'08 37

Franck Cassez ATVA'07

## Stuttering-Free Invariant Observations

(On, 0)  $\xrightarrow{8}$  (Sensor, 0)  $\xrightarrow{9}$  (Sensed, 0)  $\xrightarrow{8.7}$  (Paint, 0)  $\xrightarrow{8.7}$  (Piston, 0)  $\xrightarrow{\text{kick?}}$  Off

Assumption: the controller can only see **changes** of observations

Stuttering-free observation:

Must play based on **stuttering-free** observations

13-03-2008 Alexandre David, TOV'08 38

Franck Cassez ATVA'07

## Rules of the Game

Requirements

- ▶ Observations have **special shapes**: must become true at some first instant / partition the state space e.g.  $20 \leq y < 24$  or  $1 \leq x < 2$
- ▶ The control objective  $\Phi$  is **stuttering closed**

Observation-Based Stuttering Invariant Strategies  
f is a **OBSI strategy** if:

Observation(p)  $\equiv_{\text{stutter}}$  Observation(p') implies  $f(p) = f(p')$

Control under Partial Observation Problem

Input: a TGA automaton  $G$   
a finite set of observations  $\mathcal{O}$ ,  
a control objective  $\Phi \subseteq \mathcal{O}^w$ .

Problem: Is there an **OBSI winning strategy** for  $(G, \Phi)$  ?

13-03-2008 Alexandre David, TOV'08 39

Franck Cassez ATVA'07

## On-the-Fly Algorithm

```

Initialization
Passed := {};
Waiting := {(s0, alpha, W) | alpha in Sigma, s0 in C, W = Next(s0) intersect alpha W not empty};
Win(s0) := {(s0) subset gamma(Goal) ? 1 : 0};
Losing(s0) := {(s0) subset gamma(Goal) and (Waiting = empty or exists s in Sigma, Sink(s0) not empty) ? 1 : 0};
Depend(s0) := 0;

Main
while (Waiting not empty and Win(s0) not 1 and Losing(s0) not 1) do
  e = (W, alpha, W') = pop(Waiting);
  if e not Passed then
    Passed := Passed union {e};
    Depend(W') := {W};
    Win(W') := 1 if Goal in Sink(W') else 0;
    Losing(W') := 1 if W' subset gamma(Goal) and Sink(W') not empty else 0;
    if (Losing(W') not 1) then
      if (Win(W') not 1) then
        if (NewTrans := {(W', alpha, W'') | alpha in Sigma, W' = Sink(W') and W'' not empty}) then
          if (NewTrans not empty) then
            Losing(W') := 1;
            if (Win(W') or Losing(W')) then
              Waiting := Waiting union {e};
              Waiting := Waiting union NewTrans;
            else
              recalculate 1;
          else
            Win(W') := 1 if (Goal in Sink(W')) else 0;
            if (Win(W') then
              Waiting := Waiting union Depend(W');
              Win(W') := 1;
              Losing(W') := 1 if (Losing(W') or exists s in Sigma, Sink(s0) not empty) else 0;
            else
              Losing(W') := 1;
              if (Win(W') or Losing(W') or 0) then
                Depend(W') := Depend(W') union {e};
            else
              recalculate 1;
          end if
        end if
      end if
    end if
  end while
end
  
```

13-03-2008 Alexandre David, TOV'08 40

Franck Cassez ATVA'07

## Algorithm

Partition the state-space w.r.t. observations.  
Observations 1 2 3.  
Winning/losing is observable.

13-03-2008 Alexandre David, TOV'08 41

Franck Cassez ATVA'07

## Algorithm

Initial state in some partition.  
Compute successors { set of states } w.r.t. a controllable action.  
Successors distinguished by observations.

13-03-2008 Alexandre David, TOV'08 42

### Algorithm

Construct the graph of sets of symbolic states.  
Back-propagate winning/losing states.

13-03-2008 Alexandre David, TOV'08 43

### Algorithm

- Back-propagation.
  - If all successors<sup>a</sup> are winning, declare current state winning, strategy: take action *a*.
  - If one successor<sup>a</sup> is losing, avoid action *a*.  
If no action is winning the current state is losing.

13-03-2008 Alexandre David, TOV'08 44

### Example

Observations: L, H, E, B, *y* in [0,1[

13-03-2008 Alexandre David, TOV'08 45

### Example

Partition:  $y=0$  (yellow circle),  $y=1$  (red circle), L, H, E (circles)

Actions: delay (solid arrow),  $y=0$  (dotted arrow), eject (blue arrow)

13-03-2008 Alexandre David, TOV'08 46