# Tutorial on Parametric Timed Automata for RT Scheduling

## Examples of scheduling models

Giuseppe Lipari and Étienne André

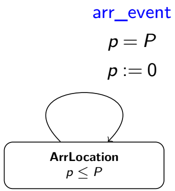
CRIStAL - Université de Lille

# Outline

# Outline

# Arrival models

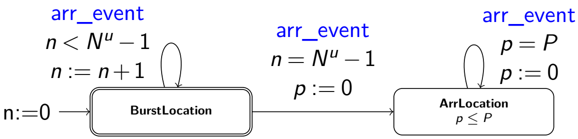
- A task is modeled by (at least) two automata: arrival automaton and job automaton
- The arrival automaton generates arrival events to activate the task's job
- A simple periodic arrival automaton:




- The sporadic arrival automaton is similar, substitute $=$ with $\geq$ in the guard condition and remove the invariant
- WARNING: sporadic tasks generate a large state space, use with care!

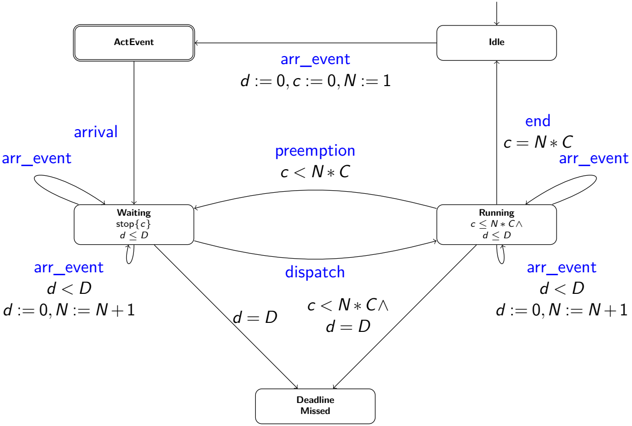
# Arrival curve

- Useful to upper bound burst arrivals



# Job automaton

- The job automaton synchronizes with the arrival automaton and with the scheduler automaton



# Job automaton

- The only possible parameter is the deadline $D$:
  - The computation time $C$ cannot be a parameter
  - current limitation of the tool: we cannot express a multiplication between a parameter and a discrete variable in an expression
- To avoid this problem, we can use a different model which assumes deadline = period
  - therefore, the only parameter is the computation time $C$
- Currently, we cannot have a generic model with $C$ and $D$ both parameters
  - Work is in progress !!
- Notice that the period $T$ can be a parameter of the arrival automaton (no restrictions)

# Scheduler automaton

- Interacts with the job automata by synchronizing on events arrival, dispatch, preemption, end
- Basically, it represents the ready queue ordered by priority
  - One location for every configuration of the queue
- Not shown here
- This model can be reused for all task models

# Outline

Models

Example 1: response time calculation

Example 2: Sensitivity on computation times

Example 3: Offsets

Conclusions

# Generating PTA models

- In the following, we will generate PTA models for the IMITATOR tool using the RETIMI python scripts
  - https://github.com/YIYAYIYAYOUCHENG/RETIMI
  - Authors: Y. Sun and G. Lipari
- Usage:
  - to just generate the imitator model:

    ```
    generator.py --norun model.txt
    ```
  - to generate and launch IMITATOR

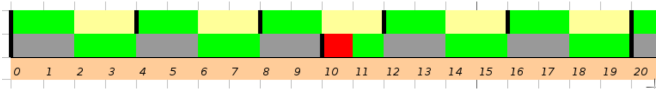
    ```
    generator.py model.txt
    ```

# Response time calculation

- Compute the worst-case response time of task $\tau_2$ in the following system

| Task | C | T |
|---|---|---|
| $\tau_1$ | 2 | 4 |
| $\tau_2$ | 5 | 10 |

- Iterative formula in Classic Scheduling Analysis (CSA)
- Gantt chart



# Response time calculation

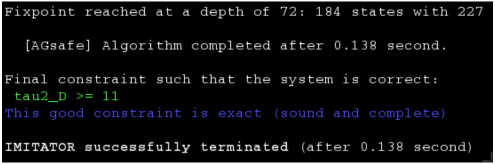
- The system (again)

| Task | C | T |
|---|---|---|
| $\tau_1$ | 2 | 4 |
| $\tau_2$ | 5 | 10 |

- With PTA:
  - use $D_2$ as a parameter
  - Files: response-time/response-time.txt and response-time/response-time.imi
- Imitator model generated with

  generator.py --norun response-time.txt
- You can run imitator with

  imitator -mode EF -incl -merge response-time.imi



# Response time as a function of the execution time

- Compute the response time of $\tau_2$ as a function of $C_1$

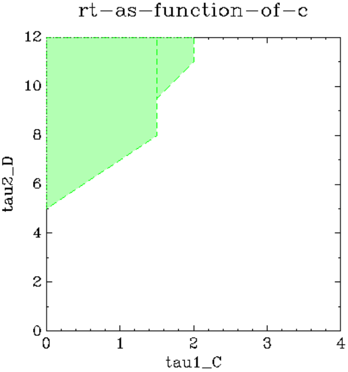
| Task | C | T |
|---|---|---|
| $\tau_1$ | ? | 4 |
| $\tau_2$ | 5 | 10 |

- Not possible with CSA (iterative formula)
  - need to compute the response time for different values of $C_1$
- With PTA:
  - Use $C_1$ and $D_2$ as parameters
  - See file `response-time/rt-as-function-of-c.txt` and `response-time/rt-as-function-of-c.imi`
  - Imitator model and analysis generated with generator.py --cart --xmax 4 --ymax 12 rt-as-function-of-c.txt
  - Results:

```
Final constraint such that the system is correct:
 tau2_D >= 5 + 2*tau1_C
& tau1_C >= 0
& 3 >= 2*tau1_C
OR
  2 >= tau1_C
& 2*tau1_C > 3
& tau2_D >= 5 + 3*tau1_C
This good constraint is exact (sound and complete)
```

# Cartography

- Option cart generates a cartography between [0,xmax] and [0,ymax]
- Observe how response time grows linearly until $C_1 = 1.6$, then there is a jump



# Outline

Models

Example 1: response time calculation

Example 2: Sensitivity on computation times

Example 3: Offsets

Conclusions

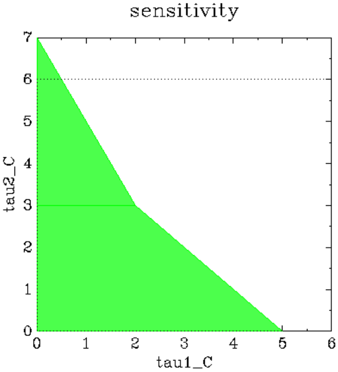
# Sensitivity analysis for periodic tasks

- Compute the schedulability region of the following a periodic tasks set task set

| Task | C | T |
|---|---|---|
| $\tau_1$ | ?? | 8 |
| $\tau_2$ | ?? | 12 |
| $\tau_3$ | 2 | 15 |
| $\tau_4$ | 3 | 18 |

- With CSA we can use the Hyperplane analysis
- With PTA:
  - `sensitivity/sensitivity.txt` and `sensitivity/sensitivity.imi`
  - Warning:
    - we use the `idlesched` model which stops the analysis at the first idle time (thanks to the critical instant theorem)
    - In this case the analysis time is extremely fast because of the short generated traces
    - If you use the `sched` model, the analysis takes $> 5$ minutes on my laptop

# Cartography



# Sensitivity on periods

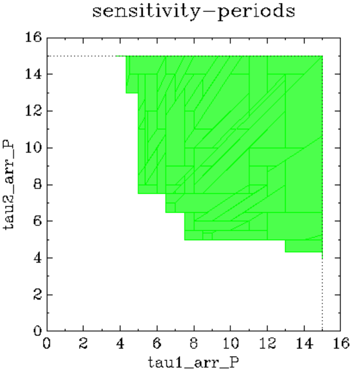
- For which values of $T_1$ and $T_2$ is the system schedulable ?

| Task | C | T |
|---|---|---|
| $\tau_1$ | 2 | ?? |
| $\tau_2$ | 2 | ?? |
| $\tau_3$ | 2 | 15 |
| $\tau_4$ | 3 | 18 |

- With CSA, there exists a complex analysis for it[1]
- With PTA:
  - `sensitivity/sensitivity-periods.txt` and `sensitivity/sensitivity-periods.imi`
- Warning
  - Here we are obliged to use `idlesched` otherwise the analysis does not converge!

[1] E. Bini, M. Di Natale, G. C. Buttazzo, *Sensitivity Analysis for Fixed-Priority Real-Time Systems*, Real-Time Systems 39 (1-3), pp. 5-30, August 2008.

# Cartography



# Period as a parameter

- Unfortunately, when periods are used as parameters, the analysis in general does not converge
- Reason:
  - the schedule has not a specified lenght
  - the analysis keeps producing longer and longer traces, without finding any fixed point
- (Partial) solution:
  - stop the analysis as soon as we are sure the system is schedulable
- In the case of sporadic tasks (or synchronous periodic tasks)
  - stop at the first idle time

# Outline

# Response time with offsets

- Compute the response time of all tasks in the following system

| Task | C | T | Off |
|---|---|---|---|
| $\tau_1$ | 3 | 10 | 1 |
| $\tau_2$ | 4 | 12 | 4 |
| $\tau_3$ | 6 | 18 | 0 |

- With CSA, we have to look at the hyperperiod
- With PTA
  - Use $D_i$ as parameters
  - See files offsets/offsets.txt and offsets/offsets.imi
  - Output:

```
Final constraint such that the system is correct:
 tau2_D >= 7
& tau1_D >= 3
& tau3_D >= 18
This good constraint is exact (sound and complete)
```

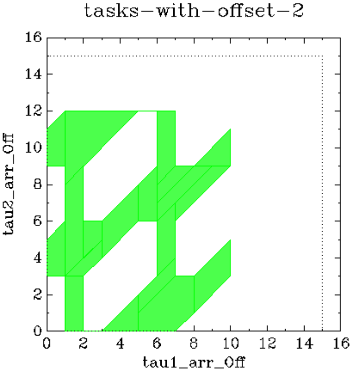
# Find optimal offsets

- For which values of the offsets this system is schedulable?

| Task | C | T | Off |
|---|---|---|---|
| $\tau_1$ | 2 | 10 | ?? |
| $\tau_2$ | 5 | 12 | ?? |
| $\tau_3$ | 5 | 20 | 0 |

- Not possible in CSA (only possibility: exhaustive enumeration)
- With PTA:
  - `offsets/tasks-with-offset-2.txt` and `offsets/tasks-with-offset-2.imi`
- Here we are obliged to analyse the entire schedule (so we must use the `sched`)
  - analysis time : 102 secs on my laptop

# Cartography



# Outline

# Complexity

- Depending on the model, analysis can take some time
- Complexity depends on
  - number of clocks (consider 2 or 3 clocks per task)
  - number of parameters
  - lenght of the traces
- Hints:
  - Whenever possible use the critical instant theorem to shorten the lenght of the traces
    - periodic synchronous (no offsets), sporadic
    - Not possible for offset based tasks
  - Avoid sporadic tasks when possible
  - Avoid arbitrary deadlines (greater than period)

# Extensions

- Work is under way to
  - further optimize the standard task model
  - generate models for other schedulers (EDF, and non preemptive versions of FP and EDF)
  - Generate precedence between tasks