Multiprocessor Global Schedulability Analysis in Continuous Time

Youcheng Sun

Grenoble, 2nd February 2016

Sant'Anna
Scuola Universitaria Superiore Pisa

ISTITUTO
DI TECNOLOGIE DELLA
COMUNICAZIONE,
DELL'INFORMAZIONE
E DELLA
PERCEZIONE

Scuola Superiore
Sant'Anna

retis
Real-Time Systems Laboratory

- Youcheng Sun and Giuseppe Lipari.
  "A Pre-Order Relation for Exact Schedulability Test of Sporadic Tasks on Multiprocessor Global Fixed-Priority Scheduling".
  @Real-Time Systems journal, 2015.

- Youcheng Sun.
  "Real-Time Schedulability Analysis with Formal Techniques (Chapter 9)". PhD thesis, Scuola Superiore Sant'Anna, 2015.

1. Introduction

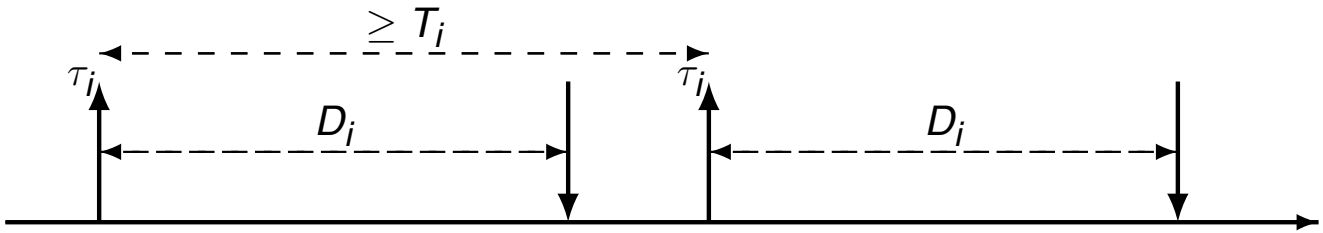
2. Multiprocessor G-FP Scheduling in LHA

3. Exact Schedulability Analysis

4. Approximate Schedulability Analysis

System model

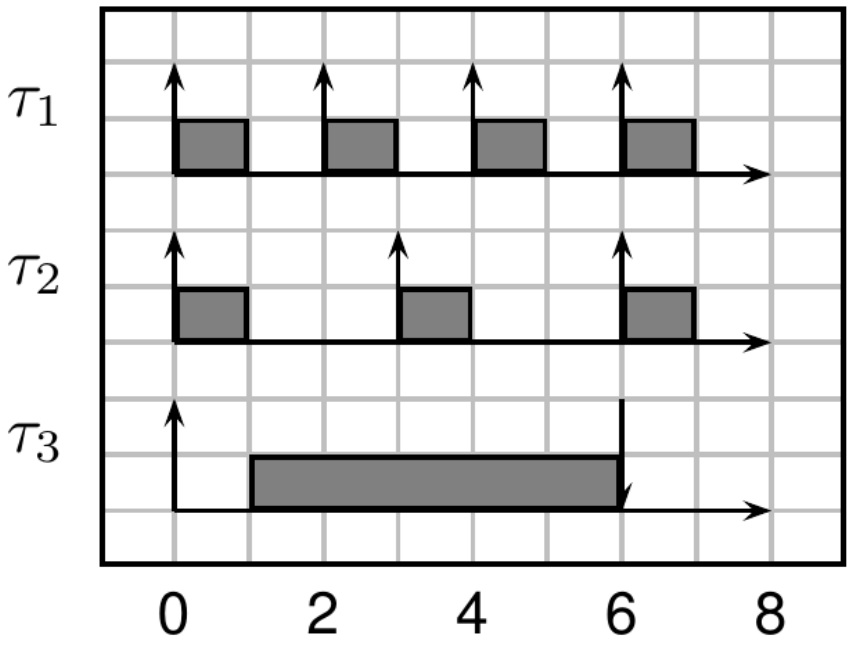
- n sporadic tasks $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$
  - $\tau_i = (C_i, D_i, T_i)$

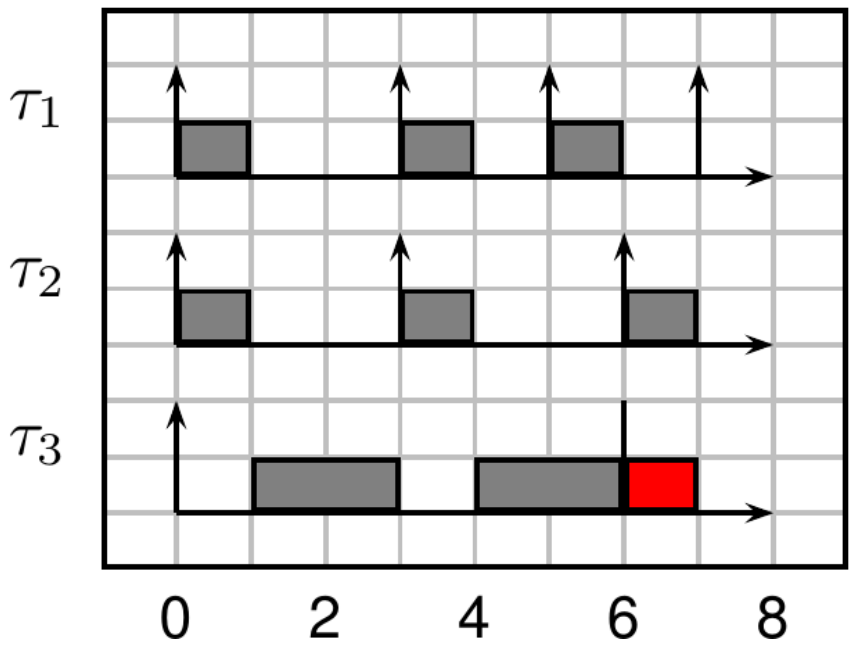


- m ($m \leq n$) identical processors
- the Global scheduling policy
  - Global Fixed-Priority (G-FP)

Uniprocessor vs multiprocessor

- The Worst-Case Scenario is known upon uniprocessor
  but not true for multiprocessor
- An example from [Baruah@RTSS07]





Multiprocessor global schedulability analysis

- Exact schedulability analysis needs to consider a large amount possible arrival patterns
  - There is no worst-case scenario
  - Sporadic task activations add further complexity

State of the art

- [Baker and Cirinei@PDS2007]
- [Bonifaci and Marchetti-Spaccamela @Algorithmica2012]
- [Geeraerts et al@RTSJ2013]
- [Burmyakov and Bini@RTNS2015]

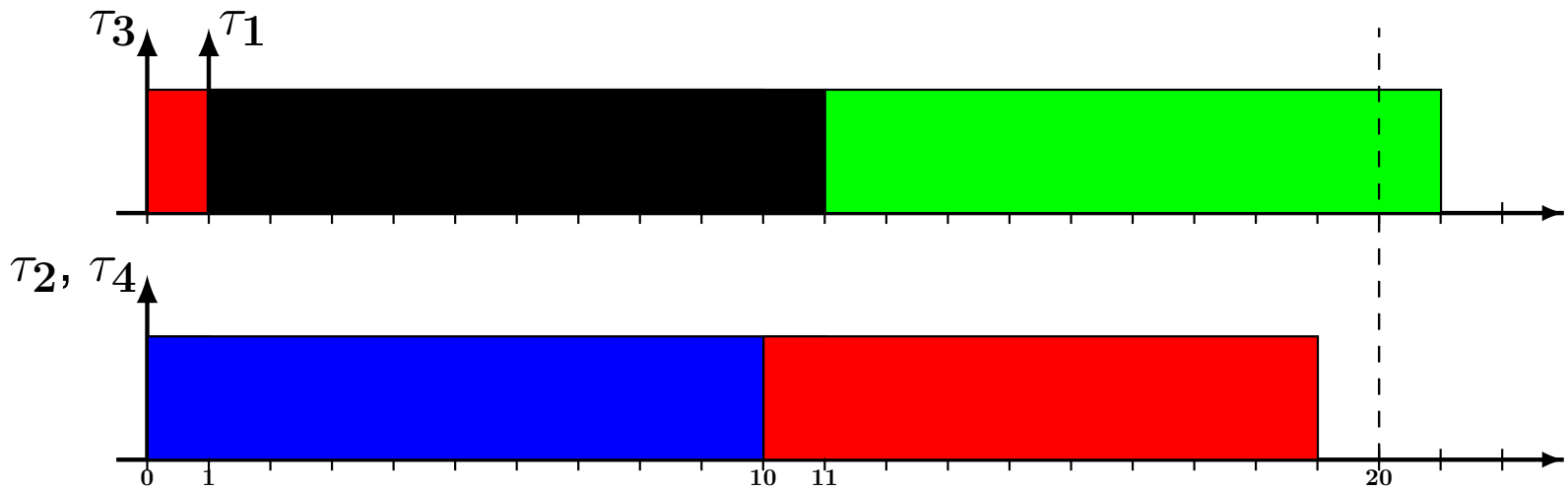
1. All in discrete time domain
2. Exact analysis in discrete time may be not always a good/safe idea

Continuous time vs discrete time

- Given a task set $\mathcal{T}$ that is schedulable by G-FP, is the schedulability preserved after scaling task parameters?
  - $\mathcal{T} = \{\tau_1 = (1, 4, 4), \tau_2 = (2, 6, 6)\} \rightarrow$
    $10 \cdot \mathcal{T} = \{10 \cdot \tau_1 = (10, 40, 40), 10 \cdot \tau_2 = (20, 60, 60)\}$
  - Continuous time domain: YES!
  - Discrete time domain: ?

Example

- $\tau_1 = (1, 4, 4)$, $\tau_2 = (1, 3, 3)$, $\tau_3 = (1, 3, 3)$, $\tau_4 = (1, 2, 2)$ and $m = 2 \rightarrow$ schedulable (under discrete time assumption)
- $\tau_1 = (10, 40, 40)$, $\tau_2 = (10, 30, 30)$, $\tau_3 = (10, 30, 30)$, $\tau_4 = (10, 20, 20) \rightarrow$ not schedulable



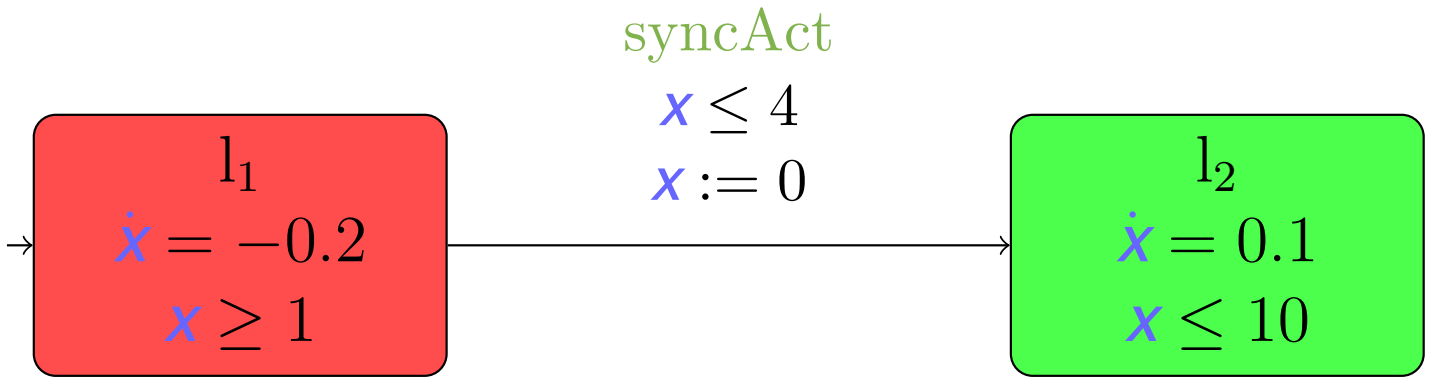
1. Introduction

2. Multiprocessor G-FP Scheduling in LHA

3. Exact Schedulability Analysis

4. Approximate Schedulability Analysis

Linear Hybrid Automata (LHA)



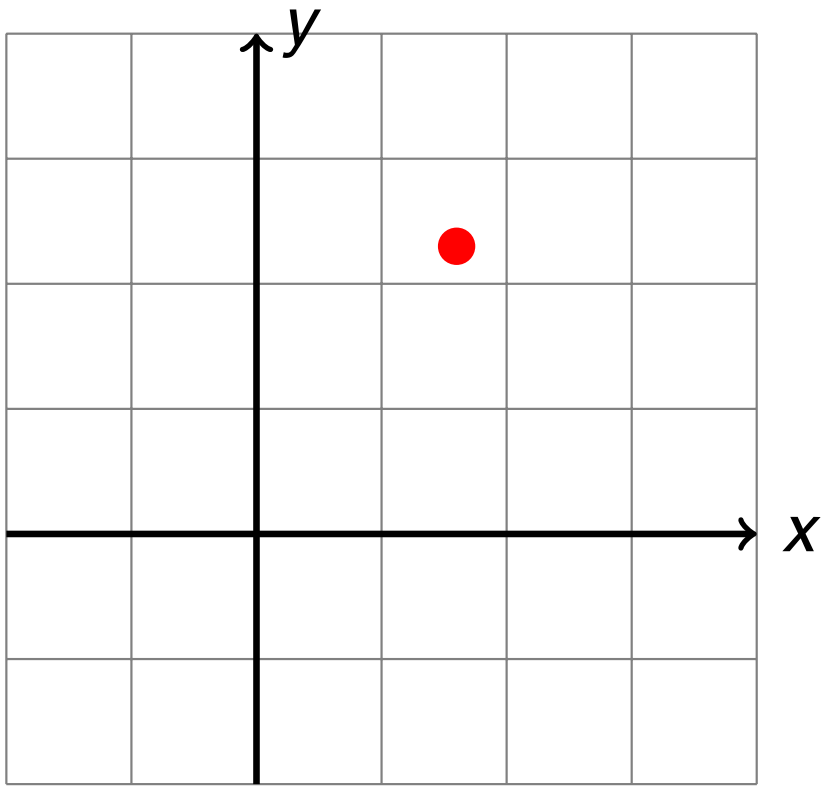
- continuous variables: x
  - its rate $\dot{x}$
- locations: l_1 and l_2
- transitions: $l_1 \rightarrow l_2$ with the synchronisation label ($\mathrm{syncAct}$), guard ($x \leq 4$) and update of the variable ($x := 0$)
- invariants: e.g. $x \geq 1$ in l_1

Concrete state and symbolic state

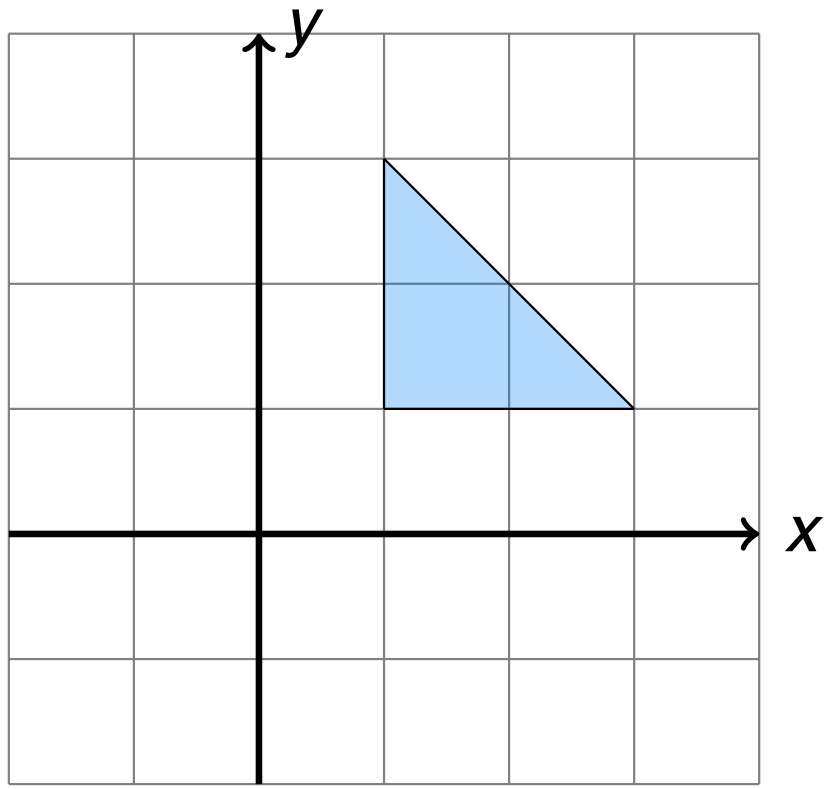
- A concrete state $s = (l, \nu)$
  - l is a location
  - ν is a valuation on variables
- A symbolic state $S = (l, \mathcal{C})$
  - l is a location
  - $\mathcal{C}$ is a linear constraint (represented by a convex region)
- $s_i \rightarrow s_j$: a step change of states due to time elapse or a transition
- $s_i \Rightarrow s_j$: a sequence of step changes

Example: Var $= \{x, y\}$

$\nu = (1.6, 2.3)$



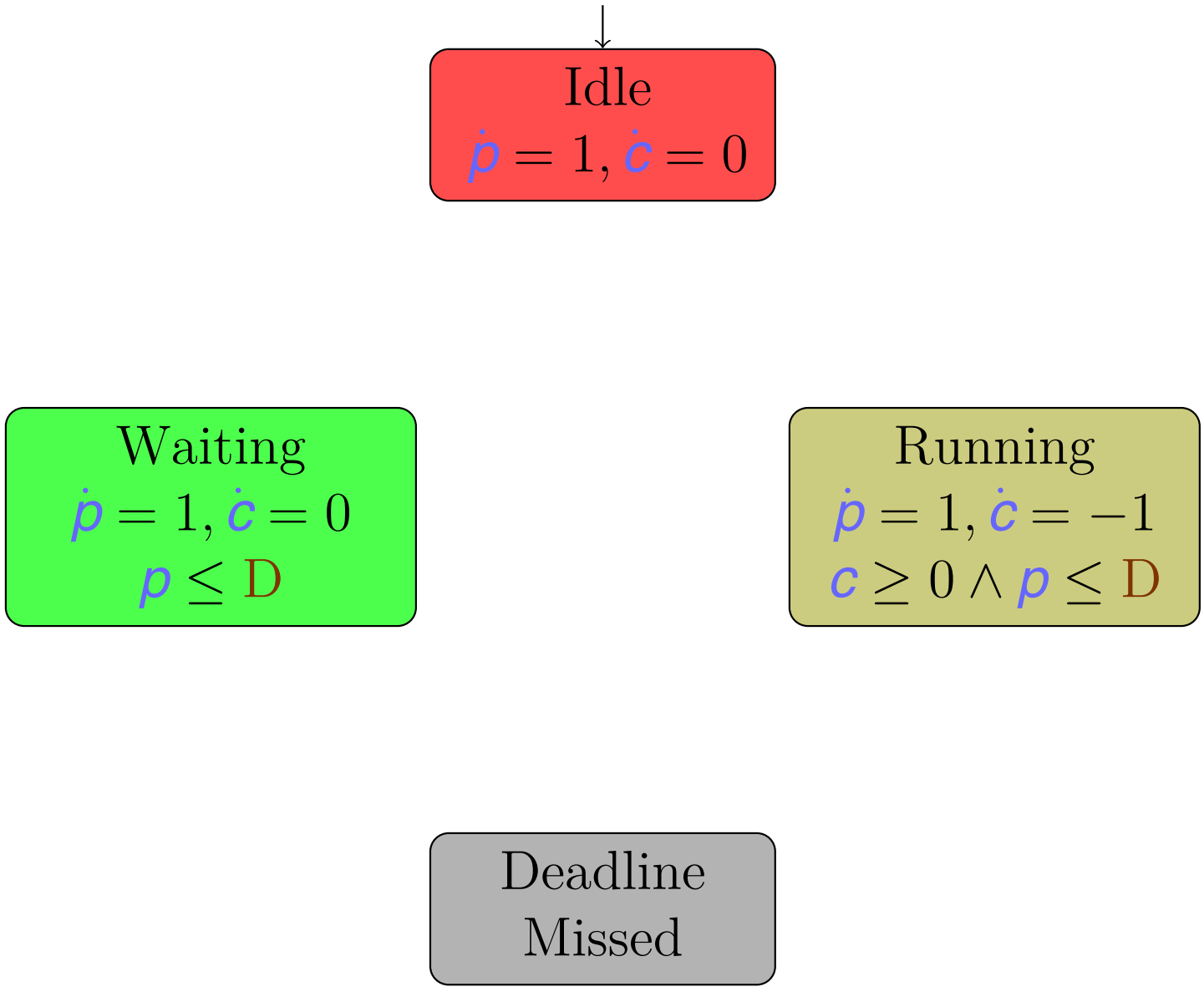
$\mathcal{C} = \{1 \leq x \leq 3 \wedge x + y \leq 4\}$



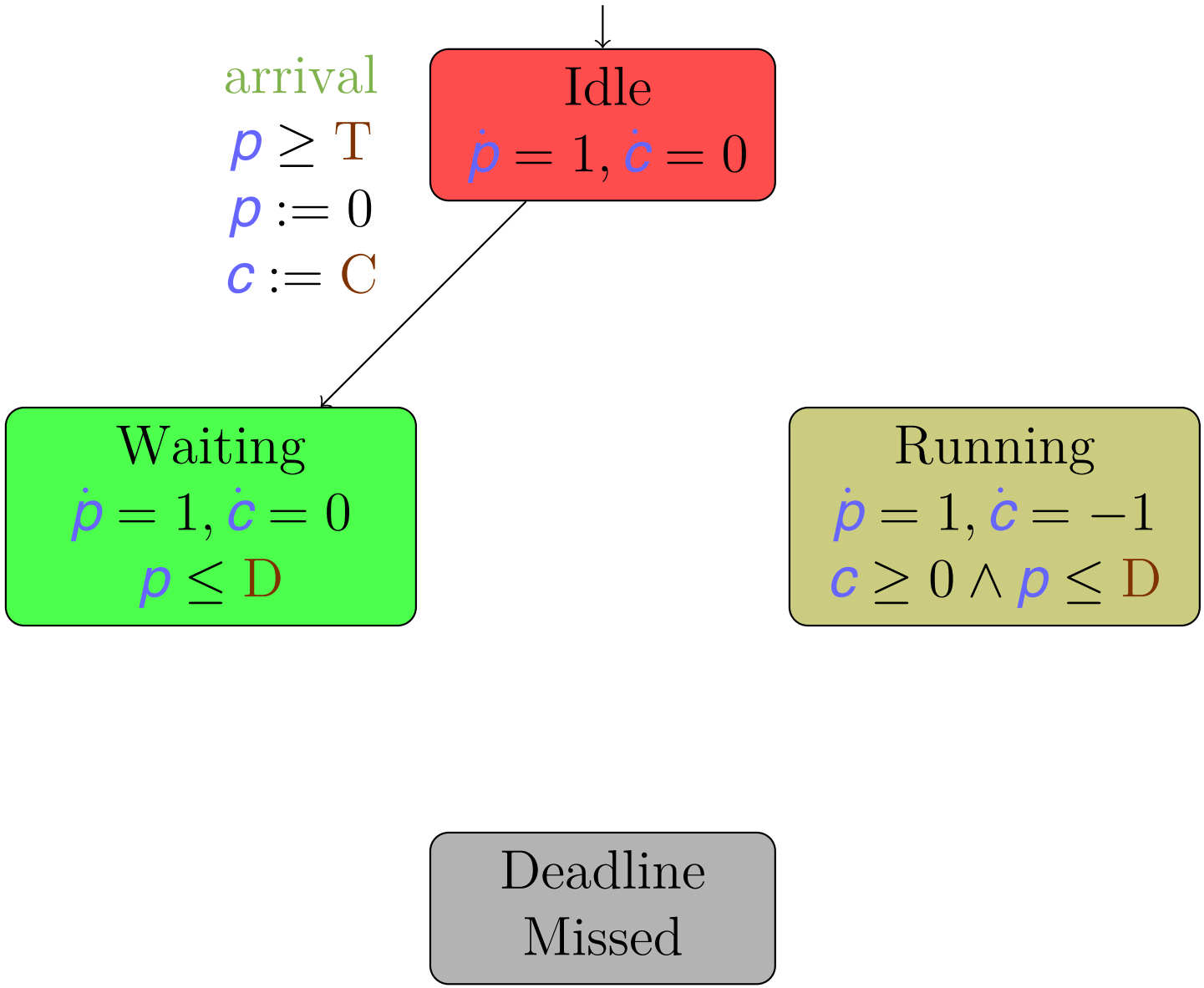
Scheduling problem in LHA

- One automaton, named task automaton, per task for modeling the task's behaviour
- One (G-FP) scheduler automaton for making scheduling decisions

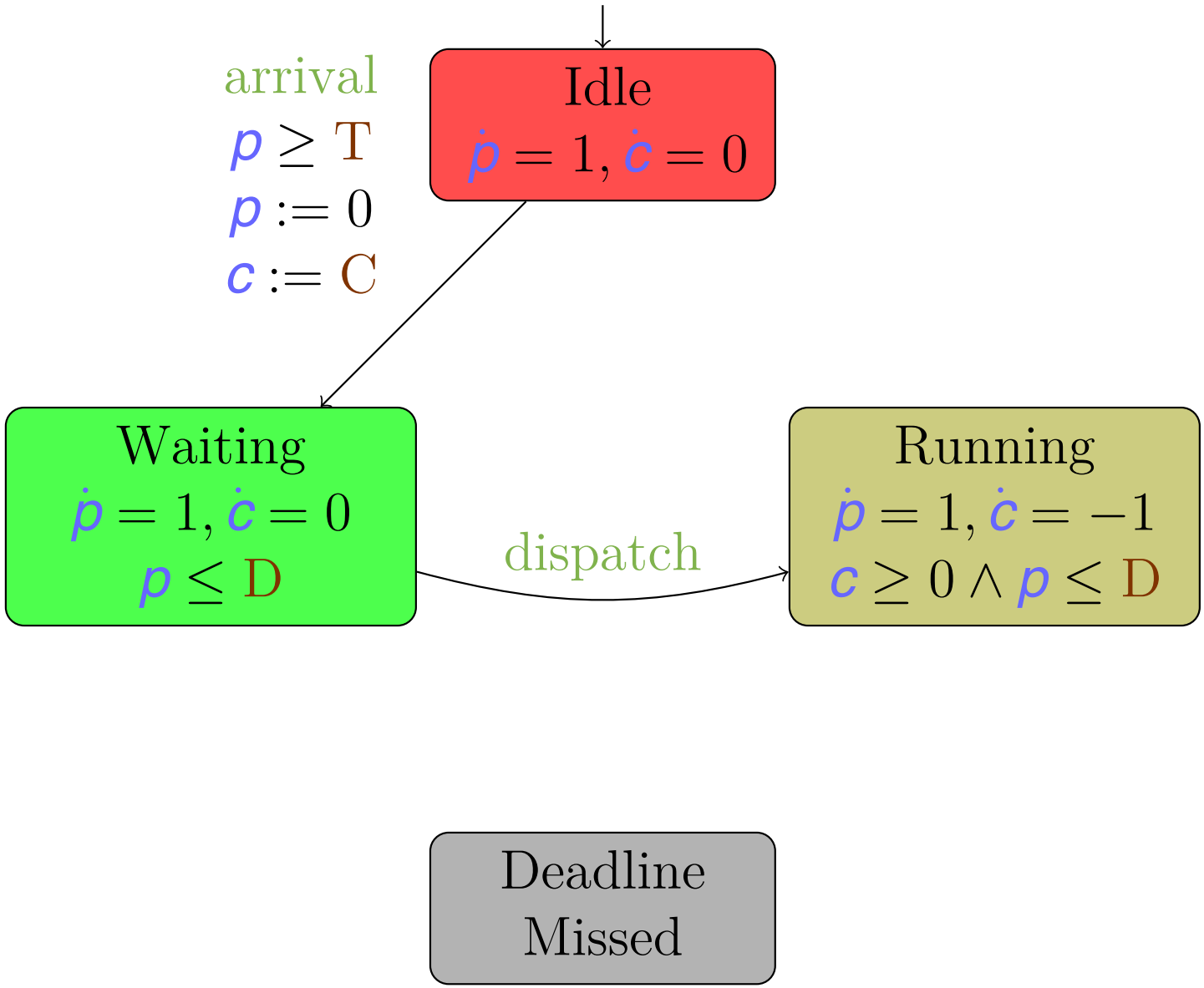
Task automaton



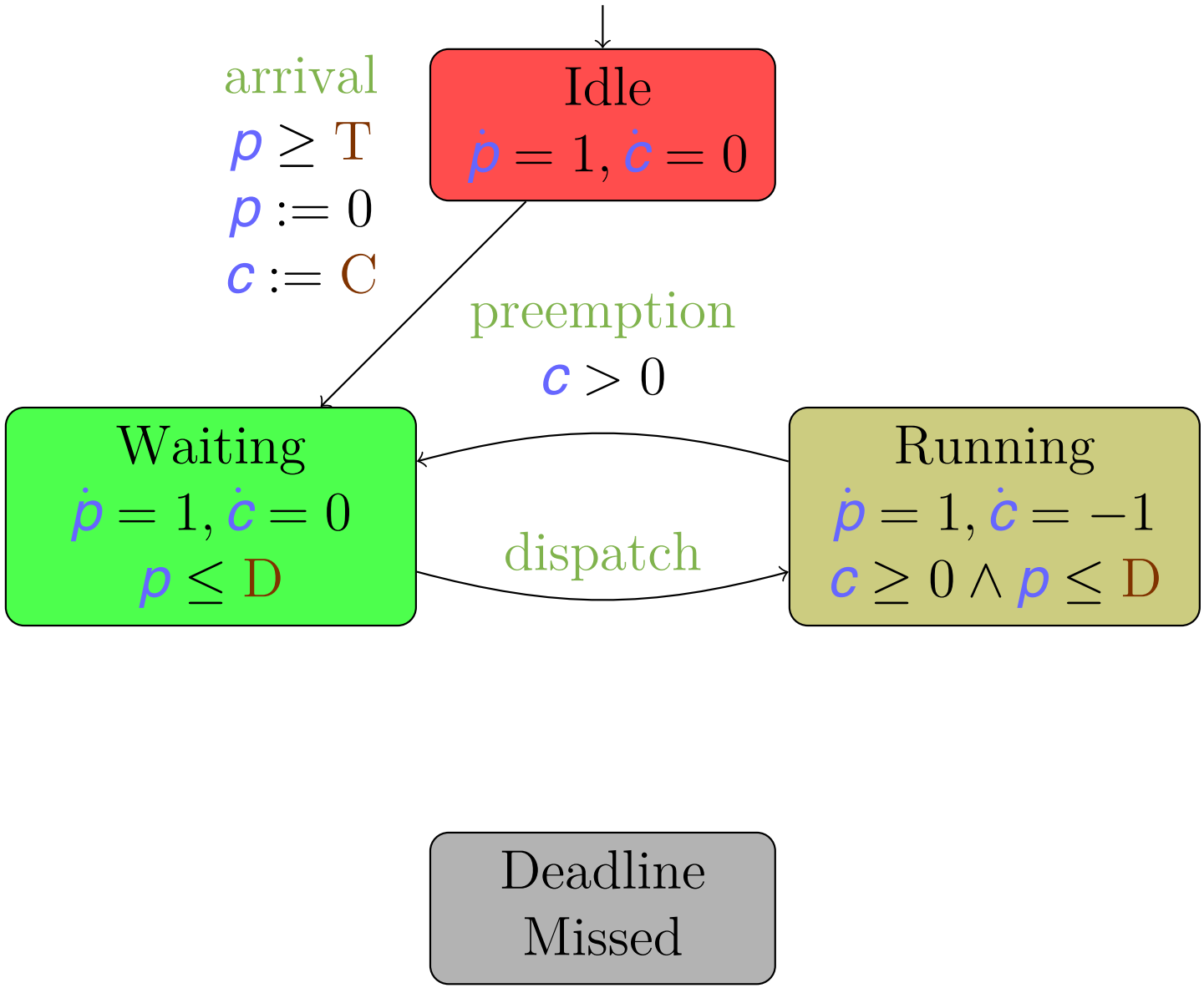
Task automaton



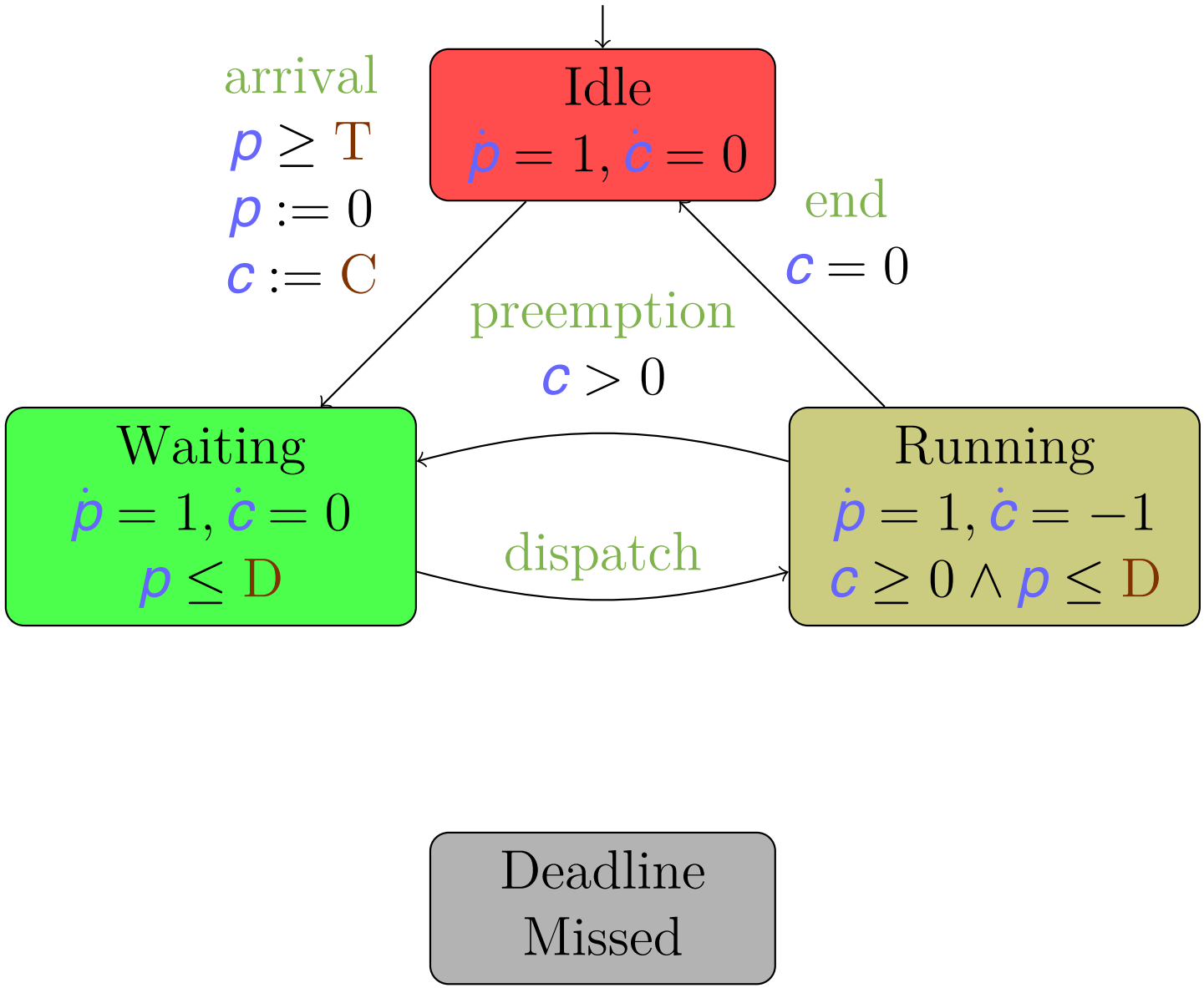
Task automaton



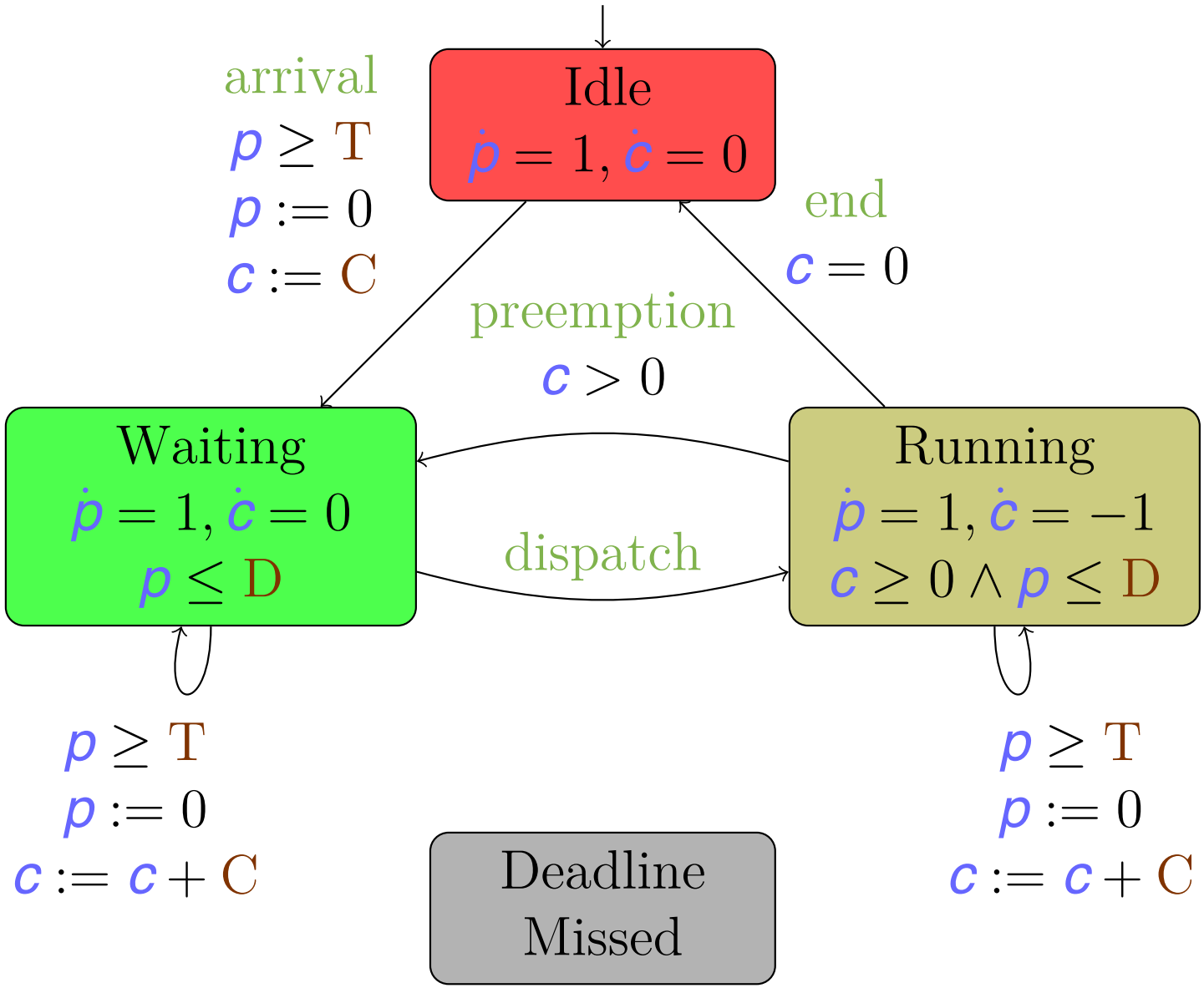
Task automaton



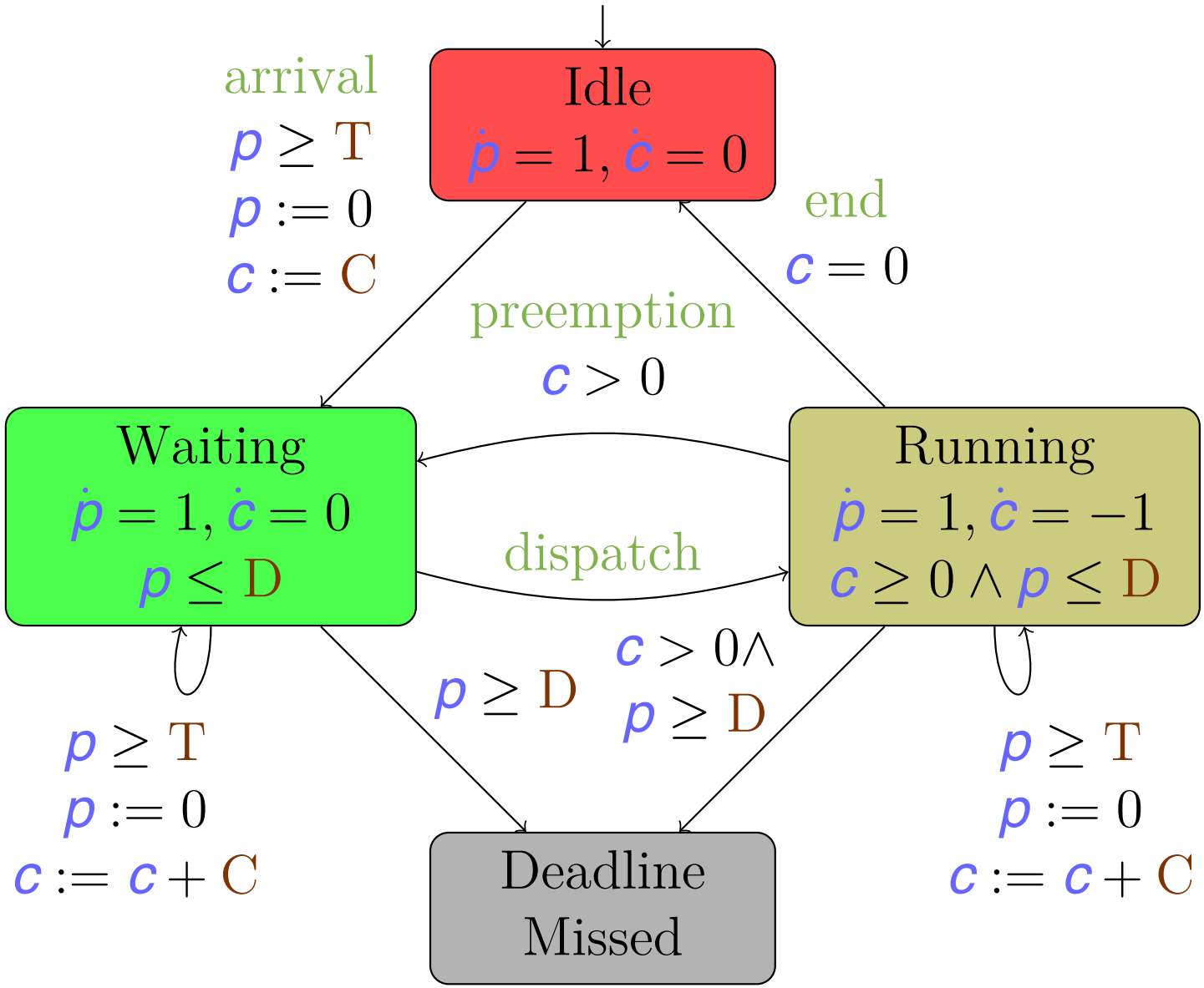
Task automaton



Task automaton



Task automaton



System automaton

- A system automaton $\mathsf{SA} = (\mathcal{T}, \mathsf{Sched})$
  - $\mathcal{T} = \{\mathsf{TA}_1, \ldots, \mathsf{TA}_n\}$ is a set of n task automata;
  - Sched is the scheduler automaton;
  - $\mathsf{SA} = \mathsf{Sched} \times \mathsf{TA}_1 \times \cdots \times \mathsf{TA}_n$.
- The schedulability problem is now the reachability problem of $\mathrm{DeadlineMissed}$ in SA.

1. Introduction

2. Multiprocessor G-FP Scheduling in LHA

3. Exact Schedulability Analysis

4. Approximate Schedulability Analysis

Slack-time pre-order

For the SA automaton, the *slack-time* pre-order is defined as follows: $\forall s_1, s_2$, $s_1 \succeq_{st} s_2$ if and only if

$$\forall \tau_i : \quad s_1.p_i \geq s_2.p_i \quad \wedge \quad s_1.c_i \geq s_2.c_i$$

and we say s_1 dominates s_2.

If a $\mathrm{DeadlineMissed}$ location is reachable from s_2, so is from s_1.

Extend $\succeq_{st}$ to symbolic states

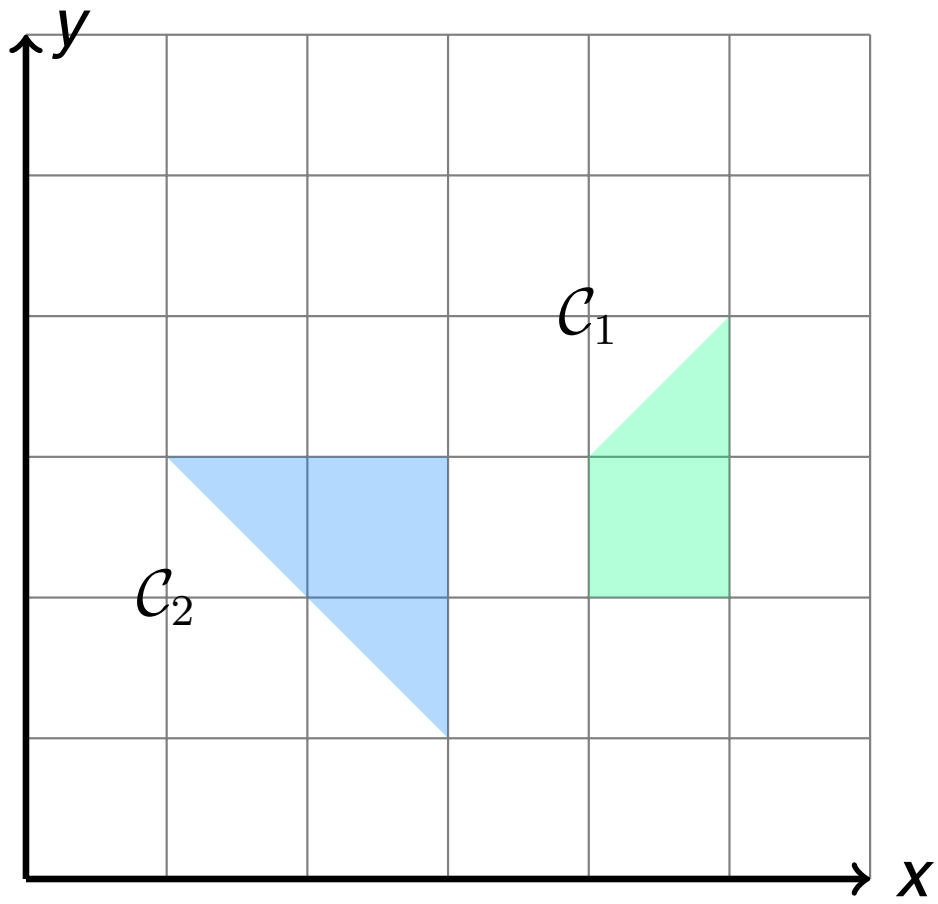
- A symbolic state $S = (I, \mathcal{C})$ abstracts a (possibly infinite) set of concrete states
- For two symbolic states S_1 and S_2, we say S_1 dominates S_2 if

$$\forall\ s_2 \in S_2,\ \exists\ s_1 \in S_1 \text{ s.t. } s_1 \succeq_{st} s_2$$

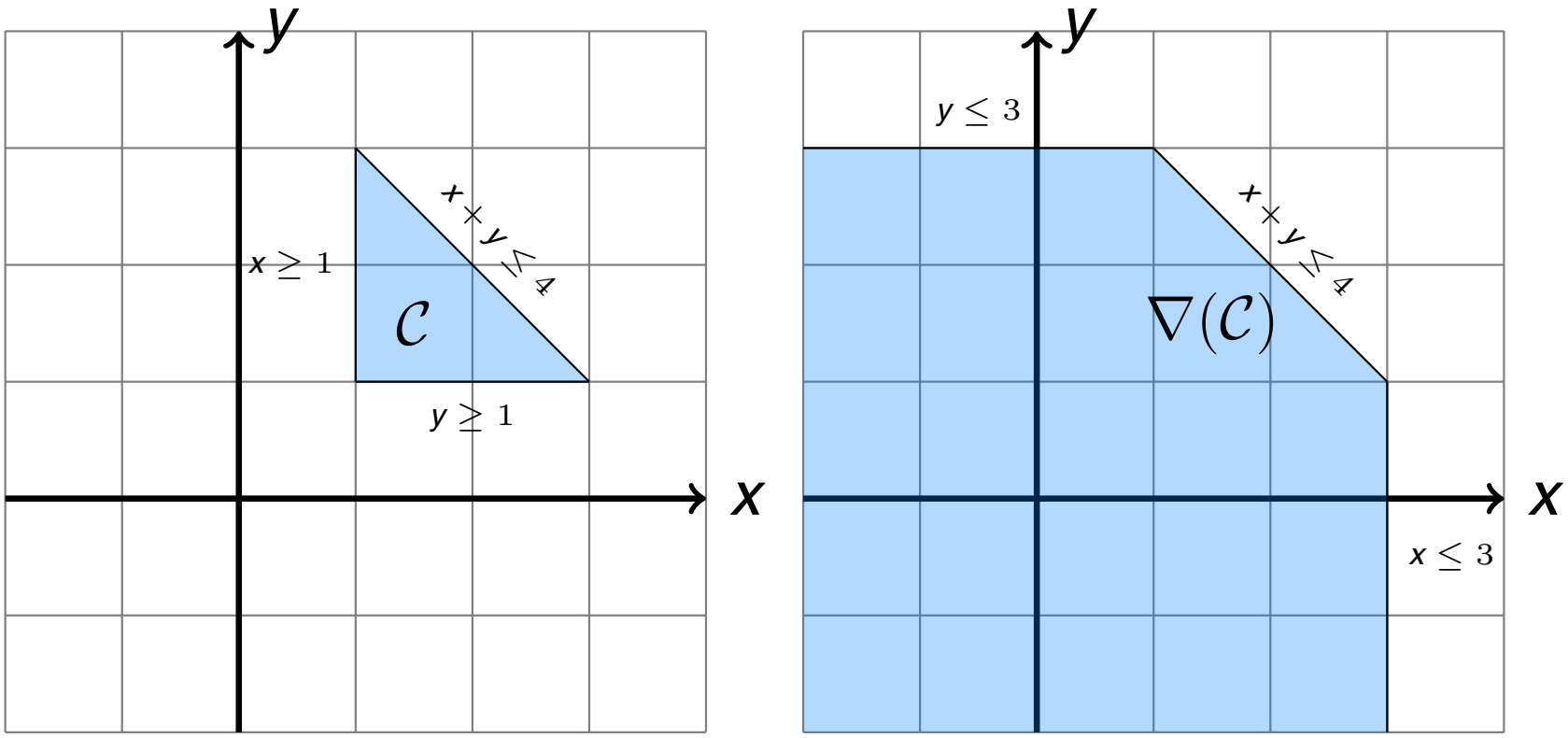
- $S_1.\mathcal{C} \supseteq S_2.\mathcal{C} \Rightarrow S_1 \succeq_{st} S_2$

More general case

- $(l_1, \mathcal{C}_1) \succeq_{st} (l_2, \mathcal{C}_2)$ if



A widening operator ∇



$(I_1, \mathcal{C}_1) \succeq_{st} (I_2, \mathcal{C}_2)$ if and only if $\nabla(\mathcal{C}_1)$ includes $\nabla(\mathcal{C}_2)$.

Schedulability analysis in system automaton

Algorithm 1 Schedulability Analysis in SA (SA-SA)

1: $R \leftarrow \{S_0\}$
2: **while** true **do**
3: $P \leftarrow \text{Post}(R)$
4: **if** $P \cap F \neq \emptyset$ **then**
5: **return** NOT schedulable
6: **end if**
7: $R' \leftarrow R \cup P$
8: $R' \leftarrow \text{Max}^{\succeq}(R')$
9: **if** $R' = R$ **then**
10: **return** schedulable
11: **else**
12: $R \leftarrow R'$
13: **end if**
14: **end while**

Decidability interval (when $D \leq T$)

For a set $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ of n sporadic tasks running on m processors, it is schedulable iff there is no deadline miss happens within $[0, L]$ with $L = \sum_{1 \leq i \leq m} C_i + \sum_{m < i \leq n} D_i$.

Proof sketch

(Assumption) $\mathcal{T}_k = \{\tau_1, \ldots, \tau_k\} \subseteq \mathcal{T}$ and there is a dominant time interval $[0, L_k]$: $\forall t' > L_k \;\; \exists t \in [0, L_k]$ s.t.

$$\forall \tau_i \in \mathcal{T}_k \;\; c_i(t) \geq c_i(t') \wedge p_i(t) \geq p_i(t')$$

$\Longrightarrow [0, L_k + D_{k+1}]$ is enough for detecting if τ_{k+1} misses a deadline ($c_{k+1} > 0 \wedge p_{k+1} = D_{k+1}$)

The base: $L_m = \sum_{1 \leq i \leq m} C_i \Longrightarrow L_n = L_m + \sum_{m < i \leq n} D_i$

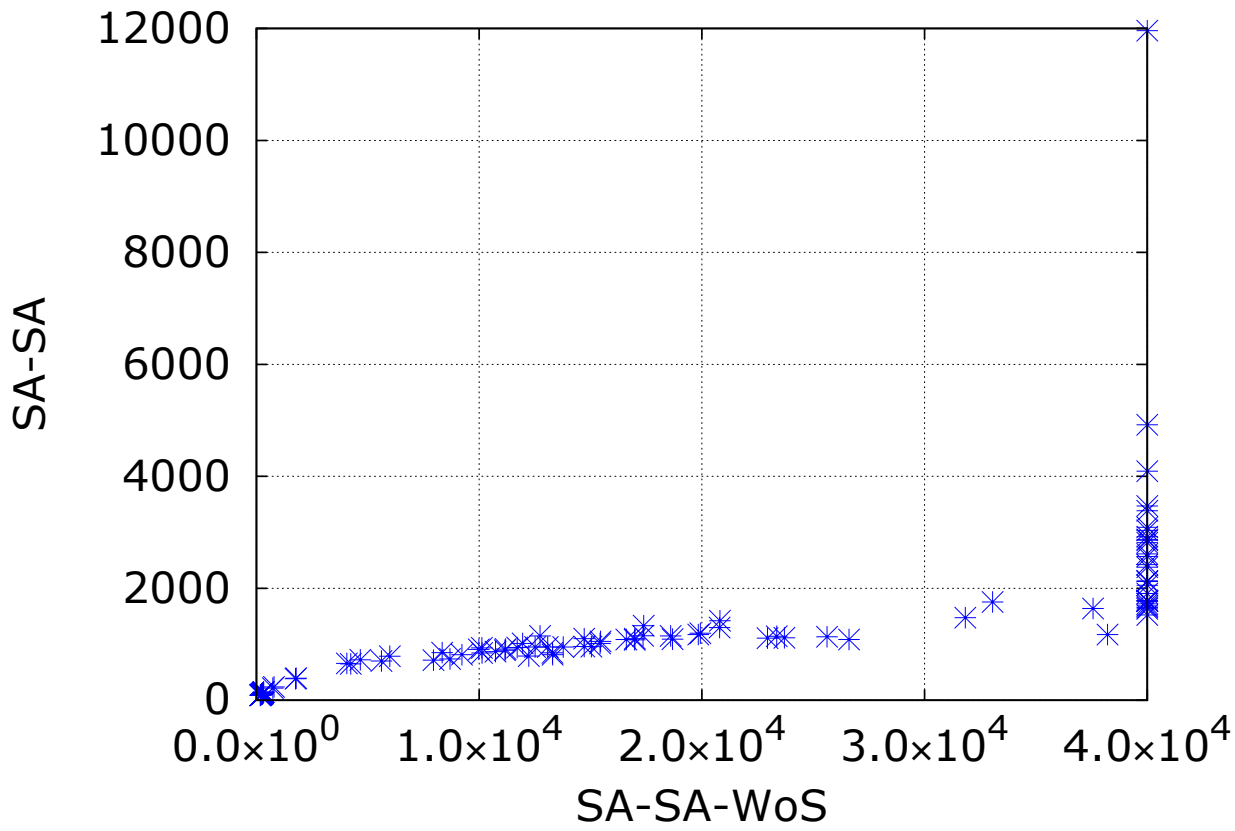
Termination

The termination of algorithm SA-SA is guaranteed.

Some simulation results

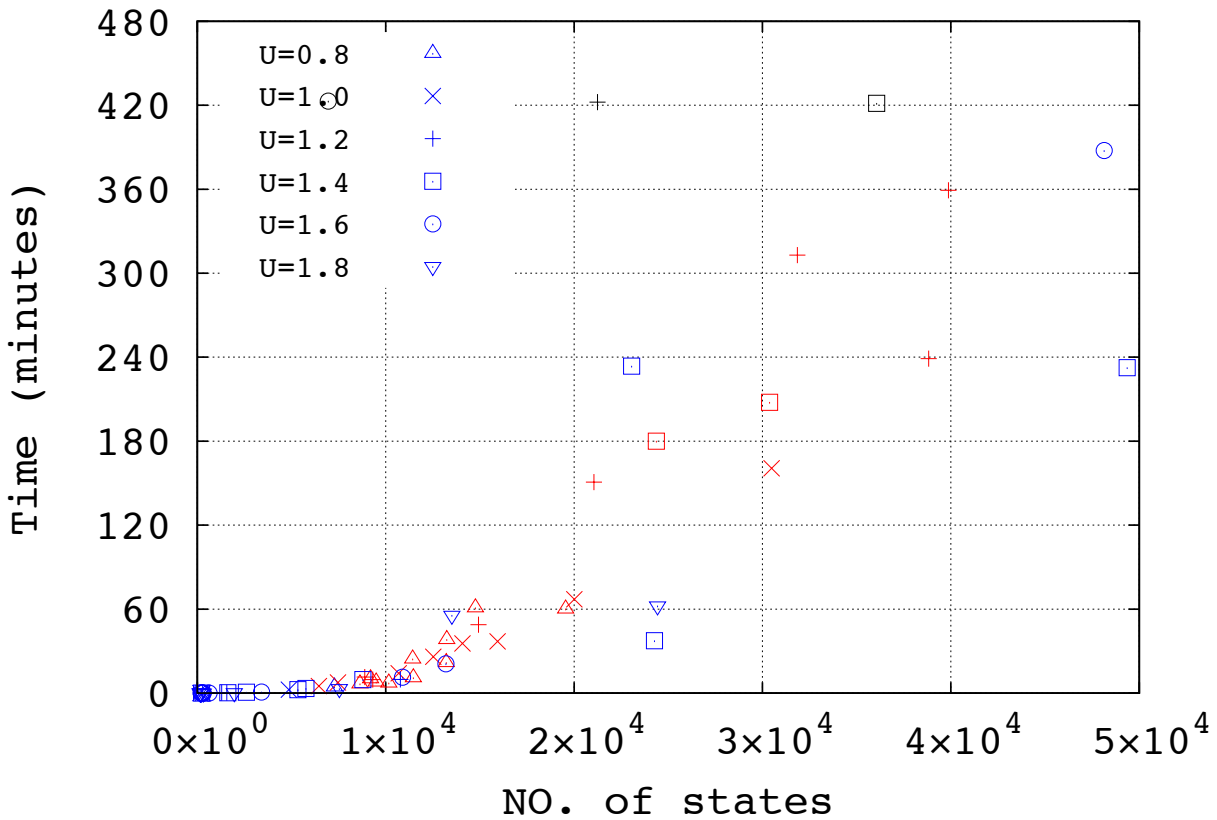
With or WithOut Simulation relation

- State space size



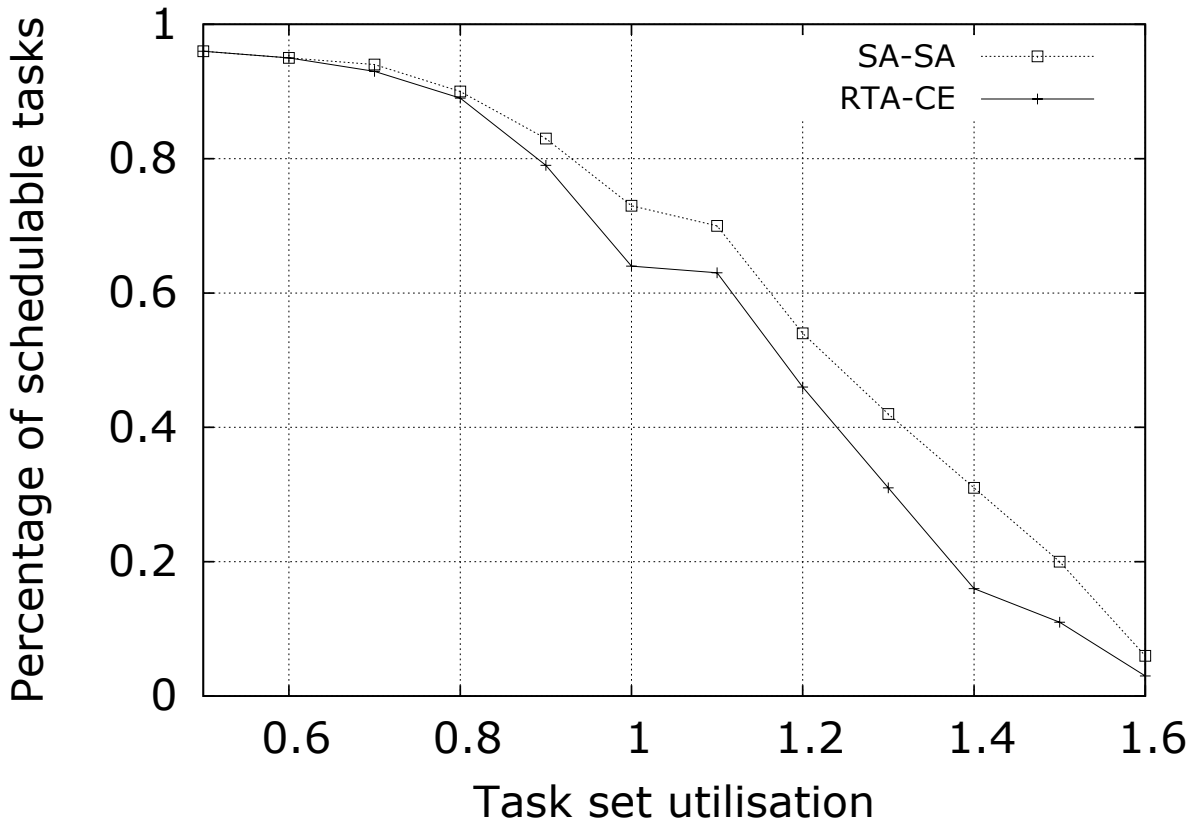
Runtime complexity

- $m = 2$ and $n = 6$



Exact analysis vs approximate analysis

- $m = 2$ and $n = 5$



Summary

- An exact G-FP schedulability test in continuous time
  - Deal with general task parameters
- A pre-order relation to for faster reachability analysis
  - Open source tool: **FO**rmal **R**eal-**T**ime **S**cheduler (**FORTS**)
- The decidability interval for multiprocessor global scheduling of sporadic tasks
- Complexity remains high...

1. Introduction

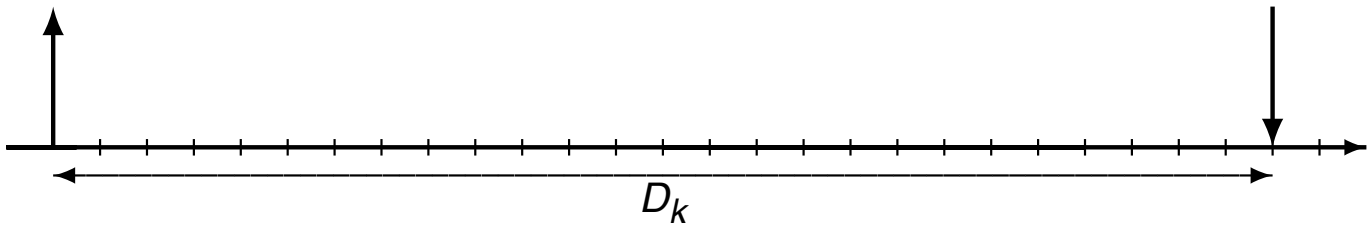
2. Multiprocessor G-FP Scheduling in LHA

3. Exact Schedulability Analysis

4. Approximate Schedulability Analysis

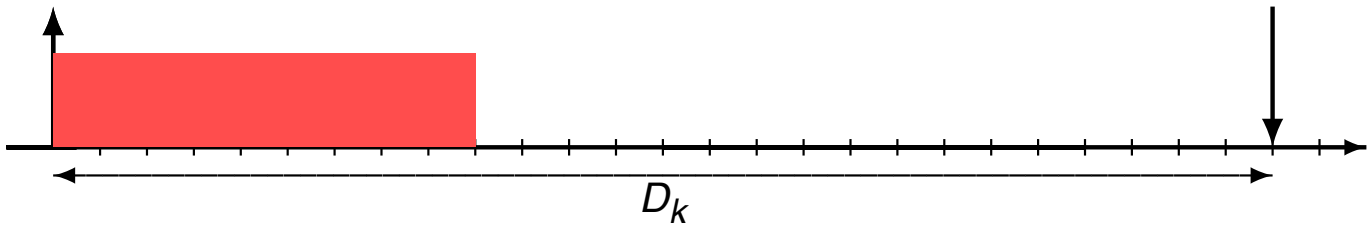
Response Time Analysis (RTA)

- The response time of a task τ_k is the duration it needs to complete its worst-case execution C_k



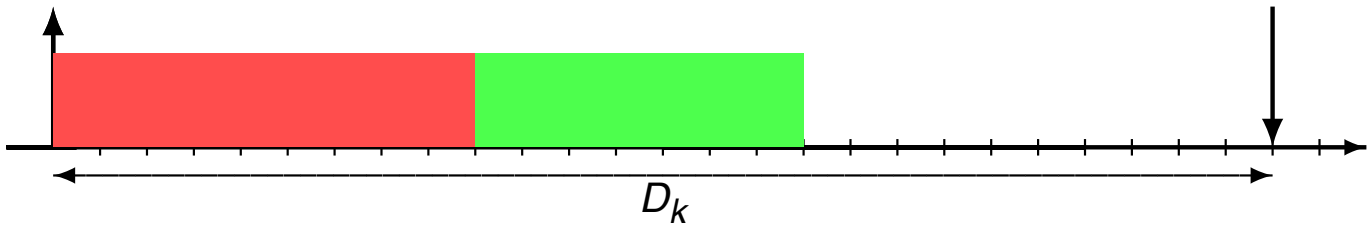
Response Time Analysis (RTA)

- The response time of a task τ_k is the duration it needs to complete its worst-case execution C_k



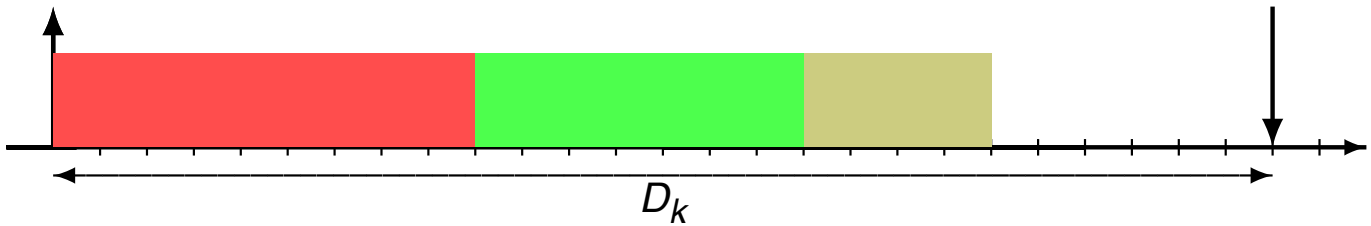
Response Time Analysis (RTA)

- The response time of a task τ_k is the duration it needs to complete its worst-case execution C_k



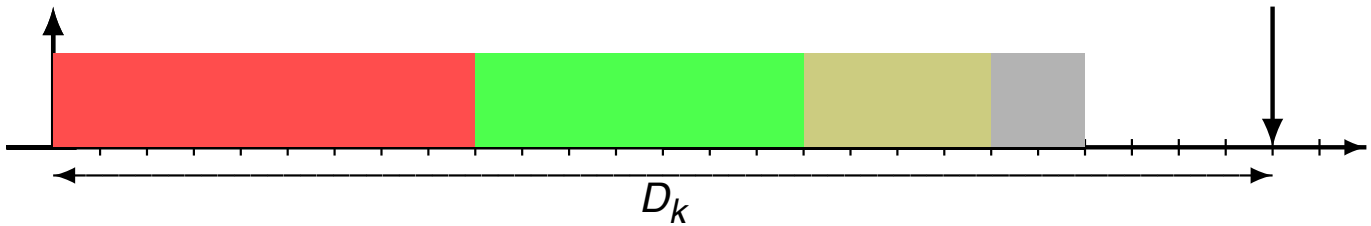
Response Time Analysis (RTA)

- The response time of a task τ_k is the duration it needs to complete its worst-case execution C_k



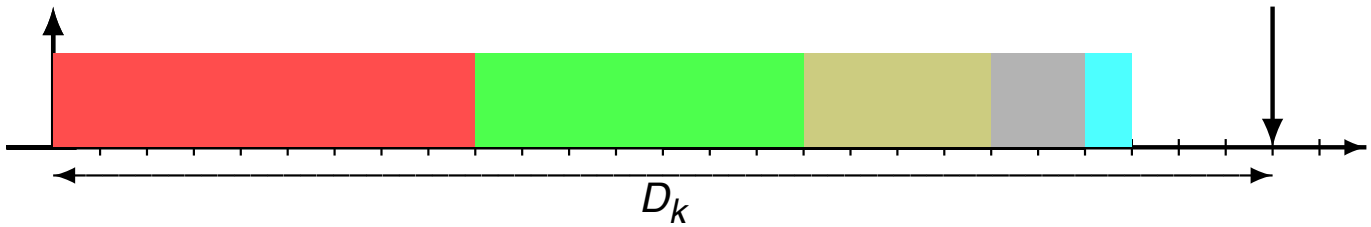
Response Time Analysis (RTA)

- The response time of a task τ_k is the duration it needs to complete its worst-case execution C_k



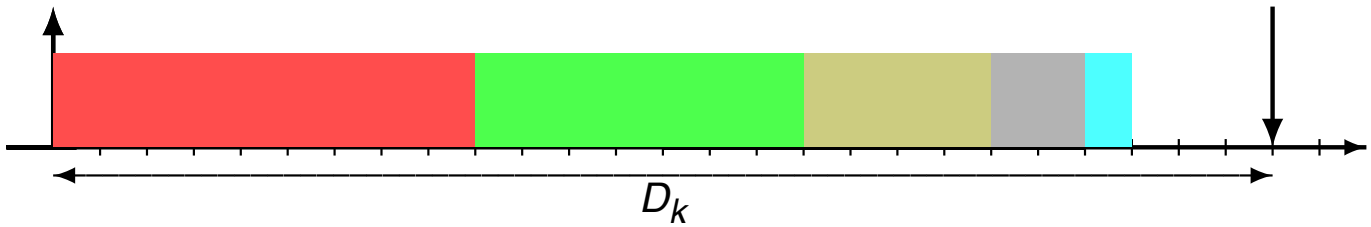
Response Time Analysis (RTA)

- The response time of a task τ_k is the duration it needs to complete its worst-case execution C_k



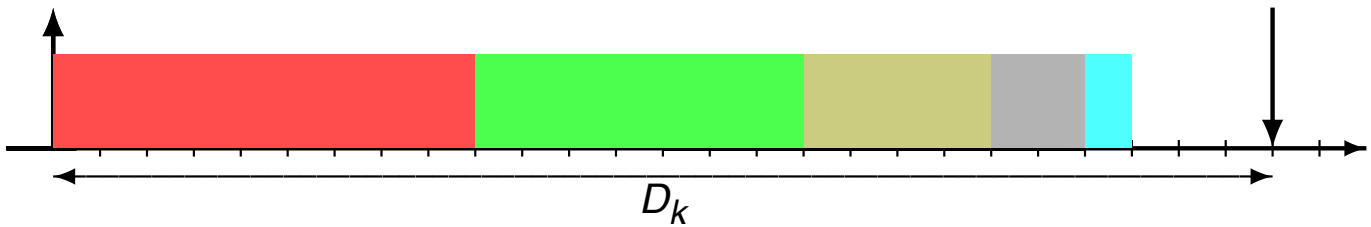
Response Time Analysis (RTA)

- The response time of a task τ_k is the duration it needs to complete its worst-case execution C_k



Response Time Analysis (RTA)

- The response time of a task τ_k is the duration it needs to complete its worst-case execution C_k



- RTA tests
  - e.g. [Guan et al@RTSS09], [Sun et al@RTCSA14]

$$X \leftarrow \left\lfloor \frac{\Omega_k(X)}{m} \right\rfloor + C_k$$

RTA (cont.)

- τ_k can complete its execution within a time interval of length x if

$$\left\lfloor \frac{\Omega_k(x)}{m} \right\rfloor + C_k \leq x$$

RTA (cont.)

- τ_k can complete its execution within a time interval of length x if

$$\left\lfloor \frac{\Omega_k(x)}{m} \right\rfloor + C_k \leq x$$

- Given a task set that is schedulable by the previous RTA, the schedulability may NOT be preserved under the same test after scaling task parameters.

- hint: $\left\lfloor \frac{15}{2} \right\rfloor = 7$, $\left\lfloor \frac{15 \cdot 10}{2} \right\rfloor = 75 > 7 \cdot 10$

Suggestions

τ_k can complete its execution within a time interval of length x if

$$\frac{\Omega_k(x)}{m} + C_k \leq x$$

IRTA (Integer Response Time Analysis)

$$X \leftarrow \left\lceil \frac{\Omega_k(X)}{m} \right\rceil + C_k$$

Thank you!