Schedulability and Verification of Real-Time Discrete-Event Systems

Christos Stergiou

University of California, Berkeley

September 16, 2013
Cyber-Physical System (CPS):
Orchestrating networked computational resources with physical systems
Motivation

- Programming CPS is flawed
- Timing affects behavior & correctness
- Insufficient software abstractions
- Lack of temporal semantics
- Thesis: time has to be a first-class citizen in CPS programming
PTIDES

Discrete-Event Systems

Real-time semantics

PTIDES

Platform

Schedulability Analysis
PTIDES

Discrete-Event Systems

Verification

Real-time semantics

PTIDES

Platform

Schedulability Analysis
Discrete-Event Systems

Clock

\(D_1\) \quad \(D_2\)

\(C\)

Display
Discrete-Event Systems

$t_0 + P, v$
Discrete-Event Systems

Clock

\[ (t_0 + 2P, v) \]

\[ D_1 \]

\[ D_2 \]

\[ C \]

Display
Discrete-Event Systems

Clock: \((t_0 + 3P, v)\)

\[ D_1 \]

\[ D_2 \]

\[ C \]

Display
Discrete-Event Systems

\[ e(t_0, \nu) \]

Clock

\[ D_1 \]

\[ D_2 \]

\[ C \]

Display
Discrete-Event Systems

\[ e(t_1 = t_0 + D_1, v) \]
Discrete-Event Systems

\[ e(t_1 = t_0 + D_1, v) \]

\[ e'(t_2, v') \]
Discrete-Event Systems

Clock

$D_1$

$D_2$

$C$

Display

$e\left(t_1, v''\right)$

$e\left(t_2, v'\right)$
PTIDES: Sensors and Actuators

Ptides Platform

Sensor → $D$ → $C$ → Actuator

real-time $t$
PTIDES: Sensors and Actuators

Ptides Platform

Sensor $\rightarrow D \rightarrow C \rightarrow$ Actuator

$t_1$ Sensor fires

real-time $t$
Ptides Platform

$e_1(t_1)$

Sensor $\rightarrow D \rightarrow C \rightarrow$ Actuator

t_1

Sensor fires

real-time $t$
PTIDES: Sensors and Actuators

Ptides Platform

Sensor $D$ $e_2(t_1 + D)$ $C$ Actuator

$t_1$ Sensor fires

real-time $t$
PTIDES: Sensors and Actuators
PTIDES: Sensors and Actuators

Ptides Platform

Sensor \rightarrow D \rightarrow C \rightarrow \text{Actuator}

e_3(t_1 + D)

Sensor fires at \( t_1 \) and \( t_2 \) is the time of actuation.
PTIDES: Sensors and Actuators

Ptides Platform

Sensor → $D$ → $C$ → Actuator

e_3(t_1 + D)

$t_1$ Sensor fires
$t_2$ Actuator actuates
$t_1 + D$ real-time $t$
If $t_2 > t_1 + D$ then $e_3$ misses its *deadline*
When should $e$ be processed?

- After $t_1 + D_1$ any event that arrives at $S_2$ will have timestamp $> t_1 + D_1$
- Safe-to-process analysis
PTIDES: Scheduling

\[ \text{deadline}(e_2) = t_2 \]
\[ \text{deadline}(e_3) = t_3 + D_3 \]

\( e_1(t_1) \)
\( e_2(t_2) \)
\( e_3(t_3) \)
PTIDES: Scheduling

\[ e_1(t_1) \rightarrow C_1 \rightarrow e_2(t_2) \]

\[ e_3(t_3) \rightarrow C_3 \rightarrow D_3 \]

\[ \text{deadline}(e_2) = t_2 \]
\[ \text{deadline}(e_3) = t_3 + D_3 \]
PTIDES: Scheduling

- $S_1 \rightarrow D_1 \rightarrow C_1 \rightarrow e_1(t_1) \rightarrow e_2(t_2) \rightarrow C_2 \rightarrow A_1$
- $S_2 \rightarrow D_2 \rightarrow C_1 \rightarrow e_2(t_2) \rightarrow C_2 \rightarrow A_2$
- $S_3 \rightarrow e_3(t_3) \rightarrow C_3 \rightarrow D_3 \rightarrow A_3$

- deadline($e_2$) = $t_2$
- deadline($e_3$) = $t_3 + D_3$
- deadline($e$) = $t +$ (delay to actuators)
PTIDES: Scheduling

- $\text{S1} \to \text{D}_1 \to \text{C}_1 \to \text{C}_2 \to \text{A}_1$
- $\text{S2} \to \text{D}_2 \to \text{C}_1 \to \text{C}_2 \to \text{A}_2$
- $\text{S3} \to \text{e}_3(t_3) \to \text{C}_3 \to \text{D}_3 \to \text{A}_3$

- deadline($\text{e}_2$) = $t_2$
- deadline($\text{e}_3$) = $t_3 + D_3$
- deadline($\text{e}$) = $t + (\text{delay to actuators})$
- EDF with preemption
Schedulability Problem

- Worst-case execution time per actor
- Models for sensor and network inputs
  - Periodic, sporadic (min. inter-arrival time)
- Schedulability problem: 
  *Does the program always meet its deadlines?*
Challenges

- Difficult to identify worst-case scenario
- Two dependent objectives:
  - Processor demand
  - Safe-to-process waiting
- Expressiveness of programming model
Our Approach

- Address infinite state space
  - Real-time and timestamps
  - Number of events
- Reduce schedulability to reachability in timed automata
  - Implement DE semantics
  - Simulate EDF with preemption
Real-time & Timestamps

- Real-time and timestamps can grow without bound
- Their absolute value is not necessary for execution
- Difference between timestamp and real-time is sufficient for PTIDES semantics
  - Discrete-event semantics and safe-to-process
  - EDF scheduling, i.e., compare deadlines
  - Deadline misses
Relative Timestamps

- Relative timestamp, timestamp - real-time: \( \tau - t \)
- Starts at 0
- Decreases continuously as real-time advances
- Makes discrete jumps when an event is processed by delay actor
- Has to be \( \geq 0 \) when an event reaches an actuator
Bounding relative timestamps

- Find $L, U$ such that $L \leq \tau - t \leq U$
Bounding relative timestamps

- Find \( L, U \) such that \( L \leq \tau - t \leq U \)
- \( L \leq \tau - t \)
  - Can real-time \( t \) grow unboundedly relative to a timestamp \( \tau \)?

Can a timestamp grow unboundedly relative to real-time?

No: \( t + (\text{delay from sensors}) \) or else we could violate timestamp order.
Bounding relative timestamps

- Find $L, U$ such that $L \leq \tau - t \leq U$
- $L \leq \tau - t$
  - Can real-time $t$ grow unboundedly relative to a timestamp $\tau$?
  - No: $t \leq \tau + (\text{delay to actuators})$
    or else deadline miss
Bounding relative timestamps

- Find $L, U$ such that $L \leq \tau - t \leq U$
- $L \leq \tau - t$
  - Can real-time $t$ grow unboundedly relative to a timestamp $\tau$?
  - No: $t \leq \tau +$ (delay to actuators) or else deadline miss
- $\tau - t \leq U$
  - Can a timestamp grow unboundedly relative to real-time?
Bounding relative timestamps

- Find $L, U$ such that $L \leq \tau - t \leq U$
- $L \leq \tau - t$
  - Can real-time $t$ grow unboundedly relative to a timestamp $\tau$?
  - No: $t \leq \tau + (\text{delay to actuators})$
or else deadline miss
- $\tau - t \leq U$
  - Can a timestamp grow unboundedly relative to
    real-time?
  - No: $\tau \leq t + (\text{delay from sensors})$
or else we could violate timestamp order
Bounding relative timestamps

- Find $L, U$ such that $L \leq \tau - t \leq U$
- $L \leq \tau - t$
  - Can real-time $t$ grow unboundedly relative to a timestamp $\tau$?
  - No: $t \leq \tau + (\text{delay to actuators})$ or else deadline miss
- $\tau - t \leq U$
  - Can a timestamp grow unboundedly relative to real-time?
  - No: $\tau \leq t + (\text{delay from sensors})$ or else we could violate timestamp order
- $-(\text{delay to actuators}) \leq \tau - t \leq (\text{delay from sensors})$
Queue-size bounds (1)

- How big can $N$ be?
- If a request arrives at $t$, its deadline is $t + D$
- Total execution time of $N$ events is $N \cdot W$
- $N \leq \left\lceil \frac{D}{W} \right\rceil$
Queue-size bounds (1)

- How big can $N$ be?
- If a request arrives at $t$, its deadline is $t + D$
- Total execution time of $N$ events is $N \cdot W$
- $N \leq \left\lceil \frac{D}{W} \right\rceil$

![Diagram]

Sensor $\rightarrow D \rightarrow \ldots \rightarrow C \rightarrow$ Actuator

Comp-time: $W$
Queue-size bounds (2)

- Absolute deadline of event with timestamp $\tau$:
  $$\tau + \text{(delay to actuators)}$$
Absolute deadline of event with timestamp $\tau$:

$$\tau + \text{(delay to actuators)}$$

Relative deadline associated with channel is:

$$\tau - t + \text{delay}(C_2, \text{actuators})$$
Queue-size bounds (2)

- Absolute deadline of event with timestamp $\tau$:
  $$\tau + \text{(delay to actuators)}$$

- Relative deadline associated with channel is:
  $$\tau - t + \text{delay}(C_2, \text{actuators})$$

- Upper bound on relative deadline
  $$(\text{delay from sensors}) + \text{(delay to actuators)}$$
Our Approach

- Address infinite state space
  - Real-time and timestamps
  - Number of events
- Reduce schedulability to reachability in timed automata
  - Implement DE semantics
  - Simulate EDF with preemption
Schedulability using Timed Automata
Timed Automata

- Finite automata + finite set of real-valued clocks
- Time elapses at locations
- Clocks can be reset on transitions
- Guards: clock constraints on transitions
- Invariants: clock constraints on locations
Periodic Source

clock $c$;
const period;

---

**guard:** $c = \text{period}$

**reset:** $c := 0$

**invariant:** $c \leq \text{period}$
TA example

Sporadic

Periodic Source

clock c;
const period;

Guard: \( c \neq \text{period} \)
Reset: \( c := 0 \)

Invariant: \( c \leq \text{period} \)
Timestamps with clocks (1)

- Associate a clock $c$ and a discrete variable $d$ with each event
- Reset clock when event enters platform
- $c$ measures the relative time in the platform: $t - c =$ time the event entered platform
- $d$ accumulates the delay added to the event: $t - c + d =$ timestamp of event
Timestamps with clocks (2)

Ptides Platform

Sensor $\rightarrow D \rightarrow C \rightarrow$ Actuator

real-time $t$
Timestamps with clocks (2)

Ptides Platform

Sensor $\rightarrow D \rightarrow C \rightarrow$ Actuator

Sensor fires $t_1$

real-time $t$
Timestamps with clocks (2)

Ptides Platform

\[ e_1(c, d) \]

Sensor \( \rightarrow D \rightarrow C \rightarrow \) Actuator

\[ t_1 \]

Sensor fires

\[ c := 0, d := 0 \]
Timestamps with clocks (2)

Ptides Platform

Sensor \rightarrow D \rightarrow C \rightarrow \text{Actuator}

\[ e_2(c, d) \]

Sensor fires

\[ c := 0, \ d := 0 \]

\[ d := d + D \]

real-time \( t \)

\( t_1 \)
Timestamps with clocks (2)

Ptides Platform

Sensor $\rightarrow D \rightarrow C \rightarrow$ Actuator

c := 0, d := 0

d := d + D
Timestamps with clocks (2)

Ptides Platform

Sensor $\rightarrow D \rightarrow C \rightarrow$ Actuator

$e_3(c, d)$

real-time $t$

$t_1 \rightarrow t_2$

Sensor fires

$c := 0, d := 0$

$d := d + D$
Ptides Platform

Sensor → $D$ → $C$ → Actuator

e_3(c, d)

c := 0, d := 0

d := d + D

t_1

Sensor fires

t_2

$t_1 + D$

Actuator actuates

c = d

real-time $t$
Schedulability using Timed Automata

Input Automata
one per sensor

Task Automata
one per actor in the model

Scheduler Automaton
one per platform
Task Automaton

idle

executing

clock := 0

clock = W

clock \leq W
Task Automaton

idle

clock := 0

clock = W

eexecuting

clock ≤ W

pause clock?

preempted
Task Automaton

idle

clock := 0

clock = W

eexecuting

clock ≤ W

pause clock?
P(i)++

preempted
Task Automaton

idle

clock := 0

clock = W + \sum W_i \cdot P(i)

executing

clock = W + \sum W_i \cdot P(i)

pause clock?

P(i)++

preempted

clock \leq W + \sum W_i \cdot P(i)
Schedulability using Timed Automata

Input Automata
one per sensor

Task Automata
one per actor in the model

Scheduler Automaton
one per platform
Schedulability using Timed Automata

Input Automata

Task Automata

Scheduler Automaton

error state

E<> (Scheduler.deadline miss state)
Our Approach

- Address infinite state space
  - Real-time and timestamps
  - Number of events
- Reduce schedulability to reachability in timed automata
  - Implement DE semantics
  - Simulate EDF with preemption
Can we leverage traditional hard real-time theory for more efficient and sufficient schedulability tests?
Can we leverage traditional hard real-time theory for more efficient and sufficient schedulability tests?

We described two dependent objectives:

- Processor demand
- Safe-to-process waiting
Can we leverage traditional hard real-time theory for more efficient and sufficient schedulability tests?

We described two dependent objectives:
- Processor demand
- Safe-to-process waiting

We will try to factor the latter in the real-time task system
Periodic and Sporadic Task Systems

- Periodic Task

\[ P \]
\[ W, D \]

- Sporadic Task

\[ W \]
\[ D \]

execution time

absolute deadline

0
Periodic and Sporadic Task Systems

- Periodic Task
  - Execution time: $W$
  - Absolute deadline: $D$
  - Start time: $0$
  - Period: $P$
  - Completion time: $P + D$

- Sporadic Task
  - Execution time: $W$
  - Absolute deadline: $D$
  - Start time: $0$
  - Period: $P$
Periodic and Sporadic Task Systems

- **Periodic Task**
  - $P$
  - Execution time: $W$
  - Absolute deadline: $D$
  - $W + D$

- **Sporadic Task**
  - $I$
  - Execution time: $W$
  - Absolute deadline: $D$
  - $0 + D$

Diagram:
- Periodic Task:
  - $P$:
  - $W, D$
  - Execution time $W$
  - Absolute deadline $D$
  - $0 + D$

- Sporadic Task:
  - $I$:
  - $W, D$
  - Execution time $W$
  - Absolute deadline $D$
  - $0 + D$
Periodic and Sporadic Task Systems

- **Periodic Task**

- **Sporadic Task**

![Diagram showing execution time and absolute deadline for periodic and sporadic tasks.](image-url)
Periodic and Sporadic Task Systems

- **Periodic Task**
  - Execution time: \( P \)
  - Absolute deadline: \( W \)
  - Period: \( P + D \)

- **Sporadic Task**
  - Execution time: \( I \)
  - Absolute deadline: \( W \)
  - Time: \( t \geq l \)
Multiframe Tasks

\[ I \]

\[ W_0, D_0 \xrightarrow{S_1} W_1, D_1 \xrightarrow{S_2} \ldots \xrightarrow{S_{N-1}} W_{N-1} \]

\[ D_{N-1} \]
Multiframe Tasks

$W_0, D_0 \xrightarrow{S_1} W_1, D_1 \xrightarrow{S_2} \ldots \xrightarrow{S_{N-1}} W_{N-1}, D_{N-1}$

$W_0 \quad D_0 \quad W_1 \quad S_1 + D_1$

$0 \quad S_1$
Multiframe Tasks

I

$W_0, D_0 \xrightarrow{S_1} W_1, D_1 \xrightarrow{S_2} \ldots \xrightarrow{S_{N-1}} W_{N-1}$

$W_0 \quad D_0 \quad W_1 \quad S_1 + D_1 \quad W_2 \quad S_1 + S_2 + D_2 \quad \ldots$

0 \quad S_1 \quad S_1 + S_2
Multiframe Tasks

\[ \begin{align*}
&W_0, D_0 \\
&\rightarrow S_1 \\
&W_1, D_1 \\
&\rightarrow S_2 \\
&\rightarrow \ldots \\
&\rightarrow S_{N-1} \\
&D_{N-1}
\end{align*} \]
Multiframe Tasks

\[ I \]

\[ W_0, D_0 \xrightarrow{S_1} W_1, D_1 \xrightarrow{S_2} \ldots \xrightarrow{S_{N-1}} W_{N-1}, D_{N-1} \]

\[ W_0 \quad D_0 \quad W_1 \quad S_1 + D_1 \quad W_2 \quad S_1 + S_2 + D_2 \]

\[ \begin{align*}
0 & \quad W_0 & \quad t + D_0 & \quad W_1 & \quad t + S_1 + D_1 & \quad \ldots \\
& \quad W_0 & \quad t + D_0 & \quad W_1 & \quad t + S_1 + D_1 & \quad \ldots \\
& \quad t \geq I & \quad t + S_1 & \quad & \quad & \quad
\end{align*} \]
PTIDES as multiframe tasks

- Reduction to tasks is easy for parallel chains of actors
- What about merging and splitting paths?
Merging as multiframe tasks

Event $e$ is safe to process at $t' \geq t + D_1$
Merging as multiframe tasks

\[ e(t + D_1) \]

- Event \( e \) is safe to process at \( t' + D_1 \).
- If it is not safe at \( t + D_1 \), \( W_2 \) is executing an event with smaller deadline than \( e \).
- Under EDF, \( task \) can be released at \( t + D_1 \).

\[ S_1 \rightarrow D_1 \rightarrow W_1 \rightarrow e(t + D_1) \]
\[ S_2 \rightarrow W_2 \]
\[ D_3 \rightarrow W_3 \rightarrow A_1 \]
Merging as multiframe tasks

$S_1 \rightarrow D_1 \rightarrow W_1 \xrightarrow{e(t+D_1)} D_3 \rightarrow W_3 \rightarrow A_1$

$S_2 \rightarrow W_2$

$W_1 \xrightarrow{D_1+D_3} W_3 \xrightarrow{D_1} W_2 + W_3$

$I_1$

$I_2$
Schedulability as multiframe tasks

- Statically compute lower bounds for release time of tasks
- Reduce to schedulability of multiframe tasks:
  - EDF
  - Sporadic input sources
  - Input-agnostic safe-to-process analysis
PTIDES

Discrete-Event Systems

Verification

PTIDES

Schedulability Analysis

Platform
Example DE
Example DE
Example DE

t: 0 → 2
Example DE

\[ t: 0 \rightarrow 2 \]
Example DE

\[
t: 0 \rightarrow 2 \rightarrow 3
\]
Example DE

\[ t: \ 0 \rightarrow 2 \rightarrow 3 \]
Example DE

\[ t: 0 \rightarrow 2 \rightarrow 3 \rightarrow 4 \]
Example DE

$t: 0 \rightarrow 2 \rightarrow 3 \rightarrow 4$
Example DE

t: 0 → 2 → 3 → 4 → 6
Timed transition system

- **State** \((r, t)\)
  - \(r\) is a map from channels to sets of timestampes
  - \(t\) is global time

- **Two types of transitions**
  - *delay* transitions: set global time equal to the smallest timestamp in the map
  - *discrete* transitions: fire actor with smallest timestamp in its input channels

- Actors fire in timestamp order
- Execution is deterministic
Boundedness of DE

Events in $c_1$: $\{i \cdot P \mid i \in \mathbb{N}\}$

Events in $c_2$: $\{(i + 1) \cdot P \mid i \in \mathbb{N}\}$
Boundedness of DE

Events in $c_1$: $\{i \cdot P \mid i \in \mathbb{N}\}$

Events in $c_2$: $\{(i + 1) \cdot P \mid i \in \mathbb{N}\}$

Is the number of events in any channel at a state of the TTS bounded?

Can we address the issue of timestamps and time being infinite?
Bounding number of events

- Let $\tau_{\text{min}}(s)$ be the min. timestamp in state $s$

1 initial events
Bounding number of events

- Let $\tau_{\text{min}}(s)$ be the min. timestamp in state $s$
- if $s \rightarrow s'$ then $\tau_{\text{min}}(s) \leq \tau_{\text{min}}(s')$

\(^1\)initial events
Bounding number of events

- Let $\tau_{\text{min}}(s)$ be the min. timestamp in state $s$
- if $s \rightarrow s'$ then $\tau_{\text{min}}(s) \leq \tau_{\text{min}}(s')$
- for any event $\tau$ in a state $s$,\(^1\)

$$\tau \leq \tau_{\text{min}}(s) + \max\{\text{delay}(a) \mid a \in A\}$$

\(^1\)initial events
Bounding number of events

- Let $\tau_{\text{min}}(s)$ be the min. timestamp in state $s$
- if $s \rightarrow s'$ then $\tau_{\text{min}}(s) \leq \tau_{\text{min}}(s')$
- for any event $\tau$ in a state $s$,$^1$

\[ \tau \leq \tau_{\text{min}}(s) + \max\{\text{delay}(a) \mid a \in A\} \]

- Lower and upper bound for all events in a state
- Fractional part of every timestamp is determined by fractional part of initial events

$^1$initial events
Bounding timestamps

- In TTS timestamps and global time can grow unbounded
Bounding timestamps

- In TTS timestamps and global time can grow unbounded
- Bounded timed transition system $BTS(G, r_0)$
- Delay transitions: subtract minimum timestamp from all events
- Discrete transitions: process event with timestamp equal to 0

$BTS$ transitions:

- delay transition: $r \xrightarrow{\delta} b r'$ where $\delta = \tau_{\text{min}}(r)$, $r' = r - \tau_{\text{min}}(r)$
- discrete transition: $r \xrightarrow{a} b r'$ with $r' = f(a, r, D(a)), \tau_{\text{min}}(a, r) = \tau_{\text{min}}(r) = 0$
Example BTS
Example BTS
Example BTS
Example BTS

![Diagram of Example BTS](image-url)
Example BTS

1

2

3

1

2
Example BTS
Example BTS
Bounded Transition System

- The set of reachable states of $BTS(G, r_0)$ is finite
  - Number of events bounded as in $TTS$
  - Possible timestamps finite, despite initial events $\in \mathbb{R}$
- A bisimulation exists between $TTS$ and $BTS$
  - $R$ contains all pairs $((r, t), r - t)$
Verification - Queries

- Signal queries
  - A channel signal denotes the set of all events that occur in a channel along an execution
  - “an event occurs in \( c \)”, \( \phi := \exists \tau : \tau \geq 0 \)
  - “two events occur in \( c \) separated by at most 1 time unit”, \( \exists \tau_1, \tau_2 : |\tau_1 - \tau_2| \leq 1 \).
Verification - Queries

- **Signal queries**
  - A channel signal denotes the set of all events that occur in a channel along an execution
  - “an event occurs in \( c \)”, \( \phi := \exists \tau : \tau \geq 0 \)
  - “two events occur in \( c \) separated by at most 1 time unit”, \( \exists \tau_1, \tau_2 : |\tau_1 - \tau_2| \leq 1 \).

- **State queries**
Verification - Algorithms for DE

- Construct lasso from BTS
  - Merge all enabled transitions in one
  - Finite and deterministic transition system

Ane expression:

\[ 1 = i_1 P + j_1 D \]
\[ 2 = i_2 P + j_2 D \]

\[ 1 + 2 = 5 \]
Verification - Algorithms for DE

- Construct lasso from BTS
  - Merge all enabled transitions in one
  - Finite and deterministic transition system
- Compute for every channel $c$, an affine expression that describes channel signal of $c$
Verification - Algorithms for DE

- Construct lasso from BTS
  - Merge all enabled transitions in one
  - Finite and deterministic transition system

- Compute for every channel \( c \), an affine expression that describes channel signal of \( c \)

- Reduce the problem of checking whether \( \sigma_c \models \phi \) to an SMT problem
  - Affine expression: \( i \cdot P + j \cdot D \)
  - \( \tau_1, \tau_2 \) such that \( \tau_1 - \tau_2 = 5 \)
  - \( \tau_1 = i_1P + j_1D \land \tau_2 = i_2P + j_2D \land \tau_1 - \tau_2 = 5 \)
Verification - Algorithms for DE

- Construct lasso from BTS
  - Merge all enabled transitions in one
  - Finite and deterministic transition system

- Compute for every channel $c$, an affine expression that describes channel signal of $c$

- Reduce the problem of checking whether $\sigma_c \models \phi$ to an SMT problem
  - Affine expression: $i \cdot P + j \cdot D$
  - $\tau_1, \tau_2$ such that $\tau_1 - \tau_2 = 5$
  - $\tau_1 = i_1 P + j_1 D \land \tau_2 = i_2 P + j_2 D \land \tau_1 - \tau_2 = 5$

- Similarly for state queries
Conclusions

Discrete-Event Systems

PTIDES

Verification

Platform

Schedulability Analysis
Thank you

Questions?