Monitoring and Fault-Diagnosis with Digital Clocks

Karine Altisen[1] Franck Cassez[2] Stavros Tripakis[1]

[1]VERIMAG
Grenoble, France

[2]IRCCyN
Nantes, France

ACSD'06
June 27th, 2006

Turku, Finland

Motivation & Context

Monitoring

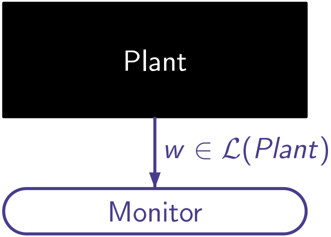
Plant generates $\mathcal{L}(Plant) \subseteq \Sigma^*$

Specification $= \mathcal{L}(S) \subseteq \Sigma^*$

Diagnosis

Plant generates $\mathcal{L}(Plant) \subseteq (\Sigma \cup \{\varepsilon, f\})^*$

Spec. $= \mathcal{L}(S) = \{\rho.f.\rho' \text{ s.t. } |\rho'| \geq k\}$



Role of the monitor:

- can shout when $w \notin \mathcal{L}(S)$
- never shout when $w \in \mathcal{L}(S)$

Role of the k-diagnoser:

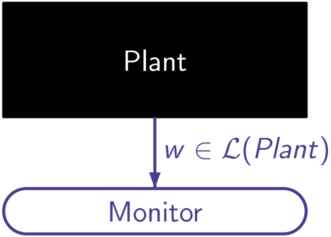
- must shout when $w \in \mathcal{L}(S)$
- never shout when no f in w

Motivation & Context

Monitoring

Plant generates $\mathcal{L}(Plant) \subseteq \Sigma^*$

Specification $= \mathcal{L}(S) \subseteq \Sigma^*$



Role of the monitor:

- can shout when $w \notin \mathcal{L}(S)$
- never shout when $w \in \mathcal{L}(S)$

Diagnosis

Plant generates $\mathcal{L}(Plant) \subseteq (\Sigma \cup \{\varepsilon, f\})^*$

Spec. $= \mathcal{L}(S) = \{\rho.f.\rho' \text{ s.t. } |\rho'| \geq k\}$

Role of the k-diagnoser:

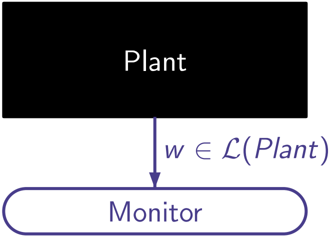
- must shout when $w \in \mathcal{L}(S)$
- never shout when no f in w

Motivation & Context

Monitoring

Plant generates $\mathcal{L}(Plant) \subseteq \Sigma^*$

Specification $= \mathcal{L}(S) \subseteq \Sigma^*$



Role of the monitor:

- can shout when $w \notin \mathcal{L}(S)$
- never shout when $w \in \mathcal{L}(S)$

Diagnosis

Plant generates $\mathcal{L}(Plant) \subseteq (\Sigma \cup \{\varepsilon, f\})^*$

Spec. $= \mathcal{L}(S) = \{\rho.f.\rho' \text{ s.t. } |\rho'| \geq k\}$

Role of the k-diagnoser:

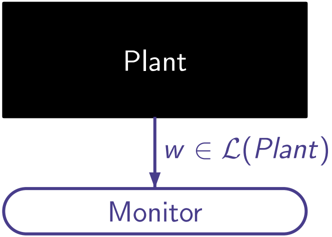
- must shout when $w \in \mathcal{L}(S)$
- never shout when no f in w

Motivation & Context

Monitoring

Plant generates $\mathcal{L}(Plant) \subseteq \Sigma^*$

Specification $= \mathcal{L}(S) \subseteq \Sigma^*$



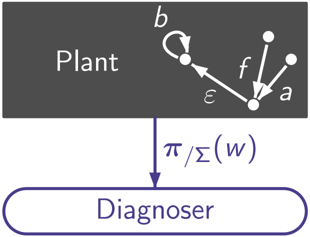
Role of the monitor:

- can shout when $w \notin \mathcal{L}(S)$
- never shout when $w \in \mathcal{L}(S)$

Diagnosis

Plant generates $\mathcal{L}(Plant) \subseteq (\Sigma \cup \{\varepsilon, f\})^*$

Spec. $= \mathcal{L}(S) = \{\rho.f.\rho' \text{ s.t. } |\rho'| \geq k\}$



Role of the k-diagnoser:

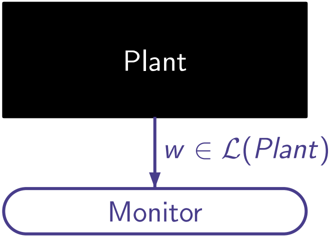
- must shout when $w \in \mathcal{L}(S)$
- never shout when no f in w

Motivation & Context

Monitoring

Plant generates $\mathcal{L}(Plant) \subseteq \Sigma^*$

Specification $= \mathcal{L}(S) \subseteq \Sigma^*$



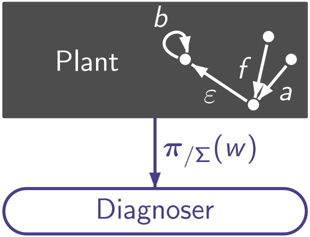
Role of the monitor:

- can shout when $w \notin \mathcal{L}(S)$
- never shout when $w \in \mathcal{L}(S)$

Diagnosis

Plant generates $\mathcal{L}(Plant) \subseteq (\Sigma \cup \{\varepsilon, f\})^*$

Spec. $= \mathcal{L}(S) = \{\rho.f.\rho' \text{ s.t. } |\rho'| \geq k\}$



Role of the k-diagnoser:

- must shout when $w \in \mathcal{L}(S)$
- never shout when no f in w

Known Results & Related Work

- Discrete Events Systems [Sampath et al., IEEE'95] — Finite Automata
  - Monitoring ≡ determinize the specification
  - Diagnosis
    1. Check diagnosability (PTIME)
    2. Compute a diagnoser (EXPTIME)
- Dense-time Systems — Timed Automata
  - Monitoring
    TA are not determinizable – Checking determinizability is undecidable
    On-the-fly solutions [Krichen, Tripakis, FORMATS'04]
  - Diagnosis
    1. Diagnoser ≡ Turing Machine [Tripakis, FTRTFT'02]
    2. TA-Diagnosability: the diagnoser is a deterministic/Event-Recording timed automaton [Bouyer et al., FoSSaCS'05]

Known Results & Related Work

- Discrete Events Systems [Sampath et al., IEEE'95] — Finite Automata
  - Monitoring ≡ determinize the specification
  - Diagnosis
    1. Check diagnosability (PTIME)
    2. Compute a diagnoser (EXPTIME)
- Dense-time Systems — Timed Automata
  - Monitoring
    TA are not determinizable – Checking determinizability is undecidable
    On-the-fly solutions [Krichen, Tripakis, FORMATS'04]
  - Diagnosis
    1. Diagnoser ≡ Turing Machine [Tripakis, FTRTFT'02]
    2. TA-Diagnosability: the diagnoser is a deterministic/Event-Recording timed automaton [Bouyer et al., FoSSaCS'05]

Known Results & Related Work

- Discrete Events Systems [Sampath et al., IEEE'95] — Finite Automata
  - Monitoring ≡ determinize the specification
  - Diagnosis
    1. Check diagnosability (PTIME)
    2. Compute a diagnoser (EXPTIME)
- Dense-time Systems — Timed Automata
  - Monitoring
    TA are not determinizable – Checking determinizability is undecidable
    On-the-fly solutions [Krichen, Tripakis, FORMATS'04]
  - Diagnosis
    1. Diagnoser ≡ Turing Machine [Tripakis, FTRTFT'02]
    2. TA-Diagnosability: the diagnoser is a deterministic/Event-Recording timed automaton [Bouyer et al., FoSSaCS'05]

Known Results & Related Work

- Discrete Events Systems [Sampath et al., IEEE'95] — Finite Automata
  - Monitoring ≡ determinize the specification
  - Diagnosis
    1. Check diagnosability (PTIME)
    2. Compute a diagnoser (EXPTIME)
- Dense-time Systems — Timed Automata
  - Monitoring
    TA are not determinizable – Checking determinizability is undecidable
    On-the-fly solutions [Krichen, Tripakis, FORMATS'04]
  - Diagnosis
    1. Diagnoser ≡ Turing Machine [Tripakis, FTRTFT'02]
       Checking Diagnosability PSPACE
    2. TA-Diagnosability: the diagnoser is a deterministic/Event-Recording timed automaton [Bouyer et al., FoSSaCS'05]
       building the Diagnoser: 2EXPTIME-complete/PSPACE-complete

Known Results & Related Work

- Discrete Events Systems [Sampath et al., IEEE'95] Finite Automata
  - Monitoring ≡ determinize the specification
  - Diagnosis
    1. Check diagnosability (PTIME)
    2. Compute a diagnoser (EXPTIME)
- Dense-time Systems Timed Automata
  - Monitoring
    TA are not determinizable – Checking determinizability is undecidable
    On-the-fly solutions [Krichen, Tripakis, FORMATS'04]
  - Diagnosis
    1. Diagnoser ≡ Turing Machine [Tripakis, FTRTFT'02]
       Checking Diagnosability PSPACE
    2. TA-Diagnosability: the diagnoser is a deterministic/Event-Recording timed automaton [Bouyer et al., FoSSaCS'05]
       building the Diagnoser: 2EXPTIME-complete/PSPACE-complete

Known Results & Related Work

- Discrete Events Systems [Sampath et al., IEEE'95] — Finite Automata
  - Monitoring ≡ determinize the specification
  - Diagnosis
    1. Check diagnosability (PTIME)
    2. Compute a diagnoser (EXPTIME)
- Dense-time Systems — Timed Automata
  - Monitoring
    TA are not determinizable – Checking determinizability is undecidable
    On-the-fly solutions [Krichen, Tripakis, FORMATS'04]
  - Diagnosis
    1. Diagnoser ≡ Turing Machine [Tripakis, FTRTFT'02]
       Checking Diagnosability PSPACE
    2. TA-Diagnosability: the diagnoser is a deterministic/Event-Recording timed automaton [Bouyer et al., FoSSaCS'05]
       building the Diagnoser: 2EXPTIME-complete/PSPACE-complete

Known Results & Related Work

- Discrete Events Systems [Sampath et al., IEEE'95] Finite Automata
  - Monitoring ≡ determinize the specification
  - Diagnosis
    1. Check diagnosability (PTIME)
    2. Compute a diagnoser (EXPTIME)
- Dense-time Systems Timed Automata
  - Monitoring
    TA are not determinizable – Checking determinizability is undecidable
    On-the-fly solutions [Krichen, Tripakis, FORMATS'04]
  - Diagnosis
    1. Diagnoser ≡ Turing Machine [Tripakis, FTRTFT'02]
       Checking Diagnosability PSPACE
    2. TA-Diagnosability: the diagnoser is a deterministic/Event-Recording timed automaton [Bouyer et al., FoSSaCS'05]
       building the Diagnoser: 2EXPTIME-complete/PSPACE-complete

Use of Analog Clocks = arbitrarily precise

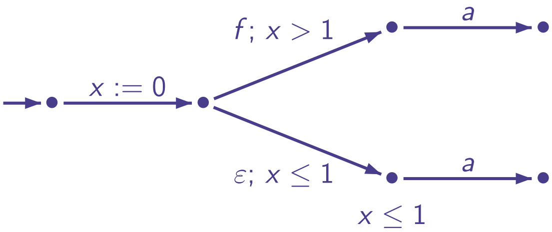
Known Results & Related Work

- Discrete Events Systems [Sampath et al., IEEE'95] — Finite Automata
  - Monitoring ≡ determinize the specification
  - Diagnosis
    1. Check diagnosability (PTIME)
    2. Compute a diagnoser (EXPTIME)
- Dense-time Systems — Timed Automata
  - Monitoring
    TA are not determinizable – Checking determinizability is undecidable
    On-the-fly solutions [Krichen, Tripakis, FORMATS'04]
  - Diagnosis
    1. Diagnoser ≡ Turing Machine [Tripakis, FTRTFT'02]
       Checking Diagnosability PSPACE
    2. TA-Diagnosability: the diagnoser is a deterministic/Event-Recording timed automaton [Bouyer et al., FoSSaCS'05]
       building the Diagnoser: 2EXPTIME-complete/PSPACE-complete

Our contribution: Monitoring & Fault Diagnosis with Digital Clocks

Perfect Clocks vs. Fuzzy Clocks

Digital Clocks cannot have arbitrary precision: imprecision Δ



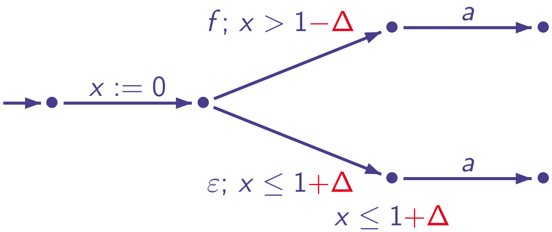
Perfect Clock t: if $a@t$ and $t > 1$ say "Fault" otherwise say nothing

Fuzzy Clocks: value of t is an interval $[t - \Delta, t + \Delta]$

$f@(1 + \frac{\Delta}{4}).a@(1 + \frac{\Delta}{3})$ and $\varepsilon@1.a@(1 + \frac{\Delta}{2})$ are indistinguishable

Perfect Clocks vs. Fuzzy Clocks

Digital Clocks cannot have arbitrary precision: imprecision Δ

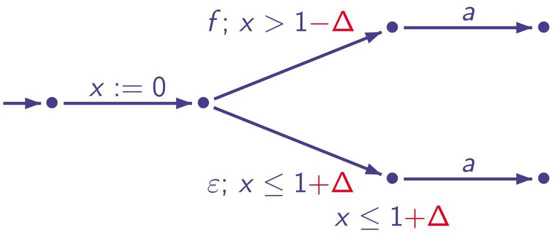


Perfect Clock t: if $a@t$ and $t > 1$ say "Fault" otherwise say nothing
Fuzzy Clocks: value of t is an interval $[t - \Delta, t + \Delta]$

$f@(1 + \frac{\Delta}{4}).a@(1 + \frac{\Delta}{3})$ and $\varepsilon@1.a@(1 + \frac{\Delta}{2})$ are indistinguishable

Perfect Clocks vs. Fuzzy Clocks

Digital Clocks cannot have arbitrary precision: imprecision Δ



Perfect Clock t: if $a@t$ and $t > 1$ say "Fault" otherwise say nothing

Fuzzy Clocks: value of t is an interval $[t - \Delta, t + \Delta]$

$f@(1 + \frac{\Delta}{4}).a@(1 + \frac{\Delta}{3})$ and $\varepsilon@1.a@(1 + \frac{\Delta}{2})$ are indistinguishable

Outline of the talk

- Models for Timed Systems & Digital Clocks
- Monitoring with Digital Clocks
- Diagnosis with Digital Clocks
- Conclusion & Open Problem

Outline of the talk

- Models for Timed Systems & Digital Clocks
- Monitoring with Digital Clocks
- Diagnosis with Digital Clocks
- Conclusion & Open Problem

Outline of the talk

- Models for Timed Systems & Digital Clocks
- Monitoring with Digital Clocks
- Diagnosis with Digital Clocks
- Conclusion & Open Problem

Outline of the talk

- Models for Timed Systems & Digital Clocks
- Monitoring with Digital Clocks
- Diagnosis with Digital Clocks
- Conclusion & Open Problem

Outline

- Models for Timed Systems & Digital Clocks
- Monitoring with Digital Clocks
- Diagnosis with Digital Clocks
- Conclusion & Open Problem

Timed Automata

[Alur & Dill, TCS'94]

- Timed Automaton = Finite Automaton + clock variables
  All clocks evolve at the same speed
  Clocks take their value in a dense-time domain
- Transitions are guarded by clocks constraints
- g: guard of the form $g ::= x \sim c \,|\, g \wedge g$
  where x is a clock and $c \in \mathbb{N}$, $\sim \in \{<, \leq, =, \geq, >\}$
- R : the set of clocks to be reset when firing the transition
- $\mathrm{Inv}(\ell)$ is an invariant to ensure "liveness"
- Semantics of TA: Timed Transition Systems

▸ TTS

Timed Automata

[Alur & Dill, TCS'94]

- Timed Automaton = Finite Automaton + clock variables
  All clocks evolve at the same speed
  Clocks take their value in a dense-time domain
- Transitions are guarded by clocks constraints

- g: guard of the form $g ::= x \sim c \,|\, g \wedge g$
  where x is a clock and $c \in \mathbb{N}$, $\sim \in \{<, \leq, =, \geq, >\}$
- R : the set of clocks to be reset when firing the transition
- Inv(ℓ) is an invariant to ensure "liveness"
- Semantics of TA: Timed Transition Systems

TTS

Timed Automata

[Alur & Dill, TCS'94]

- Timed Automaton = Finite Automaton + clock variables
  All clocks evolve at the same speed
  Clocks take their value in a dense-time domain
- Transitions are guarded by clocks constraints

- g: guard of the form $g ::= x \sim c \,|\, g \wedge g$
  where x is a clock and $c \in \mathbb{N}$, $\sim \in \{<, \leq, =, \geq, >\}$
- R : the set of clocks to be reset when firing the transition
- Inv(ℓ) is an invariant to ensure "liveness"
- Semantics of TA: Timed Transition Systems

TTS

Timed Automata

[Alur & Dill, TCS'94]

- Timed Automaton = Finite Automaton + clock variables
  All clocks evolve at the same speed

  Clocks take their value in a dense-time domain
- Transitions are guarded by clocks constraints

- g: guard of the form $g ::= x \sim c \mid g \wedge g$
  where x is a clock and $c \in \mathbb{N}$, $\sim \in \{<, \leq, =, \geq, >\}$
- R : the set of clocks to be reset when firing the transition
- $\mathrm{Inv}(\ell)$ is an invariant to ensure "liveness"
- Semantics of TA: Timed Transition Systems

▸ TTS

Timed Automata

[Alur & Dill, TCS'94]

- Timed Automaton = Finite Automaton + clock variables
  All clocks evolve at the same speed
  Clocks take their value in a dense-time domain
- Transitions are guarded by clocks constraints

- g: guard of the form $g ::= x \sim c \,|\, g \wedge g$
  where x is a clock and $c \in \mathbb{N}$, $\sim \in \{<, \leq, =, \geq, >\}$
- R : the set of clocks to be reset when firing the transition
- $\mathsf{Inv}(\ell)$ is an invariant to ensure "liveness"
- Semantics of TA: Timed Transition Systems

▸ TTS

Timed Automata

[Alur & Dill, TCS'94]

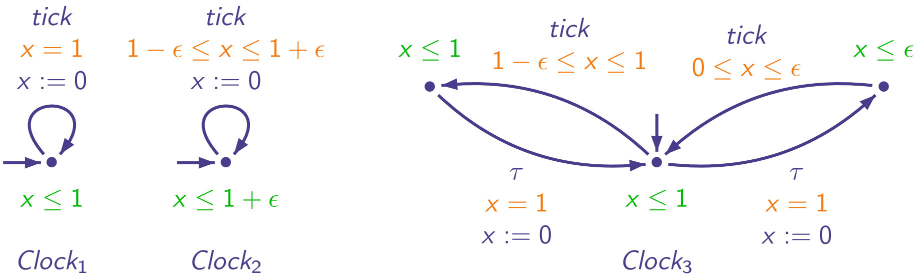
- Timed Automaton = Finite Automaton + clock variables
  All clocks evolve at the same speed

  Clocks take their value in a dense-time domain
- Transitions are guarded by clocks constraints

- g: guard of the form $g ::= x \sim c \,|\, g \wedge g$
  where x is a clock and $c \in \mathbb{N}$, $\sim \in \{<, \leq, =, \geq, >\}$
- R : the set of clocks to be reset when firing the transition
- Inv(ℓ) is an invariant to ensure "liveness"
- Semantics of TA: Timed Transition Systems

TTS

Digital-Clocks Automata



Timed Words:

$Clock_1$: $1.tick.1.tick.\cdots.1.tick.\cdots$

Let $\epsilon = 0,3$

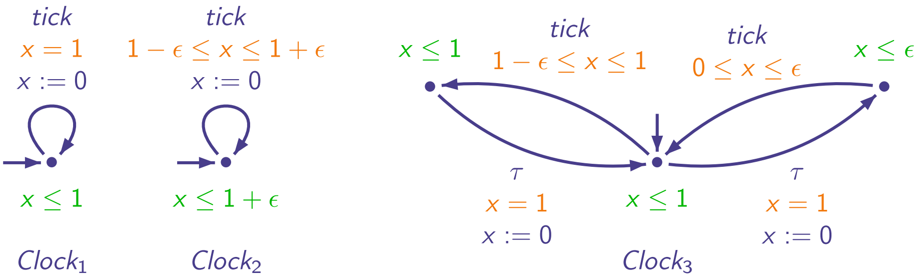
$Clock_2$: $0,8.tick.1,14.tick.\cdots.0,98.tick.\cdots$

n^{th} tick at t with $n \cdot (1 - 0,3) \leq t \leq n \cdot (1 + 0,3)$

$Clock_3$: $0,8.tick.1,3.tick.\cdots.1,15.tick.\cdots$

n^{th} tick at t with $n - 0,3 \leq t \leq n + 0,3$

Digital-Clocks Automata



Timed Words:

$Clock_1$: $1.tick.1.tick.\cdots.1.tick.\cdots$

Let $\epsilon = 0,3$

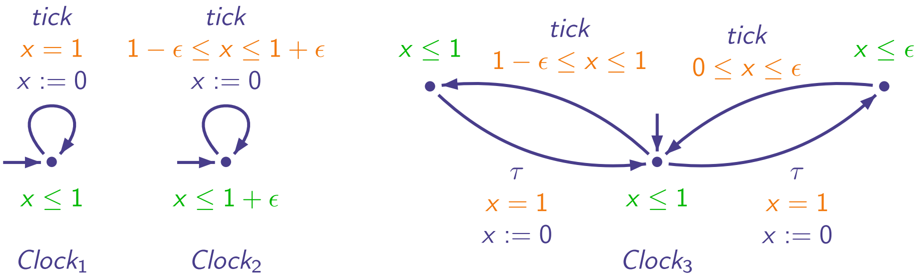
$Clock_2$: $0,8.tick.1,14.tick.\cdots.0,98.tick.\cdots$

n^{th} tick at t with $n \cdot (1 - 0,3) \leq t \leq n \cdot (1 + 0,3)$

$Clock_3$: $0,8.tick.1,3.tick.\cdots.1,15.tick.\cdots$

n^{th} tick at t with $n - 0,3 \leq t \leq n + 0,3$

Digital-Clocks Automata



Timed Words:

$Clock_1$: $1.tick.1.tick.\cdots.1.tick.\cdots$

Let $\epsilon = 0,3$

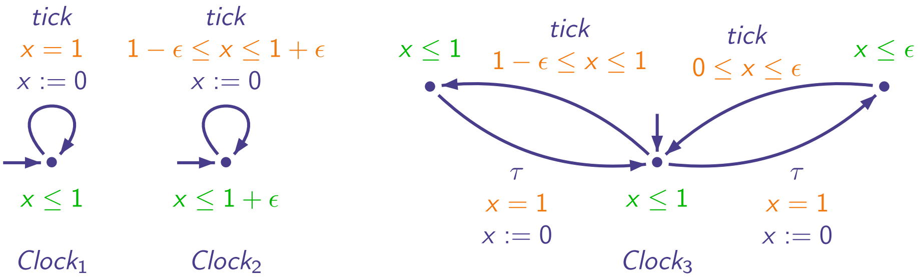
$Clock_2$: $0,8.tick.1,14.tick.\cdots.0,98.tick.\cdots$

n^{th} tick at t with $n \cdot (1 - 0,3) \leq t \leq n \cdot (1 + 0,3)$

$Clock_3$: $0,8.tick.1,3.tick.\cdots.1,15.tick.\cdots$

n^{th} tick at t with $n - 0,3 \leq t \leq n + 0,3$

Digital-Clocks Automata



Timed Words:

$Clock_1$: $1.tick.1.tick.\cdots.1.tick.\cdots$

Let $\epsilon = 0,3$

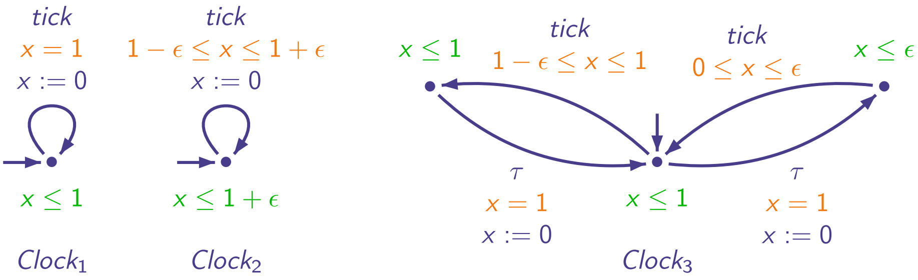
$Clock_2$: $0,8.tick.1,14.tick.\cdots.0,98.tick.\cdots$

n^{th} tick at t with $n \cdot (1 - 0,3) \leq t \leq n \cdot (1 + 0,3)$

$Clock_3$: $0,8.tick.1,3.tick.\cdots.1,15.tick.\cdots$

n^{th} tick at t with $n - 0,3 \leq t \leq n + 0,3$

Digital-Clocks Automata



Timed Words:

$Clock_1$: $1.tick.1.tick.\cdots.1.tick.\cdots$

Let $\epsilon = 0,3$

$Clock_2$: $0,8.tick.1,14.tick.\cdots.0,98.tick.\cdots$

n^{th} tick at t with $n \cdot (1 - 0,3) \leq t \leq n \cdot (1 + 0,3)$

$Clock_3$: $0,8.tick.1,3.tick.\cdots.1,15.tick.\cdots$

n^{th} tick at t with $n - 0,3 \leq t \leq n + 0,3$

Timed Languages & Region Graph/Automaton

- Timed words: alternating sequences of symbols in $\Sigma \cup \mathbb{R}_{\geq 0}$
  Dense-Time: $0.a.\pi.b.\frac{1}{3}.b.\cdots$
  $1.a.2.\varepsilon.1.b = 1.a.3.b$
- Timed Language = set of timed words accepted by a timed automaton
  $\mathcal{L}(A)$ and $\mathcal{L}^\omega(A)$
- Untimed Language = projection on Σ of the Timed Language
  $\pi_{/\Sigma}(1.a.2.\varepsilon.1.b.1) = a.b$
  $\text{Duration}(1.a.2.\varepsilon.1.b.1) = 4$
- Product of timed words/languages: $w \parallel w'$ (for languages $L \parallel L'$)
  $1.a.2.b \parallel 0,5.c.1.d = 0,5.c.0,5.a.0,5.d.1,5.b$
  $1.a \parallel 1.b = \{1.a.0.b, 1.b.0.a\}$
  $1.a \parallel 2.a = \emptyset$

Timed Languages & Region Graph/Automaton

- **Timed words**: alternating sequences of symbols in $\Sigma \cup \mathbb{R}_{\geq 0}$
  Dense-Time: $0.a.\pi.b.\frac{1}{3}.b.\cdots$
  $1.a.2.\varepsilon.1.b \equiv 1.a.3.b$
- Timed Language = set of timed words accepted by a timed automaton
  $\mathcal{L}(A)$ and $\mathcal{L}^{\omega}(A)$
- Untimed Language = projection on Σ of the Timed Language
  $\pi_{/\Sigma}(1.a.2.\varepsilon.1.b.1) = a.b$
  Duration$(1.a.2.\varepsilon.1.b.1) = 4$
- Product of timed words/languages: $w \parallel w'$ (for languages $L \parallel L'$)
  $1.a.2.b \parallel 0,5.c.1.d = 0,5.c.0,5.a.0,5.d.1,5.b$
  $1.a \parallel 1.b = \{1.a.0.b, 1.b.0.a\}$
  $1.a \parallel 2.a = \emptyset$

Timed Languages & Region Graph/Automaton

- Timed words: alternating sequences of symbols in $\Sigma \cup \mathbb{R}_{\geq 0}$
  Dense-Time: $0.a.\pi.b.\frac{1}{3}.b.\cdots$
  $1.a.2.\varepsilon.1.b \equiv 1.a.3.b$
- Timed Language = set of timed words accepted by a timed automaton
  $\mathcal{L}(A)$ and $\mathcal{L}^\omega(A)$
- Untimed Language = projection on Σ of the Timed Language
  $\pi_{/\Sigma}(1.a.2.\varepsilon.1.b.1) = a.b$
  $\text{Duration}(1.a.2.\varepsilon.1.b.1) = 4$
- Product of timed words/languages: $w \parallel w'$ (for languages $L \parallel L'$)
  $1.a.2.b \parallel 0,5.c.1.d = 0,5.c.0,5.a.0,5.d.1,5.b$
  $1.a \parallel 1.b = \{1.a.0.b, 1.b.0.a\}$
  $1.a \parallel 2.a = \emptyset$

Timed Languages & Region Graph/Automaton

- Timed words: alternating sequences of symbols in $\Sigma \cup \mathbb{R}_{\geq 0}$
  Dense-Time: $0.a.\pi.b.\frac{1}{3}.b.\cdots$
  $1.a.2.\varepsilon.1.b \equiv 1.a.3.b$
- Timed Language = set of timed words accepted by a timed automaton
  $\mathcal{L}(A)$ and $\mathcal{L}^\omega(A)$
- Untimed Language = projection on Σ of the Timed Language
  $\pi_{/\Sigma}(1.a.2.\varepsilon.1.b.1) = a.b$
  Duration$(1.a.2.\varepsilon.1.b.1) = 4$
- Product of timed words/languages: $w \parallel w'$ (for languages $L \parallel L'$)
  $1.a.2.b \parallel 0,5.c.1.d = 0,5.c.0,5.a.0,5.d.1,5.b$
  $1.a \parallel 1.b = \{1.a.0.b, 1.b.0.a\}$
  $1.a \parallel 2.a = \emptyset$

Timed Languages & Region Graph/Automaton

- Timed words: alternating sequences of symbols in $\Sigma \cup \mathbb{R}_{\geq 0}$
  Dense-Time: $0.a.\pi.b.\frac{1}{3}.b.\cdots$
  $1.a.2.\varepsilon.1.b \equiv 1.a.3.b$
- Timed Language = set of timed words accepted by a timed automaton
  $\mathcal{L}(A)$ and $\mathcal{L}^{\omega}(A)$
- Untimed Language = projection on Σ of the Timed Language
  $\pi_{/\Sigma}(1.a.2.\varepsilon.1.b.1) = a.b$
  $\text{Duration}(1.a.2.\varepsilon.1.b.1) = 4$
- Product of timed words/languages: $w \parallel w'$ (for languages $L \parallel L'$)
  $1.a.2.b \parallel 0,5.c.1.d = 0,5.c.0,5.a.0,5.d.1,5.b$
  $1.a \parallel 1.b = \{1.a.0.b, 1.b.0.a\}$
  $1.a \parallel 2.a = \emptyset$

Timed Languages & Region Graph/Automaton

- Timed words: alternating sequences of symbols in $\Sigma \cup \mathbb{R}_{\geq 0}$
  Dense-Time: $0.a.\pi.b.\frac{1}{3}.b.\cdots$
  $1.a.2.\varepsilon.1.b \equiv 1.a.3.b$
- Timed Language = set of timed words accepted by a timed automaton
  $\mathcal{L}(A)$ and $\mathcal{L}^\omega(A)$
- Untimed Language = projection on Σ of the Timed Language
  $\pi_{/\Sigma}(1.a.2.\varepsilon.1.b.1) = a.b$
  $\text{Duration}(1.a.2.\varepsilon.1.b.1) = 4$
- Product of timed words/languages: $w \parallel w'$ (for languages $L \parallel L'$)
  $1.a.2.b \parallel 0,5.c.1.d = 0,5.c.0,5.a.0,5.d.1,5.b$
  $1.a \parallel 1.b = \{1.a.0.b, 1.b.0.a\}$
  $1.a \parallel 2.a = \emptyset$

Product of Automata

Given A and B, we can effectively build a TA $(A \parallel B)$ that accepts the timed language $\mathcal{L}(A) \parallel \mathcal{L}(B)$.

Timed Languages & Region Graph/Automaton

- Timed words: alternating sequences of symbols in $\Sigma \cup \mathbb{R}_{\geq 0}$
  Dense-Time: $0.a.\pi.b.\frac{1}{3}.b.\cdots$
  $1.a.2.\varepsilon.1.b \equiv 1.a.3.b$
- Timed Language = set of timed words accepted by a timed automaton
  $\mathcal{L}(A)$ and $\mathcal{L}^\omega(A)$
- Untimed Language = projection on Σ of the Timed Language
  $\pi_{/\Sigma}(1.a.2.\varepsilon.1.b.1) = a.b$
  $\text{Duration}(1.a.2.\varepsilon.1.b.1) = 4$
- Product of timed words/languages: $w \parallel w'$ (for languages $L \parallel L'$)
  $1.a.2.b \parallel 0,5.c.1.d = 0,5.c.0,5.a.0,5.d.1,5.b$
  $1.a \parallel 1.b = \{1.a.0.b, 1.b.0.a\}$
  $1.a \parallel 2.a = \emptyset$

Theorem (Region Graph ▸ Region Graph)

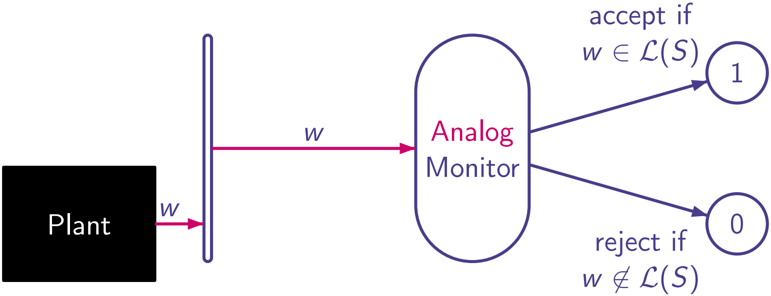
For each Timed Automaton A, we can effectively build a finite automaton $RG(A)$ s.t. $\mathcal{L}(RG(A)) = Untimed(\mathcal{L}(A))$. [Alur & Dill, TCS'94]

Outline

- Models for Timed Systems & Digital Clocks
- Monitoring with Digital Clocks
- Diagnosis with Digital Clocks
- Conclusion & Open Problem

Monitors & Digital Clocks

Specification: $\mathcal{L}(S)$



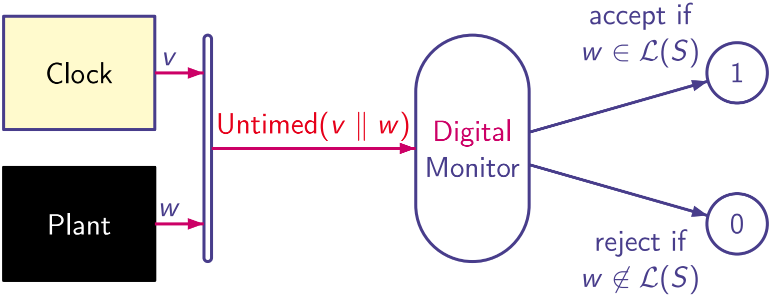
- Plant: generates timed words $w = t_0 a_0 t_1 a_1 \cdots t_n a_n$
- Digital Clock: generates $v \in (tick \cup \mathbb{R}_{\geq 0})^*$, non zeno
- Plant $\|$ Clock: generates timed words in $(\Sigma \cup \{tick\} \cup \mathbb{R}_{\geq 0})^*$

  $\rho = v \| w = 1.a.0.tick.2.b.1.tick.2.tick.8$
- Monitor: deterministic, accepts untimed words in $(\Sigma \cup \{tick\})^*$

  $\pi_{/\Sigma \cup \{tick\}}(1.a.0.tick.2.b.1.tick.2.tick.8) = a.tick.b.tick.tick$

Monitors & Digital Clocks

Specification: $\mathcal{L}(S)$



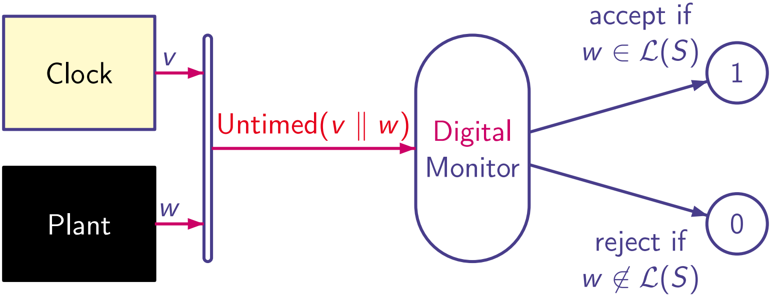
- Plant: generates timed words $w = t_0 a_0 t_1 a_1 \cdots t_n a_n$
- Digital Clock: generates $v \in (tick \cup \mathbb{R}_{\geq 0})^*$, non zeno
- Plant $\|$ Clock: generates timed words in $(\Sigma \cup \{tick\} \cup \mathbb{R}_{\geq 0})^*$

  $\rho = v \| w = 1.a.0.tick.2.b.1.tick.2.tick.8$
- Monitor: deterministic, accepts untimed words in $(\Sigma \cup \{tick\})^*$

  $\pi_{/\Sigma \cup \{tick\}}(1.a.0.tick.2.b.1.tick.2.tick.8) = a.tick.b.tick.tick$

Monitors & Digital Clocks

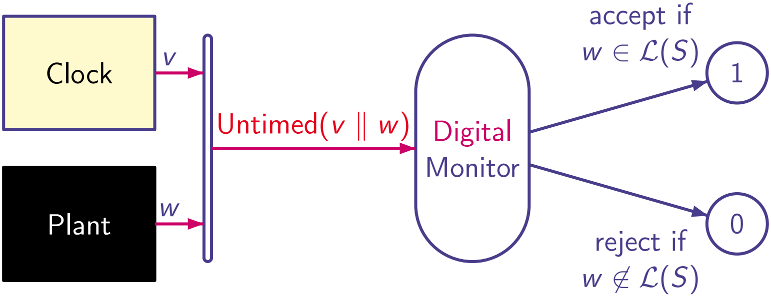
Specification: $\mathcal{L}(S)$



- Plant: generates timed words $w = t_0 a_0 t_1 a_1 \cdots t_n a_n$
- Digital Clock: generates $v \in (tick \cup \mathbb{R}_{\geq 0})^*$, non zeno
- Plant $\|$ Clock: generates timed words in $(\Sigma \cup \{tick\} \cup \mathbb{R}_{\geq 0})^*$
  $\rho = v \| w = 1.a.0.tick.2.b.1.tick.2.tick.8$
- Monitor: deterministic, accepts untimed words in $(\Sigma \cup \{tick\})^*$
  $\pi_{/\Sigma \cup \{tick\}}(1.a.0.tick.2.b.1.tick.2.tick.8) = a.tick.b.tick.tick$

Monitors & Digital Clocks

Specification: $\mathcal{L}(S)$



- Plant: generates timed words $w = t_0 a_0 t_1 a_1 \cdots t_n a_n$
- Digital Clock: generates $v \in (tick \cup \mathbb{R}_{\geq 0})^*$, non zeno
- Plant $\parallel$ Clock: generates timed words in $(\Sigma \cup \{tick\} \cup \mathbb{R}_{\geq 0})^*$
  $\rho = v \parallel w = 1.a.0.tick.2.b.1.tick.2.tick.8$
- Monitor: deterministic, accepts untimed words in $(\Sigma \cup \{tick\})^*$
  $\pi_{/\Sigma \cup \{tick\}}(1.a.0.tick.2.b.1.tick.2.tick.8) = a.tick.b.tick.tick$

Sound Monitors

Definition (Soundness)

An monitor M is sound w.r.t. $Clock$ if $\forall \rho \in \mathcal{L}(S)$ and $\rho' \in \mathcal{L}(Clock)$ M accepts $\text{Untimed}(\rho \parallel \rho')$ (or equivalently $M(\text{Untimed}(\rho \parallel \rho')) = 1$).

This is NOT equivalent to $\mathcal{L}(S) \subseteq (\mathcal{L}(M) \parallel \mathcal{L}(Clock))$

Property 1 (Better Clock Preserves Soundness)

If M is sound w.r.t. $Clock_1$ and $\mathcal{L}(Clock_2) \subseteq \mathcal{L}(Clock_1)$ then M is sound w.r.t. $Clock_2$.

Property 2 (Minimal Language of a Sound Monitor)

If M is sound w.r.t. $Clock$ then $\text{Untimed}(\mathcal{L}(S) \parallel \mathcal{L}(Clock)) \subseteq \mathcal{L}(M)$.

Sound Monitors

Definition (Soundness)

An monitor M is sound w.r.t. $Clock$ if $\forall \rho \in \mathcal{L}(S)$ and $\rho' \in \mathcal{L}(Clock)$ M accepts $\text{Untimed}(\rho \parallel \rho')$ (or equivalently $M(\text{Untimed}(\rho \parallel \rho')) = 1$).

This is NOT equivalent to $\mathcal{L}(S) \subseteq (\mathcal{L}(M) \parallel \mathcal{L}(Clock))$

Property 1 (Better Clock Preserves Soundness)

If M is sound w.r.t. $Clock_1$ and $\mathcal{L}(Clock_2) \subseteq \mathcal{L}(Clock_1)$ then M is sound w.r.t. $Clock_2$.

Property 2 (Minimal Language of a Sound Monitor)

If M is sound w.r.t. $Clock$ then $\text{Untimed}(\mathcal{L}(S) \parallel \mathcal{L}(Clock)) \subseteq \mathcal{L}(M)$.

Sound Monitors

Definition (Soundness)

An monitor M is sound w.r.t. $Clock$ if $\forall \rho \in \mathcal{L}(S)$ and $\rho' \in \mathcal{L}(Clock)$ M accepts $\text{Untimed}(\rho \parallel \rho')$ (or equivalently $M(\text{Untimed}(\rho \parallel \rho')) = 1$).

This is NOT equivalent to $\mathcal{L}(S) \subseteq (\mathcal{L}(M) \parallel \mathcal{L}(Clock))$

Property 1 (Better Clock Preserves Soundness)

If M is sound w.r.t. $Clock_1$ and $\mathcal{L}(Clock_2) \subseteq \mathcal{L}(Clock_1)$ then M is sound w.r.t. $Clock_2$.

Property 2 (Minimal Language of a Sound Monitor)

If M is sound w.r.t. $Clock$ then $\text{Untimed}(\mathcal{L}(S) \parallel \mathcal{L}(Clock)) \subseteq \mathcal{L}(M)$.

Sound Monitors

Definition (Soundness)

An monitor M is sound w.r.t. *Clock* if $\forall \rho \in \mathcal{L}(S)$ and $\rho' \in \mathcal{L}(Clock)$ M accepts $\text{Untimed}(\rho \parallel \rho')$ (or equivalently $M(\text{Untimed}(\rho \parallel \rho')) = 1$).

This is NOT equivalent to $\mathcal{L}(S) \subseteq (\mathcal{L}(M) \parallel \mathcal{L}(Clock))$

Property 1 (Better Clock Preserves Soundness)

If M is sound w.r.t. $Clock_1$ and $\mathcal{L}(Clock_2) \subseteq \mathcal{L}(Clock_1)$ then M is sound w.r.t. $Clock_2$.

Property 2 (Minimal Language of a Sound Monitor)

If M is sound w.r.t. *Clock* then $\text{Untimed}(\mathcal{L}(S) \parallel \mathcal{L}(Clock)) \subseteq \mathcal{L}(M)$.

Construction of the Optimal Sound Monitor

Problem 0

Inputs: Two timed automata S and $Clock$.

Problem: Build a sound monitor.

Construction of the Optimal Sound Monitor

Problem 0

Inputs: Two timed automata S and $Clock$.
Problem: Build a sound monitor.

Trivial Solution: $M(u) = 1$ for any u

Construction of the Optimal Sound Monitor

Problem 0

Inputs: Two timed automata S and $Clock$.
Problem: Build a sound monitor.

Trivial Solution: $M(u) = 1$ for any u

Definition (Order on Monitors)

M is better than M' if $\mathcal{L}(M) \subseteq \mathcal{L}(M')$.

Construction of the Optimal Sound Monitor

Problem 1

Inputs: Two timed automata S and $Clock$.
Problem: Build a minimal (or optimal) sound monitor.

Construction of the Optimal Sound Monitor

Problem 1

Inputs: Two timed automata S and $Clock$.

Problem: Build a minimal (or optimal) sound monitor.

1. Build the region graph of $(S \parallel Clock)$ and determinize it: result $= RG$

Construction of the Optimal Sound Monitor

Problem 1

Inputs: Two timed automata S and $Clock$.
Problem: Build a minimal (or optimal) sound monitor.

1. Build the region graph of $(S \parallel Clock)$ and determinize it: result $= RG$
2. Define M_0 by: $M_0(u) = 1$ iff u is accepted by RG

Construction of the Optimal Sound Monitor

Problem 1

Inputs: Two timed automata S and $Clock$.
Problem: Build a minimal (or optimal) sound monitor.

1. Build the region graph of $(S \parallel Clock)$ and determinize it: result $= RG$
2. Define M_0 by: $M_0(u) = 1$ iff u is accepted by RG

Theorem (Soundess and Optimality of M_0)

M_0 is sound and optimal.

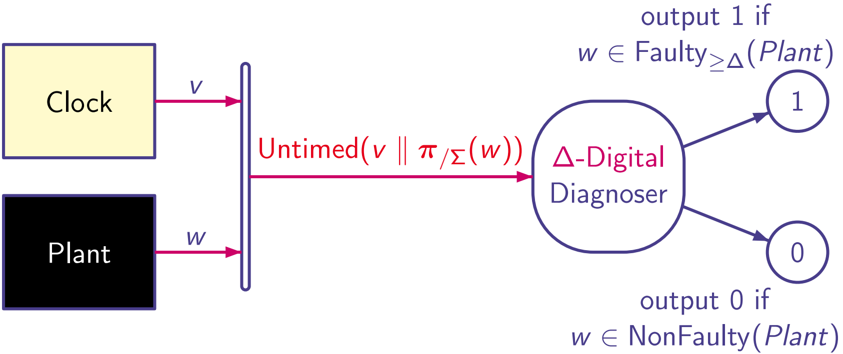
Proof

Soundness: If not, $\exists u \in \mathcal{L}(RG)$ s.t. $u \notin \text{Untimed}(\mathcal{L}(S) \parallel \mathcal{L}(Clock))$.
Optimality: By Property 2, a sound monitor must contain at least $\text{Untimed}(\mathcal{L}(S) \parallel \mathcal{L}(Clock))$ which is equal to $\mathcal{L}(RG)$.

Outline

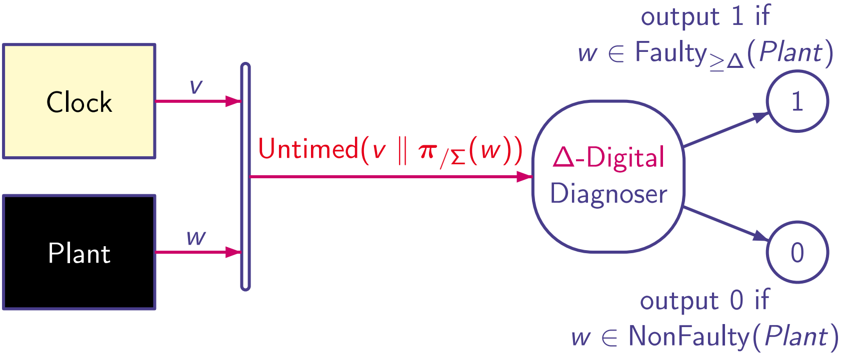
- Models for Timed Systems & Digital Clocks
- Monitoring with Digital Clocks
- **Diagnosis with Digital Clocks**
- Conclusion & Open Problem

Δ-Diagnosers & Digital Clocks



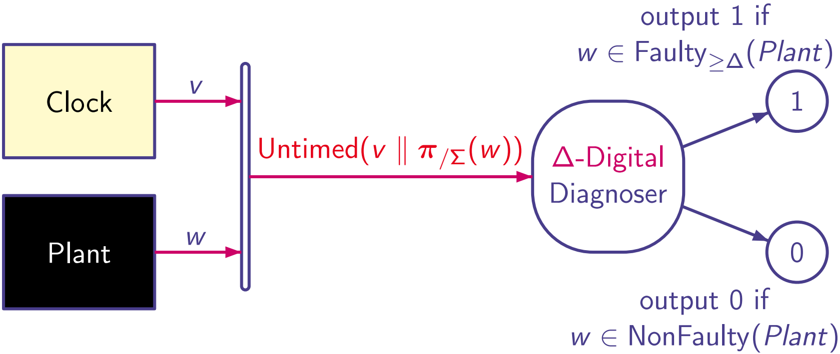
- Plant: ε and f unobservable
- $\rho = \rho_1.f.\rho_2$ is Δ-faulty if $f \notin \rho_1$ and $\text{Duration}(\rho_2) \geq \Delta$
  If $f \notin \rho$ then ρ is non faulty
- A Diagnoser D does not change its mind:
  $D(\rho) = 1 \implies D(\rho.\rho') = 1.$

Δ-Diagnosers & Digital Clocks



- Plant: ε and f unobservable
- $\rho = \rho_1.f.\rho_2$ is Δ-faulty if $f \notin \rho_1$ and Duration$(\rho_2) \geq \Delta$
  If $f \notin \rho$ then ρ is non faulty
- A Diagnoser D does not change its mind:
  $D(\rho) = 1 \implies D(\rho.\rho') = 1.$

Δ-Diagnosers & Digital Clocks



- Plant: ε and f unobservable
- $\rho = \rho_1.f.\rho_2$ is Δ-faulty if $f \notin \rho_1$ and Duration(ρ_2) $\geq \Delta$
  If $f \notin \rho$ then ρ is non faulty
- A Diagnoser D does not change its mind:
  $D(\rho) = 1 \implies D(\rho.\rho') = 1.$

Diagnosers & Diagnosability Problems

Definition ($(Clock, \Delta)$-Diagnosability)

$D : (\Sigma \cup \{tick\})^* \rightarrow \{0, 1\}$ is a $(Clock, \Delta)$-diagnoser for *Plant* if for any runs $\rho \in \mathcal{L}(Plant)$ and $\rho' \in \mathcal{L}(Clock)$ with $\text{Duration}(\rho) = \text{Duration}(\rho')$

- if $\rho \in \text{NonFaulty}(Plant)$ then $D(\text{Untimed}(\rho \parallel \rho')) = 0$
- if $\rho \in \text{Faulty}_{\geq\Delta}(Plant)$ then $D(\text{Untimed}(\rho \parallel \rho')) = 1$

Plant is $(Clock, \Delta)$-Diagnosable if $\exists$ a $(Clock, \Delta)$-diagnoser D.

Diagnosers & Diagnosability Problems

Definition (($Clock, \Delta$)-Diagnosability)

$D : (\Sigma \cup \{tick\})^* \rightarrow \{0, 1\}$ is a ($Clock, \Delta$)-diagnoser for $Plant$ if for any runs $\rho \in \mathcal{L}(Plant)$ and $\rho' \in \mathcal{L}(Clock)$ with $\text{Duration}(\rho) = \text{Duration}(\rho')$

- if $\rho \in \text{NonFaulty}(Plant)$ then $D(\text{Untimed}(\rho \parallel \rho')) = 0$
- if $\rho \in \text{Faulty}_{\geq \Delta}(Plant)$ then $D(\text{Untimed}(\rho \parallel \rho')) = 1$

$Plant$ is ($Clock, \Delta$)-Diagnosable if $\exists$ a ($Clock, \Delta$)-diagnoser D.

Property 3 (Better Clocks . . .)

For any timed automata A, $Clock_1$ and $Clock_2$, for any $\Delta_1, \Delta_2 \in \mathbb{N}$, if D is a ($Clock_1, \Delta_1$)-diagnoser for A and $L(Clock_2) \subseteq L(Clock_1)$ and $\Delta_2 \geq \Delta_1$, then D is also a ($Clock_2, \Delta_2$)-diagnoser for A.

Diagnosers & Diagnosability Problems

Problem 2: $(Clock, \Delta)$-Diagnosability

Inputs: Two timed automata *Plant* and *Clock* and $\Delta \in \mathbb{N}$.
Problem: Check whether *Plant* is $(Clock, \Delta)$-diagnosable.

Diagnosers & Diagnosability Problems

Problem 2: $(Clock, \Delta)$-Diagnosability

Inputs: Two timed automata *Plant* and *Clock* and $\Delta \in \mathbb{N}$.
Problem: Check whether *Plant* is $(Clock, \Delta)$-diagnosable.

Problem 3: *Clock*-Diagnosability

Inputs: Two timed automata *Plant* and *Clock*.
Problem: Check whether $\exists \Delta \in \mathbb{N}$ s.t. *Plant* is $(Clock, \Delta)$-diagnosable.

Diagnosers & Diagnosability Problems

Problem 2: $(Clock, \Delta)$-Diagnosability

Inputs: Two timed automata *Plant* and *Clock* and $\Delta \in \mathbb{N}$.

Problem: Check whether *Plant* is $(Clock, \Delta)$-diagnosable.

Problem 3: *Clock*-Diagnosability

Inputs: Two timed automata *Plant* and *Clock*.

Problem: Check whether $\exists \Delta \in \mathbb{N}$ s.t. *Plant* is $(Clock, \Delta)$-diagnosable.

Problem 4: Diagnosability

Inputs: A timed automaton *Plant*.

Problem: Check whether $\exists$ a TA *Clock* s.t. *Plant* is *Clock*-diagnosable.

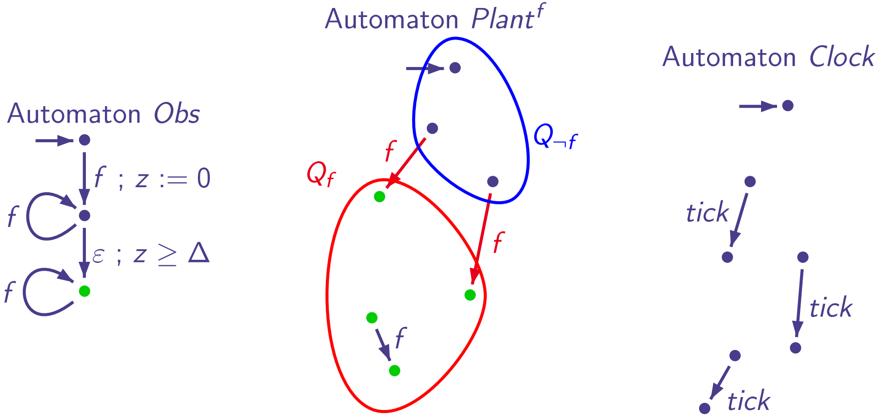
Solution to Problem 2

C_1: Necess. and Suffi. Condition for $(Clock, \Delta)$-diagnosability

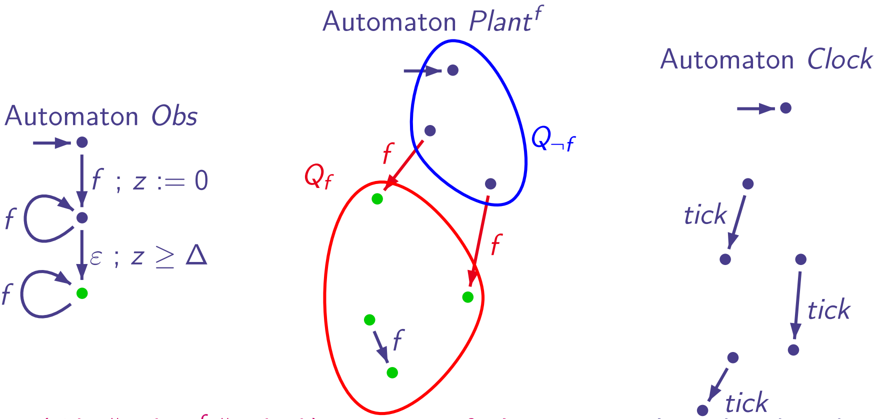
Plant is $(Clock, \Delta)$-diagnosable iff $\forall \rho, \rho' \in \mathcal{L}(Plant)$, $\sigma, \sigma' \in \mathcal{L}(Clock)$

$$\left.\begin{array}{c} \rho \in \text{Faulty}_{\geq \Delta}(Plant) \\ \rho' \in \text{NonFaulty}(Plant) \\ \text{Duration}(\rho) = \text{Duration}(\sigma) \\ \text{Duration}(\rho') = \text{Duration}(\sigma') \end{array}\right\} \implies \text{Untimed}(\rho \parallel \sigma) \cap \text{Untimed}(\rho' \parallel \sigma') = \emptyset$$

Solution to Problem 2

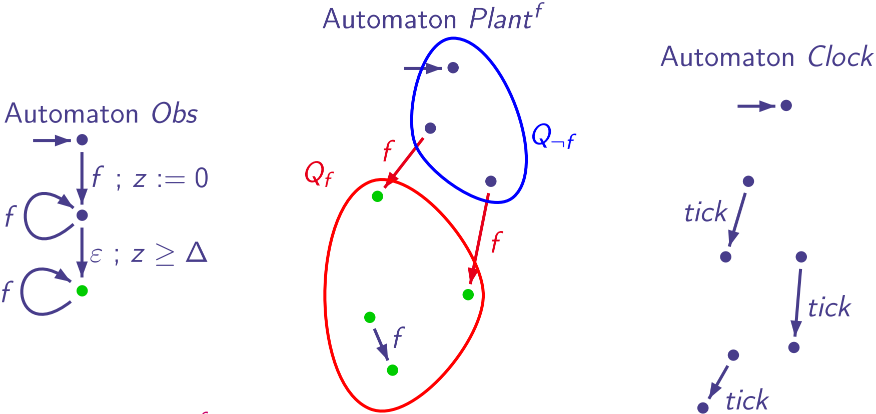


Solution to Problem 2



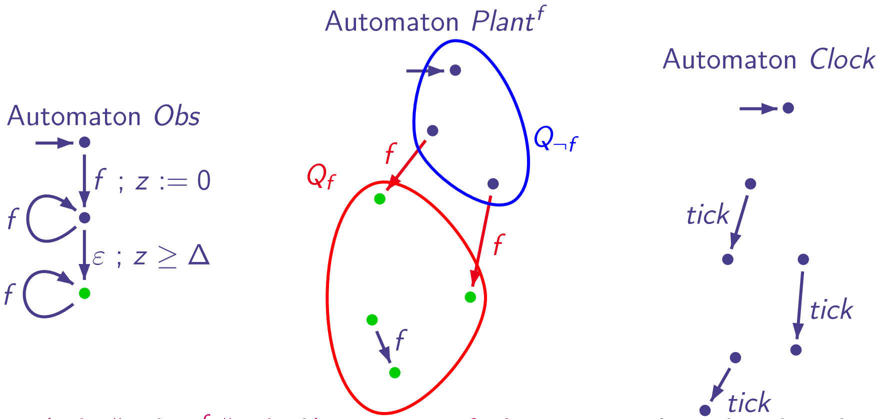
$P = (Obs \parallel Plant^f \parallel Clock)$ accepts Δ-faulty runs interleaved with *ticks*

Solution to Problem 2



$P = (Obs \parallel Plant^f \parallel Clock)$ accepts Δ-faulty runs interleaved with *ticks*
$P' = (Plant^{\neg f} \parallel Clock)$ accepts non faulty runs

Solution to Problem 2



$P = (Obs \parallel Plant^f \parallel Clock)$ accepts Δ-faulty runs interleaved with *ticks*
$P' = (Plant^{\neg f} \parallel Clock)$ accepts non faulty runs

$$C_1 \iff \text{Untimed}(\mathcal{L}(P)) \cap \text{Untimed}(\mathcal{L}(P')) = \emptyset$$

Solution to Problem 3

Problem 3: Diagnosability

Inputs: Two timed automata *Plant* and *Clock*.
Problem: Check whether *Plant* is *Clock*-diagnosable for some TA *Clock*.

For DES: amounts to checking (Büchi) emptyness

Algorithm for DES

Assumption: *Plant* is non zeno

C_2: Necess. and Suffi. Condition for *Clock*-diagnosability

Plant is NOT *Clock*-diagnosable iff $\exists \rho, \rho' \in \mathcal{L}^\omega(Plant), \sigma, \sigma' \in \mathcal{L}^\omega(Clock)$

$$\left.\begin{array}{r}\rho \in \text{Faulty}_{\geq\Delta}(Plant) \\ \rho' \in \text{NonFaulty}(Plant)\end{array}\right\} \Longrightarrow \text{Untimed}(\rho \parallel \sigma) \cap \text{Untimed}(\rho' \parallel \sigma') \neq \emptyset$$

$$C_2 \iff \text{Untimed}(\mathcal{L}^\omega(P)) \cap \text{Untimed}(\mathcal{L}^\omega(P')) \neq \emptyset$$

Solution to Problem 3

Problem 3: Diagnosability

Inputs: Two timed automata *Plant* and *Clock*.
Problem: Check whether *Plant* is *Clock*-diagnosable for some TA *Clock*.

For DES: amounts to checking (Büchi) emptyness

Algorithm for DES

Assumption: *Plant* is non zeno

C_2: Necess. and Suffi. Condition for *Clock*-diagnosability

Plant is NOT *Clock*-diagnosable iff $\exists \rho, \rho' \in \mathcal{L}^\omega(Plant)$, $\sigma, \sigma' \in \mathcal{L}^\omega(Clock)$

$$\left.\begin{array}{r}\rho \in \text{Faulty}_{\geq\Delta}(Plant)\\ \rho' \in \text{NonFaulty}(Plant)\end{array}\right\} \Longrightarrow \text{Untimed}(\rho \parallel \sigma) \cap \text{Untimed}(\rho' \parallel \sigma') \neq \emptyset$$

$$C_2 \iff \text{Untimed}(\mathcal{L}^\omega(P)) \cap \text{Untimed}(\mathcal{L}^\omega(P')) \neq \emptyset$$

Solution to Problem 3

Problem 3: Diagnosability

Inputs: Two timed automata *Plant* and *Clock*.
Problem: Check whether *Plant* is *Clock*-diagnosable for some TA *Clock*.

For DES: amounts to checking (Büchi) emptyness

Algorithm for DES

Assumption: *Plant* is non zeno

C_2: Necess. and Suffi. Condition for *Clock*-diagnosability

Plant is NOT *Clock*-diagnosable iff $\exists \rho, \rho' \in \mathcal{L}^\omega(Plant)$, $\sigma, \sigma' \in \mathcal{L}^\omega(Clock)$

$$\left.\begin{array}{r}\rho \in \text{Faulty}_{\geq\Delta}(Plant)\\ \rho' \in \text{NonFaulty}(Plant)\end{array}\right\} \Longrightarrow \text{Untimed}(\rho \parallel \sigma) \cap \text{Untimed}(\rho' \parallel \sigma') \neq \emptyset$$

$$C_2 \iff \text{Untimed}(\mathcal{L}^\omega(P)) \cap \text{Untimed}(\mathcal{L}^\omega(P')) \neq \emptyset$$

Solution to Problem 3

Problem 3: Diagnosability

Inputs: Two timed automata *Plant* and *Clock*.
Problem: Check whether *Plant* is *Clock*-diagnosable for some TA *Clock*.

For DES: amounts to checking (Büchi) emptyness

▸ Algorithm for DES

Assumption: *Plant* is non zeno

C_2: Necess. and Suffi. Condition for *Clock*-diagnosability

Plant is NOT *Clock*-diagnosable iff $\exists \rho, \rho' \in \mathcal{L}^\omega(Plant)$, $\sigma, \sigma' \in \mathcal{L}^\omega(Clock)$

$$\left.\begin{array}{r}\rho \in \text{Faulty}_{\geq\Delta}(Plant)\\ \rho' \in \text{NonFaulty}(Plant)\end{array}\right\} \Longrightarrow \text{Untimed}(\rho \parallel \sigma) \cap \text{Untimed}(\rho' \parallel \sigma') \neq \emptyset$$

$$C_2 \iff \text{Untimed}(\mathcal{L}^\omega(P)) \cap \text{Untimed}(\mathcal{L}^\omega(P')) \neq \emptyset$$

Solution to Problem 3

Problem 3: Diagnosability

Inputs: Two timed automata *Plant* and *Clock*.
Problem: Check whether *Plant* is *Clock*-diagnosable for some TA *Clock*.

For DES: amounts to checking (Büchi) emptyness

▸ Algorithm for DES

Assumption: *Plant* is non zeno

C_2: Necess. and Suffi. Condition for *Clock*-diagnosability

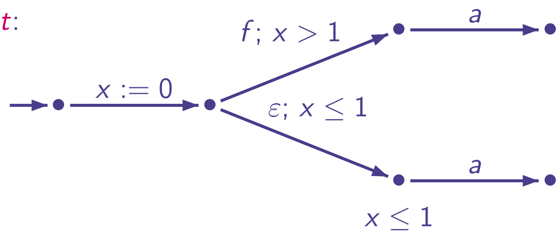
Plant is NOT *Clock*-diagnosable iff $\exists \rho, \rho' \in \mathcal{L}^\omega(Plant)$, $\sigma, \sigma' \in \mathcal{L}^\omega(Clock)$

$$\left.\begin{array}{r}\rho \in \text{Faulty}_{\geq\Delta}(Plant)\\ \rho' \in \text{NonFaulty}(Plant)\end{array}\right\} \implies \text{Untimed}(\rho \parallel \sigma) \cap \text{Untimed}(\rho' \parallel \sigma') \neq \emptyset$$

$$C_2 \iff \text{Untimed}(\mathcal{L}^\omega(P)) \cap \text{Untimed}(\mathcal{L}^\omega(P')) \neq \emptyset$$

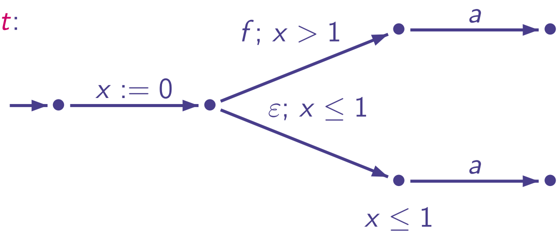
Problem 4: Existence of a Digital Clock Diagnoser

Plant:



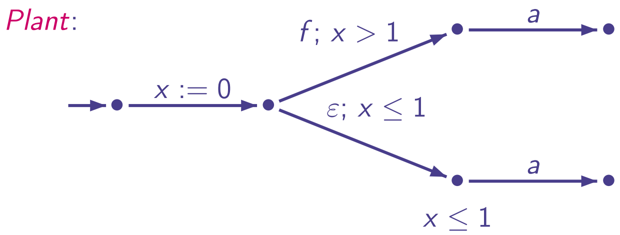
Problem 4: Existence of a Digital Clock Diagnoser

Plant:



Plant is NOT *Clock*-diagnosable for any TA *Clock*.

Problem 4: Existence of a Digital Clock Diagnoser



Time: 1 t' 2

ticks: $\bullet^1$ $\cdots$ $\bullet$ $\bullet^n$ $\cdots$ $\bullet^{n+1}$ $\bullet^{n+m}$

$f.a$

$\varepsilon.a$

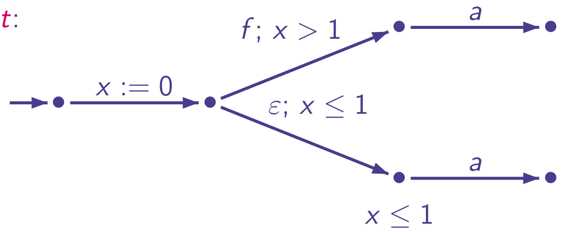
run ρ' of *Clock*

run ρ_1 of *Plant*

run ρ_2 of *Plant*

Problem 4: Existence of a Digital Clock Diagnoser

Plant:



Time: 1 t' 2

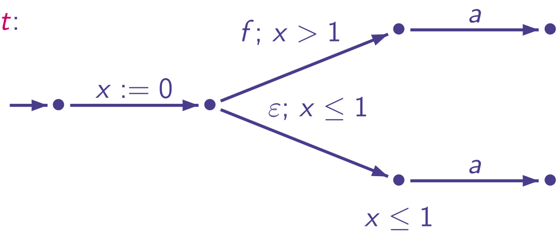
ticks: $\bullet^1$ $\cdots$ $\bullet$ $\bullet^n$ $\cdots$ $\bullet^{n+1}$ $\bullet^{n+m}$ run ρ' of *Clock*

$f.a$ run ρ_1 of *Plant*

$\varepsilon.a$ run ρ_2 of *Plant*

Problem 4: Existence of a Digital Clock Diagnoser

Plant:



Time: 1 t' 2

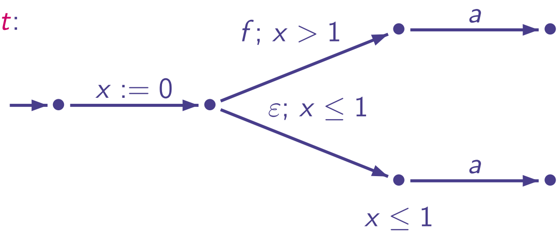
ticks: $\bullet^{1}\ \cdots\ \bullet\ \bullet^{n}\ \cdots\ \bullet^{n+1}\ \bullet^{n+m}$ — run ρ' of *Clock*

$f.a$ — run ρ_1 of *Plant*

$\varepsilon.a$ — run ρ_2 of *Plant*

Problem 4: Existence of a Digital Clock Diagnoser

Plant:



Time: 1 t' 2

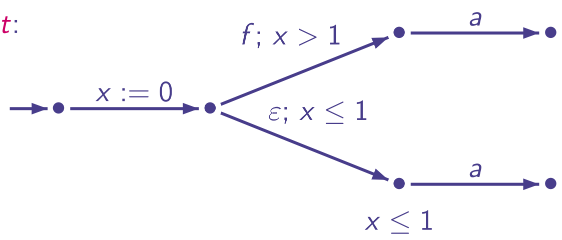
ticks: $\bullet^1$ $\cdots$ $\bullet$ $\bullet^n$ $\cdots$ $\bullet^{n+1}$ $\bullet^{n+m}$ run ρ' of *Clock*

$f.a$ run ρ_1 of *Plant*

$\varepsilon.a$ run ρ_2 of *Plant*

Problem 4: Existence of a Digital Clock Diagnoser

Plant:



Time: 1 t' 2

ticks: $\bullet^1$ $\cdots$ $\bullet$ $\bullet^n$ $\cdots$ $\bullet^{n+1}$ $\bullet^{n+m}$ — run ρ' of *Clock*

$f.a$ — run ρ_1 of *Plant*

$\varepsilon.a$ — run ρ_2 of *Plant*

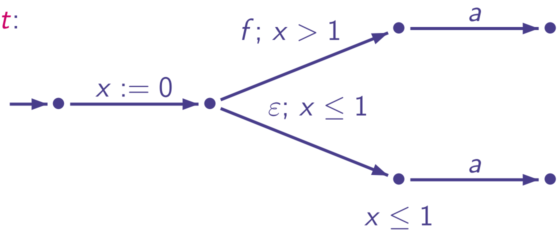
$\rho_1 = t'.f.0.a.(2 - t')$ $\quad \rho_2 = 1.\varepsilon.0.a.1$

$\rho_1 \parallel \rho' = tick^n.f.a.tick^m$ $\quad \rho_2 \parallel \rho' \ni tick^n.\varepsilon.a.tick^m$

$tick^n.a.tick^m \in \text{Untimed}(\rho_1 \parallel \rho') \cap \text{Untimed}(\rho_2 \parallel \rho')$

Problem 4: Existence of a Digital Clock Diagnoser

Plant:



Time: 1 t' 2

ticks: $\bullet^1$ $\cdots$ $\bullet$ $\bullet^n$ $\cdots$ $\bullet^{n+1}$ $\bullet^{n+m}$ — run ρ' of *Clock*

$f.a$ — run ρ_1 of *Plant*

$\varepsilon.a$ — run ρ_2 of *Plant*

$\rho_1 = t'.f.0.a.(2 - t')$ $\quad \rho_2 = 1.\varepsilon.0.a.1$

$\rho_1 \parallel \rho' = tick^n.f.a.tick^m$ $\quad \rho_2 \parallel \rho' \ni tick^n.\varepsilon.a.tick^m$

$$tick^n.a.tick^m \in \text{Untimed}(\rho_1 \parallel \rho') \cap \text{Untimed}(\rho_2 \parallel \rho')$$

Outline

- Models for Timed Systems & Digital Clocks
- Monitoring with Digital Clocks
- Diagnosis with Digital Clocks
- **Conclusion & Open Problem**

Conclusion & Open Problem

- Monitoring with digital clocks: region graph
- (Δ, Clock) and *Clock*-diagnosability decidable
- Diagnosability (existence of a digital clock): Open
- Recent Related Work: [Jiang, Kumar, ACC'06]
  - Digital Clocks and Fault-Diagnosis
  - Periodic clock: ticks every $\Delta \pm \epsilon$
  - Problem 4 not considered

Conclusion & Open Problem

- Monitoring with digital clocks: region graph
- (Δ, *Clock*) and *Clock*-diagnosability decidable
- Diagnosability (existence of a digital clock): Open

- Recent Related Work: [Jiang, Kumar, ACC'06]
  - Digital Clocks and Fault-Diagnosis
  - Periodic clock: ticks every $\Delta \pm \epsilon$
  - Problem 4 not considered

References

[Alur & Dill, TCS'94] R. Alur and D. Dill.
A theory of timed automata.
Theoretical Computer Science, 126:183–235, 1994.

[Bouyer et al., FoSSaCS'05] P. Bouyer, F. Chevalier, and D. D'Souza.
Fault diagnosis using timed automata.
In *FoSSaCS'05*, volume 3441 of *LNCS*, pages 219–233. Springer, 2005.

[Jiang, Kumar, ACC'06] S. Jiang and R. Kumar.
Diagnosis of dense-time systems using digital clocks.
In *American Control Conference'06*, 2006.
To appear.

[Krichen, Tripakis, SPIN'04] M. Krichen and S. Tripakis.
Black-box conformance testing for real-time systems.
In *11th International SPIN Workshop on Model Checking of Software (SPIN'04)*, volume 2989 of *LNCS*. Springer, 2004.

[Krichen, Tripakis, FORMATS'04] M. Krichen and S. Tripakis.
Real-time testing with timed automata testers and coverage criteria.
In *Formal Techniques, Modelling and Analysis of Timed and Fault Tolerant Systems (FORMATS-FTRTFT'04)*, volume 3253 of *LNCS*. Springer, 2004.

[Sampath et al., IEEE'95] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis.
Diagnosability of discrete event systems.
IEEE Transactions on Automatic Control, 40(9), Sept. 1995.

References (cont.)

[Tripakis, FTRTFT'02] S. Tripakis.
Fault diagnosis for timed automata.
In *Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFT'02)*, volume 2469 of *LNCS*. Springer, 2002.

Acknowledgements: ACI-CORTOS

Timed Automata [Alur & Dill, TCS'94]

A Timed Automaton $\mathcal{A}$ is a tuple $(L, \ell_0, \mathrm{Act}, X, \mathrm{inv}, \longrightarrow)$ where:

- L is a finite set of locations
- ℓ_0 is the initial location
- X is a finite set of clocks
- Act is a finite set of actions
- $\longrightarrow$ is a set of transitions of the form $\ell \xrightarrow{g\,,\,a\,,\,R} \ell'$ with:
  - $\ell, \ell' \in L$,
  - $a \in \mathrm{Act}$
  - a guard g which is a clock constraint over X
  - a reset set R which is the set of clocks to be reset to 0

Clock constraints are boolean combinations of $x \sim k$ with $x \in C$ and $k \in \mathbb{Z}$ and $\sim \in \{\leq, <\}$.

Back

Semantics of Timed Automata

Let $\mathcal{A} = (L, \ell_0, \mathrm{Act}, X, \mathrm{inv}, \longrightarrow)$ be a Timed Automaton.

A state (ℓ, v) of $\mathcal{A}$ is in $L \times \mathbb{R}_{\geq 0}^X$

The semantics of $\mathcal{A}$ is a Timed Transition System $S_{\mathcal{A}} = (Q, q_0, \mathrm{Act} \cup \mathbb{R}_{\geq 0}, \longrightarrow)$ with:

- $Q = L \times \mathbb{R}_{\geq 0}^X$
- $q_0 = (\ell_0, \overline{0})$
- $\longrightarrow$ consists in:

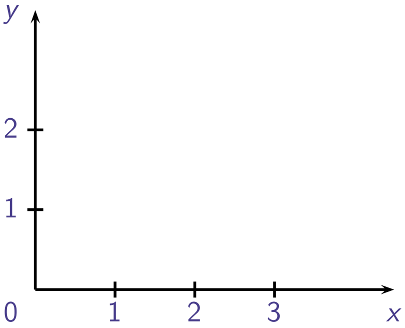
discrete transition: $(\ell, v) \xrightarrow{a} (\ell', v') \iff \begin{cases} \exists\, \ell \xrightarrow{g\,,\,a\,,\,r} \ell' \in \mathcal{A} \\ v \models g \\ v' = v[r \leftarrow 0] \\ v' \models \mathrm{inv}(\ell') \end{cases}$

delay transition: $(\ell, v) \xrightarrow{d} (\ell, v + d) \iff d \in \mathbb{R}_{\geq 0} \wedge v + d \models \mathrm{inv}(\ell)$

Back

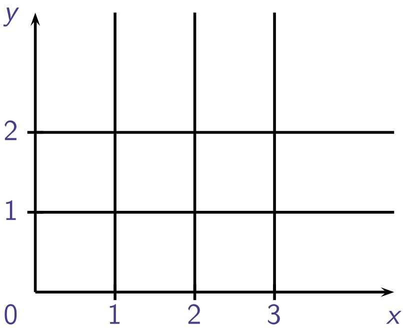
The Region Abstraction

[Alur & Dill, TCS'94]



The Region Abstraction

[Alur & Dill, TCS'94]



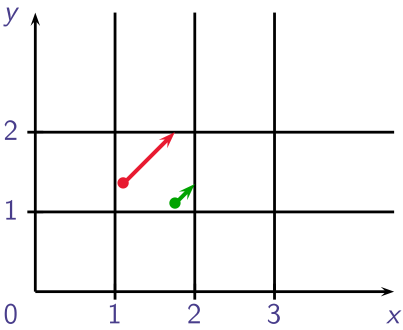
Build an equivalence relation which is of finite index and is:

- "compatible" with clock constraints ($g ::= x \sim c \quad g \wedge g$)

  $r, r' \in R \implies \forall$ constraints $g, \quad r \models g \iff r' \models g$

The Region Abstraction

[Alur & Dill, TCS'94]



Build an equivalence relation which is of finite index and is:

- "compatible" with clock constraints $(g ::= x \sim c \quad g \wedge g)$

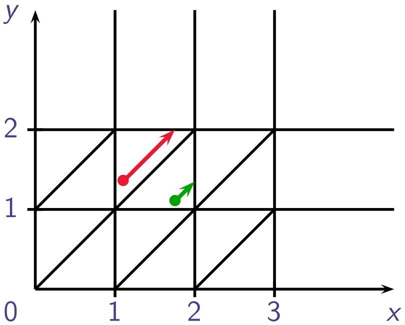
  $r, r' \in R \implies \forall$ constraints $g, \quad r \models g \iff r' \models g$

- "compatible" with time elapsing

  $r, r' \in R \implies$ same delay successor regions

The Region Abstraction

[Alur & Dill, TCS'94]



Build an equivalence relation which is of finite index and is:

- "compatible" with clock constraints ($g ::= x \sim c \quad g \wedge g$)

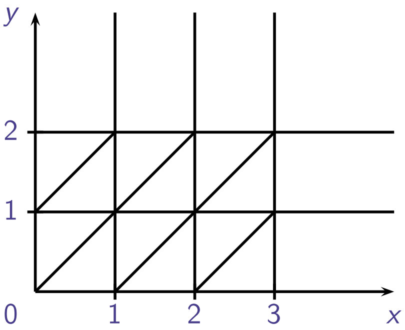
  $r, r' \in R \implies \forall$ constraints $g, \quad r \models g \iff r' \models g$

- "compatible" with time elapsing

  $r, r' \in R \implies$ same delay successor regions

The Region Abstraction

[Alur & Dill, TCS'94]



Build an equivalence relation which is of finite index and is:

- "compatible" with clock constraints ($g ::= x \sim c \quad g \wedge g$)

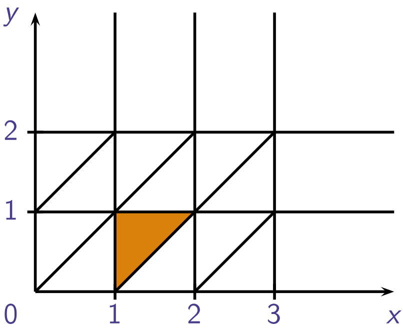
  $r, r' \in R \implies \forall$ constraints $g, \quad r \models g \iff r' \models g$

- "compatible" with time elapsing

  $r, r' \in R \implies$ same delay successor regions

The Region Abstraction

[Alur & Dill, TCS'94]



region defined by

$I_x =]1;2[\; I_y =]0;1[$

$\{x\} < \{y\}$

Build an equivalence relation which is of finite index and is:

- "compatible" with clock constraints $(g ::= x \sim c \quad g \wedge g)$

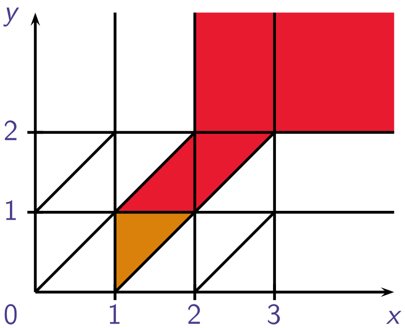
  $r, r' \in R \implies \forall$ constraints $g, \quad r \models g \iff r' \models g$

- "compatible" with time elapsing

  $r, r' \in R \implies$ same delay successor regions

The Region Abstraction

[Alur & Dill, TCS'94]



region defined by
$I_x =]1;2[\; I_y =]0;1[$
$\{x\} < \{y\}$

delay successors

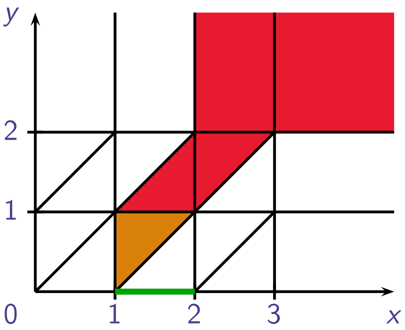
Build an equivalence relation which is of finite index and is:

- "compatible" with clock constraints ($g ::= x \sim c \quad g \wedge g$)

  $r, r' \in R \implies \forall$ constraints $g, \quad r \models g \iff r' \models g$

- "compatible" with time elapsing

  $r, r' \in R \implies$ same delay successor regions



region defined by
$I_x =]1;2[\; I_y =]0;1[$
$\{x\} < \{y\}$

delay successors

successor by reset

Build an equivalence relation which is of finite index and is:

- "compatible" with clock constraints $(g ::= x \sim c \quad g \wedge g)$

  $r, r' \in R \implies \forall$ constraints $g, \quad r \models g \iff r' \models g$

- "compatible" with time elapsing

  $r, r' \in R \implies$ same delay successor regions

The Region Automaton

- For each transition $\ell \xrightarrow{g,a,C:=0} \ell'$ of the TA
- Build transitions in the region automaton RA: $(\ell, R) \xrightarrow{a} (\ell', R')$ if:
  - there exists R'' a delay successor of R s.t.
  - R'' satisfies the guard g ($R'' \subseteq [\![g]\!]$)
  - $R''[C \leftarrow 0]$ is included in R'

a TA and its region automaton RA are time-abstract bisimilar

- The region automaton is finite
- Language accepted by the RA = untimed language accepted by the TA
  a timed word $w = (a, 1.2)(b, 3.4)(a, 6.256)$; $\text{untimed}(w) = aba$
- Language Emptyness can be decided on the RA

The Region Automaton

- For each transition $\ell \xrightarrow{g,a,C:=0} \ell'$ of the TA
- Build transitions in the region automaton RA: $(\ell, R) \xrightarrow{a} (\ell', R')$ if:
  - there exists R'' a delay successor of R s.t.
  - R'' satisfies the guard g ($R'' \subseteq [\![g]\!]$)
  - $R''[C \leftarrow 0]$ is included in R'

> a TA and its region automaton RA are time-abstract bisimilar

- The region automaton is finite
- Language accepted by the RA = untimed language accepted by the TA

  a timed word $w = (a, 1.2)(b, 3.4)(a, 6.256)$; $\text{untimed}(w) = aba$
- Language Emptyness can be decided on the RA

The Region Automaton

- For each transition $\ell \xrightarrow{g,a,C:=0} \ell'$ of the TA
- Build transitions in the region automaton RA: $(\ell, R) \xrightarrow{a} (\ell', R')$ if:
  - there exists R'' a delay successor of R s.t.
  - R'' satisfies the guard g ($R'' \subseteq [\![g]\!]$)
  - $R''[C \leftarrow 0]$ is included in R'

a TA and its region automaton RA are time-abstract bisimilar

- The region automaton is finite
- Language accepted by the RA = untimed language accepted by the TA

  a timed word $w = (a, 1.2)(b, 3.4)(a, 6.256)$; $\text{untimed}(w) = aba$
- Language Emptyness can be decided on the RA

The Region Automaton

- For each transition $\ell \xrightarrow{g,a,C:=0} \ell'$ of the TA
- Build transitions in the region automaton RA: $(\ell, R) \xrightarrow{a} (\ell', R')$ if:
  - there exists R'' a delay successor of R s.t.
  - R'' satisfies the guard g ($R'' \subseteq [\![g]\!]$)
  - $R''[C \leftarrow 0]$ is included in R'

a TA and its region automaton RA are time-abstract bisimilar

- The region automaton is finite
- Language accepted by the RA = untimed language accepted by the TA
  a timed word $w = (a, 1.2)(b, 3.4)(a, 6.256)$; untimed$(w) = aba$
- Language Emptyness can be decided on the RA

The Region Automaton

- For each transition $\ell \xrightarrow{g,a,C:=0} \ell'$ of the TA
- Build transitions in the region automaton RA: $(\ell, R) \xrightarrow{a} (\ell', R')$ if:
  - there exists R'' a delay successor of R s.t.
  - R'' satisfies the guard g ($R'' \subseteq [\![g]\!]$)
  - $R''[C \leftarrow 0]$ is included in R'

a TA and its region automaton RA are time-abstract bisimilar

- The region automaton is finite
- Language accepted by the RA = untimed language accepted by the TA
  a timed word $w = (a, 1.2)(b, 3.4)(a, 6.256)$; $\text{untimed}(w) = aba$
- Language Emptyness can be decided on the RA

The Region Automaton

- For each transition $\ell \xrightarrow{g,a,C:=0} \ell'$ of the TA
- Build transitions in the region automaton RA: $(\ell, R) \xrightarrow{a} (\ell', R')$ if:
  - there exists R'' a delay successor of R s.t.
  - R'' satisfies the guard g ($R'' \subseteq [\![g]\!]$)
  - $R''[C \leftarrow 0]$ is included in R'

a TA and its region automaton RA are time-abstract bisimilar

- The region automaton is finite
- Language accepted by the RA = untimed language accepted by the TA

  a timed word $w = (a, 1.2)(b, 3.4)(a, 6.256)$; untimed$(w) = aba$
- Language Emptyness can be decided on the RA

The Region Automaton

- For each transition $\ell \xrightarrow{g,a,C:=0} \ell'$ of the TA
- Build transitions in the region automaton RA: $(\ell, R) \xrightarrow{a} (\ell', R')$ if:
  - there exists R'' a delay successor of R s.t.
  - R'' satisfies the guard g $(R'' \subseteq [\![g]\!])$
  - $R''[C \leftarrow 0]$ is included in R'

> a TA and its region automaton RA are time-abstract bisimilar

- The region automaton is finite
- Language accepted by the RA = untimed language accepted by the TA
  a timed word $w = (a, 1.2)(b, 3.4)(a, 6.256)$; $\text{untimed}(w) = aba$
- Language Emptyness can be decided on the RA

The Region Automaton

- For each transition $\ell \xrightarrow{g,a,C:=0} \ell'$ of the TA
- Build transitions in the region automaton RA: $(\ell, R) \xrightarrow{a} (\ell', R')$ if:
  - there exists R'' a delay successor of R s.t.
  - R'' satisfies the guard g ($R'' \subseteq [\![g]\!]$)
  - $R''[C \leftarrow 0]$ is included in R'

> a TA and its region automaton RA are time-abstract bisimilar

- The region automaton is finite
- Language accepted by the RA = untimed language accepted by the TA
  a timed word $w = (a, 1.2)(b, 3.4)(a, 6.256)$; untimed$(w) = aba$
- Language Emptyness can be decided on the RA

The Region Automaton

- For each transition $\ell \xrightarrow{g,a,C:=0} \ell'$ of the TA
- Build transitions in the region automaton RA: $(\ell, R) \xrightarrow{a} (\ell', R')$ if:
  - there exists R'' a delay successor of R s.t.
  - R'' satisfies the guard g ($R'' \subseteq [\![g]\!]$)
  - $R''[C \leftarrow 0]$ is included in R'

> a TA and its region automaton RA are time-abstract bisimilar

- The region automaton is finite
- Language accepted by the RA = untimed language accepted by the TA
  a timed word $w = (a, 1.2)(b, 3.4)(a, 6.256)$; $\text{untimed}(w) = aba$
- Language Emptyness can be decided on the RA

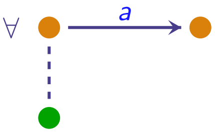
The Region Automaton

- For each transition $\ell \xrightarrow{g,a,C:=0} \ell'$ of the TA
- Build transitions in the region automaton RA: $(\ell, R) \xrightarrow{a} (\ell', R')$ if:
  - there exists R'' a delay successor of R s.t.
  - R'' satisfies the guard g $(R'' \subseteq [\![g]\!])$
  - $R''[C \leftarrow 0]$ is included in R'

> a TA and its region automaton RA are time-abstract bisimilar

- The region automaton is finite
- Language accepted by the RA = untimed language accepted by the TA
  a timed word $w = (a, 1.2)(b, 3.4)(a, 6.256)$; $\text{untimed}(w) = aba$
- Language Emptyness can be decided on the RA

Time-abstract bisimulation



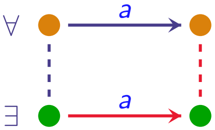
$(\ell_0, v_0) \xrightarrow{a_1, t_1} (\ell_1, v_1) \xrightarrow{a_2, t_2} (\ell_2, v_2) \xrightarrow{a_3, t_3} \cdots$

$(\ell_0, R_0) \xrightarrow{a_1} (\ell_1, R_1) \xrightarrow{a_2} (\ell_2, R_2) \xrightarrow{a_3} \cdots$

with $v_i \in R_i$ for all i.

Timed Auto.

Time-abstract bisimulation



$$(\ell_0, v_0) \xrightarrow{a_1, t_1} (\ell_1, v_1) \xrightarrow{a_2, t_2} (\ell_2, v_2) \xrightarrow{a_3, t_3} \cdots$$

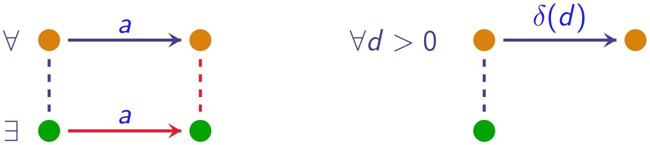
$$\updownarrow \qquad \updownarrow \qquad \updownarrow$$

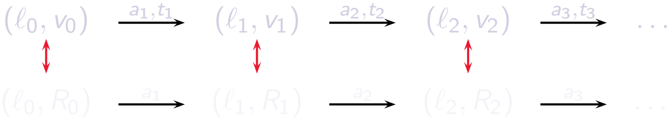
$$(\ell_0, R_0) \xrightarrow{a_1} (\ell_1, R_1) \xrightarrow{a_2} (\ell_2, R_2) \xrightarrow{a_3} \cdots$$

with $v_i \in R_i$ for all i.

◂ Timed Auto.

Time-abstract bisimulation

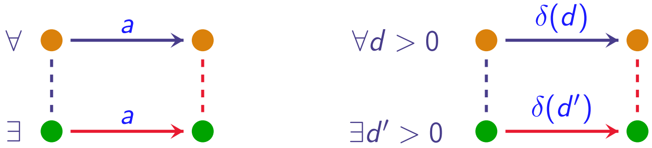




with $v_i \in R_i$ for all i.

◂ Timed Auto.

Time-abstract bisimulation

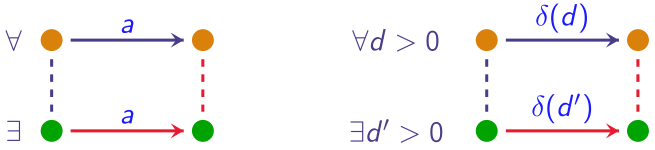


$$
\begin{array}{ccccccc}
(\ell_0, v_0) & \xrightarrow{a_1, t_1} & (\ell_1, v_1) & \xrightarrow{a_2, t_2} & (\ell_2, v_2) & \xrightarrow{a_3, t_3} & \cdots \\
\updownarrow & & \updownarrow & & \updownarrow & & \\
(\ell_0, R_0) & \xrightarrow{a_1} & (\ell_1, R_1) & \xrightarrow{a_2} & (\ell_2, R_2) & \xrightarrow{a_3} & \cdots
\end{array}
$$

with $v_i \in R_i$ for all i.

◂ Timed Auto.

Time-abstract bisimulation



$$(\ell_0, v_0) \xrightarrow{a_1, t_1} (\ell_1, v_1) \xrightarrow{a_2, t_2} (\ell_2, v_2) \xrightarrow{a_3, t_3} \cdots$$

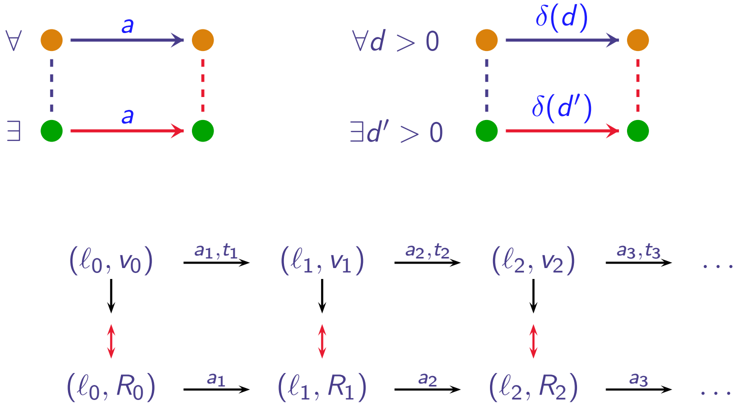
$$\updownarrow \qquad \updownarrow \qquad \updownarrow$$

$$(\ell_0, R_0) \xrightarrow{a_1} (\ell_1, R_1) \xrightarrow{a_2} (\ell_2, R_2) \xrightarrow{a_3} \cdots$$

with $v_i \in R_i$ for all i.

◂ Timed Auto.

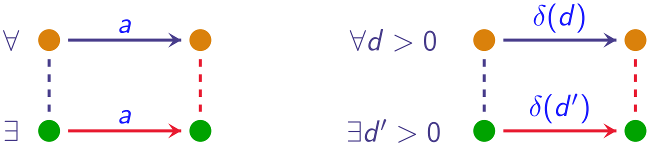
Time-abstract bisimulation



with $v_i \in R_i$ for all i.

Timed Auto.

Time-abstract bisimulation



$$
\begin{array}{ccccccc}
(\ell_0, v_0) & \xrightarrow{a_1, t_1} & (\ell_1, v_1) & \xrightarrow{a_2, t_2} & (\ell_2, v_2) & \xrightarrow{a_3, t_3} & \dots \\
\updownarrow & & \updownarrow & & \updownarrow & & \\
(\ell_0, R_0) & \xrightarrow{a_1} & (\ell_1, R_1) & \xrightarrow{a_2} & (\ell_2, R_2) & \xrightarrow{a_3} & \dots
\end{array}
$$

with $v_i \in R_i$ for all i.

◂ Timed Auto.

Algorithm for Checking Diagnosability

Necessary and Sufficient Condition for Diagnosability:

$$
\begin{aligned}
A \text{ is not } \Sigma\text{-diagnosable} &\iff \forall k \in \mathbb{N}^*, A \text{ is not } (\Sigma, k)\text{-diagnosable} \\
&\iff \forall k \in \mathbb{N}^*, \begin{cases} \exists \rho \in \mathsf{NonFaulty}(A) \\ \exists \rho' \in \mathsf{Faulty}_{\geq k}(A) \\ \qquad \pi_{/\Sigma}(\rho) = \pi_{/\Sigma}(\rho') \end{cases}
\end{aligned}
$$

Algorithm for Checking Diagnosability

Necessary and Sufficient Condition for Diagnosability:

$$A \text{ is not } \Sigma\text{-diagnosable} \iff \forall k \in \mathbb{N}^*, A \text{ is not } (\Sigma, k)\text{-diagnosable}$$

$$\iff \forall k \in \mathbb{N}^*, \begin{cases} \exists \rho \in \mathsf{NonFaulty}(A) \\ \exists \rho' \in \mathsf{Faulty}_{\geq k}(A) \\ \qquad \pi_{/\Sigma}(\rho) = \pi_{/\Sigma}(\rho') \end{cases}$$

Let $A_1 = (Q \times \{0,1\}, (q_0, 0), \Sigma^\varepsilon, \rightarrow_1)$ s.t.

- $(q,k) \xrightarrow{l}_1 (q',k')$ iff $q \xrightarrow{l} q'$ and $l \in \Sigma$ and $k = k'$;
- $(q,k) \xrightarrow{\varepsilon}_1 (q',1)$ iff $q \xrightarrow{f} q'$, (k is set to 1 after a fault occurs and will remain 1 once it has been set to 1);
- $(q,k) \xrightarrow{\varepsilon}_1 (q',k)$ iff $q \xrightarrow{\varepsilon} q'$.

Algorithm for Checking Diagnosability

Necessary and Sufficient Condition for Diagnosability:

$$A \text{ is not } \Sigma\text{-diagnosable} \iff \forall k \in \mathbb{N}^*, A \text{ is not } (\Sigma, k)\text{-diagnosable}$$

$$\iff \forall k \in \mathbb{N}^*, \begin{cases} \exists \rho \in \mathsf{NonFaulty}(A) \\ \exists \rho' \in \mathsf{Faulty}_{\geq k}(A) \\ \qquad \pi_{/\Sigma}(\rho) = \pi_{/\Sigma}(\rho') \end{cases}$$

Define $A_2 = (Q, q_0, \Sigma^\varepsilon, \rightarrow_2)$ with

- $q \xrightarrow{l}_2 q'$ if $q \xrightarrow{l} q'$ and $l \in \Sigma$;
- $q \xrightarrow{\varepsilon}_2 q'$ if $q \xrightarrow{\varepsilon} q'$.

Algorithm for Checking Diagnosability

Necessary and Sufficient Condition for Diagnosability:

$$A \text{ is not } \Sigma\text{-diagnosable} \iff \forall k \in \mathbb{N}^*, A \text{ is not } (\Sigma, k)\text{-diagnosable}$$

$$\iff \forall k \in \mathbb{N}^*, \begin{cases} \exists \rho \in \text{NonFaulty}(A) \\ \exists \rho' \in \text{Faulty}_{\geq k}(A) \\ \qquad \pi_{/\Sigma}(\rho) = \pi_{/\Sigma}(\rho') \end{cases}$$

Let $\mathcal{B} = A_1 \times A_2$

Büchi acceptance condition: infinitely many faulty states and A_1-actions

Algorithm for Checking Diagnosability

Necessary and Sufficient Condition for Diagnosability:

$$A \text{ is not } \Sigma\text{-diagnosable} \iff \forall k \in \mathbb{N}^*, A \text{ is not } (\Sigma, k)\text{-diagnosable}$$

$$\iff \forall k \in \mathbb{N}^*, \begin{cases} \exists \rho \in \mathrm{NonFaulty}(A) \\ \exists \rho' \in \mathrm{Faulty}_{\geq k}(A) \\ \qquad \pi_{/\Sigma}(\rho) = \pi_{/\Sigma}(\rho') \end{cases}$$

Let $\mathcal{B} = A_1 \times A_2$

Büchi acceptance condition: infinitely many faulty states and A_1-actions

Theorem

$\mathrm{Lang}^\omega(\mathcal{B}) \neq \emptyset \iff A$ *is not* Σ*-diagnosable.*

Algorithm for Checking Diagnosability

Necessary and Sufficient Condition for Diagnosability:

$$A \text{ is not } \Sigma\text{-diagnosable} \iff \forall k \in \mathbb{N}^*, A \text{ is not } (\Sigma, k)\text{-diagnosable}$$

$$\iff \forall k \in \mathbb{N}^*, \begin{cases} \exists \rho \in \mathsf{NonFaulty}(A) \\ \exists \rho' \in \mathsf{Faulty}_{\geq k}(A) \\ \qquad \pi_{/\Sigma}(\rho) = \pi_{/\Sigma}(\rho') \end{cases}$$

Let $\mathcal{B} = A_1 \times A_2$

Büchi acceptance condition: infinitely many faulty states and A_1-actions

Theorem

$\mathsf{Lang}^\omega(\mathcal{B}) \neq \emptyset \iff$ *A is not Σ-diagnosable.*

Theorem

The minimum k s.t. A is (Σ, k)-diagnosable can be computed in PTIME.