Contribution to the analysis of Discrete Event Systems

Hervé Marchand

Inria Rennes - Bretagne Atlantique

Habilitation defense - June 6th 2017
Control command systems and software are pervasive
Control command systems and software are pervasive

More and more complex to design, program and operate

Safety critical systems w.r.t. devices, people (transportation, surgery), environment

Increase of security/privacy aspects
Control command systems and software are pervasive

More and more complex to design, program and operate

Safety critical systems w.r.t. devices, people (transportation, surgery), environment

Increase of security/privacy aspects

Need formal methods to ensure the reliability of such systems
Formal Analysis of Discrete Event Systems

- **Model Checking:** verifying that a mathematical model of the system satisfies its specification

\[ \text{Model } G \quad \text{Property } \Phi \quad \Rightarrow \quad \text{@Model Checker } \quad G \models \Phi ? \quad \Rightarrow \quad \text{Yes} \quad \text{No: counter-example} \]
- **Model Checking**: verifying that a mathematical model of the system satisfies its specification
- **Model-Based Test generation**: synthesis of a set of test cases from a model of the system
Formal Analysis of Discrete Event Systems

- **Model Checking**: verifying that a mathematical model of the system satisfies its specification
- **Model-Based Test generation**: synthesis of a set of test cases from a model of the system
- **Diagnosis**: synthesis of a diagnoser which has the ability to detect a fault in a system based on the observation

![Diagram]

- Model $G$
- Fault $\Phi$
- $\emptyset$DS
- Imp $I$
- Observation
- Diagnoser

- Yes: a fault occurred in $I$
- No: no fault occurred in $I$
Formal Analysis of Discrete Event Systems

- **Model Checking**: verifying that a mathematical model of the system satisfies its specification
- **Model-Based Test Generation**: synthesis of a set of test cases from a model of the system
- **Diagnosis**: synthesis of a diagnoser which has the ability to detect a fault in a system based on the observation
- **Controller Synthesis**: enforcing a property $\Phi$ on the system (on which $\Phi$ does not hold) by means of a controller

```
Model $G$  \rightarrow[@DCS]  \rightarrow Controller $C$
```

$\text{Obs. Decision} \implies C\parallel I \models \Phi$

$\text{Imp } I$

Contribution to the analysis of Discrete Event Systems – Hervé Marchand

Model Checking: verifying that a mathematical model of the system satisfies its specification

Model-Based Test generation: synthesis of a set of test cases from a model of the system

Diagnosis: synthesis of a diagnoser which has the ability to detect a fault in a system based on the observation

Controller Synthesis: enforcing a property $\Phi$ on the system (on which $\Phi$ does not hold) by means of a controller

Enforcement correcting possibly incorrect output sequences of a system
Contribution to Formal Analysis of Discrete Event Systems

Model-Based Test generation: synthesis of a set of test cases for a system modelled by symbolic transition system using abstract interpretation technique

IEEE-TSE’07

Diagnosis:
- Unification of various notions of diagnosability
- Diagnosis of transient faults
- Diagnosis of infinite systems

Wodes06, Ifac08, Wodes16, DEDS13a, DEDS15

Controller Synthesis:
- Control of concurrent & distributed systems
  B. Gaudin & G. Kalyon’s PhD Thesis
- Control of symbolic transition system within the synchronous paradigm
  N. Berthier & G. Delaval’s Postdoc
- Decentralized or hierarchical control of DES
  EJC04, IFAC17

Enforcement of real-time properties modeled by Timed Automata
S. Pinisetty’s PhD Thesis
Contribution to Formal Analysis of Discrete Event Systems

- **Model-Based Test generation:** synthesis of a set of test cases for a system modelled by symbolic transition system using abstract interpretation technique

- **Diagnosis:**
  - Unification of various notions of diagnosability
  - Diagnosis of transient faults
  - Diagnosis of infinite systems

- **Controller Synthesis:**
  - Control of concurrent & distributed systems
  - Control of symbolic transition system within the synchronous paradigm
  - Decentralized or hierarchical control of DES

- **Enforcement** of real-time properties modeled by Timed Automata

- **Formal analysis of security properties:**
  - Integrity, availability and confidentiality
  - Use of various formal methods

[IEEE-TSE’07]

[Wodes06,Ifac08]

[Wodes16]

[DEDS13a,DEDS15]

B. Gaudin & G. Kalyon’s PhD Thesis

N. Berthier & G. Delaval’s Postdoc

[EJC04,IFAC17]

S. Pinisetty’s PhD Thesis

J. Dubreil’s PhD Thesis

Observation

???
Contribution to Formal Analysis of Discrete Event Systems

- Model-Based Test generation
- Diagnosis
- Controller Synthesis
- Enforcement
- Formal analysis of security properties
Contribution to Formal Analysis of Discrete Event Systems

- Model-Based Test generation
- Diagnosis
- Controller Synthesis
- Enforcement
- Formal analysis of security properties

Common features in a nut-shell:

- Model-based approach (system & properties)
- Synthesis of components
- Partial observation
- Model-checking techniques
Contribution to Formal Analysis of Discrete Event Systems

- Model-Based Test generation
- Diagnosis
- Controller Synthesis
- Enforcement
- Formal analysis of security properties

Common features in a nut-shell:
- Model-based approach (system & properties)
- Synthesis of components
- Partial observation
- Model-checking techniques
Outline

- Model & Notations
- Diagnosis of Discrete Event Systems
  - Diagnosis of general fault patterns
  - Diagnosis of transient faults
Outline

► Model & Notations

► Diagnosis of Discrete Event Systems
  • Diagnosis of general fault patterns
  • Diagnosis of transient faults

► Controller Synthesis
  • Control of concurrent systems
    - Synchronous communication
  • Control of distributed systems
    - Asynchronous communication
Outline

► Model & Notations
► Diagnosis of Discrete Event Systems
  • Diagnosis of general fault patterns
  • Diagnosis of transient faults
► Controller Synthesis
  • Control of concurrent systems
    - Synchronous communication
  • Control of distributed systems
    - Asynchronous communication
► Formal analysis of security properties
  • Focus on confidentiality properties
  • Use of diagnosis and control techniques
Outline

- Model & Notations
- Diagnosis of Discrete Event Systems
  - Diagnosis of general fault patterns
  - Diagnosis of transient faults
- Controller Synthesis
  - Control of concurrent systems
    - Synchronous communication
  - Control of distributed systems
    - Asynchronous communication
- Formal analysis of security properties
  - Focus on confidentiality properties
  - Use of diagnosis and control techniques
- Conclusion & General Perspectives
Model & Notations

- System modelled by LTS $\mathcal{G} = (Q, \Sigma, \rightarrow, q_0, Q_F)$
  - $Q_F \subseteq Q$: set of marked states
  - $\Sigma$: set of actions

\[
\begin{array}{c}
0 \\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7 \\
\end{array}
\]

- Behaviours
  - $L(\mathcal{G}) = \{f, fa, fab, a, ab, aba, \ldots\}$: language of the system
  - $L_{Q_F}(\mathcal{G}) = \{fab, ab, fabab, \ldots\}$: marked language

- Partial observation
  - $\Pi_o: \Sigma^* \rightarrow \Sigma_o$: natural projection
  - $L_o(\mathcal{G}) = \Pi_o(L(\mathcal{G}))$: observable trajectories
  - $\Pi^{-1}_o: \Sigma^*_o \rightarrow 2\Sigma^*$: inverse projection

- $\epsilon$-closure: $\epsilon_o(\mathcal{G})$ and determinization: $\text{Det}_o(\mathcal{G})$
Model & Notations

- System modelled by LTS $\mathcal{G} = (Q, \Sigma, \rightarrow, q_0, Q_F)$
  - $Q_F \subseteq Q$: set of marked states
  - $\Sigma$: set of actions

- Behaviours
  - $\mathcal{L}(\mathcal{G}) = \{f, fa, fab, a, ab, aba, \cdots\}$: language of the system
  - $\mathcal{L}_{Q_F}(\mathcal{G}) = \{fab, ab, fabab, \cdots\}$: marked language
Model & Notations

- System modelled by LTS $\mathcal{G} = (Q, \Sigma, \rightarrow, q_0, Q_F)$
  - $Q_F \subseteq Q$: set of marked states
  - $\Sigma = \Sigma_o \cup \Sigma_{uo}$: observable & unobservable actions

- Behaviours
  - $\mathcal{L}(\mathcal{G}) = \{f, fa, fab, a, ab, aba, \cdots\}$: language of the system
  - $\mathcal{L}_{Q_F}(\mathcal{G}) = \{fab, ab, fabab, \cdots\}$: marked language

- Partial observation:
  - $\Pi_o : \Sigma^* \rightarrow \Sigma_o$ natural projection
    - $\Pi_o(fab) = ab$
  - $\mathcal{L}_o(\mathcal{G}) = \Pi_o(\mathcal{L}(\mathcal{G}))$: observable trajectories
  - $\Pi_o^{-1} : \Sigma_o^* \rightarrow 2^{\Sigma^*}$ inverse projection
    - $\Pi_o^{-1}(ab) = \{fab, ab\}$
Model & Notations

- System modelled by LTS $\mathcal{G} = (Q, \Sigma, \rightarrow, q_0, Q_F)$
  - $Q_F \subseteq Q$: set of marked states
  - $\Sigma = \Sigma_o \cup \Sigma_{uo}$: observable & unobservable actions

- Behaviours
  - $\mathcal{L}(\mathcal{G}) = \{ f, fa, fab, a, ab, aba, \cdots \}$: language of the system
  - $\mathcal{L}_{Q_F}(\mathcal{G}) = \{ fab, ab, fabab, \cdots \}$: marked language

- Partial observation:
  - $\Pi_o: \Sigma^* \rightarrow \Sigma_o$ natural projection \hspace{1cm} $\Pi_o(fab) = ab$
  - $\mathcal{L}_o(\mathcal{G}) = \Pi_o(\mathcal{L}(\mathcal{G}))$: observable trajectories
  - $\Pi^{-1}_o: \Sigma_o^* \rightarrow 2^{\Sigma^*}$ inverse projection \hspace{1cm} $\Pi^{-1}_o(ab) = \{ fab, ab \}$
  - $\epsilon$-closure: $\epsilon_o(\mathcal{G})$ & determinization: $\text{Det}_o(\mathcal{G})$
Diagnosis Overview

- **On-line diagnosis Problem:**
  - Model of the system $G$
  - Partial observation of the system through $\Sigma_o$

Diagnosability: Ability to detect every fault on the basis of the observed trace
Diagnosis Overview

- **On-line diagnosis Problem:**
  - Model of the system $G$
  - Partial observation of the system through $\Sigma_o$

Diagnosability: Ability to **detect** every fault on the basis of the observed trace

- **Various kind of faults**
  - faulty events, [SSL+95]
  - faulty states [Lin94]
  - $n$ occurrences of a fault, 2 different faults occurred, etc...

---


Diagnosis Overview

- **On-line diagnosis Problem:**
  - Model of the system $G$
  - Partial observation of the system through $\Sigma_o$

Diagnosability: Ability to detect every fault on the basis of the observed trace

- **Various kind of faults**
  - faulty events, [SSL+95]
  - faulty states [Lin94]
  - $n$ occurrences of a fault, 2 different faults occurred, etc...

$\Rightarrow$ Needs for an unification of the notions of diagnosis

---


Contribution to the analysis of Discrete Event Systems – Hervé Marchand

Diagnosis Overview

- **On-line diagnosis Problem:**
  - Model of the system $G$
  - Partial observation of the system through $\Sigma_o$

Diagnosability: Ability to **detect** every fault on the basis of the observed trace

- **Various kind of faults**
  - faulty events, [SSL+95]
  - faulty states [Lin94]
  - $n$ occurrences of a fault, 2 different faults occurred, etc...

⇒ Needs for an **unification** of the notions of diagnosis

Our approach

**Introduction of general Fault Patterns** [Wodes06]

---


Diagnosis of permanent faults

- System $\mathcal{G}$ with $\Sigma = \Sigma_{uo} \cup \Sigma_o$ and a fault pattern $\Omega$ ($\mathcal{L}_\Omega$)

$$
\begin{align*}
q_0 & \xrightarrow{a} q_1 & a & \xrightarrow{a} q_2 & b & \xrightarrow{a} q_3 \\
N & \xrightarrow{f} & F
\end{align*}
$$
Diagnosis of permanent faults

- System \( G \) with \( \Sigma = \Sigma_{uo} \cup \Sigma_o \) and a fault pattern \( \Omega \) (\( \mathcal{L}_\Omega \))

- Diagnoser: \( \Delta : \mathcal{L}_o(G) \rightarrow \{N, F, U\} \) where

\[
\Delta(w) = \begin{cases} 
F & \text{iff a fault occurred after } w \\
N & \text{iff no fault occurred after } w \\
U & \text{otherwise}
\end{cases}
\]
Diagnosis of permanent faults

- System $G$ with $\Sigma = \Sigma_{uo} \cup \Sigma_o$ and a fault pattern $\Omega (L_\Omega)$

- Diagnoser: $\Delta : L_o(G) \rightarrow \{N, F, U\}$ where

\[
\Delta(w) = \begin{cases} 
  F & \text{iff a fault occurred after } w \\
\end{cases}
\]
Diagnosis of permanent faults

- System $G$ with $\Sigma = \Sigma_{u0} \cup \Sigma_o$ and a fault pattern $\Omega$ ($L_\Omega$)

- Diagnoser: $\Delta : L_o(G) \rightarrow \{N, F, U\}$ where

$$\Delta(w) = \begin{cases} 
F & \text{iff a fault occurred after } w \\
N & \text{iff no fault occurred after } w 
\end{cases}$$
Diagnosis of permanent faults

- System $\mathcal{G}$ with $\Sigma = \Sigma_{uo} \cup \Sigma_o$ and a fault pattern $\Omega (\mathcal{L}_\Omega)$

$$\Sigma \setminus \{f\}$$

- Diagnoser: $\Delta : \mathcal{L}_o(\mathcal{G}) \rightarrow \{N, F, U\}$ where

$$\Delta(w) = \begin{cases} 
F & \text{iff a fault occurred after } w \\
N & \text{iff no fault occurred after } w \\
U & \text{otherwise}
\end{cases}$$
Diagnosis of permanent faults

- System $\mathcal{G}$ with $\Sigma = \Sigma_{uo} \cup \Sigma_{o}$ and a fault pattern $\Omega$ ($\mathcal{L}_{\Omega}$)

- $\mathcal{D}_o$:
  $$\Delta : \mathcal{L}_o(G) \rightarrow \{N, F, U\}$$

  $\Delta(w) = \begin{cases} 
  F & \text{iff a fault occurred after } w \\
  N & \text{iff no fault occurred after } w \\
  U & \text{otherwise}
  \end{cases}$

  $\Rightarrow$ derived from $\text{Det}_o(G||\Omega)$
Diagnosis of permanent faults

- System $\mathcal{G}$ with $\Sigma = \Sigma_{uo} \cup \Sigma_o$ and a fault pattern $\Omega$ ($\mathcal{L}_\Omega$)

$$\Sigma \setminus \{f\}$$

- Diagnoser: $\Delta : \mathcal{L}_o(\mathcal{G}) \rightarrow \{N, F, U\}$ where

$$\Delta(w) = \begin{cases} 
F & \text{iff a fault occurred after } w \\
N & \text{iff no fault occurred after } w \\
U & \text{otherwise}
\end{cases}$$

$\Rightarrow$ derived from $Det_o(\mathcal{G} \parallel \Omega)$

$\Rightarrow$ Boundedness of the detection delay?

Diagnosis of permanent faults

**Diagnosability**

Ability to detect every fault pattern *at most* \( n \) observations after its occurrence

- **Non-diagnosability**: existence of two arbitrary long and observationally equivalent sequences one faulty, the other non-faulty
- Test of diagnosability on the twin machine \( \varepsilon(G_\Omega) || \varepsilon(G_\Omega) \)

\[
\begin{align*}
& q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3 \\
& g_\|_\Omega
\end{align*}
\]

\[
\begin{align*}
& q_0, q_0 \xrightarrow{a} q_2, q_2 \\
& q_2, q_0 \xrightarrow{a} q_3, q_0, q_2
\end{align*}
\]

\[
\Rightarrow \text{ Non diagnosable if there exists a reachable ambiguous cycle}
\]

- Extension to the predictability of fault patterns [Ifac08]
Diagnosis of non-permanent faults

Faults might be repaired


Diagnosis of non-permanent faults

Faults might be repaired

Various notions of Diagnosability

- O-Diagnosability: a fault occurred in the system  [CLT04]
- P-Diagnosability: the system has been faulty and is now faulty  [CLT04]
- [1..K]-Diagnosability: at least K faults occurred in the system.  [JKG03]


Diagnosis of non-permanent faults

Faults might be repaired

Various notions of Diagnosability

- O-Diagnosability: a fault occurred in the system [CLT04]
- P-Diagnosability: the system has been faulty and is now faulty [CLT04]
- [1..K]-Diagnosability: at least K faults occurred in the system [JKG03]

Our approach

Ability to detect every occurrence of a fault before it is repaired and to count the number of faults

Diagnoser

Similar to the one for permanent faults


T-diagnosability

T-diagnosability w.r.t. $F$

Ability to surely detect every fault before its repair in a bounded number of observations on the basis of the observed trace
T-diagnosability

T-diagnosability w.r.t. $F$

Ability to surely detect every fault \textit{before its repair} in a bounded number of observations on the basis of the observed trace
T-diagnosability

T-diagnosability w.r.t. $F$

Ability to surely detect every fault before its repair in a bounded number of observations on the basis of the observed trace

Verification

- Twin-machine no more sufficient
**T-diagnosability**

**T-diagnosability w.r.t. \( F \)**

Ability to surely detect every fault before its repair in a bounded number of observations on the basis of the observed trace.

- **Verification**
  - Twin-machine no more sufficient
  - Need to check \( G \times Det(G) \)

- **Theorem**
  Deciding whether \( G \) is T-diagnosable w.r.t. \( F \) is PSPACE-complete.

[Reference: Wodes16]
Also need to detect all the repairs

Proposition

If $G$ is $T$-Diagnosable w.r.t. $F$ and $T$-Diagnosable w.r.t. $N$, then one can exactly count the number of faults that occurred in the system.
T-diagnosability & Counting

Also need to detect all the repairs
T-diagnosability & Counting

Also need to detect all the repairs

Proposition

If $G$ is T-Diagnosable w.r.t. $F$ and T-Diagnosable w.r.t. $N$, then one can exactly count the number of faults that occurred in the system.
Conclusion & Perspectives

▶ Summary

- General fault patterns for diagnosability and predictability
- New framework for non-permanent faults
  - Introduction of the notion of T-Diagnosability
  - Ability to count the number of faults
  - P-Diagnosability ([CLT04]) is also PSPACE-Complete

▶ Perspectives

- Active Diagnosis (on-going work with L. Hélouët)
  ⇒ Ability to perform tests allowing to partially disambiguate the set of configurations with energy constraints
- Modular systems $G_1 \parallel G_2$
  - Few results [CLT06], [YD10] (fault events local to a single component)
  - General Fault Pattern?
    - Know-how of the modular control synthesis
    - Add communication between diagnosers
- Diagnosis of LTS handling data

Controller Synthesis

- Control of concurrent systems
  - Synchronous communication
- Control of distributed systems
  - Asynchronous communication
Controller Synthesis Problem [RW89]

- model of the system $G$
- a safety property $K \subseteq \Sigma^*$
- $\Sigma = \Sigma_c \cup \Sigma_{uc}$ (controllable & uncontrollable events)

Compute a **maximal** controller such that $\mathcal{L}(C/G) \subseteq K$
Overview & Problematic

- **Controller Synthesis Problem [RW89]**
  - model of the system \( G \)
  - a safety property \( \mathcal{K} \subseteq \Sigma^* \)
  - \( \Sigma = \Sigma_c \cup \Sigma_{uc} \) (controllable & uncontrollable events)

Compute a **maximal** controller such that \( \mathcal{L}(\mathcal{C}/G) \subseteq \mathcal{K} \)

- **Controllability [RW89]**: \( \mathcal{K} \) is controllable for a controller \( \mathcal{C} \) is no uncontrollable action has to be forbidden in \( \mathcal{L}(G) \) to respect \( \mathcal{K} \).

Overview & Problematic

Controller Synthesis Problem [RW89]
- model of the system \( \mathcal{G} \)
- a safety property \( \mathcal{K} \subseteq \Sigma^* \)
- \( \Sigma = \Sigma_c \cup \Sigma_{uc} \) (controllable & uncontrollable events)

Compute a maximal controller such that \( \mathcal{L}(\mathcal{C}/\mathcal{G}) \subseteq \mathcal{K} \)

Controllability [RW89]: \( \mathcal{K} \) is controllable for a controller \( \mathcal{C} \) is no uncontrollable action has to be forbidden in \( \mathcal{L}(\mathcal{G}) \) to respect \( \mathcal{K} \).

\[ \Rightarrow \text{if } \mathcal{K} \text{ not controllable, existence of a maximal language: } \text{Sup} C_{\mathcal{K}} = \mathcal{L}(\mathcal{C}/\mathcal{G}) = \mathcal{K} \setminus [(\mathcal{L}(\mathcal{G}) \setminus \mathcal{K})/\Sigma_{uc}]\Sigma^* \]

Overview & Problematic

- Controller Synthesis Problem [RW89]
  - model of the system $G$
  - a safety property $\mathcal{K} \subseteq \Sigma^*$
  - $\Sigma = \Sigma_c \cup \Sigma_{uc}$ (controllable & uncontrollable events)

Compute a maximal controller such that $\mathcal{L}(C/G) \subseteq \mathcal{K}$

- Controllability [RW89]: $\mathcal{K}$ is controllable for a controller $C$ if no uncontrollable action has to be forbidden in $\mathcal{L}(G)$ to respect $\mathcal{K}$.

  $\Rightarrow$ if $\mathcal{K}$ not controllable, existence of a maximal language :
  $$SupC_\mathcal{K} = \mathcal{L}(C/G) = \mathcal{K} \setminus [(\mathcal{L}(G) \setminus \mathcal{K})/\Sigma_{uc}^*]\Sigma^*$$

- Problematic:
  - Large systems are often composed of several subsystems $G = G_1 \parallel \cdots \parallel G_n$ $\Rightarrow$ State space explosion!
  - Given a property $\mathcal{K} \subseteq \Sigma^*$, how to compute a (set of) controller(s) ensuring $\mathcal{K}$ without building $G$.

Control of concurrent systems

- $G = G_1 \parallel \cdots \parallel G_n$ with $\mathcal{L}(G_i) \subseteq \Sigma_i^*$ and a safety property $\mathcal{K} \subseteq \Sigma^*$.
  - $\Sigma_i = \Sigma_{i,c} \cup \Sigma_{i,uc}$
  - $\Sigma_s$: Shared events
Control of concurrent systems

Benoit Gaudin’s PhD Thesis

$G = G_1 \parallel \cdots \parallel G_n$ with $\mathcal{L}(G_i) \subseteq \Sigma_i^*$ and a safety property $\mathcal{K} \subseteq \Sigma^*$.

- $\Sigma_i = \Sigma_{i,c} \cup \Sigma_{i,uc}$
- $\Sigma_s$ : Shared events

Related work

- WH91 : Separable specification
- DC00 : $\Sigma_s = \emptyset$
- RL03 : Identical components

---

DC00 M. H. De Queiroz and J. Cury. Modular control of composed systems. ACC , pages 4051- 4055 2000.
RL03 K. Rohloff and S. Lafortune. The control and verification of similar agents operating in a broadcast network environment. In 42nd IEEE CDC, 2003.
Control of concurrent systems

\[ G = G_1 \| \cdots \| G_n \text{ with } L(G_i) \subseteq \Sigma_i^* \text{ and a safety property } \mathcal{K} \subseteq \Sigma^*. \]

- \( \Sigma_i = \Sigma_{i,c} \cup \Sigma_{i,uc} \)
- \( \Sigma_s \) : Shared events

**Related work**

- WH91 : Separable specification
- DC00 : \( \Sigma_s = \emptyset \)
- RL03 : Identical components

**Our approach**

- Abstract systems \( \Pi_i^{-1}(G_i) = G_i^{-1} \)
- \( \mathcal{K}_i = \mathcal{K} \cap L(G_i^{-1}) \)

\[ L(C_1/G_1^{-1}) \cap L(C_2/G_2^{-1}) = L(C/G) \]

Contribution to the analysis of Discrete Event Systems – Hervé Marchand

Control of concurrent systems (cont’d)

\( K' \subseteq K_i \) is partially controllable w.r.t. \( \Sigma_{i,uc}, \Sigma_{uc}, L(G^{-1}_i) \) if

(i) \( K' \) is controllable w.r.t \( \Sigma_{i,uc} \) and \( L(G^{-1}_i) \).

(ii) \( K' \) is controllable w.r.t \( \Sigma_{uc} \) and \( K_i \).
Control of concurrent systems (cont’d)

\[ K' \subseteq K_i \text{ is partially controllable w.r.t. } \Sigma_{i,uc}, \Sigma_{uc}, L(G_i^{-1}) \text{ if } \]

(i) \( K' \) is controllable w.r.t \( \Sigma_{i,uc} \) and \( L(G_i^{-1}) \).
(ii) \( K' \) is controllable w.r.t \( \Sigma_{uc} \) and \( K_i \).

There exists a unique supremal sub-language of \( K_i \) that is partially controllable: \( \text{SupPC}_i \)
Control of concurrent systems (cont’d)

\( \mathcal{K}' \subseteq \mathcal{K}_i \) is partially controllable w.r.t. \( \Sigma_{i,uc}, \Sigma_{uc}, \mathcal{L}(G_i^{-1}) \) if

(i) \( \mathcal{K}' \) is controllable w.r.t \( \Sigma_{i,uc} \) and \( \mathcal{L}(G_i^{-1}) \).

(ii) \( \mathcal{K}' \) is controllable w.r.t \( \Sigma_{uc} \) and \( \mathcal{K}_i \).

There exists a unique supremal sub-language of \( \mathcal{K}_i \) that is partially controllable: \( \text{SupPC}_i \)

**Theorem**

\[ \bigcap_{i \leq n} \text{SupPC}_i \text{ is controllable w.r.t. } \Sigma_{uc} \text{ and } \mathcal{L}(G) \]
Control of concurrent systems (cont’d)

\[ K' \subseteq K_i \text{ is partially controllable w.r.t. } \Sigma_{i,uc}, \Sigma_{uc}, \mathcal{L}(G_i^{-1}) \text{ if} \]

(i) \( K' \) is controllable w.r.t \( \Sigma_{i,uc} \) and \( \mathcal{L}(G_i^{-1}) \).

(ii) \( K' \) is controllable w.r.t \( \Sigma_{uc} \) and \( K_i \).

There exists a unique supremal sub-language of \( K_i \) that is partially controllable: \( \text{SupPC}_i \)

**Theorem**

\[ \bigcap_{i \leq n} \text{SupPC}_i \text{ is controllable w.r.t. } \Sigma_{uc} \text{ and } \mathcal{L}(G) \]

\[ \mathcal{L}(C_i/G_i^{-1}) = \text{SupPC}_i \]
Control of concurrent systems (cont’d)

\( \mathcal{K}' \subseteq \mathcal{K}_i \) is partially controllable w.r.t. \( \Sigma_{i,uc}, \Sigma_{uc}, \mathcal{L}(G_i^{-1}) \) if

(i) \( \mathcal{K}' \) is controllable w.r.t \( \Sigma_{i,uc} \) and \( \mathcal{L}(G_i^{-1}) \).
(ii) \( \mathcal{K}' \) is controllable w.r.t \( \Sigma_{uc} \) and \( \mathcal{K}_i \).

There exists a unique supremal sub-language of \( \mathcal{K}_i \) that is partially controllable: \( \text{SupPC}_i \)

**Theorem**

\( \bigcap_{i \leq n} \text{SupPC}_i \) is controllable w.r.t. \( \Sigma_{uc} \) and \( \mathcal{L}(G) \)

\( \mathcal{L}(C_i / G_i^{-1}) = \text{SupPC}_i \)

Maximality?
Control of concurrent systems (cont’d)

\( K' \subseteq K_i \) is partially controllable w.r.t. \( \Sigma_{i,uc}, \Sigma_{uc}, L(G_i^{-1}) \) if

(i) \( K' \) is controllable w.r.t \( \Sigma_{i,uc} \) and \( L(G_i^{-1}) \).

(ii) \( K' \) is controllable w.r.t \( \Sigma_{uc} \) and \( K_i \).

There exists a unique supremal sub-language of \( K_i \) that is partially controllable: \( SupPC_i \)

**Theorem**

\( \bigcap_{i \leq n} SupPC_i \) is controllable w.r.t. \( \Sigma_{uc} \) and \( L(G) \)

\[ L(C_i/G_i^{-1}) = SupPC_i \]

**Maximality?** : \( \Sigma_s \subseteq \Sigma_c \) and (\( K \subseteq L(G) \) or \( K \) locally consistent)
Outline

- Controller Synthesis
  - Control of concurrent systems
    - Synchronous communication
  - Control of distributed systems
    - Asynchronous communication
Distributed control of distributed systems $(\mathcal{T}_i)_i$

- Embedded systems, protocols
- Asynchronous communication
- CFSM $= \text{LTSs} + \text{FIFO channels}$

CFSM $\mathcal{T}_1$

$A_0$  
$A_1$  
$A_2$

CFSM $\mathcal{T}_2$

$B_0$  
$B_1$  
$B_2$  
$B_3$

CFSM $\mathcal{T}_3$

$D_0$  
$D_1$

Problem: computing $(C_i)_i$ s.t. $(\mathcal{T}_i \parallel C_i)_i | = \Phi$ (safety)

Bad $\subseteq Q_1 \times \cdots \times Q_n \times \{\text{Contents of the channels}\}$

Control mechanism for $C_i$

- $\Sigma_i, c$ = set of local outputs
- Local control decisions based on a state estimate $E_i$ $\Rightarrow$ Refinement of $E_i$ by piggybacking information to the sent messages (logical clock + peer state estimate)
Distributed control of distributed systems $(\mathcal{T}_i)_i$

- Embedded systems, protocols
- Asynchronous communication
- CFSM = LTSs + FIFO channels

Problem: computing $(C_i)_i$ s.t. $(\mathcal{T}_i \parallel C_i)_i \models \Phi$ (safety)
Distributed control of distributed systems $(T_i)_i$

- Embedded systems, protocols
- Asynchronous communication
- CFSM = LTSs + FIFO channels

Problem: computing $(C_i)_i$ s.t. $(T_i || C_i)_i \models \Phi$ (safety)

$Bad \subseteq Q_1 \times \cdots \times Q_n \times \{ \text{Contents of the channels} \}$
Control of distributed systems

Distributed control of distributed systems \((\mathcal{T}_i)_i\)

- Embedded systems, protocols
- Asynchronous communication
- CF$\text{FSM} = \text{LTSs + FIFO channels}$

Problem: computing \((C_i)_i\) s.t. \((\mathcal{T}_i || C_i)_i = \Phi\) (safety)

- \(Bad \subseteq Q_1 \times \cdots \times Q_n \times \{\text{Contents of the channels}\}\)
- Control mechanism for \(C_i\)
  - \(\Sigma_{i,c} = \text{set of local outputs}\)
  - Local control decisions based on a state estimate \(E_i\)
Control of distributed systems

Gabriel Kalyon’s PhD Thesis
[Forte11], [CDC’11], [IEEE-TAC’14]

Distributed control of distributed systems \((\mathcal{T}_i)_i\)

- Embedded systems, protocols
- Asynchronous communication
- \(\text{CFSM} = \text{LTSs} + \text{FIFO channels}\)

Problem : computing \((C_i)_i\) s.t. \((\mathcal{T}_i||C_i)_i \models \Phi\) (safety)

- \(Bad \subseteq Q_1 \times \cdots \times Q_n \times \{\text{Contents of the channels}\}\)
- Control mechanism for \(C_i\)
  - \(\Sigma_{i,c} = \text{set of local outputs}\)
  - Local control decisions based on a state estimate \(E_i\)
  - Refinement of \(E_i\) by piggybacking information to the sent messages (logical clock + peer state estimate)
Control of distributed systems (Cont’d)

- **Off-Line computation**
  - Approximation of the set of states reaching $Bad$ by uncontrollable events.
  - Adaptation of approximate verification for CFSMs [LGJJ06]

$\Rightarrow$ Over-approximation of the contents of the queues by regular languages

Control of distributed systems (Cont’d)

► Off-Line computation
  ● Approximation of the set of states reaching *Bad* by uncontrollable events.
  ● Adaptation of approximate verification for CFSMs [LGJJ06]

⇒ Over-approximation of the contents of the queues by regular languages

\[ \text{[LGJJ06]: T. Le Gall, B. Jeannet, and T. Jéron. Verification of communication protocols using abstract interpretation of FIFO queues. AMAST’06, July 2006.} \]
Control of distributed systems (Cont’d)

► Off-Line computation
  ● Approximation of the set of states reaching $Bad$ by uncontrollable events.
  ● Adaptation of approximate verification for CFSMs [LGJJ06]

⇒ Over-approximation of the contents of the queues by regular languages

Control of distributed systems (Cont’d)

**Off-Line computation**
- Approximation of the set of states reaching \textit{Bad} by uncontrollable events.
- Adaptation of approximate verification for CFSMs [LGJJ06]
  \[\Rightarrow\] Over-approximation of the contents of the queues by regular languages

**On-Line computation for \(C_i\)**
- Reception of a message from site \(j\) :
  \[E'_i = \text{Post}_a(E_i) \cap f(V_j, E_j)\]
- Transmission of a message \((a, V'_i, E'_i)\) :
  \[E'_i = \text{Reach}_{\Delta \setminus \Delta_i}(\text{Post}_a(E_i))\]
  \[\Rightarrow\] Computation of the new set of forbidden events w.r.t. \(E'_i\)

With the above computations, the controlled system avoids \textit{bad} and is sound

Conclusion & Perspectives

► Summary
  • Control of concurrent systems
    - Computation of local controllers w.r.t. abstract sub-systems
    - Specialisation to a state-based approach
  • Control of distributed systems
    - Use of abstract interpretation to techniques.

► Perspectives
  • Concurrent & distributed systems:
    - Non-blocking properties (liveness, etc)
    - Minimizing the exchanged information

• Quantitative properties
  Work with N. Berthier, G. Delaval & E. Rutten

Contribution to the analysis of Discrete Event Systems – Hervé Marchand
June 6th 2017 – Habilitation Defense – 22/29
Outline

- Model & Notations
  - Diagnosis of general fault patterns
  - Diagnosis of transient faults

- Model $G$
- Fault Pattern $\Phi$
- Diagnoser $\Pi$
- Assuming $G \sim I$

- Controller Synthesis
  - Control of concurrent systems
    - Synchronous communication
  - Control of distributed systems
    - Asynchronous communication

- Formal analysis of security properties
  - Focus on confidentiality properties
  - Use of diagnosis and control techniques

- Conclusion & General Perspectives

Contribution to the analysis of Discrete Event Systems – Hervé Marchand

Confidential information: secret properties

- State configurations (values of secret variables)
- Occurrence of an event (e.g. password file is unlocked)

Information flow: an attacker is able to infer confidential information based on his observation and its knowledge of the system.

The secret $S$ is opaque w.r.t. $G$ and $\Sigma_o$ if $\forall t \in L_S(G)$, $\exists s \in L_G(G) \L_S(G)$: $\Pi_{o^{-1}}(s) = \Pi_{o^{-1}}(t)$.

$\Sigma = \{h, p, a, b\}$, $\Sigma_o = \{a, b\}$, $S = \{2, 4, 5\}$, $L_S = \Sigma^* h \Sigma^*$.

Confidential information

- **Confidential information**: secret properties
  - State configurations (values of secret variables)
  - Occurrence of an event (e.g. password file is unlocked), Trajectories

---

Confidential information

- **Confidential information**: secret properties
  - State configurations (values of secret variables)
  - Occurrence of an event (e.g. password file is unlocked), Trajectories
- **Information flow**: an attacker is able to infer confidential information based on his observation and its knowledge of the system.

\[ \text{System } G \xrightarrow{\Sigma_o} \text{Attacker } A \]

---

Confidential information

- **Confidential information**: secret properties
  - State configurations (values of secret variables)
  - Occurrence of an event (e.g. password file is unlocked), Trajectories
- **Information flow**: an attacker is able to infer confidential information based on his observation and its knowledge of the system.

The secret $S$ is opaque w.r.t. $\mathcal{G}$ and $\Sigma_o$ if

$$\forall t \in L_S(\mathcal{G}), \exists s \in L(\mathcal{G}) \setminus L_S(\mathcal{G}) : \Pi_{o}^{-1}(s) = \Pi_{o}^{-1}(t)$$

\[\Sigma = \{h, p, a, b\}, \quad \Sigma_o = \{a, b\}\]

\[S = \{2, 4, 5\}, \quad L_S = \Sigma^* \cdot h \cdot \Sigma^*\]
**Verification of Opacity**

**Algorithm**

1. Determinization of $G$
2. Check whether a macro-state $F \subseteq S$ is reachable

**Theorem**

*Checking opacity is PSPACE complete*  

$\Rightarrow$ Reduction from universality problem

[Amast08]
Monitoring an information flow

\[ M \xrightarrow{\Sigma_m} G \xleftarrow{\Sigma_o} A \]

Observations leading to an information flow:

\[ \Sigma_o = \{a, b\} \]

\[ f_{a, c}a, b, c \]

M should diagnose/predict the occurrence of the trajectories:

\[ I = L(G) \cap (\Sigma_o(F)) \]

No \[ \Sigma_m = \{a, c\} \]

\[ M \Rightarrow \{P_{\Sigma_o(s)} = aab, P_{\Sigma_m(s)} = aac\} \]

Every information flow is detectable iff \[ I \] is diagnosable w.r.t. \[ \Sigma_m \].

[ ECC2009 ]
Monitoring an information flow

\[ F = \mathcal{L}_{2s}(\text{Det}_{\Sigma_o}(G)) \]: observations leading to an information flow

\[ \Sigma_o = \{a, b\} \]

\[ \text{Det}_{\Sigma_o}(G) \]

\[ \Sigma_o = \{a, b\} \]
Monitoring an information flow

\[ F = \mathcal{L}_{2^s}(\text{Det}_{\Sigma_o}(G)) \]: observations leading to an information flow

\[ M \] should diagnose/predict the occurrence of the trajectories

\[ I = \mathcal{L}(G) \cap P_{\Sigma_o}^{-1}(F).\Sigma^* \]

by observing \( \Sigma_m \).

\[ \Sigma_m = \{a, c\} \]

\[ s = aabc \Rightarrow \begin{cases} P_{\Sigma_o}(s) = aab \\ P_{\Sigma_m}(s) = aac \end{cases} \]
Monitoring an information flow

\[ \mathcal{F} = \mathcal{L}_{2s}(\text{Det}_{\Sigma_o}(G)) \]: observations leading to an information flow

\[ \Sigma_o = \{a, b\} \]

\[ \text{Det}_{\Sigma_o}(G) \]

\[ a, c \]

\[ b, c \]

\[ \Sigma_o = \{a, b\} \]

\[ 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \]

\[ M \]

\[ \Sigma_m = \{a, c\} \]

\[ \text{No} \]

\[ s = aabc \Rightarrow \left\{ \begin{array}{l} P_{\Sigma_o}(s) = aab \\ P_{\Sigma_m}(s) = aac \end{array} \right. \]

\[ \Rightarrow \text{Every information flow is detectable iff } \mathcal{I} \text{ is diagnosable w.r.t. } \Sigma_m.\]
Control problem (Language-Based)

Compute a maximally permissive controller $C$ observing $\Sigma_m$ and controlling $\Sigma_c \subseteq \Sigma_m$ s.t. $\mathcal{L}_\varphi$ is opaque w.r.t. $G \times C$ and $\Sigma_o$.

$$\Sigma_o = \{a, b, d, e\}$$
$$\Sigma_m = \{a, c_1, c_2, b, d, e\}$$
$$\Sigma_c = \{b, c_1, c_2, e\}$$

Secret: $\mathcal{L}_\varphi = \Sigma^* . h . \Sigma^*$
Supervisory Control for opacity

Control problem (Language-Based)

Compute a maximally permissive controller $C$ observing $\Sigma_m$ and controlling $\Sigma_c \subseteq \Sigma_m$ s.t. $L_\varphi$ is opaque w.r.t. $G \times C$ and $\Sigma_o$.

$$\Sigma_o = \{a, b, d, e\}$$
$$\Sigma_m = \{a, c_1, c_2, b, d, e\}$$
$$\Sigma_c = \{b, c_1, c_2, e\}$$

Secret: $L_\varphi = \Sigma^* \cdot h \cdot \Sigma^*$
Supervisory Control for opacity

[Wodes08], [IEEE-TAC10]

Control problem (Language-Based)

Compute a maximally permissive controller $C$ observing $\Sigma_m$ and controlling $\Sigma_c \subseteq \Sigma_m$ s.t. $L_\varphi$ is opaque w.r.t. $G \times C$ and $\Sigma_o$.

\[ \Sigma_o = \{a, b, d, e\} \]
\[ \Sigma_m = \{a, c_1, c_2, b, d, e\} \]
\[ \Sigma_c = \{b, c_1, c_2, e\} \]

Secret: $L_\varphi = \Sigma^* . h . \Sigma^*$
Control problem (Language-Based)

Compute a maximally permissive controller $C$ observing $\Sigma_m$ and controlling $\Sigma_c \subseteq \Sigma_m$ s.t. $L_\varphi$ is opaque w.r.t. $G \times C$ and $\Sigma_o$.

$\Sigma_o = \{a, b, d, e\}$
$\Sigma_m = \{a, c_1, c_2, b, d, e\}$
$\Sigma_c = \{b, c_1, c_2, e\}$

Secret: $L_\varphi = \Sigma^* . h . \Sigma^*$

Theorem

If $\Sigma_c \subseteq \Sigma_m$ and $\Sigma_m \subseteq \Sigma_o$, or $\Sigma_o \subseteq \Sigma_m$, there exists a maximal controller $C$ s.t. $L_\varphi$ is opaque w.r.t. $G \times C$ and $\Sigma_o$
Conclusion & Perspectives

▶ Summary
- Monitoring information flows
- Ensuring Opacity by control
- Ensuring Opacity Dynamic Filtering

▶ Perspectives
- Opacity control problem
  - Remove the assumption $\Sigma_m \subseteq \Sigma_o$, or $\Sigma_o \subseteq \Sigma_m$ [TML+16]
  - Distributed control of Concurrent secrets

preliminary work in [BBB+07]

- Active Attacker
  - Ability to change the observable status of events (c.f. [KWK16])

- Opacity of systems handling data


General perspectives

Towards more flexible systems and requirements
General perspectives

Towards more flexible systems and requirements

off-the shelf components

- Adaptive controllers valid for a set of "similar" devices

⇒ Modal transition systems?
General perspectives

Towards more flexible systems and requirements

off-the-shelf components

- Adaptive controllers valid for a set of "similar" devices
  ⇒ Modal transition systems?

Control under failures

- switch between controllers reacting to diagnosed failures
  ⇒ Needs for a coordinator
General perspectives

Towards more flexible systems and requirements

off-the shelf components
- Adaptive controllers valid for a set of "similar" devices
  ⇒ Modal transition systems?

Control under failures
- switch between controllers reacting to diagnosed failures
  ⇒ Needs for a coordinator

Reconfigurable systems: IoT, Cloud Center, Automotive, etc
General perspectives

Towards more flexible systems and requirements

off-the shelf components
- Adaptive controllers valid for a set of "similar" devices
  ⇒ Modal transition systems?

Control under failures
- switch between controllers reacting to diagnosed failures
  ⇒ Needs for a coordinator

Reconfigurable systems: IoT, Cloud Center, Automotive, etc
- Components can leave or join the system (highly distributed)
  ⇒ Detect and learn the specification of this component
- Adaptation of the requirements (safety, privacy, etc)
  ⇒ on-line computation and deployment of new controllers

Revisit the Control and Diagnosis theory
Ensuring opacity by Dynamic filtering

\[ u \in \Sigma^* \rightarrow D(u) \rightarrow \text{Attacker } \mathcal{A} \]

- **Static Filter**: \( \Sigma_o = \{a\} \) or \( \Sigma_o = \{b\} \) \( \Rightarrow \) \( S \) is opaque
- **Dynamic Filter**: Hide “b” after the observation of an ”a” and keep everything observable after the observation of an ”a”

**Problem**

Computing a Dynamic filter so that \( S \) is opaque w.r.t. \( G \times D \) and \( \Sigma_o \)

Ensuring opacity by Dynamic filtering

- Reduction to a safety 2-player game

(a): the LTS $G$

(b): The game LTS

⇒ Solving the Dynamic filter problem is EXPTIME
Ensuring opacity by Dynamic filtering

- Reduction to a safety 2-player game

(a): the LTS $G$

(b): The game LTS

⇒ Solving the Dynamic filter problem is EXPTIME