Control Problems for DES are Model-Checking Problems

Sophie Pinchinat
IRISA-INRIA
Campus de Beaulieu
F-35042, Rennes - France
Sophie.Pinchinat@irisa.fr

Stéphane Riedweg
joint work with
LSV, ENS Cachan
61, avenue du Président Wilson
F-94235 CACHAN Cedex - France
Stephane.Riedweg@lsv.ens-cachan.fr
Plan

- Discrete Events Systems (DES)
- Control Problems for DES
- An overview of the Approach
  
  Control Problems are Model-Checking problems

- Control Objectives as Temporal Formulas
- Examples of Specifications
- The Synthesis Procedure Principles
- Remarks on the Results
Processes for Systems

\[ \mathcal{S} = \langle S, s^0, t, L \rangle \] on \( \Gamma \subseteq AP \)

where

- \( \Sigma = \{a, b, \ldots \} \) events
- \( AP = \{p, p', c, c', \ldots \} \) (atomic) propositions
- \( S, s^0 \in S \) set of states, initial state,
- \( t : S \times \Sigma \rightarrow S \) transition (partial) function,
- \( L : S \rightarrow 2^\Gamma \) labeling of states by propositions
A Example of Process

\[ S = \langle S, s^0, t, L \rangle \text{ on } \{p_0, p_1\} \]
Synchronous Product

- \( S_1 = \langle S_1, s_1^0, t_1, L_1 \rangle \) on \( \Gamma_1 \) and \( S_2 = \langle S_2, s_2^0, t_2, L_2 \rangle \) on \( \Gamma_2 \)
  (with \( \Gamma_1 \cap \Gamma_2 = \emptyset \))

\[ S_1 \times S_2 = \langle S_1 \times S_2, (s_1^0, s_2^0), t, L \rangle \]

\[ t((s_1, s_2), a) = (s_1', s_2') \text{ whenever} \]
\[ \left\{ \begin{array}{l}
      s_1' = t_1(s_1, a), \text{ and} \\
      s_2' = t_2(s_2, a)
   \end{array} \right. \]

\[ L(s_1, s_2) = L_1(s_1) \cup L_2(s_2) \]
Example of Synchronous Product

$S_1$

$S_1 \times S_2$

$S_2$
Control Problems

- $\Sigma = \Sigma_{uc} \uplus \Sigma_c$
- Controllers are processes on $\emptyset$
- Admissible Controllers
- $C$ is an admissible controller of $S$ for property $P$ whenever $C$ is admissible and $S \times C$ satisfies the property $P$
- Also $C_1$ and $C_2$...

\[ a \in \Sigma_{uc} \]

\[ \begin{array}{c}
  p_0 \\
  \rightarrow a \\
  b \\
  \rightarrow p_1
\end{array} \quad \begin{array}{c}
  \circ \\
  \rightarrow a \\
  \circ \\
  \rightarrow b
\end{array} \quad \begin{array}{c}
  \circ \\
  \rightarrow a \\
  \circ \\
  \rightarrow b
\end{array} \]

$S, C, S \times C$ satisfies $P$
Pruning

computation tree of $S$

computation tree of $S \times C$
c-labeling of $S$

computation tree of $S$

computation tree of $S \times E_c$
Labeling Processes are Controllers

- c-labeling process

\[ \mathcal{E} = \langle E, e^0, t', L' \rangle \] a process on \{c\}

which is complete, i.e. \( t'(e, a) \) is always defined.
Labeling Processes and Controllers

\[ S \]

\[ C \]

\[ S \times C \]

\[ E_C \]

\[ S \times E_C \]
c-Adjustment and c-Pruning

\[ S \times \mathcal{E} \text{ satisfies } P^c \text{ iff } S \times \mathcal{E}_{\rightarrow c} \text{ satisfies } P \]
The Property $P$

- Mu-calculus $L_\mu$
  
  [Kozen 1983], [Arnold & Niwinski 2001]

- Fix-point operators to build your own modalities

  $L_\mu$ subsumes CTL, LTL, CTL*, ...

  Safety, Liveness, Fairness, ...

- But you can use your favorite logic HML, CTL ....
The Mu-Calculus $L_\mu$

- **Syntax of $L_\mu$**
  \[ T \mid p \mid \neg \beta \mid \beta \lor \beta' \mid <a>\beta \mid X \mid \mu X.\beta(X) \]
  where $p \in AP$, $a \in \Sigma$, conditions on $\beta(X)$, ...

- **Semantics of $L_\mu$**
  $S, s \models \beta$ for “state $s$ of $S$ satisfies $\beta$”

- **Semantics**
  $S, s \models T$ always
  $S, s \models p$ iff $s$ has label $p$
  $S, s \models <a>\beta$ iff there exists $s' \in S$, $\begin{cases} t(s, a) = s' \\ S, s' \models \beta \end{cases}$
  $S, s \models \mu X.\beta(X)$ is technical ....

\[ [a]\alpha \overset{\text{def}}{=} \neg <a>\neg \alpha \] means “if any $a$-successeurs then it satisfies $\alpha$”

\[ \mu X. <a>X \lor m \] means “some $a^*$ trace reaches a marked state”

\[ AG(\alpha) \overset{\text{def}}{=} \neg \mu X. \neg ( [ ] \neg X \land \beta) \] means “property $\alpha$ is invariant”
The Quantified Mu-Calculus $QL_\mu$

- Syntaxe of $QL_\mu$: $\exists c. \alpha | \neg \alpha | \alpha \lor \alpha' | \beta$
  where $c \in AP$ and $\beta \in L_\mu$

- $S, s \models \exists c. \alpha$ iff there exists $E = \langle E, \varepsilon^0, t', L' \rangle$ on $\{c\}$ s.t.
  $S \times E, (s, \varepsilon^0) \models \alpha$
Remember

- $S \times \mathcal{E}$ is a $c$-labeling of $S$
  
  that is (an unfolding of) $S$ with the new label $c$ put somehow.

- This labeling denotes a control policy
Remember the Pruning and the Adjustment

\[ S \times (\mathcal{E}_c) \models \alpha \]

\[ S \times \mathcal{E}_c \models \alpha^c \]

\[ \text{c-pruning} \]

\[ \text{c-adjustment} \]
c-Adjustment of Mu-calculus formulas, but actually of $QL_\mu$-formulas

- $\mathcal{E}_{\rightarrow c}$ is $c$-pruning the of $\mathcal{E}$ keeps states “inside” $c$ and forget $c$
- the $c$-adjustment of $\alpha$ adjusts the formula to talk only about trajectories “inside” $c$

$\alpha * c$ essentially relies on $(<a>...)*c \overset{\text{def}}{=} <a>(c \land ...)$

**Theorem (c-Pruning and c-Adjustment)**

$$\mathcal{S} \times \mathcal{E}_{\rightarrow c} \models \alpha \text{ iff } \mathcal{S} \times \mathcal{E} \models \alpha * c$$

- **RMK: CTL logic**
  - $(EX...)*c \overset{\text{def}}{=} EX(c \land ...)$,
  - $(AX...)*c \overset{\text{def}}{=} AX(c \Rightarrow ...)$,
  - $(E...U...)*c \overset{\text{def}}{=} E(c \land ...Uc \land ...)$,
  - $(A...U...)*c \overset{\text{def}}{=} A(c \Rightarrow ...Uc \Rightarrow ...)$
Theorem for Basic Controllers Specification

Write $\text{Admissible}(c) \in L_\mu$ for $\text{AG} (\bigwedge_{u \in \Sigma_c} [u] c)$

For any $\alpha \in Q L_\mu$,

there exists an admissible controller $C$ of $S$ for $\alpha$

if and only if

$S \models \exists c. \text{Admissible}(c) \land \alpha^* c$
Proof sketch for $\iff$

There exists an admissible controller $C$ of $S$ for $\alpha$ if and only if

$$S \models \exists c. \text{Admissible}(c) \land \alpha \ast c$$

1. $S \times \mathcal{E} \models \text{Admissible}(c) \iff \mathcal{E}_c \rightarrow c$ is admissible.
2. $S \times \mathcal{E} \models \alpha \ast c \iff S \times \mathcal{E}_c \models \alpha$
Other Examples of Specifications

Use the pattern

$$\exists c. \cdots \ast c \land \cdots$$

- **Admissible** we’ve seen Admissible\(c\)
  
  $$\text{AG} (\bigwedge_{a \in \Sigma} \text{UnCont}(a) \Rightarrow [a]c)$$

- **Non-Blocking**
  
  $$\text{AG} (\text{EF}(m) \ast c)$$ where \text{EF} \(m\) means “Reachable\(m\)”,
  
  $$\text{AG} (\bigvee_{a \in \Sigma} <a> \top \ast c$$ for liveness

- **Open Systems** is Admissible\(c\)
  
  $$\cdots \text{ is } \forall e. \text{AG} (\bigwedge_{\bar{u} \in \Sigma} [\bar{u}]e) \Rightarrow (\alpha \ast e) \ast c$$
  
  where \(\forall e\) is \(\lnot \exists e \lnot\)
My Favorite: the Theorem for Maximally Permissive Controllers

For any $\alpha \in QL_\mu$, there exists a controller of $S$ for $\alpha$ which is maximally permissive

iff

$$S \models \exists c. \text{Admissible}(c) \land \alpha \land \forall c'. [c \sqsubseteq c' \Rightarrow \neg \text{Solution}(c', \alpha)]$$

Solution$(c, \alpha)$

- $c \sqsubseteq c'$ means “if a state is labeled by $c$, it is also labeled by $c'$”; it is $L_\mu$ definable
Proof sketch

- \( \mathcal{C} \) is more permissive than \( \mathcal{C'} \) whenever \( S \times \mathcal{C'} \preceq S \times \mathcal{C} \)
  
  (\( \preceq \) means there exists a simulation)

- \( c \sqsubseteq c' \overset{\text{def}}{=} ([\ ] \text{AG} (c')) \ast c \) and \( c \sqsubseteq c' \overset{\text{def}}{=} (c \sqsubseteq c') \land \neg(c' \sqsubseteq c) \)

- In fact,

\[
S \times \mathcal{E} \times \mathcal{E} \models c \sqsubseteq c' \iff S \times \mathcal{E} \rightarrow c \preceq S \times \mathcal{E} \rightarrow c'
\]
Model-Checking and Synthesis

We want to model-check $S \models \exists c. \beta$ for $\beta \in L_\mu$

- **Mu-calculus formulas are equivalent to (parity) tree automata**
  
  \[ [\text{Emerson & Jutla 1991}] \quad \beta \sim A_\beta \]
  
  $S \models \beta$ iff $S \in L(A_\beta)$
  
  iff $\exists$ a winning strategy in the two-players game $G(S, A_\beta)$

- **[RP03]** $QL_\mu$ formulae also have their automata semantics.
  
  $A_{\exists c. \alpha}$ is the $c$-projection of $A_\alpha$ [Rabin69]
Model-Checking and Synthesis for a formula $\exists c. \beta$

- 1. build the parity tree automaton $A_{\exists c. \beta}$;
- 2. compute the parity game $G(A, S)$;
- 3. find a winning memoryless strategy (if any) (determined by [EJ91])
- 4. it tells how to $c$-label the comp. tree of $S$ in a regular manner
- 5. take the finite state corresponding $\mathcal{E}$ to compute $\mathcal{C} = \mathcal{E}_{\rightarrow c}$

Complexity $O(|S|^{O(|\beta|)2^{O(|\beta|)}})$

Polynomial in the size of $S$
A few Remarks

- **Max Perm Controllers** difficult problem in the full temporal logic setting [Thistle94]
  
  \[ \exists c. \text{Solution}(c, \beta) \land \forall c'. [c \sqsupseteq c' \Rightarrow \neg \text{Solution}(c', \beta)] \]

  Independent of the plant

  Complexity of MPC \[ O(|S|^{2|\beta|}2^{2|\beta|}) \]

  Still Polynomial in the size of \( S \)

- **Decentralized Controllers**
  
  \[ \exists c_1, \exists c_2, \beta \ast (c_1 \land c_2) \land \beta'(c_1, c_2) \]

  Other Architecture? \( (c_1 \lor c_2), \text{Bool}(c_1, \ldots, c_n) \)

- We know how to deal with partial observation