

**Control Problems for DES
are
Model-Checking Problems**

Sophie Pinchinat

IRISA-INRIA

Campus de Beaulieu

F-35042, Rennes - France

Sophie.Pinchinat@irisa.fr

joint

work

with

Stéphane Riedweg

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

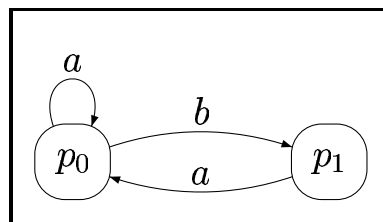
$$\mathcal{S} = \langle S, s^0, t, L \rangle \text{ on } \Gamma \subseteq AP$$

where

- $\Sigma = \{a, b, \dots\}$ events
- $AP = \{p, p', c, c', \dots\}$ (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

An Example of Process

$\mathcal{S} = \langle S, s^0, t, L \rangle$ on $\{p_0, p_1\}$



\mathcal{S}



Synchronous Product

- $\mathcal{S}_1 = \langle S_1, s_1^0, t_1, L_1 \rangle$ on Γ_1 and $\mathcal{S}_2 = \langle S_2, s_2^0, t_2, L_2 \rangle$ on Γ_2
(with $\Gamma_1 \cap \Gamma_2 = \emptyset$)

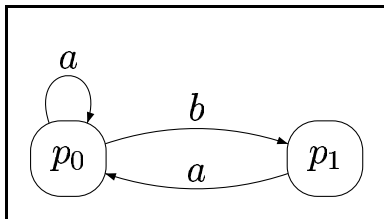
$$\mathcal{S}_1 \times \mathcal{S}_2 = \langle S_1 \times S_2, (s_1^0, s_2^0), t, L \rangle$$

$t((s_1, s_2), a) = (s'_1, s'_2)$ whenever

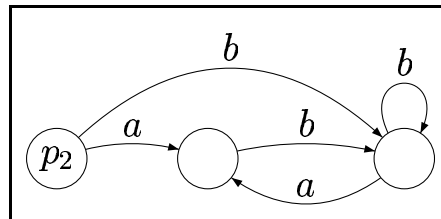
$$\begin{cases} s'_1 = t_1(s_1, a), \text{ and} \\ s'_2 = t_2(s_2, a) \end{cases}$$

$$L(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$$

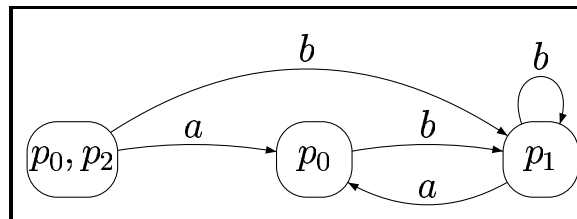
Example of Synchronous Product



S_1



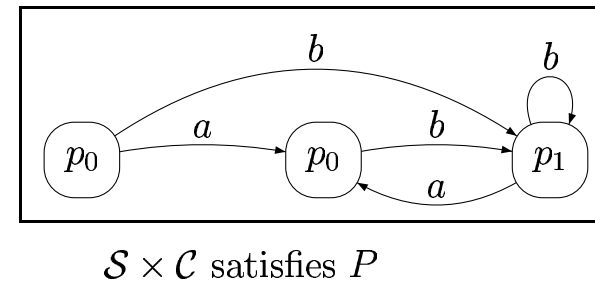
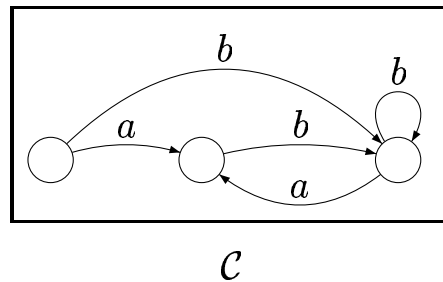
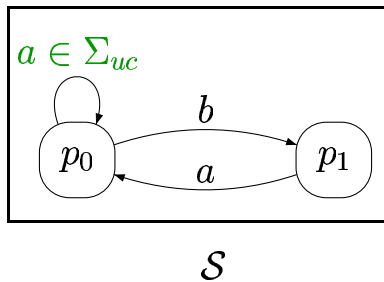
S_2



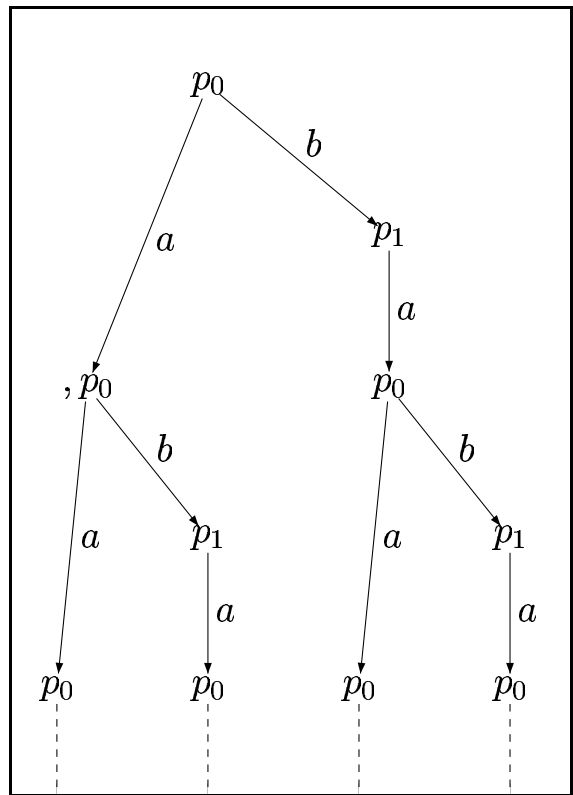
$S_1 \times S_2$

Control Problems

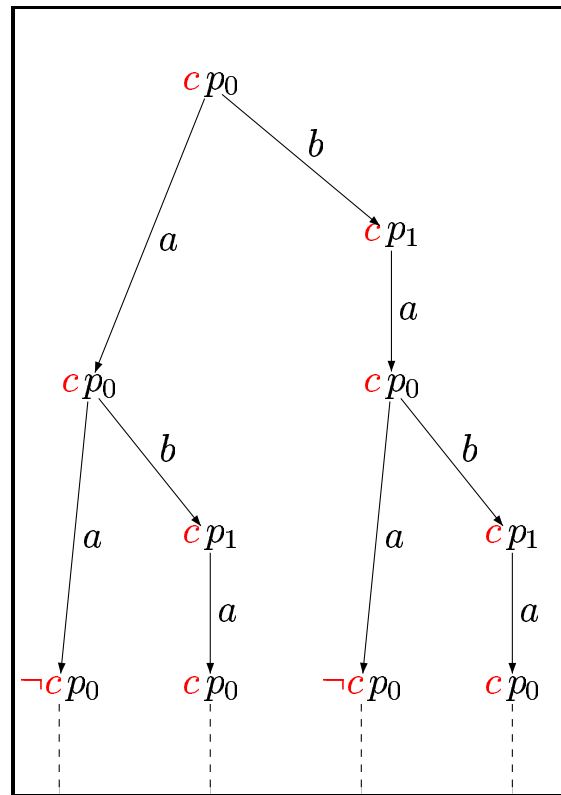
- $\Sigma = \Sigma_{uc} \uplus \Sigma_c$
- **Controllers** are processes on \emptyset
- **Admissible** Controllers
- \mathcal{C} is an admissible controller of \mathcal{S} for property P whenever
 - \mathcal{C} is admissible and $\mathcal{S} \times \mathcal{C}$ satisfies the property P
 - Also \mathcal{C}_1 and $\mathcal{C}_2 \dots$



c-labeling of \mathcal{S}



computation tree of \mathcal{S}



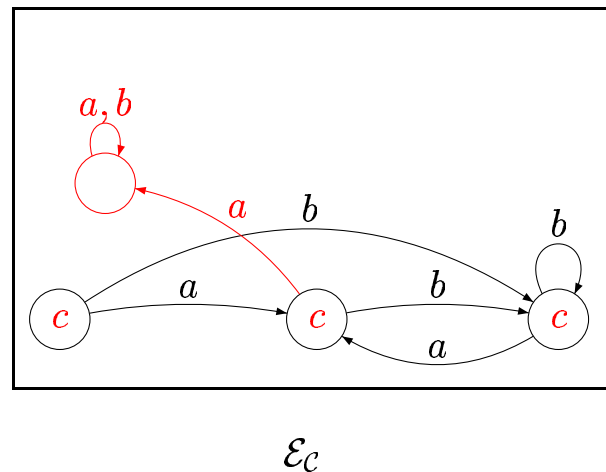
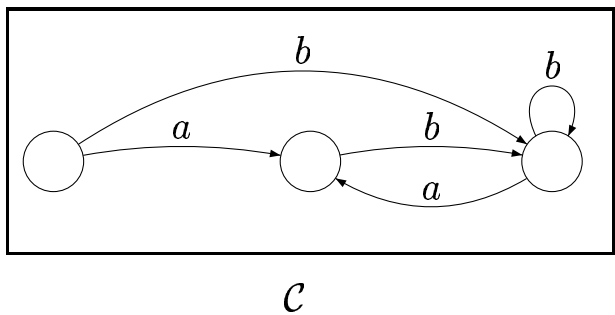
computation tree of $\mathcal{S} \times \mathcal{E}_c$

Labeling Processes are Controllers

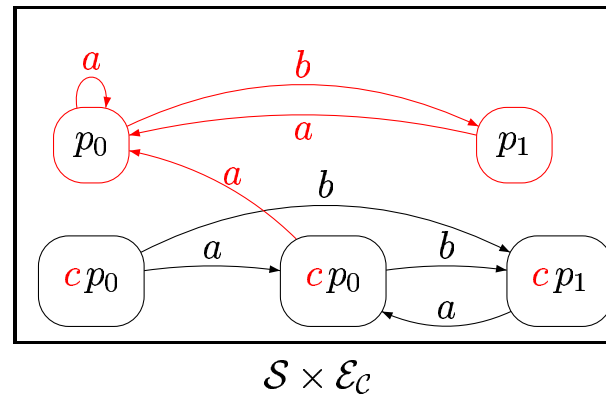
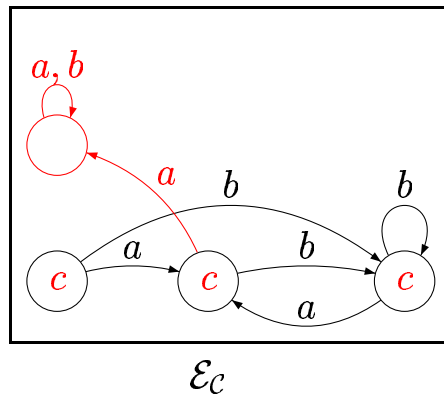
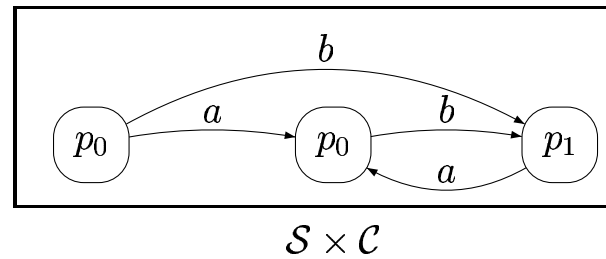
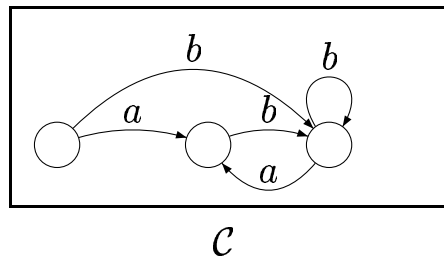
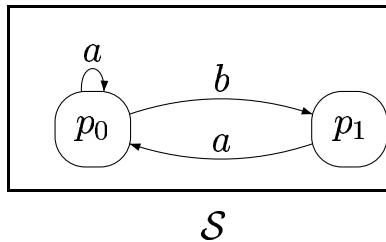
- c -labeling process

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

which is **complete**, i.e. $t'(e, a)$ is always defined.

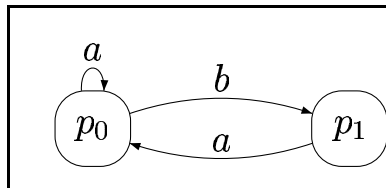


Labeling Processes and Controllers

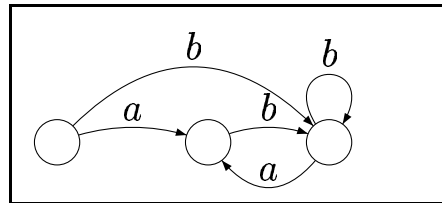


c-Adjustment and c-Pruning

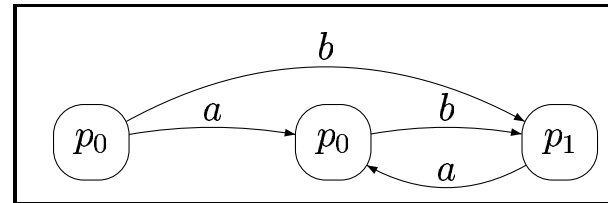
$S \times \mathcal{E}$ satisfies $P * c$ iff $S \times \mathcal{E}_{\rightarrow c}$ satisfies P



S



C



$S \times (\mathcal{E}_c)_{\rightarrow c}$

satisfies

P

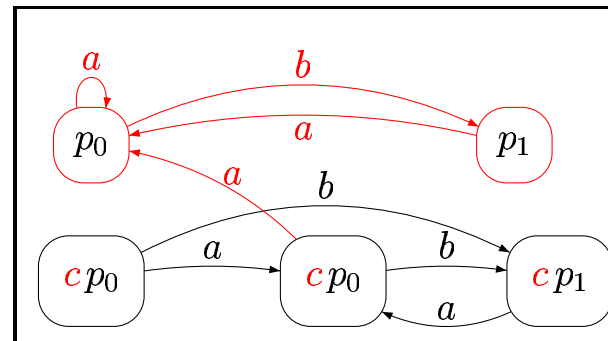
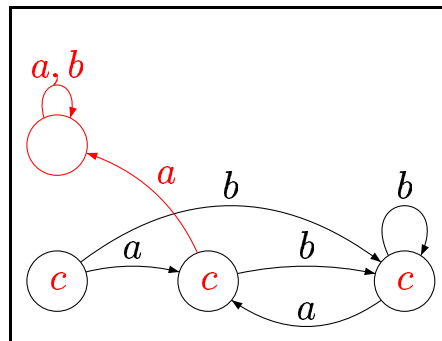
c-pruning

$S \times \mathcal{E}_c$

satisfies

$P * c$

c-adjustment



The Property P

- Mu-calculus L_μ
[Kozen 1983], [Arnold & Niwinski 2001]
- Fix-point operators to build your own modalities
 L_μ subsumes CTL, LTL, CTL*, ...
Safety, Liveness, Fairness, ...
- But you can use your favorite logic HML, CTL

The Mu-Calculus L_μ

- Syntax of L_μ $\top \mid p \mid \neg\beta \mid \beta \vee \beta' \mid \langle a \rangle\beta \mid X \mid \mu X.\beta(X)$

where $p \in AP$, $a \in \Sigma$, conditions on $\beta(X)$, ...

- Semantics of L_μ $\mathcal{S}, s \models \beta$ for “state s of \mathcal{S} satisfies β ”

- Semantics $\mathcal{S}, s \models \top$ always

$\mathcal{S}, s \models p$ iff s has label p

$\mathcal{S}, s \models \langle a \rangle\beta$ iff there exists $s' \in S$, $\begin{cases} t(s, a) = s' \\ \mathcal{S}, s' \models \beta \end{cases}$

$\mathcal{S}, s \models \mu X.\beta(X)$ is technical

$[a]\alpha \stackrel{\text{def}}{=} \neg \langle a \rangle \neg\alpha$ means “if any a -successors then it satisfies α ”

$\mu X. \langle a \rangle X \vee m$ means “some a^* trace reaches a marked state”

AG (α) $\stackrel{\text{def}}{=} \neg\mu X. \neg([\] \neg X \wedge \beta)$ means “property α is invariant”



The Quantified Mu-Calculus QL_μ

- Syntaxe of QL_μ $\exists c.\alpha \mid \neg\alpha \mid \alpha \vee \alpha' \mid \beta$

where $c \in AP$ and $\beta \in L_\mu$

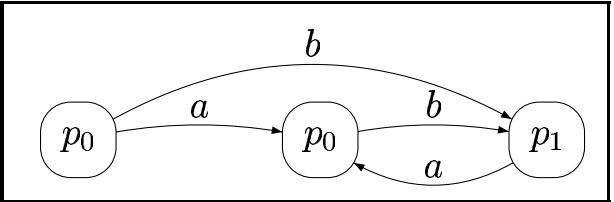
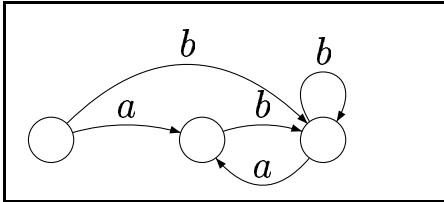
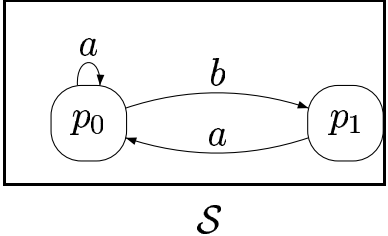
- $\mathcal{S}, s \models \exists c.\alpha$ iff there exists $\mathcal{E} = \langle E, \varepsilon^0, t', L' \rangle$ on $\{c\}$ s.t.

$$\mathcal{S} \times \mathcal{E}, (s, \varepsilon^0) \models \alpha$$

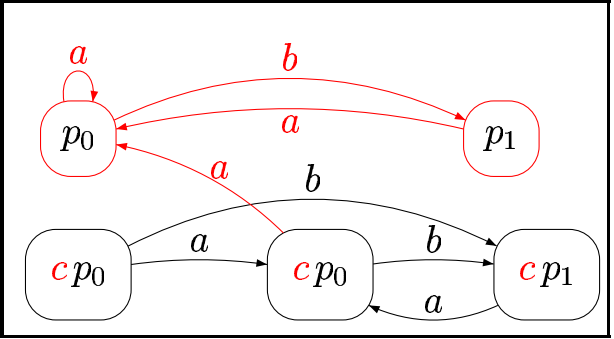
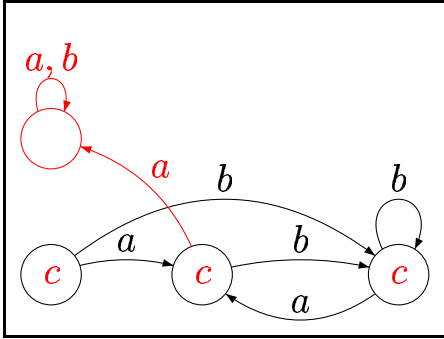
Remember

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

Remember the Pruning and the Adjustment



$S \times (\mathcal{E}c)_{\rightarrow c} \models \alpha$
 \uparrow c -pruning \downarrow c -adjustment
 $S \times \mathcal{E}c \models \alpha * c$



c-Adjustment of Mu-calculus formulas, but actually of QL_μ -formulas

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

$\alpha * c$ essentially relies on $(\langle a \rangle \dots) * c \stackrel{\text{def}}{=} \langle a \rangle (c \wedge \dots)$

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** $(\mathbf{EX} \dots) * c \stackrel{\text{def}}{=} \mathbf{EX}(c \wedge \dots)$, $(\mathbf{AX} \dots) * c \stackrel{\text{def}}{=} \mathbf{AX}(c \Rightarrow \dots)$, $(\mathbf{E} \dots \mathbf{U} \dots) * c \stackrel{\text{def}}{=} \mathbf{E}(c \wedge \dots \mathbf{U} c \wedge \dots)$, $(\mathbf{A} \dots \mathbf{U} \dots) * c \stackrel{\text{def}}{=} \mathbf{A}(c \Rightarrow \dots \mathbf{U} c \Rightarrow \dots)$

The Theorem for Basic Controllers Specification

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

For any $\alpha \in QL_\mu$,

there exists an admissible controller \mathcal{C} of \mathcal{S} for α

if and only if

$$\mathcal{S} \models \exists c. \text{Admissible}(c) \wedge \alpha * c$$

Proof sketch for \Leftarrow)

There exists an **admissible controller** \mathcal{C} of \mathcal{S} for α

if and only if

$$\mathcal{S} \models \exists c. \text{Admissible}(c) \wedge \alpha * c$$

1. $\mathcal{S} \times \mathcal{E} \models \text{Admissible}(c) \Leftrightarrow \mathcal{E}_{\rightarrow c}$ is admissible.

2. $\mathcal{S} \times \mathcal{E} \models \alpha * c \Leftrightarrow \mathcal{S} \times \mathcal{E}_{\rightarrow c} \models \alpha$

Other Examples of Specifications

Use the pattern

$$\exists c. \boxed{\dots} * c \wedge \boxed{\dots}$$

- **Admissible** we've seen $\text{Admissible}(c)$

$$\mathbf{AG} (\bigwedge_{a \in \Sigma} \text{UnCont}(a) \Rightarrow [a]c)$$

- **Non-Blocking**

$$\mathbf{AG} (\mathbf{EF} (m) * c) \quad \text{where } \mathbf{EF} m \text{ means "Reachable}(m)",$$

$$\mathbf{AG} (\bigvee_{a \in \Sigma} \langle a \rangle \top) * c \text{ for liveness}$$

- **Open Systems** $\boxed{\dots}$ is $\text{Admissible}(c)$

$$\boxed{\dots} \text{ is } \forall e. \mathbf{AG} (\bigwedge_{\bar{u} \in \Sigma_c} [\bar{u}]e) \Rightarrow (\alpha * e) * c$$

where $\forall e$ is $\neg \exists e \neg$

My Favorite : the Theorem for Maximally Permissive Controllers

For any $\alpha \in QL_\mu$,

there exists a controller of \mathcal{S} for α which is **maximally permissive**

iff

$$\mathcal{S} \models \exists c. \underbrace{\text{Admissible}(c) \wedge \alpha * c}_{\text{Solution}(c, \alpha)} \wedge \forall c'. [c \sqsubseteq c' \Rightarrow \neg \text{Solution}(c', \alpha)]$$

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

Proof sketch

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

- $c \sqsubseteq c' \stackrel{\text{def}}{=} ([] \mathbf{AG} (c')) * c$ and $c \sqsubset c' \stackrel{\text{def}}{=} (c \sqsubseteq c') \wedge \neg(c' \sqsubseteq c)$

- In fact,

$$\mathcal{S} \times \mathcal{E} \times \mathcal{E}' \models c \sqsubseteq c' \Leftrightarrow \mathcal{S} \times \mathcal{E}_{\rightarrow c} \preceq \mathcal{S} \times \mathcal{E}'_{\rightarrow c'}$$

Model-Checking and Synthesis

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

- Mu-calculus formulas are equivalent to (parity) tree automata

[Emerson & Jutla 1991] $\beta \rightsquigarrow \mathcal{A}_\beta$

$\mathcal{S} \models \beta$ iff $\mathcal{S} \in \mathcal{L}(\mathcal{A}_\beta)$

iff \exists a winning strategy in the two-players game $\mathcal{G}(\mathcal{S}, \mathcal{A}_\beta)$

- [RP03] QL_μ formulae also have their automata semantics.

$\mathcal{A}_{\exists c.\alpha}$ is the c -projection of \mathcal{A}_α [Rabin69]

Model-Checking and Synthesis for a formula $\exists c.\beta$

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

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

Polynomial in the size of \mathcal{S}

A few Remarks

- **Max Perm Controllers** difficult problem in the full temporal logic setting [Thistle94]

Remember $\exists c. \text{Solution}(c, \beta) \wedge \forall c'. [c \sqsubset c' \Rightarrow \neg \text{Solution}(c', \beta)]$

Independent of the plant

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

Still Polynomial in the size of \mathcal{S}

- **Decentralized Controllers**

$\exists c_1. \exists c_2. \beta^*(c_1 \wedge c_2) \wedge \beta'(c_1, c_2)$

Other Architecture ? $(c_1 \vee c_2), \text{Bool}(c_1, \dots, c_n)$

- We know how to deal with partial observation