Controller Synthesis for stochastic systems

BHARATH SIVA KUMAR TATI

MARKUS SIEGLE
Outline

- Introduction
- ASMC and asCSL
- Example for Parameter Synthesis (special case of controller synthesis)
- Example for Controller Synthesis
- Conclusion
- Future work
Introduction

Plant P
Introduction

Plant P

Φ
Introduction
Introduction

Plant $P$ → Model checking

$\Phi$ → Satisfied

$\Phi$ → Violated
Introduction

Plant P

Model checking

Satisfied

Violated
Introduction

- Plant P
- Model checking

- Satisfied
- Violated

Modify parameters
- Using reduction factor k
- Reducing rates,
- not increasing rates
Introduction

Plant P

Model checking

Model checking

Modify parameters
- Using reduction factor k
- Reducing rates,
- not increasing rates

Synthesize controller

Satisfied

Violated
Introduction

ASMC’s

Plant P

Model checking

Satisfied

Violated

Modify parameters
- Using reduction factor k
- Reducing rates,
- not increasing rates

Synthesize controller
Introduction

**ASMC's**

Plant $P$

**Untimed $\text{asCSL}$**

- **Model checking**
  - Satisfied
  - Violated
  - Modify parameters
    - Using reduction factor $k$
    - Reducing rates,
    - not increasing rates
  - Synthesize controller
Introduction

**ASMC's**

Plant P

**Untimed asCSL**

- Using reduction factor k
- Reducing rates,
- not increasing rates

Modify parameters

Synthesize controller
ASMC’s

Markov Chain with actions and state labels is called as ASMC

\[ \mathcal{M} = (S, \Sigma, R, L) \]
ASMC’s

Markov Chain with actions and state labels is called as ASMC

\[ \mathcal{M} = (S, \Sigma, R, L) \]

\( S = \{ P_1, P_2, P_3, P_4, P_5 \} \)

\( \Sigma = \{ \text{on, off, full, down, up} \} \)

\( L = \{ \text{Off, Full, Empty, Green, Yellow, Red} \} \)

Plant \( \mathcal{P} \)
ASMC’s

Markov Chain with actions and state labels is called as ASMC

\[ \mathcal{M} = (S, \Sigma, R, L) \]

\[ S = \{ P_1, P_2, P_3, P_4, P_5 \} \]

\[ \Sigma = \{ \text{on}, \text{off}, \text{full}, \text{down}, \text{up} \} \]

\[ L = \{ \text{Off}, \text{Full}, \text{Empty}, \text{Green}, \text{Yellow}, \text{Red} \} \]

\[ \text{ASMC} = \text{CTMC} + \text{action labels} + \text{state labels} \]
asCSL

asCSL is a temporal logic used to specify complex path-based behavior with the help of regular expressions over state properties and action labels.
asCSL

asCSL is a temporal logic used to specify complex path-based behavior with the help of regular expressions over state properties and action labels.

State formulas of asCSL:

\[ \Phi ::= q \mid \neg \Phi \mid \Phi \land \Phi \mid P_{\sim b}(\alpha) \]
asCSL

asCSL is a temporal logic used to specify complex path-based behavior with the help of regular expressions over state properties and action labels.

State formulas of asCSL:

\[ \Phi ::= q \mid \neg \Phi \mid \Phi \land \Phi \mid P_{\sim b}(\alpha) \]

Grammar of asCSL programs is given by:

\[ \alpha ::= \epsilon \mid (\phi, b) \mid \alpha; \alpha \mid \alpha \cup \alpha \mid \alpha^* \]
asCSL

asCSL is a temporal logic used to specify complex path-based behavior with the help of regular expressions over state properties and action labels.

State formulas of asCSL:

\[
\Phi ::= q \mid \neg \Phi \mid \Phi \land \Phi \mid P_{=b}(\alpha)
\]

Grammar of asCSL programs is given by:

\[
\alpha = \epsilon \mid (\phi, b) \mid \alpha; \alpha \mid \alpha \cup \alpha \mid \alpha^*
\]
asCSL

asCSL is a temporal logic used to specify complex path-based behavior with the help of regular expressions over state properties and action labels.

State formulas of asCSL:

\[ \Phi ::= q \mid \neg \Phi \mid \Phi \land \Phi \mid P_{\sim b}(\alpha) \]

Grammar of asCSL programs is given by:

\[ \alpha ::= \epsilon \mid (\phi, b) \mid \alpha ; \alpha \mid \alpha \cup \alpha \mid \alpha^* \]

asCSL program

(State formula, Action)
asCSL

asCSL is a temporal logic used to specify complex path-based behavior with the help of regular expressions over state properties and action labels.

State formulas of asCSL:

\[ \Phi ::= q \mid \neg \Phi \mid \Phi \land \Phi \mid P_{\sim b}(\alpha) \]

Grammar of asCSL programs is given by:

\[ \alpha = \epsilon \mid (\phi, b) \mid \alpha; \alpha \mid \alpha \cup \alpha \mid \alpha^* \]

asCSL program \rightarrow (State formula, Action) \rightarrow Sequential composition
asCSL

asCSL is a temporal logic used to specify complex path-based behavior with the help of regular expressions over state properties and action labels.

State formulas of asCSL:

\[ \Phi ::= q \mid \neg \Phi \mid \Phi \land \Phi \mid P_{\sim b}(\alpha) \]

Grammar of asCSL programs is given by:

\[ \alpha = \epsilon \mid (\phi, b) \mid \alpha; \alpha \mid \alpha \cup \alpha \mid \alpha^* \]

asCSL program

(State formula, Action)

choice

Sequential composition
asCSL

asCSL is a temporal logic used to specify complex path-based behavior with the help of regular expressions over state properties and action labels.

State formulas of asCSL:

\[ \Phi ::= q \mid \neg \Phi \mid \Phi \land \Phi \mid P_{\sim b}(\alpha) \]

Grammar of asCSL programs is given by:

- \( \alpha = \epsilon \mid (\phi, b) \mid \alpha; \alpha \mid \alpha \cup \alpha \mid \alpha^* \)
- asCSL program
- (State formula, Action)
  - choice
- Sequential composition
  - Kleene star
Untimed Until

- Untimed Until formulas are also used to specify path behaviors

$$\sigma \models \Phi_1 \mathcal{U} \Phi_2$$
Untimed Until

- Untimed Until formulas are also used to specify path behaviors

\[ \sigma \models \Phi_1 \mathcal{U} \Phi_2 \]

- Any untimed CSL *Until* property can be expressed in asCSL

\[ \Phi_1 \mathcal{U} \Phi_2 = (\Phi_1, \Sigma)^* : (\Phi_2, \sqrt{\cdot}) \]
Example for Par. Synthesis

\[ \Phi = P_{\leq 0.7}(\varphi), \text{ where } \varphi = \text{Empty } \cup \text{ Full} \]

\[
\begin{align*}
Pr(P_1, \varphi) &= p_{15} = 0 < 0.7 \\
Pr(P_2, \varphi) &= p_{25} = 0.69473 < 0.7 \\
Pr(P_3, \varphi) &= p_{35} = 0.80589 > 0.7 \\
Pr(P_4, \varphi) &= p_{45} = 0.90294 > 0.7 \\
Pr(P_5, \varphi) &= p_{55} = 1
\end{align*}
\]
Example for Par. Synthesis

\[ \Phi = P_{\leq 0.7}(\varphi), \text{ where } \varphi = \text{Empty} \cup \text{Full} \]

\[ Pr(P_1, \varphi) = p_{15} = 0 < 0.7 \]
\[ Pr(P_2, \varphi) = p_{25} = 0.69473 < 0.7 \]
\[ Pr(P_3, \varphi) = p_{35} = 0.80589 > 0.7 \]
\[ Pr(P_4, \varphi) = p_{45} = 0.90294 > 0.7 \]
\[ Pr(P_5, \varphi) = p_{55} = 1 \]
Example for Par. Synthesis

\[ \Phi = P_{\leq 0.7}(\varphi), \text{ where } \varphi = Empty \cup Full \]

\[
\begin{align*}
Pr(P_1, \varphi) &= p_{15} = 0 < 0.7 \\
Pr(P_2, \varphi) &= p_{25} = 0.69473 < 0.7 \\
Pr(P_3, \varphi) &= p_{35} = 0.80589 > 0.7 \\
Pr(P_4, \varphi) &= p_{45} = 0.90294 > 0.7 \\
Pr(P_5, \varphi) &= p_{55} = 1
\end{align*}
\]
Example for Par. Synthesis

\( \Phi = P_{\leq 0.7}(\varphi) \), where \( \varphi = \text{Empty} \cup \text{Full} \)

\[
\begin{align*}
Pr(P_1, \varphi) &= p_{15} = 0 < 0.7 \\
Pr(P_2, \varphi) &= p_{25} = 0.69473 < 0.7 \\
Pr(P_3, \varphi) &= p_{35} = 0.80589 > 0.7 \\
Pr(P_4, \varphi) &= p_{45} = 0.90294 > 0.7 \\
Pr(P_5, \varphi) &= p_{55} = 1
\end{align*}
\]
\[ \Phi = P_{\leq 0.7}(\varphi), \text{ where } \varphi = \text{Empty} \cup \text{Full} \]
Contd...

\[ \Phi = P_{\leq 0.7}(\varphi), \text{ where } \varphi = \text{Empty} \cup \text{Full} \]
\[ \Phi = P_{\leq 0.7}(\varphi), \text{ where } \varphi = \text{Empty} \cup \text{Full} \]
Contd...

\[ \Phi = P_{\leq 0.7}(\varphi), \text{ where } \varphi = \text{Empty} \cup \text{Full} \]

Plant \( \mathcal{G} \)

\[ 0 < k \leq 1 \]
Contd...

\[ \Phi = P_{\leq 0.7}(\varphi), \text{ where } \varphi = \text{Empty } \cup \text{ Full} \]

Plant \( \mathcal{G} \)

\[ 0 < k \leq 1 \]
\[
\Phi = P_{\leq 0.7}(\varphi), \text{ where } \varphi = \text{Empty U Full}
\]

\[
0 < k \leq 1
\]
Contd...

\[ \Phi = P_{\leq 0.7}(\varphi), \text{ where } \varphi = \text{Empty} \cup \text{Full} \]

Plant \( \mathcal{G} \)

\[ 0 < k \leq 1 \]

\[ \mathcal{G} \models_v \Phi \implies \mathcal{P}_{\text{mod}} \models_v \Phi \]
CONTROLLER SYNTHESIS FOR STOCHASTIC SYSTEMS

Plant P

Untimed asCSL

Model checking

Satisfied

Violated

Modify parameters
- Using reduction factor k
- Reducing rates,
- not increasing rates

Synthesize controller
ASMC's

Plant P

Untimed asCSL

Model checking

Satisfied

Violated

Modify parameters
- Using reduction factor k
- Reducing rates,
- not increasing rates

Synthesize controller
Example for Cntrl. Synthesis

Plant $\mathcal{P}$

State transitions:
- $S_1$ to $S_2$: $a, \alpha$
- $S_1$ to $S_3$: $d, \delta$
- $S_2$ to $S_3$: $b_1, \beta_1$
- $S_2$ to $S_4$: $b_2, \beta_2$
- $S_3$ to $S_4$: $c_1, \gamma_1$
- $S_3$ to $S_1$: $c_2, \gamma_2$
Example for Cntrl. Synthesis

\[ \Phi = P_{>0.5}(\alpha) \]

Plant \( \mathcal{P} \)
Example for Cntrl. Synthesis

\[ \Phi = P_{>0.5}(\alpha) \]

\[ \alpha' = ((tt, b_1); (tt, c_1)) \cup ((tt, b_2); (tt, c_2)) \]
Example for Cntrl. Synthesis

\[ \Phi = P_{>0.5}(\alpha) \]

\[ \alpha = ((tt, a) \cup (tt, c_1) \cup (tt, c_2) \cup (tt, d))^*; \left( ((tt, b_1); (tt, c_1)) \cup ((tt, b_2); (tt, c_2)) \right) \]
Example for Cntrl. Synthesis

\[ \Phi = P_{>0.5}(\alpha) \]
Let \( \beta_1 = 4, \beta_2 = 6, \gamma_1 = 6, \gamma_2 = 4, \delta = 3, \alpha = 2 \)

\[ \alpha = ((tt, a) \cup (tt, c_1) \cup (tt, c_2) \cup (tt, d))^*; \left( ((tt, b_1); (tt, c_1)) \cup ((tt, b_2); (tt, c_2)) \right) \]
Example for Cntrl. Synthesis

$$\Phi = P_{>0.5}(\alpha)$$

Let $\beta_1 = 4, \beta_2 = 6, \gamma_1 = 6, \gamma_2 = 4, \delta = 3, \alpha = 2$

$$\alpha = ((tt, a) \cup (tt, c_1) \cup (tt, c_2) \cup (tt, d))^*; \left( ((tt, b_1); (tt, c_1)) \cup ((tt, b_2); (tt, c_2)) \right)$$

Plant $\mathcal{P}$

NPA $\mathcal{Z}_\alpha$ for given asCSL program $\alpha$
Cntrl. Synthesis contd...

\[ \Phi = P_{>0.5}(\alpha) \]
\[ \alpha = tt \cup target \]

Product automaton of \( P \times Z_\alpha \)
Cntrl. Synthesis contd...

\[ \Phi = P_{>0.5}(\alpha) \]
\[ \alpha = \text{tt } U \text{ target} \]

Product automaton of \( \mathcal{P} \times \mathbb{Z}_\alpha \)
Cntrl. Synthesis contd...

Product automaton of $\mathcal{P} \times \mathcal{Z}_\alpha$
CNTRL. SYNTHESIS CONTD...

Product automaton of $\mathcal{P} \times \mathcal{Z}_\alpha$

$\Phi = P_{>0.5}(\alpha)$
$\alpha = tt \cup \text{target}$

$Pr(Q_4, \varphi) = p_{48} = 0.48 < 0.5$
Cntrl. Synthesis contd...

\[ \Phi = P_{>0.5}(\alpha) \]
\[ \alpha = tt \cup \text{target} \]

Product automaton of \( P \times \mathcal{Z}_\alpha \)
Cntrl. Synthesis contd...

Product automaton of $\mathcal{P} \times \mathcal{Z}_\alpha$

$$\Phi = P_{>0.5}(\alpha)$$
$$\alpha = tt \cup target$$

$$Pr(Q_4, \varphi) = p_{48} = 0.48 < 0.5$$
Cntrl. Synthesis contd...

Product automaton of $\mathcal{P} \times \mathcal{Z}_\alpha$

$\Phi = P_{> 0.5}(\alpha)$
$\alpha = \text{tt } \cup \text{ target}$

$\text{Pr}(Q_4, \varphi) = p_{48} = 0.48 < 0.5$
Cntrl. Synthesis contd...

\[ \Phi = P_{>0.5}(\alpha) \]
\[ \alpha = tt U \text{ target} \]

Blueprint automaton \( \mathcal{B} \)
Cntrl. Synthesis contd...

\[ \Phi = P_{>0.5}(\alpha) \]
\[ \alpha = \text{tt } \mathcal{U} \text{ target} \]

Blueprint automaton $\mathcal{B}$
Cntrl. Synthesis contd...

$$\Phi = P_{>0.5}(\alpha)$$
Cntrl. Synthesis contd...

\[ \Phi = P_{>0.5}(\alpha) \]
Cntrl. Synthesis contd...

\[ \Phi = P_{>0.5}(\alpha) \]
Cntrl. Synthesis contd...

\[ \Phi = P_{>0.5}(\alpha) \]
Cntrl. Synthesis contd...

\[ \Phi = P_{>0.5}(\alpha) \]
Cntrl. Synthesis contd...

\[ \Phi = P_{>0.5}(\alpha) \]

Controller \( \mathcal{C} \)
Cntrl. Synthesis contd...

\[ \Phi = P_{>0.5}(\alpha) \]

\[ \mathcal{P} \models \Sigma_C \ C \models_{\nu} \Phi \]
Conclusion

- Parameter synthesis deals with finding the satisfactory rates as per the given user requirement.
  \[ \mathcal{M} \not \vDash \Phi \xrightarrow{k} \mathcal{M}' \models \Phi \]

- Controller synthesis changes the behavior of the plant using Par. synthesis.
  \[ \mathcal{M} \not \vDash \Phi \xrightarrow{k} \mathcal{M} \parallel_{\Sigma_C} \mathcal{C} \models \Phi \]

- More unknowns will make the computations harder, hence the common reduction factor k.
Future work

- Nesting of probabilistic properties is not handled by the algorithm.
- Time bounded asCSL properties
- Complexity and optimality
Thank you