Documentation SySCoRe

Birgit van Huijgevoort

Code is tested for Matlab R2018b. This code requires YALMIP (https://yalmip.github.io) and solver Sedumi (should be installed when YALMIP is installed) or Mosek (https://www.mosek.com). The solver is set in Similarity/ComputeDelta.m line 66: `options = sdpsettings(...)`.

Preliminaries

LTI system M

$$M : \begin{cases} x(k+1) & = Ax(k) + Bu(k) + B_w w(k) \\ y(k) & = Cx(k), \end{cases} \tag{1}$$

and abstract system $\hat{M}$

$$\hat{M} : \begin{cases} \hat{x}(k+1) = A\hat{x}(k) + B\hat{u}(k) + B_w\hat{w}(k) + \beta(k) \\ \hat{y}(k) \quad = C\hat{x}(k). \end{cases} \tag{2}$$

We use a maximal coupling $\mathscr{W}$ that maximizes the probability of event $w - \hat{w}_\gamma = 0$, since conditioned on this event the error dynamics reduce to

$$x_\Delta^+ = Ax_\Delta + B_w\gamma_k - \beta, \tag{3}$$

with $x_\Delta^+ = x(k+1) - \hat{x}(k+1)$ and $x_\Delta = x(k) - \hat{x}(k)$.

Function `TranslateSpec.m`

Translate the (sc)LTL formula to a DFA and rewrite to other format.

We use the toolbox LTL2BA (function `spec2buchi`) to translate the LTL formula to a Buchi automaton. Next, we change the format of the result. First, we redefine the actions of the DFA based on the alphabet as $[AP(i), !AP(i)]$, with $AP(i)$ the i-th atomic proposition. Next, we construct the matrix DFA.trans describing the transitions of the DFA. Here, $DFA.trans(i, j)$ defines the state to which you go to by starting at state i using the action as in $DFA.act\{j\}$.

Finally, we check whether the Buchi automaton we obtained is actually a DFA or not by checking 1. determinism and 2. a self-loop at the accepting state(s).

Function `GridSpace.m`

Make an abstraction of the original model by using a uniform grid.

Function `ComputeDelta.m`

Compute δ for given ε using an optimization problem with parameterized LMIs constraints.

Consider interface function $u(k) = \hat{u}(k)$, relation

$$\mathcal{R} := \left\{ (\hat{x}, x) \in \hat{\mathbb{X}} \times \mathbb{X} \,|\, ||x - \hat{x}||_D \leq \varepsilon \right\}, \tag{4}$$

where $||x||_D$ denotes the weighted 2-norm, that is, $||x||_D = \sqrt{x^T D x}$ with D a symmetric positive-definite matrix $D = D^T \succ 0$. Furthermore, consider an input gain for the error dynamics (3) denoted as $\gamma_k = Fx_\Delta$.

As described in [1], we can compute δ for a given ε by solving the following optimization problem.

$$\min_{D_{inv},L,r} r^2 \tag{5a}$$

$$\text{s.t. } D_{inv} \succ 0,$$

$$\begin{bmatrix} D_{inv} & D_{inv}C^T \\ CD_{inv} & I \end{bmatrix} \succeq 0, \qquad \textit{(ε-deviation)} \tag{5b}$$

$$\begin{bmatrix} \frac{1}{\varepsilon^2}D_{inv} & L^T \\ L & r^2 I \end{bmatrix} \succeq 0, \qquad \textit{(input bound)} \tag{5c}$$

$$\begin{bmatrix} \lambda D_{inv} & * & * \\ 0 & (1-\lambda)\varepsilon^2 & * \\ AD_{inv}+B_wL & \beta_l & D_{inv} \end{bmatrix} \succeq 0 \textit{ (invariance)} \tag{5d}$$

where $D_{inv} = D^{-1}$, $L = FD_{inv}$, $\beta_l \in vert(\mathcal{B})$ and $l \in \{0,1,\ldots,q\}$. Finally, we can compute δ as

$$\delta = |1 - 2\text{cdf}(-\frac{r}{2}, \mu, \sigma)|, \tag{6}$$

with $\text{cdf}(-\frac{r}{2}, \mu, \sigma)$ the cumulative distribution function of a Gaussian distribution with mean μ and variance σ until the line $w_1 = -\frac{r}{2}$.

Function `SynthesizeController.m`

Compute the robust satisfaction probability and robust control policy. This function is not correctly working! Currently we only use δ and not ε.

Value iteration

Value function $V(i,j)$ describes the probability of reaching the final DFA state `DFA.F` from DFA state `DFA.S(i)` and abstract state `sysAbs.states.XhatSpace(:,j)` within N time steps (or less).

We perform the value iteration N times or until it has converged. First, we set the row of $V(k+1)$ that corresponds to DFA state `DFA.F` equal to 1. Next, we pick the correct elements out of $V(k)$ by identifying the allowed transitions in the product automaton. We do this by comparing the labels of the abstract states to the actions of the DFA. The allowed transitions are stored in `V_sort`.

For each abstract input `uhat` we compute the value function as follows $V(k+1) = V(k) \cdot P$, where P is the matrix containing the probabilities of the transitions in the abstract system. Finally, we subtract δ and optimize over `uhat`.

References

[1] B.C. van Huijgevoort, and S. Haesaert. "Similarity quantification for linear stochastic systems as a set-theoretic control problem." *arXiv preprint*, 2020.