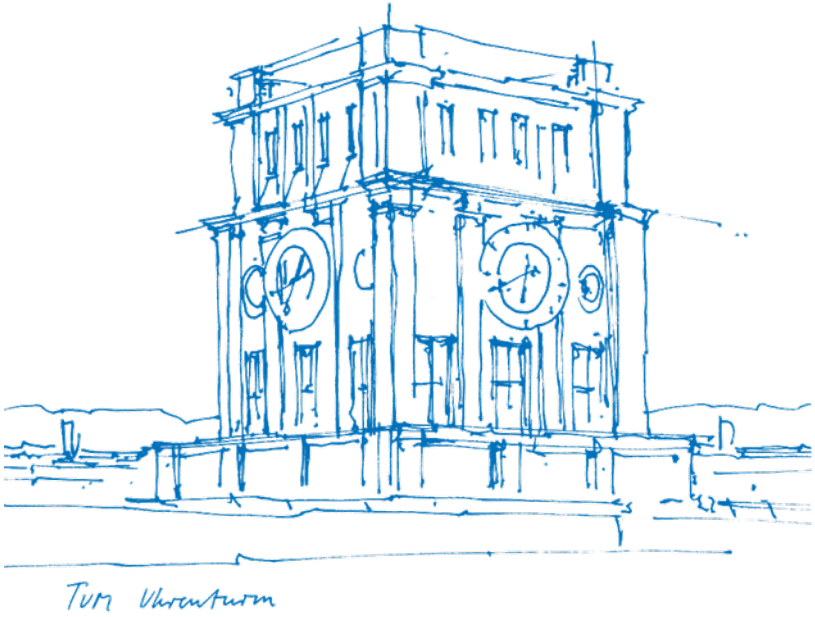
SENSE: Abstraction-Based Synthesis of Networked Control Systems

Mahmoud Khaled, Matthias Rungger, and Majid Zamani
Hybrid Control Systems Group
Electrical and Computer Engineering
Technical University of Munich

MeTRiD 2018
Thessaloniki, Greece
April, 15 2018



NCS: Motivation

Networked control systems (NCS):

- Remote access/control.
- Flexibility in deployment.
- Easy maintenance/upgrading.
- Reduced cost and complexity.

Source: Wikipedia, Spaceflight

Wide range of applications:

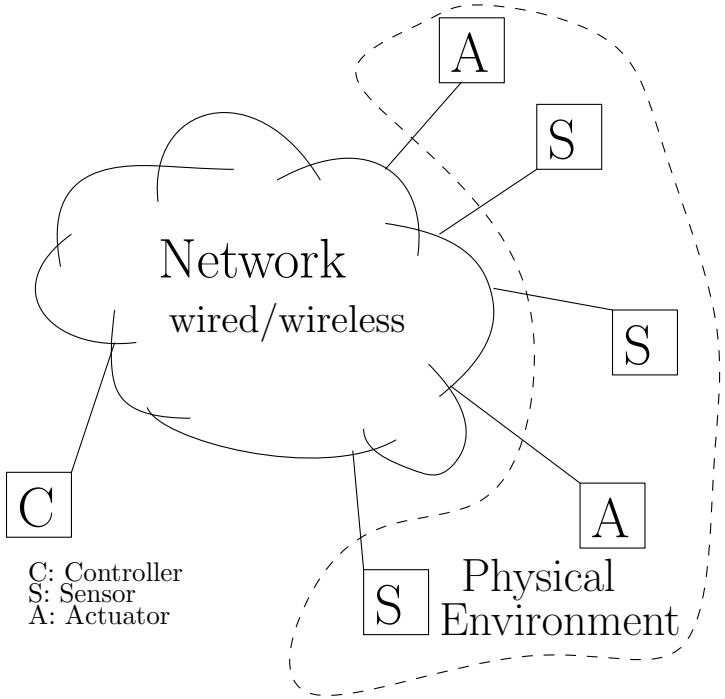
- Factory automation.
- Space exploration.
- Remote diagnosis and troubleshooting.
- Aircrafts and automobiles.

Source: WifiBot, www.net-rob.com

NCS: a Complex Cyber-Physical System

NCS combine:

- Physical environment: a plant, sensors and actuators.
- Computing devices: a remote digital controller.
- Communication networks: transfer state information and control actions.



Imperfections of communication networks:

- Time-varying sampling/transmission intervals.
- Time-varying communication delays
- Limited bandwidth.
- Quantization errors.
- Packet dropouts.

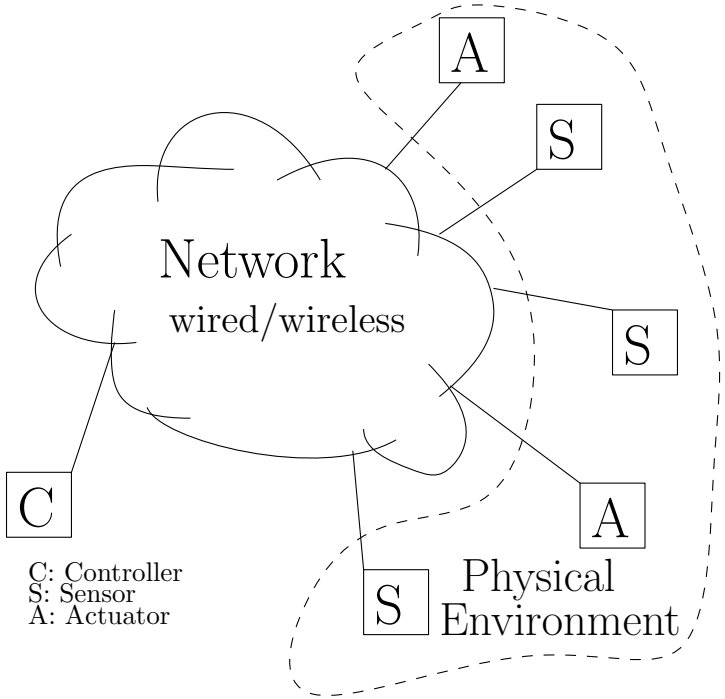
Control of NCS:

- Traditionally: classical control, existing methods are limited to stabilizability problem.
- Recently: formal methods, automated synthesis, correct-by-construction, developing research area.

NCS: a Complex Cyber-Physical System

NCS combine:

- Physical environment: a plant, sensors and actuators.
- Computing devices: a remote digital controller.
- Communication networks: transfer state information and control actions.



Imperfections of communication networks:

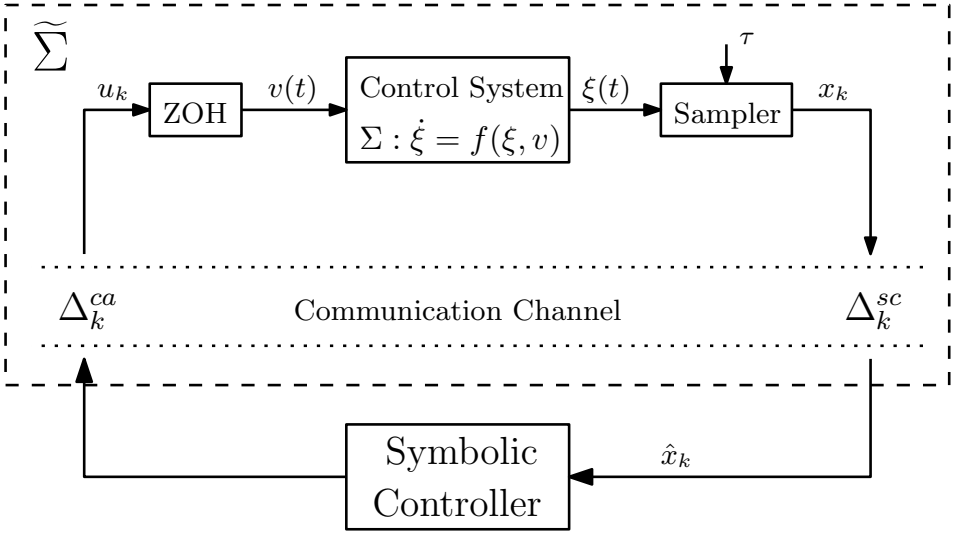
- Time-varying sampling/transmission intervals.
- Time-varying communication delays
- Limited bandwidth.
- Quantization errors.
- Packet dropouts.

Control of NCS:

- Traditionally: classical control, existing methods are limited to stabilizability problem.
- Recently: formal methods, automated synthesis, correct-by-construction, developing research area.
- **SENSE**: a framework for symbolic control of NCS

SENSE: Modeling the NCS $\widetilde{\Sigma}$

- A control system (plant): $\Sigma = (\mathbb{R}^n, \mathsf{U}, \mathcal{U}, f)$.
- A sensor/sampler (time-driven): non-varying sampling-time: τ.
- A zero-order-hold (ZOH).
- Two communication channels:
  - Delays as integer multiples of τ: $\Delta_k^{sc} := N_k^{sc}\tau$ and $\Delta_k^{ca} := N_k^{ca}\tau$.
  - Delays are bounded: $N_k^{sc} \in [N_{\min}^{sc}; N_{\max}^{sc}] \subset \mathbb{N}_+$ $N_k^{ca} \in [N_{\min}^{ca}; N_{\max}^{ca}] \subset \mathbb{N}_+$.



Considered Imperfections:

- Bounded delays.
- Packet dropouts (emulation).
- Limited bandwidth.
- Quantization errors.

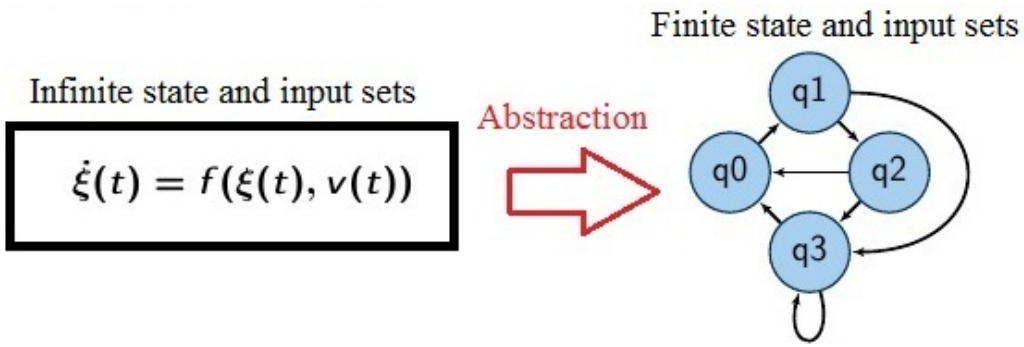
Next

- What is symbolic control ?
- How to apply it to NCS ?

Symbolic Control: Abstraction-based Synthesis

Abstraction[1]**:**

- Plants: physical systems (e.g. differential equations).
- Symbolic models: finite state/input systems representing plants up to some predefined accuracy.



Specifications:

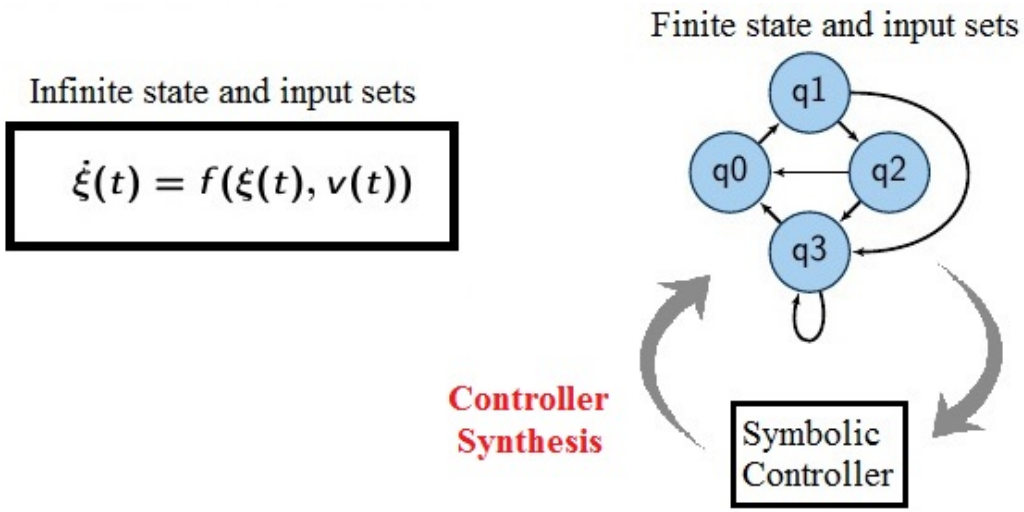
- Linear temporal logic (LTL) formulae; or
- Automata on infinite strings.

[1] P. Tabuada, Verification and control of hybrid systems. New York, Springer, 2009.

Symbolic Control: Abstraction-based Synthesis

Abstraction[1]**:**

- Plants: physical systems (e.g. differential equations).
- Symbolic models: finite state/input systems representing plants up to some predefined accuracy.



Specifications:

- Linear temporal logic (LTL) formulae; or
- Automata on infinite strings.

Controller Synthesis:

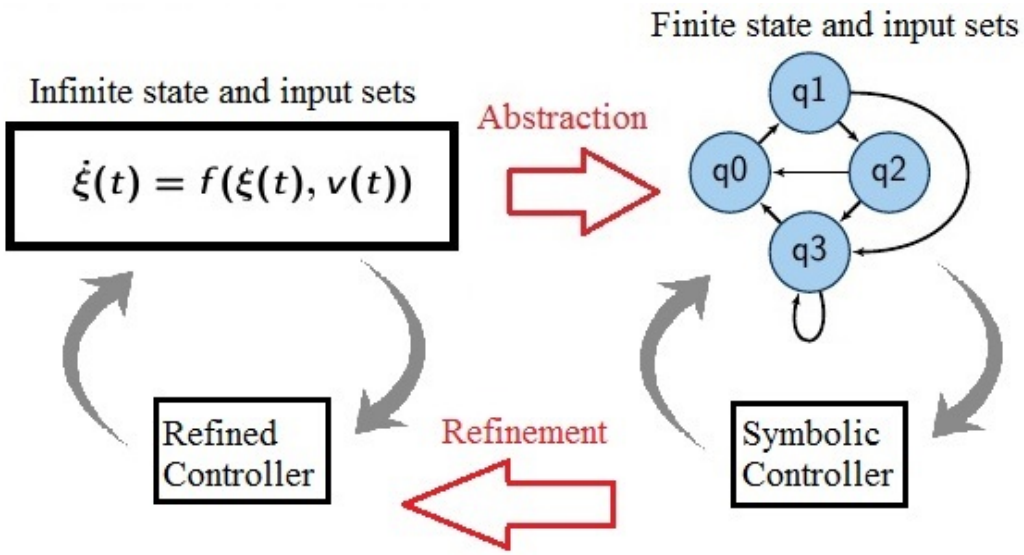
- Automated synthesis: Fixed-point or graph-search algorithms.
- Complex specifications can be handled.
- Symbolic controllers are refined with suitable interfaces.

[1] P. Tabuada, Verification and control of hybrid systems. New York, Springer, 2009.

Symbolic Control: Abstraction-based Synthesis

Abstraction[1]:

- Plants: physical systems (e.g. differential equations).
- Symbolic models: finite state/input systems representing plants up to some predefined accuracy.



Specifications:

- Linear temporal logic (LTL) formulae; or
- Automata on infinite strings.

Controller Synthesis:

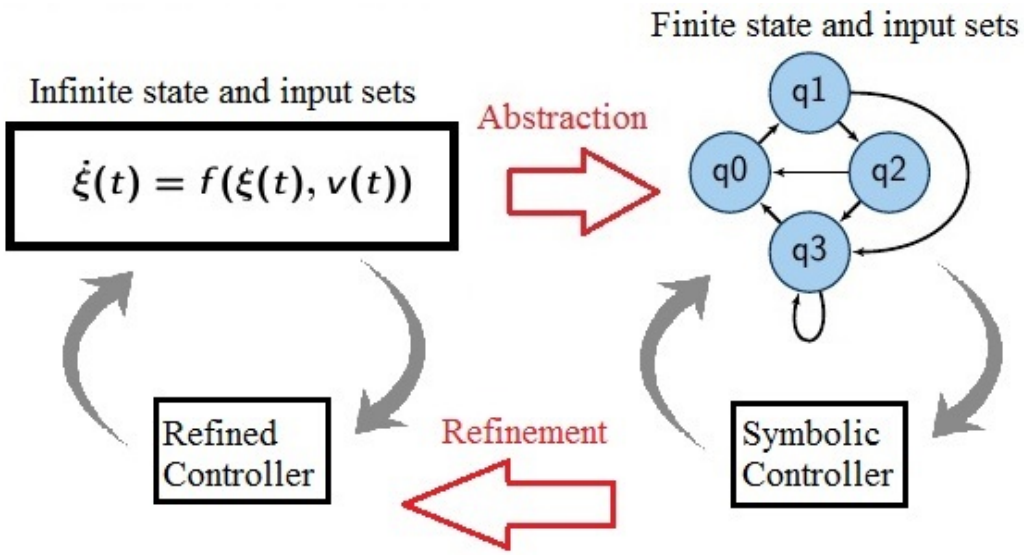
- Automated synthesis: Fixed-point or graph-search algorithms.
- Complex specifications can be handled.
- Symbolic controllers are refined with suitable interfaces.

[1] P. Tabuada, Verification and control of hybrid systems. New York, Springer, 2009.

Symbolic Control: Abstraction-based Synthesis

Abstraction[1]:

- Plants: physical systems (e.g. differential equations).
- Symbolic models: finite state/input systems representing plants up to some predefined accuracy.



Specifications:

- Linear temporal logic (LTL) formulae; or
- Automata on infinite strings.

Controller Synthesis:

- Automated synthesis: Fixed-point or graph-search algorithms.
- Complex specifications can be handled.
- Symbolic controllers are refined with suitable interfaces.

[1] P. Tabuada, Verification and control of hybrid systems. New York, Springer, 2009.

Symbolic Control of NCS: Plants as Systems

Definition 1: Systems

A system S is a tuple

$$S = (X, X_0, U, \longrightarrow)$$

consisting of:

- a (possibly infinite) set of states X.
- a (possibly infinite) set of initial states $X_0 \subseteq X$.
- a (possibly infinite) set of inputs U.
- a transition relation $\longrightarrow \subseteq X \times U \times X$.

Describing the plant as a system:
Given a sampling period τ,

$$\Sigma : \dot{\xi}(t) = f(\xi(t), v(t))$$

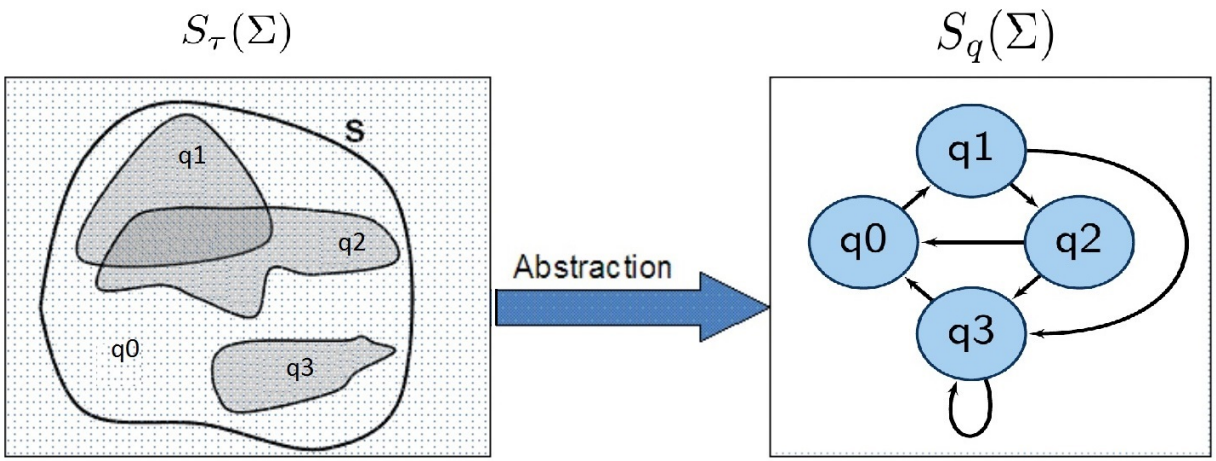
is described by the system:

$$S_\tau(\Sigma) = (X_\tau, X_{\tau,0}, U_\tau, \underset{\tau}{\longrightarrow})$$

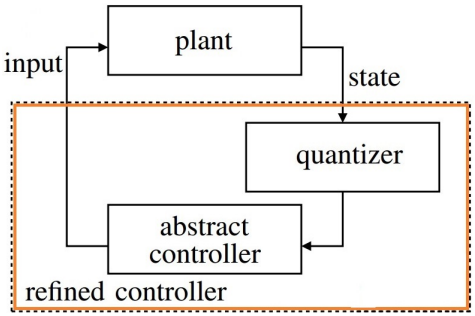
which encapsulates all the information contained in the control system Σ at sampling times $k\tau$, $k \in \mathbb{N}_0$, where:

- $X_\tau = \mathbb{R}^n$, $X_{\tau,0} = \mathbb{R}^n$.
- $U_\tau = \{v : \mathbb{R}_0^+ \to \mathrm{U} | v(t) = v((s-1)\tau), t \in [(s-1)\tau, s\tau[, s \in \mathbb{N}\}$.
- $x_\tau \xrightarrow[\tau]{v_\tau} x'_\tau \iff \exists \xi_{x_\tau v_\tau} : [0, \tau] \to \mathbb{R}^n$ in the system Σ s.t. $\xi_{x_\tau v_\tau}(\tau) = x'_\tau$.

Symbolic Control of NCS: Symbolic Models of Plants

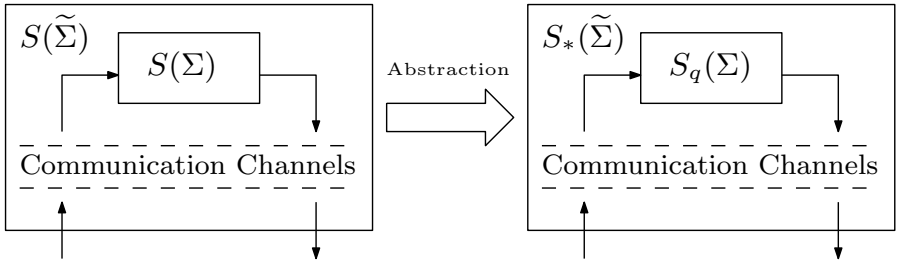


- Used relation: feedback refinement relation[2](denoted by Q).
- $S_q(\Sigma) = (X_q, X_q, U_\tau, \underset{q}{\longrightarrow})$
- X_q is a finite cover/partition over X_τ
- Advantage: only a static quantization map to refine the synthesized controller.



[2] G. Reissig, A. Weber and M. Rungger, "Feedback Refinement Relations for the Synthesis of Symbolic Controllers," in IEEE Transactions on Automatic Control

SENSE: Symbolic Models of NCS



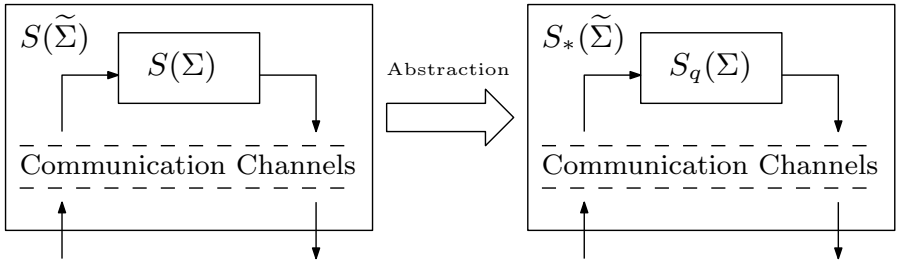
- The $\mathcal{L}$-map is introduced[3]::

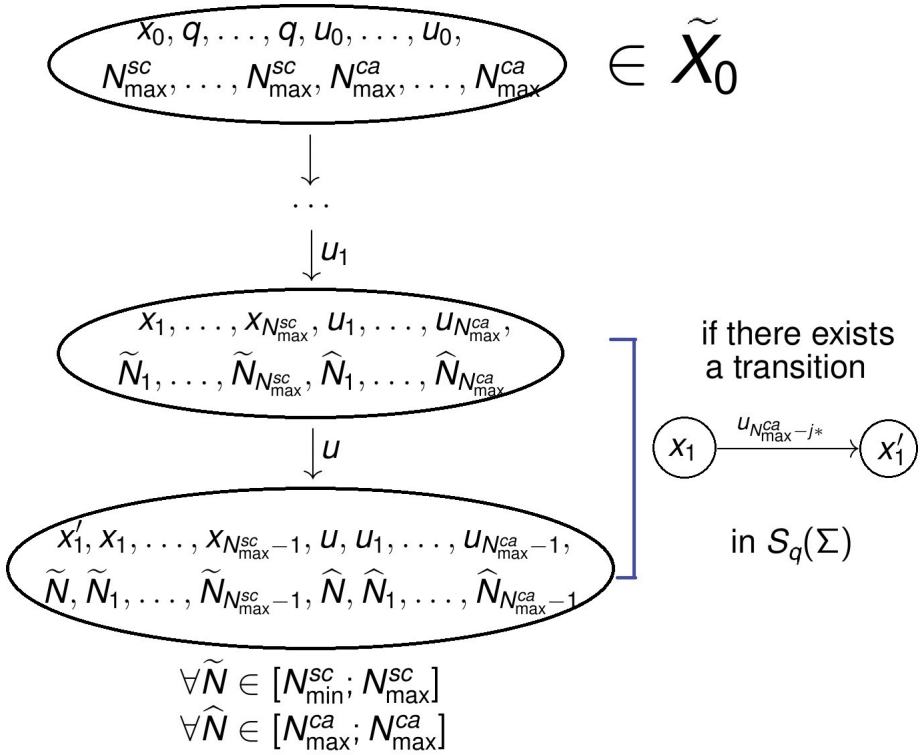
$$\mathcal{L} : \mathcal{T}(U, X) \times \mathbb{N}^4 \to \mathcal{T}(U, \widetilde{X}).$$

$$\begin{aligned} S_*(\widetilde{\Sigma}) &= (\widetilde{X}, \widetilde{X}_0, U_\tau, \underset{*}{\longrightarrow}) \\ &= \mathcal{L}(S_q(\Sigma), N^{sc}_{\min}, N^{sc}_{\max}, N^{ca}_{\min}, N^{ca}_{\max}), \end{aligned}$$

[3] M. Khaled, M. Rungger and M. Zamani (2016): Symbolic models of networked control systems: A feedback refinement relation approach. In: 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 187–193

SENSE: Symbolic Models of NCS





- The $\mathcal{L}$-map is introduced[3]::

$$\mathcal{L} : \mathcal{T}(U, X) \times \mathbb{N}^4 \to \mathcal{T}(U, \widetilde{X}).$$

$$\begin{aligned} S_*(\widetilde{\Sigma}) &= (\widetilde{X}, \widetilde{X}_0, U_\tau, \underset{*}{\longrightarrow}) \\ &= \mathcal{L}(S_q(\Sigma), N^{sc}_{\min}, N^{sc}_{\max}, N^{ca}_{\min}, N^{ca}_{\max}), \end{aligned}$$

where

- $\widetilde{X} = \{X_q \cup q\}^{N^{sc}_{\max}} \times U_\tau^{N^{ca}_{\max}} \times [N^{sc}_{\min}; N^{sc}_{\max}]^{N^{sc}_{\max}} \times [N^{ca}_{\min}; N^{ca}_{\max}]^{N^{ca}_{\max}}$, where q is a dummy symbol.
- $\widetilde{X}_0$ and $\underset{*}{\longrightarrow}$ are depicted visually.

[3] M. Khaled, M. Rungger and M. Zamani (2016): Symbolic models of networked control systems: A feedback refinement relation approach. In: 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 187–193

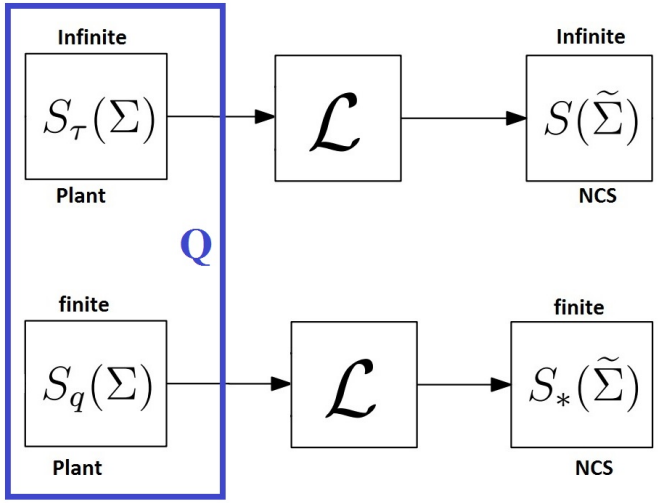
SENSE: Efficient Symbolic[4] Abstractions of NCS

Remark 1

The map $\mathcal{L}$ provides a systematic technique of constructing the NCS from its plant and the delays.

Theorem 1

Consider an NCS $\widetilde{\Sigma}$ and suppose there exists a simple finite system $S_q(\Sigma)$ such that $S_\tau(\Sigma) \preccurlyeq_Q S_q(\Sigma)$. Then we have $S(\widetilde{\Sigma}) \preccurlyeq_{\widetilde{Q}} S_*(\widetilde{\Sigma})$, for some feedback refinement relation $\widetilde{Q}$, where $S_*(\widetilde{\Sigma}) := \mathcal{L}(S_q(\Sigma), N^{sc}_{min}, N^{sc}_{max}, N^{ca}_{min}, N^{ca}_{max})$ is finite.



[4] M. Khaled, M. Rungger and M. Zamani (2016): Symbolic models of networked control systems: A feedback refinement relation approach. 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)

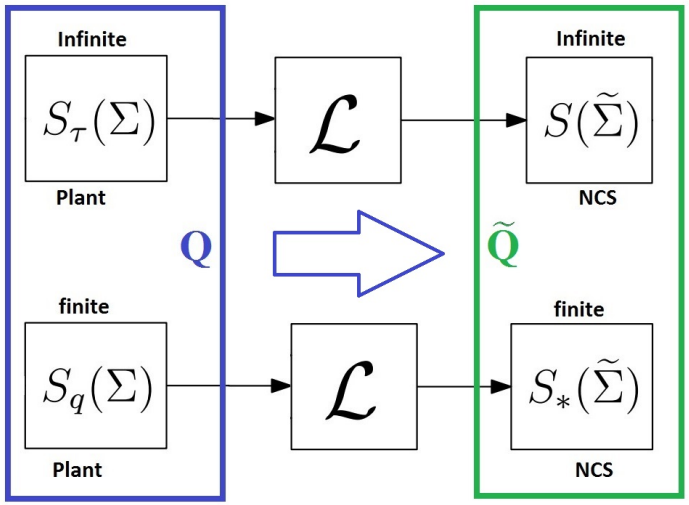
SENSE: Efficient Symbolic[4] Abstractions of NCS

Remark 1

The map $\mathcal{L}$ provides a systematic technique of constructing the NCS from its plant and the delays.

Theorem 1

Consider an NCS $\widetilde{\Sigma}$ and suppose there exists a simple finite system $S_q(\Sigma)$ such that $S_\tau(\Sigma) \preccurlyeq_Q S_q(\Sigma)$. Then we have $S(\widetilde{\Sigma}) \preccurlyeq_{\widetilde{Q}} S_*(\widetilde{\Sigma})$, for some feedback refinement relation $\widetilde{Q}$, where $S_*(\widetilde{\Sigma}) := \mathcal{L}(S_q(\Sigma), N^{sc}_{min}, N^{sc}_{max}, N^{ca}_{min}, N^{ca}_{max})$ is finite.



[4] M. Khaled, M. Rungger and M. Zamani (2016): Symbolic models of networked control systems: A feedback refinement relation approach. 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)

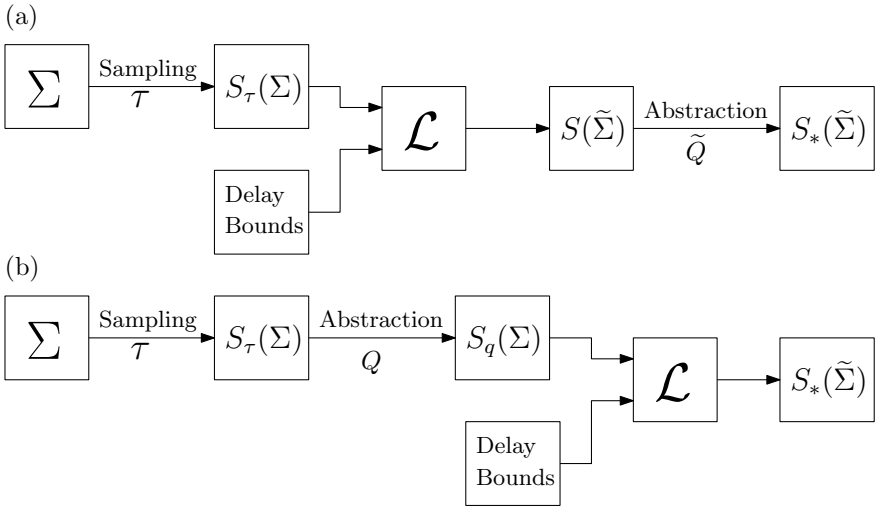
SENSE: Efficient Symbolic[4] Abstractions of NCS

Remark 1

The map $\mathcal{L}$ provides a systematic technique of constructing the NCS from its plant and the delays.

Theorem 1

Consider an NCS $\widetilde{\Sigma}$ and suppose there exists a simple finite system $S_q(\Sigma)$ such that $S_\tau(\Sigma) \preccurlyeq_Q S_q(\Sigma)$. Then we have $S(\widetilde{\Sigma}) \preccurlyeq_{\widetilde{Q}} S_*(\widetilde{\Sigma})$, for some feedback refinement relation $\widetilde{Q}$, where $S_*(\widetilde{\Sigma}) := \mathcal{L}(S_q(\Sigma), N^{sc}_{min}, N^{sc}_{max}, N^{ca}_{min}, N^{ca}_{max})$ is finite.



Remark 2

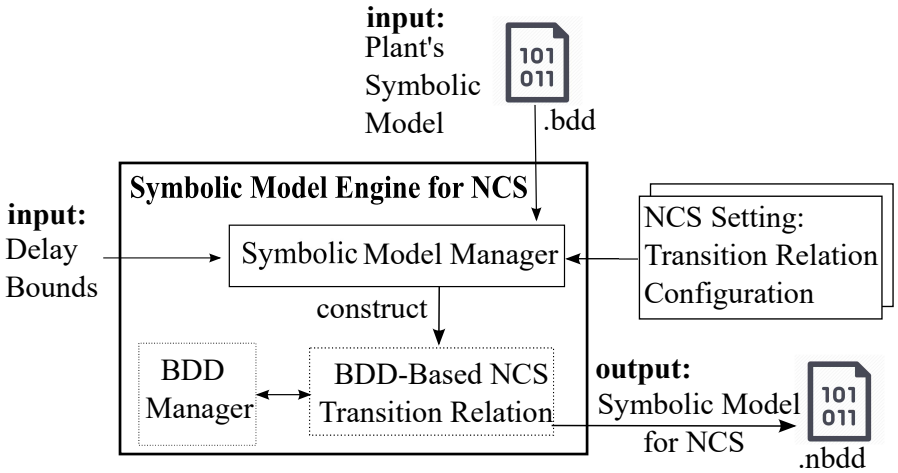
a- **Normal methodology:** one should first derive an infinite system $S(\widetilde{\Sigma})$ that captures all NCS information and then, construct the symbolic model $S_*(\widetilde{\Sigma})$ from it.

b- **Efficient methodology:** systematically construct the symbolic abstraction $S_*(\widetilde{\Sigma})$ directly from the symbolic model $S_q(\Sigma)$.

[4] M. Khaled, M. Rungger and M. Zamani (2016): Symbolic models of networked control systems: A feedback refinement relation approach. 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)

SENSE: Abstraction Construction Engine

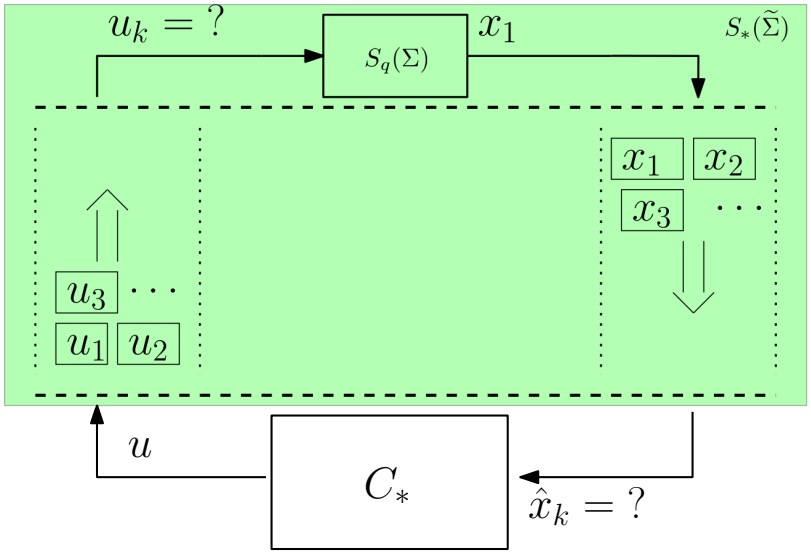
- BDD-based abstraction construction:
  - $S_q(\Sigma)$ is given as a BDD-object.
  - The library SCOTS can be used to generate $S_q(\Sigma)$.
  - Apply $\mathcal{L}$ to $S_q(\Sigma)$ via BDD operations.
  - NCS settings guides the engine about the class of NCS.
  - Fully customizable: different classes of NCS.



- Efficient implementation of the map $\mathcal{L}$
  - BDD bit-list of $S_q(\Sigma)$ gets more bits for: $\{X_q \cup q\}^{N^{sc}_{max}}$, $\{U^{N^{ca}_{max}}$, $[N^{sc}_{\min}; N^{sc}_{\max}]^{N^{sc}_{\max}}$, and $[N^{ca}_{\min}; N^{ca}_{\max}]^{N^{ca}_{\max}}$.
  - Transition relation of $S_*(\widetilde{\Sigma})$: BDD operations on expanded $S_q(\Sigma)$.

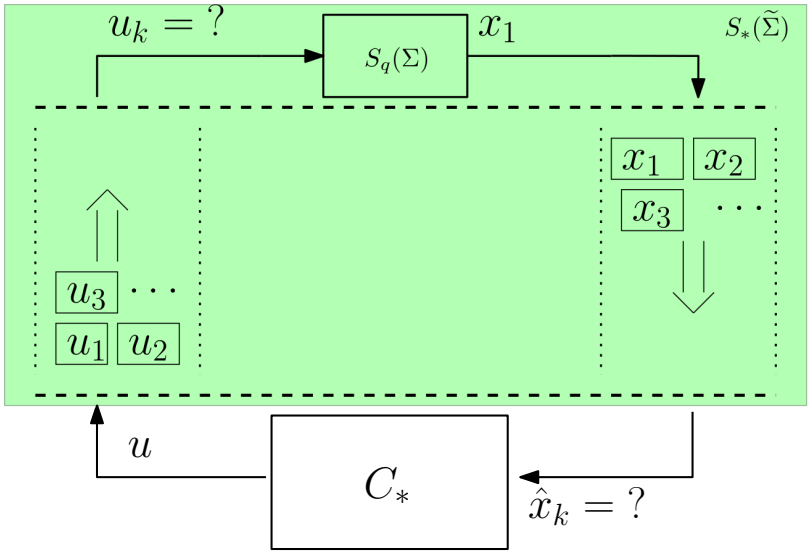
SENSE: Controller Synthesis and Refinement

- Synthesis of symbolic controllers C_*:
  - BDD-based fixed-point operations on $S_*(\widetilde{\Sigma})$.
  - Supports safety, reachability, persistence and recurrence specifications.



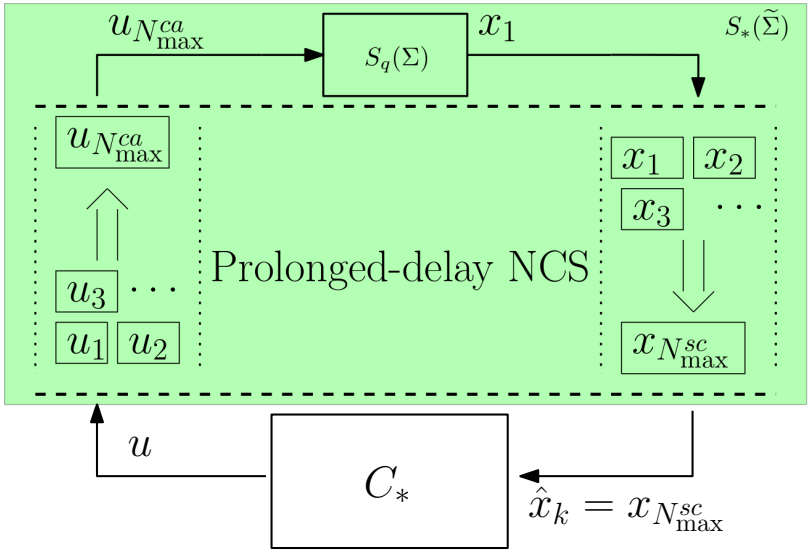
SENSE: Controller Synthesis and Refinement

- Synthesis of symbolic controllers C_*:
  - BDD-based fixed-point operations on $S_*(\widetilde{\Sigma})$.
  - Supports safety, reachability, persistence and recurrence specifications.
- **Problem:**
  - C_* has no access to full state of NCS.
  - Not possible to construct the states of $S_*(\widetilde{\Sigma})$.



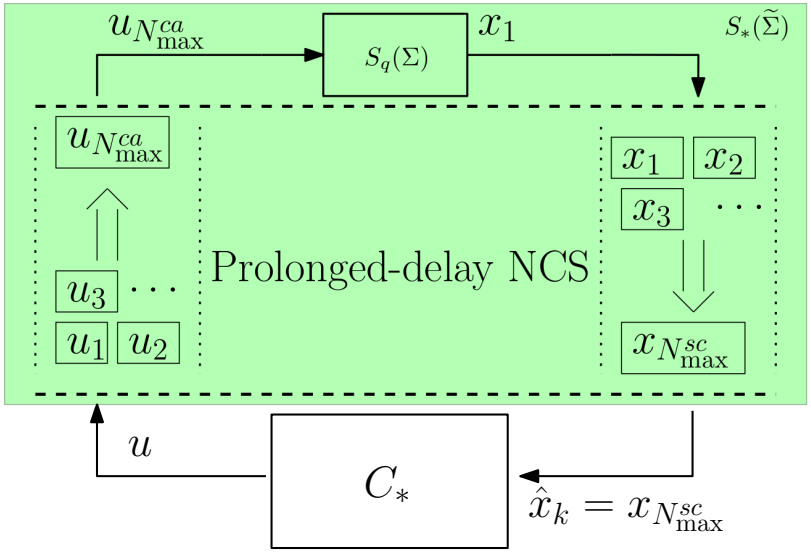
SENSE: Controller Synthesis and Refinement

- Synthesis of symbolic controllers C_*:
  - BDD-based fixed-point operations on $S_*(\widetilde{\Sigma})$.
  - Supports safety, reachability, persistence and recurrence specifications.
- **Problem:**
  - C_* has no access to full state of NCS.
  - Not possible to construct the states of $S_*(\widetilde{\Sigma})$.
- **Solution:** delay prolongation:
  - All packets suffer same delay.
  - Realizable via buffers.
  - Results in simpler map $\mathcal{L}$.
  - Less conservative for existence of controllers.
  - Controllers refined with state reconstructors.



SENSE: Controller Synthesis and Refinement

- Synthesis of symbolic controllers C_*:
  - BDD-based fixed-point operations on $S_*(\widetilde{\Sigma})$.
  - Supports safety, reachability, persistence and recurrence specifications.
- **Problem:**
  - C_* has no access to full state of NCS.
  - Not possible to construct the states of $S_*(\widetilde{\Sigma})$.
- **Solution:** delay prolongation:
  - All packets suffer same delay.
  - Realizable via buffers.
  - Results in simpler map $\mathcal{L}$.
  - Less conservative for existence of controllers.
  - Controllers refined with state reconstructors.



Remark 3

SENSE constructs abstractions of any class of NCS. However, only to resolve the refinement issue, prolonged-delay NCS need to be considered.

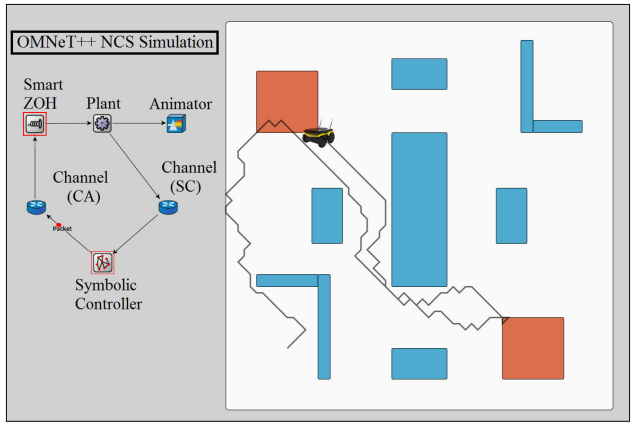
SENSE: Simulation, Analysis and Code Generation

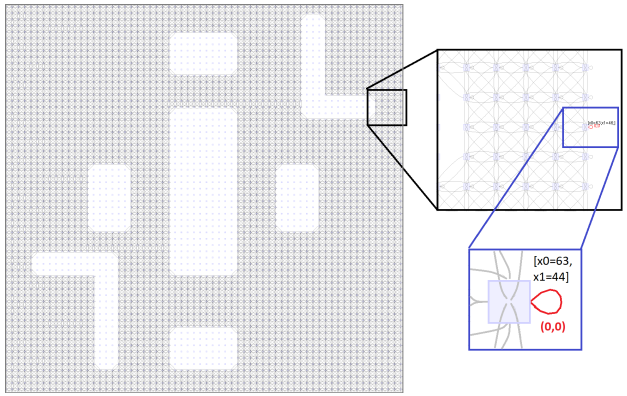
Interfaces to simulate the closed-loop:

- MATLAB: closed-loop simulation.
- OMNet++: realistic NCS simulation and visualization.

Supporting tools:

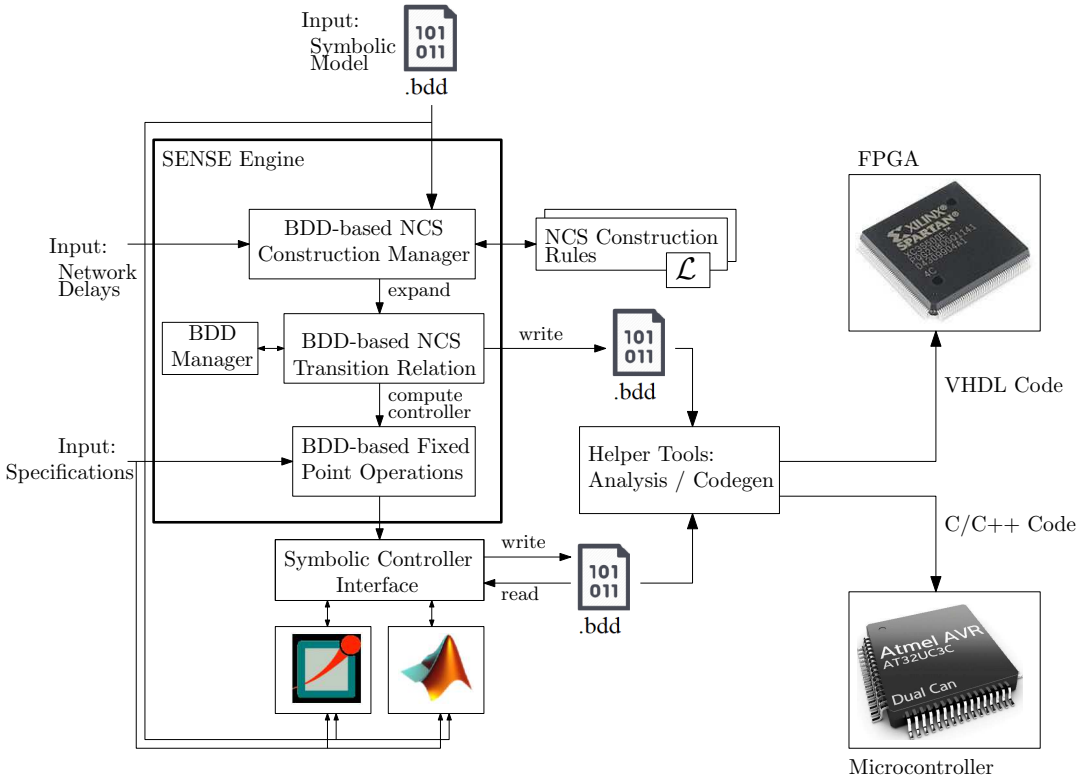
- `bdd2implement`: automatically generate C/C++ or VHDL/Verilog for controllers.
- `bdd2fsm`: generate files following the FSM data format to be used for visualization.
- `bddDump`: extract the meta-data information stored inside BDD files.
- `contCoverage`: fast terminal-based ASCII-art visualization.
- `sysExplorer`: testing the expanded transition relation.





SENSE: Work-Flow Summary

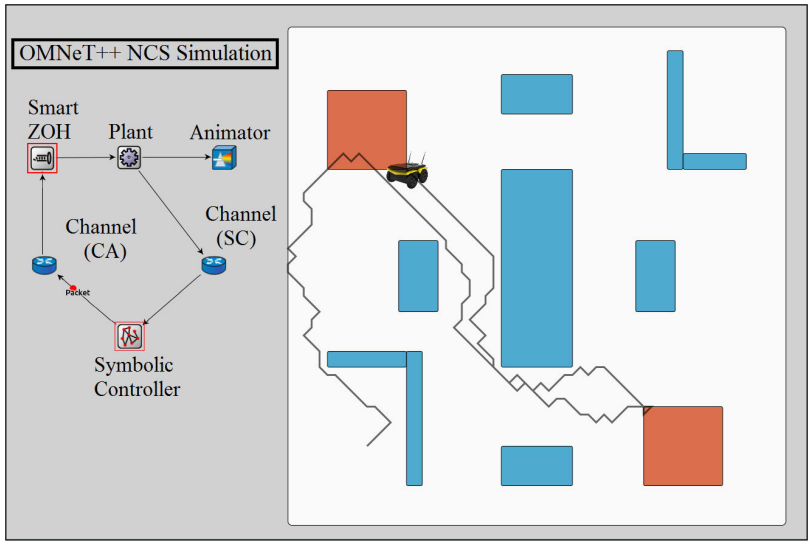
- Compute $S_q(\Sigma)$ as a BDD using SCOTS.
- Pass the NCS delays and BDD file of $S_q(\Sigma)$ to SENSE.
- SENSE takes the specifications as input.
- SENSE generates BDD files for $S_*(\widetilde{\Sigma})$ and C_*.
- Use the interfaces for MATLAB and/or OMNet++ to simulate/visualize the closed-loop NCS.
- Use `bdd2implement` to generate C/C++ or VHDL/Verilog source codes for implementations of final controllers.



- Use the interfaces for MATLAB and/or OMNet++ to simulate/visualize the closed-loop NCS.
- Use `bdd2implement` to generate the final controller for implementation.

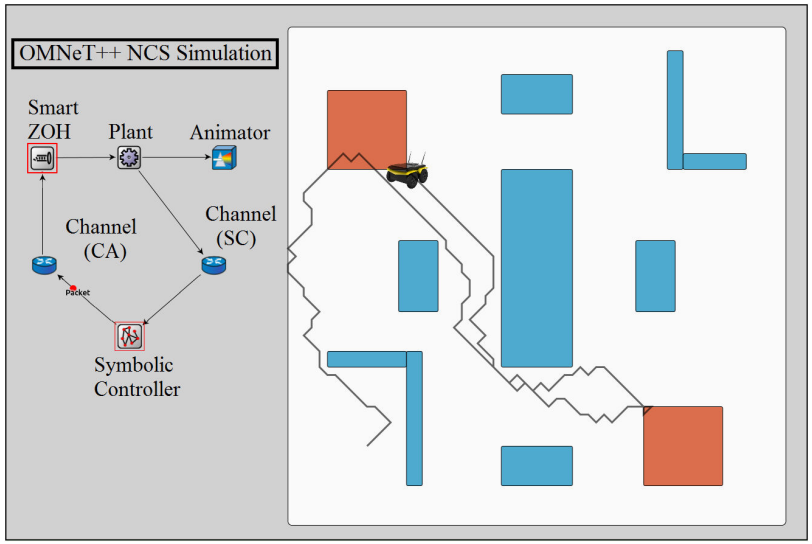
SENSE: Test Drive

- Available download/manual in:
  https://www.hcs.ei.tum.de
- Requires:
  - C/C++ Compiler.
  - CUDD Library.
  - MATLAB or OMNet++ (optional).
- Install and usage: Video !



SENSE: Test Drive

- Available download/manual in:
  https://www.hcs.ei.tum.de
- Requires:
  - C/C++ Compiler.
  - CUDD Library.
  - MATLAB or OMNet++ (optional).
- Install and usage: Video !



Example:
Robot in 2D arena:

$$\begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \end{bmatrix} = \begin{bmatrix} \upsilon_1 \\ \upsilon_2 \end{bmatrix}$$

$(\xi_1, \xi_2) \in X = [0, 64] \times [0, 64]$ quantized by $(1, 1)$.
$(\upsilon_1, \upsilon_2) \in U = [-1, 1] \times [-1, 1]$ quantized by $(1, 1)$.

Specifications:

$$\psi = \left(\bigwedge_{i=1}^{9} \Box(\neg \text{Obstacle}_i)\right) \wedge \Box\Diamond(\text{Target1}) \wedge \Box\Diamond(\text{Target2}),$$

SENSE: Test Drive

- Available download/manual in:
  https://www.hcs.ei.tum.de
- Requires:
  - C/C++ Compiler.
  - CUDD Library.
  - MATLAB or OMNet++ (optional).
- Install and usage: Video !

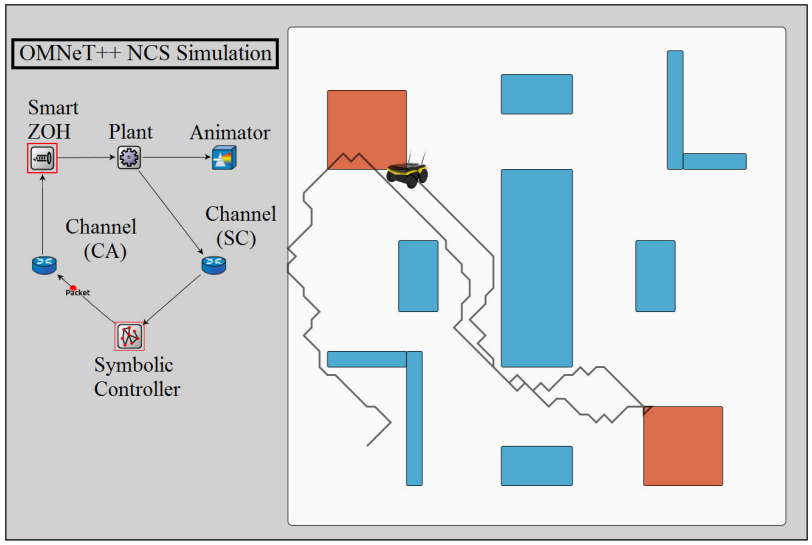
Example:
Robot in 2D arena:

$$\begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \end{bmatrix} = \begin{bmatrix} \upsilon_1 \\ \upsilon_2 \end{bmatrix}$$

$(\xi_1, \xi_2) \in X = [0, 64] \times [0, 64]$ quantized by $(1, 1)$.
$(\upsilon_1, \upsilon_2) \in U = [-1, 1] \times [-1, 1]$ quantized by $(1, 1)$.

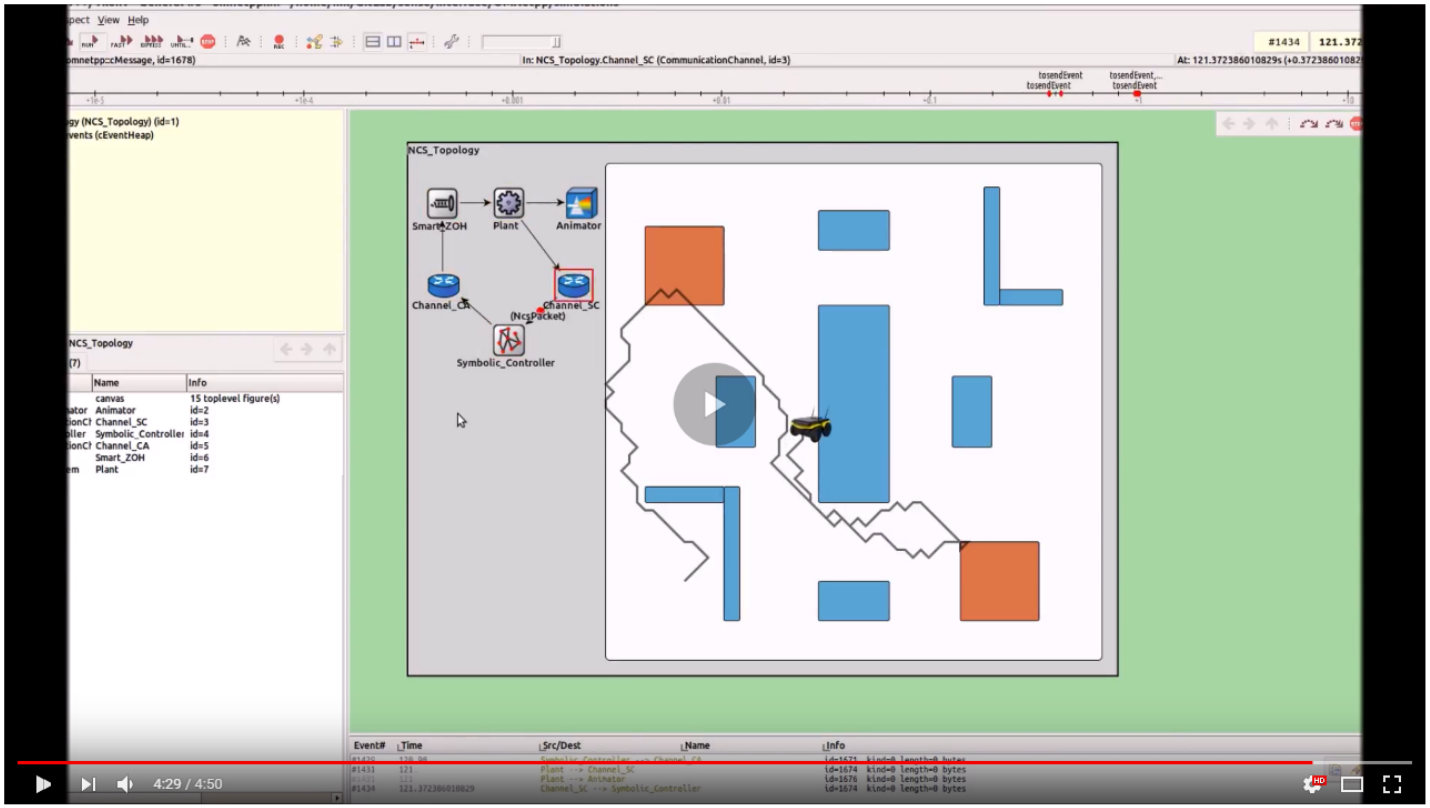
Specifications:

$$\psi = \left(\bigwedge_{i=1}^{9} \Box(\neg \text{Obstacle}_i)\right) \wedge \Box\Diamond(\text{Target1}) \wedge \Box\Diamond(\text{Target2}),$$



Results:
$S_*(\widetilde{\Sigma})$: in 0.49 seconds (15 KB).
C_*: in 8 seconds (11 KB).
Visualized by OMNet++.

Video: Installation and Usage of SENSE.



YouTube DE

Link

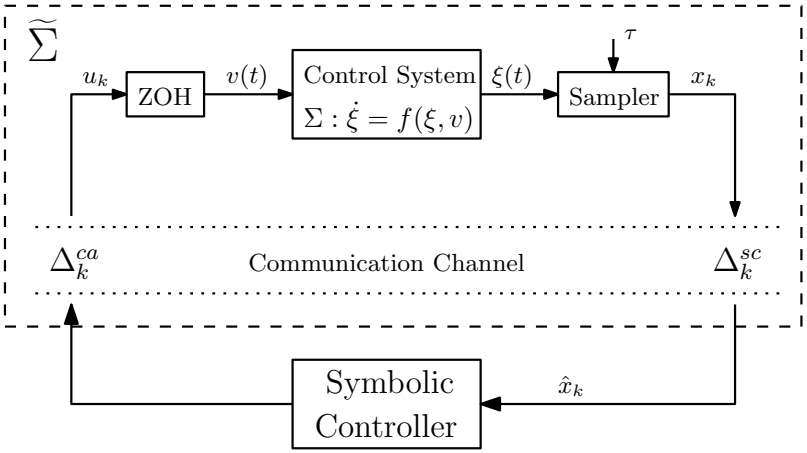
Thanks! Any Questions?

Table: Results for constructing symbolic models of prolonged-delay NCS using those of their plants.

Case Study	$\lvert S_q(\Sigma)\rvert$		(2,2)	(2,3)	(2,4)	(2,5)	(3,2)	(3,3)	(3,4)	(3,5)	(4,2)	(4,3)
Double	2039	$\lvert S_*(\widetilde{\Sigma})\rvert$	14096	56336	225296	901136	22832	91184	364592	1.45×10^{6}	38340	152964
		Time (sec)	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1
Integrator		Memory (KB)	2.0	2.4	3.1	2.9	3.0	2.9	3.1	3.2	5.2	4.3
Robot	29280	$\lvert S_*(\widetilde{\Sigma})\rvert$	4.1×10^{6}	6.5×10^{7}	1.04×10^{9}	1.67×10^{10}	3.4×10^{7}	5.4×10^{8}	8.7×10^{9}	1.4×10^{11}	2.8×10^{8}	4.6×10^{9}
		Time (sec)	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	1.4	1.6
		Memory (KB)	15	14	17	16	16	21	22	19	35	33
Jet Engine	9.0×10^{5}	$\lvert S_*(\widetilde{\Sigma})\rvert$	1.5×10^{11}	1.0×10^{13}	6.5×10^{14}	4.2×10^{16}	7.2×10^{12}	4.6×10^{14}	2.9×10^{16}	1.9×10^{19}	3.3×10^{14}	2.1×10^{16}
		Time (sec)	1970	1637	1674	2172	3408	1772	2111	7107	4011	2854
		Memory (KB)	4323.3	2374.2	2389.1	2392.6	3683.2	4098.2	3317.2	3582.6	5894.3	4784.5
DC-DC	3.8×10^{6}	$\lvert S_*(\widetilde{\Sigma})\rvert$	8.9×10^{7}	1.8×10^{8}	3.6×10^{8}	7.1×10^{8}	5.2×10^{8}	1.0×10^{9}	2.0×10^{9}	4.1×10^{9}	3.0×10^{9}	6.0×10^{9}
Converter		Time (sec)	672	681	530	1131	13690	10114	9791	10084	139693	137648
		Memory (KB)	3347.2	3145.0	3176.7	2784.8	10875.2	11543.8	11572.0	11592.1	34169.2	37852.6
Vehicle	1.9×10^{7}	$\lvert S_*(\widetilde{\Sigma})\rvert$	4.2×10^{12}	2.7×10^{14}	1.7×10^{16}	1.1×10^{18}	2.3×10^{14}	1.4×10^{16}	9.3×10^{17}	5.9×10^{19}	1.3×10^{16}	7.94×10^{17}
		Time (sec)	273.3	285	238.4	173	22344	54919	27667	36467	39065	145390
		Memory (KB)	1638.4	1945.6	1843.2	1945.6	23040	40652.8	30208	22425.6	21094.4	36556.3
Inverted	4.3×10^{8}	$\lvert S_*(\widetilde{\Sigma})\rvert$	2.4×10^{14}	1.5×10^{16}	1.0×10^{18}	6.5×10^{19}	4.6×10^{16}	2.9×10^{18}	1.9×10^{20}	1.2×10^{22}	9.2×10^{16}	5.8×10^{20}
Pendulum		Time (sec)	361.4	349.1	340.9	347.8	942	793	1218	910	58411	57110
		Memory (KB)	723.1	808.4	349.6	1012.7	2958	2840	3104	2898	35907	35628

Appendix 1: More detailed model

- A control system (plant): $\Sigma = (\mathbb{R}^n, \mathsf{U}, \mathcal{U}, f)$.
- A sensor/sampler (time-driven):
  - Non-varying sampling-time: τ.
  - $x_k := \xi(s_k)$.
  - Triggered: $s_k := k\tau$, $k \in \mathbb{N}_0$.
- A zero-order-hold (ZOH).
- Two communication channels:
  - Delays are integer multiples of the sampling time: $\Delta_k^{sc} := N_k^{sc}\tau$ and $\Delta_k^{ca} := N_k^{ca}\tau$.
  - Delays are bounded: $N_k^{sc} \in [N_{\min}^{sc}; N_{\max}^{sc}]$, $N_k^{ca} \in [N_{\min}^{ca}; N_{\max}^{ca}]$.
  - Packet dropouts: emulation (i.e. increasing the delays), assuming subsequent dropouts are bounded.



- Packet rejection:
  - $v(t) = u_{k+j*-N_{\max}^{ca}}$.
  - $j*$: time-index shifting used to determine the correct control input by taking care of message rejection due to out of order packet arrival: two packets arrive at the same time, reject the older one.
  - Applied to both channels.

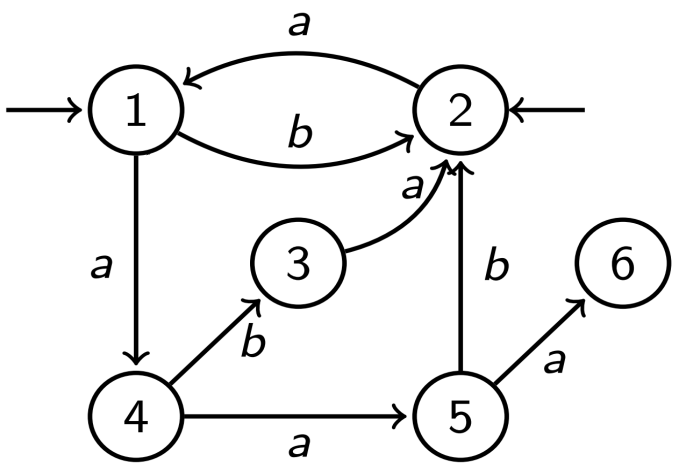
Appendix 2: Controller Synthesis and Refinement

- Fixed-point operations on $X \times U$.
- Possible LTL:
  $\Box\varphi_S$, $\Diamond\varphi_T$, $\Diamond\Box\varphi_S$, $\Box\Diamond\varphi_T$.
- A Pre-map is a base for all four algorithms:

$$\begin{aligned} pre(Z) =& \{(x,u) \in X_q \times U_q | \\ & \emptyset \neq F_q(x,u) \subseteq \pi_{X_q}(Z)\} \\ \pi_{X_q}(Z) =& \{x \in X_q | \exists_{u \in U_q} (x,u) \in Z\} \end{aligned}$$

SCOTS[5]:

- Symbolic Models: via FRR and encoded in BDD.
- Synthesis: $\Box\varphi_S$ and $\Diamond\varphi_T$ all via BDD operations.



Evolution of the Fixed-point set for the specification: $\Diamond 6$.

[5] M. Rungger and M. Zamani (2016): SCOTS: A Tool for the Synthesis of Symbolic Controllers. International Conference on Hybrid Systems: Computation and Control, HSCC 2016

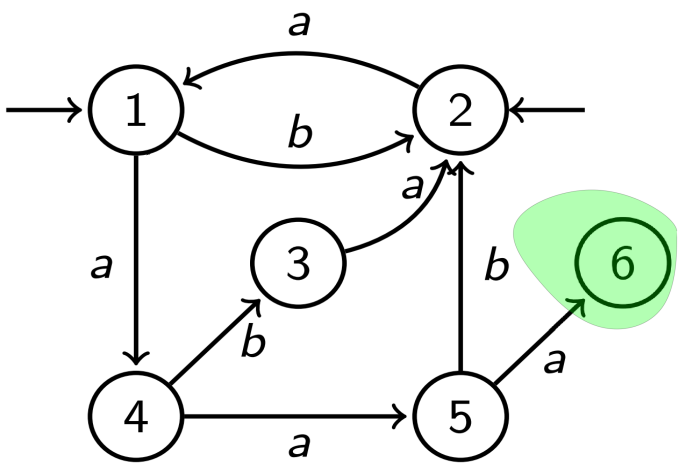
Appendix 2: Controller Synthesis and Refinement

- Fixed-point operations on $X \times U$.
- Possible LTL:
  $\Box\varphi_S$, $\Diamond\varphi_T$, $\Diamond\Box\varphi_S$, $\Box\Diamond\varphi_T$.
- A Pre-map is a base for all four algorithms:

$$\begin{aligned} pre(Z) =& \{(x,u) \in X_q \times U_q | \\ & \emptyset \neq F_q(x,u) \subseteq \pi_{X_q}(Z)\} \\ \pi_{X_q}(Z) =& \{x \in X_q | \exists_{u \in U_q} (x,u) \in Z\} \end{aligned}$$

SCOTS[5]:

- Symbolic Models: via FRR and encoded in BDD.
- Synthesis: $\Box\varphi_S$ and $\Diamond\varphi_T$ all via BDD operations.



Evolution of the Fixed-point set for the specification: $\Diamond 6$.

[5] M. Rungger and M. Zamani (2016): SCOTS: A Tool for the Synthesis of Symbolic Controllers. International Conference on Hybrid Systems: Computation and Control, HSCC 2016

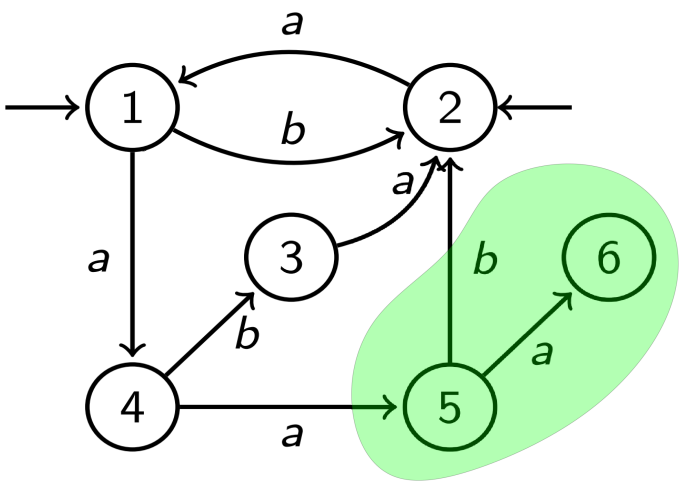
Appendix 2: Controller Synthesis and Refinement

- Fixed-point operations on $X \times U$.
- Possible LTL:
  $\Box\varphi_S$, $\Diamond\varphi_T$, $\Diamond\Box\varphi_S$, $\Box\Diamond\varphi_T$.
- A Pre-map is a base for all four algorithms:

$$\begin{aligned} pre(Z) =& \{(x,u) \in X_q \times U_q | \\ & \emptyset \neq F_q(x,u) \subseteq \pi_{X_q}(Z)\} \\ \pi_{X_q}(Z) =& \{x \in X_q | \exists_{u \in U_q} (x,u) \in Z\} \end{aligned}$$

SCOTS[5]:

- Symbolic Models: via FRR and encoded in BDD.
- Synthesis: $\Box\varphi_S$ and $\Diamond\varphi_T$ all via BDD operations.



Evolution of the Fixed-point set for the specification: $\Diamond 6$.

[5] M. Rungger and M. Zamani (2016): SCOTS: A Tool for the Synthesis of Symbolic Controllers. International Conference on Hybrid Systems: Computation and Control, HSCC 2016

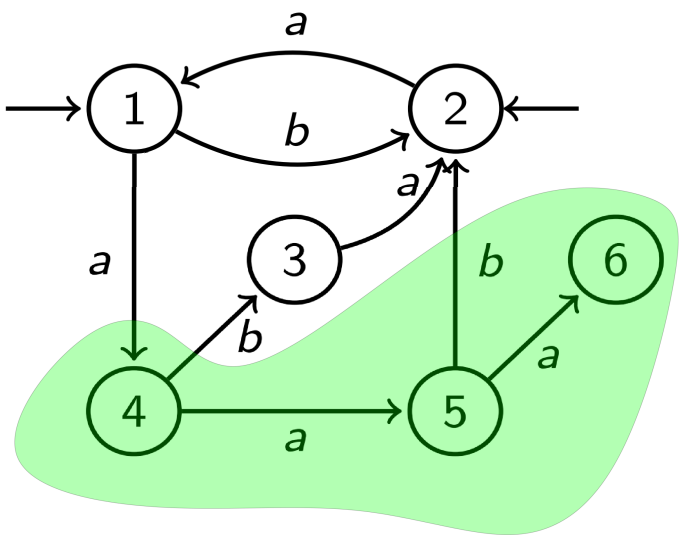
Appendix 2: Controller Synthesis and Refinement

- Fixed-point operations on $X \times U$.
- Possible LTL:
  $\Box\varphi_S$, $\Diamond\varphi_T$, $\Diamond\Box\varphi_S$, $\Box\Diamond\varphi_T$.
- A Pre-map is a base for all four algorithms:

$$\begin{aligned} pre(Z) =& \{(x,u) \in X_q \times U_q | \\ & \emptyset \neq F_q(x,u) \subseteq \pi_{X_q}(Z)\} \\ \pi_{X_q}(Z) =& \{x \in X_q | \exists_{u \in U_q} (x,u) \in Z\} \end{aligned}$$

SCOTS[5]:

- Symbolic Models: via FRR and encoded in BDD.
- Synthesis: $\Box\varphi_S$ and $\Diamond\varphi_T$ all via BDD operations.



Evolution of the Fixed-point set for the specification: $\Diamond 6$.

[5] M. Rungger and M. Zamani (2016): SCOTS: A Tool for the Synthesis of Symbolic Controllers. International Conference on Hybrid Systems: Computation and Control, HSCC 2016

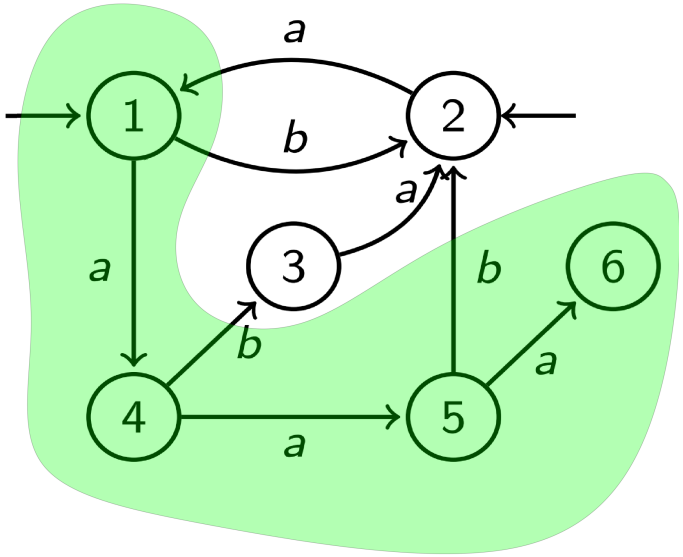
Appendix 2: Controller Synthesis and Refinement

- Fixed-point operations on $X \times U$.
- Possible LTL:
  $\Box\varphi_S$, $\Diamond\varphi_T$, $\Diamond\Box\varphi_S$, $\Box\Diamond\varphi_T$.
- A Pre-map is a base for all four algorithms:

$$\begin{aligned} pre(Z) =& \{(x,u) \in X_q \times U_q | \\ & \emptyset \neq F_q(x,u) \subseteq \pi_{X_q}(Z)\} \\ \pi_{X_q}(Z) =& \{x \in X_q | \exists_{u \in U_q} (x,u) \in Z\} \end{aligned}$$

SCOTS[5]:

- Symbolic Models: via FRR and encoded in BDD.
- Synthesis: $\Box\varphi_S$ and $\Diamond\varphi_T$ all via BDD operations.



Evolution of the Fixed-point set for the specification: $\Diamond 6$.

[5] M. Rungger and M. Zamani (2016): SCOTS: A Tool for the Synthesis of Symbolic Controllers. International Conference on Hybrid Systems: Computation and Control, HSCC 2016

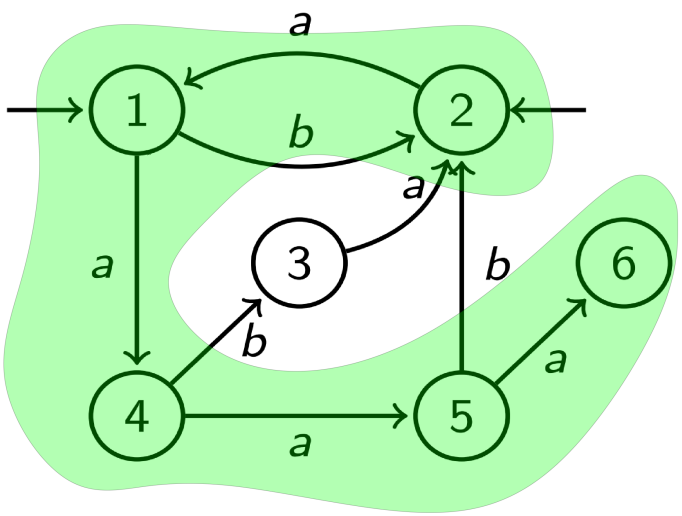
Appendix 2: Controller Synthesis and Refinement

- Fixed-point operations on $X \times U$.
- Possible LTL:
  $\Box\varphi_S$, $\Diamond\varphi_T$, $\Diamond\Box\varphi_S$, $\Box\Diamond\varphi_T$.
- A Pre-map is a base for all four algorithms:

$$\begin{aligned} pre(Z) =& \{(x,u) \in X_q \times U_q | \\ & \emptyset \neq F_q(x,u) \subseteq \pi_{X_q}(Z)\} \\ \pi_{X_q}(Z) =& \{x \in X_q | \exists_{u \in U_q} (x,u) \in Z\} \end{aligned}$$

SCOTS[5]:

- Symbolic Models: via FRR and encoded in BDD.
- Synthesis: $\Box\varphi_S$ and $\Diamond\varphi_T$ all via BDD operations.



Evolution of the Fixed-point set for the specification: $\Diamond 6$.

[5] M. Rungger and M. Zamani (2016): SCOTS: A Tool for the Synthesis of Symbolic Controllers. International Conference on Hybrid Systems: Computation and Control, HSCC 2016

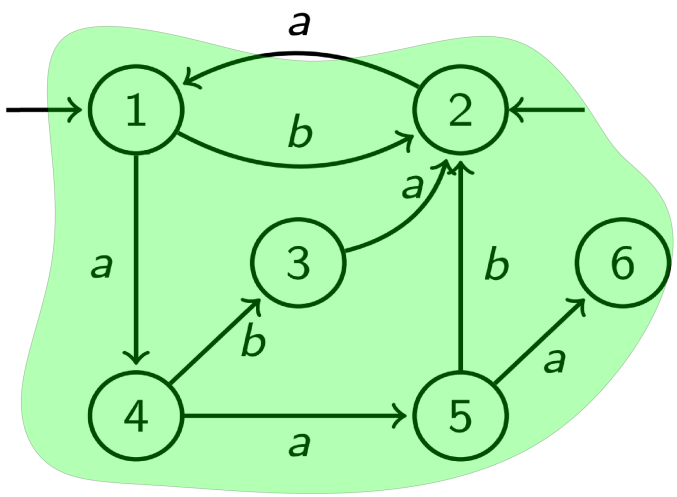
Appendix 2: Controller Synthesis and Refinement

- Fixed-point operations on $X \times U$.
- Possible LTL:
  $\Box\varphi_S$, $\Diamond\varphi_T$, $\Diamond\Box\varphi_S$, $\Box\Diamond\varphi_T$.
- A Pre-map is a base for all four algorithms:

$$\begin{aligned} pre(Z) =& \{(x,u) \in X_q \times U_q | \\ & \emptyset \neq F_q(x,u) \subseteq \pi_{X_q}(Z)\} \\ \pi_{X_q}(Z) =& \{x \in X_q | \exists_{u \in U_q} (x,u) \in Z\} \end{aligned}$$

SCOTS[5]:

- Symbolic Models: via FRR and encoded in BDD.
- Synthesis: $\Box\varphi_S$ and $\Diamond\varphi_T$ all via BDD operations.



Evolution of the Fixed-point set for the specification: $\Diamond 6$.

[5] M. Rungger and M. Zamani (2016): SCOTS: A Tool for the Synthesis of Symbolic Controllers. International Conference on Hybrid Systems: Computation and Control, HSCC 2016