Major Computational Breakthroughs in the Synthesis of Symbolic Controllers via Decomposed Algorithms

Eric S. Kim, Mahmoud Khaled, Murat Arcak, and Majid Zamani

HSCC 2018 : Poster Session, Porto, Portugal
April 11, 2018

Professorship of
Hybrid Control Systems

Computational Bottlenecks in Symbolic Control Synthesis

Problem: Abstraction and controller synthesis exhibit exponential time and space bottlenecks with respect to state + input space dimension in existing tools.
Contributions: Two approaches that leverage the inherent parallelism and structure in system dynamics

1. **Exploiting State Independence (Right column):** Parallelized core takes advantage of independence and locality of abstraction and synthesis subroutines across different states.
2. **Exploiting Sparsity (Left column):** Leverages the coordinate structure of the state space and sparsity (if present) to eliminate redundant computations.

Example: Vehicle Dynamics

3-dimensional state $(x_1, x_2, x_3) \in X$ and 2-dimensional input $(u_1, u_2) \in U$.

$$\dot{x}_1 = u_1 \cos(\alpha + u_2)/\cos(\alpha)$$
$$\dot{x}_2 = u_1 \sin(\alpha + u_2)/\sin(\alpha)$$
$$\dot{x}_3 = u_1 \tan(u_2)$$

where $\alpha = \arctan(\tan(u_2)/2)$.

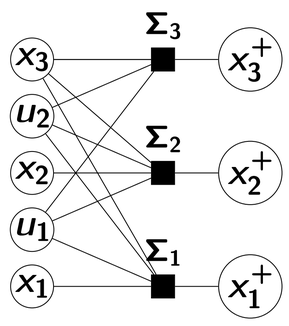
Discretization:

- 93k states: $51 \times 51 \times 36$ grid
- 49 inputs: 7×7 grid

Sparsity-Aware Abstraction

Each next state x_1^+, x_2^+, x_3^+ in discrete time vehicle dynamics only depends on a subset of $(x_1, x_2, x_3, u_1, u_2)$.

$$\Sigma(x, u, x^+) = \bigwedge_{i=1}^{3} \begin{pmatrix} \Sigma_1(x_1, x_3, u_1, u_2, x_1^+) \\ \Sigma_2(x_2, x_3, u_1, u_2, x_2^+) \\ \Sigma_3(x_3, u_1, u_2, x_3^+) \end{pmatrix}$$



Sparsity-aware abstraction computes and combines abstractions of lower dimensional components $\Sigma_1, \Sigma_2, \Sigma_3$ and eliminates redundant computations.

- **Equality:** Yields same abstraction as the regular algorithm.
- **Efficiency:** *Linear* with respect to state dimension, *exponential* with respect to sparsity parameter.
- **Generality:** Only assumption is Cartesian product state space and component-wise dynamics.

Decomposed Controller Predecessor

Controllable predecessor operator is a key subroutine in formal control synthesis:

$$\texttt{CPRE}(Z) = \exists u.\ (\exists x^+.\Sigma(x, u, x^+) \wedge \forall x^+.(\Sigma(x, u, x^+) \Rightarrow Z(x^+)))$$

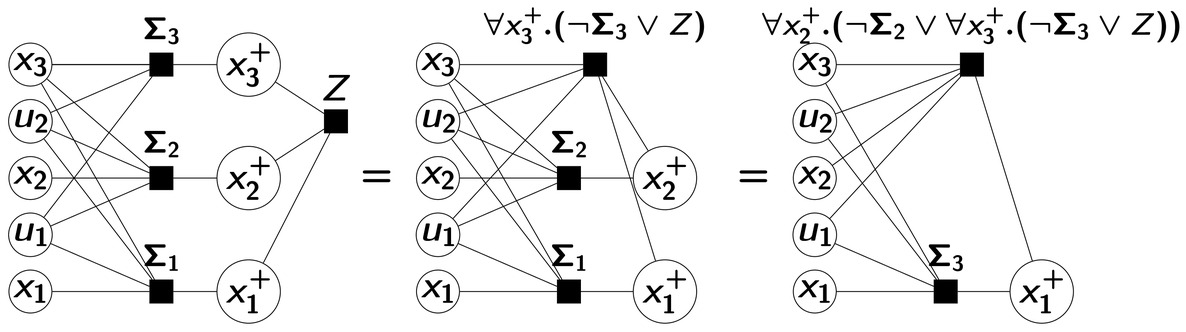
Goal: Create a controllable predecessor without constructing monolithic system $\Sigma(x, u, x^+)$. Substituting decomposed representation of Σ into red term yields

$$\begin{aligned} &\forall x^+.(\Sigma(x, u, x^+) \Rightarrow Z(x^+)) \\ &= \forall x^+.((\Sigma_1 \wedge \Sigma_2 \wedge \Sigma_3) \Rightarrow Z(x^+)) \\ &= \forall x_1^+.\forall x_2^+.\forall x_3^+.(\neg\Sigma_1 \vee \neg\Sigma_2 \vee \neg\Sigma_3 \vee Z(x^+)) \quad (1) \end{aligned}$$

Obstacle: The universal quantifier $\forall x^+$ doesn't distribute over disjunctions.
Solution: Iteratively eliminate x_1^+, x_2^+, x_3^+ variables over smaller formulas

$$\begin{aligned} \text{Equation } (1) &= \forall x_1^+.\forall x_2^+.(\neg\Sigma_1 \vee \neg\Sigma_2 \vee \forall x_3^+.(\neg\Sigma_3 \vee Z)) \quad (2) \\ &= \forall x_1^+.(\neg\Sigma_1 \vee \forall x_2^+.(\neg\Sigma_2 \vee \forall x_3^+(\neg\Sigma_3 \vee Z))) \quad (3) \end{aligned}$$

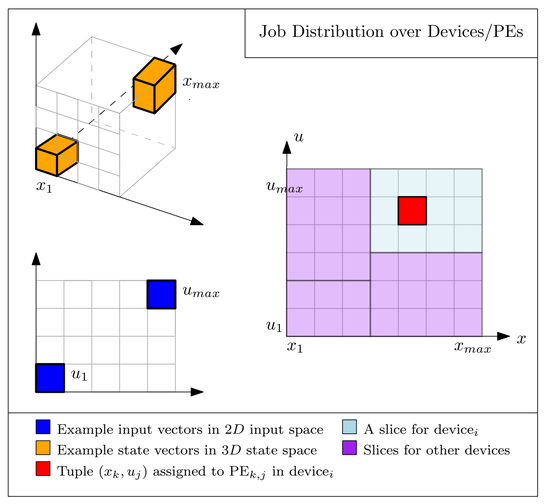
Visualization of equations (1), (2), (3):



Equation (1) Equation (2) Equation (3)

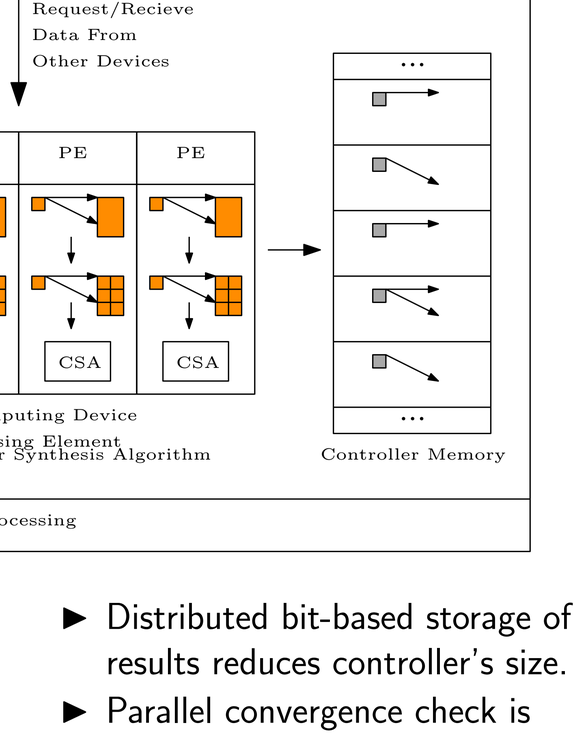
Preparations for Efficient Parallel Execution

- X and U: bounded, quantized and then flattened.
- Multi-precision flat spaces.
- $(\hat{x}, \hat{u})$ is an element of the 2D flat space $\widehat{X} \times \widehat{U}$.
- 2D task scheduling problem.
- Devices are tuned with sample slices of $\widehat{X} \times \widehat{U}$.
- Each $(\hat{x}, \hat{u})$ is assigned to a processing element (PE).



Parallel Computation of Symbolic Models

- Each PE computes the over approximation of reachable sets (OARS) initiated from one/more $(\hat{x}, \hat{u})$.
- OARS is stored into the memory without discretization.
- Abstraction memory size is fixed w.r.t number of transitions.
- Abstraction memory is distributed among devices.
- CPUs perform better due to Vectorization and Pipelining.



Parallel Construction of Symbolic Controllers

- Currently, a Fixed-point (FP) implementation is considered.
- Efficient on-the-fly memory-less discretization of OARS is used.
- Distributed bit-based storage of results reduces controller's size.
- Parallel convergence check is applied after some FP iterations.

Initial Results and Future Work

- OARS in a lock-free fast-to-query data structure.
- Combining sparsity and parallel implementations.
- Testing on FPGAs and the Cloud.
- Python wrappers or domain specific language

Sparsity Benchmarks

Ex. / Trans.	SCOTSv0.2 Abs.	Sparsity-Aware Abs.	SCOTSv0.2 Synth.	Decomposed Synth.
Vehicle/ 4M	84.1	4.82	43.1	29.8

Table: Results on 2013 Macbook Pro with 2.4GHz Intel Core i7 and 8GB RAM. Time in sec.

Parallel Benchmarks

Ex. / Trans.	SCOTS	SCOTS v0.2	Parallel
DCDC / 1M	30	2	0.037
Vehicle/ 4M	739	203	0.96
Unicycle / 105M	5898	2797	8.3

Table: Results with NVIDIA P5000 GPU. Time in sec and includes abstraction and synthesis. SCOTS and SCOTS v0.2 use FP and run on Intel Xeon E5-2630.