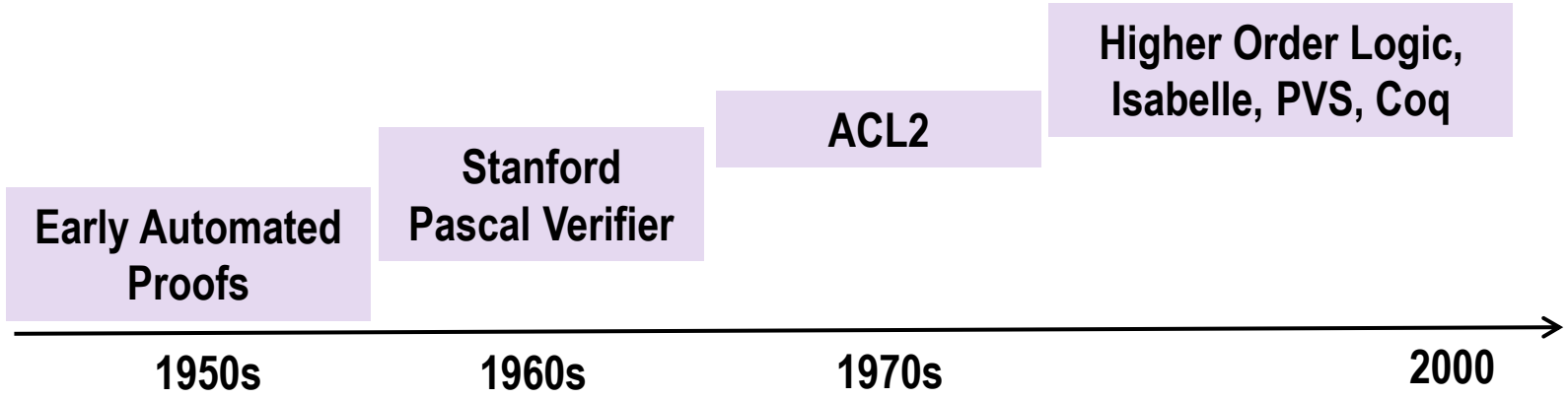
“Formal for Arithmetic” and “Arithmetic for Formal”

Dr. Pallab Dasgupta
Head of Research and Innovation, Formal Verification
ARITH 2023

The genesis of Formal for Arithmetic Circuits

For many decades, the focus was on proving software, algorithms, axiomatic theories – using theorem provers



And then in 1994, the focus shifted to arithmetic circuits …


Business Day

The New York Times

Digest

Flaw Undermines Accuracy of Pentium Chips

Four Sharp Corrections in the Dow

Gibson Suit On Trades Is Settled

Reasoning about Arithmetic – Fundamental Limitations

Gödel's incompleteness theorems:

1. ***No consistent system of axioms whose theorems can be listed by an effective procedure is capable of proving all truths about the arithmetic of natural numbers. For any such consistent formal system, there will always be statements about natural numbers that are true, but that are unprovable within the system.***

2. ***The system cannot demonstrate its own consistency.***

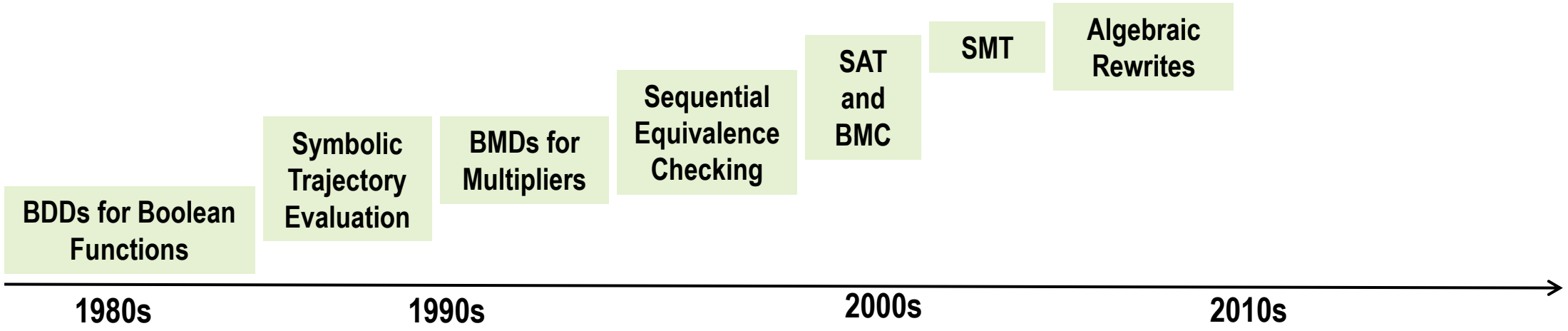
What can we do?

- ***Focus on soundness, sacrificing completeness*** **–**

  what we prove is true, but we cannot prove everything that is true

- ***Use soundness preserving abstractions*** **– *essentially attempt to prove stronger results***

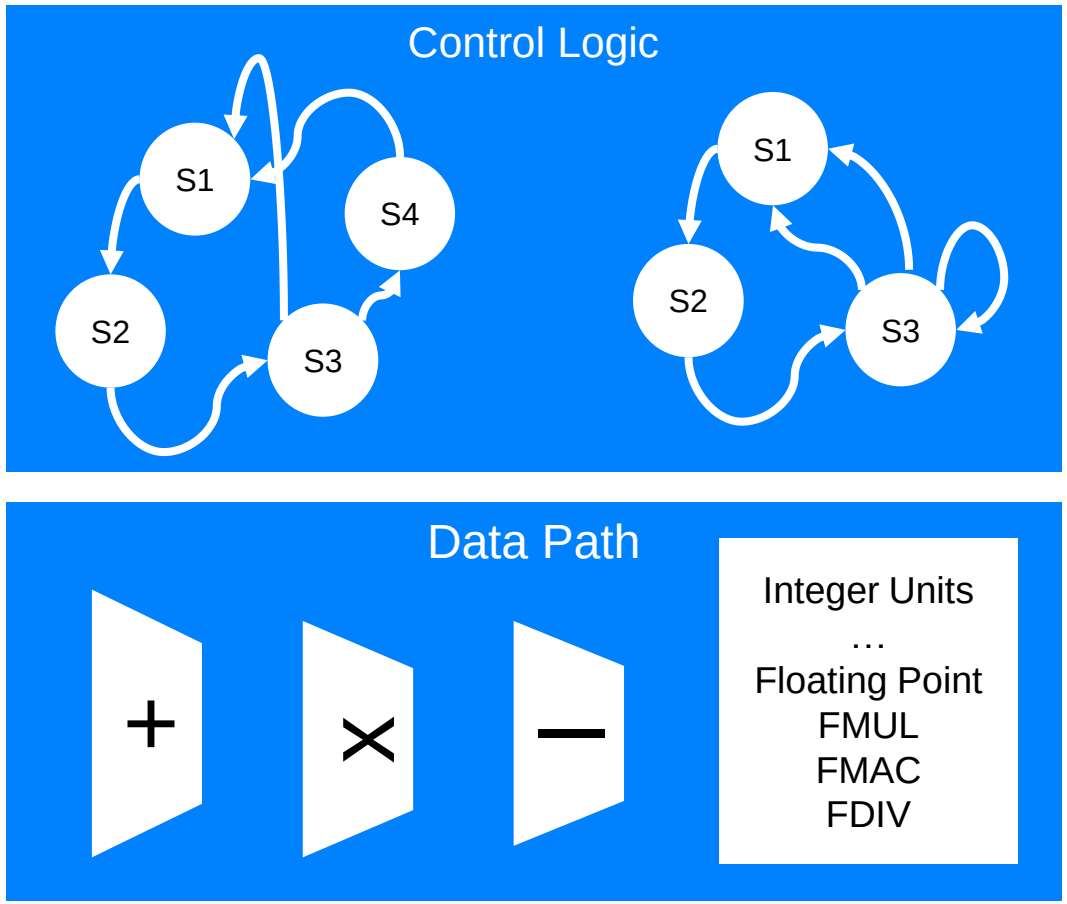
Evolution of Formal for Arithmetic



- Formal approaches to check equivalence between a known function and it's hardware / software implementation
  - Reduction to canonical forms for combinational functions
  - SAT/SMT-based equivalence checking
  - Sequential equivalence / Transactional equivalence
  - Algebraic rewrites

Datapath Validation Requires Specialized Formal Technology

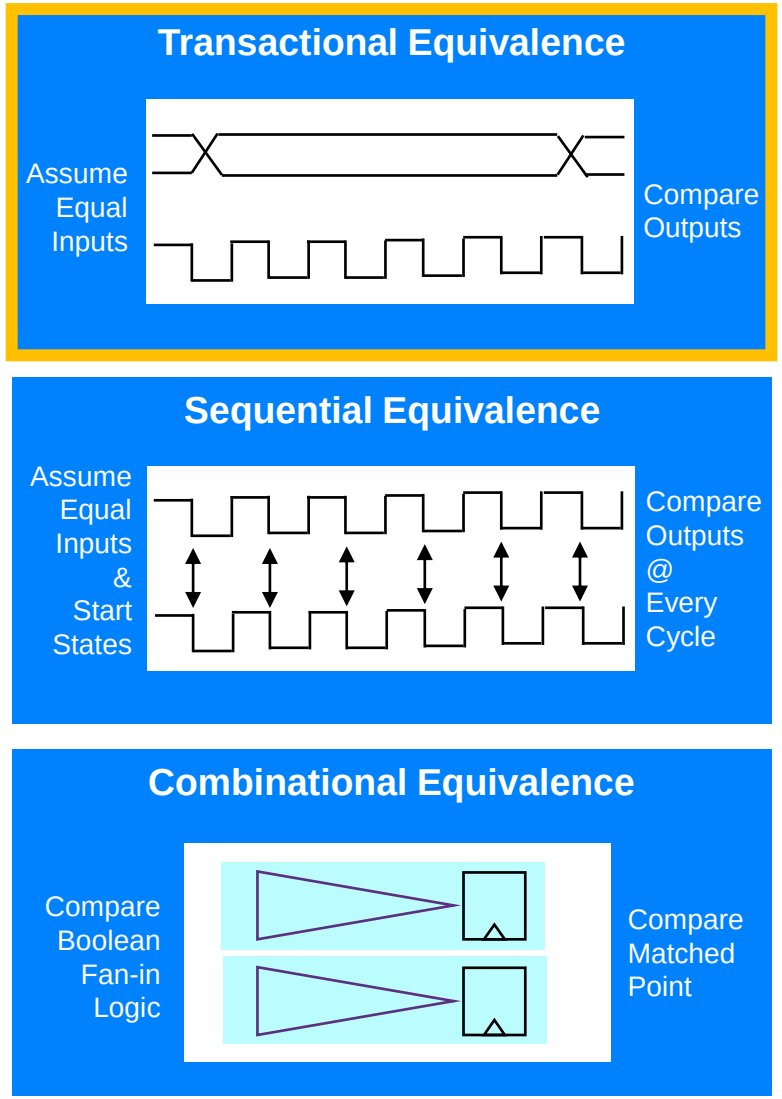
- **Formal property verification (FPV) is best suited for control logic verification**
  - **Proving that specified properties hold true for the DUT given input constraints**
- **FPV is not efficient in verifying datapath**
  - **Wide datapath operations lead to enormous formal search space and complexity**
  - **Bounded proof is not good enough for precise math function requirements**
- **Formal equivalence checking is needed to ensure datapath implementation correctness**



Formal Property Verification is not the best strategy for verifying datapath operations

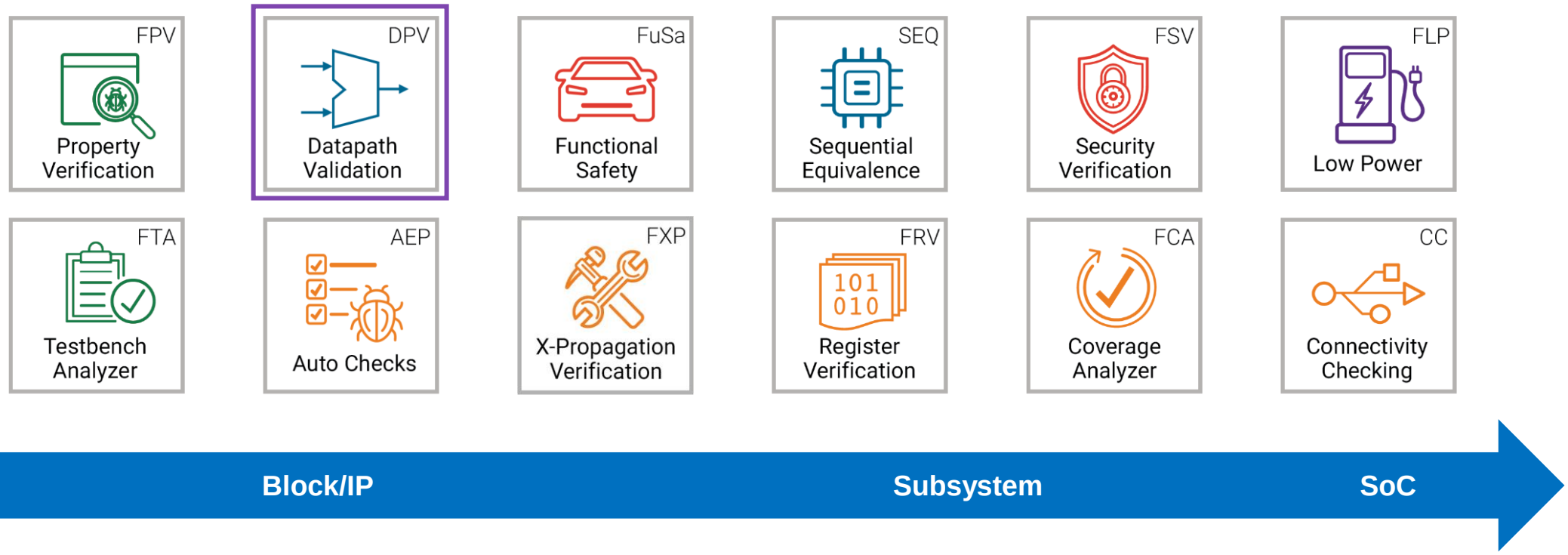
Datapath Validation Requires Transactional Equivalence Checking

- Datapath blocks often have C/C++ specifications
  - There is no notion of timing in the C/C++ models
- Transactional Equivalence means that RTL and the C/C++ model produce the same outputs, given equal inputs
- Transactional Equivalence is different from:
  - Sequential equivalence
    - Often used for RTL/RTL comparison after clock gating/retiming
  - Combinational equivalence
    - Often used for RTL/gate comparison after logic synthesis



VC Formal DPV: Synopsys Datapath Validation Solution

Shared Compilation, Formal Engines, Debug Interface with Other VC Formal Apps



High Performance: ML powered proprietary engines for hard proofs, liveness, and deep bug-hunting

High Confidence Formal Signoff: Native Certitude integration for fast and high-quality Formal Signoff

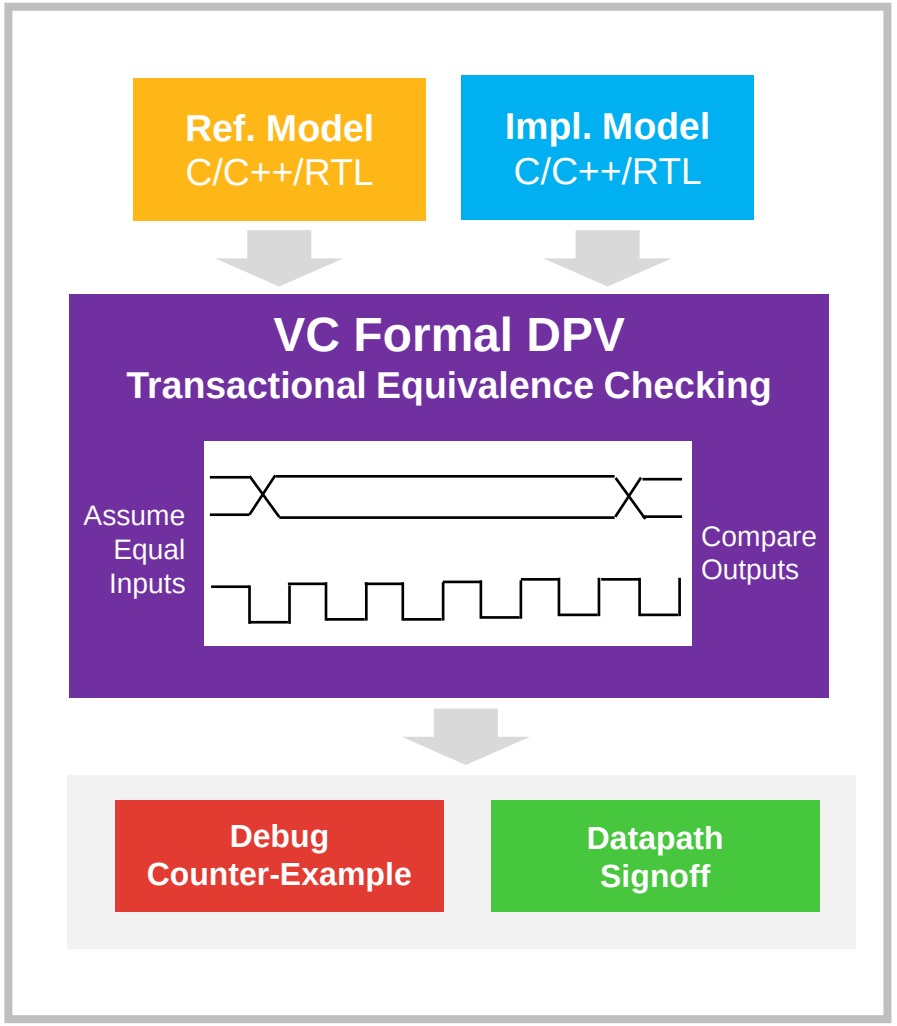
VC Formal DPV: Benefits & Features

DPV BENEFITS

- Exhaustively verify datapath design refinements
- Prove consistency of independently developed reference & implementation models
- Achieve datapath signoff without any testbench

DPV FEATURES

- Integrated mature HECTOR technology
- Supports ADD, SUB, MULT, DIV, SQRT operators
- Applicable to CPU, GPU, DSP, AI/ML (CNN) and other data processing designs

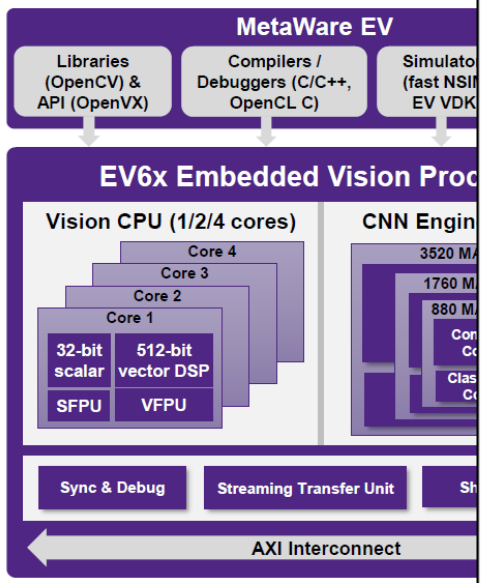


VC Formal DPV Application: Verification of AI/ML Designs

DesignWare ARC EV6x Example

DesignWare® EV6x Embedded Vision Processor IP

Scalable Hardware-Software Solu



EV6x Vector Floating Point Unit

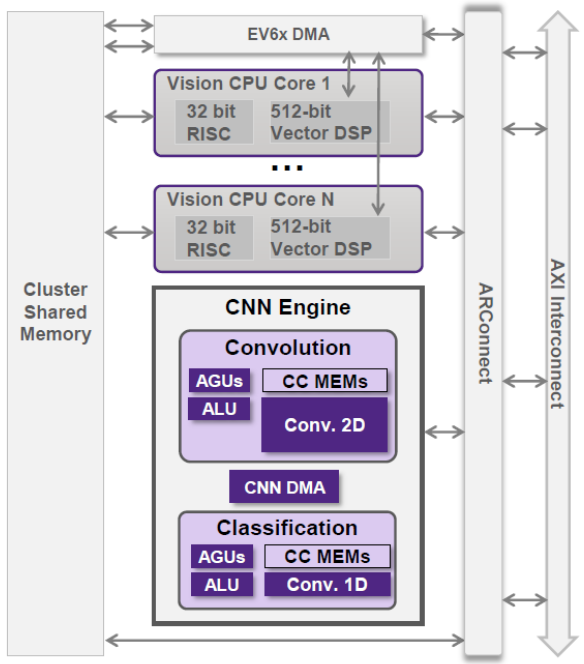
Optional extension

- High performance
  - Up to 512 GFLOP
- IEEE-754 compliar
- Support both 32-bi
- ISA: multiply-add/s
- Math functions: div

Verified with

Third Generation EV6x CNN Engine

Verified with VC Formal DPV



- Dedicated EV6x CNN Engine
  - Builds on experience of two previous generations
- Fully programmable to support full range of CNN graphs
- State-of-the-art power-efficiency >2000 GMAC/s/W
- Scalable to 4.5 TMAC/s (16 nm FFC, typical)
- Supports 8 bit and 12 bit data precision
- Real-time, high quality image classification, object recognition, detection, semantic segmentation
- Operates in parallel with Vision CPUs increasing efficiency and throughput

synopsys

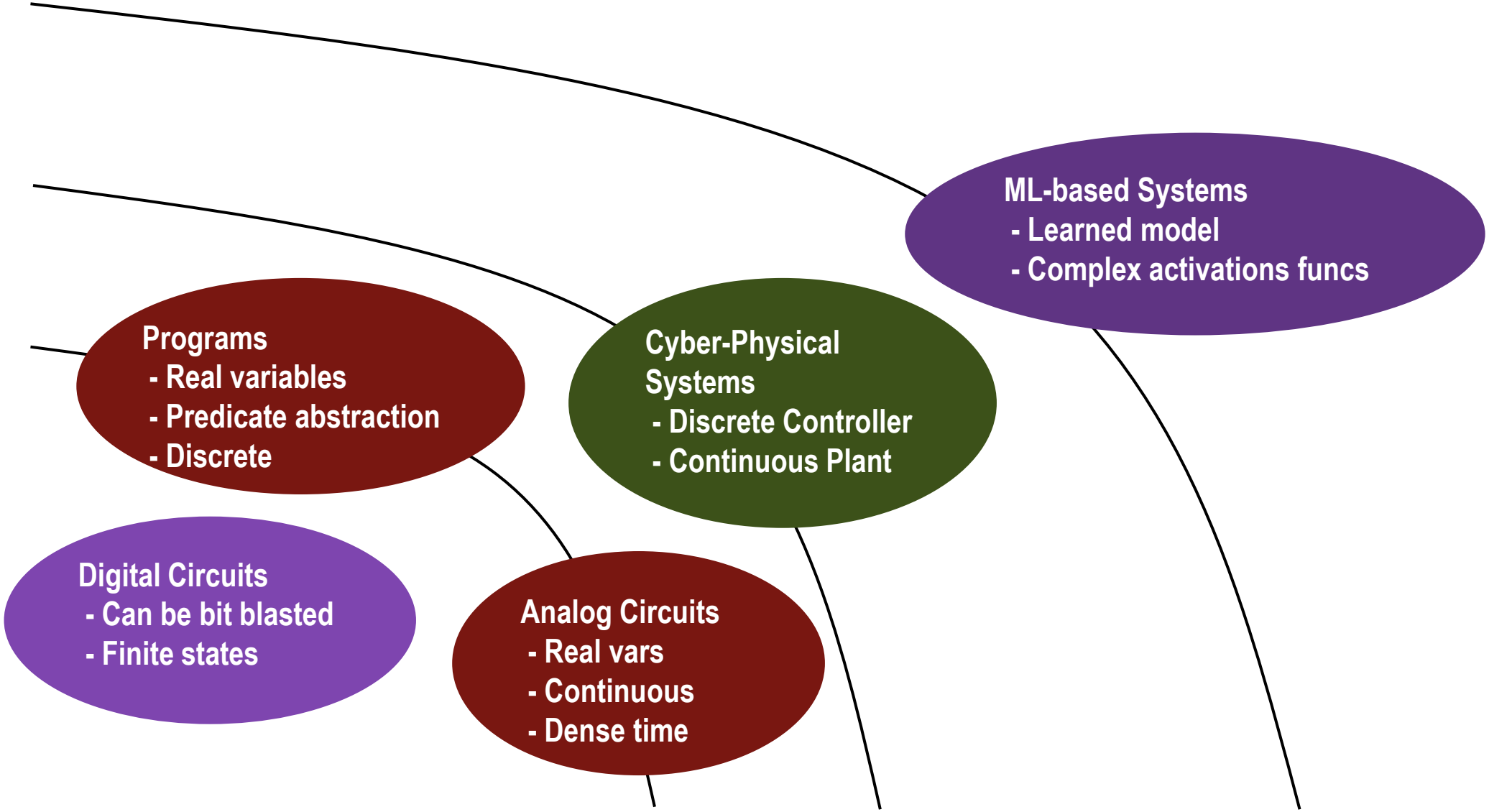
Synopsys Confidential Information

 13

The Future of Arithmetic and Formal

- Most industrial safety standards recommend the use of formal verification for safety critical designs
  - Industrial process automation (IEC 61508), Automotive (ISO 26262), Railway (EN 50128), Avionics (DO-178C/ DO-133), Nuclear (IEC 60880), Space (ECSS-Q-ST-80C)
  - These systems are of diverse types and of diverse complexities
- We wish to expand the universe of formal verification and its deployment
- Towards that direction
  - We need more complex arithmetic to model systems and “formal for such arithmetic”
  - The methods for formal verification must look beyond SAT/SMT to prove correctness – essentially a new “arithmetic for formal”
  - In reality, these two directions complement each other, and the challenges overlap

Complexity of Arithmetic for Formal Systems



Challenge 1: Handling continuous time

- Existing EDA tools on formal verification use discrete time. This allows us to:
  - Define the semantics of *next time* and count time in terms of number of cycles
  - For example, the property: always (@posedge clk) a |-> ##[3:5} b requires b to hold 3 to 5 cycles after a.
  - We can model behaviors in terms of discrete state sequences (runs of Büchi automata)

- On the other hand, assertions over dense time allow:
  - Assertions to hold continuously over an interval of time
  - Failures can also happen over intervals of time
  - There is no notion of *next* time

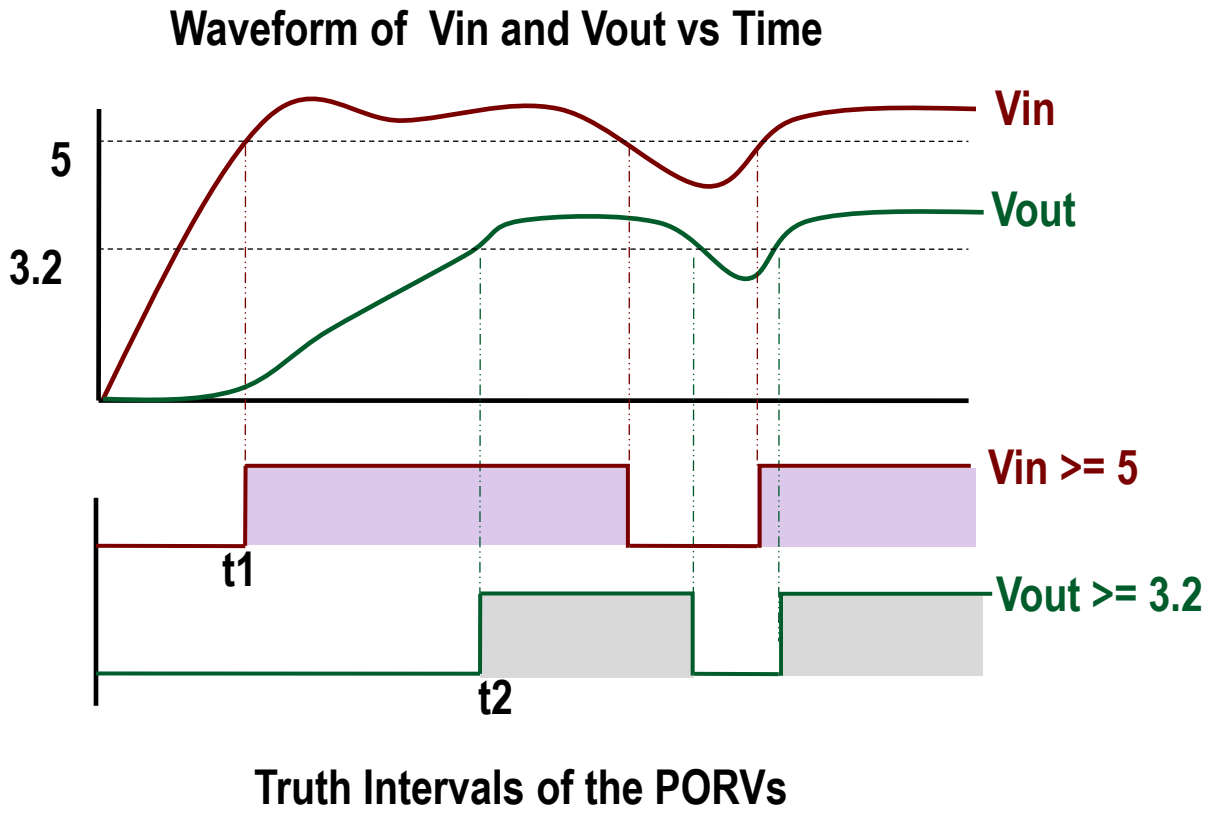
- We need a different type of arithmetic to reason about dense time

AMS Assertions

- Motivated by increasing analog content in AMS SOCs
- Key attributes of AMS assertions
  - Dense real time
  - Predicates over Real Valued Variables (PORVs)

(Vin >= 5) |-> ##[2.3 : 8.6] (Vout >= 3.2)

- (Vin >= 5) and (Vout >= 3.2) are two PORVs.
- PORVs are true over continuous intervals of time



- Instrumenting AMS Assertion Verification on Commercial Platforms, Rajdeep Mukhopadhyay, S K Panda, Pallab Dasgupta, John Gough, ACM Trans. on Design Automation of Electronic Systems, 14 (2), 2009.
- Feature Indented Assertions for Analog and Mixed-Signal Validation. Antara Ain, Antonio A Bruto Da Costa, Pallab Dasgupta, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 35(11), 2016.

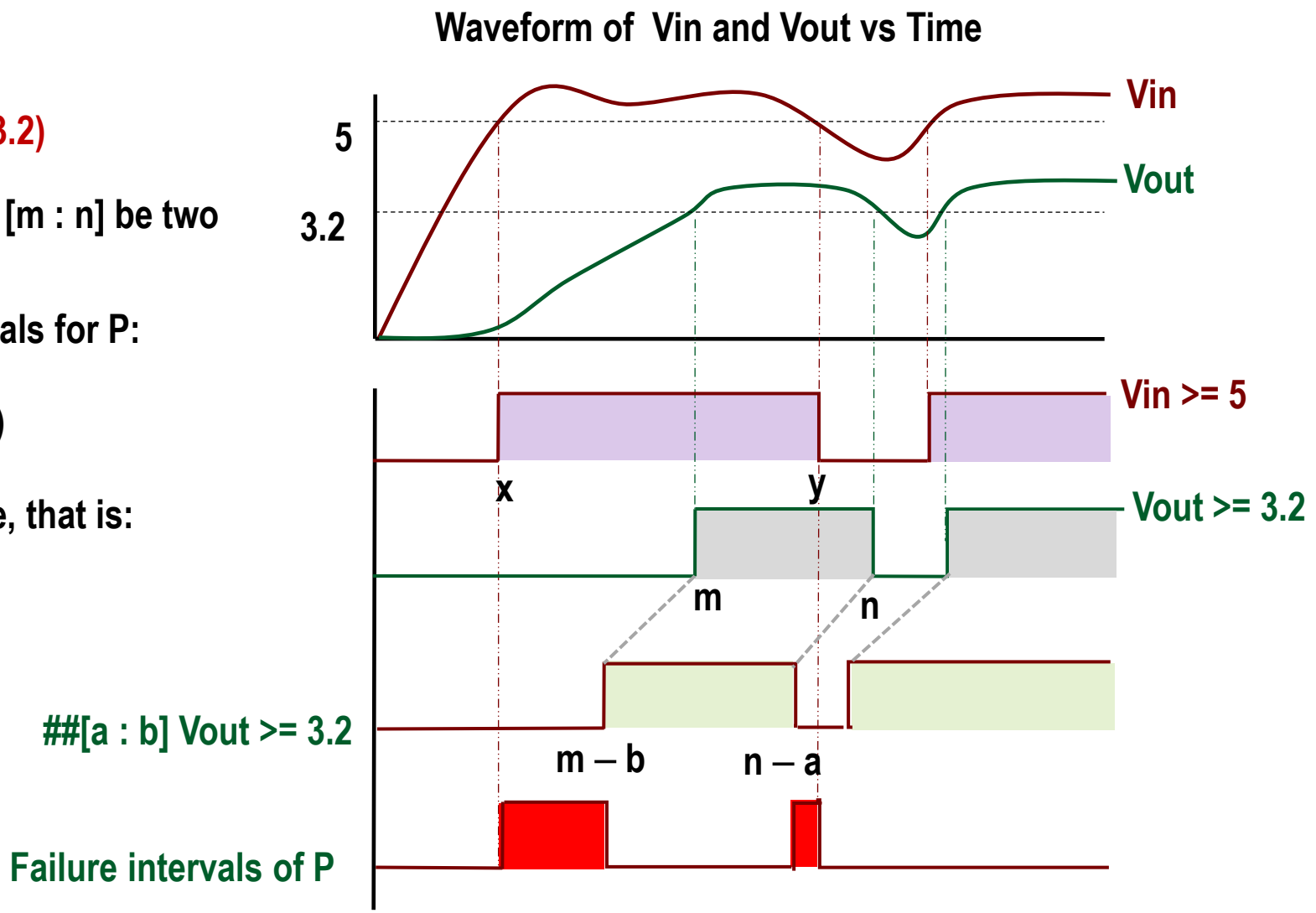
Interval Arithmetic

- Property P: (Vin >= 5) |-> ##[a : b] (Vout >= 3.2)
- Let T(Vin >= 5) = [x : y] and T(Vout >= 3.2) = [m : n] be two truth intervals
- Then the following is one of the truth intervals for P:

$$T(P) = [x : y] \cap ([m : n] \ominus [a : b])$$

where $\ominus$ denotes the Minkowski difference, that is:

$$[m : n] \ominus [a : b] = [m - b : n - a]$$



AMS Assertions with Local Variables

- **Local variables significantly add to the expressive power of SystemVerilog Assertions. Very powerful in AMS-SVA as well**

- **If Vin rises from 1V to 5V in less than 2.5s then Vout must rise by at least 3.6V.**

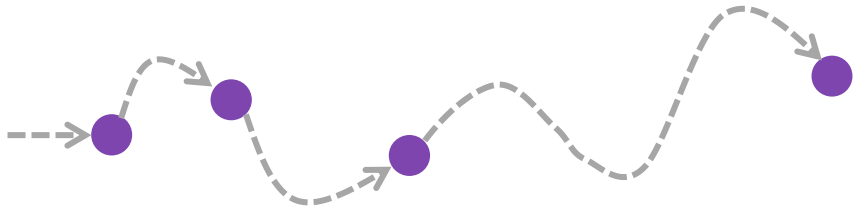
  @+(Vin>=1), v1=Vout ##[0:2.5] @+(Vin>=5),v2=Vout |-> (v2 - v1 > 3.6)

- **In AMS-SVA, the local variables are unbounded reals**
  - **This is very powerful – can model recursive arithmetic computations**
  - **Satisfiability of AMS-SVA is computationally undecidable (recall Godel's Theorem)**
    - **Reduction from 2-counter machines**

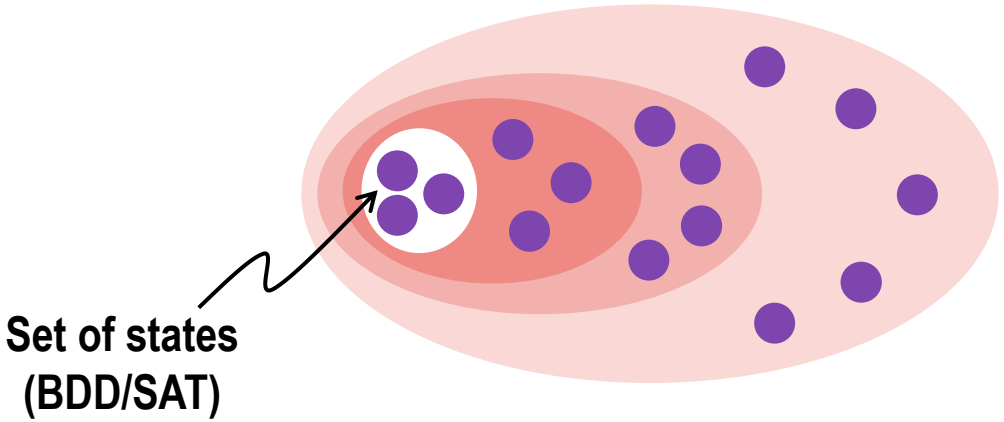
- Synchronizing AMS Assertions with AMS Simulation: From Theory to Practice, Subhankar Mukherjee, Pallab Dasgupta, Siddhartha Mukhopadhyay, Scott Little, John Havlicek, Srikanth Chandrasekaran, ACM Trans. on Design Automation of Electronic Systems, 17(4), 2012.
- Interpreting Local Variables in AMS Assertions during Simulation, Antara Ain, Pallab Dasgupta, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 38(5), 2019.

States → Discrete State Sets → Regions

A simulation run: *State Traversal*

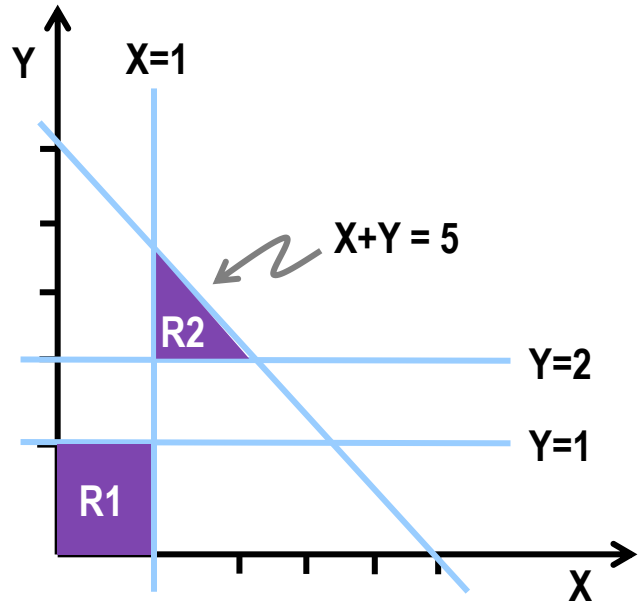


Symbolic traversal: *Traversal through Discrete State Sets*



Challenge 2: Symbolic Reachability in Dense Spaces

- Each state set is uncountable – cannot be represented by SAT / BDD
- Hence the notion of *regions*



State sets are defined by conjunction of constraints:

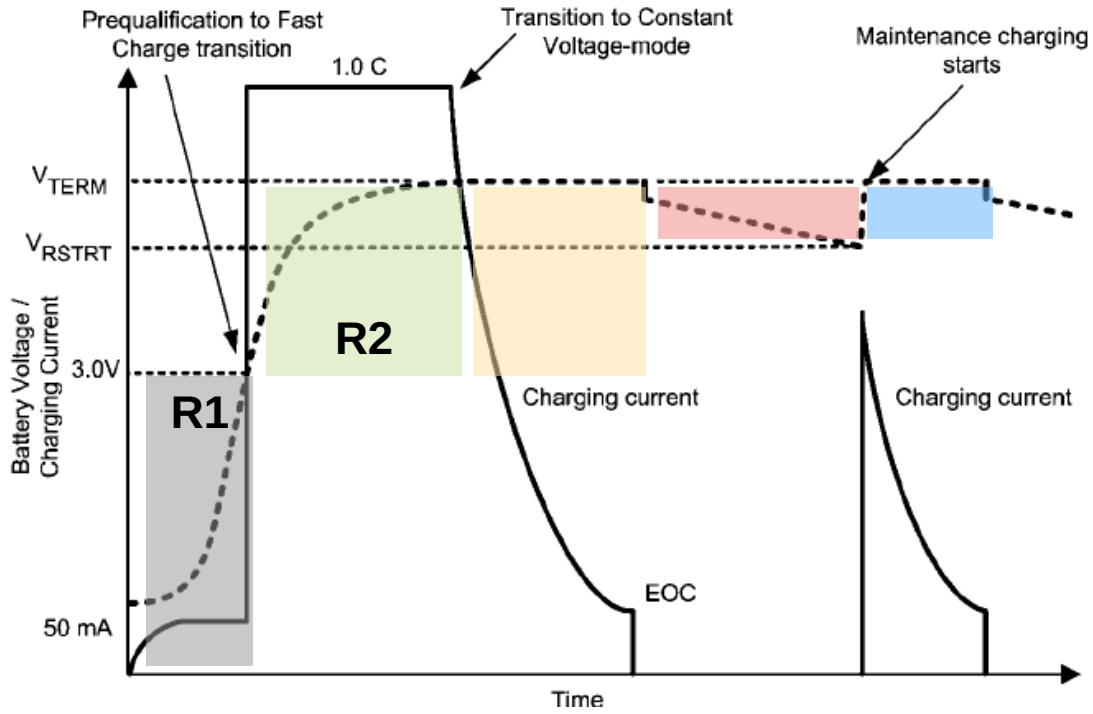
- Region R1 = (0 < X < 1) && (0 < Y < 1)
- Region R2 = (X > 1) && (Y > 2) && (X+Y < 5)

One may conceive a region graph, where each vertex is a region

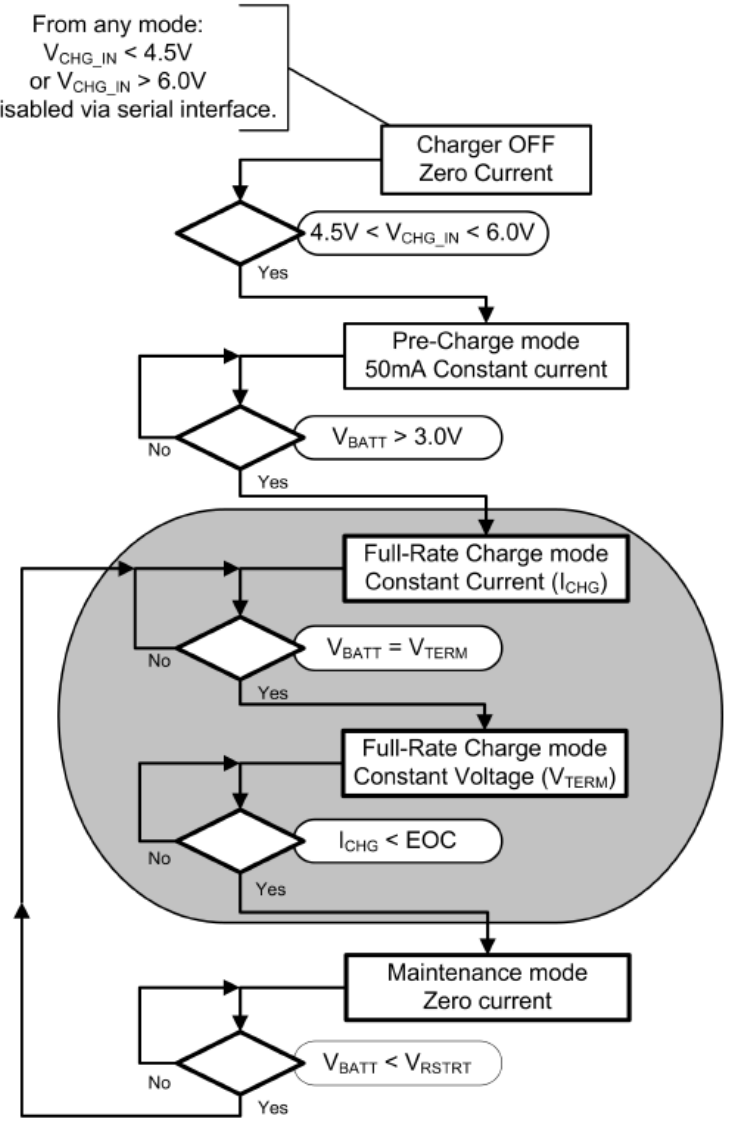
- Edges between adjacent regions
- Here R1 and R2 are not adjacent regions, hence no edge between them

Regions in Analog / Mixed-Signal

- **Regions boundaries are predicates specified using SV real net types in digital AMS models, or VAMS wreal in VAMS models / spice circuits**

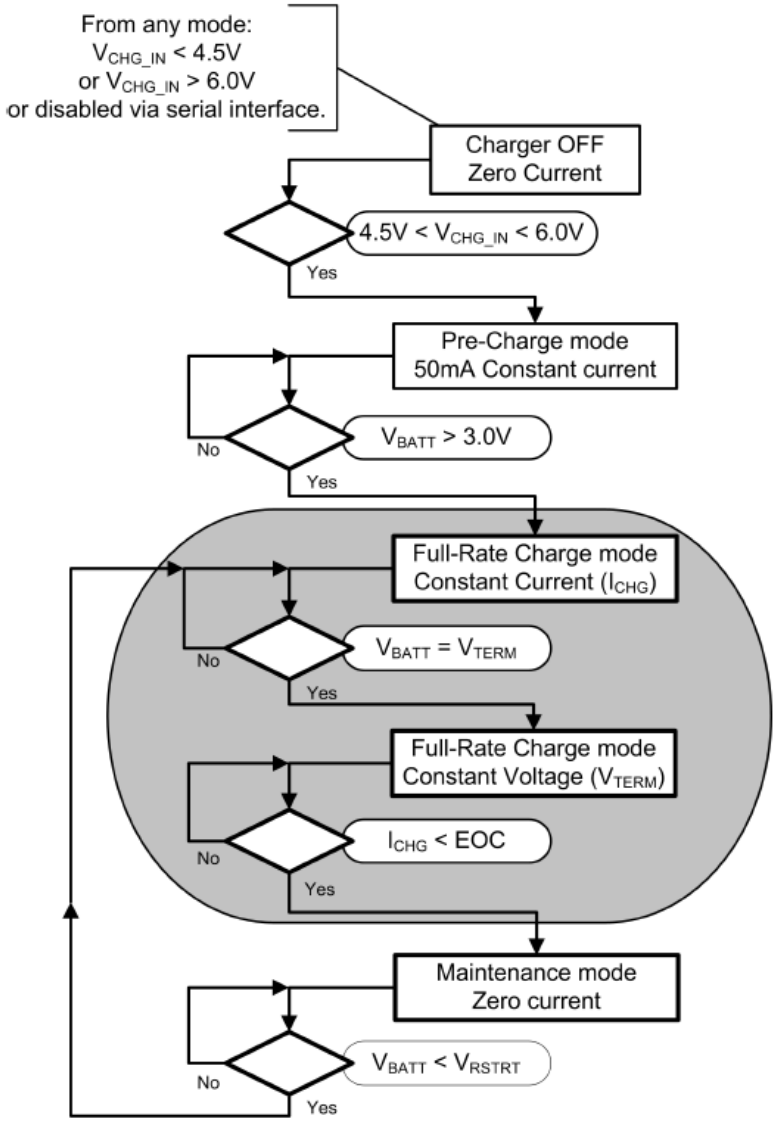


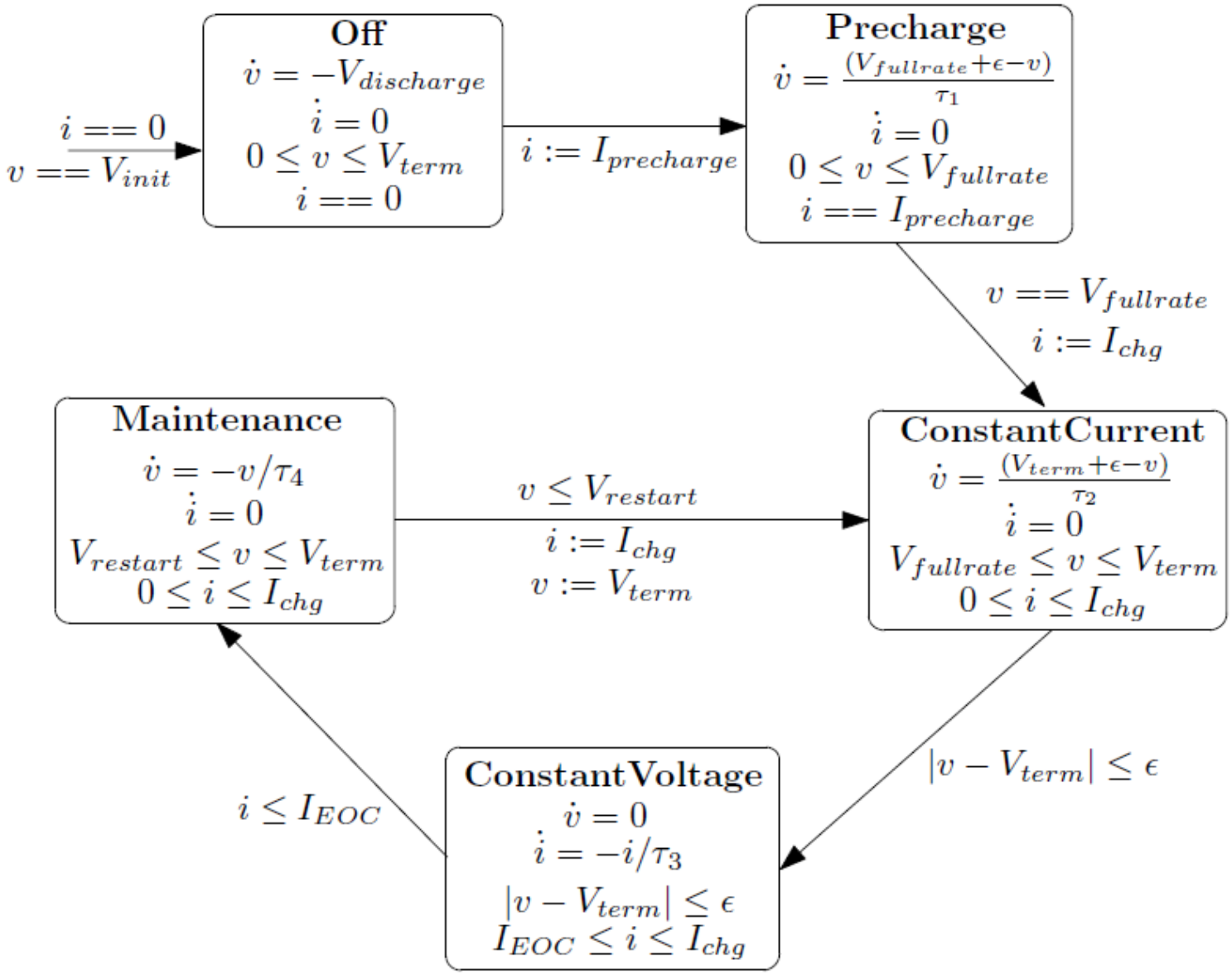
Charging Characteristics (Li-ion battery charger)



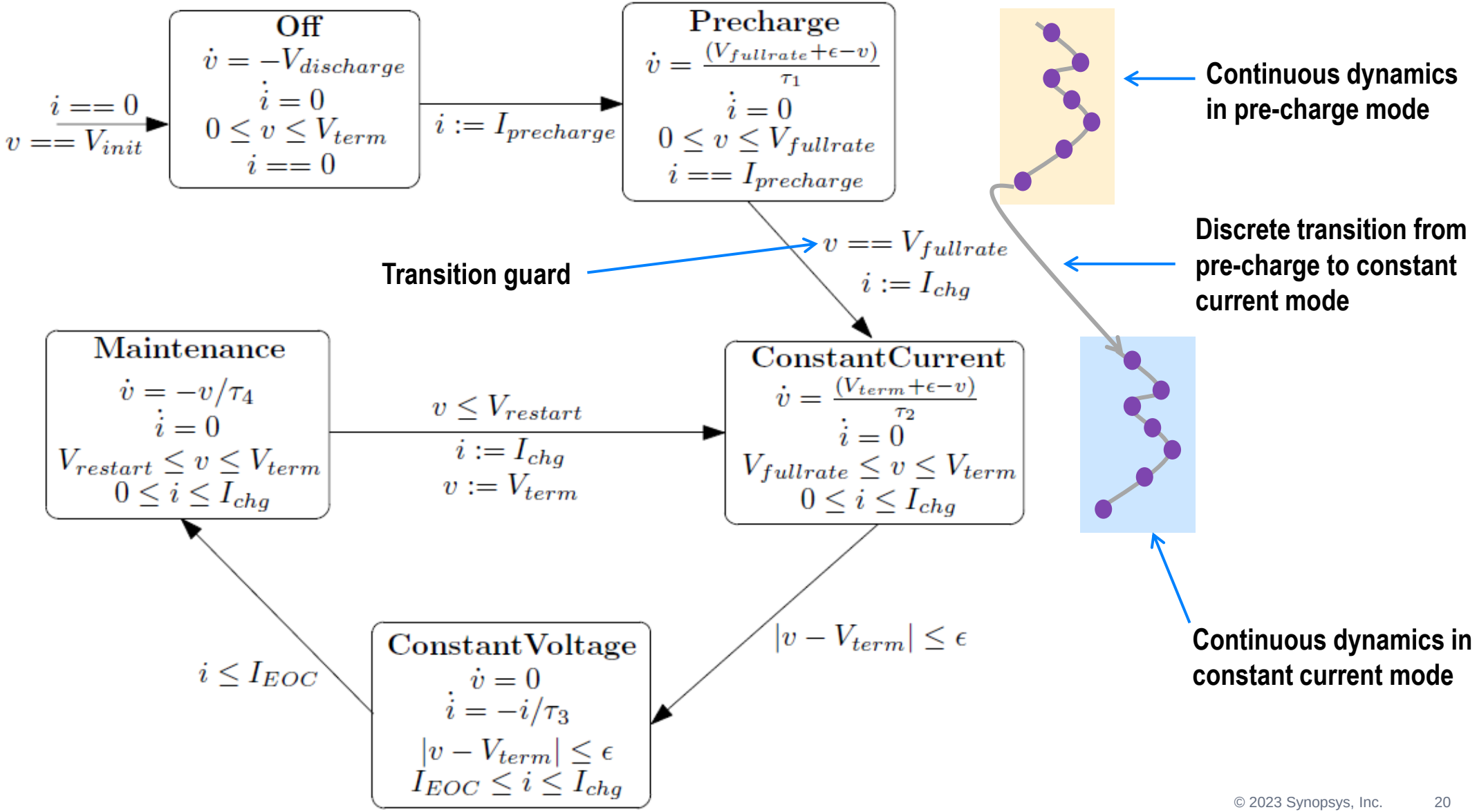
Modes: Pre-charge, Constant Current, Constant Voltage, Maintenance
Mode boundaries are defined by predicates

The notion of hybrid automata



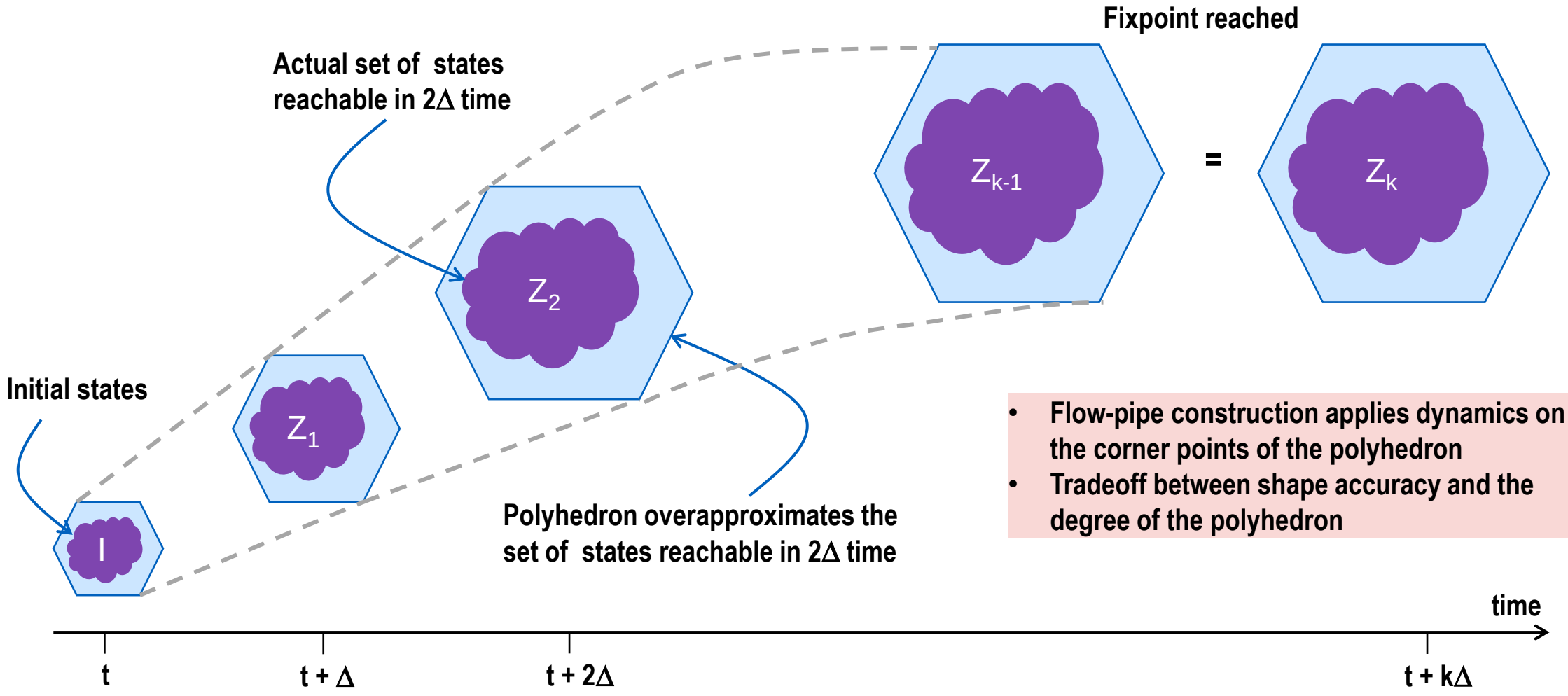


The notion of hybrid automata



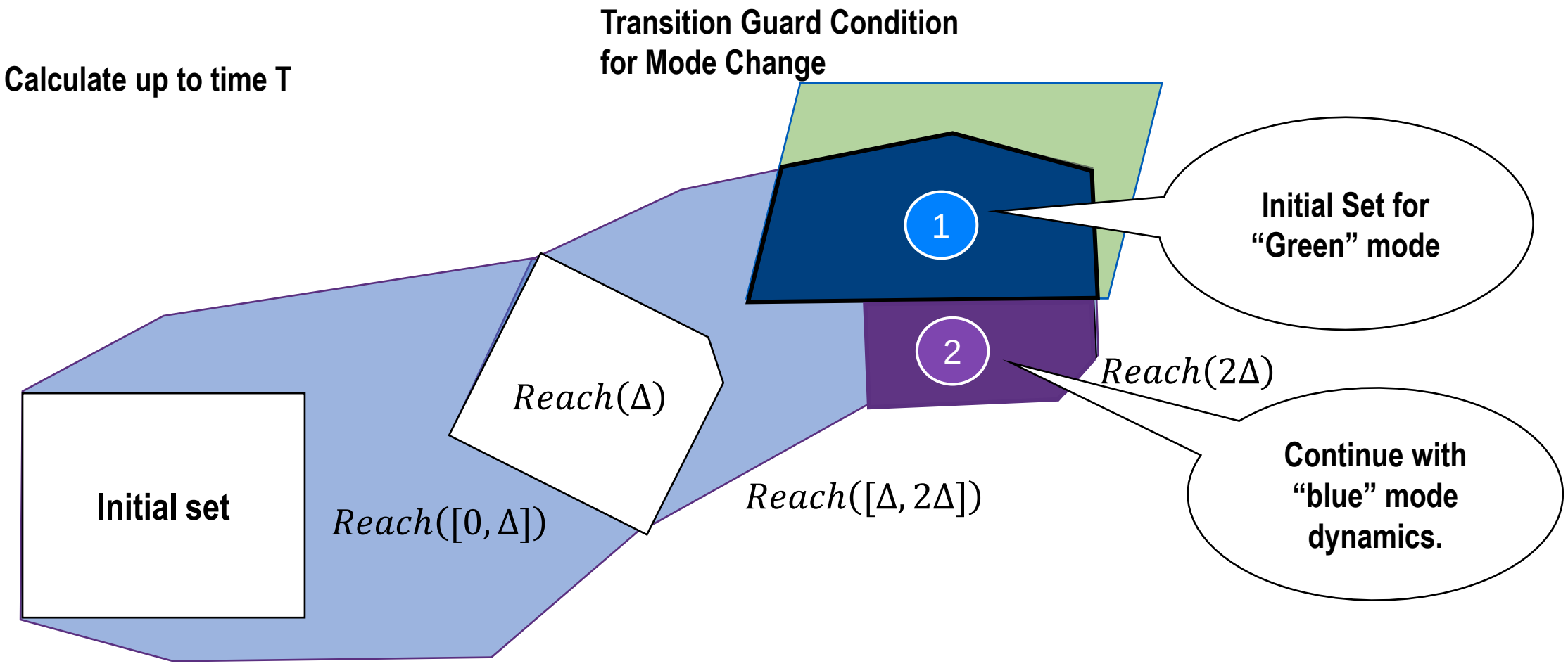
Reachability and Fixpoint Computations

- **Polyhedral approximations (convexity assumptions)**

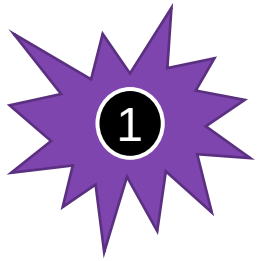


- **Flow-pipe construction applies dynamics on the corner points of the polyhedron**
- **Tradeoff between shape accuracy and the degree of the polyhedron**

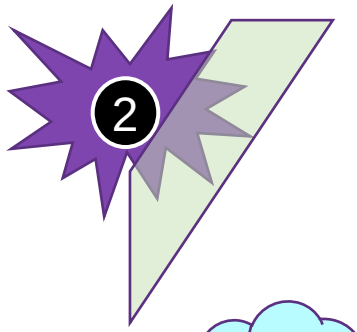
Flow-pipes for Hybrid Systems



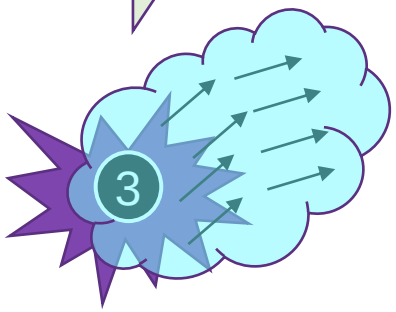
Flow-pipe Construction: Challenges



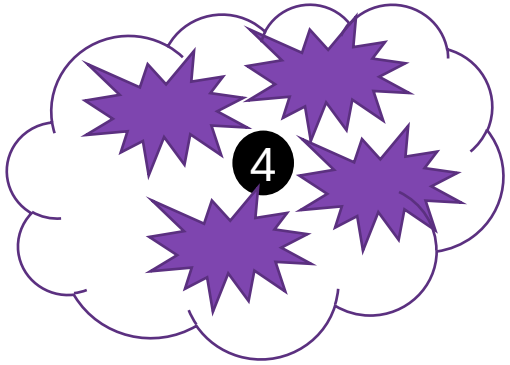
Representation: sets of states.



Intersection and Set Difference.



"Time Elapse": ODE Integration.



Union: aggregate multiple segments into one.

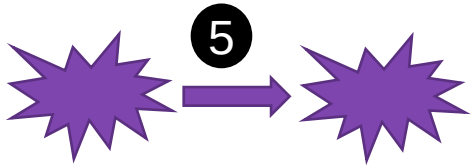


Image under nonlinear transformations.

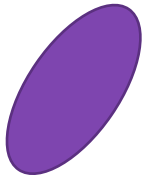
6

Projection onto a subset of dimensions
Complexity reduction.
Containment checking.

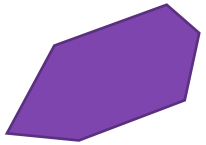
Set Representations

Intervals/
Hyper-rectangles

Zonotopes
[Girard et al'05]



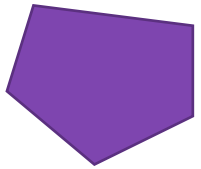
Ellipsoids
[Kurzhanskiy+Varaiya]



Convex Polyhedra
[Sank. et al'09, Frehse et al.'06]

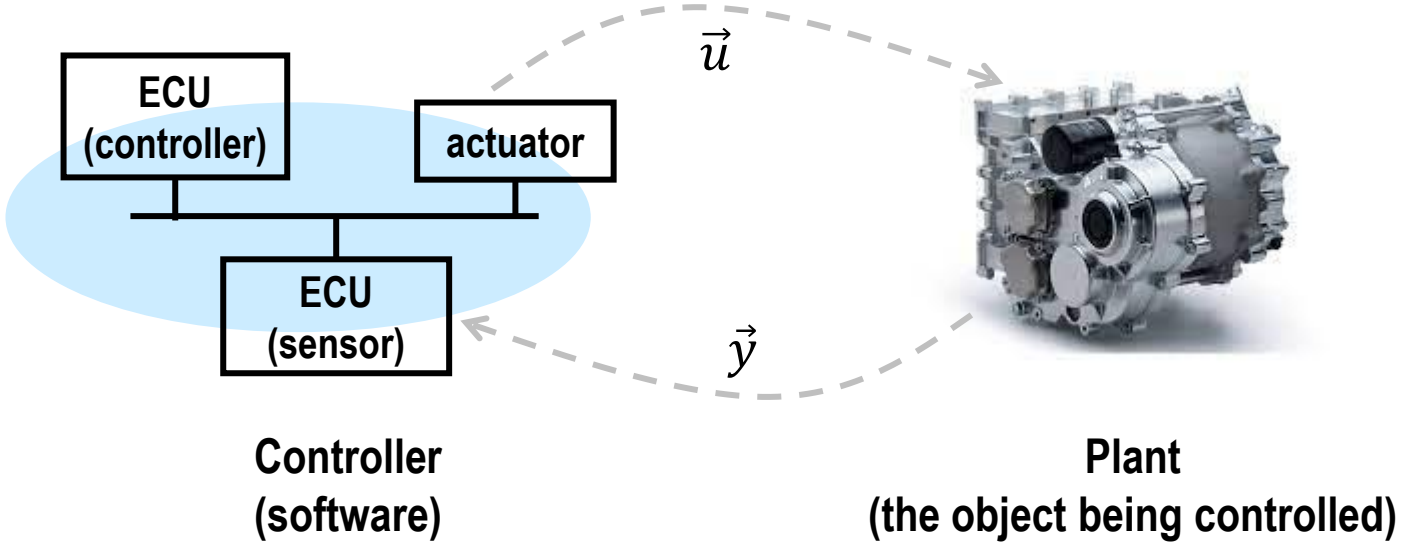
$\lambda c. \max c.x \text{ st.} x \in S$

Support Functions
[Frehse et al.'09]



Star Sets
[Bak et al'16]

Formal Modeling of Cyber-Physical Systems



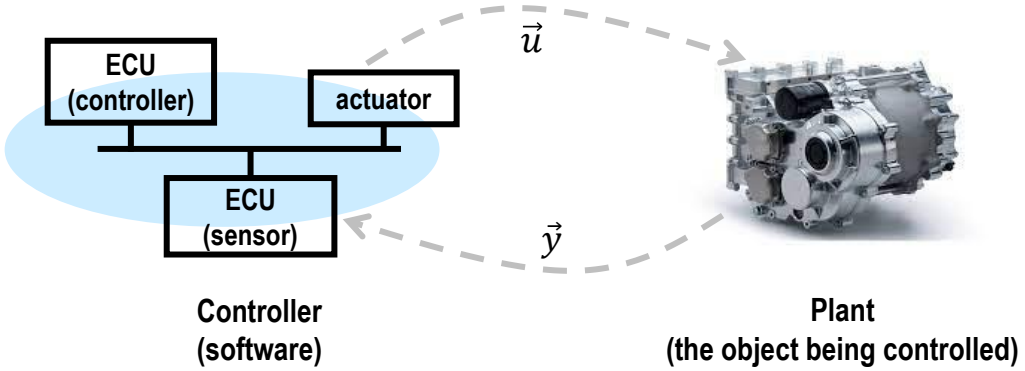
Design Flow:

Continuous/Hybrid dynamics → Discrete (sampled) dynamics → Discrete Control Law → Control software

Mathematical Representation of Dynamical Systems

Standard practice in control design:

Continuous/Hybrid dynamics → Discrete (sampled) dynamics



$$\dot{x} = Ax + Bu$$
$$y = Cx$$

select a sampling rate *h*

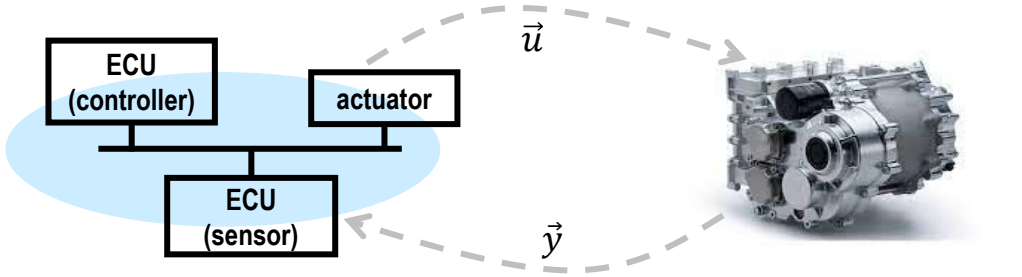
$$x[k+1] = \Phi x[k] + \Gamma u[k]$$
$$y[k] = Cx[k]$$

where

$$\Phi = e^{Ah} = I + Ah + \frac{A^2h^2}{2!} + \frac{A^3h^3}{3!} + \ \ldots$$

$$\Gamma = Cx[k] \int_0^h e^{As} B ds$$

Mathematical Representation of Dynamical Systems



Controller Dynamics:

$$x_c[t+1] = A_c x_c[t] + B_c y[t]$$
$$u[t] = C_c x_c[t]$$

Plant Dynamics:

$$x_p[t+1] = A_p x_p[t] + B_p u[t]$$
$$y[t] = C_p x_p[t]$$

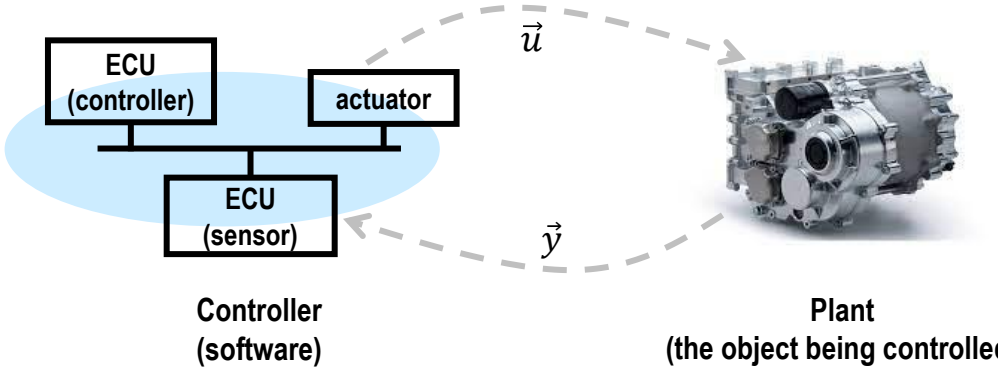
Closed Loop (Linear) Dynamics:

$$x[t+1] = \begin{bmatrix} A_p & B_p C_c \\ B_c C_p & A_c \end{bmatrix} x[t]$$

or, $x[t+1] = Ax[t]$

- **Most real-world systems are non-linear, and can be approximated by piecewise linear dynamics**

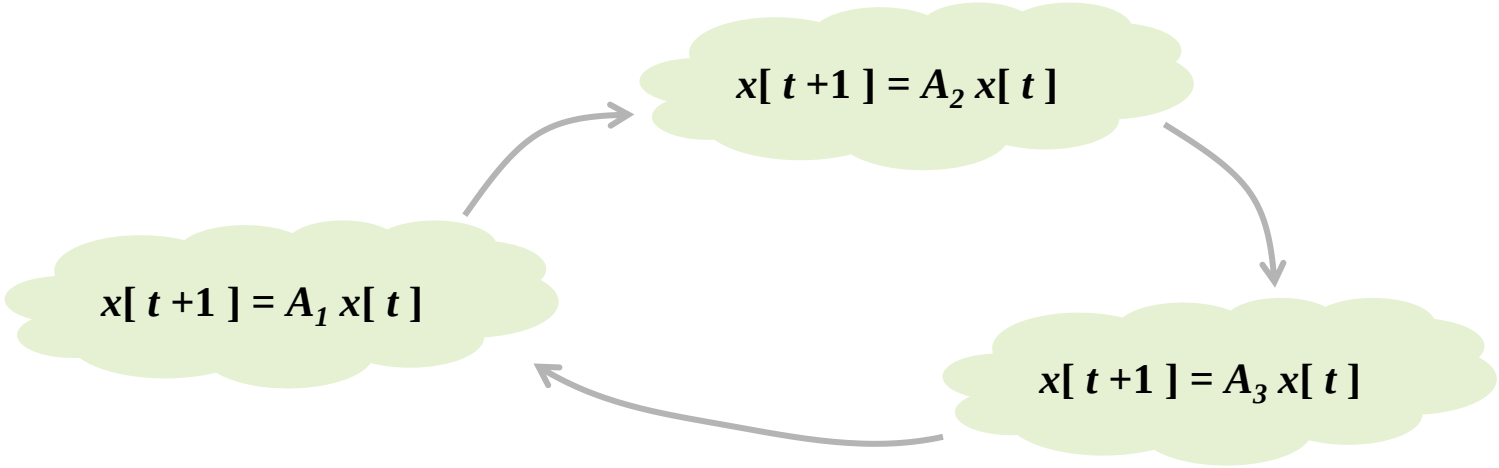
Mathematical Representation of Dynamical Systems



Closed Loop (Linear) Dynamics:

$$x[t+1] = Ax[t]$$

Most real world systems are non-linear. May be approximated by piecewise linear dynamics



Stability as a Safety Property

- Given a set of linear dynamics, A_1, …, A_n, the following represents a switched execution of k cycles:

$$x[t+k] = (A_{\sigma_k} \dots A_{\sigma_2} A_{\sigma_1}) x[t] \quad \text{where } \sigma_j \in \{1, \dots, n\}$$

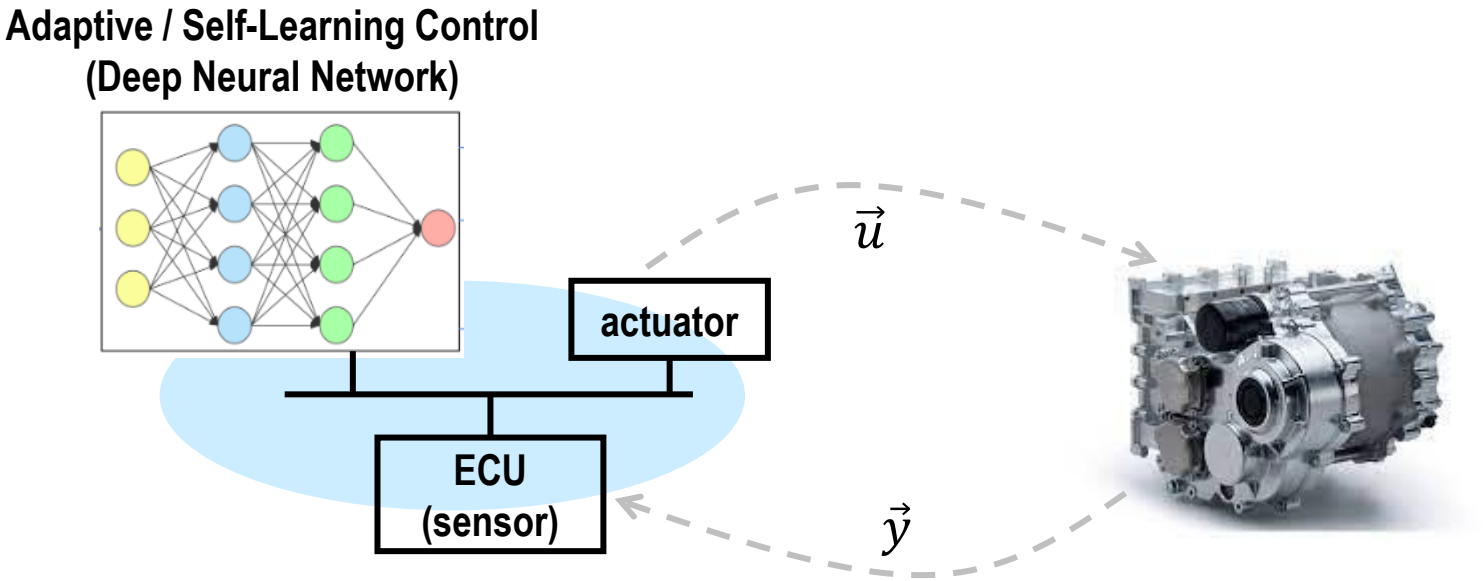
- What can we infer from k length sequences?

- Exponential Stability:
  - For $0 < \varepsilon < 1$ and $k \in N$, consider the language of all schedules such that any interval of length k is contracting by at least ε

    ExpStab(k, ε) = {$\sigma = \sigma_1 \dots \sigma_k$, such that $||A_{\sigma_k} \dots A_{\sigma_2} A_{\sigma_1}|| < \varepsilon$ for every $k \in N$}
  - The above language is omega regular and can be accepted by a Büchi automaton
  - Therefore we can formally prove stable execution patterns of switching in a controller

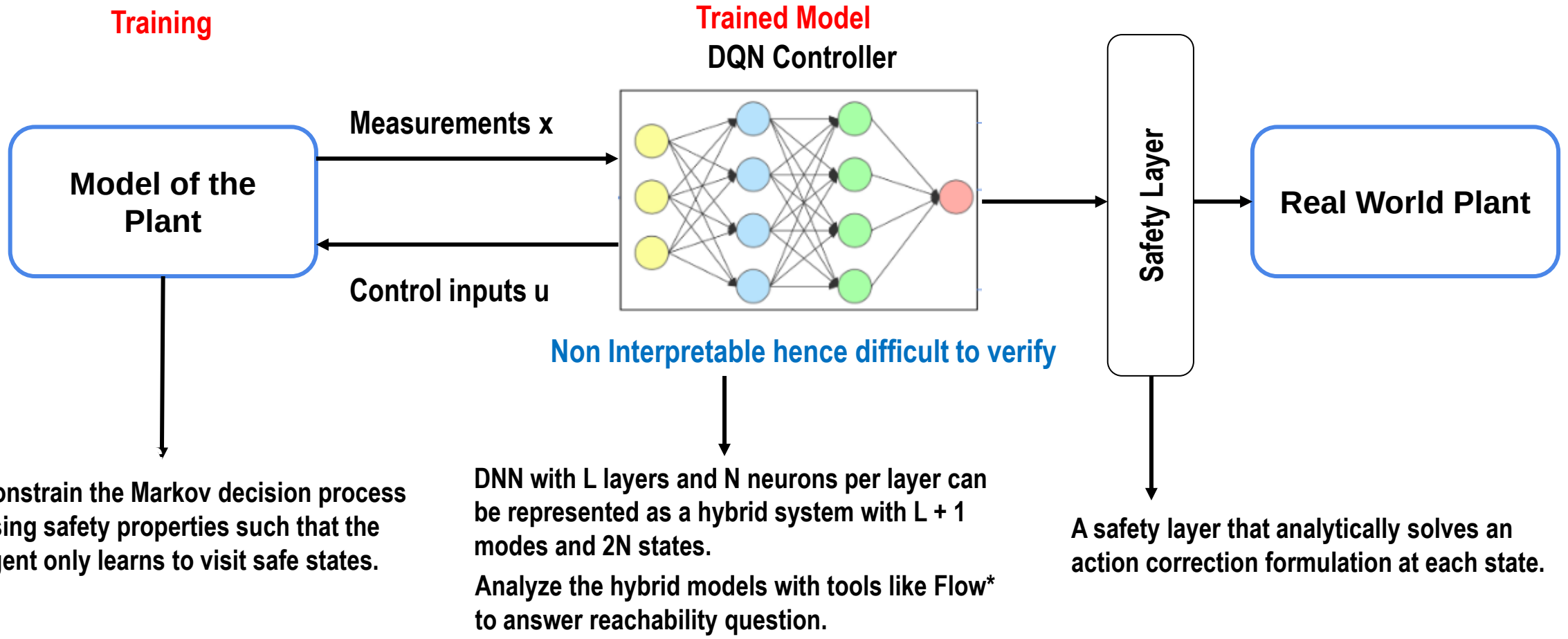
- Likewise we can formally prove control performance, directional growth, admissible loop skipping, etc.
- These guarantees have strong inductive underpinnings

Challenge 3: Intelligent Cyber-Physical Systems



- Neural networks model highly non-linear functions
- Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks
  - Only neural networks using the RELU activation function can be used (RELU is piecewise linear)
- Going forward, we need the ability to handle more complex activation functions – sigmoid, tanh, etc.

Building Trust in RL-based controllers

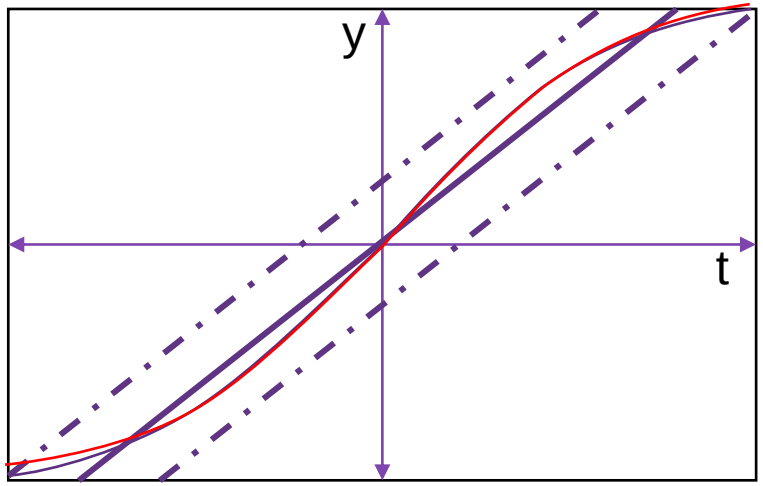


Inspired by CEGAR:

Counterexample Guided RL Policy Refinement Using Bayesian Optimization, Briti Gangopadhyay, Pallab Dasgupta, Advances in Neural Information Processing Systems (NeurIPS). 2021.

Towards Flow*: Taylor Models

$y = \sin(t), t \in [-1.2, 1.2]$



$y \in t + [-0.3, 0.3]$

$y \in t - \frac{t^3}{6} + [-0.03, 0.03]$

$y \in t - \frac{t^3}{6} + \frac{t^5}{120} + [-0.0007, 0.0007]$

$y = f(x_1, x_2, \ldots, x_n), \quad x_1, \ldots, x_n \in X_0$

$y \in p(x_1, \ldots, x_n) + [l, u]$

Polynomial of degree d

Error Interval

Slide courtesy: Prof. Sriram Sankaranarayanan, UC Boulder

Taylor Model Calculus

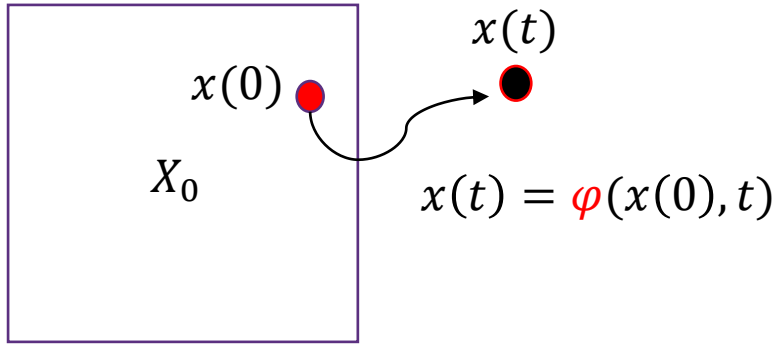
Taylor model approximation of functions

$f(x, y, t) = \left(x - 2\sin(x + y\,t^2) - 1.5\,e^{(0.5\,x^2 - t^2 - 0.2)}\right)$ ⟸ **Smooth function**

$x \in [-0.5, 0.5], y \in [-1.1, -0.8], z \in [0.2, 1.4]$ ⟸ **Bounded Domain**

$\Downarrow$

$f \in (0.9\,t^2 x^2\,y^3 + \cdots + 0.15) + [-0.05, 0.05]$ ⟸ **Taylor model approximation**



Idea: Taylor model for $\boldsymbol{\varphi(x(0), t) = p(x(0), t) + Intvl.}$

Slide courtesy: Prof. Sriram Sankaranarayanan, UC Boulder

The Flow* Tool

Work of Prof Erika Abraham, RWTH Aachen,
Prof Sriram Sankaranarayanan and Xin Chen, UC Boulder
http://www.flowstar.org

- **Case studies**
  - **Descent guidance program for a UAV autolander [Zhan et al.]**
  - **Verifying UAV controllers [Ethan Jackson et al. Microsoft Research]**
  - **Verification of automotive controllers [Xiaoqing Jin et al. Toyota Motors]**
  - **Analysis of medical device control algorithms [Dutta et al.'2017]**
  - **Data-driven control verification [Chen+Dutta+Sank.'2019]**
  - **Interval-based Bayesian Inference [Yi+Sank.'2019]**

Concluding Remarks

- Most industrial safety standards (automotive, avionics, railways, industrial automation, space, nuclear) recommend the use of formal verification for safety critical components
  - These come under various computational structures – models, software, hardware, English specifications

- The complexity of the arithmetic is growing, and we need to invest in new types of methodologies, looking beyond SAT/SMT

- We have a deep innovation pipeline in VC Formal, and we are continuously building new products and applications soon that we can help the industry solve the hard problems of the future

Together, we will ensure a safe and reliable world !!

Thanks for your attention !!