Synthesizing Self-Stabilizing Parameterized Protocols with Unbounded Variables

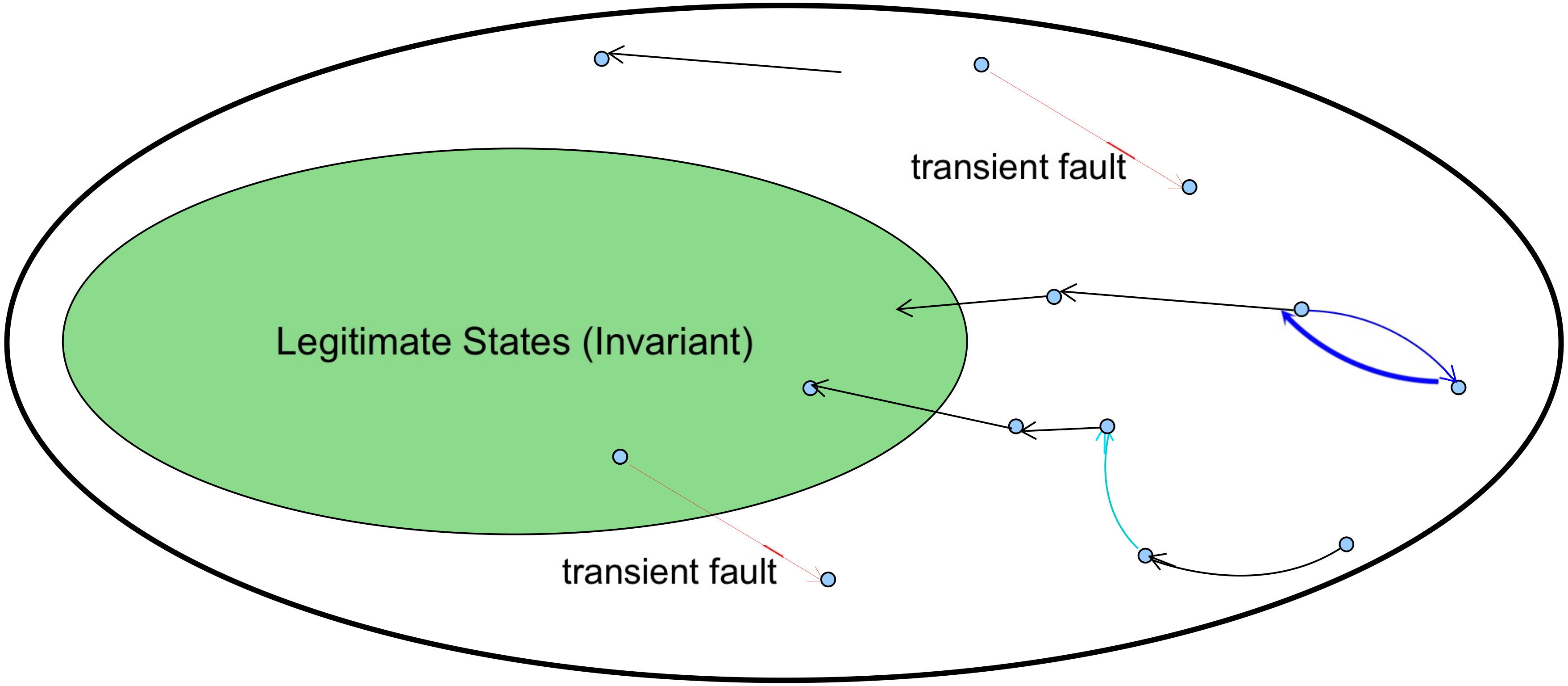
Ali Ebnenasir
aebnenas@mtu.edu

Department of Computer Science
College of Computing
Michigan Technological University
Houghton MI 49931

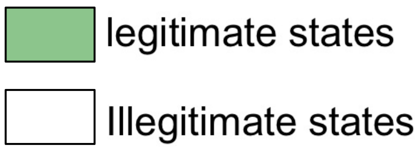
http://asd.cs.mtu.edu/

Self-Stabilization

"The ability of a distributed system to resume its legal behavior in a finite number of steps regardless of its initial configuration/state" [Dijkstra'74, Arora and Gouda'93]



Self-stabilization = closure + convergence



[1] E. W. Dijkstra, **Self-stabilizing systems in spite of distributed control**. *Communications of the ACM*, vol. 17, no. 11, pp. 643-644, 1974

[2] A. Arora and M. Gouda, **Closure and Convergence: A foundation of fault-tolerant computing**. *IEEE Transactions on Software Engineering*, vol 19, no. 11, pp. 1015-1027, 1993.

Modeling
Parameterized Distributed Protocols (PDP)

Family 2: just one process

Dijkstra's token passing:

π_2: Template process 2

Action$_0$: $x_0 = x_{N-1} \rightarrow x_0 := x_{N-1} + 1$

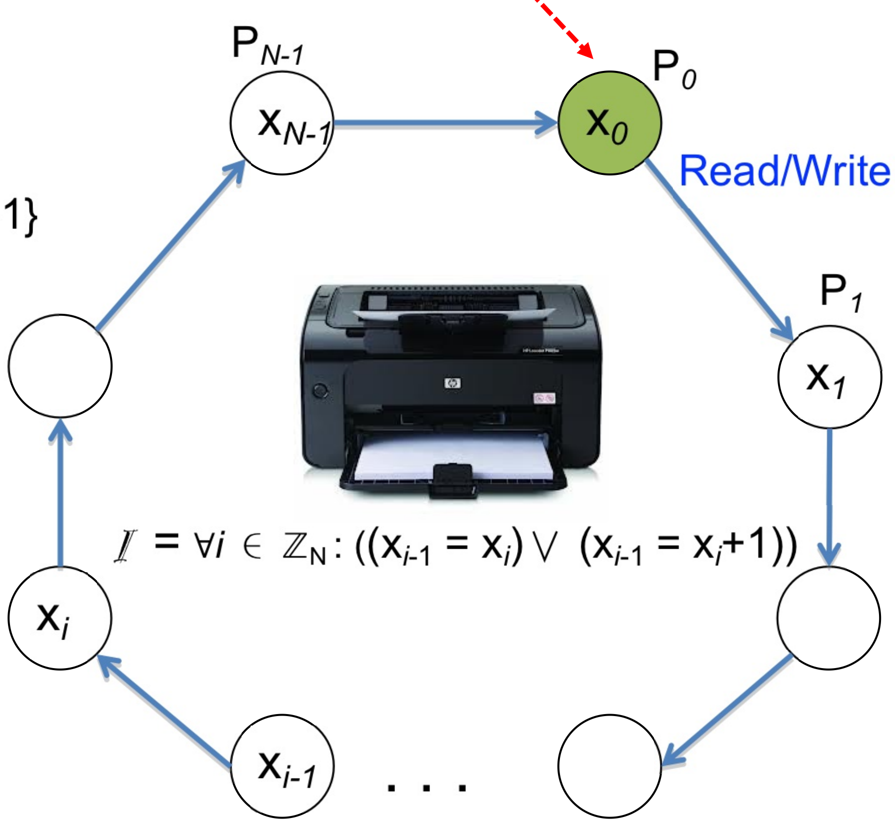
- Process P_i has a variable $x_i \in \mathbb{Z}_N = \{0, 1, \ldots, N-1\}$

- N denotes the total number of processes
- Addition and subtraction are done in modulo N

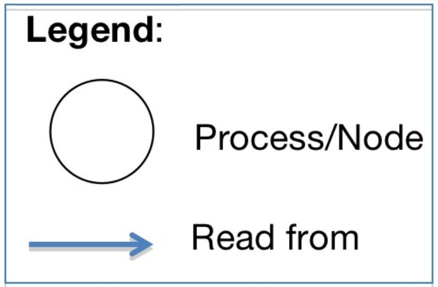
self-disabling actions

π_1: Template process 1

Action$_i$: $x_i \neq x_{i-1} \rightarrow x_i := x_{i-1}$



$\mathcal{I} = \forall i \in \mathbb{Z}_N : ((x_{i-1} = x_i) \vee (x_{i-1} = x_i + 1))$



Family 1: N-1 *symmetric* processes

Problem Statement

$\mathcal{I} = \forall i \in \mathcal{N} : L(x_{i-1}, x_i)$
Variable x_i has **unbounded domain**

Synthesis Algorithm for
Symmetric Uni-Ring

Parameterized Actions

self-stabilizing for

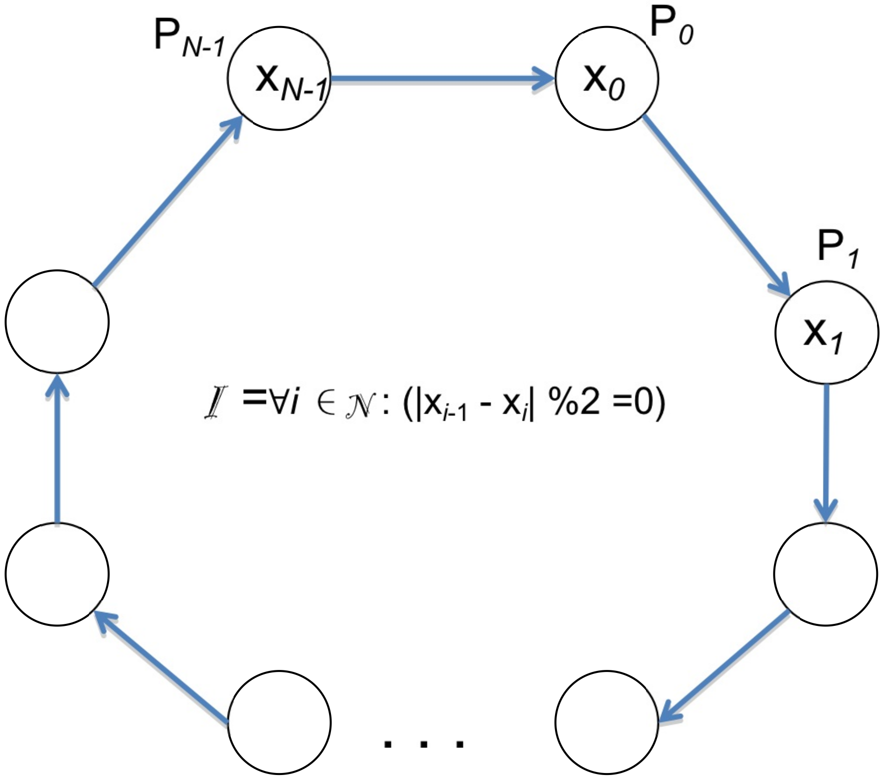
1. an arbitrary number of processes, and
2. **unbounded domain sizes**.

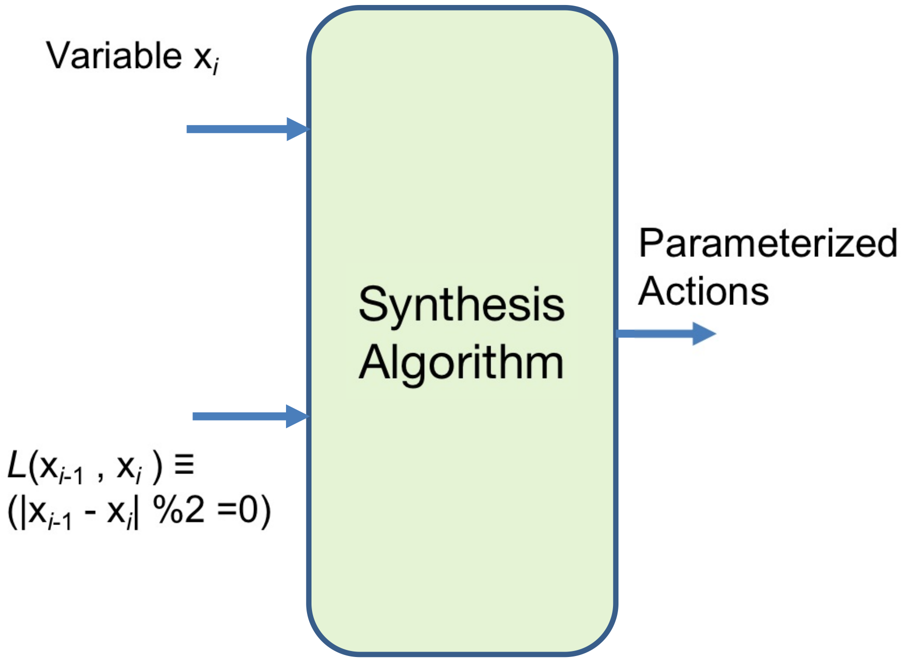
- From any global state, the entire ring eventually converges to a global state in $\mathcal{I}$; i.e., global liveness.

Example: Parity Protocol

Starting from any state, the *symmetric ring* reaches states where all processes agree on a common odd/even parity.

$\mathcal{I} = \forall i \in \mathcal{N}: L(x_{i-1}, x_i)$ where $L(x_{i-1}, x_i) \equiv (|x_{i-1} - x_i| \% 2 = 0)$ and $x_i \in \mathcal{N}$





Graph-Theoretic Representations

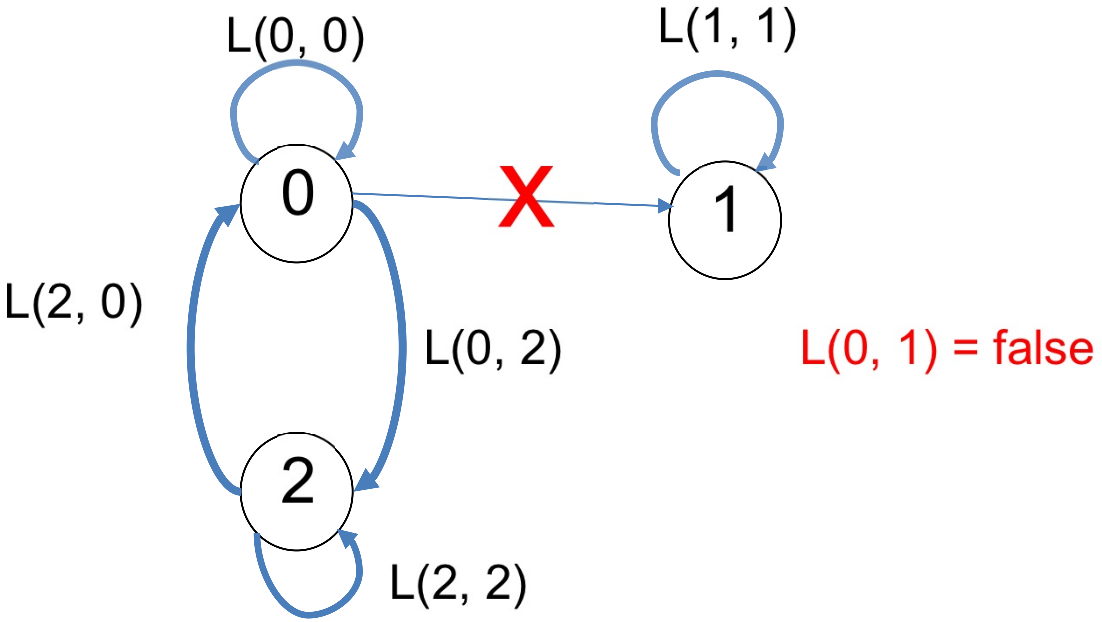
- A goal: Facilitate reasoning in the local state space of the template process; i.e., local reasoning for global correctness.
  - State predicates → Locality Graph
  - Parameterized Actions → Action Graph

Locality Graph of Parity Protocol

- *Vertices*: values in domain of x_i
- *Arcs*: there is an arc from vertex *a* to *b iff L(a, b)* holds.

$\mathcal{I} = \forall i \in \mathbb{Z}_N : L(x_{i-1}, x_i)$ where $L(x_{i-1}, x_i) \equiv (|x_{i-1} - x_i| \% 2 = 0)$

$x_i \in \mathbb{Z}_3 = \{0, 1, 2\}$; i.e., constant-space processes

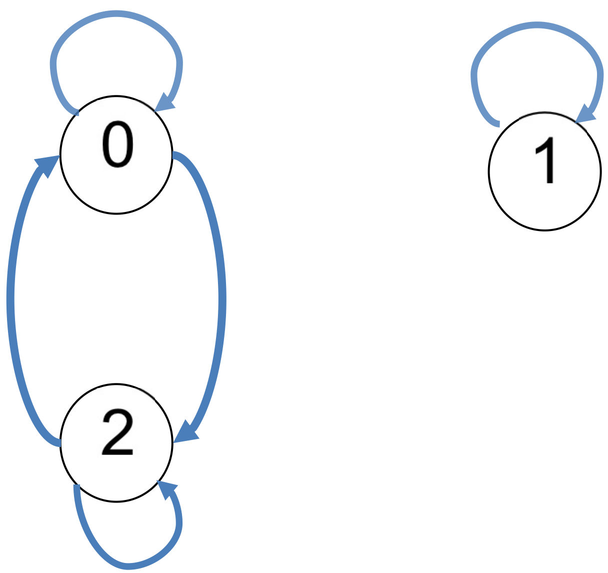


Locality Graph of Parity Protocol

- *Vertices*: values in domain of x_i
- *Arcs*: there is an arc from vertex *a* to *b* *iff* *L(a, b)* holds.

$\mathcal{I} = \forall i \in \mathbb{Z}^+ : L(x_{i-1}, x_i)$ where $L(x_{i-1}, x_i) \equiv (|x_{i-1} - x_i| \% 2 = 0)$

$x_i \in \mathbb{Z}_3 = \{0, 1, 2\}$

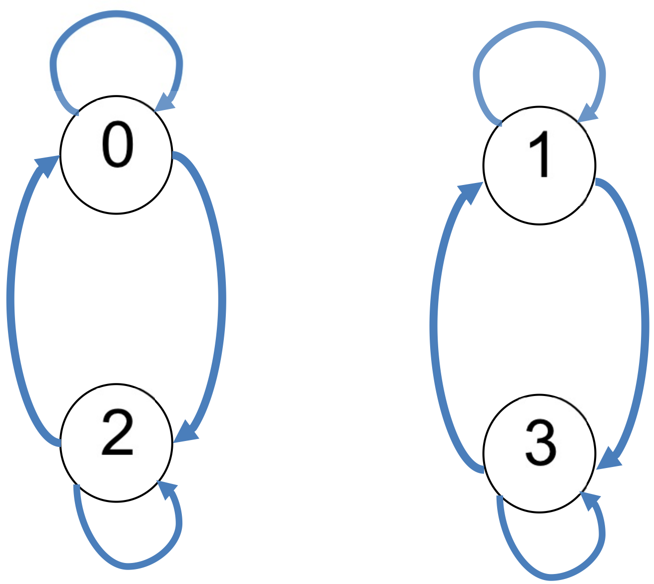


Locality Graph of Parity Protocol

- *Vertices*: values in domain of x_i
- *Arcs*: there is an arc from vertex *a* to *b iff L(a, b)* holds.

$\mathcal{I} = \forall i \in \mathbb{Z}^+ : L(x_{i-1}, x_i)$ where $L(x_{i-1}, x_i) \equiv (|x_{i-1} - x_i| \% 2 = 0)$

$x_i \in \mathbb{Z}_4 = \{0, 1, 2, 3\}$



Action Graph of Parity Protocol

- *Vertices*: values in domain of x_i
- *Labeled arcs*: there is an arc from vertex *a* to *c* with a label *b* iff there is an action $x_{i-1} = a \wedge x_i = b \rightarrow x_i := c$.

$I = \forall i \in \mathbb{Z}^+ : L(x_{i-1}, x_i)$ where $L(x_{i-1}, x_i) \equiv (|x_{i-1} - x_i| \% 2 = 0)$

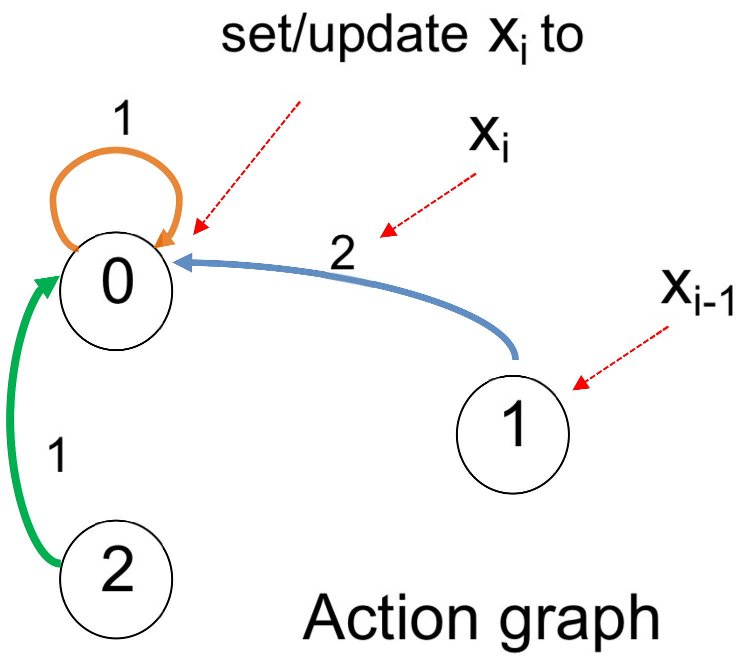
$x_i \in \mathbb{Z}_3 = \{0, 1, 2\}$

$\mathbf{x_{i-1} = 1 \wedge x_i = 2 \rightarrow x_i := 0}$

$x_{i-1} = 2 \wedge x_i = 1 \rightarrow x_i := 0$

$x_{i-1} = 0 \wedge x_i = 1 \rightarrow x_i := 0$

- Each labeled arc is an atomic action



Action graph

Synthesis of Constant-Space Parameterized Protocols

- Theorem: [IEEE TSE 2019]

  Synthesizing SS parameterizes protocols on symmetric uni-rings is decidable for deterministic, *constant-space* and self-disabling processes.

- Theorem: (necessary and sufficient condition) [IEEE TSE 2019]

  There is a PDP p that self-stabilizes to $\mathcal{I} = \forall i \in \mathcal{N}: L(x_{i-1}, x_i)$

  if and only if

  There is some value γ in the domain of x_i such that $L(\gamma, \gamma)$ holds, and the action graph of p is a directed spanning tree rooted at γ.

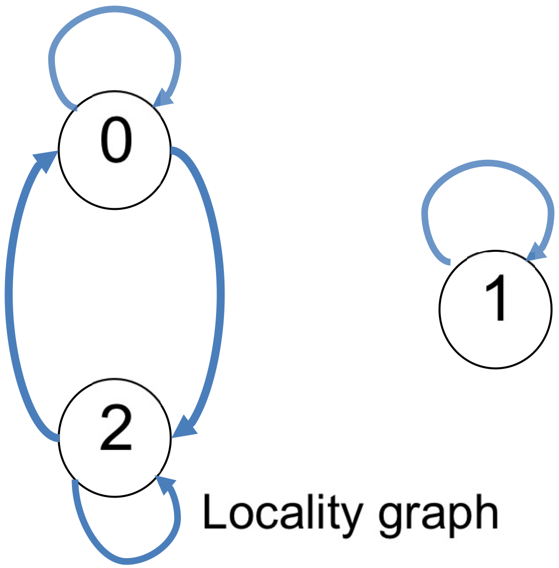
[TSE 2019] Ali Ebnenasir and Alex Klinkhamer, **Topology-specific synthesis of self stabilizing parameterized systems with constant-space processes**, *IEEE Transactions on Software Engineering*, vol. 47, no. 3, pp. 614–629, 2019.

Synthesis for Constant Space

Example: Agree on a common Parity in uni-ring

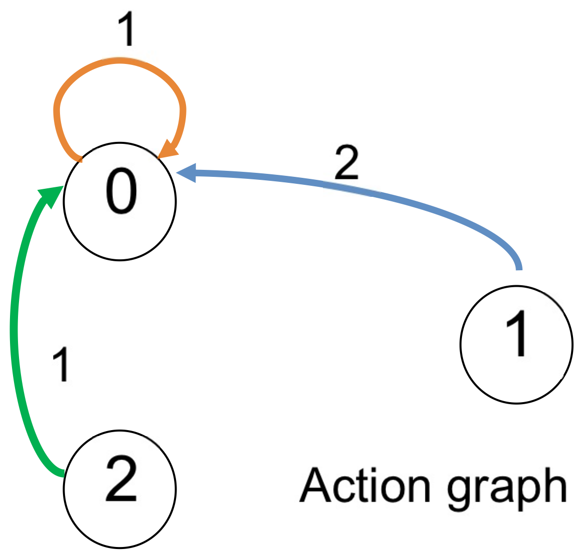
$\mathcal{I} = \forall i \in \mathbb{Z}^+ : L(x_{i\text{-}1}, x_i)$ where $L(x_{i\text{-}1}, x_i) \equiv (|x_{i\text{-}1} - x_i| \,\%2 = 0)$ $x_i \in \mathbb{Z}_3 = \{0, 1, 2\}$

$x_i \in \mathbb{Z}_3 = \{0, 1, 2\}$



Locality graph

Synthesis Algorithm for Constant-Space Uni-Ring [TSE 2019]



Action graph

Parameterized actions:

$x_{i-1} = 1 \wedge x_i = 2 \rightarrow x_i := 0$

$x_{i-1} = 2 \wedge x_i = 1 \rightarrow x_i := 0$

$x_{i-1} = 0 \wedge x_i = 1 \rightarrow x_i := 0$

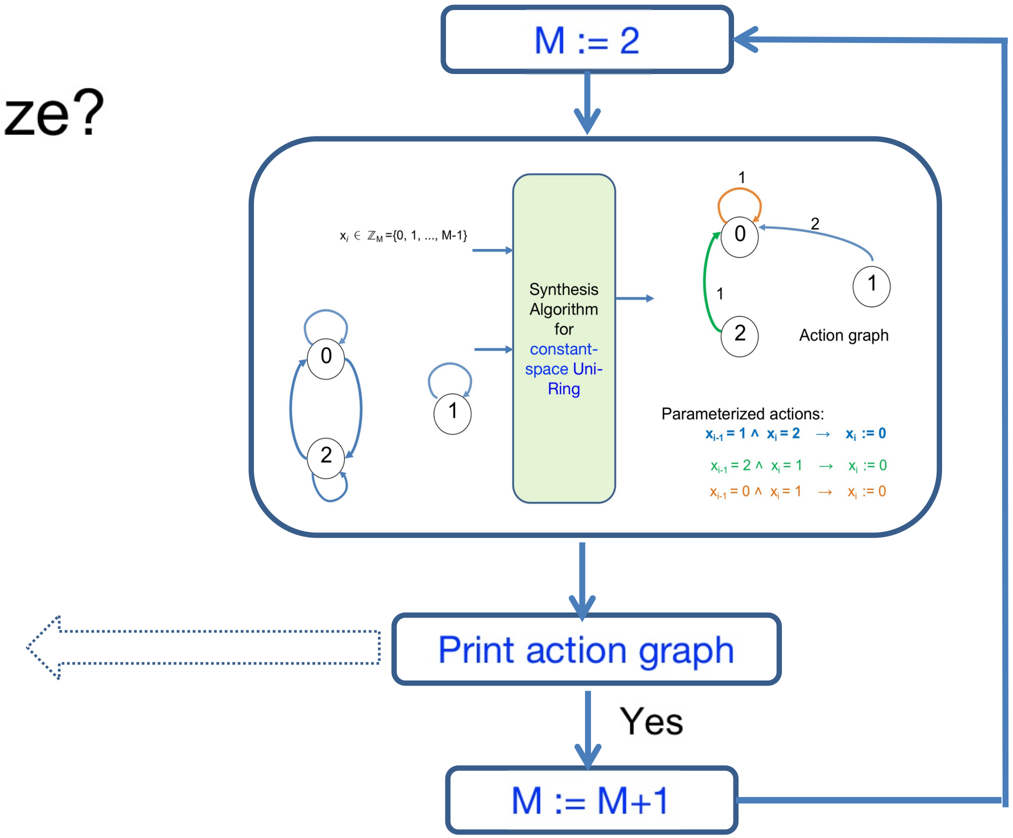
[TSE 2019] Ali Ebnenasir and Alex Klinkhamer, **Topology-specific synthesis of self stabilizing parameterized systems with constant-space processes**, *IEEE Transactions on Software Engineering*, vol. 47, no. 3, pp. 614–629, 2019.

Locality/Action graphs are good for constant-space processes.
What if the variable domain is unbounded?

How to synthesize in unbounded domain?

Enumerate and generalize?

$\{\tau_0, \tau_1, \tau_2, \tau_3, \dots\}$



How to synthesize in unbounded domain?

- Is there a mathematical structure that can generalize such an unbounded set of spanning trees?

- What properties should the unbounded set of spanning trees have so there is a solution?

The spanning trees should grow in a periodic way, eventually forming an unbounded tree.

Linear and Semilinear Sets

- A vector of non-negative integers with dimension $d \geq 1$ is a tuple $(a_1, a_2, ..., a_d) \in \mathcal{N}^d$ where $a_i \in \mathcal{N}$ for $1 \leq i \leq d$

- A non-empty subset of $\mathcal{N}^d$ is linear if it can be represented as a periodic set of vectors

  $\mathcal{L} = \{v_b + \Sigma^n_{i=1} \lambda_i p_i : \lambda_i \in \mathcal{N}\}$ where v_b is the *base vector* and $\{p_1, p_2, ..., p_n\}$ $(n \geq 1)$ in $\mathcal{N}^d$ is a finite set of *period vectors*.

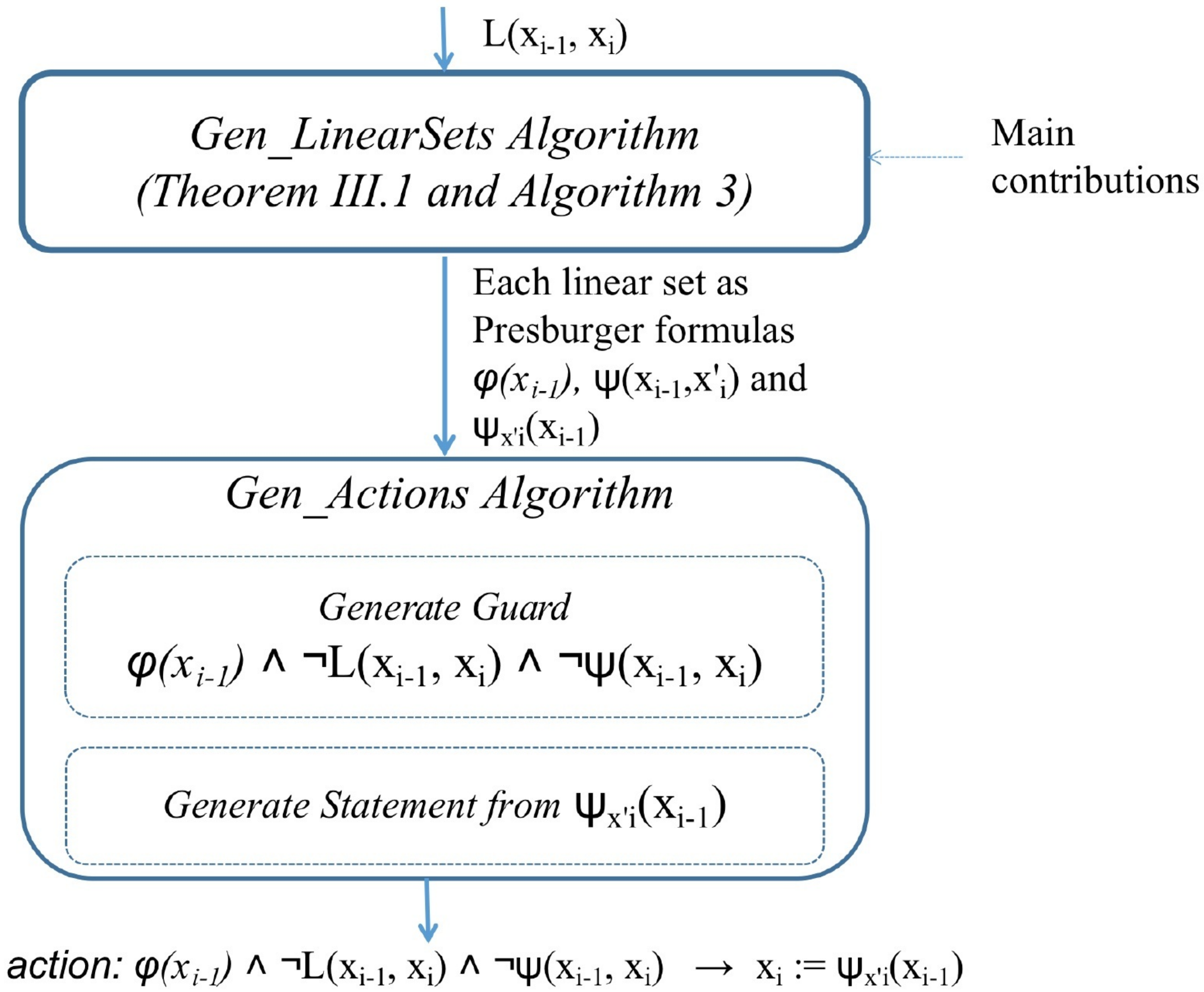
- A semilinear set is a finite union of some linear sets.
  - Semilinear sets are Presburger-definable. [Ginsburg & Spanier 1964]

S. Ginsburg and E. H. Spanier, "**Bounded algol-like languages**," Transactions of the American Mathematical Society, vol. 113, no. 2, pp. 333–368, 1964.

Sufficient Condition for Solvability

- Theorem: (sufficiency)
  - IF the arcs of a γ-rooted unbounded tree for domain sizes $k \geq M$ represent a semilinear set,
  - THEN there is a symmetric protocol *p* that self-stabilizes to $\mathcal{I}$ regardless of
    - the ring size,
    - and variable domain size.

Overview of the Synthesis Algorithm



Generating Semilinear Sets

Finding the Starting Domain Size

- **Step 1**: Search for some domain size M for which there is a γ such that $L(\gamma , \gamma)$ holds and there are solutions modulo M and M+1.
    - Conduct this search up to some upper bound $\mathscr{B}$.

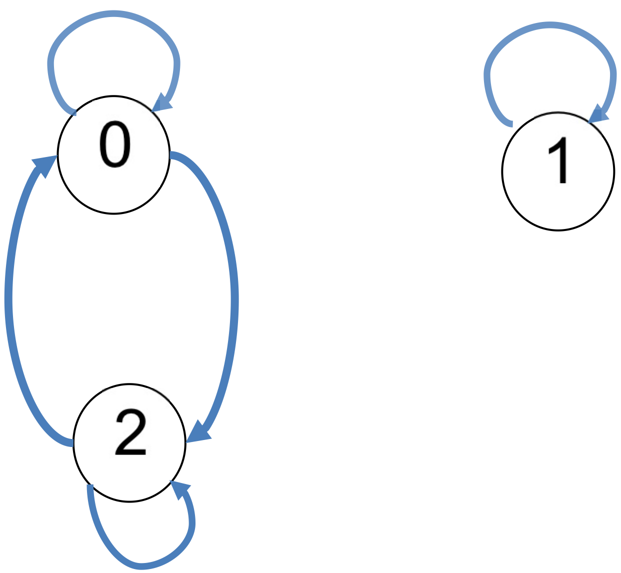
Example: Parity Protocol

Example: Agree on a common Parity in uni-ring

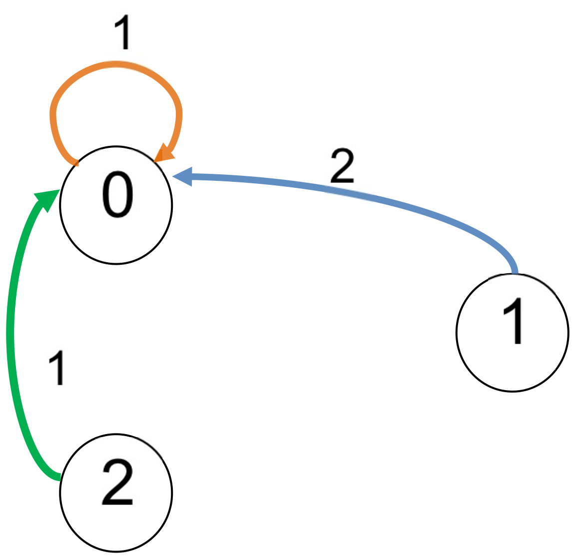
$\mathcal{I} = \forall i \in \mathcal{N}: L(x_{i-1}, x_i)$ where $L(x_{i-1}, x_i) \equiv (|x_{i-1} - x_i| \, \%2 = 0)$

$x_i \in \mathcal{N}$

M=3



Locality graph



Action graph

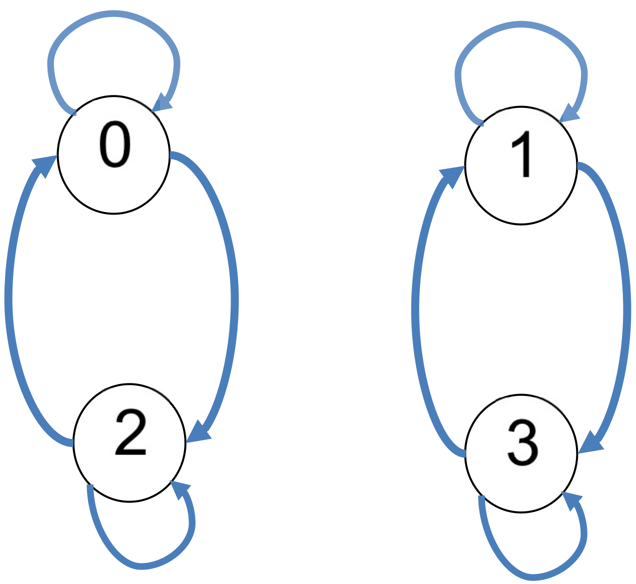
Example: Parity Protocol

Example: Agree on a common Parity in uni-ring

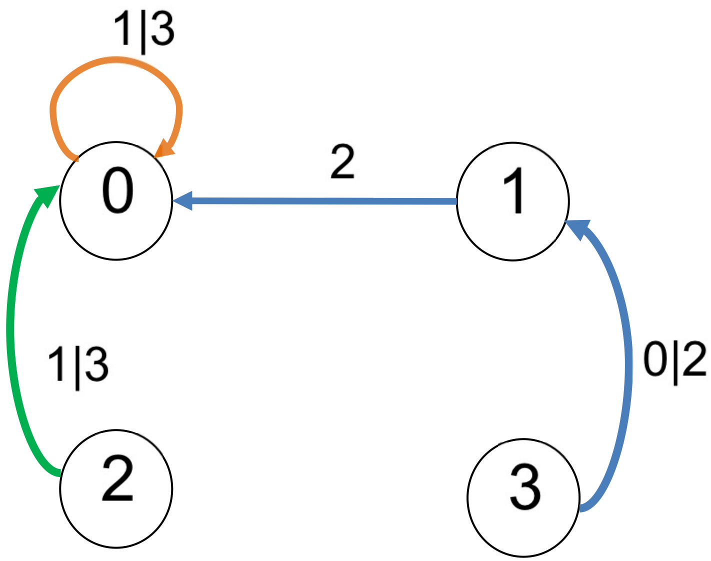
$\mathcal{I} = \forall i \in \mathcal{N} : L(x_{i-1}, x_i)$ where $L(x_{i-1}, x_i) \equiv (|x_{i-1} - x_i| \,\%2 = 0)$

$x_i \in \mathcal{N}$

M=4



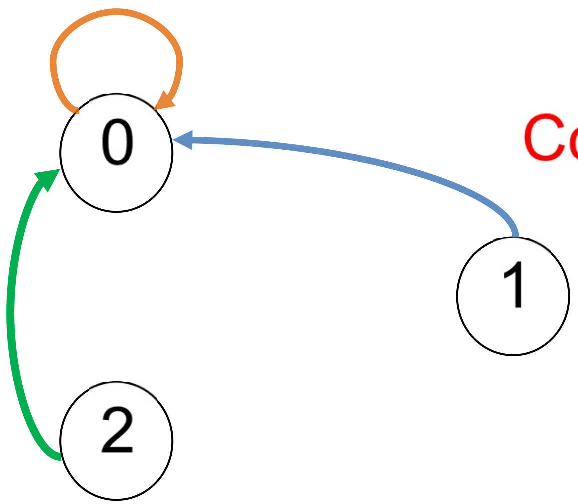
Locality graph



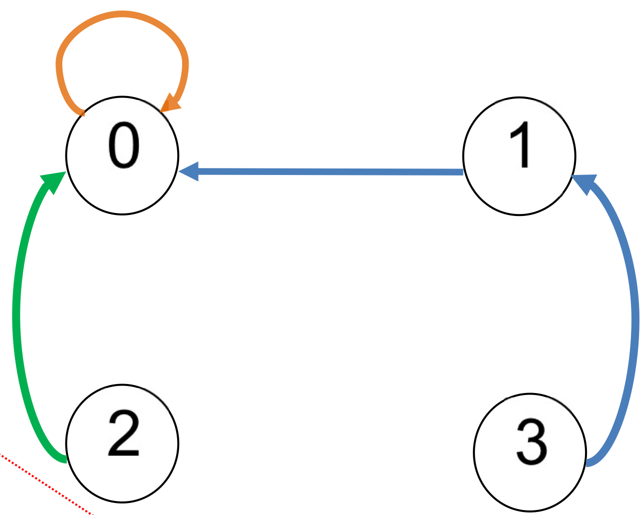
Action graph

Computing the Common Core

- **Step 2**: Compute the Common Core (CC) by taking the intersection of two vector sets



Common Core

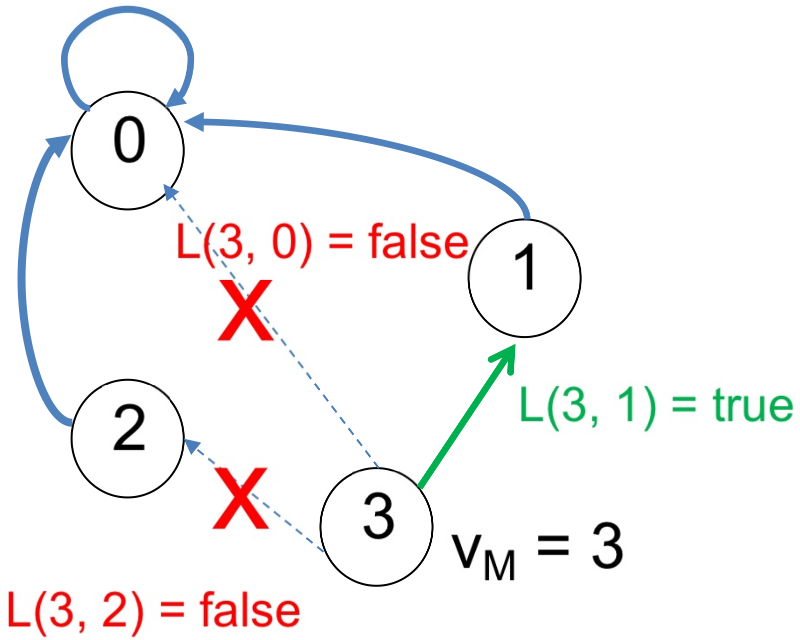


Action graph for M=3
vector set {(0, 0), (1,0), (2,0)}

Action graph for M=4
vector set {**(0, 0), (1,0), (2,0)**, (3,1)}

Compute the Set of Connecting Vertices

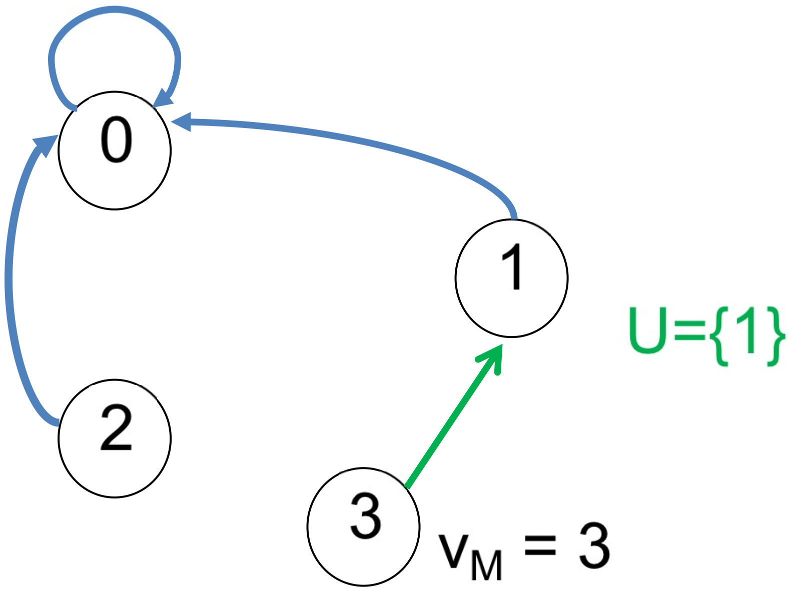
- **Step 3**: Compute the set of vertices $U=\{u \mid L(v_M, u) \text{ holds}\}$ where v_M is the new node due to domain size increase.
  - E.g., Parity $L(x_{i-1}, x_i) \equiv (|x_{i-1} - x_i| \% 2 = 0)$ and $v_M = 3$



Extending the Common Core

Compute the Set of Connecting Vertices

- **Step 3**: Compute the set of vertices U={u | L(v_M, u) holds} where v_M is the new node due to domain size increase.
  - E.g., Parity $L(x_{i-1}, x_i) \equiv (|x_{i-1} - x_i| \%2 = 0)$ and $v_M = 3$

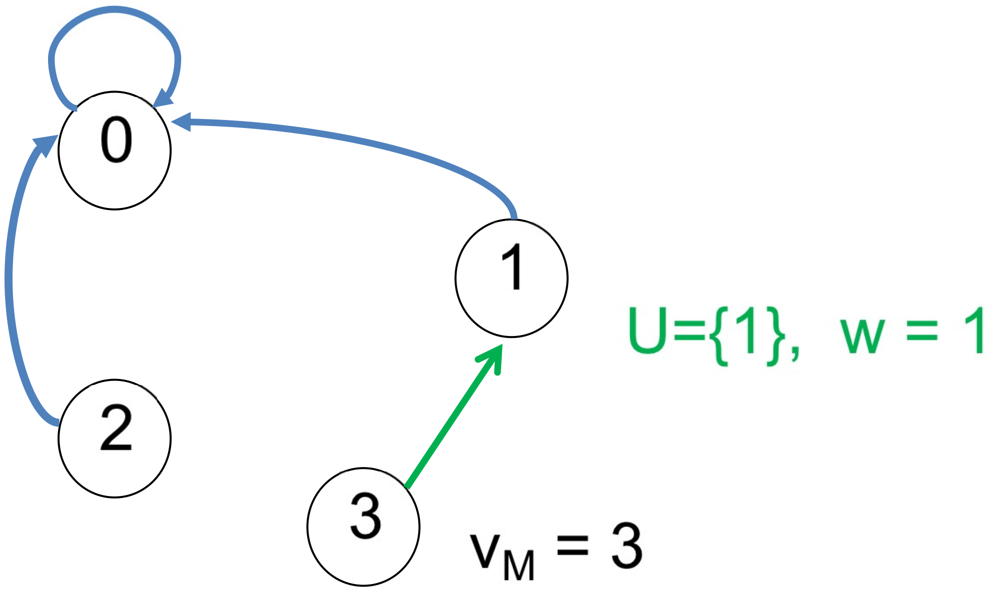


Extending the Common Core

Compute the Unbounded Core

- **Step 4**: Select some vertex w in U and set the base vector to (v_M , w) and the period vector to (1,1)

Base vector (3,1).
Period vector (1,1).

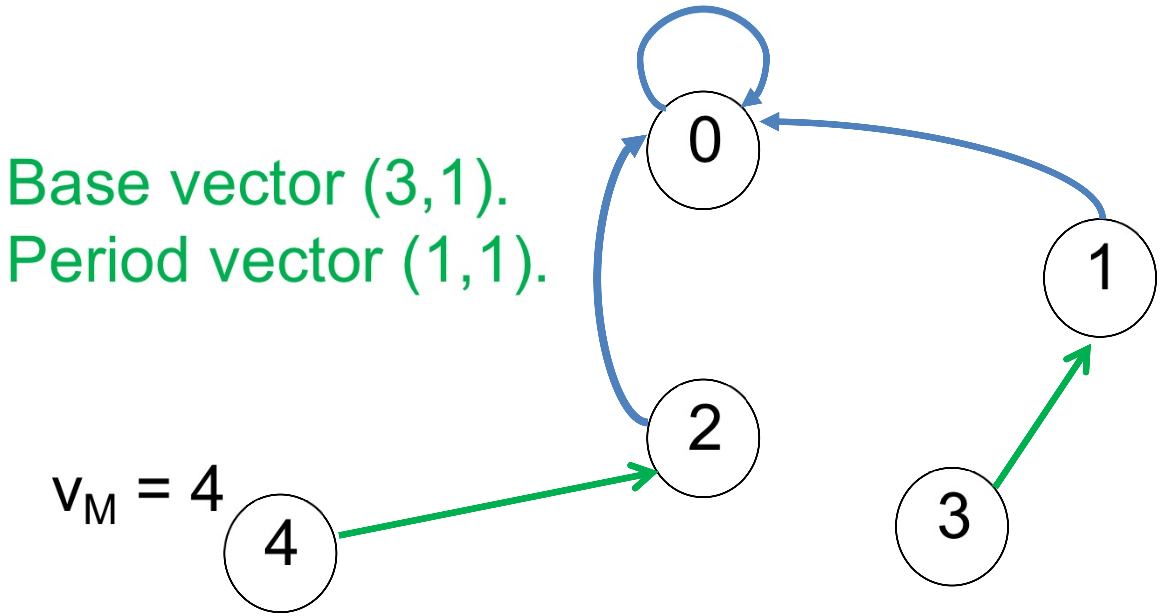


Linear set of Unbounded Core = $\{(v_M , w) + \lambda (1,1) : \lambda \in \mathcal{N}\}$

Linear set of Common Core = { **(0, 0), (1,0), (2,0)**}

Compute the Unbounded Core

- **Step 4**: Select some vertex w in U and set the base vector to (v_M , w) and the period vector to (1,1)
  - *If U = Φ, set the base vector to (v_M , γ) and the period vector to (1,0)*



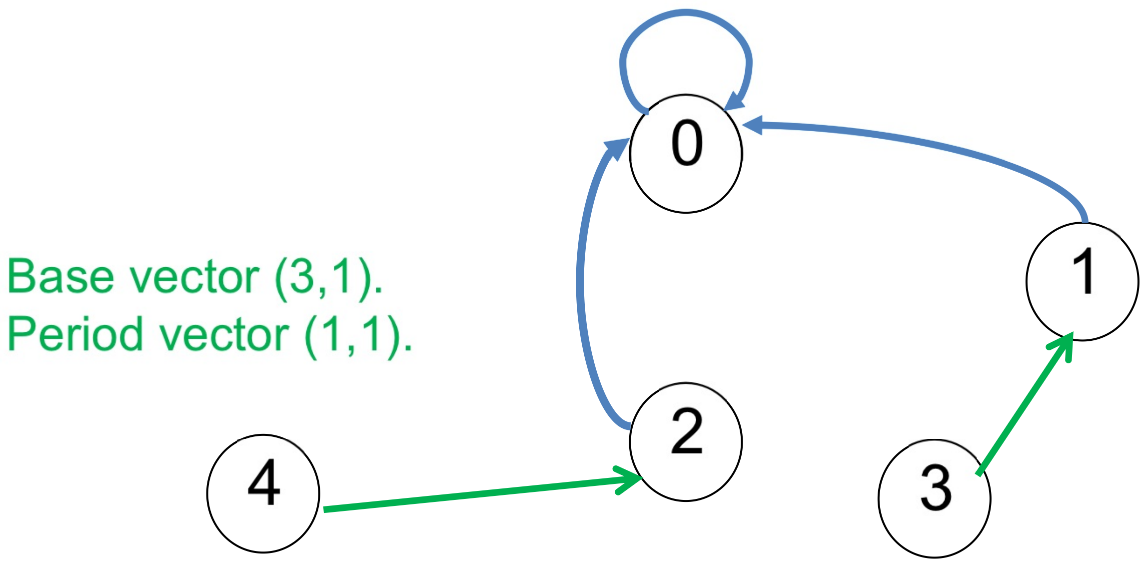
Linear set of Unbounded Core = {(v_M , w) + λ (1,1) : λ ∈ $\mathcal{N}$}

Linear set of Common Core = { **(0, 0), (1,0), (2,0)**}

Linear Sets of Parity Example

- **CC** = {(0, 0), (1,0), (2,0)}
  - Each vector in CC is a linear set
    - Linear set 1 = {(0, 0)}
    - Linear set 2 = {(1, 0)}
    - Linear set 3 = {(2, 0)}

- **UC** = {(3, 1) + λ (1,1) : λ $\in \mathcal{N}$} = {(3,1), (4,2), (5,3), ...}



Specifying Linear Sets as Presburger Formulas

A Linear Set as An Action

- **Step 5**: Linear set $\mathscr{L}$ with base vector (b,b'), and period vector (p,p').

$$\mathscr{L} = \{(x_{i-1}, x'_i) \mid (x_{i-1} = b + \lambda p) \wedge (x'_i = b' + \lambda p') : \lambda \in \mathscr{N}\}$$

x_{i-1}: value of predecessor and x'_i: updated value of x_i

- Represent $\mathscr{L}$ as a parameterized action with unbounded variables

- General format of a parameterized action in a uni-ring:

(Value of x_{i-1} in my predecessor) AND

($\neg L(x_{i-1}, x_i)$) AND (relation of x_i and x_{i-1} that triggers the action) $\longrightarrow$ How x_i should be updated

Extract Three Formulas From Each Linear Set

- **Step 5**: Linear set $\mathcal{L}$ with base vector (b,b'), and period vector (p,p').

$$\mathcal{L} = \{(x_{i-1}, x'_i) \mid (x_{i-1} = b + \lambda p) \wedge (x'_i = b' + \lambda p') : \lambda \in \mathcal{N}\}$$

x_{i-1}: value of predecessor and x'_i: updated value of x_i

- $\varphi(x_{i-1}) \equiv (x_{i-1} = b + \lambda p)$ // Predecessor's value before taking an action
- Relation between x_{i-1} and x'_i, denoted $\psi(x_{i-1}, x'_i)$, that should be established:

$$\psi(x_{i-1}, x'_i) \equiv (x'_i = x_{i-1} + (b' - b) + \lambda(p' - p))$$

$$\psi(x_{i-1}, x_i) \equiv (x_i = x_{i-1} + (b' - b) + \lambda(p' - p))$$

- Factor out x'_i:

$\psi_{x'i}(x_{i-1}) \equiv x_{i-1} + (b' - b) + \lambda(p' - p)$ // Expression that should be assigned to x_i

Action: $\varphi(x_{i-1}) \wedge \neg L(x_{i-1}, x_i) \wedge \neg\psi(x_{i-1}, x_i) \rightarrow x_i := \psi_{x'i}(x_{i-1})$

Linear Sets of Parity Example

- UC $=\{(x_{i-1}, x'_i) \mid (x_{i-1} = 3 + \lambda) \wedge (x'_i = 1 + \lambda) : \lambda \in \mathcal{N}\} =$
  $\{(3, 1) + \lambda (1,1) : \lambda \in \mathcal{N}\} = \{(3,1), (4,2), (5,3), ...\}$

- Formulas:
  - $\varphi(x_{i-1}) \equiv (x_{i-1} = 3 + \lambda) \equiv (x_{i-1} \geq 3)$
  - $\psi(x_{i-1}, x'_i) \equiv (x'_i = x_{i-1} + (b' - b) + \lambda(p' - p)) \equiv (x'_i = x_{i-1} + (1 - 3) + \lambda(1 - 1))$
    - $\psi(x_{i-1}, x'_i) \equiv (x'_i = x_{i-1} - 2)$ // Thus, the action assignment is $x_i := x_{i-1} - 2$ and self-disabling constraint $(x_i \neq x_{i-1} - 2)$
  - $\psi_{x'i}(x_{i-1}) \equiv (x_{i-1} - 2)$

Action: $\varphi(x_{i-1}) \wedge \neg L(x_{i-1}, x_i) \wedge \neg\psi(x_{i-1}, x_i) \rightarrow x_i := \psi_{x'i}(x_{i-1})$

$(x_{i-1} \geq 3) \wedge (|x_{i-1} - x_i| \,\%2 \neq 0) \wedge (x_i \neq x_{i-1} - 2) \rightarrow x_i := x_{i-1} - 2$

Actions of Parity Example

- Self-stabilizing Parity protocol:

1. Action synthesized corresponding to the first three linear sets in the common core CC:

$(x_{i-1} \leq 2) \wedge (|x_{i-1} - x_i| \% 2 \neq 0) \wedge (x_i \neq 0) \rightarrow x_i := 0$

2. Action synthesized corresponding to the linear set of the unbounded core UC:

$(x_{i-1} \geq 3) \wedge (|x_{i-1} - x_i| \% 2 \neq 0) \wedge (x_i \neq x_{i-1} - 2) \rightarrow x_i := x_{i-1} - 2$

More examples in the paper and tech report.

Linear and Semilinear Sets

- When variable domains are unbounded:
  - a parameterized action is captured by a linear set, and
  - the template process is represented by a semilinear set.

Related Work

- Verification and Synthesis (V&S) of PDS are in general undecidable problems.
- Existing methods:
  - *Pairwise synthesis*: safety properties and local liveness in symmetric systems [Attie and Emerson 1998]
  - *Abstraction methods*: create finite approximations of PDS (e.g., counter abstraction) and conduct verification [Pnueli et al. 2002]
  - *Regular model checking*: use regular languages to model PDS [Abdulla et al. 2004]
  - *Invisible invariants/ranking*: generate implicit local invariants and generalize [Fang et al. 2006]
  - *Network invariants*: prove safety by parallel compositions that are invariant to correctness [Wolper and Lovinfosse 1989]
  - *Parameterized synthesis*: based on small model theorems (i.e., cutoff) and SMT-based bounded synthesis [Jacobs and Bloem 2012]
  - *Well-founded proof spaces*: prove safety and liveness of infinite traces by showing that traces terminate [Farzan et al. 2016]
  - *Synthesis of Threshold Automata (TA)*: complete sketches of TA using counter abstraction [Lazi et al. 2018

Mostly focus on safety and local liveness under restrictive assumptions (e.g., fair scheduling).

Contributions

- Utilize semilinear sets for synthesis of unbounded SS PDP on uni-rings
- Sufficient condition for synthesis of SS PDP on uni-rings with
  - unbounded number of processes, and
  - unbounded variable domains.
- A sound synthesis algorithm

Open Problems

- A foundation for synthesis of unbounded parametrized protocols using semilinear sets
  - Other topologies, both uni-directional and bi-directional
- Parameterized protocols with multiple families of symmetric processes (e.g., Dijkstra's token passing)
- Composition of elementary topologies

Thank you.

- Acknowledgement
  - Former graduate students:
    - Dr. Alex Klinkhamer
      - Google (Mountain View, CA)
    - Dr. Aly Farahat
      - Intuitive Surgical Inc. (Bay Area, CA)
    - Dr. Amer Tahat
      - Pennsylvania State University
    - Several other M.Sc. students
  - NSF grants CCF-1116546 and CCF-0950678
  - Michigan Tech's Research Excellence Fund