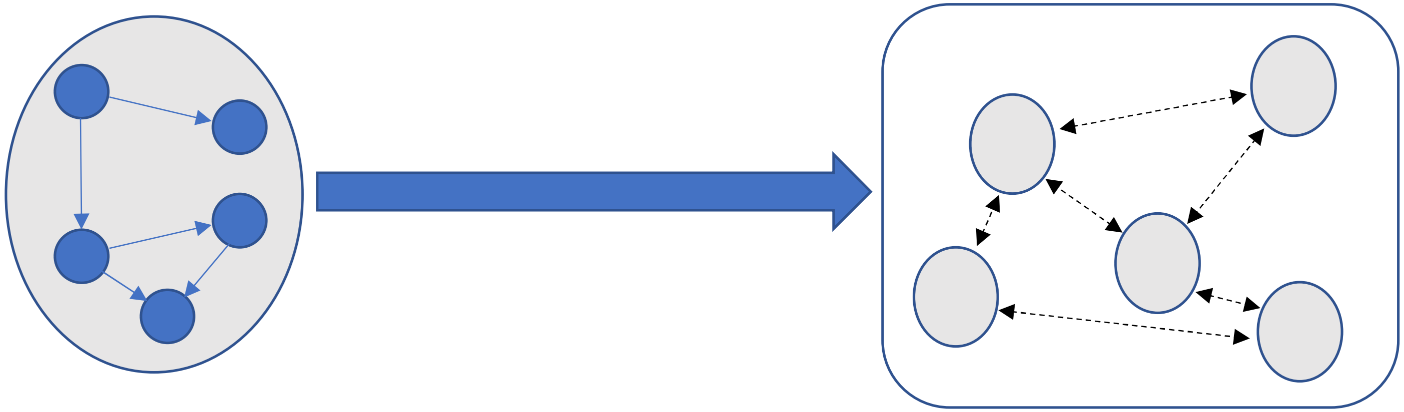
Approaching Safety for Parameterized Systems via View Abstraction

Philip Offtermatt

Parameterized Systems



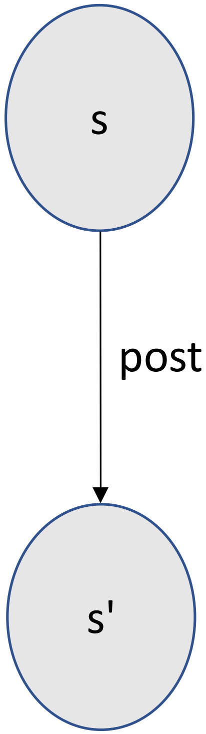
One Process

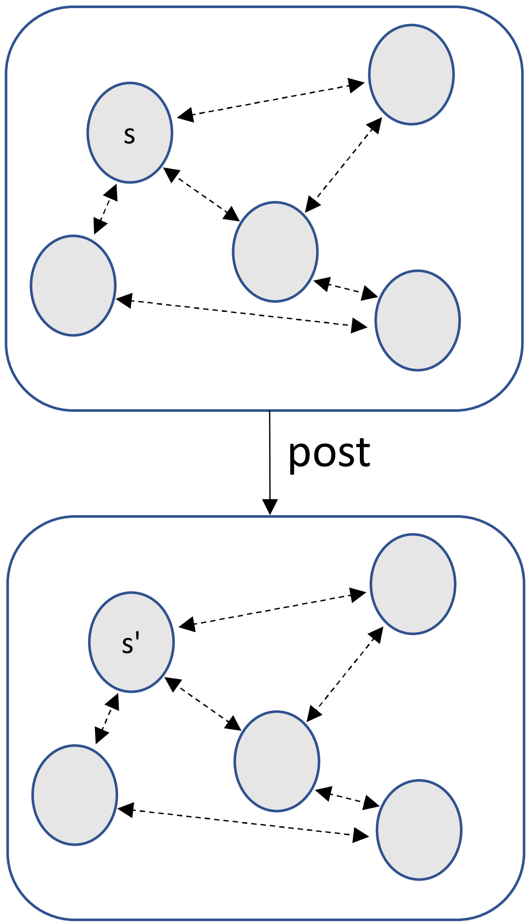
Many interacting Processes

Why "parameterized"?

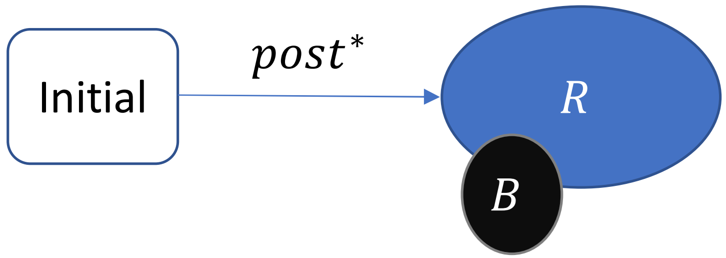
Parameter = interaction topology, number of processes

Parameterized Systems





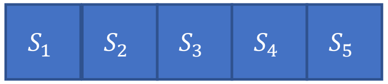
Safety: Can we reach bad configurations?



Do R and B intersect?

Formal Model

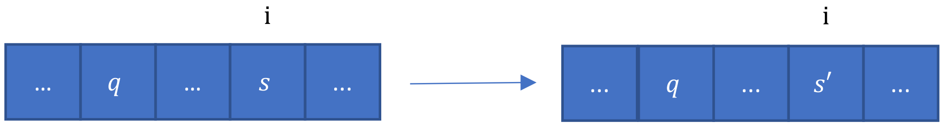
Configuration: word of states $c =$



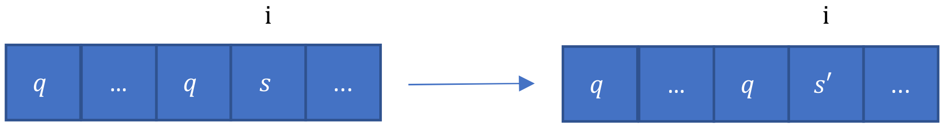
Transitions:

- Local: s → s’

- Global: From point of view of process i
  - Existential: $If\ \exists j \circ i{:}\ c[j] = q\ then\ s \rightarrow s' else\ s\ \rightarrow\ s''$



  - Universal: $If\ \forall j\ \circ\ i{:}\ c[j] = q\ then\ s \rightarrow\ s' else\ s\ \rightarrow\ s''$

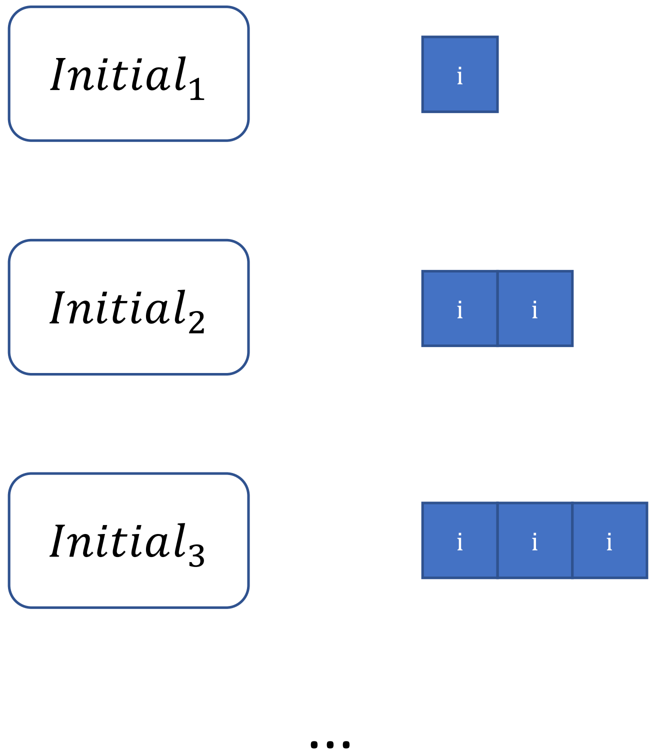


$\circ$: <, >, ≠

Parameterized Verification

Challenge: Infinitely many instances!

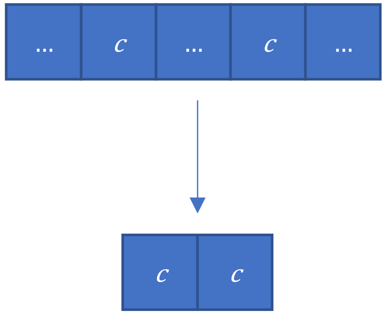
Can we prove all of them safe?



Parameterized Verification

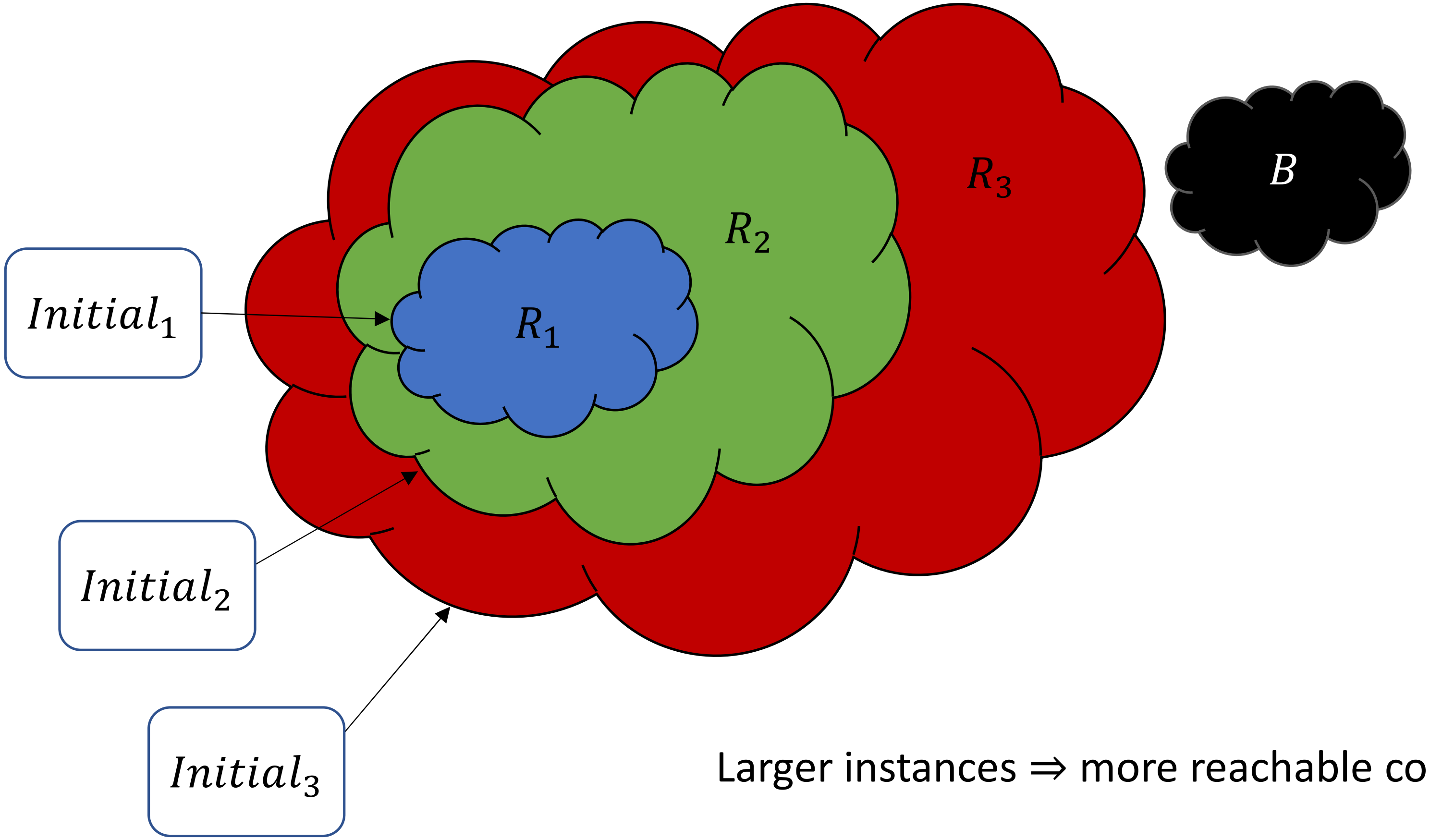
Bad configurations:

- Example - Mutual Exclusion: No two processes in the critical section c at the same time.
- $B = (\uparrow B_{min})$: Upward closure of minimal bad configurations



Bad configurations have a minimal bad element as subword

Forward Reachability



Larger instances ⇒ more reachable configurations!

If system is unsafe, forward reachability (eventually) finds out!

View Abstraction

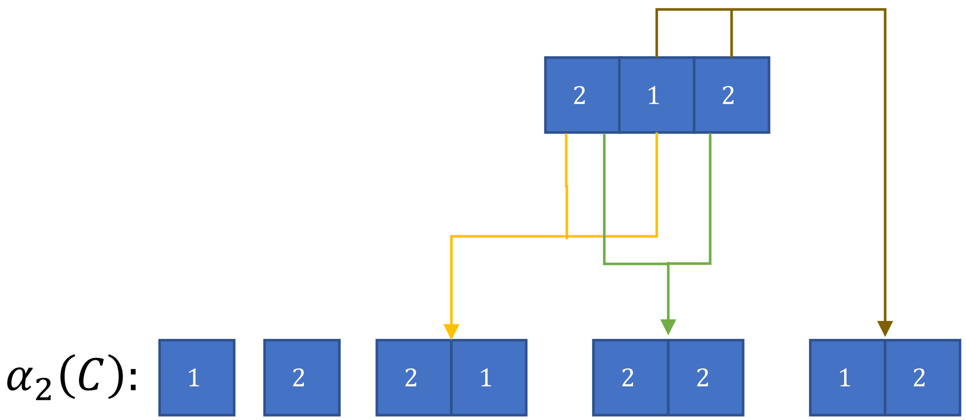
Unsafe case: Forward Reachability until we find an unsafe example

Safe case: Need to prove all instances safe! $\Rightarrow$View Abstraction

View Abstraction

Input: A configuration C

Abstraction:
$\alpha_k(C)$: Subwords (views)
of length up to k of C



View Abstraction

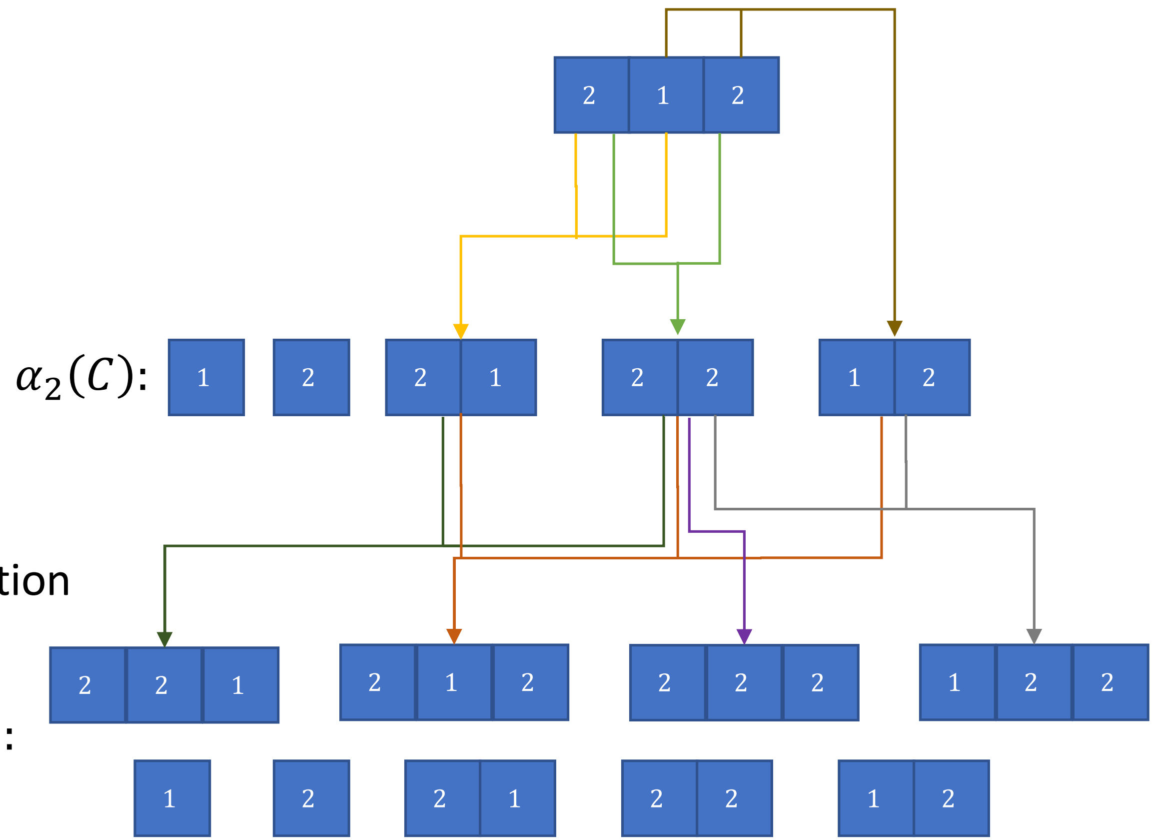
Input: A configuration C

Abstraction:

$\alpha_k(C)$: Subwords (views) of length up to k of C

Reconstruction:

$\oint_k^l(V)$: Configurations up to size l where the k-abstraction is a subset of V



View Abstraction

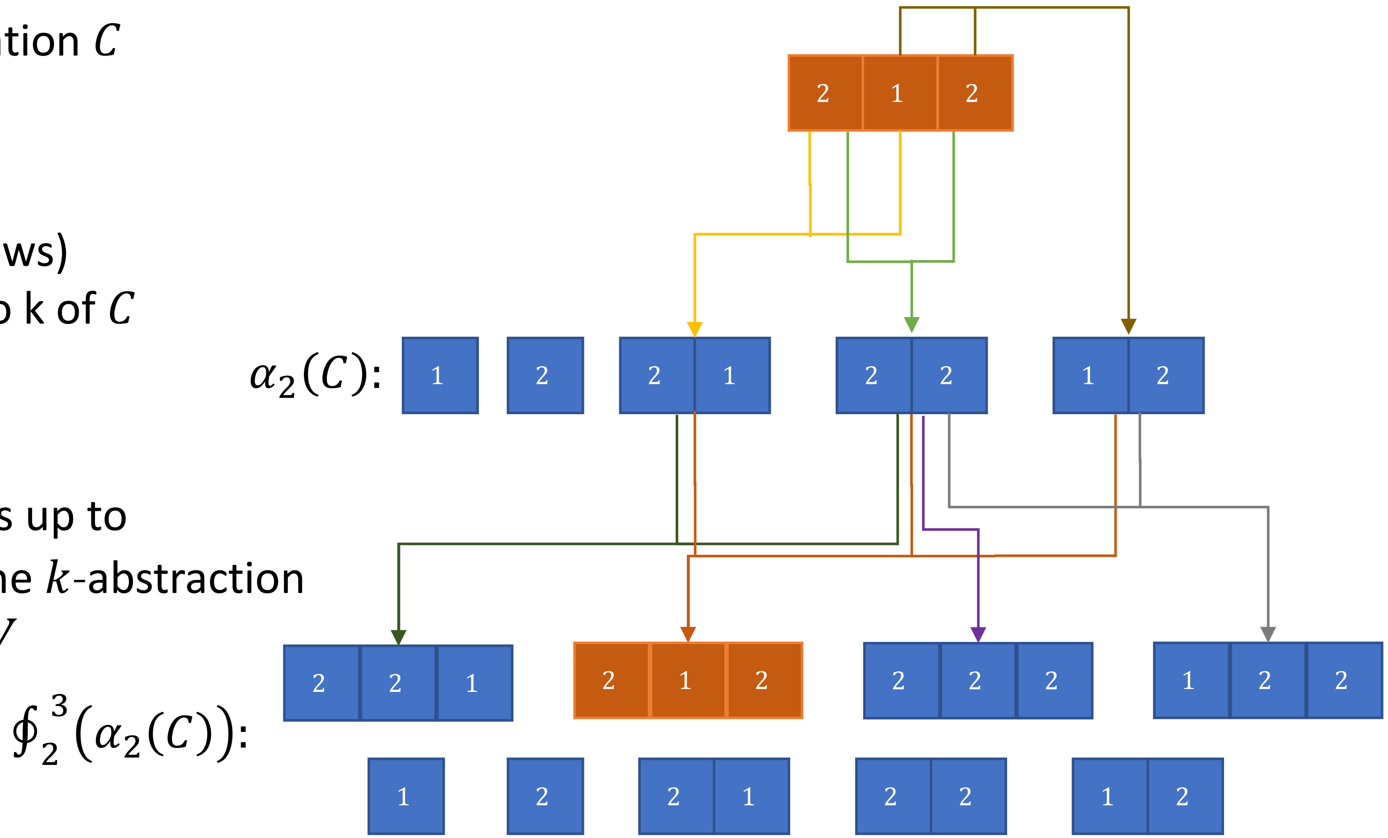
Input: A configuration C

Abstraction:

$\alpha_k(C)$: Subwords (views)
of length up to k of C

Reconstruction:

$\oint_k^l(V)$: Configurations up to
size l where the k-abstraction
is a subset of V



Overapproximation of C

We also allow abstraction to be
applied to sets of configurations.

View Abstraction

$$C = L(1^*23^+)$$

$$\alpha_1 = \{1, 2, 3\}$$
$$\oint_1^\infty = L((1|2|3)^*)$$

$$\alpha_2 = \alpha_1 \cup \{11, 12, 13, 23, 33\}$$
$$\oint_2^\infty = L(1^*(2|\epsilon)3^*)$$

$$\alpha_3 = \alpha_2 \cup \{111, 112, 113, 123, 133, 233, 333\}$$
$$\oint_3^\infty = L(1^*(2|\epsilon)3^*)$$

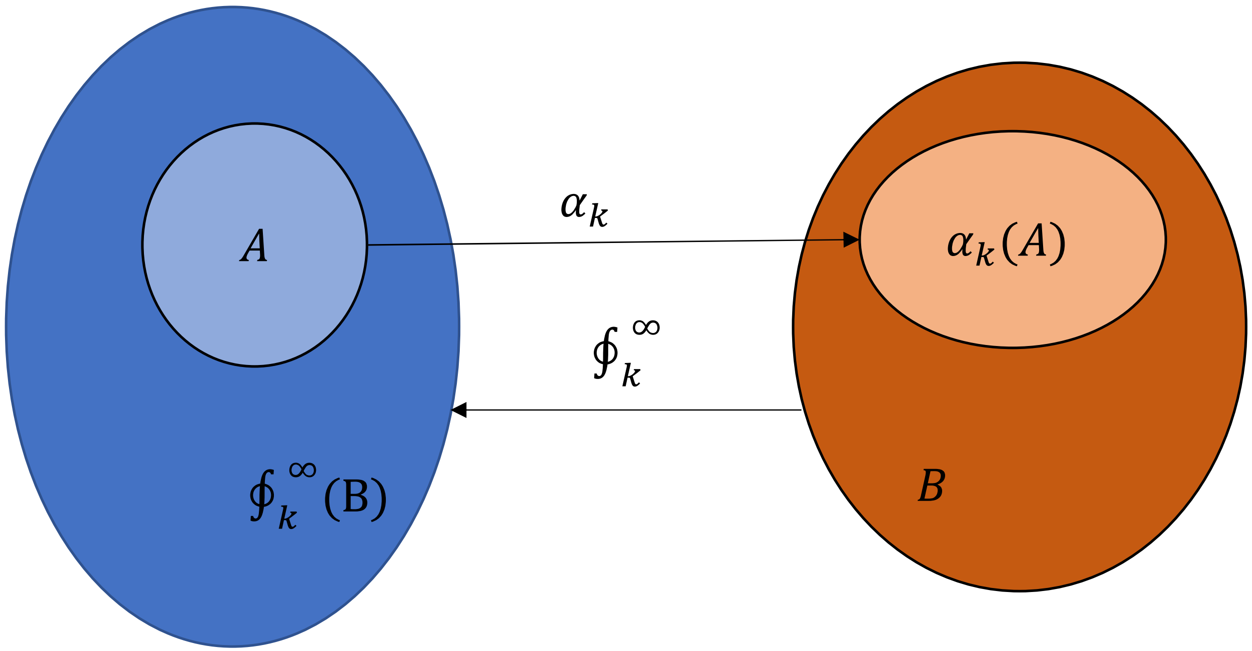
...

Overapproximation becomes more precise with growing k

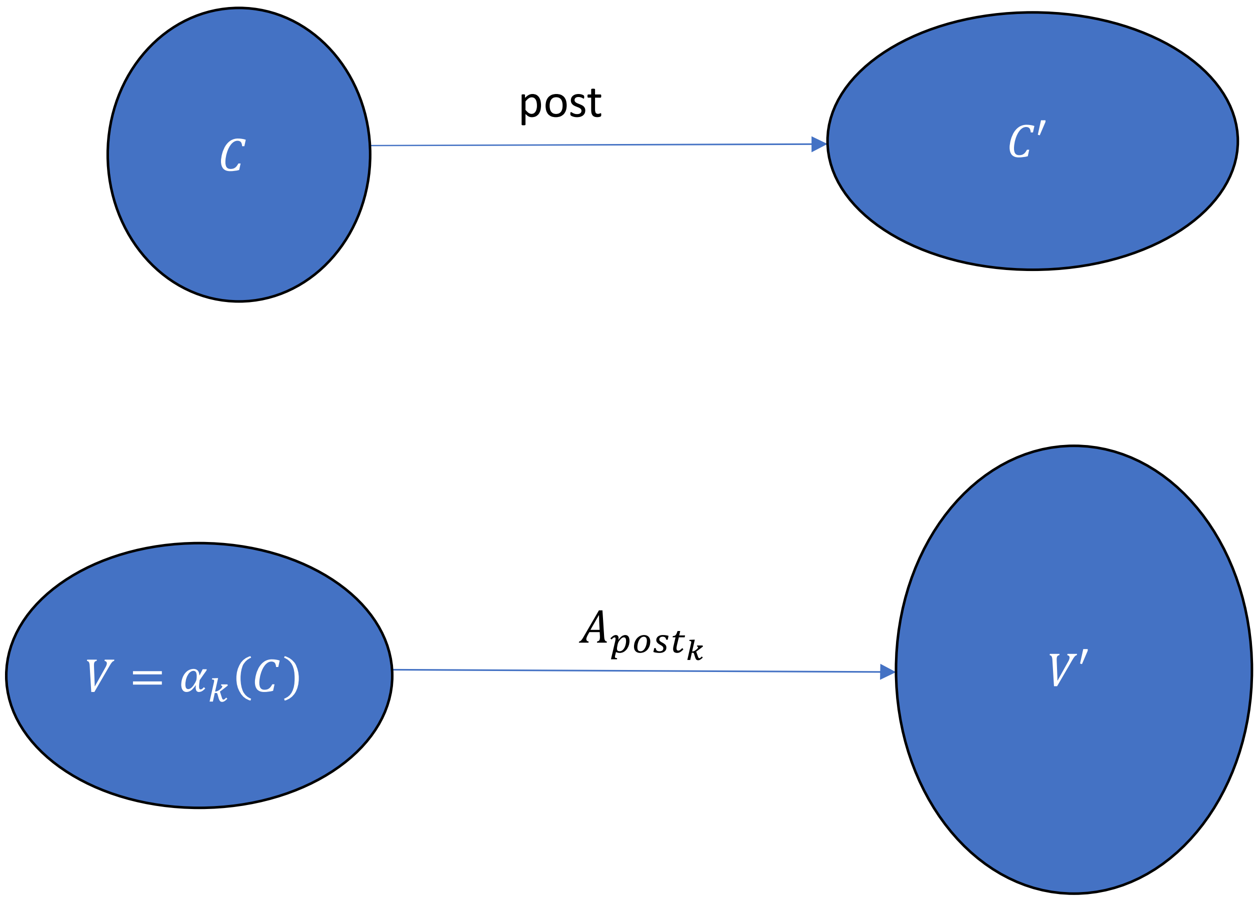
After some point, no new patterns appear

Abstraction/Reconstruction is a Galois Connection

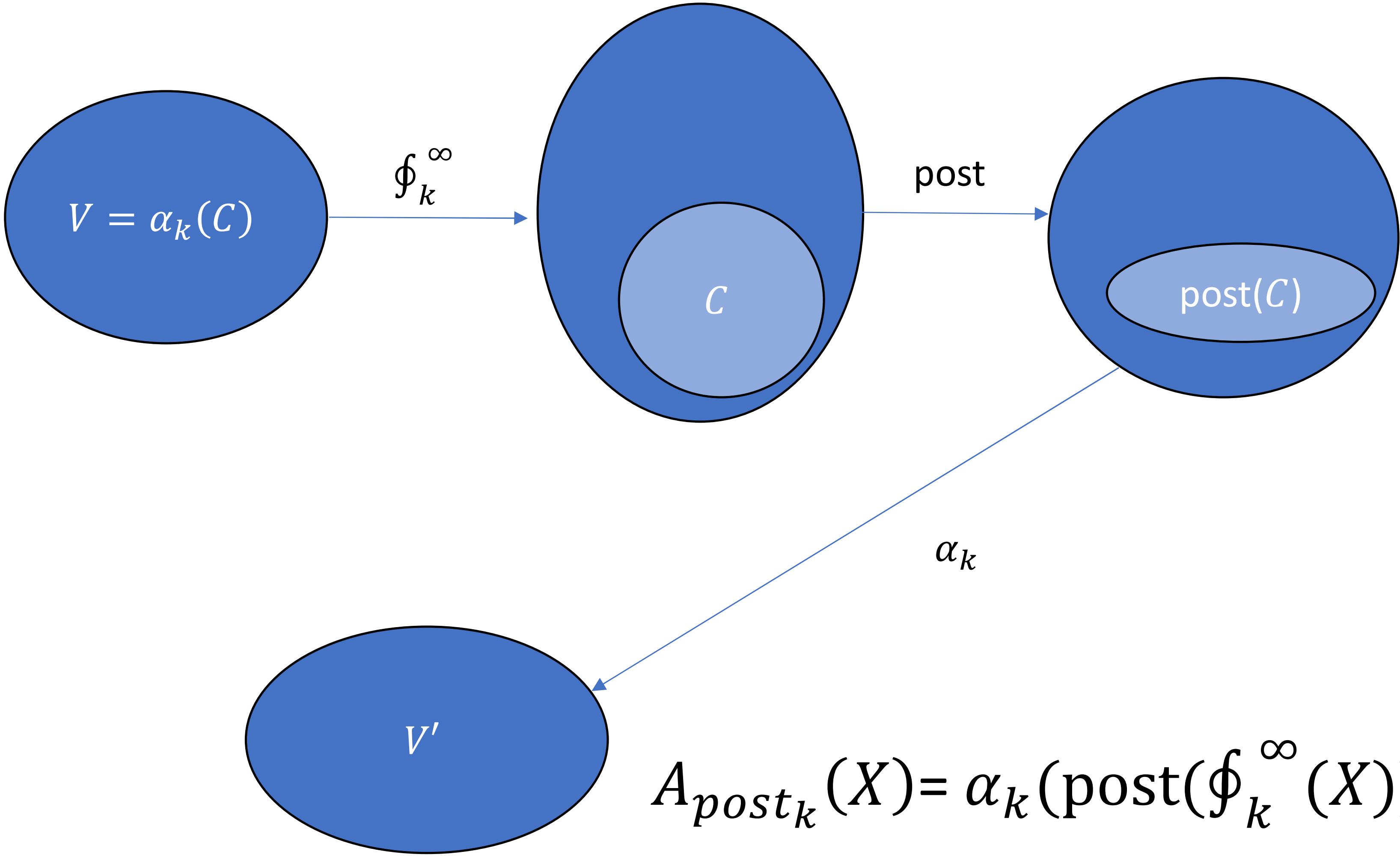
$$\alpha_k(A) \subseteq B \leftrightarrow A \subseteq \oint_k^{\infty}(B)$$



Abstract Post



Abstract Post



$$A_{post_k}(X) = \alpha_k(\text{post}(\oint_k^\infty(X)))$$

Abstract Post Fixpoint

$$V_k^0 = \alpha_k(Initial)$$
$$V_k^{i+1} = V_k^i \cup \alpha_k(\mathrm{post}(\oint_k^{\infty}(V_k^i)))$$

V_k: Least fixpoint

Abstract Post Fixpoint

$$\alpha_k(\text{post}(\oint_k^{\infty}(V_k))) \subseteq V_k \text{ and } \alpha_k(Initial) \subseteq V_k$$

$$\Rightarrow \text{post}(\oint_k^{\infty}(V_k))) \subseteq \oint_k^{\infty} V_k \text{ and } Initial \subseteq \oint_k^{\infty} V_k$$

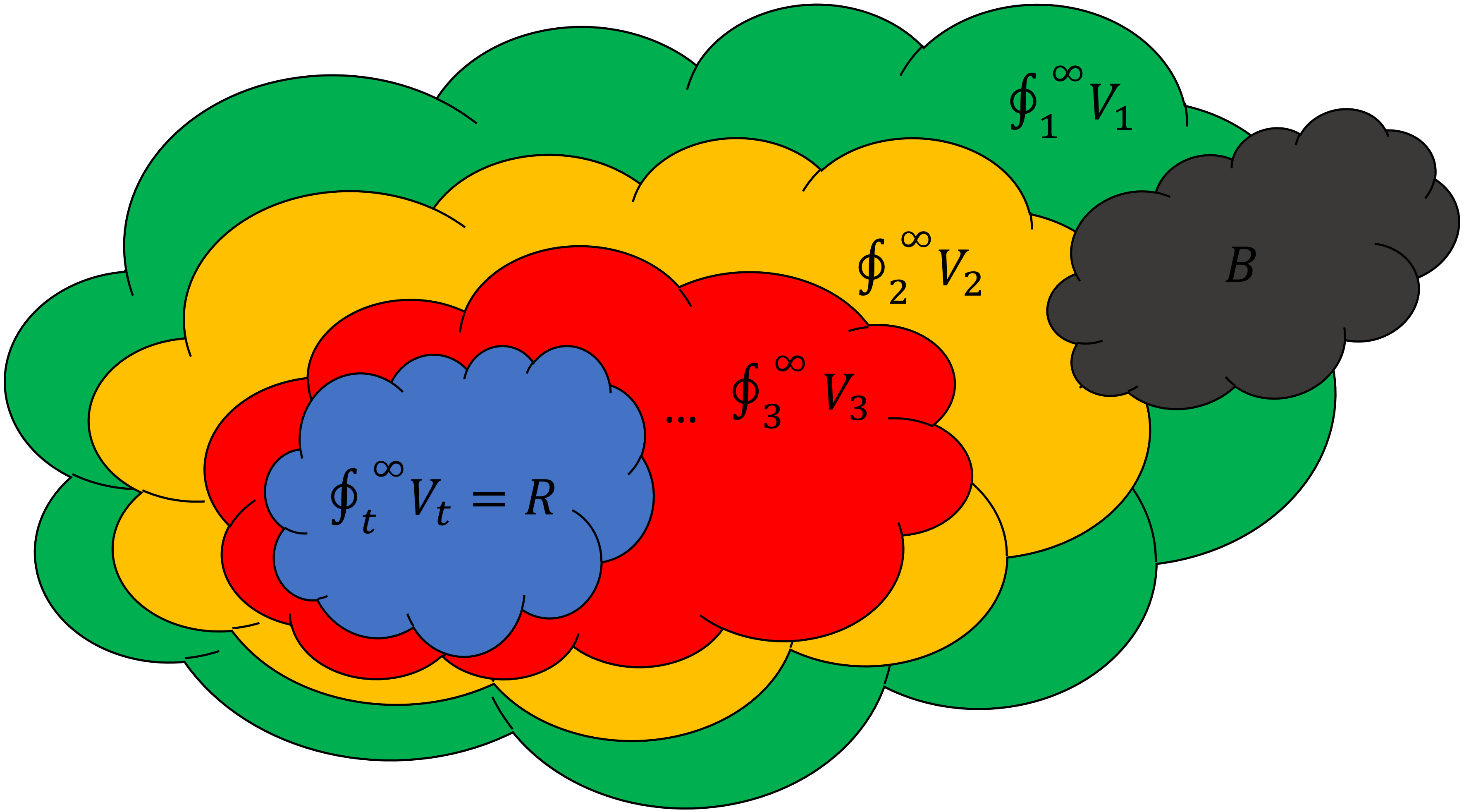
$$\Rightarrow \oint_k^{\infty} V_k \text{ is a fixpoint of post that covers } Initial$$

$$\Rightarrow R \subseteq \oint_k^{\infty} V_k$$

$Galois\ Connection$: $\alpha_k(A) \subseteq B \;\; A \subseteq \oint_k^{\infty}(B)$

Abstract Post Fixpoint

Fixpoints have increasing precision and
eventually reach R



Algorithm Sketch

$\boldsymbol{for}\ k := 1\ \boldsymbol{to}\ \infty\ \boldsymbol{do}$

$\quad \boldsymbol{if}\ R_k \cap B \neq \emptyset\ \boldsymbol{then}\ return\ \mathbf{Unsafe}$

$\quad V_k := \mu X.\, \alpha_k(Initial) \cup \alpha_k(\mathrm{post}(\oint_k^{\infty}(X)))$

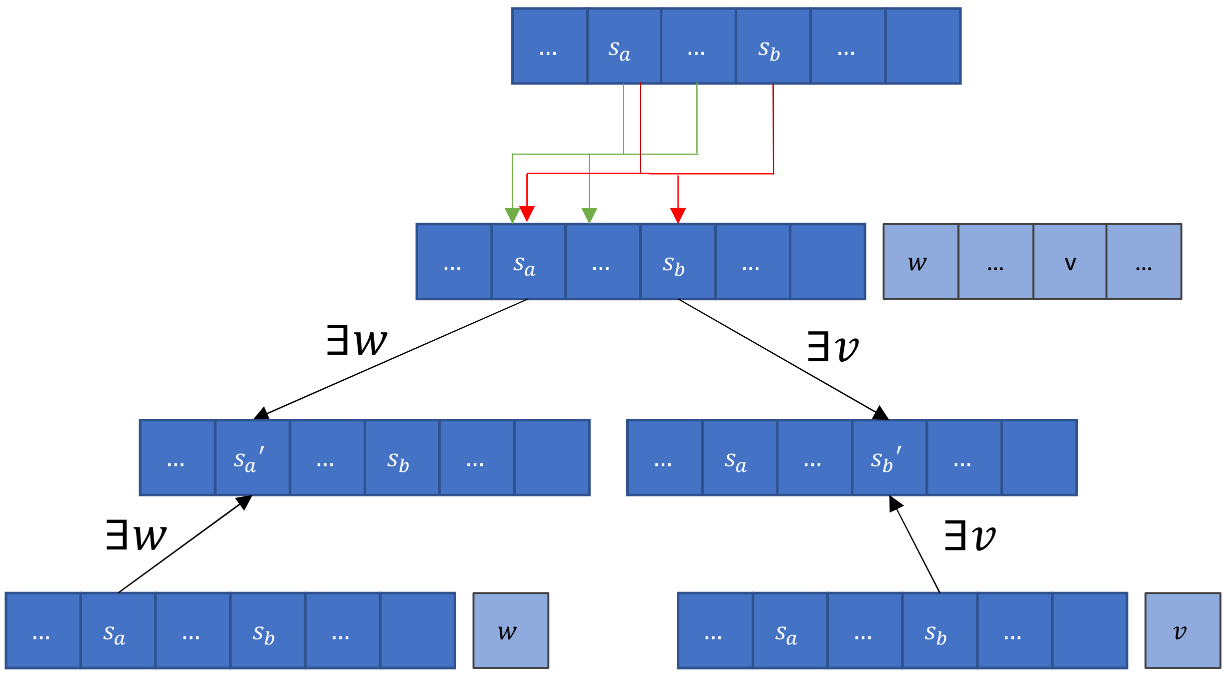
$\quad \boldsymbol{if}\ \oint_k^{\infty}(V_k) \cap B = \emptyset\ \boldsymbol{then}\ return\ \mathbf{Safe}$

Problem: $\oint_k^{\infty}(X)$ and $\oint_k^{\infty}(V_k)$ can be infinite!

Witness Processes

Reconstruction with one
additional process is enough!

$$\alpha_k(\text{post}(\oint_k^{\infty}(X))) \cup X = \alpha_k(\text{post}(\oint_k^{k+1}(X))) \cup X$$



Algorithm Sketch

$\boldsymbol{for}\ k := 1\ \boldsymbol{to}\ \infty\ \boldsymbol{do}$

$\quad \boldsymbol{if}\ R_k \cap B \neq \emptyset\ \boldsymbol{then}\ return\ \mathbf{Unsafe}$

$\quad V_k := \mu X.\, \alpha_k(Initial) \cup \alpha_k(\mathrm{post}(\oint_k^{\infty}(X)))$

$\quad \boldsymbol{if}\ \oint_k^{\infty}(V_k) \cap B = \emptyset\ \boldsymbol{then}\ return\ \mathbf{Safe}$

Problem: $\oint_k^{\infty}(X)$ and $\oint_k^{\infty}(V_k)$ can be infinite!

$\boldsymbol{for}\ k := \max_{b \in B_{min}} |b|\ \boldsymbol{to}\ \infty\ \boldsymbol{do}$

$\quad \boldsymbol{if}\ \alpha_k(R_k) \cap B_{min} \neq \emptyset\ \boldsymbol{then}\ return\ \mathbf{Unsafe}$

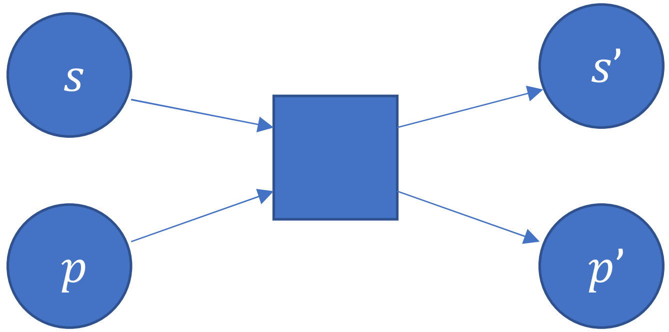
$\quad V_k := \mu X.\, \alpha_k(Initial) \cup \alpha_k(\mathrm{post}(\oint_k^{k+1}(X)))$

$\quad \boldsymbol{if}\ V_k \cap B_{min} = \emptyset\ \boldsymbol{then}\ return\ \mathbf{Safe}$

View Abstraction for Petri Nets

What we can handle:

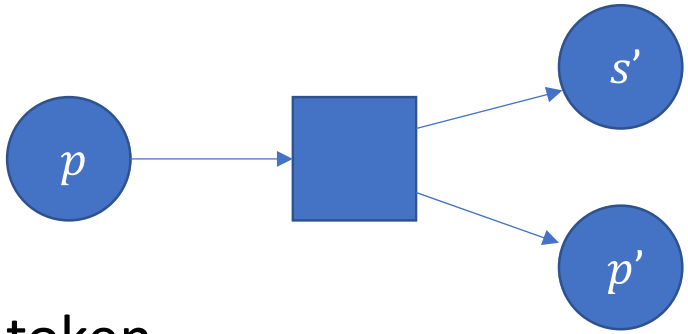
Rendez-vouz transitions: $s \to s', p \to p'$



Modify A_{post_k}:
Use $\oint_k^{k+m-1}$ instead of $\oint_k^{k+1}$
(m: Largest arity among rendez-vouz transitions)

What we can't handle:

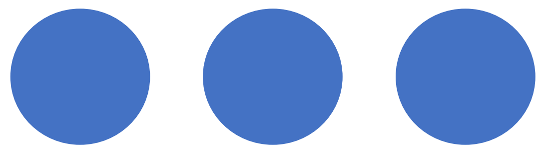
Token creation/deletion



Petri Nets without token creation/deletion ⇒ Population Protocols

Population Protocols

- Finitely many agents



Population Protocols

- Finitely many agents
- Each in one of finitely many states
- Configuration: Map states to multiplicities in the population
- States have outputs – often Boolean (here: colors)

C(Y) = 1
C(N) = 2

Population Protocols

- Finitely many agents
- Each in one of finitely many states
- Configuration: Map states to multiplicities in the population
- States have outputs – often Boolean (here: colors)
- In each step, pairwise interaction of two agents
- Transitions: give new states for agents, depending on their old states

C(Y) = 1
C(N) = 2

$$t_1 : Y, N \rightarrow y, n$$
$$t_2 : Y, n \rightarrow Y, y$$
$$t_3 : N, y \rightarrow N, n$$
$$t_4 : n, y \rightarrow y, y$$

Population Protocols

- Finitely many agents
- Each in one of finitely many states
- Configuration: Map states to multiplicities in the population
- States have outputs – often Boolean (here: colors)
- In each step, pairwise interaction of two agents
- Transitions: give new states for agents, depending on their old states
- Implicitly assume *silent* transition when none is given

C(Y) = 1
C(N) = 2

$$t_1: Y, N \rightarrow y, n$$
$$t_2: Y, n \rightarrow Y, y$$
$$t_3: N, y \rightarrow N, n$$
$$t_4: n, y \rightarrow y, y$$

Population Protocols

- Finitely many agents
- Each in one of finitely many states
- Configuration: Map states to multiplicities in the population
- States have outputs – often Boolean (here: colors)
- In each step, pairwise interaction of two agents
- Transitions: give new states for agents, depending on their old states
- Implicitly assume *silent* transition when none is given for two states
- Execution: Infinite sequence of configurations

C(Y) = 1
C(N) = 2

$t_1: Y, N \rightarrow y, n$
$t_2: Y, n \rightarrow Y, y$
$t_3: N, y \rightarrow N, n$
$t_4: n, y \rightarrow y, y$

Population Protocols

- Finitely many agents
- Each in one of finitely many states
- Configuration: Map states to multiplicities in the population
- States have outputs – often Boolean (here: colors)
- In each step, pairwise interaction of two agents
- Transitions: give new states for agents, depending on their old states
- Implicitly assume *silent* transition when none is given for two states
- Execution: Infinite sequence of configurations

C(Y) = 1
C(N) = 2

$$t_1 : Y, N \rightarrow y, n$$
$$t_2 : Y, n \rightarrow Y, y$$
$$t_3 : N, y \rightarrow N, n$$
$$t_4 : n, y \rightarrow y, y$$

Population Protocols

- Finitely many agents
- Each in one of finitely many states
- Configuration: Map states to multiplicities in the population
- States have outputs – often Boolean (here: colors)
- In each step, pairwise interaction of two agents
- Transitions: give new states for agents, depending on their old states
- Implicitly assume *silent* transition when none is given for two states
- Execution: Infinite sequence of configurations

C(y) = 1
C(N) = 1
C(n) = 1

$$t_1: Y, N \rightarrow y, n$$
$$t_2: Y, n \rightarrow Y, y$$
$$t_3: N, y \rightarrow N, n$$
$$t_4: n, y \rightarrow y, y$$

Population Protocols

- Finitely many agents
- Each in one of finitely many states
- Configuration: Map states to multiplicities in the population
- States have outputs – often Boolean (here: colors)
- In each step, pairwise interaction of two agents
- Transitions: give new states for agents, depending on their old states
- Implicitly assume *silent* transition when none is given for two states
- Execution: Infinite sequence of configurations
- Convergence: Eventually, all agents will have same output forever

C(N) = 1
C(n) = 2

$$t_1: Y, N \rightarrow y, n$$
$$t_2: Y, n \rightarrow Y, y$$
$$t_3: N, y \rightarrow N, n$$
$$t_4: n, y \rightarrow y, y$$

PO2

Population Protocols

- Computing a predicate: always converge to right output for given initial configuration eventually

- Assume fairness:
  If during the execution, C occurs infinitely often, and from C one can reach C', then C' must occur infinitely often.

- Convergence time:
  How long until all agents keep correct output forever?

Slide 32

PO2 Change Bullet point to red

Philip Offtermatt; 02.05.2019

Population Protocols

Automatic Generation of Protocols: (Blondin *et. al* 2019)
Small (polynomial number of states) protocols, generated fast (also polynomial), but: not (yet) fast convergence

Humans are needed for fast protocols!

Population Protocols

Creating (correct) population protocols is hard:

- No way of composing subfunctionalities into a bigger functionality
- No way to know for sure that a computation is done

We look for properties that are:

- Computable via View Abstraction
- Useful to help humans construct protocols

Consensus Stability

Consensus-stable set of states:
Configurations of states from the set are already in consensus and outputs cannot change

o-consensus-stable:
Set of all states with output o is consensus-stable

View Abstraction for Consensus Stability

Is $\{q_1, q_2, q_3, \dots\}$ consensus-stable for output o?

Initial: $(q_1|q_2|q_3 \dots)^+$
Bad configurations: Those that enable transitions that lead to states with output other than o

Bad configurations are upward closed $\Rightarrow$ We can use View Abstraction

Is Consensus Stability Useful?

	Protocol	True-consensus-stable	False-consensus-stable
Flock-of-Birds	Simple Flock-of-Birds	Yes	No
	Flock-of-Birds (Tower)	Yes	No
	Flock-of-Birds (Logarithmic)	Yes	No
Majority	Simple Majority	Yes	Yes
	Average-and-conquer	Yes	Yes
	Approximate Majority	Yes	Yes

Is Consensus Stability Useful?

	Protocol	True-consensus-stable	False-consensus-stable
Flock-of-Birds	Simple Flock-of-Birds	Yes	No
	Flock-of-Birds (Tower)	Yes	No
	Flock-of-Birds (Logarithmic)	Yes	No
Majority	Simple Majority	Yes	Yes
	Average-and-conquer	Yes	Yes
	Approximate Majority	Yes	Yes

Is Consensus Stability Useful?

	Protocol	True-consensus-stable	False-consensus-stable
Flock-of-Birds	Simple Flock-of-Birds	Yes	No
	Flock-of-Birds (Tower)	Yes	No
	Flock-of-Birds (Logarithmic)	Yes	No
Majority	Simple Majority	Yes	Yes
	Average-and-conquer	Yes	Yes
	Approximate Majority	Yes	Yes

Is Consensus Stability Useful?

For certain predicates, (almost) all protocols exhibit the same consensus-stability properties!

If a protocol for such a predicate has different properties:
Hint for unnecessary states or errors

Demo

https://gitlab.lrz.de/philip_offtermatt/viewabstraction-protocolassist

Thanks!

Questions?