INTERLEAVE: A Faster Symbolic Algorithm for MEC Decomposition of an MDP

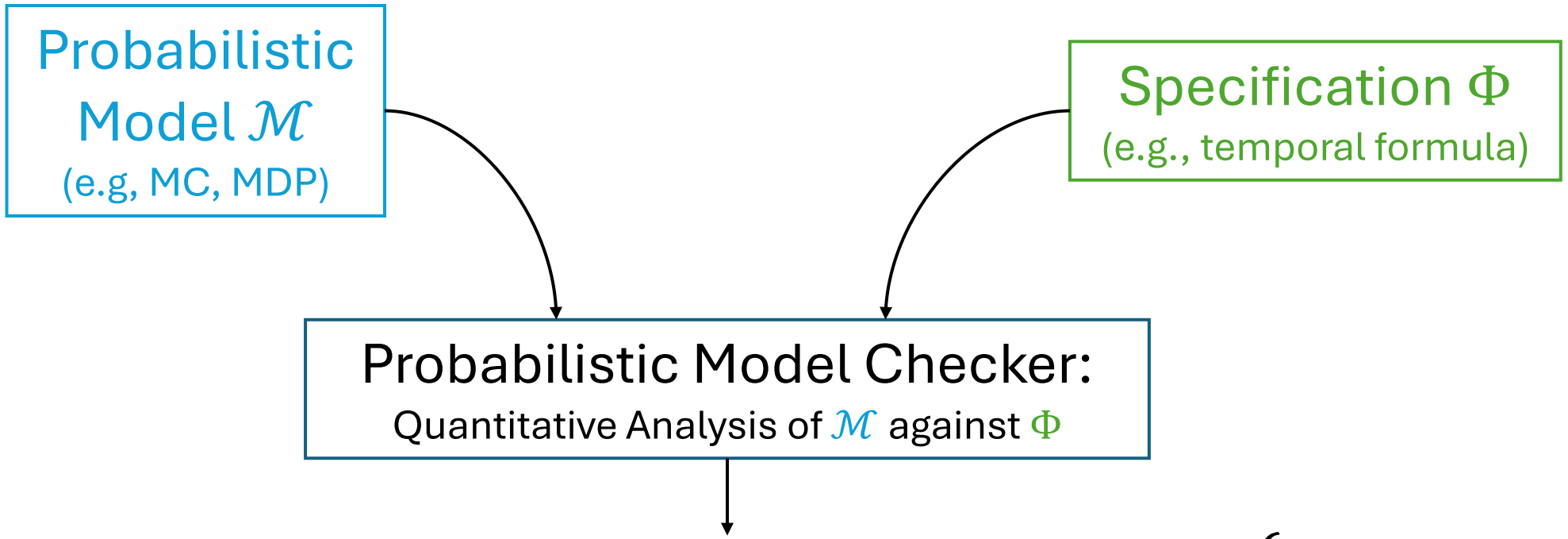
Ramneet Singh, IIT Delhi
Suguman Bansal, Georgia Tech
[CAV 2025]

A Faster Symbolic Algorithm for (MEC Decomposition of an MDP)

MEC Decomposition: why?

Fundamental to Probabilistic Model Checking

Probabilistic Model Checking: what?



Probabilistic Model Checking: why?

Evaluate and control the **reliability/performance** of:

- Randomised Algorithms
- Network Protocols
- Physical Systems
- Agent/Environment Dynamics

(CLOSE)

IEEE 802.3 CSMA/CD Protocol (csma)

A PRISM MDP model. Version 1. Last updated on 2018-10-11.

Created by IEEE and submitted by Michaela Klauck.
First presented in www.cs.bilkent.edu.tr/~tugrul/CS518/Papers/802.3-2000.pdf.
Part of the PRISM Benchmark Suite.

This is a PRISM case study [1]. The CSMA/CD (Carrier Sense, Multiple Access with Collision Detection) IEEE protocol is designed for networks with a single channel and specifies the behaviour of `N` stations with the aim of minimising simultaneous use of the channel (data collision). When a station has data

Properties

`all_before_max`	(P)	The maximum probability all stations send successfully before a collision with max backoff.
`all_before_min`	(P)	Minimum probability all stations send successfully before a collision with max backoff.
`some_before`	(P)	Minimum probability that some station eventually delivers with less than K backoffs.
`time_max`	(E)	The maximum expected time for all messages to be sent.
`time_min`	(E)	Minimum expected time for all messages to be sent.

MEC Decomposition: why?

Fundamental to Probabilistic Model Checking

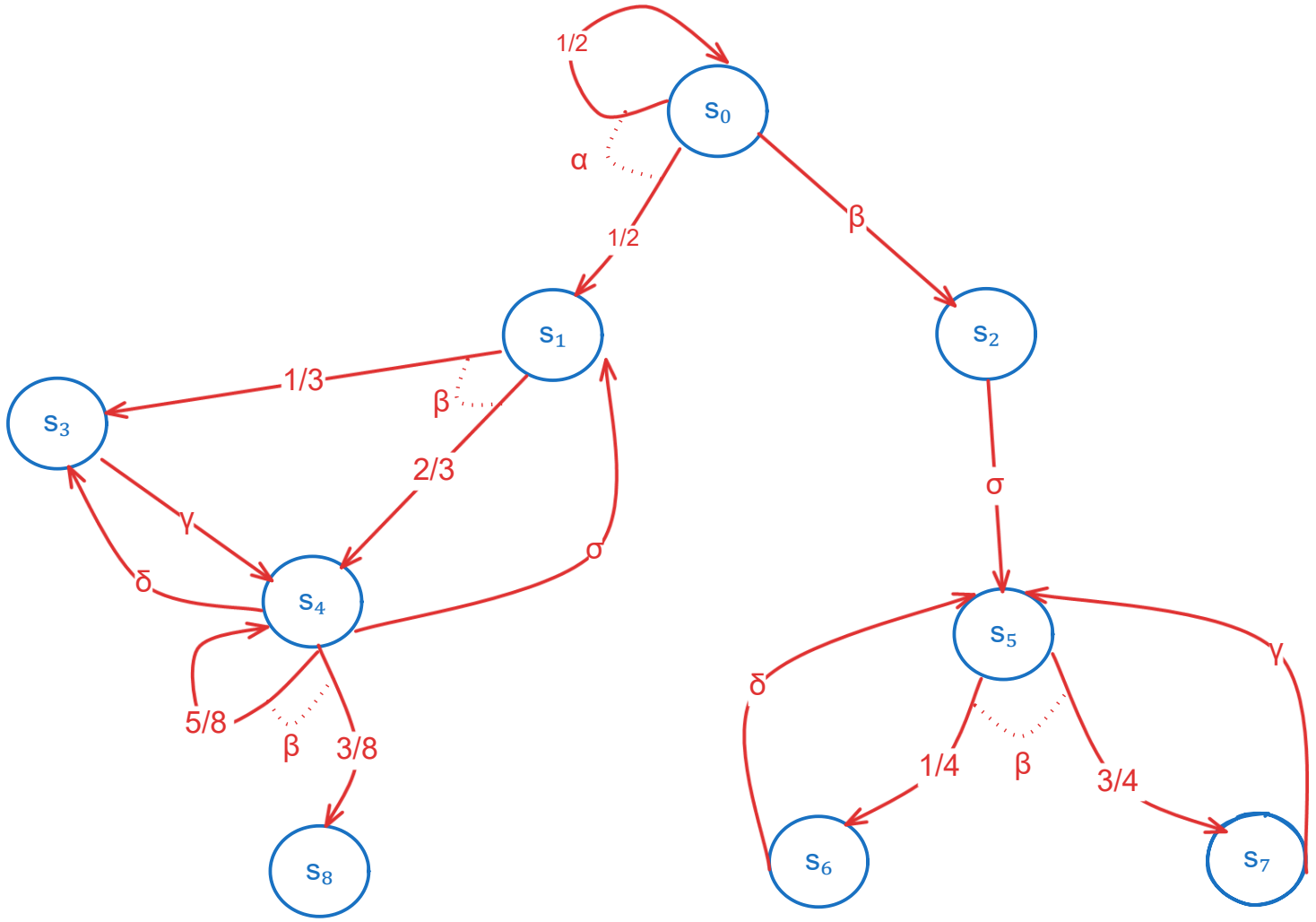
- Almost sure reachability
- Interval Iteration
- Verification of ω-regular properties
- Learning for prob. model checking
- … many more

Critical in practical tools like STORM and PRISM

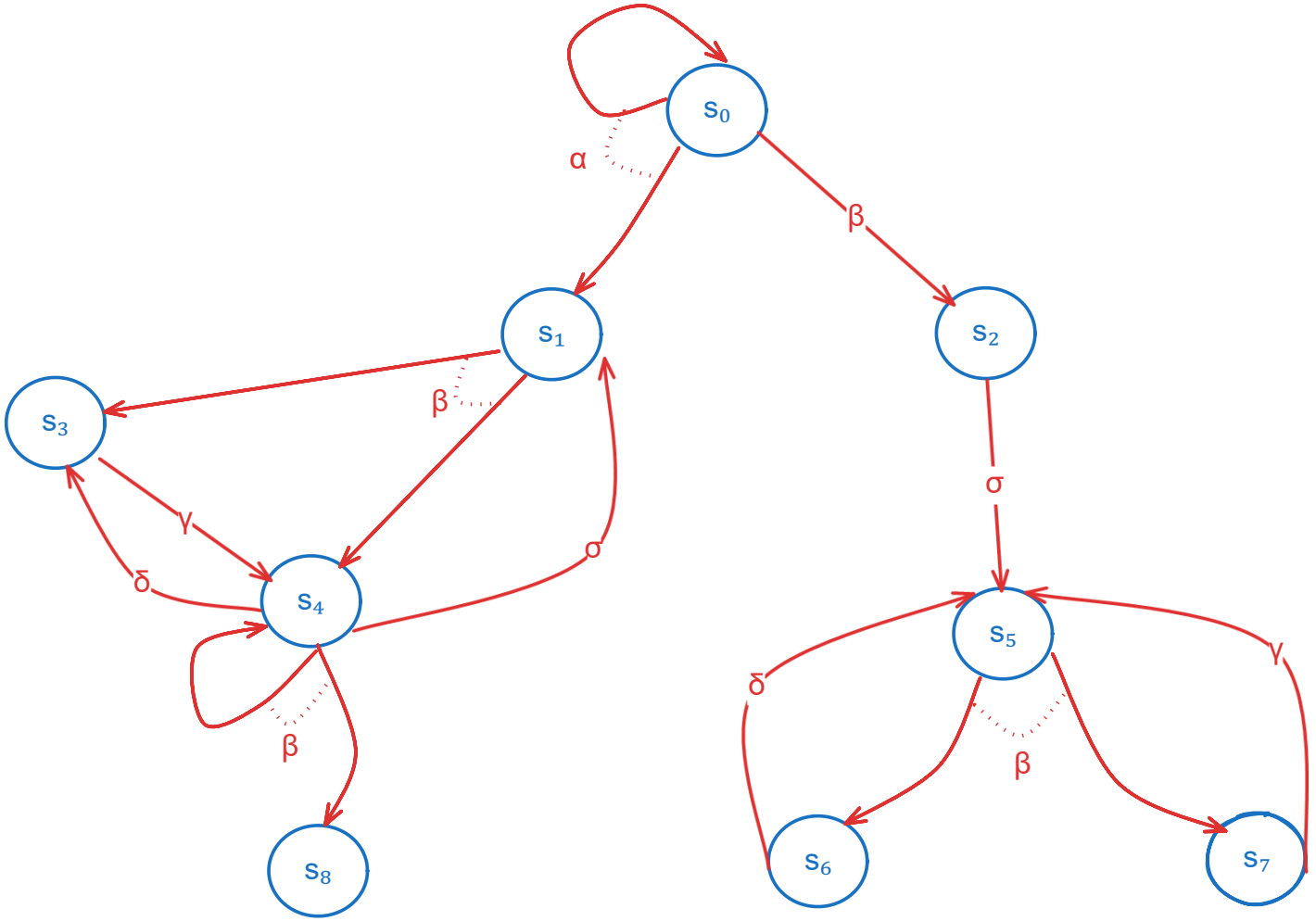
This work:

Faster Symbolic Algorithm for MEC Decomposition

Markov Decision Process (MDP)



Markov Decision Process (MDP)



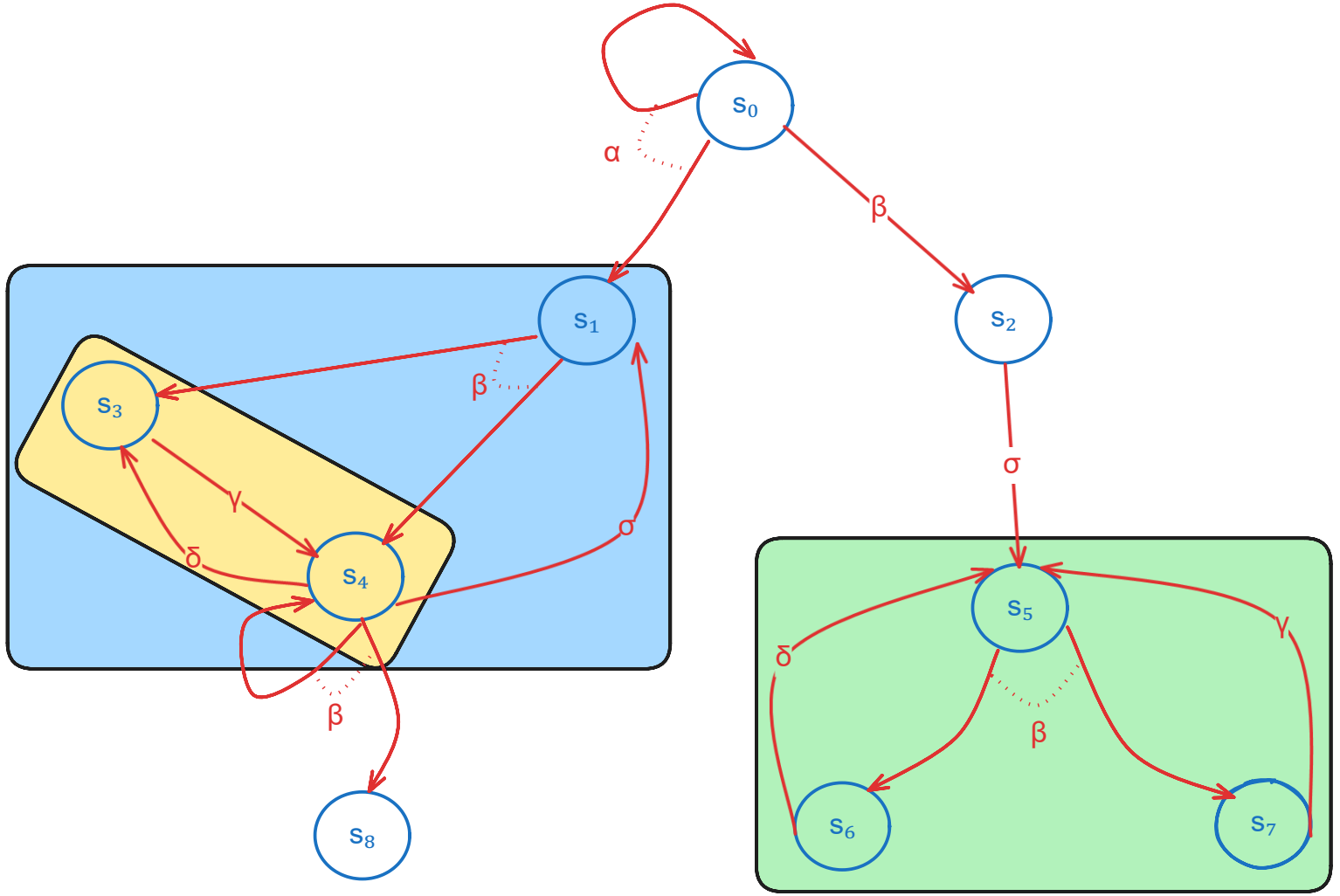
Maximal End Component (MEC)

End Component EC is a set of **state-action** pairs which is

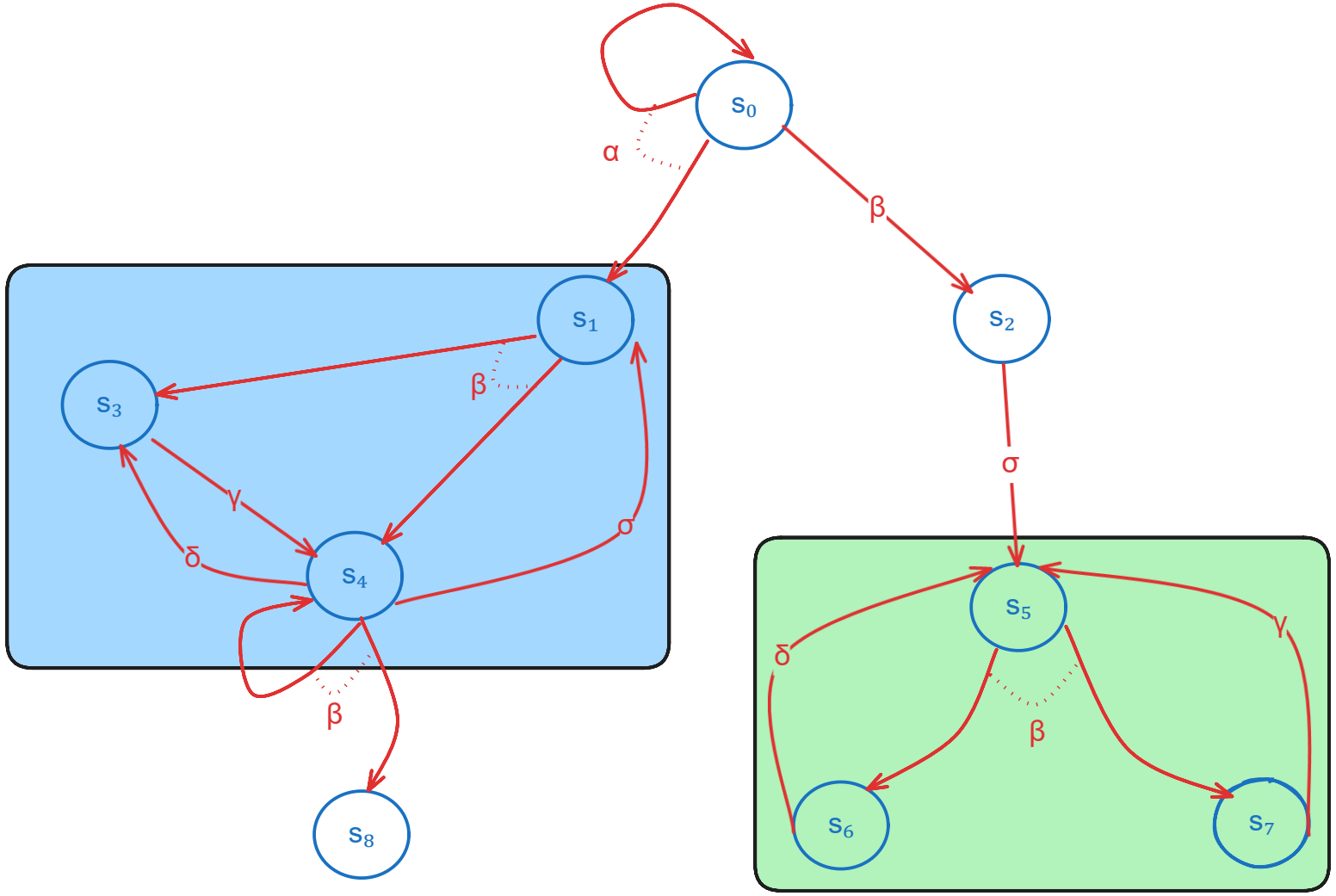
- Strongly Connected
  - For every $s, t \in EC$, there is a path from s to t using $(s_i, \alpha_i) \in EC$
- Self-contained
  - For every $(s, \alpha) \in EC$, for all $t \sim P(s, \alpha)$ we have $t \in EC$

Maximal if end-component is not present in any other end-component

End Components: example



Maximal End Components: example



Symbolic Algorithms

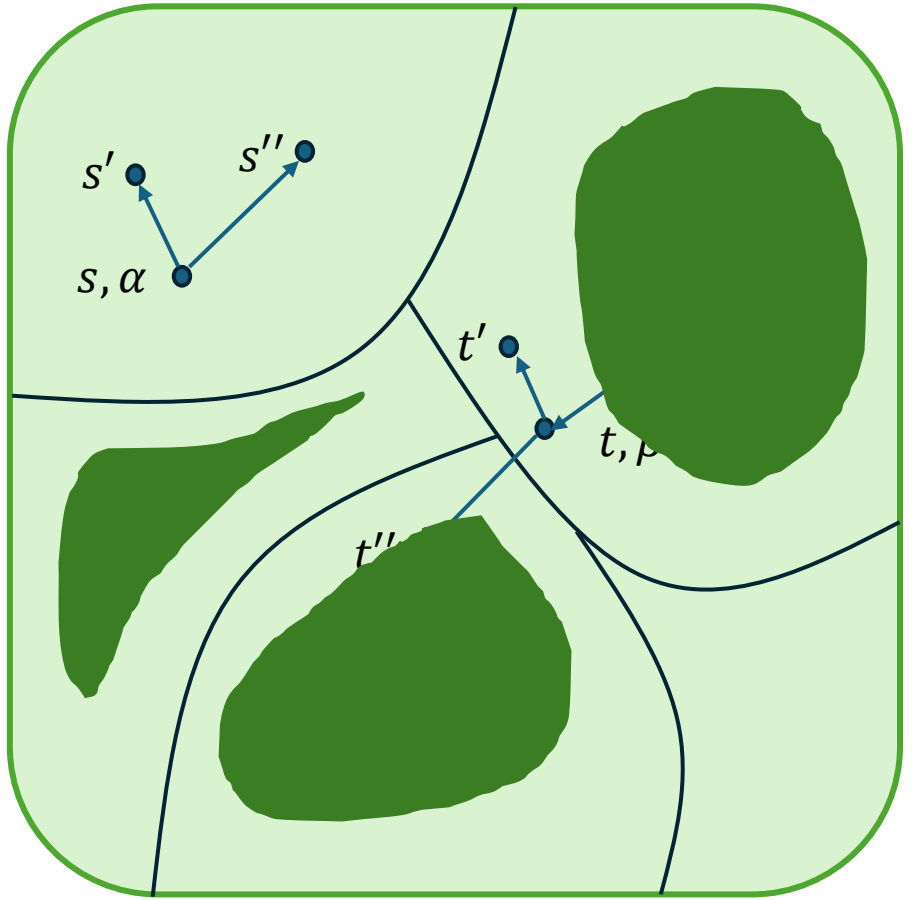
Work with *sets* (such as S, $S \times A \times S$), not individual states/actions using *symbolic operations*:

- Basic set operations ($\cap$, $\cup$, $\neg$, $\times$)
- $\exists$ (exists) to factor out some parts from a set of tuples
- Pre/Post on a set of vertices
- Pick(S) to pick an arbitrary element of S
- $|S|$ to get the cardinality of S

BASIC: State-of-the-art MEC Decomposition

[deAlfaro. PhD Thesis. 1998]

- Find all Strongly Connected Components
- Check each component for self-containment
  - If self-contained, return as an MEC
  - Else, recursively remove all non-self-contained state-action pairs
- BASIC on all non-MECs

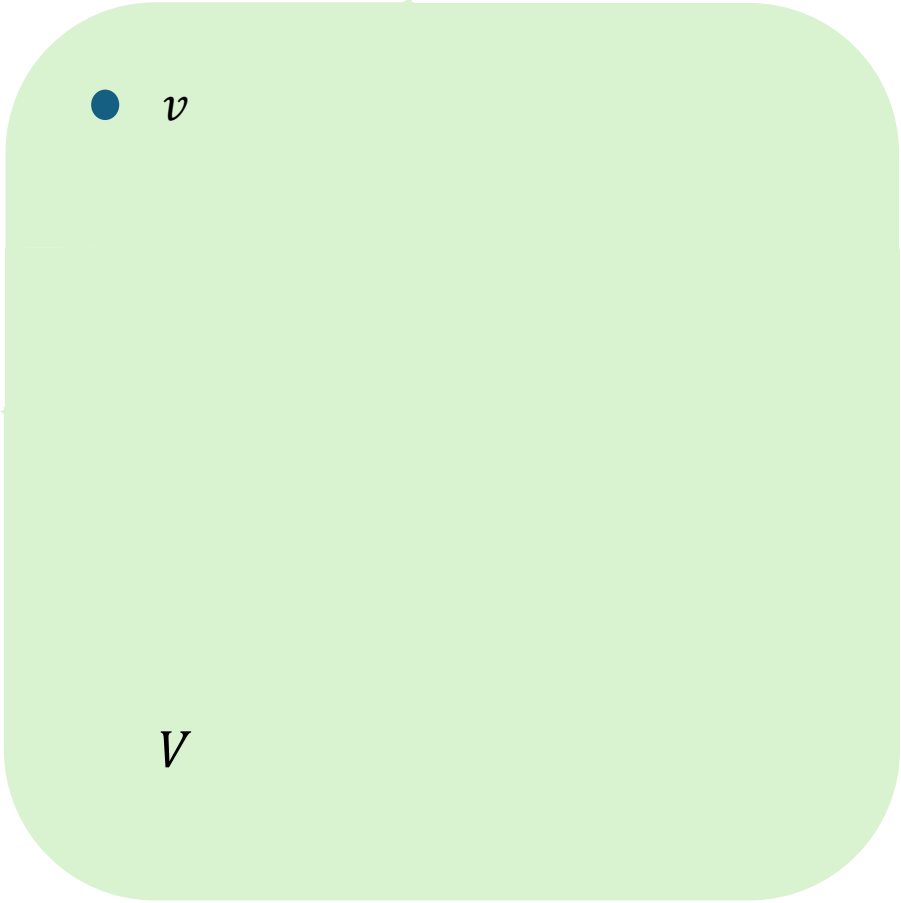


SKELETON: SCC Computation inside BASIC

[Gentilini et. al. SODA 2003]

SKELETON for SCC Computation

- Start in a state v
- Compute its Forward set F_v

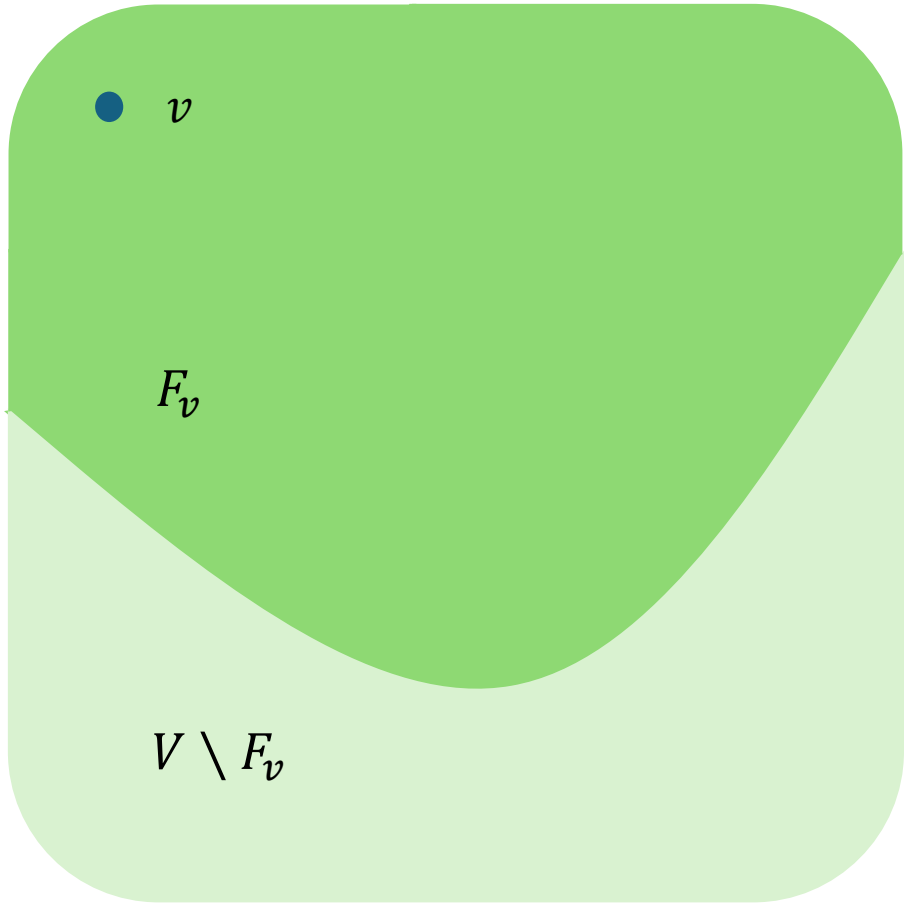


SKELETON: SCC Computation inside BASIC

[Gentilini et. al. SODA 2003]

SKELETON for SCC Computation

- Start in a state v
- Compute its *Forward set* F_v
- Compute its *Backward set* B_v



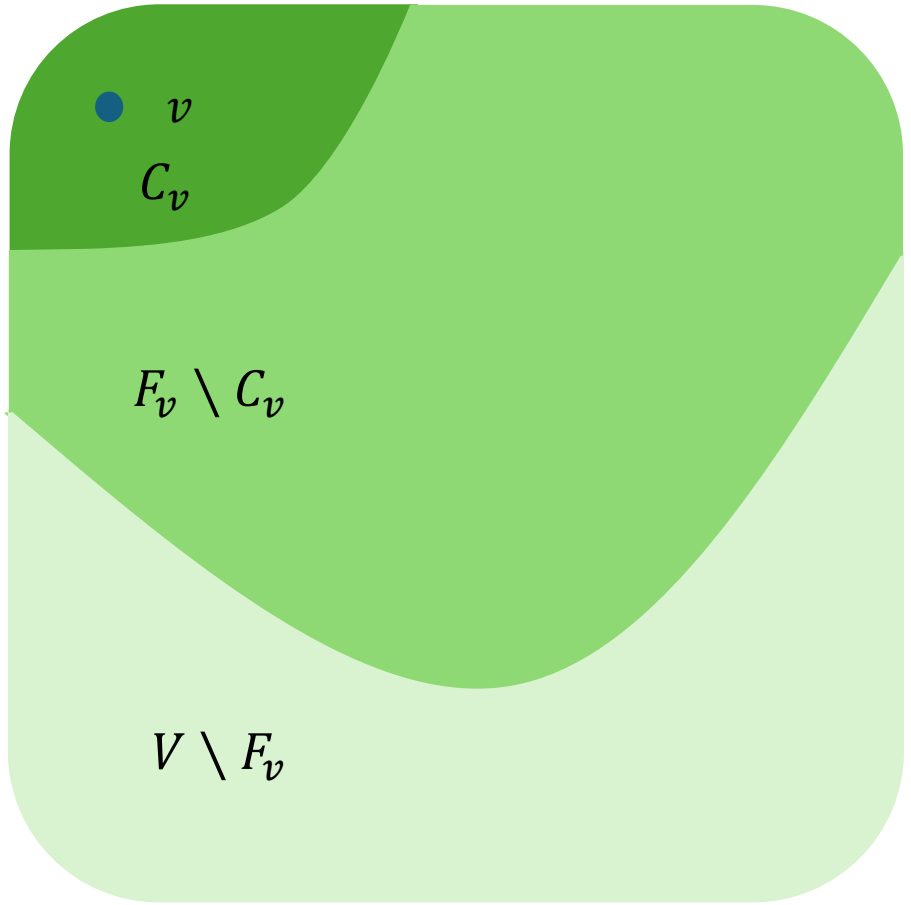
SKELETON: SCC Computation inside BASIC

[Gentilini et. al. SODA 2003]

SKELETON for SCC Computation

- Start in a state v
- Compute its *Forward set* F_v
- Compute its *Backward set* $B_v (\cap F_v)$
- SCC of v is $C_v = F_v \cap B_v$

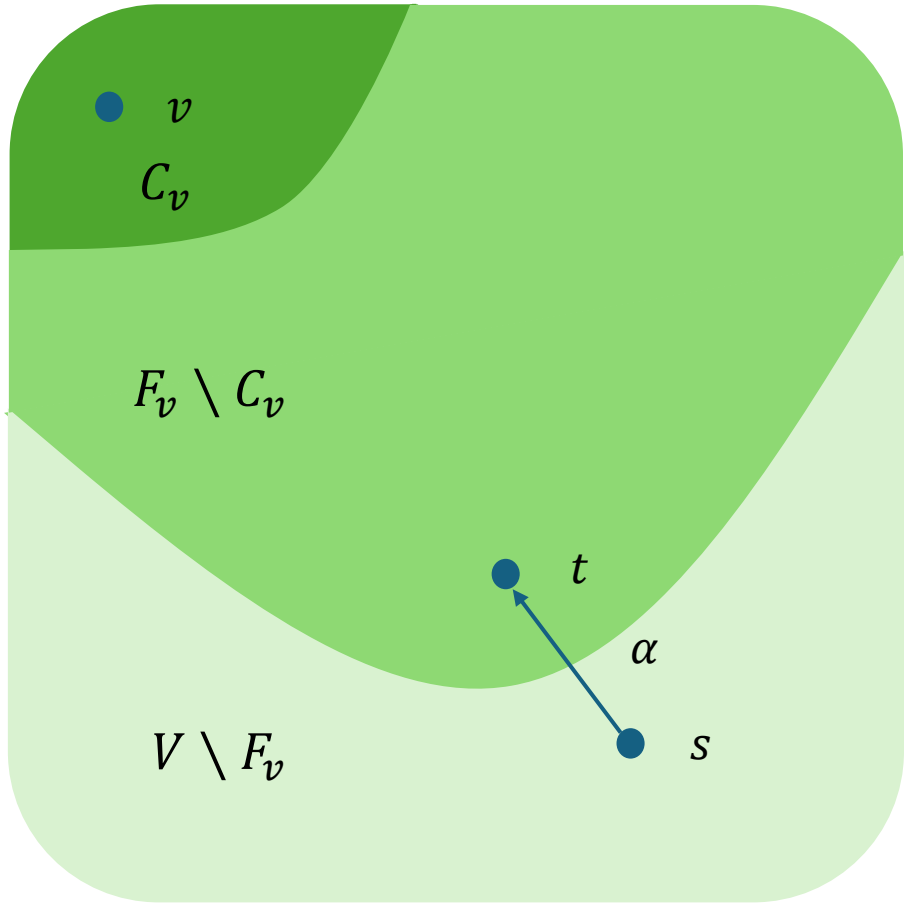
SKELETON on $F_v \setminus C_v$ and $V \setminus F_v$



Redundancy inside BASIC

Consider a transition (s, α, t) s.t. $s \in V \setminus F_v$ and $t \in F_v \setminus C_v$

Claim: (s, α) belongs to no MEC

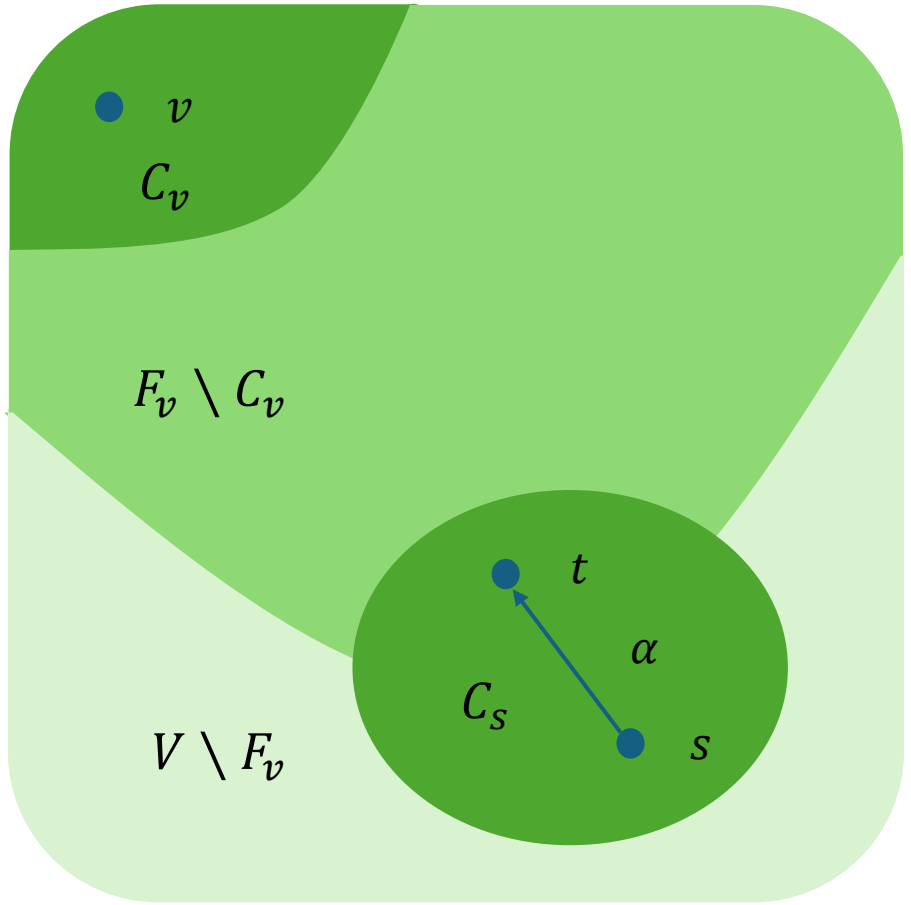


Redundancy inside BASIC

Consider a transition (s, α, t) s.t. $s \in V \setminus F_v$ and $t \in F_v \setminus C_v$

Claim: (s, α) belongs to no MEC

Suppose (s, α) is in an MEC,
then SCC of s, say C_s,
is present across $V \setminus F_v$ and $F_v \setminus C_v$

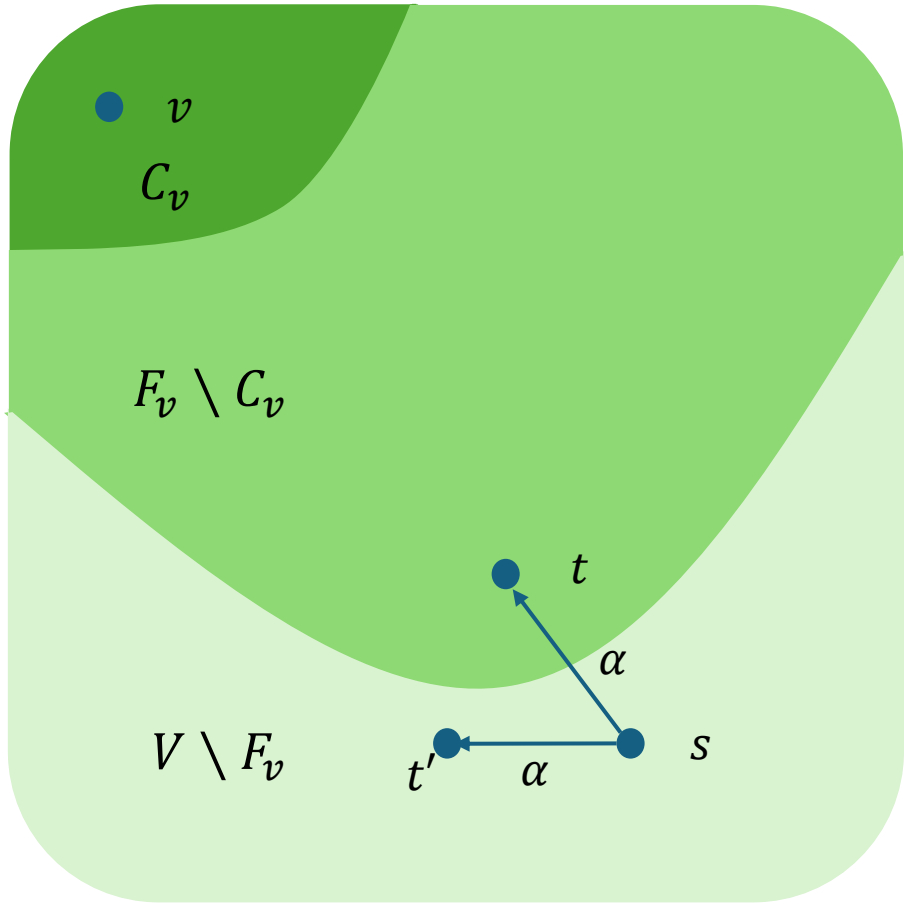


Redundancy inside BASIC

Consider a transition (s, α, t) s.t. $s \in V \setminus F_v$ and $t \in F_v \setminus C_v$

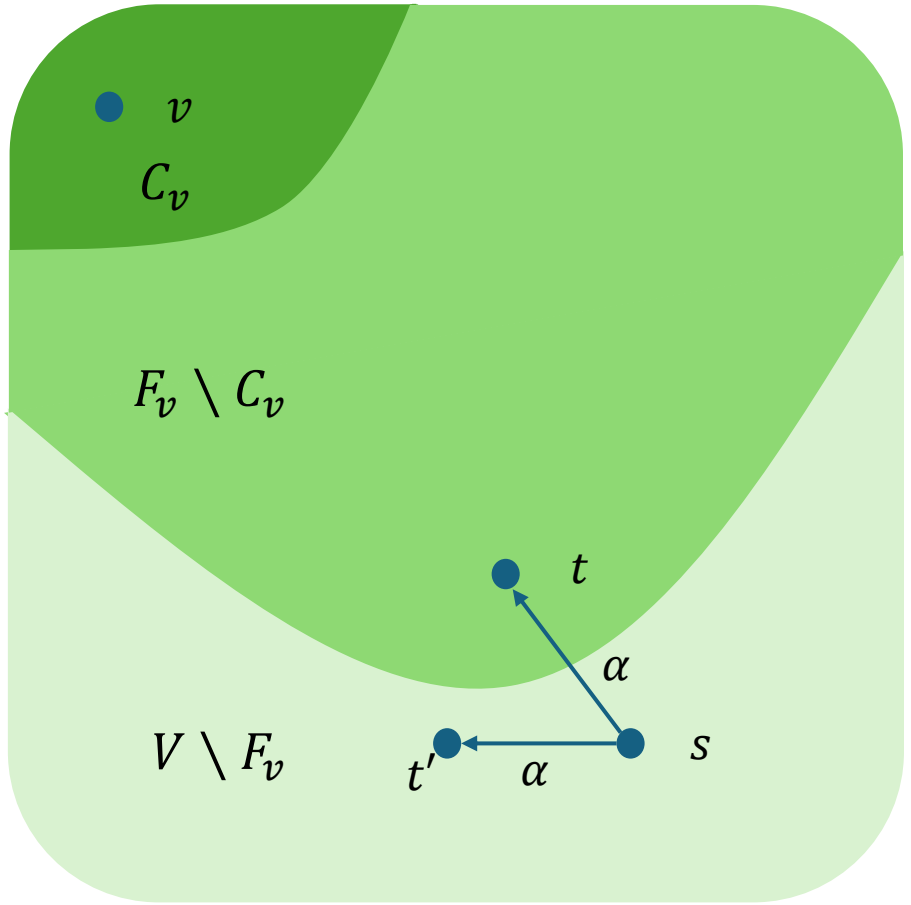
Claim: (s, α) belongs to no MEC

Redundancy because the unnecessary edge (s, α, t') will be processed multiple times for MEC computation from now on



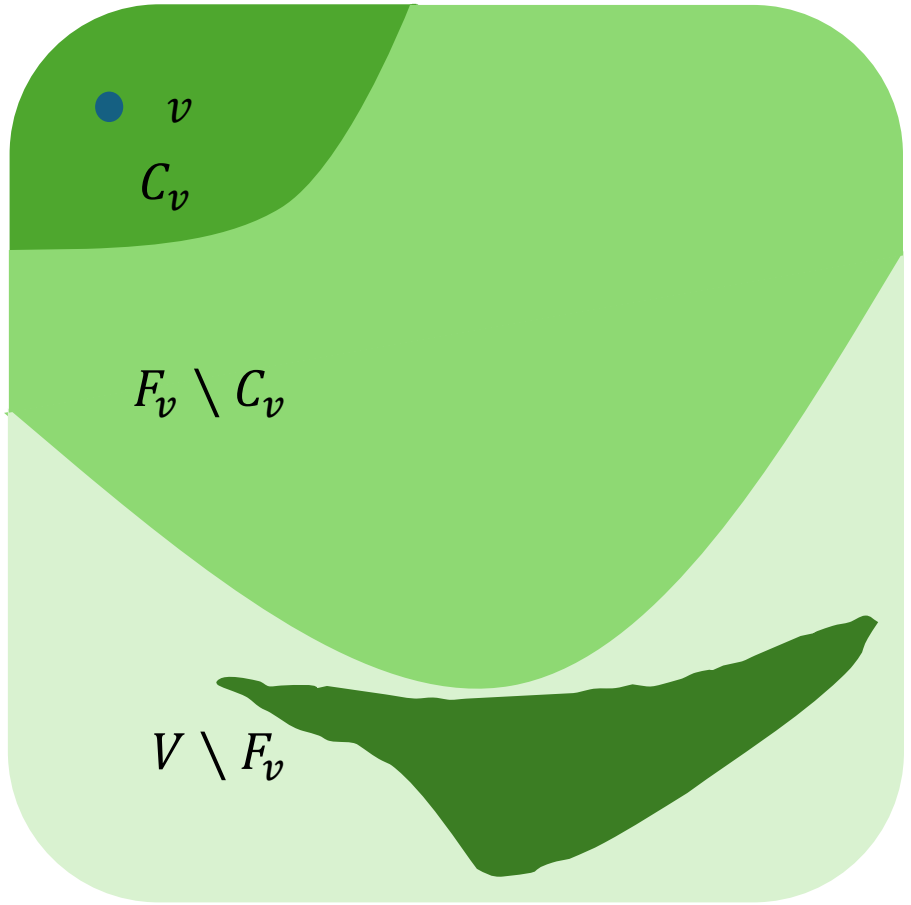
INTERLEAVE: Optimizations

- Eliminate redundancy: Recursively remove all non-self-contained state-action pairs from $V \setminus F_v$



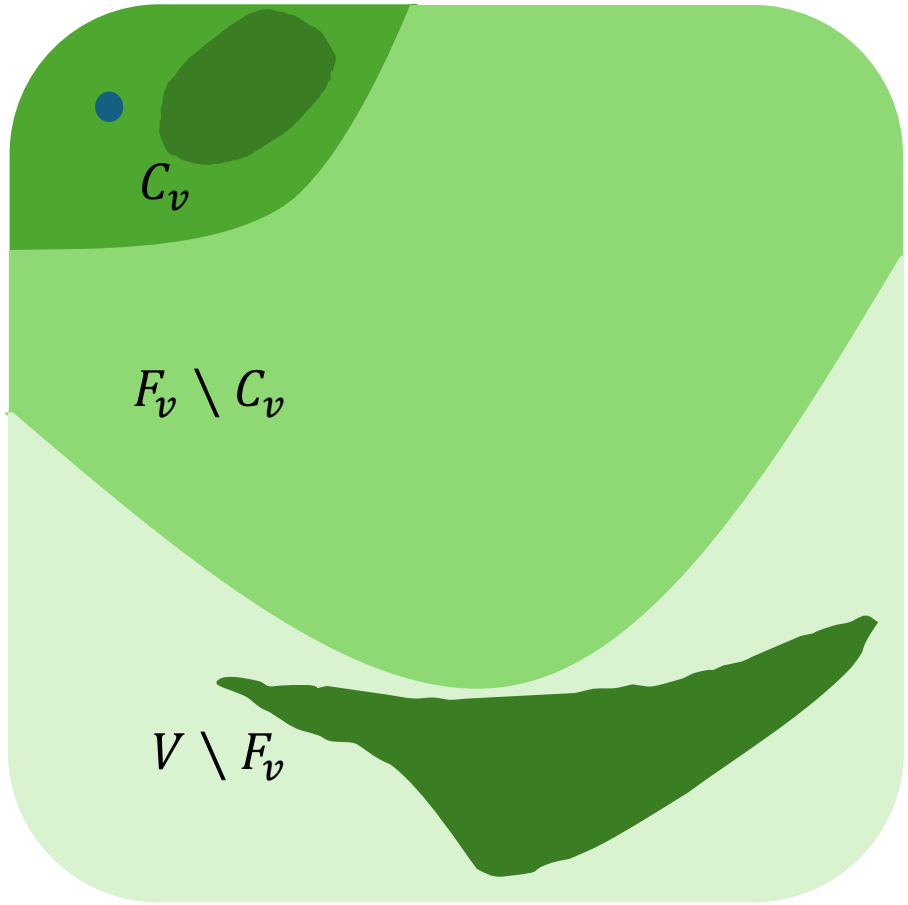
INTERLEAVE: Optimizations

- Eliminate redundancy: Recursively remove all non-self-contained state-action pairs from $V \setminus F_v$
- Claim: All state-action pairs in $F_v \setminus C_v$ are self-contained



INTERLEAVE: Full Algorithm

- From an arbitrary state v, identify C_v, $F_v \setminus C_v$, and $V \setminus F_v$
- If C_v is self-contained, return as MEC
- Else, INTERLEAVE(Self-Contained(C_v))
- INTERLEAVE($F_v \setminus C_v$)
- INTERLEAVE(Self-Contained($V \setminus F_v$))

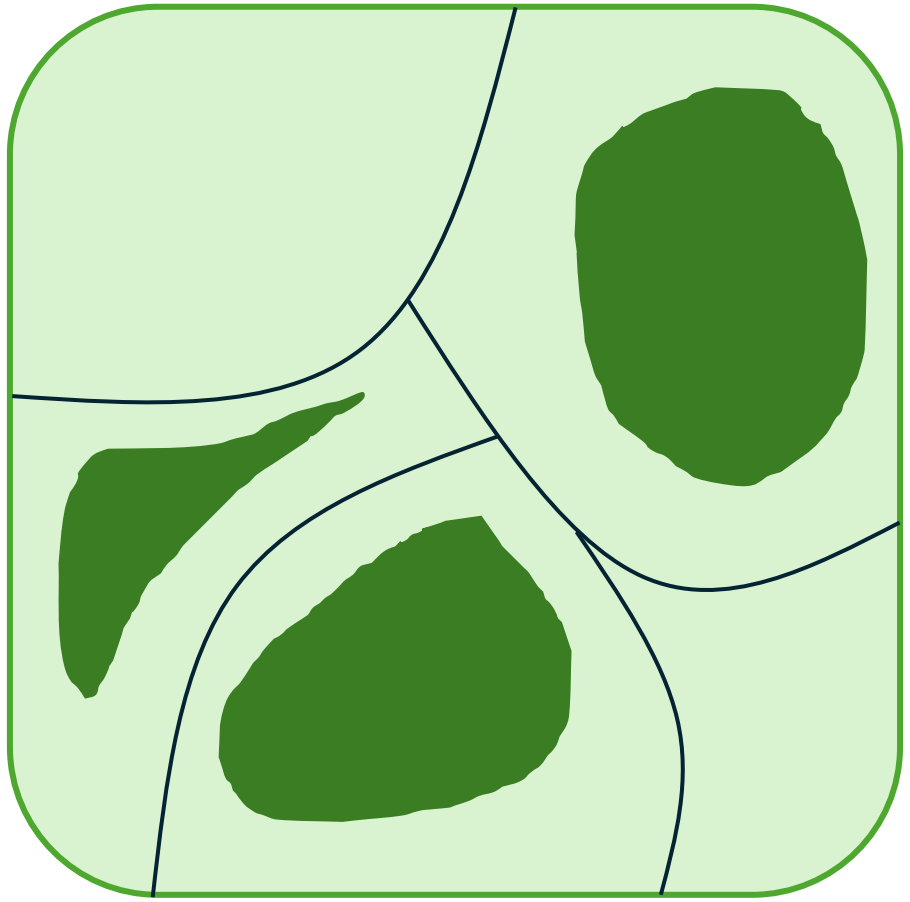


BASIC:

1. All SCCs

2. Self-containment

3. Recurse

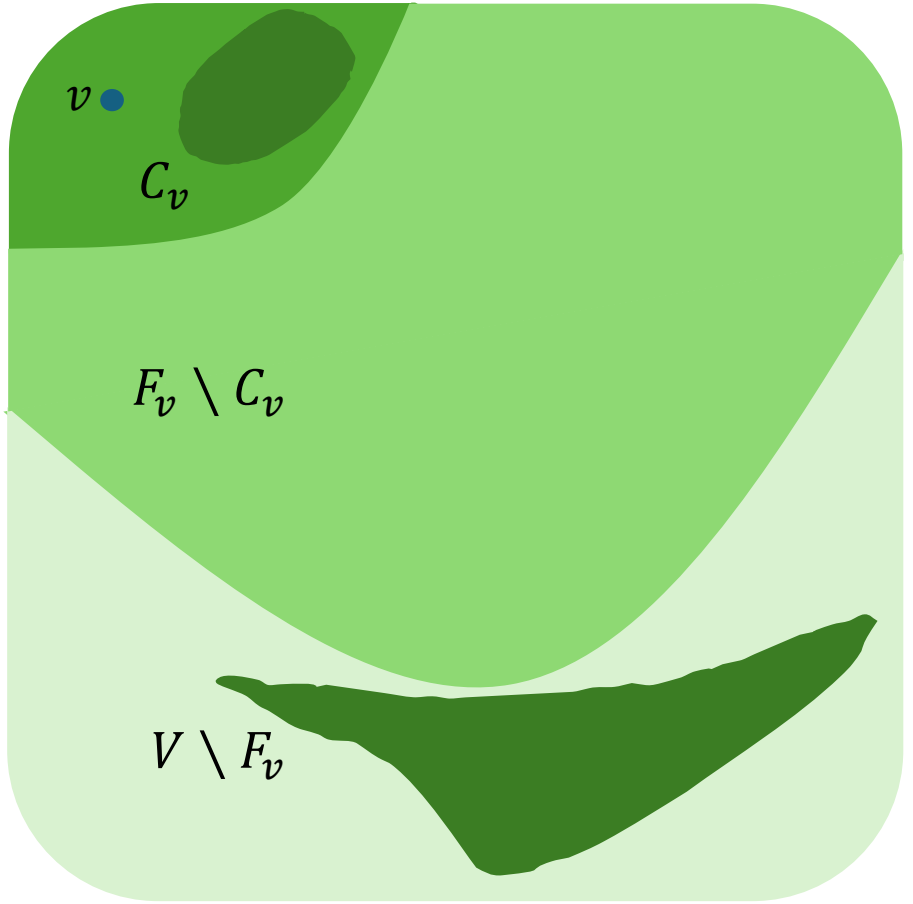


INTERLEAVE:

1. First SCC

2. Self-containment

3. Recurse



Complexity

Algorithm	Symbolic Operations	Symbolic Space
BASIC [deAlfaro. PhD Thesis. 1998]	$O(n^2)$	$O(\log n)$
LOCKSTEP [Chatterjee et. al. CAV 2018]	$O(n\sqrt{m})$	$O(\sqrt{m})$
INTERLEAVE (Ours)	$\boldsymbol{O(n^2)}$	$\boldsymbol{O(\log n)}$

n is number of states
m is number of edges

Empirical Evaluation

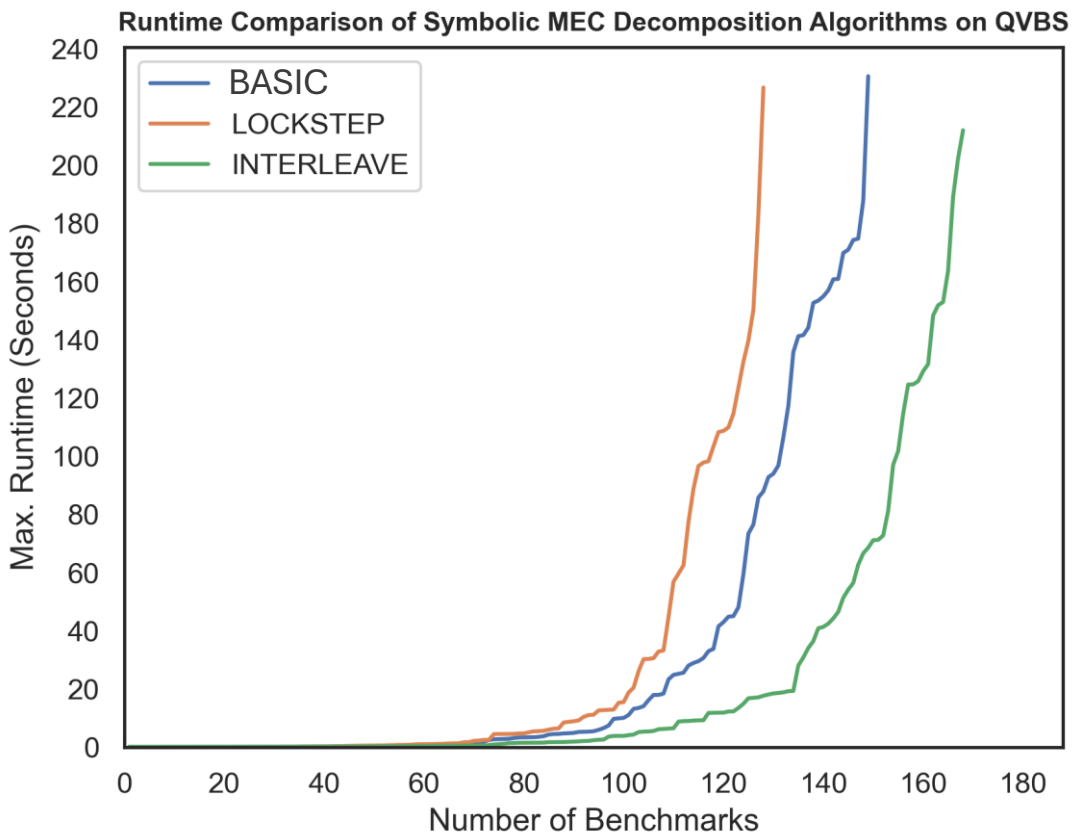
Implementation Details

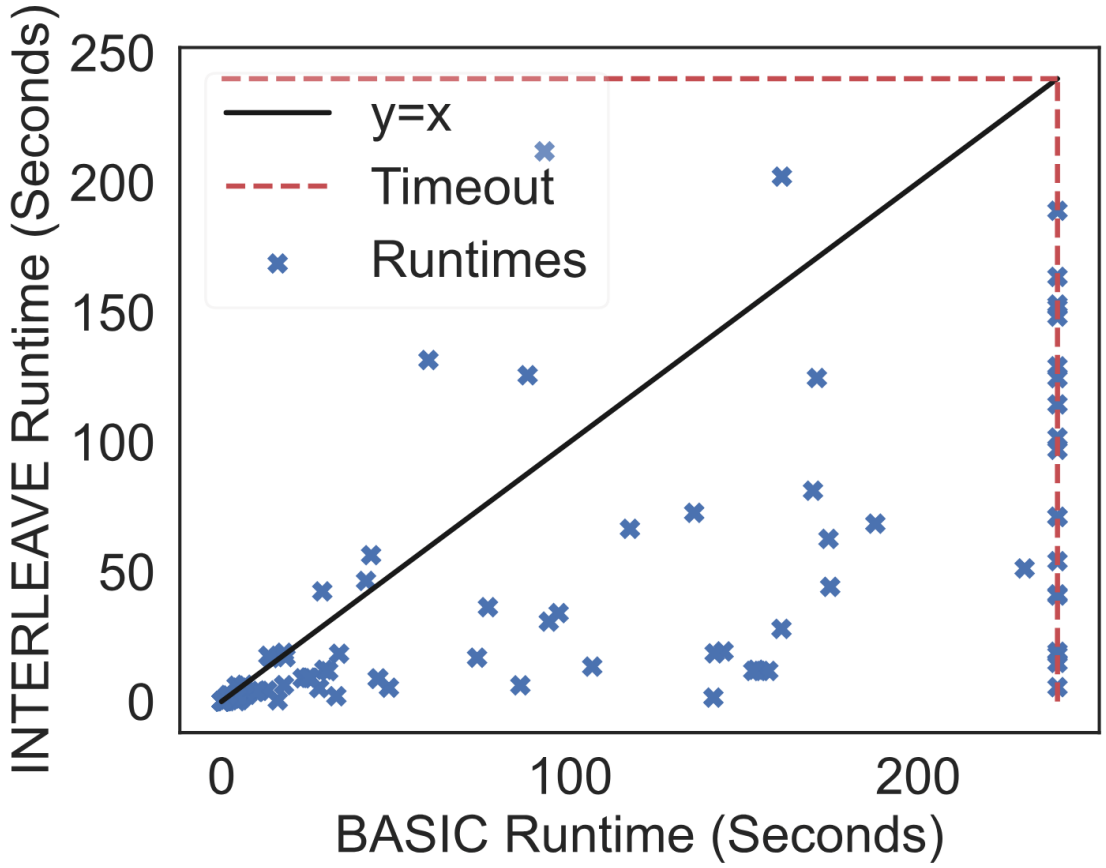
- Implemented inside STORM Model Checker
- Prior implementations of BASIC and LOCKSTEP available
- Ensures fair comparison of tools as all share commonalities

Benchmark Suite:

- Quantitative Verification Benchmark Suite (QVBS) [Hartmanns et. al. TACAS 2019]

Runtime Performance





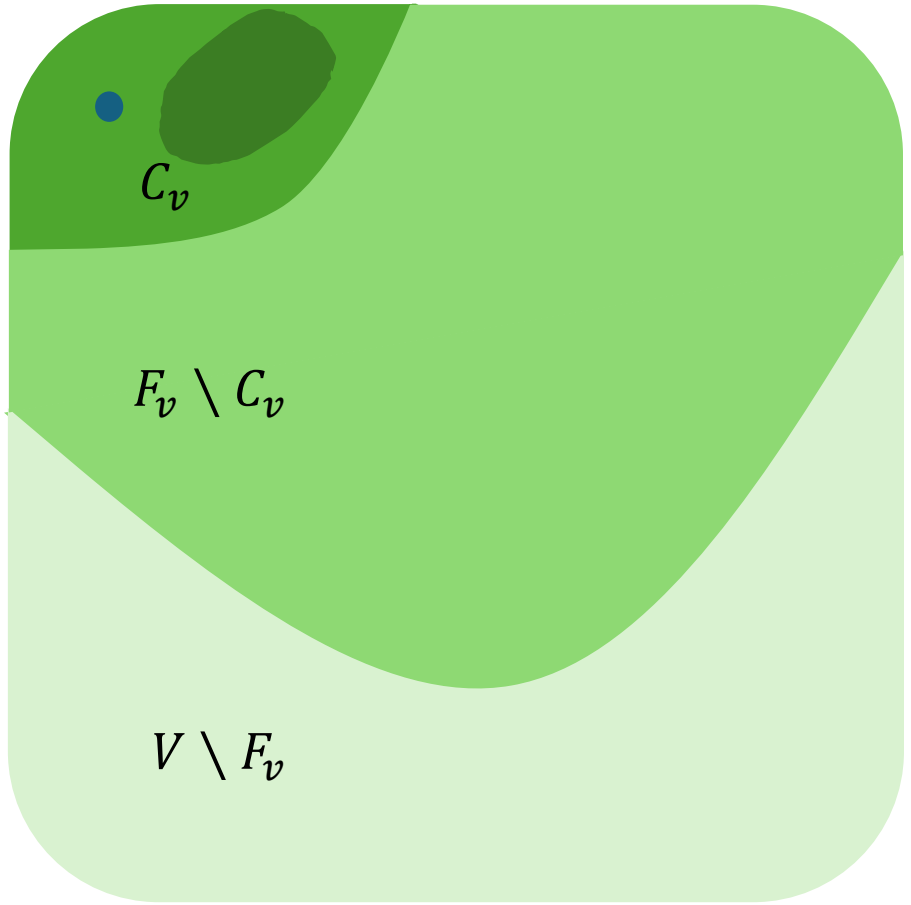
3.81 times faster than BASIC on average

INTERLEAVE: In a nutshell

- Simple, easy to implement algorithm
- Same complexity, Faster on standardized benchmark

Future work

- Is this useful for LLMs?
- *Compositional* Reinforcement Learning
- Parallel implementations



Links

- Masters Thesis
- GitHub Repository
- CAV'25 Paper
- Desk 1036