Complementing Emerson–Lei Elevator Automata

Ondrej Alexaj[1] Vojtěch Havlena[1] Ondřej Lengál[1] **Yong Li**[2] Nicolas Mazzocchi[3]

[1]Brno University of Technology, Czech Republic
[2]Institute of Software Chinese Academy of Sciences, China
[3]Slovak University of Technology in Bratislava, Slovakia

Automata complementation is fundamental

Given an ω-automaton $\mathcal{A}$, complementation constructs $\overline{\mathcal{A}}$ over the same alphabet with the **complement language**:

$$\mathcal{L}(\overline{\mathcal{A}}) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A}) \qquad \text{or, equivalently,} \qquad w \in \mathcal{L}(\overline{\mathcal{A}}) \iff w \notin \mathcal{L}(\mathcal{A}).$$

Complementation is **fundamental** in many applications:

Theorem Proving
Pecan Theorem Prover

Termination Checking
Ultimate Automizer

Model Checking
Spot Automata Library

[Oei et al. '21; Chen et al. '18; Duret-Lutz et al. '22]

Why Emerson–Lei elevator automata?

Emerson–Lei elevator automata are as *expressive as* Büchi automata.

Rich acceptance condition

Arbitrary Boolean formula over **repeated** and **finite** reachability obligations

Compact representation

More acceptance obligations instead of control states

Rich condition ⇒ fewer states

Motivation

Smaller state space to work with in complementation

[Müller–Sickert '17; Boker '18]

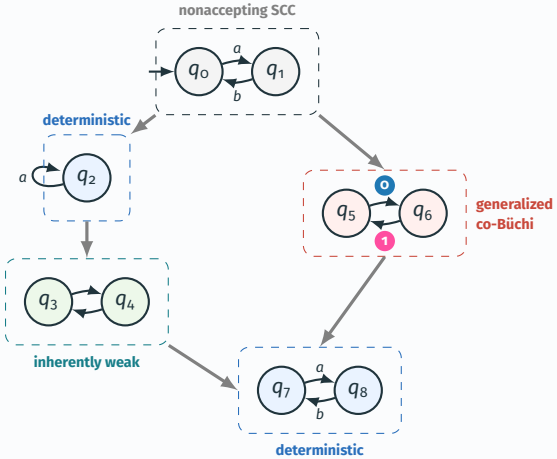
Why Emerson–Lei elevator automata?

- An infinite run **stays in one SCC**
- Decompose automaton into **strongly connected components** (SCCs)
- Practical automata often have **elevator structure**

Motivation

Leverage **SCC structural properties**
Complement each SCC **independently**
with its specialized construction

SCC decomposition for elevator structure



[Li et al. '22; Havlena et al. '23; Alexaj et al. '26]

Our contributions

1. **Novel** complementation algorithm for **Emerson–Lei elevator automata**, exploiting different SCC structures

2. **Better state-complexity bounds** for complementing automata with **rich acceptance conditions**

3. Experiments show **smaller complement Büchi automata** than Spot

compact Emerson–Lei automaton + elevator SCC structure ⇒ smaller Büchi complement

Emerson–Lei automata and acceptance

An automaton with the Emerson–Lei condition generated by

$$\alpha ::= \mathsf{true} \mid \mathsf{false} \mid \mathsf{Inf}(c) \mid \mathsf{Fin}(c) \mid (\alpha \wedge \alpha) \mid (\alpha \vee \alpha).$$

A run accepts iff its **infinite color set** sequence satisfies α.

Semantics

- $\mathsf{Inf}(c)$: color c occurs *infinitely* often
- $\mathsf{Fin}(c)$: color c occurs *finitely* often

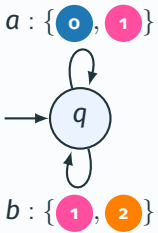
Examples

Büchi:	$\mathsf{Inf}(\mathrm{O})$
co-Büchi:	$\mathsf{Fin}(\mathrm{O})$
Rabin pair:	$\mathsf{Fin}(\mathrm{B}) \wedge \mathsf{Inf}(\mathrm{G})$
Streett pair:	$\mathsf{Inf}(\mathrm{G}) \vee \mathsf{Fin}(\mathrm{B})$

[Emerson–Lei '87; Boker '18]

Running example: one generalized Rabin pair

$\varphi = \mathrm{Fin}(0) \wedge \mathrm{Inf}(1) \wedge \mathrm{Inf}(2)$.



Trivial SCC structure; **colors determine acceptance**

$w_+ = a\,b^\omega$ is accepted

0 occurs once, while 1 and 2 occur infinitely often:

$$\underbrace{\{0,1\}}_{a}\ \underbrace{\{1,2\}\{1,2\}\cdots}_{b^\omega} \models \varphi$$

$w_- = (ab)^\omega$ is rejected

0 recurs forever, so $\mathrm{Fin}(0)$ fails despite both Inf obligations

To complement: $\neg\varphi = \psi = \mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Fin}(2)$

[Chatterjee et al. '13]

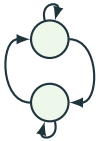
Elevator structure

An **Emerson–Lei elevator automaton** (ELEA) has only elevator **accepting** SCCs:

inherently weak, **generalized co-Büchi** or **deterministic**

Inherently weak

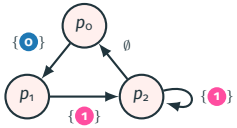
φ



all cycles in the SCC are accepting

Generalized co-Büchi

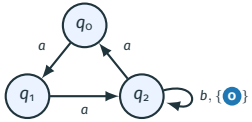
$(\mathrm{Fin}(\mathbf{O}) \vee \mathrm{Fin}(\mathbf{1})) \wedge \mathrm{Fin}(\mathbf{B})$



accepting runs eventually *avoid* ≥ 1 relevant color

Deterministic

φ



all runs are *deterministic*

Main complementation idea

Goal: accept words whose runs satisfy $\psi = \neg\varphi$

Algorithm: Identify **all non-accepting** runs in SCCs or all runs satisfying ψ

Inherently weak

φ

Easy: Similar to ensure **reachability**;
Identify all runs in SCC that are either **finite** or **exiting** the SCC

Generalized co-Büchi

$(\mathrm{Fin}(\mathbf{0}) \vee \mathrm{Fin}(\mathbf{1})) \wedge \mathrm{Fin}(\mathbf{B})$

Easy: Similar to ensure **fairness**;
Identify all runs in SCC that visit all relevant colors **infinitely often**

Deterministic

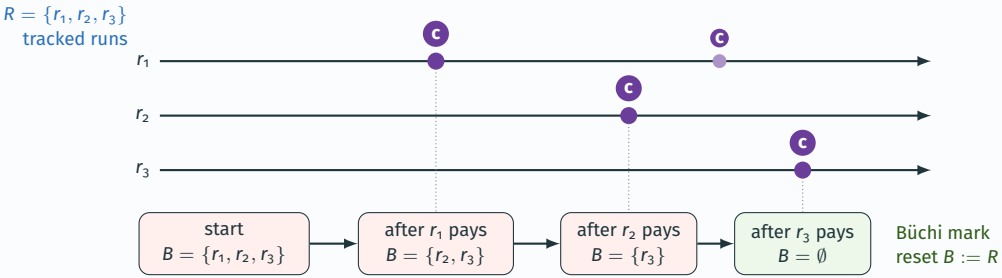
φ

Hard: Need to consider **complex** acceptance condition

Breakpoint construction: two easy SCC types

For the condition φ, **all runs** staying in the SCC must satisfy $\psi = \neg\varphi$.

Breakpoint construction (R, B): R (for *Runs*) tracks **all current runs**, while B (for *Breakpoint*) records those that still **owe a visit** to color c



One completed round: $B = \emptyset$ means **all runs** in R have visited c once

Miyano–Hayashi breakpoint construction [Miyano–Hayashi '84].

Breakpoint construction: two easy SCC types

For the condition φ, **all runs** staying in the SCC must satisfy $\psi = \neg\varphi$

A **breakpoint** construction (R, B) suffices: R for *Runs* and B for *Breakpoint*

Inherently weak SCC

All runs need **discontinue** or **exit**

- B records those still **owing an exit**
- $B = \emptyset$ **discharges** the old obligations

Generalized co-Büchi SCC

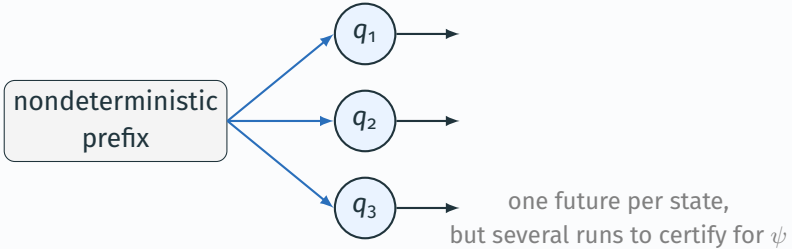
All runs satisfy $\mathsf{Inf}(c_1), \ldots, \mathsf{Inf}(c_k)$

- z selects color c_z **currently checked**
- B *stores* runs **owing visit to** c_z
- $B = \emptyset$ **advances** z in cyclic manner

Miyano–Hayashi breakpoint construction [Miyano–Hayashi '84].

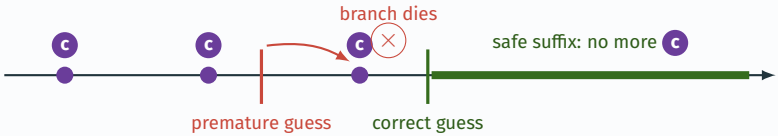
The difficult case: deterministic SCCs

- **Many deterministic** runs enter into the SCC at different time
- Formula $\psi = \neg\varphi$ may contain **arbitrary mixtures** of Fin, Inf, $\wedge$, and $\vee$



Main idea: guess when a run satisfies $\psi = \neg\varphi$

For $\psi = \mathsf{Fin}(\boxed{c})$, **guess when** runs do not see $\boxed{c}$ any more and then **verify**



A pair (C, S) suffices to deal with $\psi = \mathsf{Fin}(\boxed{c})$:

- C: collects **deterministic** runs still to check, and
- S: runs guessed to see **no** more $\boxed{c}$

Easy to verify guesses since runs are **deterministic** and **discontinue** if wrong

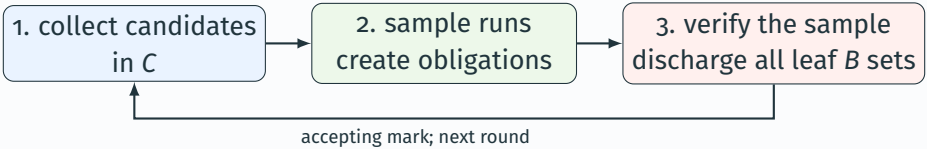
NCSB complementation: [Blahoudek et al. '16]

Main idea: sample runs, then verify them

For Emerson–Lei condition φ, **guess when** a run satisfy $\psi = \neg\varphi$ and then **verify**

Our construction (C, M_ψ) to deal with abitrary ψ:

- C: collects the **deterministic** runs, and
- M_ψ: run **guesses** organized with the syntax **tree** of ψ
  - Inf(c) leaf: runs in breakpoint (R, B)
  - Fin(c) leaf: runs in safe set S



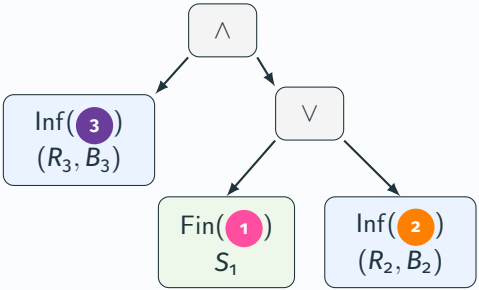
Tree structure M_ψ

For a formula such as

$$\psi = \mathrm{Inf}(3) \wedge (\mathrm{Fin}(1) \vee \mathrm{Inf}(2)),$$

we build a tree whose leaves store run sets:

- $\mathrm{Inf}(k)$ leaf stores (R, B);
- $\mathrm{Fin}(\ell)$ leaf stores S



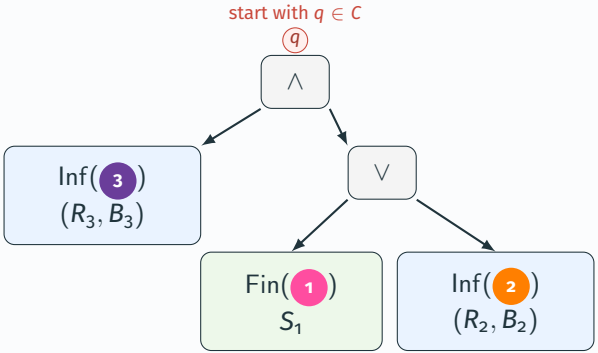
At $\wedge$, copy each run to both children. At $\vee$, partition runs between them

Sampling a run into the tree

For

$$\psi = \mathrm{Inf}(3) \wedge (\mathrm{Fin}(1) \vee \mathrm{Inf}(2)),$$

sample a new run $q \in C$.



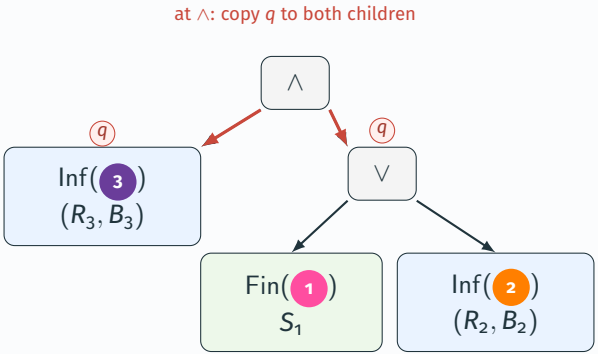
q is waiting to be sampled from C.

Sampling a run into the tree

For

$$\psi = \mathrm{Inf}(3) \wedge (\mathrm{Fin}(1) \vee \mathrm{Inf}(2)),$$

sample a new run $q \in C$.



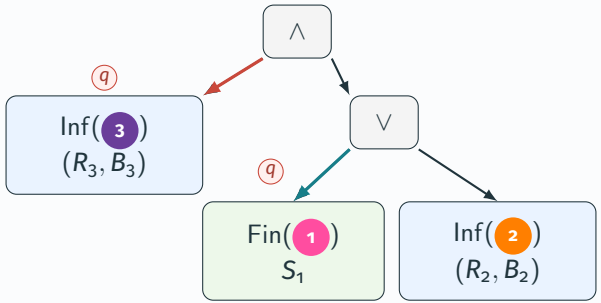
Conjunction means both obligations must be satisfied.

Sampling a run into the tree

For

$$\psi = \mathsf{Inf}(3) \wedge (\mathsf{Fin}(1) \vee \mathsf{Inf}(2)),$$

sample a new run $q \in C$.



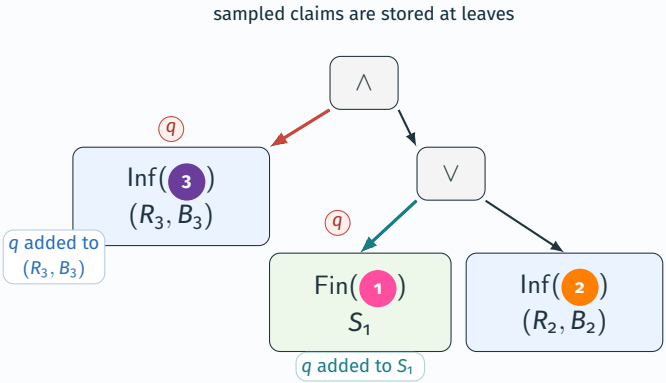
Disjunction means one satisfying branch is chosen.

Sampling a run into the tree

For

$$\psi = \mathrm{Inf}(3) \wedge (\mathrm{Fin}(1) \vee \mathrm{Inf}(2)),$$

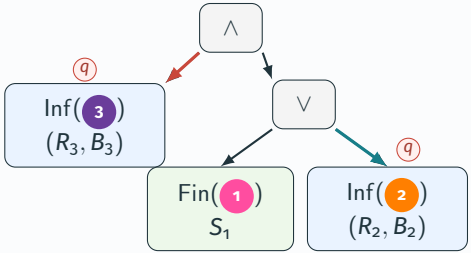
sample a new run $q \in C$.



Later verification checks whether these obligations remain valid.

Main idea: sample runs, then verify

After sampling q for $\psi = \mathrm{Inf}(3) \wedge (\mathrm{Fin}(1) \vee \mathrm{Inf}(2))$, the chosen leaf stores an **obligation** that must be checked later



Chosen branch: $\mathrm{Inf}(2)$

q in (R_2, B_2): the breakpoint B_2 empties infinitely often

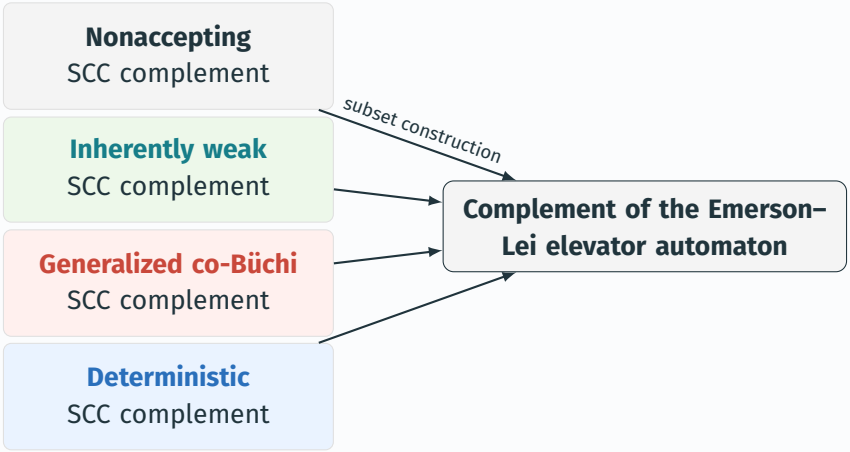
Alternative branch: $\mathrm{Fin}(1)$

$q \in S_1$: No color 1 in future

Takeaway

Sample runs to make guesses; verify the guess

Combining the SCC complements



Accept when all B sets become empty infinitely often

Complexity comparison with existing approaches

Automata parameters: n: states; k: colors; ℓ: pairs/atoms.

Acceptance condition	Unrestricted automata	Elevator automata (this work)
Büchi	$O(n(0.76n)^n)$	4^n
Generalized Büchi	$O(n(0.76nk)^n)$	$(k+3)^n$
Rabin (ℓ pairs)	$O\big(n\ell(0.76n)^{n\ell}\big)$	$3 \cdot 2^{(3\ell+2)n}$
Generalized Rabin (ℓ pairs)	$O\big(n\ell(0.76kn)^{\ell n}\big)$	$3 \cdot 2^{(3k\ell+2)n}$
Streett (ℓ pairs)	$2^{\Theta(n \log n + n\ell \log \ell)}$	$6\ell^n \cdot 2^{(2\ell+2)n}$
Parity (min-odd index ℓ)	$2^{O(n \log n)}$	$3 \cdot 2^{(\frac{3}{2}\ell+2)n}$
Emerson–Lei (ℓ atoms)	$O\Big(n^{2^k}(0.76nk)^{n2^k}\Big)$	$3\ell \cdot 2^{(2\ell+3)n}$

Experimental results

Implemented in Kofola

2,693 automata

● GRA-Rand ● LTL-Rand ● LTL-Lit

120s timeout

`autcross` ✓

119

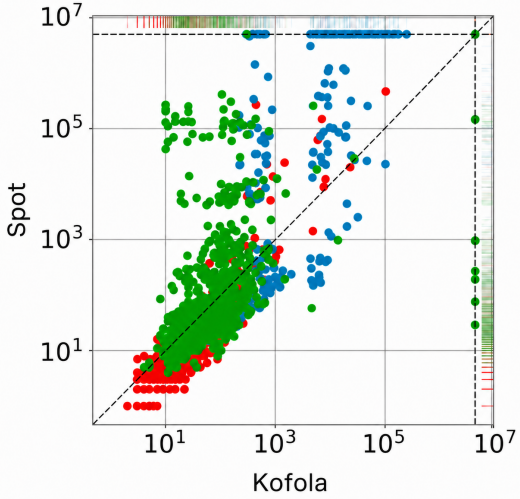
more instances solved than Spot

4.4×

smaller total output than Spot

40

unsolved instances; best robustness



Summary

Takeaway

- **Specialized** algorithm to complement Emerson–Lei elevator automata
- Exploit **acceptance** syntax and **SCC structure**
- Obtain **better** upper bounds and **smaller** Büchi complements

More results not in the paper

Determinization of Emerson–Lei elevator automata: $\Theta(n!)$

Backup: complexity parameters for the inductive construction

Let $\psi = \neg \mathsf{Acc}$. Consider only determinsitic SCCs since it is the dominant component.

- x: number of Fin atoms in ψ.
- y: number of Inf atoms in ψ.
- d: maximum number of Inf leaves in a $\wedge$-only subtree.
- m: number of maximal $\wedge$-only subtrees with an Inf leaf.

With shared breakpoints, a general bound is

$$3 \cdot \max\{d, 1\} \cdot \left(1 + 2 \cdot 2^{y} \cdot 2^{x} \cdot 2^{m}\right)^{n}.$$