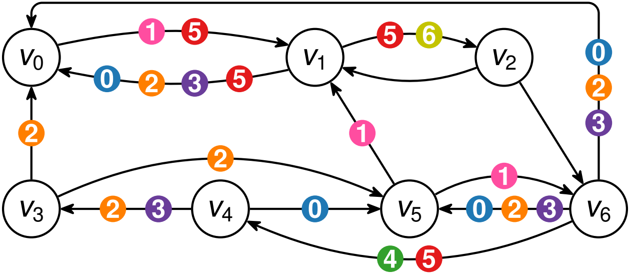
Generic Emptiness Check for Fun and Profit

Christel Baier František Blahoudek Alexandre Duret-Lutz
Joachim Klein David Müller Jan Strejček

MeFoSyLoMa — May 24th

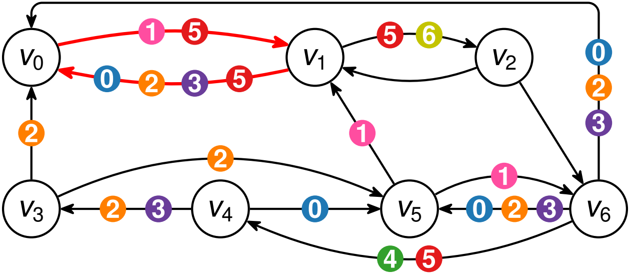
A Fun Puzzle



Find a cycle whose set of marks satisfies this formula:

$$\big((\neg 0 \wedge 1) \vee (\neg 2 \wedge 3)\big) \wedge (\neg 4 \vee 5) \wedge (\neg 6 \vee 7)$$

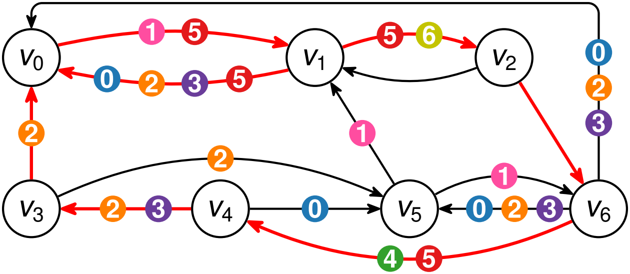
A Fun Puzzle



Find a cycle whose set of marks satisfies this formula:

$$\big((\neg 0 \wedge 1) \vee (\neg 2 \wedge 3)\big) \wedge (\neg 4 \vee 5) \wedge (\neg 6 \vee 7)$$

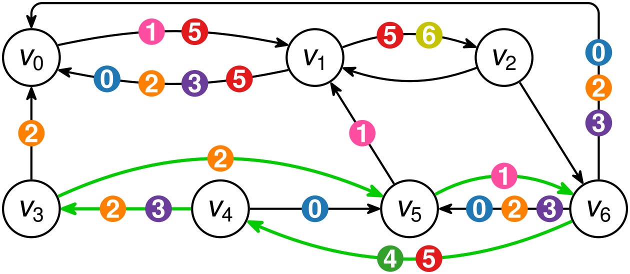
A Fun Puzzle



Find a cycle whose set of marks satisfies this formula:

$$\Big(\big(\neg \textcircled{0} \land \textcircled{1}\big) \lor \big(\neg \textcircled{2} \land \textcircled{3}\big)\Big) \land \big(\neg \textcircled{4} \lor \textcircled{5}\big) \land \big(\neg \textcircled{6} \lor \textcircled{7}\big)$$

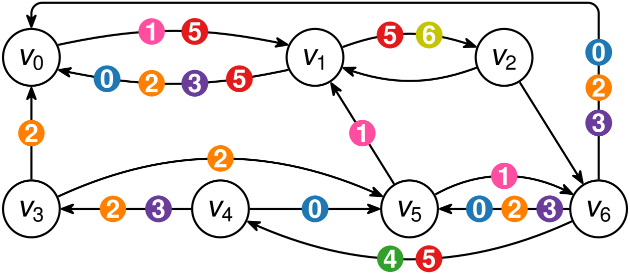
A Fun Puzzle



Find a cycle whose set of marks satisfies this formula:

$$\Big(\big(\neg 0 \wedge 1\big) \vee \big(\neg 2 \wedge 3\big)\Big) \wedge \big(\neg 4 \vee 5\big) \wedge \big(\neg 6 \vee 7\big)$$

A Fun Puzzle



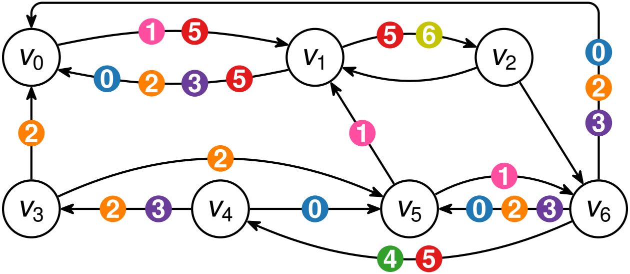
Find a cycle whose set of marks satisfies this formula:

$$\big((\neg 0 \wedge 1) \vee (\neg 2 \wedge 3)\big) \wedge (\neg 4 \vee 5) \wedge (\neg 6 \vee 7)$$

Idea 1

Enumerate all cycles of $G=(V,E)$;
evaluate φ on each.
Runs in $\mathcal{O}\left(2^{|E|} \cdot |\varphi|\right)$.

A Fun Puzzle



Find a cycle whose set of marks satisfies this formula:

$$\big((\neg 0 \wedge 1) \vee (\neg 2 \wedge 3)\big) \wedge (\neg 4 \vee 5) \wedge (\neg 6 \vee 7)$$

Idea 1

Enumerate all cycles of $G=(V,E)$;
evaluate φ on each.
Runs in $\mathcal{O}\left(2^{|E|} \cdot |\varphi|\right)$.

Idea 2

Enumerate all models m_i of φ;
check $G_{m_i}=(V, E_{|m_i})$ for a cycle.
Runs in $\mathcal{O}\left(2^{|\varphi|} \cdot n \cdot |E|\right)$.

Two Serious Applications

1. Emptiness check of a transition-based EL-automaton.
2. Model checking problem of probabilistic positiveness of MDP under a property given as a deterministic EL-automaton.

Two Serious Applications

1. Emptiness check of a transition-based EL-automaton.
2. Model checking problem of probabilistic positiveness of MDP under a property given as a deterministic EL-automaton.

EL-automata?

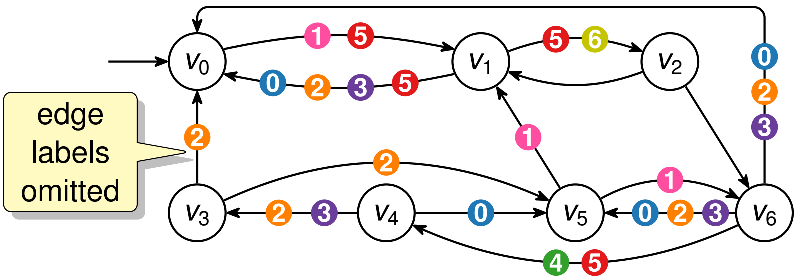
ω-automata where acceptance conditions are given as Boolean formulas over terms like $\mathrm{Fin}(T)$ or $\mathrm{Inf}(T)$ where T is a set of transitions (or states).

E.g. Rabin: $(\mathrm{Fin}(U_1) \wedge \mathrm{Inf}(L_1)) \vee (\mathrm{Fin}(U_2) \wedge \mathrm{Inf}(L_2))$.

E. A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 1987

S. Safra and M. Y. Vardi. On ω-automata and temporal logic. *STOC'89*

Emptiness Check of EL-Automata

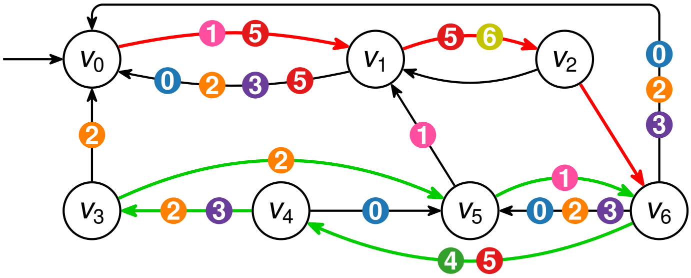


Is there a *reachable* cycle satisfying:

$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big) \wedge \big(\mathrm{Fin}(6) \vee \mathrm{Inf}(7)\big)$$

Where Fin(i) is satisfied iff the mark i is not on the cycle and Inf(j) is satisfied iff the mark j is on the cycle.

Emptiness Check of EL-Automata

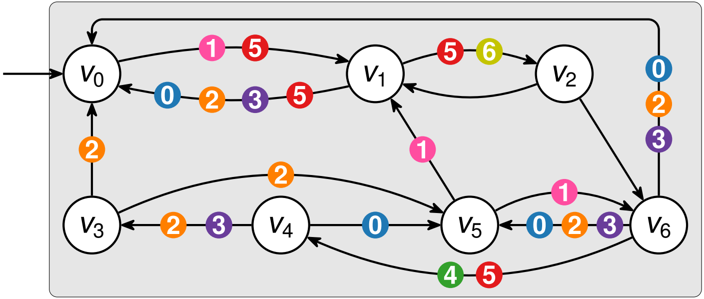


Is there a *reachable* cycle satisfying:

$$\Big(\big(\mathrm{Fin}(0)\wedge\mathrm{Inf}(1)\big)\vee\big(\mathrm{Fin}(2)\wedge\mathrm{Inf}(3)\big)\Big)\wedge\big(\mathrm{Fin}(4)\vee\mathrm{Inf}(5)\big)\wedge\big(\mathrm{Fin}(6)\vee\mathrm{Inf}(7)\big)$$

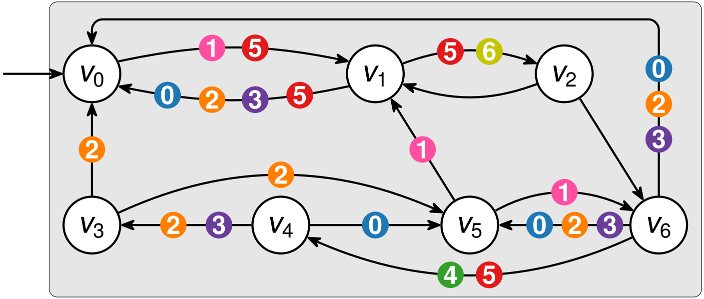
Where Fin(i) is satisfied iff the mark i is not on the cycle and Inf(j) is satisfied iff the mark j is on the cycle.

Emptiness Check of EL-Automata



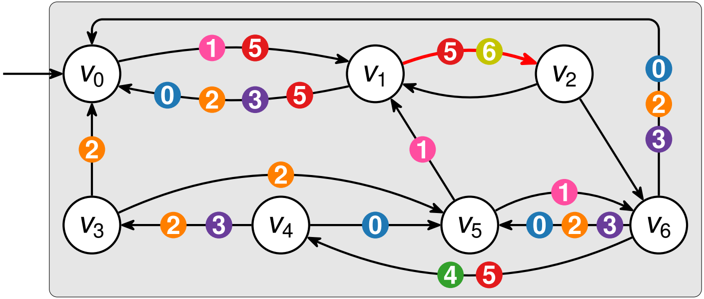
$$\Big(\big(\mathrm{Fin}(0)\wedge\mathrm{Inf}(1)\big)\vee\big(\mathrm{Fin}(2)\wedge\mathrm{Inf}(3)\big)\Big)\wedge\big(\mathrm{Fin}(4)\vee\mathrm{Inf}(5)\big)\wedge\big(\mathrm{Fin}(6)\vee\mathrm{Inf}(7)\big)$$

Emptiness Check of EL-Automata

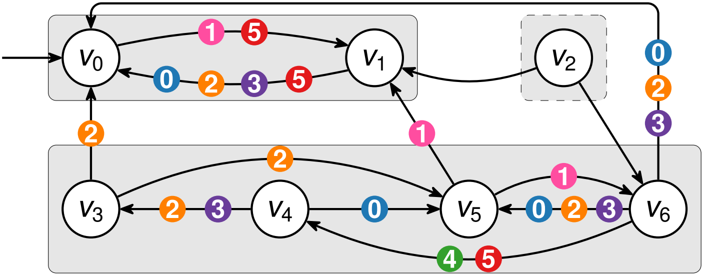


$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big) \wedge \mathrm{Fin}(6)$$

Emptiness Check of EL-Automata

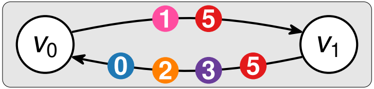


$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big) \wedge \mathrm{Fin}(6)$$

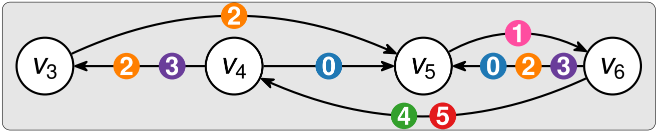


$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

Emptiness Check of EL-Automata

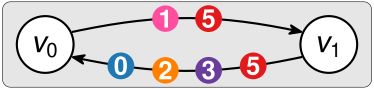


$$\big((\mathrm{Fin}(0) \land \mathrm{Inf}(1)) \lor (\mathrm{Fin}(2) \land \mathrm{Inf}(3))\big) \land (\mathrm{Fin}(4) \lor \mathrm{Inf}(5))$$

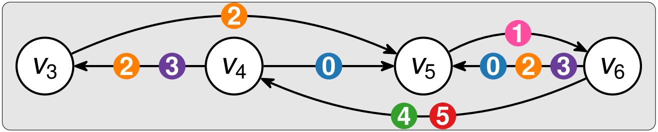


$$\big((\mathrm{Fin}(0) \land \mathrm{Inf}(1)) \lor (\mathrm{Fin}(2) \land \mathrm{Inf}(3))\big) \land (\mathrm{Fin}(4) \lor \mathrm{Inf}(5))$$

Emptiness Check of EL-Automata

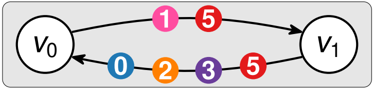


$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

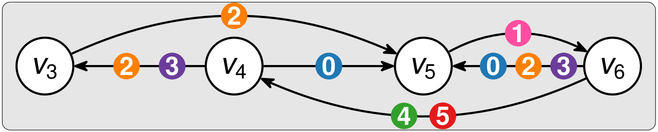


$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

Emptiness Check of EL-Automata

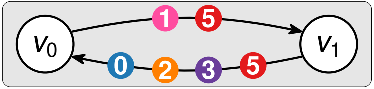


$$\big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\big)$$

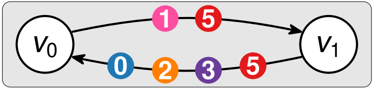


$$\big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

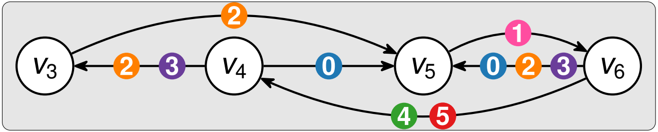
Emptiness Check of EL-Automata



$\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)$

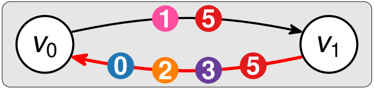


$\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)$

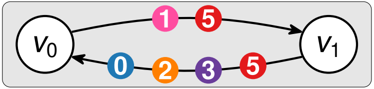


$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

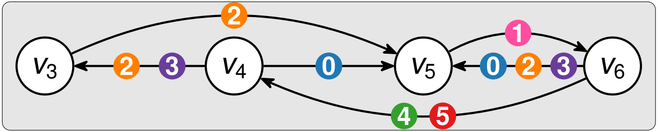
Emptiness Check of EL-Automata



$\mathrm{Fin}(0) \land \mathrm{Inf}(1)$

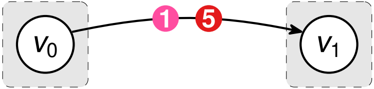


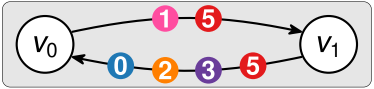
$\mathrm{Fin}(2) \land \mathrm{Inf}(3)$



$$\Big(\big(\mathrm{Fin}(0) \land \mathrm{Inf}(1)\big) \lor \big(\mathrm{Fin}(2) \land \mathrm{Inf}(3)\big)\Big) \land \big(\mathrm{Fin}(4) \lor \mathrm{Inf}(5)\big)$$

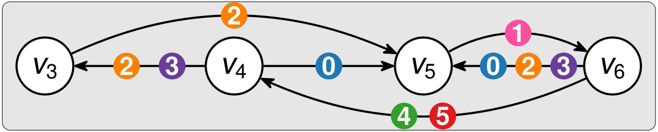
Emptiness Check of EL-Automata





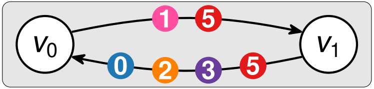
$\mathrm{Inf}(1)$

$\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)$

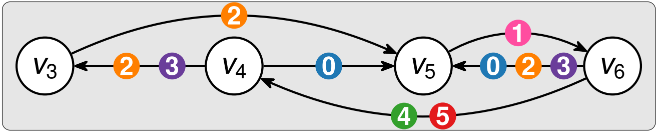


$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

Emptiness Check of EL-Automata

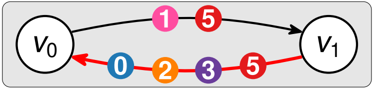


$$\mathrm{Fin}(\textcircled{2}) \wedge \mathrm{Inf}(\textcircled{3})$$

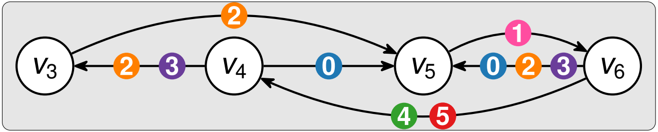


$$\Big(\big(\mathrm{Fin}(\textcircled{0}) \wedge \mathrm{Inf}(\textcircled{1})\big) \vee \big(\mathrm{Fin}(\textcircled{2}) \wedge \mathrm{Inf}(\textcircled{3})\big)\Big) \wedge \big(\mathrm{Fin}(\textcircled{4}) \vee \mathrm{Inf}(\textcircled{5})\big)$$

Emptiness Check of EL-Automata

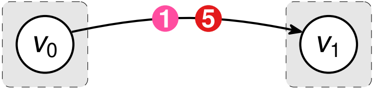


$$\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)$$

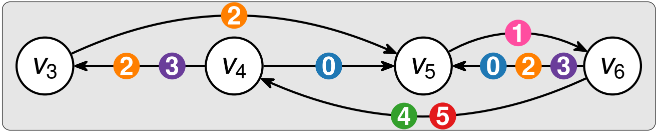


$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

Emptiness Check of EL-Automata

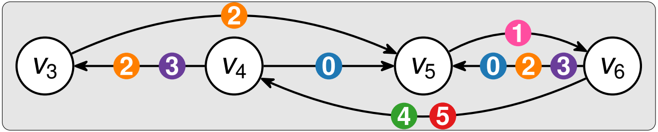


Inf(3)



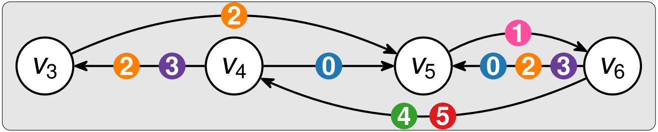
$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

Emptiness Check of EL-Automata



$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

Emptiness Check of EL-Automata

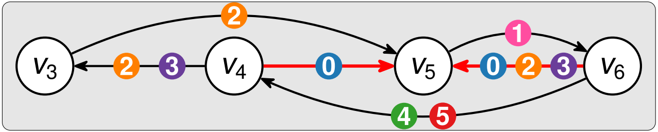


$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

Fin(0) is either true or false...

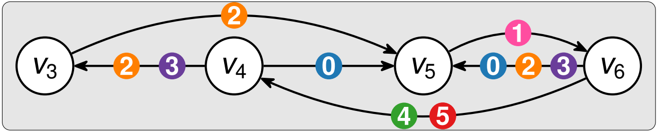
Emptiness Check of EL-Automata

If Fin(0) is true:



$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

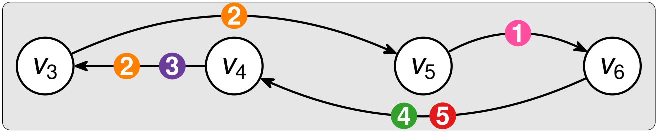
If Fin(0) is false:



$$\Big(\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

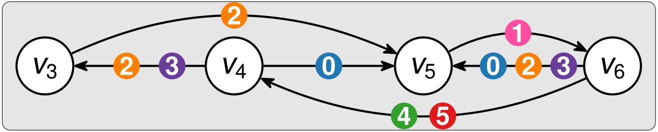
Emptiness Check of EL-Automata

If Fin(0) is true:



$$\Big(\quad \mathrm{Inf}(1) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)\Big) \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

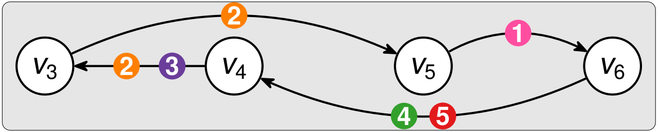
If Fin(0) is false:



$$\mathrm{Fin}(2) \wedge \mathrm{Inf}(3) \quad \wedge \big(\mathrm{Fin}(4) \vee \mathrm{Inf}(5)\big)$$

Emptiness Check of EL-Automata

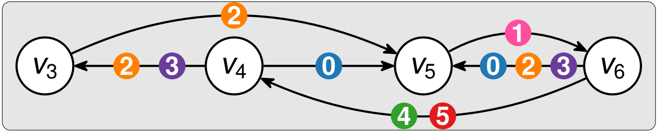
If Fin(0) is true:



$\Big(\quad \text{Inf}(1) \lor \big(\text{Fin}(2) \land \text{Inf}(3)\big)\Big) \land \big(\text{Fin}(4) \lor \text{Inf}(5)\big)$

Satisfied! $\Rightarrow$ automaton is non-empty!

If Fin(0) is false:



$\text{Fin}(2) \land \text{Inf}(3) \quad \land \big(\text{Fin}(4) \lor \text{Inf}(5)\big)$

Our Algorithm

IS_EMPTY$(G \in \mathbf{G}, \varphi \in \mathcal{C})$:

  foreach non-trivial $S \in$ SCCS_OF(G) **do** IS_SCC_EMPTY(S, φ)

IS_SCC_EMPTY$(S \in \mathbf{G}, \varphi \in \mathcal{C})$:

  $M_{\text{occur}} \longleftarrow$ MARKS_OF(S)

  $\varphi \longleftarrow \varphi[\forall m \notin M_{\text{occur}} : \mathsf{Inf}(m) \leftarrow \mathsf{f}, \mathsf{Fin}(m) \leftarrow \mathsf{t}]$

  if $\varphi = \mathsf{f}$ **then return**

  if $\varphi[\forall m \in M_{\text{occur}} : \mathsf{Inf}(m) \leftarrow \mathsf{t}] = \mathsf{t}$ **then raise** NONEMPTY

  foreach disjunct φ_j of φ **do**

    if $\varphi_j = \varphi' \wedge \bigwedge_{m \in M'} \mathsf{Fin}(m)$ **then**

      IS_EMPTY(REMOVE$(S, M'), \varphi'$)

    else

      pick some m such that $\mathsf{Fin}(m)$ occurs in φ_j

      IS_EMPTY(REMOVE$(S, \{m\}), \varphi_j[\mathsf{Fin}(m) \leftarrow \mathsf{t}]$)

      IS_SCC_EMPTY$(S, \varphi_j[\mathsf{Fin}(m) \leftarrow \mathsf{f}])$

IS_EMPTY terminates normally if G is empty.

Behavior on Classical Acceptance Conditions

Generalized-Büchi

$\bigwedge_i \mathsf{Inf}(m_i)$

Fin-less

any positive formula of Inf(...)

Our Algorithm for Fin-Less

IS_EMPTY$(G \in \mathbf{G}, \varphi \in \mathcal{C})$:

foreach non-trivial $S \in$ SCCS_OF(G) **do** IS_SCC_EMPTY(S, φ)

IS_SCC_EMPTY$(S \in \mathbf{G}, \varphi \in \mathcal{C})$:

$M_{\text{occur}} \longleftarrow$ MARKS_OF(S)

$\varphi \longleftarrow \varphi[\forall m \notin M_{\text{occur}} : \mathrm{Inf}(m) \leftarrow \mathrm{f}, \mathrm{Fin}(m) \leftarrow \mathrm{t}]$

if $\varphi = \mathrm{f}$ **then return**

if $\varphi[\forall m \in M_{\text{occur}} : \mathrm{Inf}(m) \leftarrow \mathrm{t}] = \mathrm{t}$ **then raise** NONEMPTY

foreach disjunct φ_j of φ **do**

if $\varphi_j = \varphi' \wedge \bigwedge_{m \in M'} \mathrm{Fin}(m)$ **then**

IS_EMPTY(REMOVE$(S, M'), \varphi')$

else

pick some m such that $\mathrm{Fin}(m)$ occurs in φ_j

IS_EMPTY(REMOVE$(S, \{m\}), \varphi_j[\mathrm{Fin}(m) \leftarrow \mathrm{t}])$

IS_SCC_EMPTY$(S, \varphi_j[\mathrm{Fin}(m) \leftarrow \mathrm{f}])$

Behavior on Classical Acceptance Conditions

Generalized-Büchi $\mathcal{O}(n \cdot |E| + |\varphi| \cdot |V|)$

$\bigwedge_i \mathrm{Inf}(m_i)$

Fin-less $\mathcal{O}(n \cdot |E| + |\varphi| \cdot |V|)$

any positive formula of Inf(...)

number of marks $n \leq |\varphi|$

Behavior on Classical Acceptance Conditions

Generalized-Büchi $\mathcal{O}\left(n \cdot |E| + |\varphi| \cdot |V|\right)$

$\bigwedge_i \mathsf{Inf}(m_i)$

Fin-less $\mathcal{O}\left(n \cdot |E| + |\varphi| \cdot |V|\right)$

any positive formula of Inf(...)

Rabin

$\bigvee_i \left(\mathsf{Fin}(m_i) \wedge \mathsf{Inf}(m'_i)\right)$

generalized Rabin

$\bigvee_i \left(\mathsf{Fin}(m_i) \wedge \bigwedge_{j \in J_i} \mathsf{Inf}(m_j)\right)$

Our Algorithm for generalized Rabin

IS_EMPTY$(G \in \mathbf{G}, \varphi \in \mathcal{C})$:

foreach non-trivial $S \in$ SCCS_OF(G) **do** IS_SCC_EMPTY(S, φ)

IS_SCC_EMPTY$(S \in \mathbf{G}, \varphi \in \mathcal{C})$:

$M_{\text{occur}} \longleftarrow$ MARKS_OF(S)

$\varphi \longleftarrow \varphi[\forall m \notin M_{\text{occur}} : \mathrm{Inf}(m) \leftarrow \mathrm{f}, \mathrm{Fin}(m) \leftarrow \mathrm{t}]$

if $\varphi = \mathrm{f}$ **then return**

if $\varphi[\forall m \in M_{\text{occur}} : \mathrm{Inf}(m) \leftarrow \mathrm{t}] = \mathrm{t}$ **then raise** NonEmpty

foreach disjunct φ_j of φ **do**

if $\varphi_j = \varphi' \wedge \bigwedge_{m \in M'} \mathrm{Fin}(m)$ **then**

IS_EMPTY(REMOVE$(S, M'), \varphi')$

else

pick some m such that $\mathrm{Fin}(m)$ occurs in φ_j

generalized Rabin example

$\big(\mathrm{Inf}(0) \wedge \mathrm{Fin}(1)\big) \vee \big(\mathrm{Inf}(2) \wedge \mathrm{Inf}(3) \wedge \mathrm{Fin}(4)\big) \vee \mathrm{Fin}(5)$

Behavior on Classical Acceptance Conditions

Generalized-Büchi $\mathcal{O}(n \cdot |E| + |\varphi| \cdot |V|)$

$\bigwedge_i \mathsf{Inf}(m_i)$

Fin-less $\mathcal{O}(n \cdot |E| + |\varphi| \cdot |V|)$

any positive formula of Inf(...)

Rabin $\mathcal{O}(n \cdot |\varphi| \cdot |E|)$

$\bigvee_i \big(\mathsf{Fin}(m_i) \land \mathsf{Inf}(m'_i)\big)$

generalized Rabin $\mathcal{O}(n \cdot |\varphi| \cdot |E|)$

$\bigvee_i \big(\mathsf{Fin}(m_i) \land \bigwedge_{j \in J_i} \mathsf{Inf}(m_j)\big)$

Behavior on Classical Acceptance Conditions

Generalized-Büchi $\mathcal{O}(n \cdot |E| + |\varphi| \cdot |V|)$

$\bigwedge_i \mathsf{Inf}(m_i)$

Fin-less $\mathcal{O}(n \cdot |E| + |\varphi| \cdot |V|)$

any positive formula of Inf(...)

Rabin $\mathcal{O}(n \cdot |\varphi| \cdot |E|)$

$\bigvee_i \Big(\mathsf{Fin}(m_i) \wedge \mathsf{Inf}(m'_i)\Big)$

generalized Rabin $\mathcal{O}(n \cdot |\varphi| \cdot |E|)$

$\bigvee_i \Big(\mathsf{Fin}(m_i) \wedge \bigwedge_{j \in J_i} \mathsf{Inf}(m_j)\Big)$

Streett

$\bigwedge_i \Big(\mathsf{Inf}(m_i) \vee \mathsf{Fin}(m'_i)\Big)$

Our Algorithm for Streett

IS_EMPTY$(G \in \mathbf{G}, \varphi \in \mathcal{C})$:

foreach non-trivial $S \in$ SCCS_OF(G) **do** IS_SCC_EMPTY(S, φ)

IS_SCC_EMPTY$(S \in \mathbf{G}, \varphi \in \mathcal{C})$:

$M_{\mathrm{occur}} \longleftarrow$ MARKS_OF(S)

$\varphi \longleftarrow \varphi[\forall m \notin M_{\mathrm{occur}} : \mathrm{Inf}(m) \leftarrow \mathrm{f}, \mathrm{Fin}(m) \leftarrow \mathrm{t}]$

if $\varphi = \mathrm{f}$ **then return**

if $\varphi[\forall m \in M_{\mathrm{occur}} : \mathrm{Inf}(m) \leftarrow \mathrm{t}] = \mathrm{t}$ **then raise** NONEMPTY

foreach disjunct φ_j of φ **do**

if $\varphi_j = \varphi' \wedge \bigwedge_{m \in M'} \mathrm{Fin}(m)$ **then**

IS_EMPTY(REMOVE$(S, M'), \varphi')$

else

pick some m such that $\mathrm{Fin}(m)$ occurs in

Streett example

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1)) \wedge (\mathrm{Inf}(2) \vee \mathrm{Fin}(3)) \wedge (\mathrm{Inf}(4) \vee \mathrm{Fin}(5))$

easily satisfied unless one of the Inf marks is missing.

Behavior on Classical Acceptance Conditions

Generalized-Büchi $\mathcal{O}(n \cdot |E| + |\varphi| \cdot |V|)$

$\bigwedge_i \mathrm{Inf}(m_i)$

Fin-less $\mathcal{O}(n \cdot |E| + |\varphi| \cdot |V|)$

any positive formula of Inf(...)

Rabin $\mathcal{O}(n \cdot |\varphi| \cdot |E|)$

$\bigvee_i \big(\mathrm{Fin}(m_i) \wedge \mathrm{Inf}(m'_i)\big)$

generalized Rabin $\mathcal{O}(n \cdot |\varphi| \cdot |E|)$

$\bigvee_i \big(\mathrm{Fin}(m_i) \wedge \bigwedge_{j \in J_i} \mathrm{Inf}(m_j)\big)$

Streett $\mathcal{O}(f \cdot (n \cdot |E| + |\varphi| \cdot |V|))$

$\bigwedge_i \big(\mathrm{Inf}(m_i) \vee \mathrm{Fin}(m'_i)\big)$

number of Fin marks

$f \leq n \leq |\varphi|$

Behavior on Classical Acceptance Conditions

Generalized-Büchi $\mathcal{O}\left(n \cdot |E| + |\varphi| \cdot |V|\right)$

$\bigwedge_i \mathsf{Inf}(m_i)$

Fin-less $\mathcal{O}\left(n \cdot |E| + |\varphi| \cdot |V|\right)$

any positive formula of Inf(...)

Rabin $\mathcal{O}\left(n \cdot |\varphi| \cdot |E|\right)$

$\bigvee_i \Big(\mathsf{Fin}(m_i) \wedge \mathsf{Inf}(m'_i)\Big)$

generalized Rabin $\mathcal{O}\left(n \cdot |\varphi| \cdot |E|\right)$

$\bigvee_i \Big(\mathsf{Fin}(m_i) \wedge \bigwedge_{j \in J_i} \mathsf{Inf}(m_j)\Big)$

Streett $\mathcal{O}\left(f \cdot (n \cdot |E| + |\varphi| \cdot |V|)\right)$

$\bigwedge_i \Big(\mathsf{Inf}(m_i) \vee \mathsf{Fin}(m'_i)\Big)$

parity (min even)

$\mathsf{Inf}(m_0) \vee (\mathsf{Fin}(m_1) \wedge (\mathsf{Inf}(m_2) \vee (\mathsf{Fin}(m_3) \wedge \ldots)))$

Behavior on Classical Acceptance Conditions

Generalized-Büchi $\mathcal{O}(n \cdot |E| + |\varphi| \cdot |V|)$

$\bigwedge_i \mathsf{Inf}(m_i)$

Fin-less $\mathcal{O}(n \cdot |E| + |\varphi| \cdot |V|)$

any positive formula of Inf(...)

Rabin $\mathcal{O}(n \cdot |\varphi| \cdot |E|)$

$\bigvee_i \big(\mathsf{Fin}(m_i) \wedge \mathsf{Inf}(m'_i)\big)$

generalized Rabin $\mathcal{O}(n \cdot |\varphi| \cdot |E|)$

$\bigvee_i \big(\mathsf{Fin}(m_i) \wedge \bigwedge_{j \in J_i} \mathsf{Inf}(m_j)\big)$

Streett $\mathcal{O}(f \cdot (n \cdot |E| + |\varphi| \cdot |V|))$

$\bigwedge_i \big(\mathsf{Inf}(m_i) \vee \mathsf{Fin}(m'_i)\big)$

parity (min even) $\mathcal{O}(f \cdot (n \cdot |E| + |\varphi| \cdot |V|))$

$\mathsf{Inf}(m_0) \vee (\mathsf{Fin}(m_1) \wedge (\mathsf{Inf}(m_2) \vee (\mathsf{Fin}(m_3) \wedge \ldots)))$

Behavior on Classical Acceptance Conditions

Generalized-Büchi $\mathcal{O}\left(n \cdot |E| + |\varphi| \cdot |V|\right)$

$\bigwedge_i \mathsf{Inf}(m_i)$

Fin-less $\mathcal{O}\left(n \cdot |E| + |\varphi| \cdot |V|\right)$

any positive formula of Inf(...)

Rabin $\mathcal{O}\left(n \cdot |\varphi| \cdot |E|\right)$

$\bigvee_i \big(\mathsf{Fin}(m_i) \wedge \mathsf{Inf}(m'_i)\big)$

generalized Rabin $\mathcal{O}\left(n \cdot |\varphi| \cdot |E|\right)$

$\bigvee_i \big(\mathsf{Fin}(m_i) \wedge \bigwedge_{j \in J_i} \mathsf{Inf}(m_j)\big)$

Streett $\mathcal{O}\left(f \cdot (n \cdot |E| + |\varphi| \cdot |V|)\right)$

$\bigwedge_i \big(\mathsf{Inf}(m_i) \vee \mathsf{Fin}(m'_i)\big)$

parity (min even) $\mathcal{O}\left(f \cdot (n \cdot |E| + |\varphi| \cdot |V|)\right)$

$\mathsf{Inf}(m_0) \vee (\mathsf{Fin}(m_1) \wedge (\mathsf{Inf}(m_2) \vee (\mathsf{Fin}(m_3) \wedge \ldots)))$

hyper-Rabin

$\bigvee_i \bigwedge_{j \in J_i} \big(\mathsf{Inf}(m_j) \vee \mathsf{Fin}(m'_j)\big)$

Our Algorithm for hyper-Rabin

IS_EMPTY($G \in \mathbf{G}, \varphi \in \mathcal{C}$):
  foreach non-trivial $S \in$ SCCS_OF(G) **do** IS_SCC_EMPTY(S, φ)

IS_SCC_EMPTY($S \in \mathbf{G}, \varphi \in \mathcal{C}$):
  $M_{\text{occur}} \longleftarrow$ MARKS_OF(S)
  $\varphi \longleftarrow \varphi[\forall m \notin M_{\text{occur}} : \mathrm{Inf}(m) \leftarrow \mathrm{f}, \mathrm{Fin}(m) \leftarrow \mathrm{t}]$
  if $\varphi = \mathrm{f}$ **then return**
  if $\varphi[\forall m \in M_{\text{occur}} : \mathrm{Inf}(m) \leftarrow \mathrm{t}] = \mathrm{t}$ **then raise** NonEmpty
  foreach disjunct φ_j of φ **do**
    if $\varphi_j = \varphi' \wedge \bigwedge_{m \in M'} \mathrm{Fin}(m)$ **then**
      IS_EMPTY(REMOVE(S, M'), φ')
    else
      pick some m such that $\mathrm{Fin}(m)$ occurs in φ_j
      IS_EMPTY(REMOVE($S, \{m\}$), $\varphi_j[\mathrm{Fin}(m) \leftarrow \mathrm{t}]$)
      IS_SCC_EMPTY($S, \varphi_j[\mathrm{Fin}(m) \leftarrow \mathrm{f}]$)

if φ is hyper-Rabin, then φ_j is Streett, and is not Inf-satisfied at this point.

Behavior on Classical Acceptance Conditions

Generalized-Büchi $\mathcal{O}\left(n \cdot |E| + |\varphi| \cdot |V|\right)$

$\bigwedge_i \mathrm{Inf}(m_i)$

Fin-less $\mathcal{O}\left(n \cdot |E| + |\varphi| \cdot |V|\right)$

any positive formula of Inf(...)

Rabin $\mathcal{O}\left(n \cdot |\varphi| \cdot |E|\right)$

$\bigvee_i \big(\mathrm{Fin}(m_i) \wedge \mathrm{Inf}(m'_i)\big)$

generalized Rabin $\mathcal{O}\left(n \cdot |\varphi| \cdot |E|\right)$

$\bigvee_i \big(\mathrm{Fin}(m_i) \wedge \bigwedge_{j \in J_i} \mathrm{Inf}(m_j)\big)$

Streett $\mathcal{O}\left(f \cdot (n \cdot |E| + |\varphi| \cdot |V|)\right)$

$\bigwedge_i \big(\mathrm{Inf}(m_i) \vee \mathrm{Fin}(m'_i)\big)$

parity (min even) $\mathcal{O}\left(f \cdot (n \cdot |E| + |\varphi| \cdot |V|)\right)$

$\mathrm{Inf}(m_0) \vee (\mathrm{Fin}(m_1) \wedge (\mathrm{Inf}(m_2) \vee (\mathrm{Fin}(m_3) \wedge \ldots)))$

hyper-Rabin $\mathcal{O}\left(|\varphi| \cdot (n \cdot |E| + |\varphi| \cdot |V|)\right)$

$\bigvee_i \bigwedge_{j \in J_i} \big(\mathrm{Inf}(m_j) \vee \mathrm{Fin}(m'_j)\big)$

Behavior on Classical Acceptance Conditions

Generalized-Büchi $\mathcal{O}\left(n \cdot |E| + |\varphi| \cdot |V|\right)$

$\bigwedge_i \mathsf{Inf}(m_i)$

Fin-less $\mathcal{O}\left(n \cdot |E| + |\varphi| \cdot |V|\right)$

any positive formula of Inf(...)

Rabin $\mathcal{O}\left(n \cdot |\varphi| \cdot |E|\right)$

$\bigvee_i \big(\mathsf{Fin}(m_i) \wedge \mathsf{Inf}(m'_i)\big)$

generalized Rabin $\mathcal{O}\left(n \cdot |\varphi| \cdot |E|\right)$

$\bigvee_i \big(\mathsf{Fin}(m_i) \wedge \bigwedge_{j \in J_i} \mathsf{Inf}(m_j)\big)$

Streett $\mathcal{O}\left(f \cdot (n \cdot |E| + |\varphi| \cdot |V|)\right)$

$\bigwedge_i \big(\mathsf{Inf}(m_i) \vee \mathsf{Fin}(m'_i)\big)$

parity (min even) $\mathcal{O}\left(f \cdot (n \cdot |E| + |\varphi| \cdot |V|)\right)$

$\mathsf{Inf}(m_0) \vee (\mathsf{Fin}(m_1) \wedge (\mathsf{Inf}(m_2) \vee (\mathsf{Fin}(m_3) \wedge \ldots)))$

hyper-Rabin $\mathcal{O}\left(|\varphi| \cdot (n \cdot |E| + |\varphi| \cdot |V|)\right)$

$\bigvee_i \bigwedge_{j \in J_i} \big(\mathsf{Inf}(m_j) \vee \mathsf{Fin}(m'_j)\big)$

worst case

Our Algorithm in its Worst Case

IS_EMPTY$(G \in \mathbf{G}, \varphi \in \mathcal{C})$:
    foreach non-trivial $S \in$ SCCS_OF(G) **do** IS_SCC_EMPTY(S, φ)

IS_SCC_EMPTY$(S \in \mathbf{G}, \varphi \in \mathcal{C})$:
    $M_{\text{occur}} \longleftarrow$ MARKS_OF(S)
    $\varphi \longleftarrow \varphi[\forall m \notin M_{\text{occur}} : \mathsf{Inf}(m) \leftarrow \mathsf{f}, \mathsf{Fin}(m) \leftarrow \mathsf{t}]$
    if $\varphi = \mathsf{f}$ **then return**
    if $\varphi[\forall m \in M_{\text{occur}} : \mathsf{Inf}(m) \leftarrow \mathsf{t}] = \mathsf{t}$ **then raise** NONEMPTY
    foreach disjunct φ_j of φ **do**
        if $\varphi_j = \varphi' \wedge \bigwedge_{m \in M'} \mathsf{Fin}(m)$ **then**
            IS_EMPTY$(\text{REMOVE}(S, M'), \varphi')$
        else
            pick some m such that $\mathsf{Fin}(m)$ occurs in φ_j
            IS_EMPTY$\big(\text{REMOVE}(S, \{m\}), \varphi_j[\mathsf{Fin}(m) \leftarrow \mathsf{t}]\big)$
            IS_SCC_EMPTY$(S, \varphi_j[\mathsf{Fin}(m) \leftarrow \mathsf{f}])$

Behavior on Classical Acceptance Conditions

Generalized-Büchi $\mathcal{O}(n \cdot |E| + |\varphi| \cdot |V|)$

$\bigwedge_i \mathrm{Inf}(m_i)$

Fin-less $\mathcal{O}(n \cdot |E| + |\varphi| \cdot |V|)$

any positive formula of Inf(...)

Rabin $\mathcal{O}(n \cdot |\varphi| \cdot |E|)$

$\bigvee_i \Big(\mathrm{Fin}(m_i) \wedge \mathrm{Inf}(m'_i)\Big)$

generalized Rabin $\mathcal{O}(n \cdot |\varphi| \cdot |E|)$

$\bigvee_i \Big(\mathrm{Fin}(m_i) \wedge \bigwedge_{j \in J_i} \mathrm{Inf}(m_j)\Big)$

Streett $\mathcal{O}(f \cdot (n \cdot |E| + |\varphi| \cdot |V|))$

$\bigwedge_i \Big(\mathrm{Inf}(m_i) \vee \mathrm{Fin}(m'_i)\Big)$

parity (min even) $\mathcal{O}(f \cdot (n \cdot |E| + |\varphi| \cdot |V|))$

$\mathrm{Inf}(m_0) \vee (\mathrm{Fin}(m_1) \wedge (\mathrm{Inf}(m_2) \vee (\mathrm{Fin}(m_3) \wedge \ldots)))$

hyper-Rabin $\mathcal{O}(|\varphi| \cdot (n \cdot |E| + |\varphi| \cdot |V|))$

$\bigvee_i \bigwedge_{j \in J_i} \Big(\mathrm{Inf}(m_j) \vee \mathrm{Fin}(m'_j)\Big)$

worst case $\mathcal{O}\Big(2^f \cdot n \cdot |\varphi| \cdot |E|\Big)$

Relation to Emerson & Lei's Emptiness Check

Hyper-Rabin

- Introduced in the 80s by EL under the name "canonical form" (any condition can be converted to it).
- Renamed hyper-Rabin by Boker in 2018.

General Case

E. A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 1987

U. Boker. Why these automata types? *LPAR'18*

Relation to Emerson & Lei's Emptiness Check

Hyper-Rabin

- Introduced in the 80s by EL under the name "canonical form" (any condition can be converted to it).
- Renamed hyper-Rabin by Boker in 2018.
- EL had an emptiness for hyper-Rabin. Our algorithm behaves similarly, with the same complexity.

General Case

E. A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 1987

U. Boker. Why these automata types? *LPAR'18*

Relation to Emerson & Lei's Emptiness Check

Hyper-Rabin

- Introduced in the 80s by EL under the name "canonical form" (any condition can be converted to it).
- Renamed hyper-Rabin by Boker in 2018.
- EL had an emptiness for hyper-Rabin. Our algorithm behaves similarly, with the same complexity.

General Case

- Emptiness-check of EL-automata is NP-complete.
- EL suggest to put φ in DNF then convert that to hyper-Rabin for emptiness check.
- We try to avoid the exponential DNF step by:
  - inspecting the automaton to simplify the formula
  - detecting cases that can be solved more easily

E. A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 1987

U. Boker. Why these automata types? *LPAR'18*

Relation to DPLL

$\textsc{is\_empty}(G \in \mathbf{G}, \varphi \in \mathcal{C})$:
  foreach non-trivial $S \in \textsc{sccs\_of}(G)$ **do** $\textsc{is\_scc\_empty}(S, \varphi)$

$\textsc{is\_scc\_empty}(S \in \mathbf{G}, \varphi \in \mathcal{C})$:
  $M_{\mathrm{occur}} \longleftarrow \textsc{marks\_of}(S)$
  $\varphi \longleftarrow \varphi[\forall m \notin M_{\mathrm{occur}} : \mathrm{Inf}(m) \leftarrow \mathrm{f}, \mathrm{Fin}(m) \leftarrow \mathrm{t}]$ — unit clause propagation
  if $\varphi = \mathrm{f}$ **then return**
  if $\varphi[\forall m \in M_{\mathrm{occur}} : \mathrm{Inf}(m) \leftarrow \mathrm{t}] = \mathrm{t}$ **then raise** NonEmpty
  foreach disjunct φ_j of φ **do**
    if $\varphi_j = \varphi' \wedge \bigwedge_{m \in M'} \mathrm{Fin}(m)$ **then** — unit clause detection
      $\textsc{is\_empty}(\textsc{remove}(S, M'), \varphi')$
    else
      pick some m such that $\mathrm{Fin}(m)$ occurs in φ_j — decision variable
      $\textsc{is\_empty}(\textsc{remove}(S, \{m\}), \varphi_j[\mathrm{Fin}(m) \leftarrow \mathrm{t}])$
      $\textsc{is\_scc\_empty}(S, \varphi_j[\mathrm{Fin}(m) \leftarrow \mathrm{f}])$

Optimizations

1. Notice marks that are present everywhere in S.

$$(M_{\text{occur}}, M_{\text{every}}) \longleftarrow \textsc{marks\_of}(S)$$
$$\varphi \longleftarrow \varphi[\forall m \notin M_{\text{occur}} : \mathsf{Inf}(m) \leftarrow \mathsf{f}, \mathsf{Fin}(m) \leftarrow \mathsf{t}]$$
$$\varphi \longleftarrow \varphi[\forall m \in M_{\text{every}} : \mathsf{Inf}(m) \leftarrow \mathsf{t}, \mathsf{Fin}(m) \leftarrow \mathsf{f}]$$

2. Implement φ-evaluation in SCCS_OF, so that IS_SCC_EMPTY is only called when its **foreach** loop will run.
3. Implement REMOVE as a filter that is passed to SCCS_OF.
4. Initialize that filter to M if $\varphi = \varphi' \wedge \bigwedge_{m\in M'} \mathsf{Fin}(m)$ at top-level.

Spot: Use-Cases for This Emptiness Check

Spot: manipulation of ω-automata with any EL-acceptance.

The following use-cases require $\textsc{is\_empty}(A \otimes B)$ where A and B have any acceptance conditions.

- `ltlcross` and `autcross`: check equivalence of automata produced by other tools.
- Deciding if an LTL formula/an automaton is stutter-invariant.
- Deciding if an LTL formula/an automaton is an obligation.

Deciding whether an SCC is *inherently weak* can be done using $\textsc{is\_scc\_empty}(S, \varphi) \vee \textsc{is\_scc\_empty}(S, \bar{\varphi})$.

Spot: Use-Cases for This Emptiness Check

Spot: manipulation of ω-automata with any EL-acceptance.

The following use-cases require $\textsc{is\_empty}(A \otimes B)$ where A and B have any acceptance conditions.

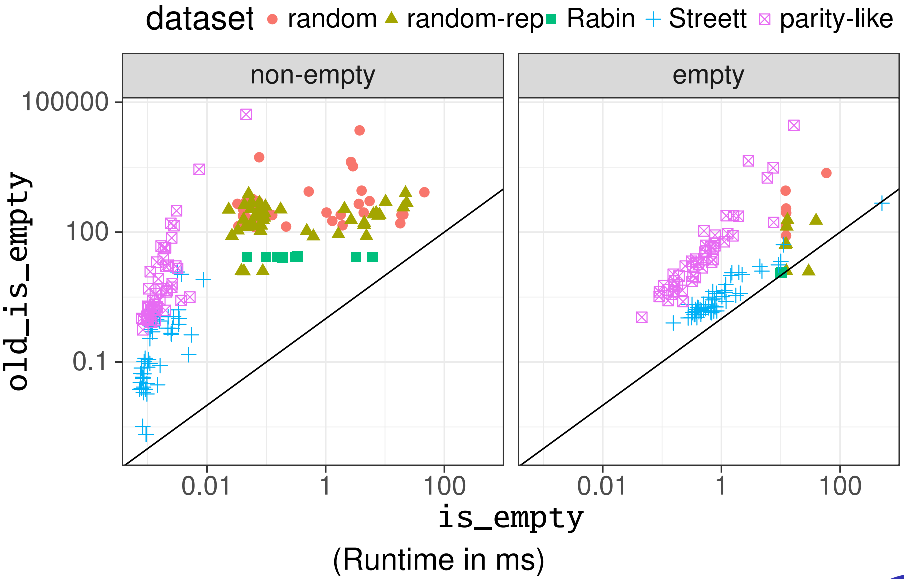
- `ltlcross` and `autcross`: check equivalence of automata produced by other tools.
- Deciding if an LTL formula/an automaton is stutter-invariant.
- Deciding if an LTL formula/an automaton is an obligation.

Deciding whether an SCC is *inherently weak* can be done using $\textsc{is\_scc\_empty}(S, \varphi) \vee \textsc{is\_scc\_empty}(S, \bar{\varphi})$.

Emptiness check for EL-automata:

Spot 2.0–2.6	Via conversion to Fin-less acceptance.
Spot 2.7–	This algorithm.

Spot: Comparison with Old Implementation



PRISM: Use-Case for This Emptiness Check

PRISM: Probabilistic Model Checker (PMC).

Can use a deterministic automaton A with EL-acceptance φ to encode an ω-regular path property to check on the model.

Consider the problem of checking whether a path property represented by A holds with positive probability.

For a Discrete-Time Markov Chain $\mathcal{M}$

For a Markov Decision Processes $\mathcal{P}$

PRISM: Use-Case for This Emptiness Check

PRISM: Probabilistic Model Checker (PMC).

Can use a deterministic automaton A with EL-acceptance φ to encode an ω-regular path property to check on the model.

Consider the problem of checking whether a path property represented by A holds with positive probability.

For a Discrete-Time Markov Chain $\mathcal{M}$

Build $\mathcal{M} \otimes A$. Search bottom SCCs (where all states are visited with probability 1); evaluate φ on each.
Polynomial for any φ. Already supported.

For a Markov Decision Processes $\mathcal{P}$

PRISM: Use-Case for This Emptiness Check

PRISM: Probabilistic Model Checker (PMC).

Can use a deterministic automaton A with EL-acceptance φ to encode an ω-regular path property to check on the model.

Consider the problem of checking whether a path property represented by A holds with positive probability.

For a Discrete-Time Markov Chain $\mathcal{M}$

Build $\mathcal{M} \otimes A$. Search bottom SCCs (where all states are visited with probability 1); evaluate φ on each.
Polynomial for any φ. Already supported.

For a Markov Decision Processes $\mathcal{P}$

Build $\mathcal{P} \otimes A$. Search maximal end-components (SCCs closed under probabilistic choice) satisfying φ.
Was supported for Rabin or generalized-Rabin only.
Patch implementing proposed algorithm written from PRISM 4.4.

PRISM: Emptiness Check Comparison

PRISM 4.4 has a generalized-Rabin emptiness check.

Maximal end-components analysis in seconds, with generalized-Rabin emptiness check (t_{Rabin})

	generalized Rabin		Rabin	
Property	t_{Rabin}	n	t_{Rabin}	n
$\Pr^{\min}(\phi_1)$	130.7	4	–	14
$\Pr^{\max}(\phi_2)$	234.3	6	–	8
$\Pr^{\max}(\phi_3)$	100.1	5	–	6
$\Pr^{\min}(\phi_4)$	251.9	6	1.6	6
$\Pr^{\max}(\phi_5)$	–	12	–	–
$\Pr^{\min}(\phi_6)$	355.3	10	54.9	6

instructions to reproduce...

PRISM: Emptiness Check Comparison

PRISM 4.4 has a generalized-Rabin emptiness check.
Patched version can run our algorithm (state-based).

Maximal end-components analysis in seconds, with generalized-Rabin emptiness check (t_{Rabin}) or our algorithm (t_{EL}).

	EL		generalized Rabin			Rabin		
Property	t_{EL}	n	t_{Rabin}	t_{EL}	n	t_{Rabin}	t_{EL}	n
$\Pr^{\min}(\phi_1)$	109.8	4	130.7	121.1	4	–	–	14
$\Pr^{\max}(\phi_2)$	0.4	3	234.3	0.7	6	–	585.9	8
$\Pr^{\max}(\phi_3)$	0.4	3	100.1	0.6	5	–	855.1	6
$\Pr^{\min}(\phi_4)$	0.6	4	251.9	119.0	6	1.6	0.6	6
$\Pr^{\max}(\phi_5)$	–	4	–	–	12	–	–	–
$\Pr^{\min}(\phi_6)$	107.0	6	355.3	127.3	10	54.9	9.6	6

instructions to reproduce...

Conclusion

Contributions

- Generic emptiness check that unifies various emptiness checks for simpler classes.
  - Polynomial on common acceptance conditions.
  - Exponential (in the number of Fin terms) in the worst case.
- Implemented in Spot and PRISM, with very clear improvements.

Possible improvements

- Parallelization
- Heuristics for non-deterministic choices