Permutation Games for the Weakly Aconjunctive μ-calculus

Daniel Hausmann daniel.hausmann@fau.de
Lutz Schröder lutz.schroeder@fau.de
Hans-Peter Deifel hans-peter.deifel@fau.de

Friedrich-Alexander-Universität Erlangen-Nürnberg
Department of Computer Science

TACAS 2018, 16-19 April 2018, Thessaloniki, Greece

Motivation

Constructing satisfiability games for the μ-calculus typically involves determinization of **tracking automata**.

Key observation: For **aconjunctive** formulas, tracking of **fixpoints** already is deterministic.

Results:

Determinization procedure for **limit-deterministic** parity automata;

Asymptotically smaller satisfiability games for **aconjunctive** formulas;

Implementation of solver for these games, **coalgebraic** and **on-the-fly**.

Limit-determinisitic parity automata

Parity automaton (PA) $\mathcal{A} = (V, \Sigma, \delta, u_0, \alpha)$ with transition relation $\delta \subseteq V \times \Sigma \times V$ and priority function $\alpha : \delta \to \mathbb{N}$. Priorities are assigned to **transitions** rather than states, e.g. [Schewe,Varghese 2014]. Write

$$\delta_l = \{t \in \delta \mid \alpha(v) = l\} \qquad \delta_{\leq l} = \{t \in \delta \mid \alpha(v) \leq l\}$$

Büchi automata are PA with priorities just 1 and 2 ($\delta_1 = \overline{F}$, $\delta_2 = F$).

For even l, the l-**compartment** $C(v, a, w)$ of $(v, a, w) \in \delta_l$ is the set of transitions reachable from w by transitions with priority at most l. PA $\mathcal{A}$ is **limit-deterministic** (LD) if all its compartments are **internally** deterministic.

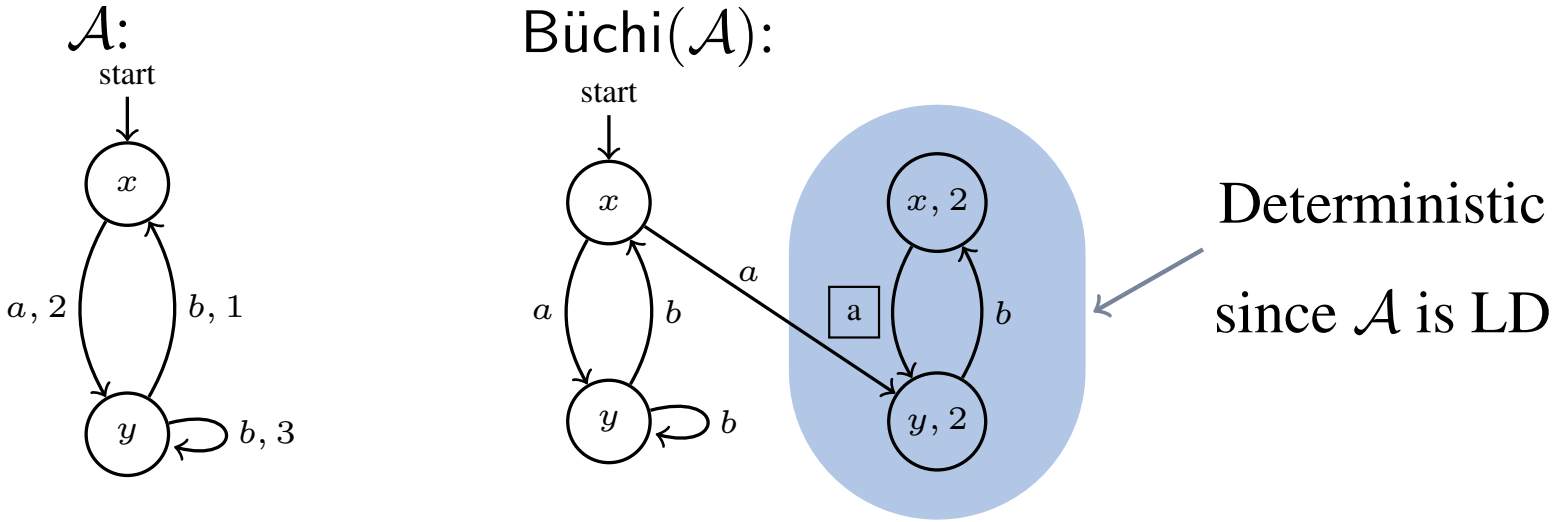
Büchi automaton is LD if reach(F) is deterministic.

LD parity automata to LD Büchi automata

Lemma

LDPA $\mathcal{A}$ of size n with k priorities can be transformed to LDBA Büchi($\mathcal{A}$) of size $\mathcal{O}(nk)$ and with $\mathrm{L}(\mathcal{A}) = L(\text{Büchi}(\mathcal{A}))$.

Eventually guess even highest priority. E.g.:



Determinizing LDBA

Safra/Piterman construction uses Safra trees as macrostates. Nodes in such trees are named and have sets of states with same origin state(s) as labels.

Slight modification: One new child node in Safra tree for each accepting state instead of joint child node for all accepting states. Then labels of new nodes are singleton sets.

In LDBA, they also **stay** singleton since tracked states evolve deterministically; then the tree structure of Safra trees collapses, **permutation structure** remains.

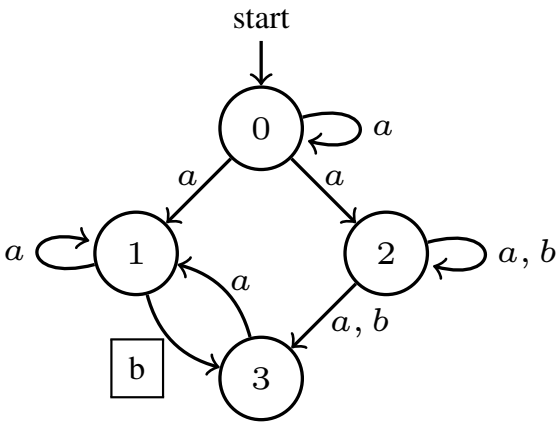
Determinizing LDBA, example

Theorem [Esparza, Kretínský, Raskin, Sickert, TACAS 2017]

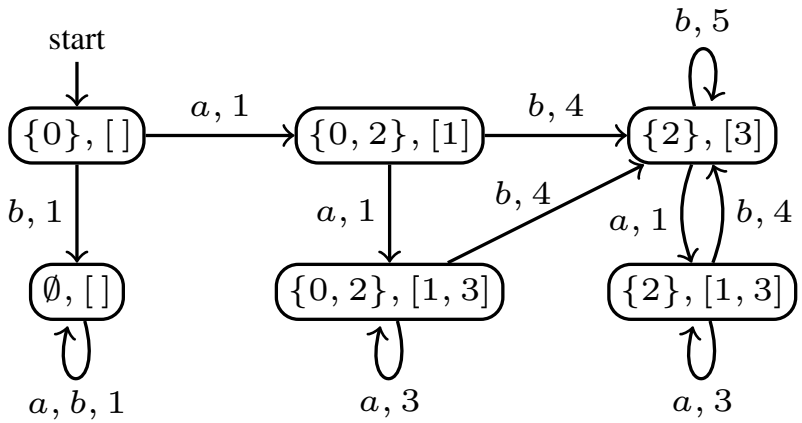
LDBA $\mathcal{B}$ of size n can be determinized to DPA $\mathrm{Det}(\mathcal{B})$ of size $\mathcal{O}(n!)$ and with $\mathrm{L}(\mathcal{B}) = L(\mathrm{Det}(\mathcal{B}))$.

Safraless determinization, using **permutations** of states. E.g.:

$\mathcal{B}$:



$\mathrm{Det}(\mathcal{B})$:



The μ-calculus

The μ-calculus

Formulas $\psi, \phi := \bot \mid \top \mid p \mid \neg p \mid \psi \wedge \phi \mid \psi \vee \phi \mid$
$\langle a \rangle \psi \mid [a]\psi \mid X \mid \mu X.\psi \mid \nu X.\psi \qquad p \in \mathbf{P}, X \in \mathbf{V}, a \in \mathbf{A},$

interpreted over Kripke structures $\mathcal{K} = (W, (R_a)_{a \in \mathbf{A}}, \tau)$; e.g.
$[\![\langle a \rangle \psi]\!]_\sigma = \{v \in W \mid \exists v R_a w. w \in [\![\psi]\!]_\sigma\}$ for $\sigma : \mathbf{V} \to \mathcal{P}(W)$

Aconjunctive formulas: in conjunctions $\psi_1 \wedge \psi_2$, at most one ψ_i contains an **active μ-variable**, i.e. a variable that can be transformed to a formula containing a free least fixpoint variable by (repeatedly) replacing variables with their binding fixpoint.

Weak aconjunctivity [Walukiewicz, 2000] relaxes this.

Assume **guardedness**.

Tracking automata

The **tracking automaton** $\mathcal{A}_\psi$ for formula ψ, is parity automaton with Fischer-Ladner closure [Kozen, 1983] of ψ as set of states and applications of tableau rules as letters, tracks single formulas through applications of tableau rules. Priorities are given by **alternation depth** of unfolded fixpoints, assigning odd priorities to greatest fixpoints, even priorities to least fixpoints.

The tracking automaton then accepts **bad branches**, i.e. infinite paths through pre-tableaux on which some least fixpoint is unfolded infinitely often.

Lemma

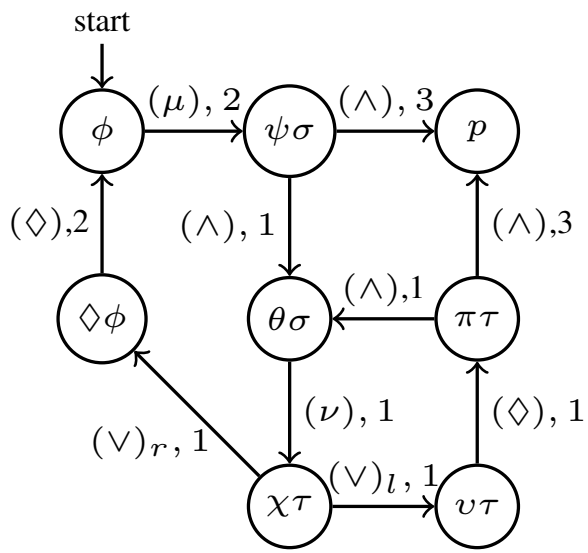
If ψ is weakly aconjunctive, then the tracking automaton $\mathcal{A}_\psi$ is limit-deterministic.

Tracking automata, example

$$\phi = \mu X.(p \wedge \nu Y.(\Diamond(Y \wedge p) \vee \Diamond X))$$

$$\phi := \mu X.\psi \qquad \psi := p \wedge \theta$$
$$\theta := \nu Y.\ \chi \qquad \chi := \upsilon \vee \Diamond X$$
$$\upsilon := \Diamond \pi \qquad \pi := Y \wedge p$$
$$\sigma := [X \mapsto \phi] \qquad \tau := [Y \mapsto \theta]; \sigma$$

$\mathcal{A}_\phi$:



$$(\mu)\ \frac{\textbf{A: } \phi}{\textbf{B: } \psi\sigma} \qquad (\wedge)\ \frac{\textbf{B: } \psi\sigma}{\textbf{C: } p, \theta\sigma} \qquad (\nu)\ \frac{\textbf{C: } p, \theta\sigma}{\textbf{D: } p, \chi\tau}$$

$$(\vee)\ \frac{\textbf{D: } p, \chi\tau}{\textbf{E: } p, \upsilon\tau} \qquad (\Diamond)\ \frac{\textbf{E: } p, \upsilon\tau}{\textbf{F: } \pi\tau} \qquad (\wedge)\ \frac{\textbf{F: } \pi\tau}{\textbf{G: C}}$$

$$(\Diamond)\ \frac{\textbf{H: } p, \Diamond\phi}{\textbf{I: A}}$$

Parity Games

- $\mathcal{G} = (V, E, \alpha)$, where $E \subseteq V \times V$, $\alpha : E \to \mathbb{N}$, $V = V_\exists \cup V_\forall$
- **plays** $\rho \in V^\omega$ s.t. for all i, $\rho_i E \rho_{i+1}$.
- **history-free strategies** $s : V_\exists \to V$, $t : V_\forall \to V$

Parity condition: Eloise wins play ρ if $Inf(\alpha \circ \text{trans}(\rho))$ is even
'Highest priority that occurs infinitely often is even'

Solving games is to compute the winning regions of the two players.

Satisfiability games

Permutation games for the aconjunctive μ-calculus

Input: Aconjunctive formula ψ

1. Tracking automaton $\mathcal{A}_\psi$ is **LDPA** of size $n = |\psi|$ with $k = \mathrm{ad}(\psi)$ priorities, recognizes **bad branches** in pre-tableaux for ψ.
2. Determinize $\mathcal{A}_\psi$ using permutation method, obtaining equivalent DPA $\mathcal{B}_\psi$ of size $\mathcal{O}((nk)!)$ and with $\mathcal{O}(nk)$ priorities.
3. Complement DPA $\mathcal{B}_\psi$, obtaining DPA $\mathcal{C}_\psi$ of same size.
4. Remove edges from $\mathcal{C}_\psi$ that do not correspond to matching applications of tableau rules.
5. Solve resulting parity game on states of $\mathcal{C}_\psi$ in time $\mathcal{O}((nk)!^{nk})$.

Build the game step by step and solve it **on-the-fly**, using the fixpoint iteration algorithm for parity games, see e.g. [Bruse, Falk, Lange, 2014].

The aconjunctive coalgebraic μ-calculus

The **coalgebraic** μ-calculus [Cîrstea, Kupke, Pattinson, 2011] generalizes the modal μ-calculus.

Models: instead of Kripke structures, T-coalgebras for a functor T

Permutation games work for the aconjunctive coalgebraic μ-calculus.

Examples:

- aconjunctive alternating-time μ-calculus – $\mu X.\, \psi \vee [D]X$
- aconjunctive probabilistic fixpoint logic – $\mu X.\, \psi \vee L_p X$
- aconjunctive game logic – $\mu X.\, \psi \vee \langle \alpha \rangle X$

New bound on model sizes for aconjunctive μ-calculi: $\mathcal{O}((nk)!)$.

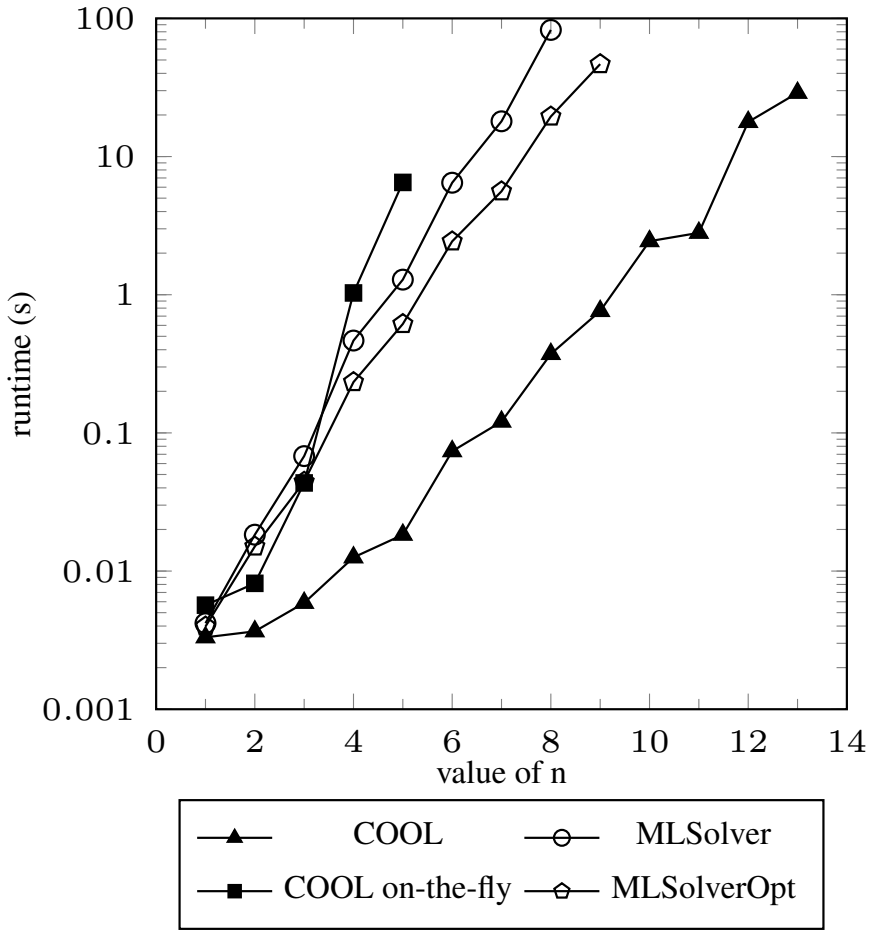
Implementation

Permutation games have been implemented as part of the **Coalgebraic Ontology Logic Reasoner** (COOL):

`https://www8.cs.fau.de/research:software:cool`

Though COOL supports various coalgebraic logics, we concentrate on the standard μ-calculus to compare COOL with MLSolver (which supports the full relational μ-calculus) on various series of aconjunctive formulas.

Results

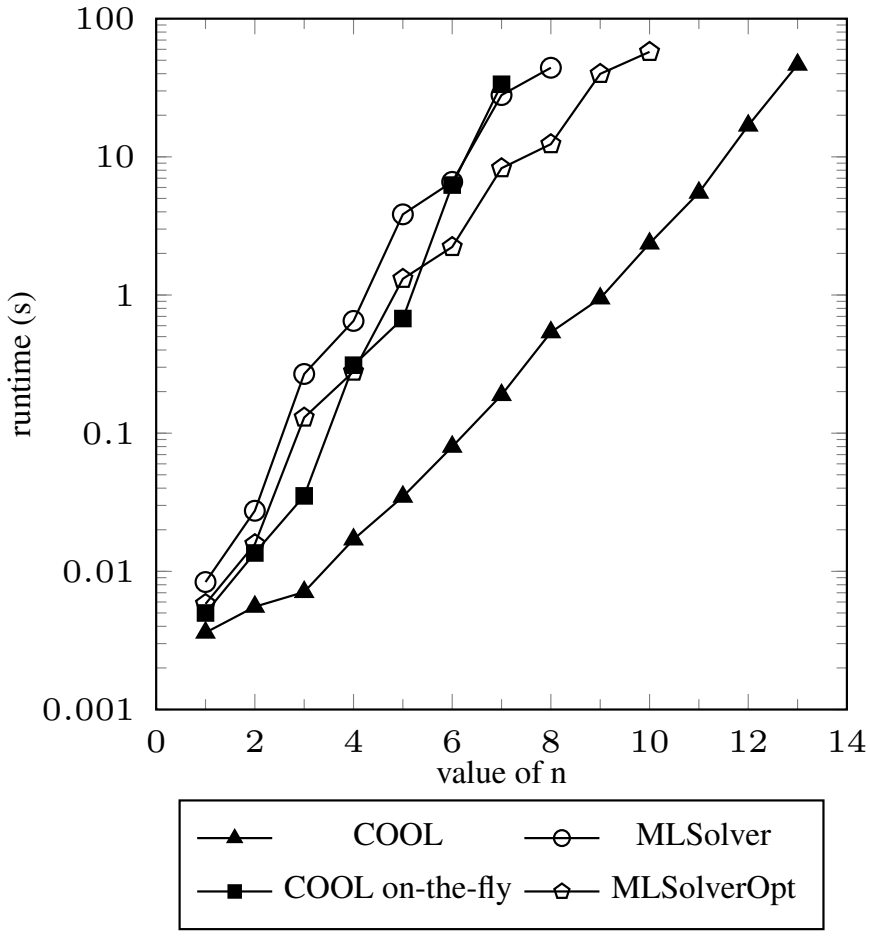


$\theta_1(n) := \phi_{\mathsf{aut}}(n) \to (\phi_{\mathsf{ne}}(n) \leftrightarrow \bigvee_{i \text{ even}} \mu X.\nu Y.\mu Z.\ \theta_{\Diamond}(i))$

$\phi_{\mathsf{aut}}(n) = \mathsf{AG}(\bigvee_{1 \le i \le n}(q_i \wedge \bigwedge_{j \ne i} \neg q_j)) \qquad \phi_{\mathsf{ne}}(n) = \eta X_n. \ldots .\nu X_2.\mu X_1.\ \bigvee_{1 \le i \le n}(q_i \wedge \Diamond X_i)$

$\theta_{\heartsuit}(i) = (q_i \wedge \heartsuit Y) \vee \bigvee_{i < j \le n}(q_j \wedge \heartsuit X) \vee \bigvee_{1 \le j \le i}(q_j \wedge \heartsuit Z)$

Results

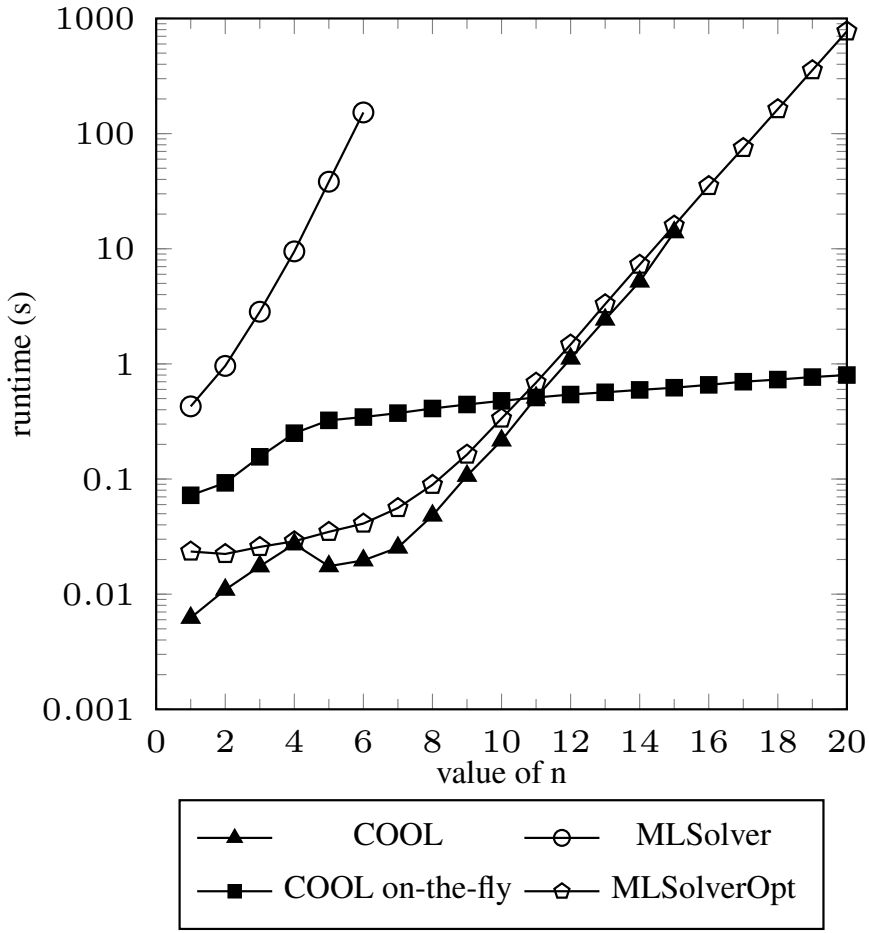


$\theta_2(n) := \phi_{\mathsf{game}}(n) \to (\phi_{\mathsf{win}}(n) \to \bigwedge_{i \text{ odd}} \nu X.\mu Y.\nu Z.\ \phi_{\mathsf{strat}}(\theta_{\heartsuit}(i)))$

$\phi_{\mathsf{game}}(n) = \phi_{\mathsf{aut}}(n) \wedge \mathsf{AG}((q_e \wedge \neg q_a) \vee (\neg q_e \wedge q_a))$

$\phi_{\mathsf{win}}(n) = \eta X_n. \ldots . \nu X_2.\mu X_1.(q_e \wedge \bigvee_{1 \leq i \leq n}(q_i \wedge \Diamond X_i)) \vee (q_a \wedge \bigvee_{1 \leq i \leq n}(q_i \wedge \Box X_i))$

Results

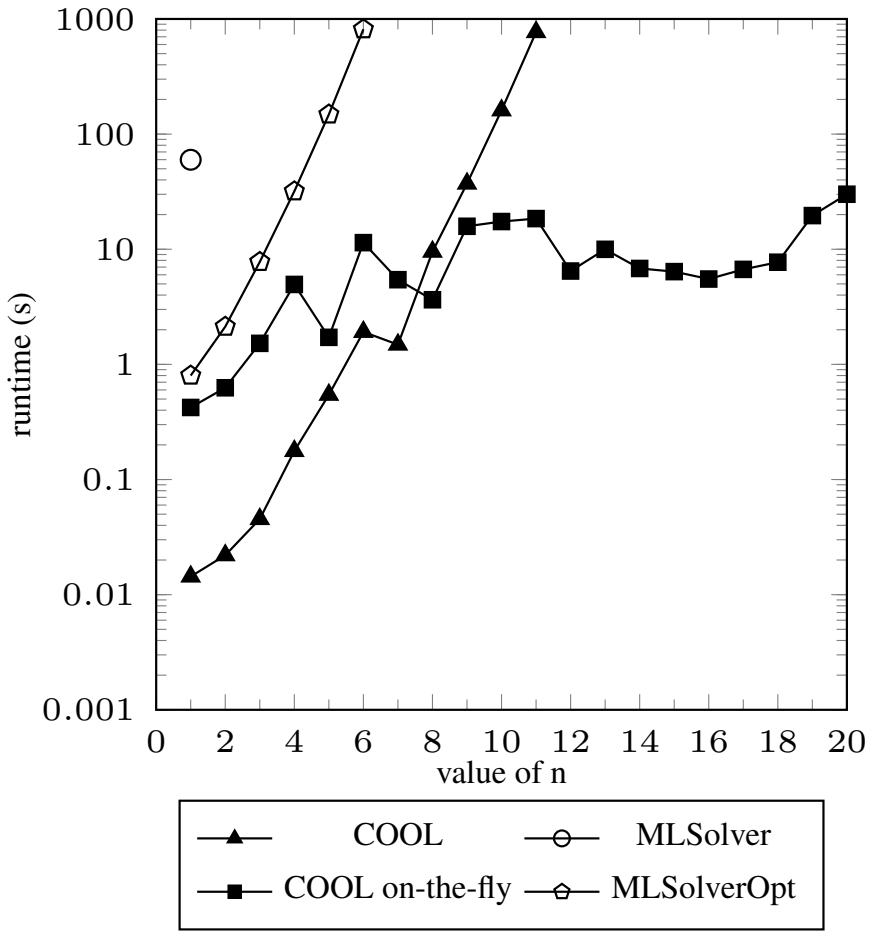


$$\text{early-ac}(n,j,k) = \mathit{start}_p \wedge \text{init}(p,n) \wedge \text{init}(r,k) \wedge \text{AG}\,((r \to \text{c}(r,k)) \wedge (p \to \text{c}(p,n))) \wedge \text{AG}\,((\bigwedge_{0\le i\le j} p_i \to \Diamond(\mathit{start}_r \wedge \theta)) \wedge \neg(p \wedge r) \wedge (r \to \Box\, r))$$

$$\text{init}(x,m) = \text{AG}\,((\mathit{start}_x \to (x \wedge \bigwedge_{0\le i<m} \neg x_i)) \wedge (x \to \Diamond\, x))$$

$$\theta = \eta X_{(2^k)}. \ldots . \nu X_2 . \mu X_1 . \bigvee_{1\le i\le 2^k} (\text{bin}(r, i-1) \wedge \Diamond X_i)$$

Results



$$\text{early-ac}_{gc}(n,j,k) = \text{early-ac}(n,j,k) \wedge b \wedge \text{init}(q,n) \wedge \text{AG}\,(\neg(p \wedge q) \wedge \neg(q \wedge r)) \wedge \text{AG}\,((q \rightarrow c(q,n)) \wedge \text{AF}\; b \wedge (b \rightarrow (\Diamond\, p \wedge \Diamond\, start_q \wedge \Box\, \neg b)))$$

Conclusion

- Determinization of **limit-deterministic** PA of size n with k priorities through limit-deterministic BA to DPA of size $\mathcal{O}((nk)!)$ with $\mathcal{O}(nk)$ priorities, using **permutation method**.
- Tracking automata for **weakly aconjunctive** μ-calculus formulas are limit-deterministic; **permutation games** of size $\mathcal{O}((nk)!)$ for aconjunctive formulas of size n and alternation depth k (down from $\mathcal{O}((nk)!^2)$).
- Permutation games work for the **coalgebraic** μ-calculus (covering e.g. alternating-time and probabilistic fixpoint logics, and game logic).
- Implemented in satisfiability solver COOL, supporting **on-the-fly** solving; implementation shows promising performance.