State Space Reduction For Parity Automata

Andreas Tollkötter
Supervisor: Dr. Christof Löding

January 11, 2019

Overview

We establish the notion of **merger functions**. Using that definition, we present three of our newly developed heuristic techniques to reduce the number of states in deterministic parity automata.

Overview

We establish the notion of **merger functions**. Using that definition, we present three of our newly developed heuristic techniques to reduce the number of states in deterministic parity automata.

1. Deterministic Parity Automata
2. Why do we need heuristic reduction?
3. Merger functions as a framework
4. Delayed Simulation
5. Congruence Path Refinement
6. Labeled SCC Filter

Table Of Contents

ω-automata

ω-words are words of one-sided infinite length: $\alpha \in \Sigma^\omega \Leftrightarrow \alpha : \mathbb{N} \to \Sigma$
a^ω, $aa(ba)^\omega$, $(abc)^\omega$

ω-automata are finite transition structures that describe a language $L(\mathcal{A}) \subseteq \Sigma^\omega$
$\{a^n b^\omega \mid n \in \mathbb{N}\}$

Deterministic parity automata (DPA):

- State set Q
- Alphabet Σ
- Transition function $\delta : Q \times \Sigma \to Q$
- Priority function $c : Q \to \mathbb{N}$

An ω-word α starting in a state $q_0 \in Q$ induces a run $q_0 q_1 q_2 \ldots$. The DPA accepts α iff the **smallest** priority that occurs infinitely often in the sequence $c(q_0)c(q_1)c(q_2)\ldots$ is **even**.

Table Of Contents

Why do we need heuristic reduction?

Goal: Reduce number of states in the automaton to ease run time of follow up algorithms.
Minimization Problem: Given an automaton $\mathcal{A}$, what is the smallest number of states required to recognize the same language as $\mathcal{A}$?
For deterministic finite automata (on finite words): Minimization is solvable in $\mathcal{O}(n \log n)$.
For DPAs: Minimization is NP-hard. []

Moore Minimization

A DPA can be interpreted as a Moore automaton with c being the output function.

Theorem.

Deterministic Moore automata can be minimized in log-linear time.

Idea: Compute equivalence $\equiv_M$ with $p \equiv_M q$ iff $\forall w \in \Sigma^* : c(\delta^*(p, w)) = c(\delta^*(q, w))$. Build the quotient automaton w.r.t. $\equiv_M$.
The same algorithm can be used to reduce DPAs but will not give minimal DPAs in general.

Table Of Contents

Merger functions

Definition.

Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be a DPA. A **merger function** is a function $\mu : D \to 2^Q \setminus \{\emptyset\}$ such that

- all sets in D are pairwise disjoint
- for all $X \in D$, $\mu(X) \cap (U \setminus X) = \emptyset$, where $U = \bigcup D$

$\mu(M) = C$
Merge all states in $M \subseteq Q$ into any one representative of $C \subseteq Q$.

For a congruence relation $\sim$, the quotient automaton is defined by state set $Q_\sim = \{[q]_\sim \mid q \in Q\}$.
This is captured by the merger function $\mu_\div : Q_\sim \to 2^Q, \kappa \mapsto \kappa$.

Table Of Contents

Delayed Simulation

Definition.

$p \equiv_{\mathrm{de}} q$ iff for all $w \in \Sigma^*$, every run that starts in $\delta^*(p, w)$ or $\delta^*(q, w)$ eventually sees a priority of at most $\min\{c(\delta^*(p, w)), c(\delta^*(q, w))\}$.

Definition.

Let $\mathfrak{C}_{\mathrm{de}} = \{[q]_{\equiv_{\mathrm{de}}} \mid q \in Q\}$ be the set of $\equiv_{\mathrm{de}}$-equivalence classes. Define the **delayed simulation merger** as
$\mu_{\mathrm{de}} : \mathfrak{C}_{\mathrm{de}} \to 2^Q, \kappa \mapsto \{q \in \kappa \mid c(q) = \min c(\kappa)\}$.

Theorem.

Merging states according to μ_{de} preserves language.

Computing Delayed Simulation

We define a deterministic Büchi automaton $\mathcal{G}_{\mathrm{de}}$ such that $p \equiv_{\mathrm{de}} q$ iff both $L(\mathcal{G}_{\mathrm{de}}, q^0_{\mathrm{de}}(p, q))$ and $L(\mathcal{G}_{\mathrm{de}}, q^0_{\mathrm{de}}(q, p))$ are universal, i.e. Σ^ω.
This automaton uses the state set $Q_{\mathrm{de}} = Q \times Q \times (c(Q) \cup \{\checkmark\})$.
Computing states of universal language in a DBA requires linear time.

Theorem.

$\equiv_{de}$ can be computed in $\mathcal{O}(n^2 k)$.

Delayed Simulation Automaton

$\mathcal{G}_{\text{de}} = (Q_{\text{de}}, \Sigma, \delta_{\text{de}}, F_{\text{de}})$
States are $Q_{\text{de}} = Q \times Q \times (c(Q) \cup \{\checkmark\})$.
The first two components are a "simulation" of the original DPA. The third component are the so called "obligations".
Transitions δ_{de}.
The first two components mimic the transitions of $\mathcal{A}$. The third component is defined by $\gamma : Q_{\text{de}} \times \Sigma \to c(Q) \cup \{\checkmark\}$. (next slide)
Accepting states are $F_{\text{de}} = Q \times Q \times \{\checkmark\}$.

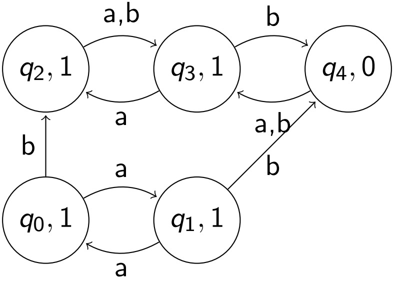
Delayed Simulation Automaton: γ

Let $0 \leq_\checkmark 1 \leq_\checkmark 2 \leq_\checkmark \cdots \leq_\checkmark \checkmark$.
For $p, q \in Q$, $k \in c(Q) \cup \{\checkmark\}$, $a \in \Sigma$, set
$\gamma((p, q, k), a) = \gamma'(\delta^*(p, a), \delta^*(q, a), k)$, where γ' is defined as follows:
If any of the following is true, then $\gamma'(i, j, k) = \checkmark$.

- i is odd, j is even, and $i \leq_\checkmark k$
- i is odd, j is even, and $j \leq_\checkmark k$
- i is odd, j is odd, $j \geq i$, and $i \leq_\checkmark k$
- i is even, j is even, $j \leq i$, and $j \leq_\checkmark k$

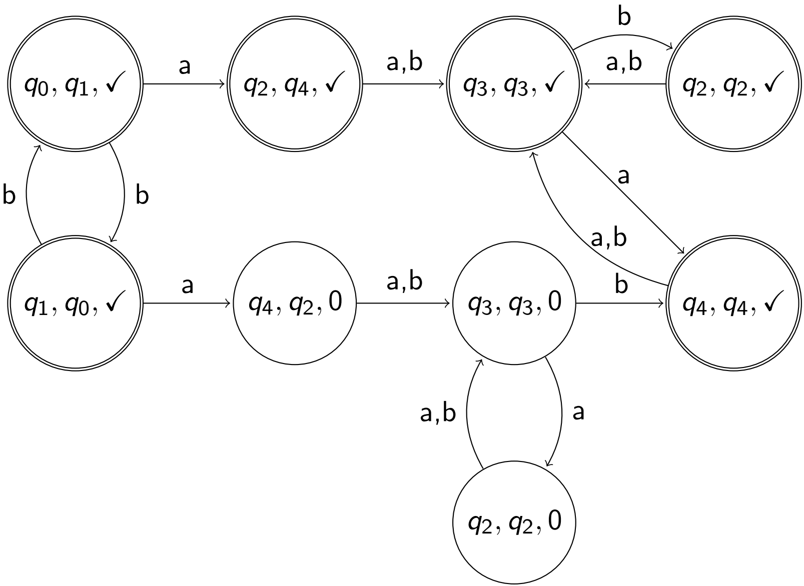
Otherwise, $\gamma'(i, j, k) = min_{\leq_\checkmark}\{i, j, k\}$.
$q^0_{\mathrm{de}}(p, q) = (p, q, \gamma'(c(p), c(q), \checkmark))$.

Delayed Simulation Automaton

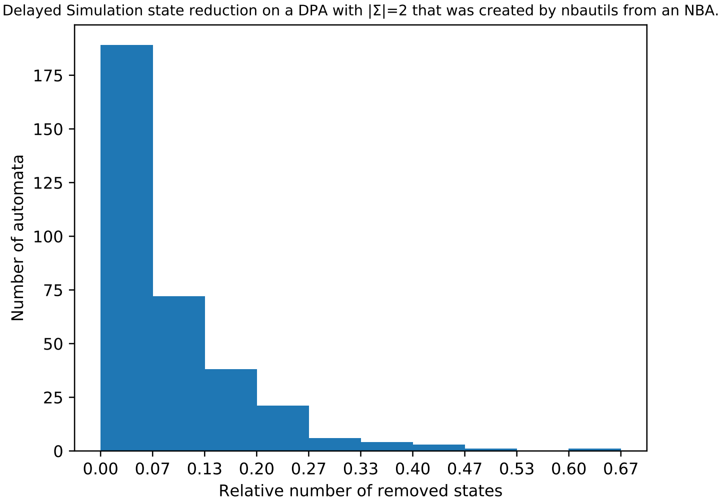


A DPA with 5 states. We want to check whether $q_0 \equiv_{\mathrm{de}} q_1$ is true.

Delayed Simulation Automaton



Efficiency



Delayed Simulation state reduction on a DPA with |Σ|=2 that was created by nbautils from an NBA.
Number of automata
0
25
50
75
100
125
150
175
0.00
0.07
0.13
0.20
0.27
0.33
0.40
0.47
0.53
0.60
0.67
Relative number of removed states

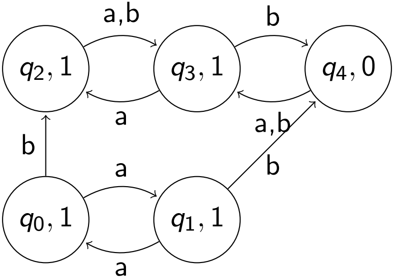
Table Of Contents

Congruence Path Refinement

Definition.

Let $\sim$ be a congruence relation and let $\lambda \subseteq Q$ be an equivalence class of $\sim$.
We define $L_{\lambda\hookleftarrow} \subseteq \Sigma^*$ as the set of all words such that the induced run from a state in λ moves back to λ exactly once and ends there.
The **path refinement** equivalence $\equiv^{\lambda}_{\mathrm{PR}}$ is the largest relation such that if $p \equiv^{\lambda}_{\mathrm{PR}} q$, then for all $w \in L_{\lambda\hookleftarrow}$, $\delta^*(p, w) \equiv^{\lambda}_{\mathrm{PR}} \delta^*(q, w)$ and the smallest priority seen when reading w is the same from p and from q.

Path Refinement Relation



Potential choices for λ are the equivalence classes of $\equiv_L$:
$\{q_0, q_1\}$, $\{q_2, q_4\}$, or $\{q_3\}$.

Path Refinement Merger

Definition.

Let $\mathfrak{C}^{\lambda}_{\mathsf{PR}} = \{[q]_{\equiv^{\lambda}_{\mathsf{PR}}} \mid q \in Q\}$ be the set of $\equiv^{\lambda}_{\mathsf{PR}}$-equivalence classes. Define the **path refinement merger** as
$\mu^{\lambda}_{\mathsf{PR}} : \mathfrak{C}^{\lambda}_{\mathsf{PR}} \to 2^{Q}, \kappa \mapsto \{q \in \kappa \mid c(q) = \min c(\kappa)\}$.

Theorem.

If all states in λ are pairwise language equivalent, merging states according to $\mu^{\lambda}_{\mathsf{PR}}$ preserves language.

Computing Path Refinement

Definition.

Define the **visit graph** DPA $\mathcal{A}^\lambda_{\text{visit}} = (Q^\lambda_{\text{visit}}, \Sigma, \delta^\lambda_{\text{visit}}, c^\lambda_{\text{visit}})$.

- $Q^\lambda_{\text{visit}} = ((Q \setminus \lambda) \times c(Q) \times \{-1\}) \cup (\lambda \times c(Q) \times c(Q))$.
- $\delta^\lambda_{\text{visit}}((q,k,k'),a) = \begin{cases} (q', \min\{k, c(q')\}, -1) & \text{if } q' \notin \lambda \\ (q', c(q'), \min\{k, c(q')\}) & \text{if } q' \in \lambda \end{cases}$, where $q' = \delta(q,a)$.
- $c^\lambda_{\text{visit}}((q,k,k')) = k'$.

States consist of three components $q \in Q \times c(Q) \times (c(Q) \cup \{-1\})$.
The first component "simulates" the original automaton $\mathcal{A}$.
The second component tracks the minimal priority seen on one run from λ to λ.
The third component is required to distinguish the different priorities.

Computing Path Refinement

Moore equivalence in $\mathcal{A}^{\lambda}_{\mathsf{visit}}$ corresponds to path refinement equivalence in $\mathcal{A}$.

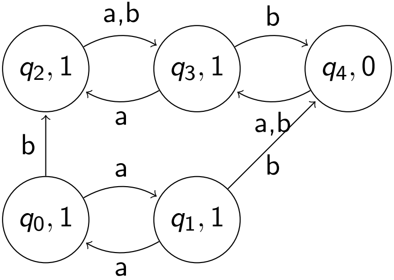
Theorem.

$p \equiv^{\lambda}_{PR} q$ iff $(p, c(p), \max c(Q)) \equiv_M (q, c(q), \max c(Q))$.

Theorem.

$\equiv^{\lambda}_{PR}$ can be computed in $\mathcal{O}(k^2 n \log n)$.

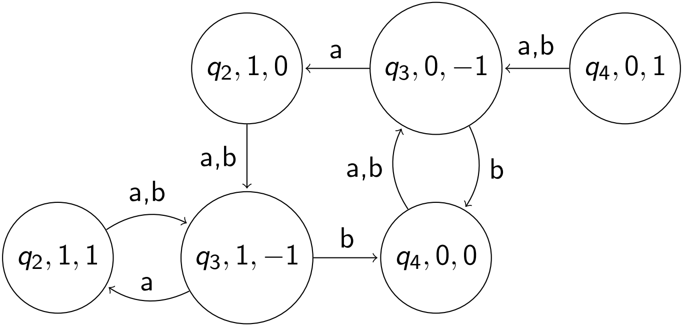
Visit Graph



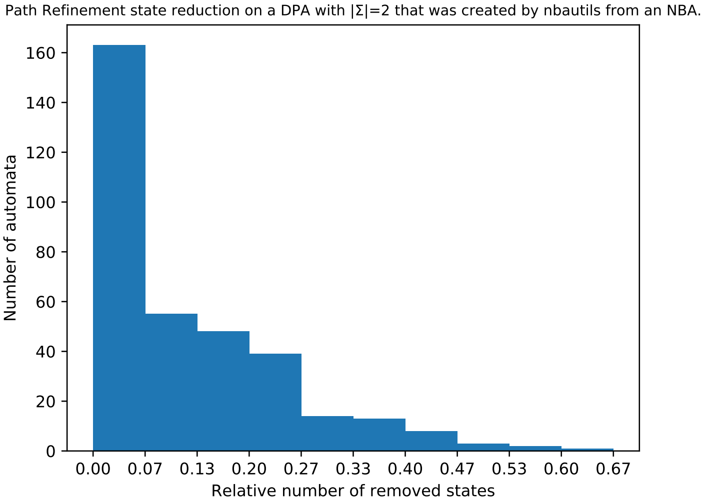
Potential choices for λ are the equivalence classes of $\equiv_L$:
$\{q_0, q_1\}$, $\{q_2, q_4\}$, or $\{q_3\}$.

Visit Graph

$\mathcal{A}_{\text{visit}}^{\{q_2,q_4\}}$



Efficiency



Path Refinement state reduction on a DPA with |Σ|=2 that was created by nbautils from an NBA.
160
140
120
100
80
60
40
20
0
Number of automata
0.00 0.07 0.13 0.20 0.27 0.33 0.40 0.47 0.53 0.60 0.67
Relative number of removed states

Table Of Contents

Labeled SCC Filter

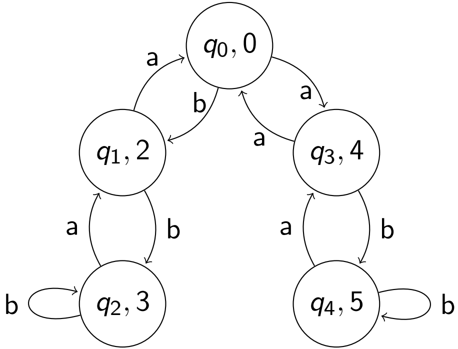
Definition.

Define $\equiv_M^{\leq k}$ as the Moore equivalence that considers all priorities **greater than k** to be equal.
$p \equiv_M^{\leq k} q$ iff for all $w \in \Sigma^*$: $c(\delta^*(p, w)) = c(\delta^*(q, w))$ or $k < c(\delta^*(p, w)), c(\delta^*(q, w))$.

Definition.

Let $\sim$ be an equivalence relation and $k \in \mathbb{N}$. We define the LSF relation $p \equiv_{\mathrm{LSF}}^{k,\sim} q$ iff $p \sim q$ and $p \equiv_M^{\leq k} q$.

LSF Relation



Equivalence classes of $\equiv_{\mathrm{LSF}}^{1,\equiv_L}$: $\{q_0\}$, $\{q_1, q_3\}$, and $\{q_2, q_4\}$.

LSF Merger

From the DPA $\mathcal{A}$, we remove all states which have priority less or equal to k and call the resulting (possibly incomplete) DPA $\mathcal{A}_k$.
We choose some total preorder $\preceq_k$ on the states of $\mathcal{A}_k$ such that p being reachable from q in $\mathcal{A}_k$ implies $q \preceq_k p$, and $p \preceq_k q \preceq_k p$ is only true if p and q are in the same SCC in $\mathcal{A}_k$. ($\preceq_k$ is a total preorder that is a minimal extension of reachability.)

In focus are the set of states that are $\preceq_k$-maximal among a given set $P \subseteq Q$. These are all states in one SCC of $\mathcal{A}_k$ such that no other states in P are reachable.

LSF Merger

Definition.

Let $\mathfrak{C}^{k,\sim}_{\mathsf{LSF}}$ be the set of equivalence classes in $\equiv^{k,\sim}_{\mathsf{LSF}}$ and let κ be such an equivalence class. Define

$$C^k_\kappa = \{r \in \kappa \mid c(r) > k \text{ and } r \text{ is } \preceq_k\text{-maximal among } \kappa\}$$

and $M^k_\kappa = \kappa \setminus C^k_\kappa$.

Definition.

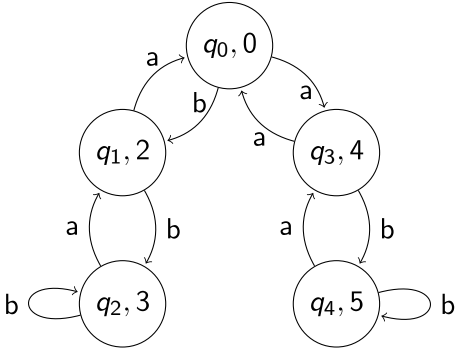
Define the **LSF merger function**

$$\mu^{k,\sim}_{\mathsf{LSF}} : \{M^k_\kappa \mid \kappa \in \mathfrak{C}^{k,\sim}_{\mathsf{LSF}}\} \to 2^Q, M^k_\kappa \mapsto C^k_\kappa$$

Theorem.

If $\sim$ implies language equivalence, merging states according to $\mu^{k,\sim}_{LSF}$ preserves language.

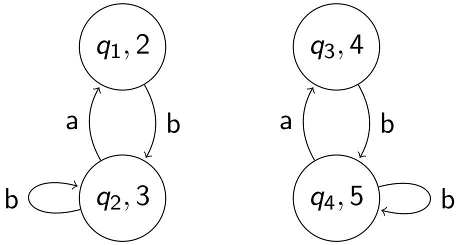
LSF example



Equivalence classes of $\equiv_{\mathsf{LSF}}^{1,\equiv_L}$: $\{q_0\}$, $\{q_1, q_3\}$, and $\{q_2, q_4\}$.

LSF example

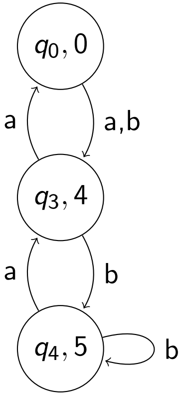
$\mathcal{A}_1$ variant of the automaton.



Possible order: $q_1 \simeq_1 q_2 \prec_1 q_3 \simeq_1 q_4$.
q_3 is the only $\preceq_1$-maximal element in $\{q_1, q_3\}$.
q_4 is the only $\preceq_1$-maximal element in $\{q_2, q_4\}$.
$\mu_{\mathsf{LSF}}^{1,\equiv_L}(\{q_1\}) = \{q_3\}$
$\mu_{\mathsf{LSF}}^{1,\equiv_L}(\{q_2\}) = \{q_4\}$

LSF example

After the merge:



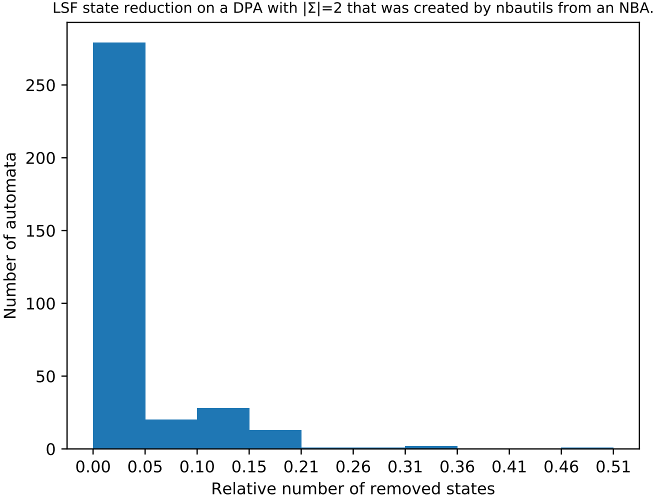
Computing LSF

The definition provides a straight-forward computation: $\equiv_M^{\leq k}$ is only a slight variation of the normal Moore equivalence and $\preceq_k$ can be computed with a topological sorting on the SCCs of $\mathcal{A}_k$.

Theorem.

$\mu_{LSF}^{k,\sim}$ can be computed in $\mathcal{O}(n \log n)$.

Efficiency



LSF state reduction on a DPA with |Σ|=2 that was created by nbautils from an NBA.
Number of automata
250
200
150
100
50
0
0.00
0.05
0.10
0.15
0.21
0.26
0.31
0.36
0.41
0.46
0.51
Relative number of removed states