Simplification of finite automata

Lorenzo Clemente (University of Warsaw)
based on joint work with Richard Mayr (University of Edinburgh)

Warsaw, November 2016

Nondeterministic finite automata

We consider classical NFA over finite words (like in Hopcroft & Ullman):

$$A = (\Sigma, Q, I, F, \Delta)$$

where

- Σ is a *finite* alphabet,
- Q is a *finite* set of states,
- $I \subseteq Q$ is a distinguished set of initial states,
- $F \subseteq Q$ is a distinguished set of final states, and
- $\Delta \subseteq Q \times \Sigma \times Q$ is the transition function.

The language of an NFA

We use an NFA A as a compact representation of a language L(A) over Σ:

$$L(A) = \{ w \in \Sigma^* \mid \exists p \in I, \exists q \in F \cdot p \text{ --}w\text{--> } q \}$$

where p --ε--> p, and p --aw--> q if $\exists$ r s.t. (p, a, r) $\in \Delta$ and r --w--> q.

Algorithmic problems for NFA

Constructions: Given NFAs A and B,

- Union: Construct C s.t. $L(C) = L(A) \cup L(B)$. -- linear
- Intersection: Construct C s.t. $L(C) = L(A) \cap L(B)$. -- quadratic
- Complement: Construct C s.t. $L(C) = \Sigma^* \setminus L(A)$. -- exponential

Decision problems:

- Non-emptiness: Given an NFA A, is $L(A) \neq \emptyset$? -- LOGSPACE
- Universality: Given an NFA A, is $L(A) = \Sigma^*$? -- PSPACE-c.
- Inclusion: Given two NFAs A and B, is $L(A) \subseteq L(B)$? -- PSPACE-c.
- Equivalence: Given two NFAs A and B, is $L(A) = L(B)$? -- PSPACE-c.

Applications of NFA

NFA is a data structure representing a language of finite words.

- Decision procedures for logical theories: Boolean operations;
  $\exists$ projection, $\wedge$ intersection, $\vee$ disjunction, $\neg$ complement.
- Reachability analysis of safe Petri nets: Boolean operations.
- Comparing the control-flow of programs: Inclusion problem.
- Program termination analysis: Inclusion problem.
- Conformance of an implementation w.r.t. a specification: Inclusion.
- …

SMALLER IS BETTER

Minimal automata

An NFA is *minimal* if it has the smallest number of states
w.r.t. all NFAs recognizing the same language.

- Minimal NFAs are *not unique*.
  There exist non-isomorphic equivalent minimal NFAs.

- Finding a minimal automaton equivalent to a given NFA A is complicated.

- Checking whether an NFA is minimal is very expensive (PSPACE-c.).

Simplification of automata

Given an NFA A, we want to *efficiently* construct a *smaller* NFA B s.t. L(A) = L(B). (Not necessarily a minimal one, which is expensive.)

There is an algorithmic tradeoff between the quality of reduction (i.e., smaller output) and the cost of achieving it.

Simplification spectrum:

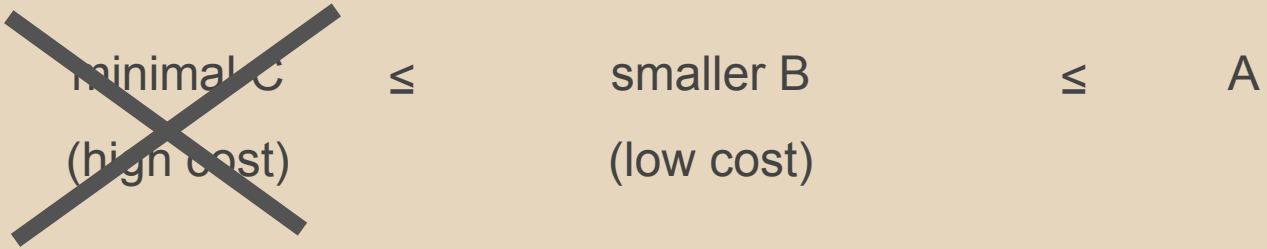
minimal C	≤	smaller B	≤	A
(high cost)		(low cost)		

Simplification of automata

Given an NFA A, we want to *efficiently* construct a *smaller* NFA B s.t. L(A) = L(B). (Not necessarily a minimal one, which is expensive.)

There is an algorithmic tradeoff between the quality of reduction (i.e., smaller output) and the cost of achieving it.

Simplification spectrum:

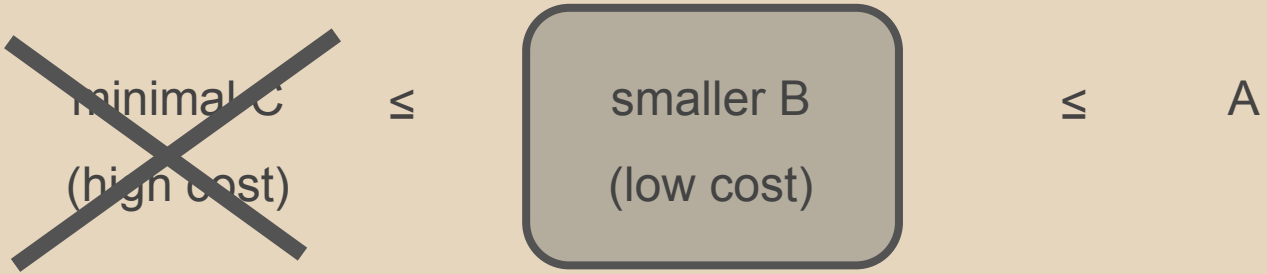


Simplification of automata

Given an NFA A, we want to *efficiently* construct a *smaller* NFA B s.t. L(A) = L(B). (Not necessarily a minimal one, which is expensive.)

There is an algorithmic tradeoff between the quality of reduction (i.e., smaller output) and the cost of achieving it.

Simplification spectrum:



Simplification #1: Remove dead states

- A state p is *dead* if either it is not reachable from I, or it cannot reach F.
- Clearly, a dead state does not contribute to the language.
- A trivial simplification is to get rid of dead states.

Simplification #2: Quotienting

Simplification #2: Quotienting

- Find an equivalence relation on states $\approx \subseteq Q \times Q$.
- Build the quotient automaton $A/\approx$ by collapsing equivalent states.
- This potentially reduces the number of states, and, consequently, the number of transitions.

- Only some equivalences $\approx$ preserve the language.
- If $L(A) = L(A/\approx)$, then $\approx$ is called

good for quotienting (**GFQ**)

Simplification #2: Quotienting

- Given an equivalence $\approx \subseteq Q \times Q$, the quotient NFA $A/\approx$ is defined as
$$(\Sigma, [Q], [I], [F], [\Delta]),$$
  - $[Q] = \{ [q] \mid q \in Q \}$,
  - $[I] = \{ [q] \mid q \in I \}$, $[F] = \{ [q] \mid q \in F \}$,
  - $[\Delta] = \{ ([p], a, [q]) \mid (p, a, q) \in \Delta \}$.

- If $L(A) = L(A/\approx)$ then $\approx$ is called

good for quotienting (**GFQ**)

Example GFQ equivalences

- The identity relation $\{(p, p) \mid p \in Q\}$ is GFQ and useless.
- Language equivalence $\{(p, q) \mid L(p) = L(q)\}$ is GFQ and PSPACE-c.
- We need coarse GFQ equivalences
  which can be computed *more efficiently*.

Solution: simulation relations.

Forward simulation

- A relation $R \subseteq Q \times Q$ is a *forward simulation* if $(p, q) \in R$ implies
  a. if $p \in F$, then $q \in F$, and
  b. for every transition $(p, a, p') \in \Delta$,
     there exists a transition $(q, a, q') \in \Delta$
     s.t. $(p', q') \in R$.
- The union of all forward simulations is also a forward simulation, called *forward simulation preorder* and denoted $\sqsubseteq_{FW}$.
- It approximates language inclusion: $p \sqsubseteq_{FW} q$ implies $L(p) \subseteq L(q)$.
- Efficient: computing $\sqsubseteq_{FW}$ can be done in PTIME.
- The induced equivalence ($\sqsubseteq_{FW} \cap \sqsupseteq_{FW}$) is GFQ.

Simulation vs. language inclusion

- Simulation approximates language inclusion: $p \sqsubseteq_{FW} q$ implies $L(p) \subseteq L(q)$.
- This is a *strict under-approximation*, i.e.,
  there are cases when $L(p) \subseteq L(q)$ holds but $\neg (p \sqsubseteq_{FW} q)$.
- Language inclusion is a property of the *global* behaviour of the automaton
  (i.e., it's about paths).
- Simulation is a property of the *local* behaviour of the automaton
  (i.e., it's about *single transitions*).
  - Not just whether you can accept a word or not,
    but also *how* you accept matters.
  - Incidentally, that's why it is PTIME to compute.

Backward simulation

- A relation $R \subseteq Q \times Q$ is a *backward simulation* if $(p, q) \in R$ implies
  a. if $p \in I$, then $q \in I$, and
  b. for every transition $(p', a, p) \in \Delta$,
     there exists a transition $(q', a, q) \in \Delta$
     s.t. $(p', q') \in R$.
- The union of all backward simulations is also a forward simulation, called *backward simulation preorder* and denoted $\sqsubseteq_{BW}$.
- It (strictly) under-approximates backward language inclusion.
- Efficient: computing $\sqsubseteq_{BW}$ can be done in PTIME.
- The induced equivalence $(\sqsubseteq_{BW} \cap \sqsupseteq_{BW})$ is GFQ.

Simplification algorithm #1

Consider the following simplification algorithm:

1. Remove dead states.
2. Quotient w.r.t. forward simulation $\sqsubseteq_{FW}$.
3. Quotient w.r.t. backward simulation $\sqsubseteq_{BW}$.
4. If the automaton changed, repeat from 1.

The order of performing 2. and 3. matters, but never too much.
(And there is no general recipe to suggest which one to take.)

Simplification #3: Transition pruning

Simplification #3: Transition pruning

- Find a preorder $\preccurlyeq \subseteq \Delta \times \Delta$ on transitions.
- Let Prune(A, $\preccurlyeq$) be the automaton obtained from A
  by removing a transition if it is $\preccurlyeq$-smaller than another transition.
- This reduces the number of transitions,
  which can result in new states becoming dead.
- Clearly, $L(\text{Prune}(A, \preccurlyeq)) \subseteq L(A)$ since we remove transitions.
- If $L(A) = L(\text{Prune}(A, \preccurlyeq))$, then $\preccurlyeq$ is called *good for pruning* (**GFP**).

Examples of GFP relations

- $(p, a, q) \leqslant_{FW} (p', a', q')$ iff $a = a'$, $p = p'$, and $q \sqsubseteq_{FW} q'$.
- $(p, a, q) \leqslant_{BW} (p', a', q')$ iff $a = a'$, $p \sqsubseteq_{BW} p'$, and $q = q'$.
- $(p, a, q) <_{BW\text{-}FW} (p', a', q')$ iff $a = a'$, $p \sqsubset_{BW} p'$, and $q \sqsubset_{FW} q'$.

(It might seem like a limitation to require the endpoints to be strict, but not really so since after iterated quotienting w.r.t. $\sqsubseteq_{FW}$ and $\sqsubseteq_{BW}$ these preorders are in fact partial orders, i.e., equivalence classes have size 1.)

Simplification algorithm #2

1. Remove dead states.
2. Quotienting:
   a. Quotient w.r.t. forward simulation $\sqsubseteq_{FW}$.
   b. Quotient w.r.t. backward simulation $\sqsubseteq_{BW}$.
   c. If the automaton changed, repeat from 2.
3. Prune transitions w.r.t. forward pruning $\preccurlyeq_{FW}$.
4. Prune transitions w.r.t. backward pruning $\preccurlyeq_{BW}$.
5. Prune transitions w.r.t. backward-forward pruning $\prec_{BW\text{-}FW}$.
6. If the automaton changed, repeat from 1.

Again the order matters, but never too much.

Next steps

- Find coarser GFQ and GFP relations:
  - Multipebble simulations ✓, lookahead simulations ✓.

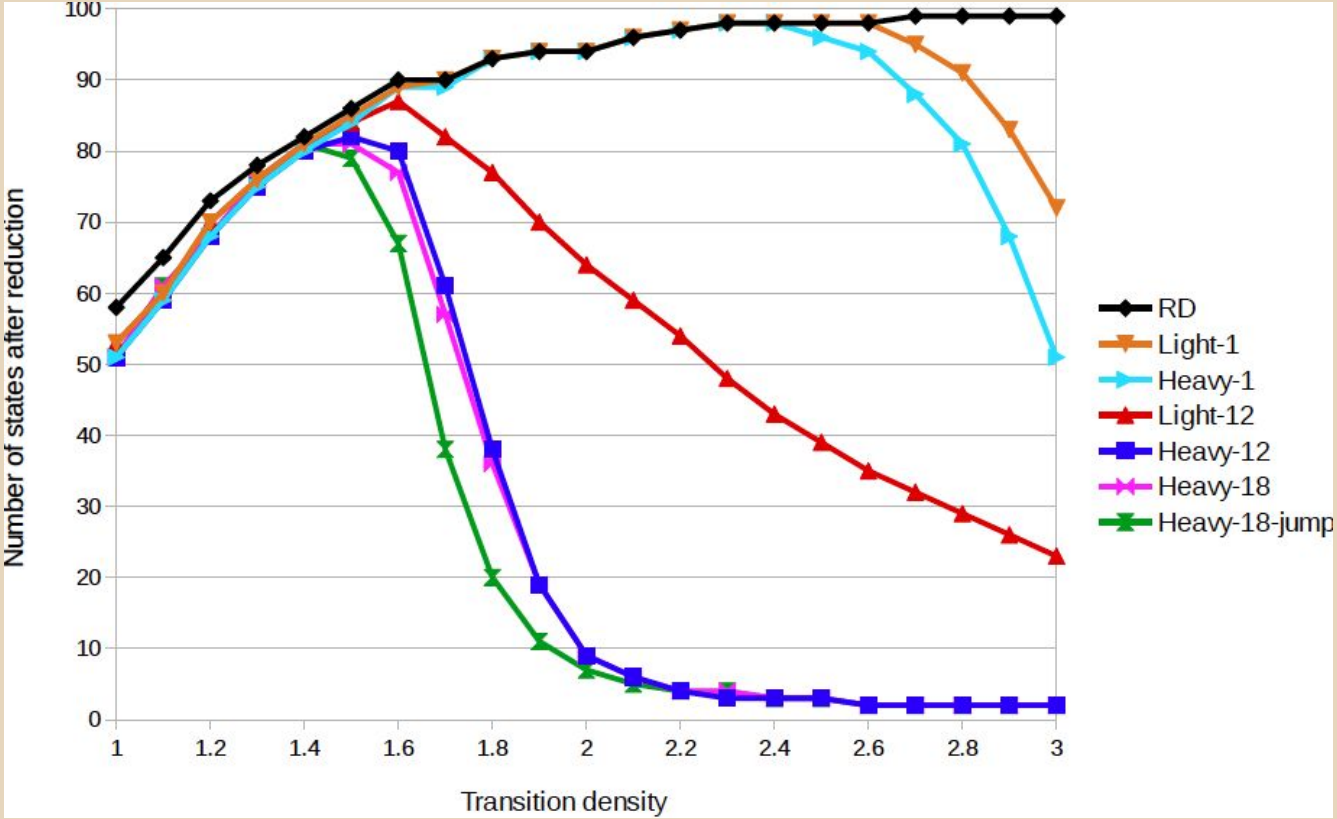
- Extend these techniques beyond NFAs:
  - Finite automata over infinite words ✓,
    over finite ✓ and infinite trees ✘.
  - Register automata ✘, timed automata ✘.
  - Restricted classes of counter automata (1-counter nets ✘, …).

Next steps

- Find coarser GFQ and GFP relations:
  - Multipebble simulations ✓, **lookahead simulations** ✓.

- Extend these techniques beyond NFAs:
  - Finite automata over infinite words ✓,
    over finite ✓ and infinite trees ✘.
  - Register automata ✘, timed automata ✘.
  - Restricted classes of counter automata (1-counter nets ✘, …).

Practical



Each point is the average of 1000 random automata with 100 states.

#transitions = td * 100
#accepting states = 50

Algorithms:

- RD: remove dead states.
- Light: RD + only one round of quotienting.
- Heavy: repeated RD + quotienting + pruning.

Algorithmic problems for DFA

Constructions: Given NFAs A and B,

- Union: Construct C s.t. $L(C) = L(A) \cup L(B)$. -- quadratic
- Intersection: Construct C s.t. $L(C) = L(A) \cap L(B)$. -- quadratic
- Complement: Construct C s.t. $L(C) = \Sigma^* \setminus L(A)$. -- linear

Decision problems:

- Non-emptiness: Given an NFA A, is $L(A) \neq \emptyset$? -- LOGSPACE
- Universality: Given an NFA A, is $L(A) = \Sigma^*$? -- PTIME
- Inclusion: Given two NFAs A and B, is $L(A) \subseteq L(B)$? -- PTIME
- Equivalence: Given two NFAs A and B, is $L(A) = L(B)$? -- PTIME