A new approach to NFA minimization

Jaco Geldenhuys

Department of Mathematical Sciences
Computer Science Division
Stellenbosch University

December 13, 2006

Disclaimer 1

Joint work with Lynette van Zijl and Brink van der Merwe

Disclaimer 2

This is a presentation about work-in-progress.

Outline

1. Supernondeterminism
2. Model checking
3. NFA minimization

Plain old finite automata

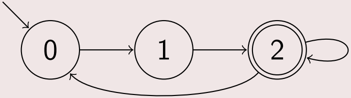
A nondeterministic finite automaton (NFA) is a tuple $M = (S, \Sigma, \Delta, \hat{s}, A)$ where

- S is a finite set of states,
- Σ is a finite alphabet,
- $\Delta \subseteq S \times \Sigma \times 2^S$ is the transition function,
- $\hat{s} \in S$ is the initial state, and
- $A \subseteq S$ is the set of accepting states.

Determinization

NFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02

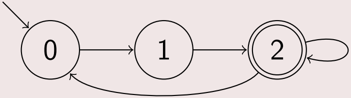


DFA

Determinization

NFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02



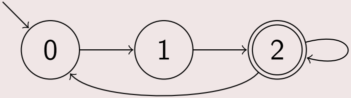
DFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02
$\leftarrow$	02	012
$\leftarrow$	012	012

Determinization

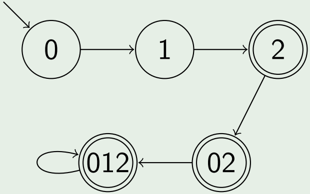
NFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02



DFA

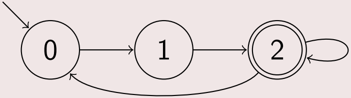
		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02
$\leftarrow$	02	012
$\leftarrow$	012	012



Determinization

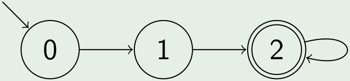
NFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02



DFA

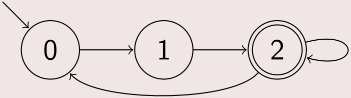
		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02
$\leftarrow$	02	012
$\leftarrow$	012	012



Determinization

NFA

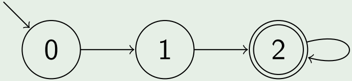
		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02



$$\mathcal{L} = \{a^k \mid k \geq 2\}$$

DFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02
$\leftarrow$	02	012
$\leftarrow$	012	012



Supernondeterminism

Supernondetermism is a *generalization* of nondetermnism.

A symmetric-difference nondet. finite automaton ($\oplus$-NFA) is a tuple $M = (S, \Sigma, \Delta, \hat{s}, A)$ where

- S is a finite set of states,
- Σ is a finite alphabet,
- $\Delta \subseteq S \times \Sigma \times 2^S$ is the transition function,
- $\hat{s} \in S$ is the initial state, and
- $A \subseteq S$ is the set of accepting states.

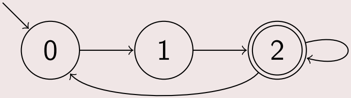
Exactly the same definition as for an NFA!

So, what's the point?

Determinization

$\oplus$-NFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02

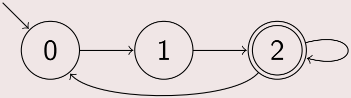


DFA

Determinization

⊕-NFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02



DFA

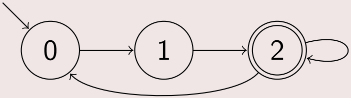
		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02
$\leftarrow$	02	012
$\leftarrow$	012	012

$$\{1\} \cup \{2\} \cup \{0, 2\}$$

Determinization

$\oplus$-NFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02



DFA

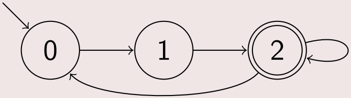
		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02
$\leftarrow$	02	012
$\leftarrow$	012	

$$\{1\} \oplus \{2\} \oplus \{0, 2\}$$

Determinization

$\oplus$-NFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02



DFA

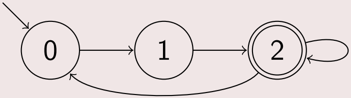
		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02
$\leftarrow$	02	012
$\leftarrow$	012	01

$$\{1\} \oplus \{2\} \oplus \{0, 2\}$$

Determinization

$\oplus$-NFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02



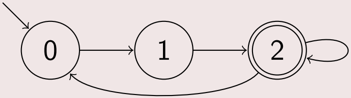
DFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02
$\leftarrow$	02	012
$\leftarrow$	012	01

Determinization

$\oplus$-NFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02

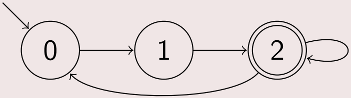


DFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02
$\leftarrow$	02	012
$\leftarrow$	012	01
	01	12
$\leftarrow$	12	0

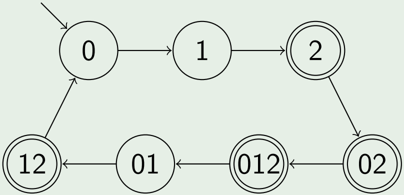
$\oplus$-NFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02



DFA

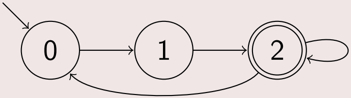
		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02
$\leftarrow$	02	012
$\leftarrow$	012	01
	01	12
$\leftarrow$	12	0



Determinization

$\oplus$-NFA

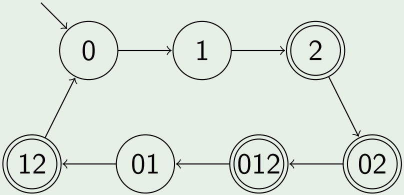
		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02



$$\mathcal{L} = \{a^{7k+m} \mid m = 2, 3, 4, 6\}$$

DFA

		a
$\rightarrow$	0	1
	1	2
$\leftarrow$	2	02
$\leftarrow$	02	012
$\leftarrow$	012	01
	01	12
$\leftarrow$	12	0



Supernondeterminism

- Plain old NFA $\Leftrightarrow$ $\cup$-NFA
- Same representation, different interpretation.

Theorem [Van Zijl 1997]

$\oplus$-NFA can be exponentially more succinct than $\cup$-NFA.

- $\cap$-NFA and other $\star$-NFA are *"not-so-hot"*
- $\oplus$-NFA related to linear feedback shift registers
- Succinctness: important implications for data representation

Model checking

- Given program P and property ϕ, does $P \models \phi$?
- P: formal description of reactive system
- ϕ: formula of propositional temporal logic

Examples

1. $P =$ description of avionic control system
   $\phi = \Box(H \geq 0)$
   aeroplane stays above sea level
2. $P =$ telecommunication protocol
   $\phi = \Box(\mathit{send} \Rightarrow \Diamond \mathit{recv})$
   if a message is sent, it is eventually received
3. $P =$ HDL of a cpu
   $\phi = \Box\Diamond \mathit{clk}$
   a clock signal occurs infinitely often

- First algorithms: Clarke & Emerson / Queille & Sifakis 1981

Model checking

- Progress over the last 25 years

1985 $\sim 10^4$ states

1995 $\sim 10^6$ states

2005 $\sim 10^8$ states

  - Figures are deceptive: states have grown larger over the years
  - More agressive techniques for exploring only some of the states
  - Model checkers for C, Java
  - Need to store all those states online
  - Space requirements $\sim$ 1 terabyte
- Three main approaches
  1. Symbolic model checking (good for hardware)
  2. Bounded model checking (based on SAT solving)
  3. Automata-theoretic model checking
- Automata already used in model checking
- We want to use minimal/reduced $\oplus$-NFA to store states

NFA minimization

- DFA minimization is straightforward
  - Hopcroft algorithm
  - Minimal DFA is unique
- NFA min. is NP-hard when $|\Sigma| \geq 2$ (Jiang/Ravikumar 1991)
- Several algorithms have been proposed
  - Kamada/Weiner 1970
  - Arnold/Dicky/Nivat 1995, Carrez 1970
  - Matz/Potthoff 1995
  - Polák 2005
- Algorithms involve combinatorial search for subautomata of a universal automaton
  - complicated (need to manipulate dual automata,...)
  - not clear how to direct search, few heuristics
  - not intensely investigated
- Our suggestion: use constraint logic programming (CLP) in particular, convert NFA minimization to SAT instance

New idea

Algorithm

Input: NFA $M = (S, \Sigma, \Delta, \hat{s}, A)$
Output: NFA $M_* = (S_*, \Sigma, \Delta_*, \hat{s}_*, A_*)$

1. Determinize + reduce $M \longrightarrow$ DFA $M_+ = (S_+, \Sigma, \Delta_+, \hat{s}_+, A_+)$
2. Let $n := \lceil \log_2 |S_+| \rceil$
3. Guess an NFA $M_? = (S_?, \Sigma, \Delta_?, \hat{s}_?, A_?)$ with $n = |S_?|$
4. Construct a SAT problem K to describe the fact that
$$M_+ = \mathit{determinize}(M_?)$$
5. Apply a SAT solver to K
6. If K is satisfiable, extract and return $M_?$
7. If $n < \min(|S|, |S_+|)$, increment n and go to step 3
8. return M or M_+

Conversion to SAT instance

DFA $M_+ = (\ldots)$

		a
$\rightarrow$	0	1
	1	2
	2	3
$\leftarrow$	3	0

Conversion to SAT instance

NFA $M_? = (\ldots)$ for $n = 2$

$\leftarrow$	$\rightarrow$		a
a_0	i_0	0	$x_{00}x_{01}$
a_1	i_1	1	$x_{10}x_{11}$

DFA $M_+ = (\ldots)$

		a
$\rightarrow$	0	1
	1	2
	2	3
$\leftarrow$	3	0

Conversion to SAT instance

NFA $M_? = (\ldots)$ for $n = 2$

$\leftarrow$	$\rightarrow$		a
a_0	i_0	0	$x_{00}x_{01}$
a_1	i_1	1	$x_{10}x_{11}$

DFA $M_+ = (\ldots)$

		a
$\rightarrow$	0	1
	1	2
	2	3
$\leftarrow$	3	0

***determinize*$(M_?)$**

		a
$\rightarrow$	$t_{00}t_{01}$	$t_{10}t_{11}$
	$t_{10}t_{11}$	$t_{20}t_{21}$
	$t_{20}t_{21}$	$t_{30}t_{31}$
$\leftarrow$	$t_{30}t_{31}$	$t_{00}t_{01}$

Conversion to SAT instance

NFA $M_? = (\ldots)$ for $n = 2$

$\leftarrow$	$\rightarrow$		a
a_0	i_0	0	$x_{00}x_{01}$
a_1	i_1	1	$x_{10}x_{11}$

DFA $M_+ = (\ldots)$

		a
$\rightarrow$	0	1
	1	2
	2	3
$\leftarrow$	3	0

***determinize*$(M_?)$**

		a
$\rightarrow$	$t_{00}\,t_{01}$	$t_{10}\,t_{11}$
	$t_{10}\,t_{11}$	$t_{20}\,t_{21}$
	$t_{20}\,t_{21}$	$t_{30}\,t_{31}$
$\leftarrow$	$t_{30}\,t_{31}$	$t_{00}\,t_{01}$

- $(t_{00} \Leftrightarrow i_0) \wedge (t_{01} \Leftrightarrow i_1)$
- $(t_{30} \wedge a_0) \vee (t_{31} \wedge a_1)$
- $t_{10} \Leftrightarrow ((t_{00} \wedge x_{00}) \vee (t_{01} \wedge x_{10}))$
- $t_{11} \Leftrightarrow ((t_{00} \wedge x_{01}) \vee (t_{01} \wedge x_{11}))$
- $t_{20} \Leftrightarrow ((t_{10} \wedge x_{02}) \vee (t_{11} \wedge x_{12}))$
- $\ldots$

NFA/SAT minimization

- Pros
  - New algorithm is straightforward
  - SAT solvers are highly optimized
  - SAT problem is actively studied
  - Easy to convert to distributed/parallel environment
  - Algorithm accepts any $\star$-NFA as input
  - Produces any $\star$-NFA as output
  - Not known if older algorithms can be used for $\star$-NFA
  - Easy to find NFA with least nr. of transitions, other properties
- Cons
  - SAT problems can grow exponentially (but not necessarily)
    (may need to be integrated with SAT solvers)
  - Not easy to know which SAT solver works best
  - SAT problem is more general,
    direct study of minimization problem may yield better results

Conclusion/future work

- Work-in-progress (viz. Disclaimer 2)
- Current implementation to be completed (excuses, excuses)
- Works well interactively for up to $\sim$ 12-state NFA
  Only available alternative is limited to 32-state DFA
- Greatest obstacle: implementing older methods