Set Automata
A presentation based on research done by M. Kutrib, A. Malcher and M. Wendlandt

František Mráz

Charles University, Prague, Czech Republic

December 11, 2015

Table of Contents

Table of Contents

Original Articles

1. Martin Kutrib, Andreas Malcher, Matthias Wendlandt: Deterministic Set Automata. In: Developments in Language Theory 2014, Volume 8633 of LNCS, Springer, Berlin, pp 303–314
2. Martin Kutrib, Andreas Malcher, Matthias Wendlandt: Regularity and Size of Set Automata. In: Descriptional Complexity of Formal Systems 2014, Volume 8614 of LNCS, Springer, Berlin, 2014, pp 282–293

Finite Automata

- Advantages
  - closed under almost most commonly studied operations (Boolean operations, concatenation, (inverse) homomorphism, substitution, etc)
  - common decidability problems are decidable; mostly in polynomial time
  - effective minimization
- Disadvantages
  - limited expressiveness

Extending Finite Automata

adding storage, but keep as many ‘good’ properties as possible

- pushdown automata
  - nondeterministic version stronger than the deterministic one
  - some closure properties preserved (union, concatenation, (inverse) homomorphism, substitution)
  - some closure properties lost (complement)
  - some problems still decidable (emptiness, finiteness)
  - some problems undecidable (equivalence decidable only for deterministic pushdown automata
- queue automata
  - in general too strong (they accept all recursively enumerable languages)
  - quasi real-time queue automata – the number of subsequent λ-moves is bounded by some constant
  - a constant number of turns – changes between an enqueuing and a dequeuing phase

Extending Finite Automata

- bag automata
  - finite automata with a finite number of bags
  - bags can store (multiple copies) of symbols
  - able to simulate some counter automata
  - "well-formed" bag automata: accept a language class in between the (deterministic) one-counter and the (D)CFL

Design Goals

- add a set storage to a deterministic finite automaton
- set operations
  - add string
  - remove string
  - test whether a string is present in the set
- how to design a string
  - compose it on a write-only tape

Table of Contents

Nondeterministic Set Automaton – Definition

$M = (Q, \Sigma, \Gamma, \lhd, \delta, q_0, F)$

- Q is the finite set of states,
- Σ input alphabet,
- Γ tape alphabet,
- $\lhd \notin \Sigma$ is the right end-marker,
- $s_0 \in Q$ is the initial state,
- $F \in Q$ is the set of accepting states, and
- $\delta : Q \times (\Sigma \cup \{\lambda, \lhd\}) \to (Q \times (\Gamma^* \cup \{\mathtt{in}, \mathtt{out}\})) \cup (Q \times \{\mathtt{test}\} \times Q)$ is the partial transition function
  - `in` – the instruction to add the content of the tape to the set,
  - `out` – the instruction to remove the content of the tape from the set, and
  - `test` – the instruction to test whether or not the content of

the tape is in the set.

Nondeterministic Set Automaton – Configuration

A configuration of NSA M is $(q, v, z, \mathbb{S})$

- $q \in Q$ is the current state,
- $v \in (\Sigma^* \triangleleft) \cup \{\lambda\}$ is the unread part of the input,
- $z \in \Gamma^*$ is the content of the tape, and
- $\mathbb{S} \subseteq \Gamma^*$ is the finite set of stored words.

initial configuration for an input string w is

$$(q_0, w\triangleleft, \lambda, \emptyset)$$

Nondeterministic Set Automaton – Step

Let $q, q', \overline{q} \in Q$, $x \in \Sigma \cup \{\lambda, \lhd\}$, $v \in (\Sigma^* \lhd) \cup \{\lambda\}$, $z, z' \in \Gamma^*$, and $\mathbb{S} \subseteq \Gamma^*$

one step relation $\vdash$:

1. $(q, xv, z, \mathbb{S}) \vdash (q', v, zz', \mathbb{S})$, if $(q', z') \in \delta(q, x)$ – write,
2. $(q, xv, z, \mathbb{S}) \vdash (q', v, \lambda, \mathbb{S} \cup \{z\})$), if $(q', \texttt{in}) \in \delta(q, x)$ – insert,
3. $(q, xv, z, \mathbb{S}) \vdash (q', v, \lambda, \mathbb{S} \setminus z)$), if $(s', \texttt{out}) \in \delta(q, x)$ – remove (no check whether $z \in \mathbb{S}$),
4. $(q, xv, z, \mathbb{S}) \vdash (q', v, \lambda, \mathbb{S})$), if $(q', \texttt{test}, \overline{q}) \in \delta(q, x)$ and $z \in \mathbb{S}$ – positive test,
5. $(q, xv, z, \mathbb{S}) \vdash (\overline{q}, v, \lambda, \mathbb{S})$, if $(q', \texttt{test}, \overline{q}) \in \delta(q, x)$ and $z \notin \mathbb{S}$ – negative test.

The language accepted by the NSA M is the set

$$L(M) = \{w \in \Sigma^* \mid (q_0, w\lhd, \lambda, \emptyset) \vdash^* (q_f, \lambda, z, \mathbb{S}) \text{ with } q_f \in F, z \in \Gamma^*, \mathbb{S} \subseteq \Gamma^*\}$$

Deterministic Set Automaton

Deterministic set automaton (DSA) – at most one choice of action for any configuration

- $|\delta(q, x)| \leq 1$, for any $q \in Q$ and $x \in \Sigma \cup \{\lambda, \triangleleft\}$,
- if $\delta(q, \lambda) \neq \emptyset$ then $\delta(q, x) = \emptyset$, for any $q \in Q, x \in \Sigma \cup \{\triangleleft\}$.

Table of Contents

Examples

$$L_1 = \{w_1\$w_2\$\cdots w_m\$w \mid \quad m \geq 1, w, w_1, w_2, \ldots, w_m \in \{a,b\}^*, \text{ and } \exists 1 \leq i \leq m : w = w_i\}$$

L_1 is accepted by a DSA:

1. read each a and b up to the letter \$ and copy it to the tape,
2. reading \$, store the word written on the tape in its set,
3. when the input head arrives at $\lhd$, test whether the contents on the tape is in the set,
4. if yes, accept, otherwise reject.

Examples

$L_2 = \{x\$w \mid x, w \in \{a, b\}^* \text{ and } w \text{ is a factor of } x\}$

L_2 is accepted by a NSA:

1. read a'a and b's, do not write on the tape,
2. nondeterministically guess that the factor w starts and continue with copying a's and b's into the tape
3. nondeterministically guess that the factor w has ended, perform an `in`-operation
4. stop copying input onto the tape and read until $\$$
5. after symbol $\$$, copy the input again into the tape
6. at the right endmarker, test whether the content on the tape is in the set,
7. if yes, accept, otherwise reject.

Examples

$L_3 = \{a^n b^m \$_0 c^n \mid m, n \geq 1\} \cup \{a^n b^m \$_1 c^m \mid m, n \geq 1\}$

L_3 is accepted by a DSA:

- copy a'a into the tape
- on the first b insert the word from the tape into the set
- copy b's into the tape
- on $\$_0$ or $\$_1$, insert the word from the tape into the set
- depending on whether there has been a $\$_0$ or a $\$_1$ in the input, write an a or a b for each c in the input on the tape,
- at $\triangleleft$, check whether the word on the tape is in the set,
- if yes, accept, otherwise reject.

Examples

$L_4 = \{a^n b^n c^n \mid n \geq 1\}$

L_4 is accepted by a DSA:

- copy every a in the input onto the tape,
- at the first b, add the content of the tape to the set,
- for every b in the input write an a on the tape
- at the first c, test whether the word on the tape is in the set
- if not, reject, otherwise, for every c in the input write an a on the tape,
- at $\lhd$, if the word on the tape is in the set, then accept, otherwise reject.

Table of Contents

Unary Languages

Theorem 1

Every unary language accepted by a DSA is semilinear, thus regular.

Proof:

- let $M = (Q, \{a\}, \Gamma, \triangleleft, \delta, q_0, F)$ be a DSA
- if M accepts a finite language
  - done, as each finite language is semilinear

Unary Languages

Theorem 2

Every unary language accepted by a DSA is semilinear, thus regular.

Proof (cont.):

- let M accept an infinite language
  - $k =$ the length of a longest word that M can write in one step on the tape
  - on input longer than $|Q|$, the automaton enters a loop

1. no `in`-, `out`-, or `test`-operation within the loop – we can transform the automaton into an equivalent deterministic finite automaton
2. M performs an `in`-, `out`, or `test`-operation
   - after such operation the content of the tape is deleted
   - in each computation step, M can write at most k symbols on the tape
   - (unary input) M can distinguish between at most $|S|$ different situations $\Rightarrow$ the words on the tape of length at most $k \cdot |S|$
   - a DFA can simulate M by storing the content on the tape and the finite number of words in the set in its state

$\Rightarrow L(M)$ can be accepted by a finite automaton, it is semilinear

Comparison to quasi-real-time queue automata

Theorem 3

The family of languages accepted by DSA is incomparable with the family of languages accepted by quasi-real-time queue automata.

Proof:

1. the non-semilinear unary language $\{a^n \mid n \text{ is a Fibonacci number}\}$ is accepted by some quasi-real-time queue automaton[1]
2. L_3 cannot be accepted by a quasi-real-time queue automaton (by contradiction)
   - let M be a quasi-real-time queue automaton accepting L_3 with the set of states Q
   - input $w = a^j b^{j'} v$, where $v \in \{\$_0, \$_1\} c^*$
   - after a^j is read, the length of z written in the queue depends on j
     - otherwise, for some $i \neq i'$ there are two accepted words $w' = a^i b \$_0 c^i$ and $w'' = a^{i'} b \$_0 c^{i'}$ such that after reading b the aut. M is in the same configuration on both words $\Rightarrow$ M accepts also $a^i b \$_0 c^{i'}$

[1] A. Cherubini, C. Citrini, S. Crespi Reghizzi and D. Mandrioli, QRT FIFO ...

Comparison to quasi-real-time queue automata

Theorem 4

The family of languages accepted by DSA is incomparable with the family of languages accepted by quasi-real-time queue automata.

Proof (cont.):

- similarly, after $a^i b^{j'}$ is read a word z' is appended to the queue and $|z'|$ depends on j'
- let after reading $a^i b^j \$_0 c^i$, $i, j \geq |Q|$ the queue contains z of length $> 2j \cdot |Q|$
- M must be in an accepting state after reading $a^i b^j \$_0 c^i$ and in the front of the queue there is still a word $\bar{z}$ such that $z = \bar{z}'\bar{z}$ and $|\bar{z}| > j \cdot |Q|$
- $\Rightarrow \exists$ a word $a^i b^j \$_0 c^{i+j}$ with $i, j \geq |Q|$ and $j' \geq 1$: M is in the same accepting state – a contradiction

Action Normal Form

A DSA M is in action normal form, if

- the initial state of M is only visited once
- each other state indicates uniquely which action the automaton M did in the last computation step

the state set is (disjointly) partitioned
$Q = Q_{\texttt{in}} \cup Q_{\texttt{out}} \cup Q_{\texttt{test+}} \cup Q_{\texttt{test-}} \cup Q_{\texttt{write}}$

Lemma 5

Any DSA M can be converted into an equivalent DSA M' in action normal form.

Action Normal Form

Lemma 6

Any DSA M can be converted into an equivalent DSA M' in action normal form.

- a new initial state which is visited only at the beginning of a computation is added
- original states are marked as writing states and instead of, e.g. $\delta(p, a) = (q, \mathtt{in})$ a detour is used $\delta'(p, a) = (q_{\mathtt{in}}, \mathtt{in})$ and $\delta'(q_{\mathtt{in}}, \lambda) = (q, \lambda)$
- similar “detours” are added for all transitions with operations on the set

Infinite Action Normal Form

A DSA $M = (Q, \Sigma, \Gamma, \lhd, \delta, q_0, F)$ in action normal form

- $Q = Q_{\texttt{in}} \cup Q_{\texttt{out}} \cup Q_{\texttt{test}} \cup Q_{\texttt{write}}$, where $Q_{\texttt{test}} = Q_{\texttt{test+}} \cup Q)_{\texttt{test-}}$
- let $q_i \in \{q_0\} \cup Q_{\texttt{in}} \cup Q_{\texttt{out}} \cup Q_{\texttt{test}}$
- let $q_j \in Q_{\texttt{in}} \cup Q_{\texttt{out}} \cup Q_{\texttt{test}}$

$$L_{q_i,q_j} = \{w_n \in \Gamma^* \mid \text{ there is } u \in \Sigma^* \text{ such that } (q_i, u, , \mathbb{S}) \vdash (q_{i+1}, u_1, w_1, \mathbb{S}) \vdash^* (q_{i+(n-1)}, u_{n-1}, w_{n-1}, \mathbb{S}) \vdash (q_{i+n}, u_n, w_n, \mathbb{S}) \vdash (q_j, \lambda, \lambda, \mathbb{S}'), \text{ and } q_{i+1}, q_{i+2}, \ldots, q_{i+n} \notin Q_{\texttt{in}} \cup Q_{\texttt{out}} \cup Q_{\texttt{test}}\}.$$

- all such L_{q_i,q_j} are regular

A DSA M is in infinite action normal form if M is in action normal form and all sets L_{q_i,q_j} are infinite.

Infinite Action Normal Form

Lemma 7

Any DSA M can be converted into an equivalent DSA M' in infinite action normal form.

- all L_{q_i,q_j} are constructed and their finiteness is tested
- let k be the maximal length of a word in all finite L_{q_i,q_j}
- no set operation on words of length less or equal k should be performed – such operations can be simulated in states of the control unit
- M' simulates all operations on word of length at most k in states and write to the tape only if a word longer than k is written
- M' is deterministic and in infinite action normal form

Comparison to DCFL

Theorem 8

The family of languages accepted by DSA is incomparable with the (deterministic) context-free languages.

- $L_4 = \{a^n b^n c^n \mid n \geq 1\}$ non-context-free and is accepted by a DSA
- $L = \{wcw^R \mid w \in \{a, b\}^*\}$ is not accepted by any DSA can be shown by contradiction
  - **idea:**
    1. if L is accepted by a DSA M, then all possible set operation on the first part of the input are a finite number of `in`-operations, and on the second part are a finite number of `test`-operations
    2. based on M an equivalent one-way multi-head finite automaton accepting L can be constructed – a contradiction

Comparison to Finite-Turn Queue Automata

Theorem 9

The family of languages accepted by DSA is incomparable with the family of languages accepted by finite-turn queue automata.

- $L_4 = \{a^n b^n c^n \mid n \geq 1\}$
  - L_4 is accepted by a DSA (see above)
  - L_4 cannot be accepted by any finite-turn deterministic queue automaton (it follows from some other papers)
- let $L = L' \cup L''$ with $L' = \{a^n b^m c^n \mid m, n \geq 1\}$ and $L'' = \{a^n b^m c^{n+m} \mid m, n \geq 1\}$
  - L is accepted by a one-turn deterministic queue automaton (easy)
  - L is not accepted by any DSA – by a contradiction
    - a long proof

Table of Contents

Closure Under Complementation

Lemma 10

The family of languages accepted by DSA is closed under complementation.

- problems with simply interchanging accepting and rejecting states
  1. the given DSA may not read its input completely – no next move from a configurations is defined,
  2. the given DSA may not read its input completely – by entering an infinite λ-loop,
  3. the given DSA may perform λ-steps from an accepting state to a rejecting state and back.
- solution:
  - (1) – missing transitions replaced by a transition to a new rejecting state q_{rej}; additionally staying in q_{rej}, the automaton will continue to read the rest of input
  - (3) – after entering an accepting state after reading the right sentinel, the automaton immediately enters a new accepting state q_{acc}
  - the modified automaton (without problems (1) and (3)) is converted into the infinite action normal form
    - the resulting automaton still does not have problems (1) and (3)
    - infinite λ-cycles still possible

Closure Under Complementation

Lemma 11

The family of languages accepted by DSA is closed under complementation.

solution (cont,):

- the modified automaton (without problems (1) and (3)) is converted into the infinite action normal form
- infinite λ-cycles still possible
  1. if in an infinite λ-cycle only states from Q_{write} can be visited
     - we can check it in advance and instead of entering the infinite λ-cycle we modify the automaton to enter q_{rej}
  2. if in an infinite λ-cycle a state $q_1 \in Q_{\text{in}} \cup Q_{\text{out}} \cup Q_{\text{test}}$ can be visited
     - let q_2 be the next non-writing state in the λ-cycle
     - L_{q_1,q_2} should be infinite (infinite action normal form)
     - however, starting from q_1, the tape is empty, at q_2 the tape is also empty, and no symbol is read, hence L_{q_1,q_2} is finite (having one elemnet) – a contradiction
- finally, switching accepting and non-accepting states works

Union, Intersection and Intersection with Regular Languages

Lemma 12

The family of languages accepted by DSA is not closed under union and intersection.

- $L = L' \cup L'' \notin \mathcal{L}(\mathrm{DSA})$, where
  - $L' = \{a^n b^m c^n \mid m, n \geq 1\} \in \mathcal{L}(\mathrm{DSA})$
  - $L'' = \{a^n b^m c^{n+m} \mid m, n \geq 1\} \in \mathcal{L}(\mathrm{DSA})$
- $\mathcal{L}(\mathrm{DSA})$ not closed under union – de Morgan rule

Lemma 13

The family of languages accepted by DSA is closed under intersection with regular languages and under union with regular languages.

- easy

Table of Contents

Regularity

regularity

- decidable for deterministic pushdown automata
- not even semi-decidable for deterministic real-time queue automata

Theorem 14

It is decidable whether or not a given deterministic set automaton accepts a regular language.

- Given a DSA M in infinite action normal form, it is possible to determine whether M performs a `test`-operation that matches for infinitely many strings inserted in the set by a related `in`-operation. If this is the case for accepting computations, the language accepted cannot be regular because there are infinitely many pairs of related input factors.
- a meta automaton M' and the computation tree built from the state graph of M' up to a certain depth. The nodes of the computation tree are labeled by some information that is used to test the finitely many paths in the tree. The results of these tests allow to determine the regularity of the language accepted by M.
- a complex proof

Emptiness, Finiteness, Infiniteness, and Universality

Theorem 15

The questions of emptiness, finiteness, infiniteness, and universality are decidable for deterministic set automata.

Table of Contents

Descriptional Complexity

Theorem 16

Let M be an n-state DSA with set of tape symbols Γ that accepts a regular language. Then an equivalent DFA with at most $2^{|\Gamma|^{O(n^2)}}$ states can effectively be constructed.

$$L_n = \{\$^* w_1 \$^+ w_2 \$^+ \cdots w_m \$^+ w \mid m \geq 1, w, w_1, w_2, \ldots, w_m \in \{a, b\}^n, \text{ and } \exists 1 \leq i \leq m : w = w_i\}$$

Theorem 17

For $n \geq 1$, language L_n is accepted by an $(n+2)$-state DSA, but any equivalent DFA needs at least 2^{2^n} states.

Corollary 18

For every $n \geq 1$, there are regular languages L_n which are accepted by an $(n+2)$-state DSA with tape alphabet Γ such that any equivalent DFA needs at least $2^{|\Gamma|^n}$ states.

Table of Contents

Nondeterminism and Non-Recursive Trade-Offs

VALC(M) is the set of valid (accepting) computations of M

- $w_0\#w_1^R\#w_2\#w_3^R\#\cdots\#w_{2n}\#w_{2n+1}^R$, where $\# \notin T \cup Q$,

$w_i, 0 \leq i \leq 2n+1$, are instantaneous description of M, w_0 is an initial ID, w_{2n+1} is an accepting (hence halting) configuration, w_{i+1} is the successor configuration of w_i, $0 \leq i \leq 2n$. Similarly, the set VALC'(M) consists of all finite strings of the form $w_0\#w_1\#w_2\#\cdots\#w_{2n+1}$. The set of invalid computations INVALC(M) respectively INVALC'(M) is the complement of VALC(M) respectively VALC'(M) with respect to the alphabet $T \cup Q \cup \{\#\}$.

Nondeterminism and Non-Recursive Trade-Offs

Lemma 19

Let M be a Turing machine. Then

1. *INVALC(M) is a context-free language and a pushdown automaton accepting it can effectively be constructed,*
2. *INVALC(M) belongs to $\mathcal{L}(DSA)$ if and only if $L(M)$ is finite,*
3. *INVALC'(M) belongs to $\mathcal{L}(NSA)$ and an NSA accepting it can effectively be constructed,*
4. *INVALC'(M) belongs to $\mathcal{L}(DSA)$ if and only if $L(M)$ is finite, and*
5. *INVALC'(M) is a deterministic context-free language if and only if $L(M)$ is finite.*

Nondeterminism and Non-Recursive Trade-Offs

Theorem 20

The trade-offs between

1. *NSA and DSA,*
2. *NSA and deterministic pushdown automata, and*
3. *pushdown automata and DSA*

are non-recursive.

Theorem 21

For NSA the questions of universality, equivalence with regular sets, equivalence, inclusion, and regularity are not semi-decidable. Furthermore, it is not semi-decidable whether the language accepted by some NSA belongs to $\mathcal{L}(DSA)$.