Store Languages of Automata Models, with Applications

Ian McQuillan

Department of Computer Science,
University of Saskatchewan,
Canada

Supported by Natural Sciences and Engineering Research Council of Canada (NSERC)

University of Saskatchewan

NSERC CRSNG

University of Saskatchewan

Store Languages

Much (but not all) of this work is based on joint work with Oscar Ibarra:

- On Store Languages of Language Acceptors. *Theoretical Computer Science*, 745: 114-132, 2018.

- On Store Languages and Applications. *Information and Computation*, 267: 28-48, 2019.

Automata Models

- There are many different types of automata models that have been studied.
- Standard models include:
    - an input tape that is either one-way or two-way,
    - a finite state set,
    - a nondeterministic or deterministic finite control,
    - zero or more types of data stores.

Pushdown Automata

- A one-way pushdown automaton (NPDA) has a one-way input and a pushdown stack as data store.

- Each transition can push, pop, or keep the same pushdown contents.

- Given a machine $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, configurations look like

$$(q, w, \gamma),$$

  where $q \in Q$ is the current state, $w \in \Sigma^*$ is the remaining input, and $\gamma \in \Gamma^*$ is the current contents of the stack.

- The language accepted by M,
  $L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash \cdots \vdash (q_f, \lambda, \gamma), q_f \in F\}$.

Example

Consider the non-regular language

$$\{w\$w^R \mid w \in \{a, b\}^*\}.$$

This can be accepted by a pushdown automaton M with transitions

$\delta(q_0, a, x) \to (q_0, push(a))$ for all $x \in \{Z_0, a, b\}$,
$\delta(q_0, b, x) \to (q_0, push(b))$ for all $x \in \{Z_0, a, b\}$,
$\delta(q_0, \$, x) \to (q_1, stay)$ for all $x \in \{Z_0, a, b\}$,
$\delta(q_1, a, a) \to (q_1, pop)$,
$\delta(q_1, b, b) \to (q_1, pop)$,
$\delta(q_1, \lambda, Z_0) \to (q_f, stay)$.

Store Configurations and Store Languages

- Given a configuration (q, w, γ), the *store configuration* is a string encoding of the state and data store, $q\gamma$.
- The *store language of* M, $S(M)$, is the set of all store configurations that can appear in any accepting computation.
- That is,

$$S(M) = \{q\gamma \mid (q_0, w, Z_0) \vdash^* (q, w', \gamma) \vdash^* (q_f, \lambda, \gamma'), q_f \in F\}.$$

Consider M accepting $\{w\$w^R \mid w \in \{a,b\}^*\}$ with transitions

$\delta(q_0, a, x) \to (q_0, push(a))$, $\delta(q_0, b, x) \to (q_0, push(b))$,
$\delta(q_0, \$, x) \to (q_1, stay) \forall x$, $\delta(q_1, a, a) \to (q_1, pop)$,
$\delta(q_1, b, b) \to (q_1, pop)$, $\delta(q_1, \lambda, Z_0) \to (q_f, stay)$.

- Given any $\gamma \in \{a,b\}^*$, $q_0 Z_0 \gamma \in S(M)$ because there is an accepting computation $(q_0, \gamma\$\gamma^R, Z_0) \vdash^* (q_0, \$\gamma^R, Z_0\gamma) \vdash^* (q_f, \lambda, Z_0)$.
- Given any $\gamma \in \{a,b\}^*$, $q_1 Z_0 \gamma \in S(M)$ because there is an accepting computation $(q_0, \gamma\$\gamma^R, Z_0) \vdash^* (q_1, \gamma^R, Z_0\gamma) \vdash^* (q_f, \lambda, Z_0)$.
- No other words that start with q_0 or q_1 are in $S(M)$ because there is no accepting computation that pops or pushes Z_0.
- The only word that starts with q_f in $S(M)$ is $q_f Z_0$, as the only way to accept when passing over q_f is to end in q_f and the stack be Z_0.

Store Languages of Pushdown Automata

- The store language $S(M)$ is $q_0 Z_0 (a+b)^* + q_1 Z_0 (a+b)^* + q_f Z_0$.
- Hence, there are NPDAs M such that $L(M)$ is not a regular language, but $S(M)$ is a regular language.
- How complex can store languages get?

Store Languages of Pushdown Automata

Shown in S. Greibach, A note on pushdown store automata and regular systems, *Proceedings of the American Mathematical Society* 18 (1967).

Theorem (Greibach, 1967)

For every NPDA M*,* $S(M)$ *is a regular language.*

See also proof in Handbook of Formal Languages, Volume A, chapter on Context-Free Languages by Autebert, Berstel, and Boasson.

Store Languages of Pushdown Automata

Theorem (Geffert, Malcher, Meckel, Mereghetti, Palano, DCFS 2013)

Given a NPDA M *with* Γ *for pushdown alphabet, an* NFA *accepting* $S(M)$ *has at most* $|M|^2|\Gamma| + 1$ *states.*

- They also prove this is optimal.

Theorem (Malcher, Meckel, Mereghetti, Palano, DCFS 2012)

Given an NPDA M*, an* NFA *accepting* $S(M)$ *can be constructed in polynomial time.*

Application

- This result on pushdown automata can be used to prove that regular canonical systems produce regular languages.

  J. R. Büchi, *The Collected Works of J. Richard Büchi*, 1990, Chapter: Regular Canonical Systems.

Other Kinds of Automata

- We wanted to study the store languages of other types of automata with different data stores.
- We defined generally something called a *store type*.
- Within this formal system, we can define many different types of stores.

Store Types

A store type describes:

- infinite alphabet Γ of available as store symbols,
- the allowable instructions I,
- the read function, a partial function from Γ^* to Γ,
- the write function, a partial function from $\Gamma^* \times I$ to new contents Γ^*,
- the initial store configuration,
- an "instruction language" that can restrict the allowable sequences of instructions.

Store Types

Given store types $\Omega_1, \ldots, \Omega_k$, we can define a

- one-way nondeterministic machine with $\Omega_1, \ldots, \Omega_k$,
- one-way deterministic machine with $\Omega_1, \ldots, \Omega_k$,
- two-way nondeterministic machine with $\Omega_1, \ldots, \Omega_k$,
- two-way deterministic machine with $\Omega_1, \ldots, \Omega_k$.

The *mode* is either one-way nondeterministic, one-way deterministic, two-way nondeterministic, or two-way deterministic.

Machines

- Given store types $\Omega_1, \ldots, \Omega_k$ and a mode, we can examine the class $\mathcal{M}$ of **all** machines with those stores using that mode.
- $\mathcal{L}(\mathcal{M})$ is the family of languages accepted by machines in $\mathcal{M}$.
- $\mathcal{S}(\mathcal{M})$ is the family of store languages of machines in $\mathcal{M}$.
- E.g. for the pushdown store type, and consider NPDA, the class of all one-way nondeterministic machines with a pushdown. Then

$$\mathcal{L}(\mathsf{NPDA}) = \mathsf{CFL} \textit{ and } \mathcal{S}(\mathsf{NPDA}) \subseteq \mathsf{REG}.$$

Multiple Stores

- When there are multiple stores, the store language concatenates them all together over separate alphabets.
- E.g. a reversal-bounded k-counter machine is a machine with k counters, where there is a bound on the number of times each counter can switch between increasing and decreasing.
- Each word of the store language is of the form $qc_1^{i_1} \cdots c_k^{i_k}$ where $c_1, \ldots, c_k$ are fixed letters associated with the counters.
- Essentially, the states are treated like their own finite store.

Turing Machines

Definition

Make a Turing tape store type. Let $\mathcal{M}$ be machines with a one-way (read-only) input and one Turing tape. Words in the store language are of form $qua\downarrow v$ where read/write head is scanning symbol a.

Theorem

There is a fixed word x such that $\mathrm{RE} = \{(x)^{-1}S(M) \mid M \in \mathcal{M}\}$.

Proof.

Given language $L \in \mathrm{RE}$, let M' be a one-tape Turing machine accepting L. Let q be a new state. Make $M \in \mathcal{M}$ which starts by copying input to worktape, move tape head left to blank, switch to state q, then simulate M' using other states. Hence, $(q_{\sqcup}\downarrow)^{-1}S(M) = L$.

□

Finite-Turn Turing Machines

Definition

A finite-turn Turing machine has a one-way read-only input tape with a read/write worktape that can change directions at most k times on any accepting computation, for some k.

Model studied by Greibach, TCS 1978.

Theorem

Let $\mathcal{M}$ be the finite-turn Turing machines. Then $\mathcal{S}(\mathcal{M}) \subseteq \text{REG}$.

Theorem

Let $\mathcal{M}$ be the reversal-bounded queue machine. Then $\mathcal{S}(\mathcal{M}) \subseteq \text{REG}$.

Finite-Visit Turing Machines

Definition

A finite-visit Turing machine has a one-way read-only input tape with a read/write worktape that can visit each cell at most k times on any accepting computation, for some k.

The family of languages is more powerful than finite-turn Turing machines (Greibach, One-way finite visit automata, TCS 1978).

Theorem (unpublished, Friesen, Jirásek, IM)

Let $\mathcal{M}$ be the finite-visit Turing machines. Then $\mathcal{S}(\mathcal{M}) \subseteq \text{REG}$.

Flip-Pushdowns

Definition

A one-way k-flip pushdown automaton has a pushdown that can be "reversed" up to k times.

Model studied by Holzer and Kutrib, DLT 2003, ICALP 2003.

Theorem

Let $\mathcal{M}$ be the k-flip pushdown automata. Then $\mathcal{S}(\mathcal{M}) \subseteq \mathrm{REG}$.

Reversal-Bounded Counters

Definition

A reversal-bounded is a counter, which can be incremented and decremented and tested for zero, but can only change directions at most a bounded number of times.

Let NCM be one-way machines with some number of reversal-bounded counters. Let DCM be one-way deterministic machines.

Theorem

$\mathcal{S}(\mathrm{NCM}) \subseteq \mathcal{L}(\mathrm{DCM})$.

Adding Reversal-Bounded Counters

Theorem

Let $\mathcal{M}$ be one-way machines with one of the following stores:

- *pushdown,*
- *finite-turn Turing tape*
- *reversal-bounded queue*
- *k-flip pushdown*

and some number of reversal-bounded counters. Then $\mathcal{S}(\mathcal{M}) \subseteq \mathcal{L}(\mathsf{NCM})$.

Stack Automata

Definition

A stack store type is like a pushdown automaton (can push/pop), but it can also read from the stack in a two-way read-only mode. It can only use push/pop when the read/write head is at the top.

Let STACK be the class of one-way nondeterministic stack automata.

Example

$\{w^k \mid w \in \{a, b\}^*, k \geq 2\} \in \mathcal{L}(\text{STACK})$.

Stack automata are far more powerful than pushdown automata.

Stack Automata

Theorem (Bensch, Björklunc, Kutrib, IJFCS 2017)

Then $\mathcal{S}(\mathsf{STACK}) \subseteq \mathsf{REG}$.

- Words in the store language contain read/write head $\downarrow$.
- This is a complicated proof.
- We'll return to this result shortly.

Other Modes

- So far we've looked at one-way nondeterministic automata.
- What about one-way deterministic and two-way inputs?

One-Way Nondeterministic vs. One-Way Deterministic

Theorem

Let $\Omega_1, \ldots, \Omega_k$ be store types, let $\mathcal{M}_N$ (resp. $\mathcal{M}_D$) be one-way nondeterministic (resp. deterministic) machines with these store types. Then $\mathcal{S}(\mathcal{M}_N) = \mathcal{S}(\mathcal{M}_D)$.

One-Way vs. Two-Way

- Given any class of two-way automata that accepts a finite-language, the store languages are "the same" as one-way automata.

Theorem

Let $\Omega_1, \ldots, \Omega_k$ be store types.
Let $1\mathcal{M}$ be one-way nondeterministic machines with those stores, and $2\mathcal{M}$ be the two-way nondeterministic machines with those stores.

Let $\mathcal{L}$ be a family closed under homomorphism. Then the following are equivalent:

1. $\mathcal{S}(1\mathcal{M}) \subseteq \mathcal{L}$,
2. $\mathcal{S}(\{M \in 1\mathcal{M} \mid L(M) = \{\lambda\}\}) \subseteq \mathcal{L}$,
3. $\mathcal{S}(\{M \in 2\mathcal{M} \mid L(M) \text{ is finite}\}) \subseteq \mathcal{L}$,

Corollary

For any $\mathcal{M}$ in the following:

- *two-way pushdown automata,*
- *two-way finite-turn Turing machines,*
- *two-way finite-visit Turing machines,*
- *two-way k-flip pushdown automata*
- *two-way stack automata,*

if $M \in \mathcal{M}$ with $L(M)$ finite, then $S(M) \in \mathrm{REG}$.

Similarly if we add reversal-bounded counters for all examples above except stack automata, then $S(M) \in \mathcal{L}(\mathrm{NCM})$.

Infinite Languages

- What about two-way machines that do not accept finite languages.

Proposition

There is a two-way deterministic one counter machine (that scans input twice) such that $S(M)$ is not regular (nor semilinear).

Stack Automata

Theorem (Ginsburg, Greibach, Harrison, Stack Automata and Compiling, JACM 1967)

The set of all words that can appear in the store of a two-way stack automaton M on a single input word $w \in \Sigma^*$ when M "falls off" the right end-marker of w is a regular language.

- This was the main step in showing all two-way stack automata languages accept recursive languages. To decide if w accepts, first build that regular set.

- From this, it is very easy to prove that for all two-way stack automata accepting $\{\lambda\}$, the store language is regular.

Generally: Applications to Decidability Properties

Theorem

Let $\Omega_1, \ldots, \Omega_k$ be store types.
Let $1\mathcal{M}$ be one-way nondeterministic machines with those stores, and $2\mathcal{M}$ be the two-way nondeterministic machines with those stores. Let $\mathcal{L}$ be a language family closed under homomorphism with a decidable emptiness problem.

If either $\mathcal{S}(1\mathcal{M}) \subseteq \mathcal{L}$ or $\mathcal{S}(\{M \mid M \in 2\mathcal{M}, L(M) \subseteq \{\lambda\}) \subseteq \mathcal{L}$ (effectively), then both are true, the emptiness and membership problems are decidable in $1\mathcal{M}$, and the membership problem is decidable in $2\mathcal{M}$.

Proof.

By previous theorem, it is enough to use $1\mathcal{M}$.
Given $M \in 1\mathcal{M}$, to decide emptiness, construct $S(M) \in \mathcal{L}$, which is empty if and only $L(M)$ is empty.
To decide membership of w in $M \in 2\mathcal{M}$, construct $M' \in 1\mathcal{M}$ that remembers w in the state and simulates M on w and accepts λ if M accepts w, and nothing otherwise. □

Application to Stack Automata

If we prove the store languages of two-way stack automata on one word are regular, this proves

- decidability of membership in two-way stack automata,
- store languages of one-way stack automata are regular,
- emptiness and membership are decidable for one-way stack automata.

If we prove the store languages of one-way stack automata are regular, this proves

- the store language of two-way automata on finite languages are regular,
- membership for two-way stack automata is decidable,
- membership and emptiness are decidable for one-way stack automata.

Application

When studying a one-way model $\mathcal{M}$, you get a lot of results for free if you can show $\mathcal{S}(\mathcal{M}) \subseteq \mathcal{L}$, where $\mathcal{L}$ has a decidable emptiness problem.

Right Quotient

Definition

Let $L, R \subseteq \Sigma^*$. $LR^{-1} = \{u \mid w = uv \in L, v \in R\}$.

- All families accepted by standard one-way nondeterministic automata are closed under right quotient with regular languages.
- For one-way deterministic machines, some families are closed under right quotient with regular languages.
- Deterministic pushdown automata and deterministic stack automata are closed under right quotient with regular languages. Both used separate difficult ad hoc proofs.
- This has been left unsolved for many classes of deterministic automata.

Right Quotient

Proposition

Let $\Omega_1, \ldots, \Omega_k$ be store types where there is a 'stay' instruction that is available at any point, and at any point it is possible to read each letter of the store one at a time, either from left-to-right or from right-to-left.

Let $\mathcal{M}_N$ (resp. $\mathcal{M}_D$) be the one-way nondeterministic (resp. deterministic) machines using these stores.

If $\mathcal{S}(\mathcal{M}_N) \subseteq \mathsf{REG}$ then $\mathcal{L}(\mathcal{M}_D)$ is closed under right quotient with regular languages.

Proof.

- Let $M_1 \in \mathcal{M}_D, M_2 \in$ DFA. We will build $M \in \mathcal{M}_D$ accepting $L(M_1)L(M_2)^{-1}$.

- First build $M_3 \in \mathcal{M}_N$ that on input w simulates M_1 until an nondeterministically guessed spot after u where $w = uv$ in state q, where it switches to q', then q'' where in parallel it both continues the simulation of M_1 and also simulates M_2 on v.

- So $S(M_3)$ is regular, and so is $T = S(M_3) \cap Q'\Gamma^*$ (Q' are the primed states). A DFA can be built accepting T and T^R.

- Finally build a deterministic $M \in \mathcal{M}_D$ that on input u simulates M_1 until the end of the input, then checks if the current store configuration is in T or T^R. □

Right Quotient

- In the above theorem, the stores had to be able to read from left-to-right or right-to-left at any point.
- Variants of Turing tapes, stack automata, checking stack automata cannot do this because they cannot read from end at any point.
- The proof can be adjusted to accommodate these stores with a slight modification.

Right Quotient

Corollary

The languages accepted by the following one-way deterministic classes are closed under right quotient with regular languages:

- *deterministic stack languages [Hopcroft, Ullman, 1968],*
- *deterministic checking stack languages,*
- *deterministic k-flip pushdown languages,*
- *deterministic pushdown automata [Ginsburg, Greibach,1966],*
- *deterministic one counter automata [Eremondi, Ibarra, IM 2017],*
- *deterministic reversal-bounded queue automata,*
- *deterministic Turing machines with a finite-turn worktape,*
- *deterministic Turing machines with a finite-crossing worktape.*

To our knowledge, all without a citation were previously unknown.

Application to Verification

Definition

Given a machine M and a set of store configurations C:

- $\mathrm{pre}^*_M(C)$ is the set of store configurations that can eventually lead to store configurations in C.
- $\mathrm{post}^*_M(C)$ is the set of store configurations that are eventually reachable from a store configuration in C.

These are commonly studied in model checking and reachability community.

Theorem (Bouajjani, Esparza, Maler, CONCUR 1997)

Given a NPDA *M and $C \in$* REG*, $\mathrm{pre}^*_M(C)$ and $\mathrm{post}^*_M(C)$ are regular.*

General Relationship With Store Languages

Definition

A set of machines $\mathcal{M}$ can be *loaded* by sets from some family $\mathcal{L}$ if, for all $M \in \mathcal{M}$ with state set Q and store configuration sets $C \in \mathcal{L}$, then there is a machine $M' \in \mathcal{M}$ with state set $Q' \supseteq Q$ that on input $q\gamma\$x\$$, can put γ on the stores using states in $Q' - Q$, then switch to q and simulate M on x.

- Every type of automata we talked about can be loaded by regular languages.
- E.g. for NPDA: given $q\gamma\$x\$$, push γ on the pushdown (using states not in Q) then switch to q and continue simulating M on x.

General Relationship With Store Languages

Definition

A set of machines $\mathcal{M}$ can be *unloaded* by sets from some family $\mathcal{L}$ if, for all $M \in \mathcal{M}$ with state set Q and store configuration sets $C \in \mathcal{L}$, then there is a machine $M' \in \mathcal{M}$ with state set $Q' \supseteq Q$ that on input $q\gamma\$x\$$, can switch to configuration $q\gamma$ using states in $Q' - Q$, then it simulates M on x, and after reading \$, checks if the current configuration is in C.

- Every type of automata we talked about can be unloaded by regular languages.
- E.g. for NPDA: given $q\gamma\$x\$$, push γ on the store (using states not in Q) then switch to q, then continue simulating M on x until \$ in configuration $p\alpha$.
- To verify $p\alpha$ is in C, M' simulates a DFA accepting C^R by popping one symbol at a time in reverse.

General Relationship With Store Languages

Theorem

Let $\mathcal{M}$ be any machine model that can be loaded and unloaded by sets C from $\mathcal{L}_1 \supseteq \mathsf{REG}$, and let $\mathcal{L}_2 \supseteq \mathsf{REG}$ be closed under intersection.

$\mathcal{S}(\mathcal{M}) \subseteq \mathcal{L}_2$ *if and only if* $\mathrm{post}^*_M(C) \in \mathcal{L}_2$ *and* $\mathrm{pre}^*_M(C) \in \mathcal{L}_2$*, for all* $C \in \mathcal{L}_1, M \in \mathcal{M}$.

General Relationship With Store Languages

Theorem

Let $\mathcal{M}$ of any of the following types:

- NPDA,
- *one-way stack automata,*
- *r-flip* NPDA,
- *reversal-bounded queue automata,*
- *finite-turn Turing machines,*
- *finite-visit Turing machines.*

For all $M \in \mathcal{M}$ and regular configuration sets C, $\mathrm{pre}^*_M(C)$ and $\mathrm{post}^*_M(C)$ are regular.

General Relationship With Store Languages

- To our knowledge, this was previously unknown for all families besides NPDA.
- The proof for NPDA can follow from the store language result.
- Also implies that for many machine models (optionally with reversal-bounded counters), and $C \in \mathcal{L}(\mathsf{NCM})$, we have $\mathrm{pre}^*_M(C)$ and $\mathrm{post}^*_M(C) \in \mathcal{L}(\mathsf{NCM})$.

Common Configurations

Definition

Given two machines M_1, M_2 from a model $\mathcal{M}$, the *common store configuration problem* is to determine if there is some non-initial configuration that appears in an accepting computation of both M_1 and M_2.

- Has applications to fault-tolerance/safety.
- Put all faulty configurations in M_2, and check if there's a common configuration.

Common Configurations

Theorem

Let $\mathcal{M}$ be a machine model, and $\mathcal{L}$ be a language family such that

- $\mathcal{S}(\mathcal{M}) \subseteq \mathcal{L}$,
- *$\mathcal{L}$ has a decidable emptiness problem,*
- *$\mathcal{L}$ is closed under intersection.*

Then $\mathcal{M}$ has a decidable common store configuration problem.

So all the models with either regular or $\mathcal{L}(\text{NCM})$ store languages have a decidable common store configuration problem.

Other Applications

Store languages were a key component of the following results:

- It is decidable whether a given finite-turn Turing machine (resp. finite-turn NPDA, finite-turn queue automaton) has Σ^* as the set of subwords. [Ibarra, IM, JALC 2017].

- For every checking stack automaton M that uses non-constant space, then M cannot use less than linear space (space measured over every accepting computation) [Ibarra, Jirásek, IM, Prigioniero, IJFCS 2021].

- Recent yet to be published proof on decidability of boundedness.

Conclusions

- Store languages are an important but understudied concept in automata theory.
- Store languages of a machine model can often be accepted by a model that is less powerful.
- Characterizing the store languages of a model often enables other algorithmic applications and proofs.

Open Problems

- There are still many types of one-way machines (with a decidable emptiness problem) where we do not have a characterization of their store languages.
- We know little about store languages of two-way machines that accept infinite languages.
- Besides NPDA, we do not know anything about the time/space required to construct store languages.
- Besides NPDA, we do not know anything about the descriptional complexity of their store languages.
- These are important and fundamental questions in automata and formal language theory, and also important for many applications.

Acknowledgements

- Thanks to past co-authors on this work: Oscar Ibarra, Jozef Jirásek Jr., Luca Prigioniero, Noah Friesen, Joey Eremondi.
- Thanks to all the other current and former graduate students in my lab.
- Thanks to the University of Saskatchewan.
- Thanks to the Natural Sciences and Engineering Research Council of Canada (NSERC) for helping fund this work.

- I'm recruiting Master's and PhD students to the Department of Computer Science at the University of Saskatchewan.

Questions?