Automata over Infinite Alphabets

Pushdown Register Automata

Andrzej Murawski
University of Warwick

Nikos Tzevelekos
Queen Mary University of London

`http://warwick.ac.uk/amurawski/esslli15`

ESSLLI 2015

Preliminaries

Let us assume an infinite alphabet $\mathcal{D}$ of **data values** or **names**.

- We shall introduce a simple formalism for computations based on
  - a finite number of $\mathcal{D}$-valued registers,
  - a $\mathcal{D}$-valued pushdown store.

- Writing $[r]$ for $\{1, \cdots, r\}$, by an r-**register assignment** we mean an injective map from $[r]$ to $\mathcal{D}$. We write Reg_r^{i} for the set of all such assignments.

Pushdown register automata

A **pushdown r-register automaton** (r-PDRA) is a tuple

$$\mathcal{A} = \langle\, Q, q_I, \tau_I, \delta, F \,\rangle$$

where:

- Q is a finite set of *states*,
- $q_I \in Q$ is the *initial state*,
- $\tau_I \in Reg_r^{\mathrm{i}}$ is the *initial register assignment*,
- $\delta \subseteq Q \times Op_r \times Q$ is the *transition relation* with

$$Op_r = \{\, i,\ i^\bullet,\ push(i),\ pop(i) \mid 1 \le i \le r \,\} \cup \{\, pop^\bullet \,\}.$$

Configurations, successors, etc

- A **configuration** of an r-PDRA $\mathcal{A}$ is a triple

$$(q, \tau, s) \in Q \times Reg_r^{\mathrm{i}} \times \mathcal{D}^*.$$

- Given $d \in \mathcal{D} \cup \{\epsilon\}$, we write $(q_1, \tau_1, s_1) \xrightarrow{d} (q_2, \tau_2, s_2)$ if $(q_1, op, q_2) \in \delta$ for some $op \in Op_r$ and one of the following conditions holds.
  - $op = i$, $d = \tau_1(i)$, $\tau_2 = \tau_1$, $s_2 = s_1$
  - $op = i^\bullet$, $\forall_i\, d \neq \tau_1(i)$, $d = \tau_2(i)$, $\forall_{j \neq i}\, \tau_2(j) = \tau_1(j)$, $s_2 = s_1$
  - $op = push(i)$, $d = \epsilon$, $\tau_2 = \tau_1$ and $s_2 = \tau_1(i) s_1$
  - $op = pop(i)$, $d = \epsilon$, $\tau_2 = \tau_1$ and $\tau_1(i) s_2 = s_1$
  - $op = pop^\bullet$, $d = \epsilon$, $\tau_2 = \tau_1$, $s_1 = d s_2$, where $\forall_i\, d \neq \tau_1(i)$

Some authors also consider $op = i^\bullet_\epsilon$ with the same meaning as $i^\bullet$ but with $d = \epsilon$.

Acceptance

A **run** of $\mathcal{A}$ is a sequence $\kappa_0, \cdots, \kappa_k$ of configurations such that

- $\kappa_0 = \kappa_I$,
- for all $0 \leq i < k$, $\kappa_i \xrightarrow{d_i} \kappa_{i+1}$ for some $d_i \in \mathcal{D} + \{\epsilon\}$.

A run is **accepting** if $\kappa_k = (q_k, \tau_k)$ for some $q_k \in F$. In this case we say that $\mathcal{A}$ **accepts** $d_0 \cdots d_k \in \mathcal{D}^*$.

The set of all sequences $w \in \mathcal{D}^*$ accepted by $\mathcal{A}$ is called the language of $\mathcal{A}$ and denoted by $\mathcal{L}(\mathcal{A})$.

A language $L \subseteq \mathcal{D}^*$ is called an **PDRA-language** (or a **quasi-context-free** language) if there exists a PDRA that accepts it.

Invariance and distinguishability

- Let $\sigma : \mathcal{D} \to \mathcal{D}$ be a permutation. If $\kappa \xrightarrow{d} \kappa'$ then

$$\sigma(\kappa) \xrightarrow{\sigma(d)} \sigma(\kappa').$$

- r-register automata (without pushdown storage) can take advantage of the registers to distinguish r elements of $\mathcal{D}$ from the rest.
- Consequently, any run can be replaced with a run that ends in the same state, yet is supported by merely $r + 1$ elements of the infinite alphabet.
- With extra pushdown storage, an r-PDRS is capable of storing unboundedly many elements of $\mathcal{D}$. How many elements can we really distinguish?

Exercise

Recall that there exists an r-RA accepting

$$\{d_1 \cdots d_{r+1} \quad | \quad \forall_{i \neq j} d_i \neq d_j\}$$

Task. Construct r-PDRA that accept the following languages.

- ☐ $\{d_1 \cdots d_{2r} \quad | \quad \forall_{i \neq j}\, d_i \neq d_j\}$?
- ☐ $\{d_1 \cdots d_{3r} \quad | \quad \forall_{i \neq j}\, d_i \neq d_j\}$?
- ☐ $\{d_1 \cdots d_{4r} \quad | \quad \forall_{i \neq j}\, d_i \neq d_j\}$?

$3r$ bound

We shall write $\nu(x)$ for the set of elements of $\mathcal{D}$ occurring in x, e.g.

$$\nu(\tau) = \tau([r]) \cap \mathcal{D}.$$

Theorem. Fix an r-PDRA. For every transition sequence transition sequence

$$\rho = (q_0, \tau_0, \epsilon) \vdash^n (q_n, \tau_n, \epsilon),$$

there is a transition sequence

$$\rho' = (q_0, \tau'_0, \epsilon) \vdash^n (q_n, \tau'_n, \epsilon)$$

with $\tau'_0 = \tau_0$, $\tau'_n = \tau_n$ and $|\nu(\rho')| \leq 3r$.

Proof

The proof is by induction on n. When $n \leq 1$, the result is trivial. Otherwise, we distinguish two cases.

- In the first case, the transition sequence is of the form:

$$(q_0, \tau_0, \epsilon) \ \vdash \ (q_1, \tau_1, d) \ \vdash^{n-2} \ (q_{n-1}, \tau_{n-1}, d) \ \vdash \ (q_n, \tau_n, \epsilon)$$

  in which the first transition is by $push(i)$ (so $d = \tau_1(i)$), the last transition is by $pop(j)$ or $pop^\bullet$ and the stack does not empty until the final transition.

- Otherwise, the transition sequence is of the form:

$$(q_0,\ \tau_0,\ \epsilon) \ \vdash^k \ (q_k,\ \tau_k,\ \epsilon) \ \vdash^{n-k} \ (q_n,\ \tau_n,\ \epsilon)$$

  with $0 < k < n$.

Case I

Since d is never popped from the stack during the middle segment, also

$$(q_1, \tau_1, \epsilon) \vdash^{n-2} (q_{n-1}, \tau_{n-1}, \epsilon)$$

is a valid transition sequence and hence, from the induction hypothesis, there is a transition sequence between the same two configurations using no more than $3r$ names.

By adding d to the bottom of every stack in this sequence one obtains another valid transition sequence: $(q_1, \tau_1', d) \vdash^{n-2} (q_{n-1}, \tau_{n-1}', d)$ with $\tau_1' = \tau_1$ and $\tau_{n-1}' = \tau_{n-1}$, and the new sequence features $\leq 3r$ names. It follows that the latter can be extended to the required:

$$(q_0, \tau_0, \epsilon) \;\vdash\; (q_1, \tau_1', d) \;\vdash^{n-2}\; (q_{n-1}, \tau_{n-1}', d) \;\vdash\; (q_n, \tau_n, \epsilon)$$

since neither $push(i)$, nor $pop(j)/pop^\bullet$ change the registers.

Case II

It follows from the induction hypothesis that there are sequences:

$$\rho_1 = (q_0, \tau_0', \epsilon) \vdash^k (q_k, \tau_k', \epsilon) \qquad \rho_2 = (q_k, \tau_k', \epsilon) \vdash^{n-k} (q_n, \tau_n', \epsilon)$$

with $\tau_0' = \tau_0$, $\tau_n' = \tau_n$, $\tau_k' = \tau_k$ and which each, individually, use no more than $3r$ names.

- Let $N \supseteq \nu(\tau_0) \cup \nu(\tau_k) \cup \nu(\tau_n)$ be a set of names of size $3r$. We aim to map $\nu(\rho_1)$ and $\nu(\rho_2)$ into N by injections i and j respectively.
- For i we set $i(a) = a$ for any $a \in \nu(\tau_0) \cup \nu(\tau_k)$ and otherwise choose some *distinct* $b \in N \setminus (\nu(\tau_0) \cup \nu(\tau_k))$.
- Similarly, for j we set $j(a) = a$ for any $a \in (\nu(\tau_k) \cup \nu(\tau_n))$ and otherwise choose some distinct $b \in N \setminus (\nu(\tau_k) \cup \nu(\tau_n))$.

Note that these choices are always possible because $|\nu(\rho_1)|, |\nu(\rho_2)| \leq |N|$. Finally, we extend i and j to permutations σ_i and σ_j on $\mathcal{D}$.

Final step

Since transition sequences are closed under permutations

$$\rho = (q_0, \sigma_i \cdot \tau_0, \epsilon) \vdash^k (q_k, \sigma_i \cdot \tau_k = \sigma_j \cdot \tau_k, \epsilon) \vdash^{n-k} (q_n, \sigma_j \cdot \tau_n, \epsilon)$$

is a valid transition sequence with

- $\sigma_i \cdot \tau_0 = \tau_0$,
- $\sigma_j \cdot \tau_n = \tau_n$,
- $\nu(\rho) \subset N$.

Closure properties

☺

- ☐ union
- ☐ concatenation
- ☐ star

☹

- ☐ complementation
- ☐ intersection

Related topic: data languages

data word = tag + data value

To come: more models of computation over infinite alphabets!

Decision problems

PDRA emptiness/reachability is decidable thanks to the $3r$ result.

Complexity table

register assignments	injective filled	injective with $\#$	non-injective
RA	NL	NP	PSPACE
PDRA	EXPTIME	EXPTIME	EXPTIME

Bibliography

- ☐ Register automata [KF94, NSV04]
- ☐ Pushdown register automata [CK98, Seg06, MRT14]
- ☐ More [NSV04, Seg06, BS07, BKL14]

[BKL14] Mikolaj Bojanczyk, Bartek Klin, and Slawomir Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3), 2014.

[BS07] H. Björklund and T. Schwentick. On notions of regularity for data languages. In *Proceedings of FCT*, volume 4639 of *Lecture Notes in Computer Science*, pages 88–99. Springer, 2007.

[CK98] E. Y. C. Cheng and M. Kaminski. Context-free languages over infinite alphabets. *Acta Inf.*, 35(3):245–267, 1998.

[KF94] M. Kaminski and N. Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

[MRT14] A. S. Murawski, S. J. Ramsay, and N. Tzevelekos. Reachability in pushdown register automata. In *Proceedings of MFCS*, LNCS, pages 464–473. Springer, 2014.

[NSV04] F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004.

[Seg06] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Proceedings of CSL*, volume 4207 of *Lecture Notes in Computer Science*. Springer, 2006.