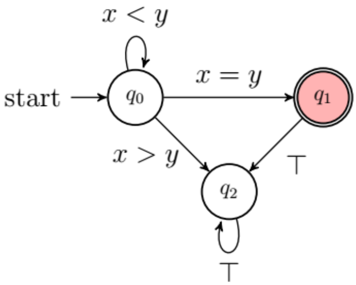
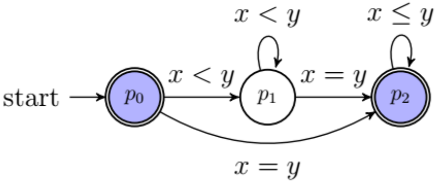
Parametrized Automata

Franziska Alber

University of Regensburg

AVM 2024





Motivation

- Finite-State Automata (FSA): Regular languages over finite alphabets
- What if the alphabet is infinite? (E.g., arrays using real numbers...)
- Lots and lots of extensions of FSA for infinite alphabets
- In this presentation: Quick, example-driven introduction to Parametrized Automata (PA) [1], especially determinism and complementation
- Not in this presentation: Implementation

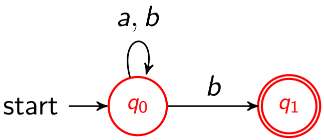
[1] See Parametrized Automata over Infinite Alphabets: Properties and Complementation, Franziska Alber, Master thesis, 2024 (coming soon!)

Finite-State Automata

An FSA $A = (D, Q, q_0, \delta, F)$ is a tuple of:

- D: finite, non-empty alphabet
- Q: finite set of states
- $q_0 \in Q$: initial state
- $\delta \subseteq Q \times D \times Q$: transition relation
- $F \subseteq Q$: set of accepting states

Alphabet: $D = \{a, b\}$, examples of words over D: $ab, babbba, \ldots$

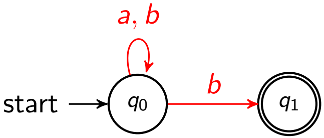


Finite-State Automata

An FSA $A = (D, Q, q_0, \delta, F)$ is a tuple of:

- D: finite, non-empty alphabet
- Q: finite set of states
- $q_0 \in Q$: initial state
- $\delta \subseteq Q \times D \times Q$: transition relation
- $F \subseteq Q$: set of accepting states

Alphabet: $D = \{a, b\}$, examples of words over D: $ab, babbba, \ldots$

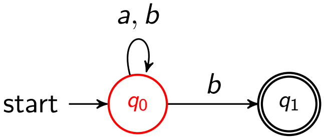


Finite-State Automata

An FSA $A = (D, Q, q_0, \delta, F)$ is a tuple of:

- D: finite, non-empty alphabet
- Q: finite set of states
- $q_0 \in Q$: initial state
- $\delta \subseteq Q \times D \times Q$: transition relation
- $F \subseteq Q$: set of accepting states

Alphabet: $D = \{a, b\}$, examples of words over D: $ab, babbba, \ldots$

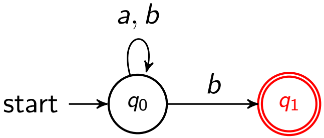


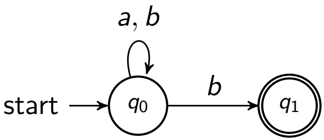
Finite-State Automata

An FSA $A = (D, Q, q_0, \delta, F)$ is a tuple of:

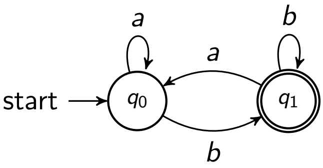
- D: finite, non-empty alphabet
- Q: finite set of states
- $q_0 \in Q$: initial state
- $\delta \subseteq Q \times D \times Q$: transition relation
- $F \subseteq Q$: set of accepting states

Alphabet: $D = \{a, b\}$, examples of words over D: $ab, babbba, \ldots$



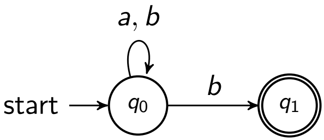


(a) A_1

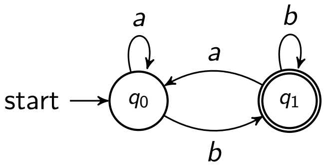


(b) A_2

Determinism and Complementation



(a) A_1



(b) A_2

Determinism:

- transition function
- exactly one possible path/run for every word
- in every state, exactly one exiting transition for every letter

Determinism leads to easy complementation!

Towards Infinite Alphabets

- Register Automata/Finite-Memory Automata [19]
- Symbolic Automata [12]
- Variable Automata [16]
- Parametrized Automata [18]
- Symbolic Register Automata [10]
- Register Set Automata [17]
- Extended Symbolic Finite Automata [11]
- Fresh-Variable Automata [3]
- Guarded Variable Automata over Infinite Alphabets [4]
- Register Automata with Linear Arithmetic [9]

Second Slide to Prove a Point

- Single-Use Register Automata [8]
- Pebble Automata [21]
- Usage Automata [2]
- Parametric Semilinear Data Automata [15]
- Parikh Automata [20]
- Timed Automata [14]
- Deterministic Memory Automata over Ordered Data[5]
- Data Automata and Two-Variable First-Order Logic [6]
- Alternating 1-Register Automata and LTL with Freeze Quantifiers [13]
- Nominal Automata [7]
- Streaming Data-String Acceptors [1]

Meet the Parents

Symbolic Automata use logical formulas ("guards") instead of letters. Properties are similar to FSA.

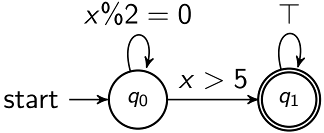


Figure: A symbolic automaton.

Variable Automata compare input letters to non-reassignable variables $y_1, y_2, \ldots, y_k$. The symbol z represents all letters not assigned to a variable.

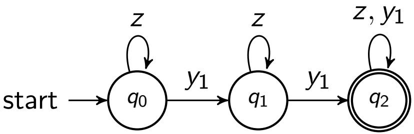


Figure: A variable automaton.

Meet the Parents

Symbolic Automata use logical formulas ("guards") instead of letters. Properties are similar to FSA.

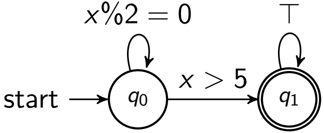
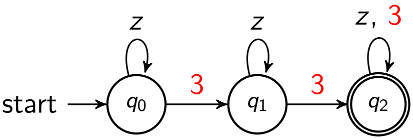


Figure: A symbolic automaton.

Variable Automata compare input letters to non-reassignable variables $y_1, y_2, \ldots, y_k$. The symbol z represents all letters not assigned to a variable.



$\mu(y_1) = 3$

Parametrized Automata

A PA $A = (M, Q, q_0, \delta, F)$ is a tuple of:

- $M = (D, I)$: structure where D is an infinite alphabet
- Q: finite set of states
- $q_0 \in Q$: initial state
- $\delta \subseteq Q \times \Phi \times Q$: transition relation ($\Phi$: set of formulas)
- $F \subseteq Q$: set of accepting states

x: placeholder for current letter, y: parameter

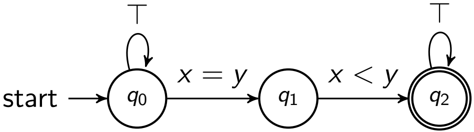


Figure: P_1 accepts all unsorted words and cannot be complemented.

Parametrized Automata are not closed under complementation! (Proof at the end)

Determinism per Assignment

- "Local" approach: Exactly one possible exiting transition for every state, letter and parameter assignment
- (+) Always works: Algorithms for Symbolic Automata are applicable
- (−) Does not imply that every word can take exactly one path ⇒ not sufficient for complementation!

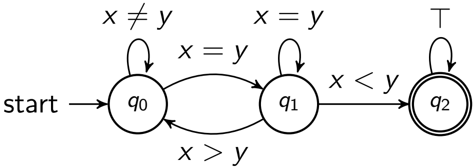


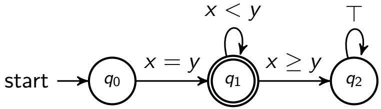
Figure: P_1': equivalent to P_1 and deterministic per assignment

Example: Word $w = (3, 2)$

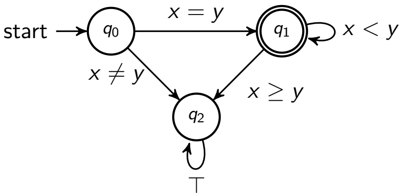
assignment	term. state
$\mu(y) = 3$	q_2
$\mu(y) = 0$	q_0
$\mu(y) = 2$	q_1

Strongly Deterministic Parametrized Automata (SDPA)

- "Global" approach: Exactly one possible path for every word
- (+) Easy complementation
- (−) Not every PA is equivalent to an SDPA (not even every complementable PA!)



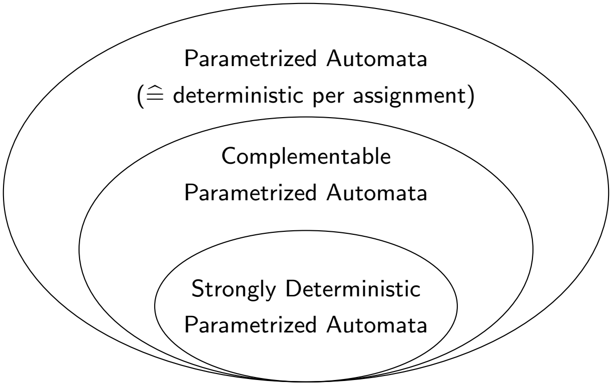
(a) Strongly deterministic



(b) Deterministic per assignment

Figure: Two different representations for the language of words in which the first letter is strictly largest.

Relations between the different subclasses of parametrized automata



Complementable PA $\subsetneq$ PA

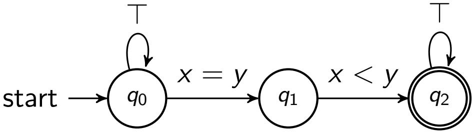
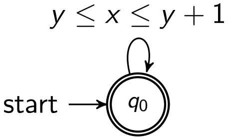


Figure: P_1 identifying all unsorted words.

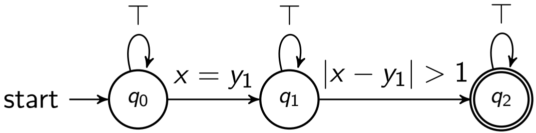
There is no PA identifying the complement language of $L(P_1)$:

- Complement PA of P_1 has to identify all sorted words (letters in ascending order)
- Assume P_1^c exists, n states $\Rightarrow$ the word $(1, 2, \ldots 3n)$ traverses some state thrice $\Rightarrow$ loop in section $(i, i+1, \ldots, i+k)$
- Word $(1, \ldots, i+k, i, \ldots, n)$ falsely accepted.

SDPA $\subsetneq$ Complementable PA



(a) A_3



(b) A_3^c

There is no SDPA A equivalent to A_3^c:

- In SDPA, the paths of prefixes of words are predetermined
- Consider sequence of words $(1, \frac{1}{2}, 2), (1, \frac{1}{2}, \frac{1}{3}, 1 + \frac{1}{2}), \ldots$
- Prove all paths of prefixes $(1, \frac{1}{2}, \ldots, \frac{1}{i})$ have to terminate in distinct states
- Number of states unbounded $\Rightarrow$ A does not exist.

Master Thesis

Rajeev Alur and Pavol Cerný.
Streaming transducers for algorithmic verification of single-pass list-processing programs.
In Thomas Ball and Mooly Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pages 599–610. ACM, 2011.

Massimo Bartoletti.
Usage automata.
In Pierpaolo Degano and Luca Viganò, editors, *Foundations and Applications of Security Analysis, Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security, ARSPA-WITS 2009, York, UK, March 28-29, 2009, Revised Selected Papers*, volume 5511 of *Lecture Notes in Computer Science*, pages 52–69. Springer, 2009.

Walid Belkhir, Yannick Chevalier, and Michaël Rusinowitch.
Fresh-variable automata for service composition.
CoRR, abs/1302.4205, 2013.

Walid Belkhir, Yannick Chevalier, and Michaël Rusinowitch.
Guarded variable automata over infinite alphabets.
CoRR, abs/1304.6297, 2013.

Michael Benedikt, Clemens Ley, and Gabriele Puppis.
What you must remember when processing data words.
In Alberto H. F. Laender and Laks V. S. Lakshmanan, editors, *Proceedings of the 4th Alberto Mendelzon International Workshop on Foundations of Data Management, Buenos Aires, Argentina, May 17-20, 2010*, volume 619 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

Mikolaj Bojanczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin.
Two-variable logic on data words.
ACM Trans. Comput. Log., 12(4):27:1–27:26, 2011.

Mikolaj Bojanczyk, Bartek Klin, and Slawomir Lasota.
Automata theory in nominal sets.
Log. Methods Comput. Sci., 10(3), 2014.

Mikolaj Bojanczyk and Rafal Stefanski.
Single-use automata and transducers for infinite alphabets.
In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 113:1–113:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

Yu-Fang Chen, Ondrej Lengál, Tony Tan, and Zhilin Wu.
Register automata with linear arithmetic.
In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017.

Loris D'Antoni, Tiago Ferreira, Matteo Sammartino, and Alexandra Silva.
Symbolic register automata.
In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I*, volume 11561 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2019.

Loris D'Antoni and Margus Veanes.
Extended symbolic finite automata and transducers.
Formal Methods Syst. Des., 47(1):93–119, 2015.

Loris D'Antoni and Margus Veanes.
The power of symbolic automata and transducers.
In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 47–67. Springer, 2017.

Stéphane Demri and Ranko Lazic.
LTL with the freeze quantifier and register automata.
ACM Trans. Comput. Log., 10(3):16:1–16:30, 2009.

Diego Figueira, Piotr Hofman, and Slawomir Lasota.
Relating timed and register automata.
In Sibylle B. Fröschle and Frank D. Valencia, editors, *Proceedings 17th International Workshop on Expressiveness in Concurrency, EXPRESS'10, Paris, France, August 30th, 2010*, volume 41 of *EPTCS*, pages 61–75, 2010.

Diego Figueira and Anthony Widjaja Lin.
Reasoning on data words over numeric domains.
In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 37:1–37:13. ACM, 2022.

Orna Grumberg, Orna Kupferman, and Sarai Sheinvald.
Variable automata over infinite alphabets.
In Adrian-Horia Dediu, Henning Fernau, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications, 4th International Conference, LATA 2010, Trier, Germany, May 24-28, 2010. Proceedings*, volume 6031 of *Lecture Notes in Computer Science*, pages 561–572. Springer, 2010.

Sabína Gulcíková and Ondrej Lengál.
Register set automata (technical report).
CoRR, abs/2205.12114, 2022.

Artur Jez, Anthony W. Lin, Oliver Markgraf, and Philipp Rümmer.
Decision procedures for sequence theories.
In Constantin Enea and Akash Lal, editors, *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part II*, volume 13965 of *Lecture Notes in Computer Science*, pages 18–40. Springer, 2023.

Michael Kaminski and Nissim Francez.
Finite-memory automata.
Theor. Comput. Sci., 134(2):329–363, 1994.

Felix Klaedtke and Harald Rueß.
Monadic second-order logics with cardinalities.
In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*, volume 2719 of *Lecture Notes in Computer Science*, pages 681–696. Springer, 2003.

Frank Neven, Thomas Schwentick, and Victor Vianu.
Finite state machines for strings over infinite alphabets.
ACM Trans. Comput. Log., 5(3):403–435, 2004.