Machines, Models, Monoids, and Modal logic

Sam van Gool

University of Amsterdam and City College of New York

September 2017

Tbilisi Symposium on Language, Logic and Computation

Lagodekhi, Georgia

Outline

1. Part I: Formal Languages, Automata, and Algebra
2. Part II: Duality and Varieties of Monoids
3. Part III: Profiniteness, Pointlikes, and the Future

Outline

1. Part I: Formal Languages, Automata, and Algebra
2. Part II: Duality and Varieties of Monoids
3. Part III: Profiniteness, Pointlikes, and the Future

Outline Part I

1. What is a formal language?
   - Alphabets and words
   - Formal languages
2. How to describe a formal language?
   - Automata
   - Logic
   - (Open) Problems
3. How to understand formal languages?
   - Boolean algebras with operators
   - Model theory
   - Monoids

Formal language theory

- A mathematical setting for analyzing computational problems.
- Or: ... formal grammars.
- All definitions are elementary.
- Many problems are difficult, interesting, and often open.

1. What is a formal language?
   - Alphabets and words
   - Formal languages

Alphabets and words

- An *alphabet* is a finite set of symbols, Σ.
- A finite Σ-word is a finite sequence of elements of Σ.

Alphabets and words

Examples

- If $\Sigma = \{b, l, i, s, t, B, L, I, S, T\}$ then three examples of (distinct!) Σ-words are: tbilisi, Tbilisi, and TBILISI.
- If $\Sigma = \{\texttt{enter\_coin}, \texttt{push\_cola}, \texttt{push\_water}\}$ then three examples of Σ-words are: $(\texttt{enter\_coin}, \texttt{push\_cola})$, $(\texttt{push\_cola}, \texttt{push\_water}, \texttt{push\_cola})$, and $(\texttt{push\_cola}, \texttt{push\_cola}, \texttt{push\_cola}, \texttt{push\_cola}, \texttt{push\_cola})$. The last one can be briefly denoted as: $\texttt{push\_cola}^5$.
- The empty word, ϵ, is a word in any alphabet.

Formal Languages

- *Notation:* Σ^* is the set of all Σ-words.
- A (formal) Σ-*language* is a subset L of Σ^*.

Examples

- The *empty language*, $\emptyset$.
- The language containing only the empty word, $\{\epsilon\}$.
- The set of all Σ-words, Σ^*.
- The set of non-empty words is a language, $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$.

Formal Languages

Examples

- Let Σ be the set of all lower-case letters, capital letters, numbers, and the symbols `!`, `@`, `#`, `$`, `*`, `(`, `)`, and `%`. An example of a Σ-language is

  $$\texttt{PW} = \{w \in \Sigma^* \mid w \text{ is at least 8 characters long and contains at least one letter, one number, and one special symbol}\}.$$

- Let $\Sigma = \{\texttt{enter\_coin}, \texttt{push\_cola}, \texttt{push\_water}\}$. An example of a Σ-language is

  $$\texttt{BUY} = \{w \in \Sigma^* \mid w \text{ contains an occurrence of } \texttt{enter\_coin} \text{ before an occurrence of } \texttt{push\_cola} \text{ or } \texttt{push\_water}\}.$$

- Let $\Sigma = \{0, 1\}$. Three examples of $\{0, 1\}$-languages are:

  $\texttt{FACTOR01} = \{w \in \{0,1\}^* \mid w \text{ contains '01' as a factor}\}$.

  $\texttt{EVENONES} = \{w \in \{0,1\}^* \mid \text{ the number of 1's in } w \text{ is even}\}$.

  $\texttt{NON1} = \{0^n 1^n \mid n \geq 0\}$.

2. How to describe a formal language?
   - Automata
   - Logic
   - (Open) Problems

Describing formal languages

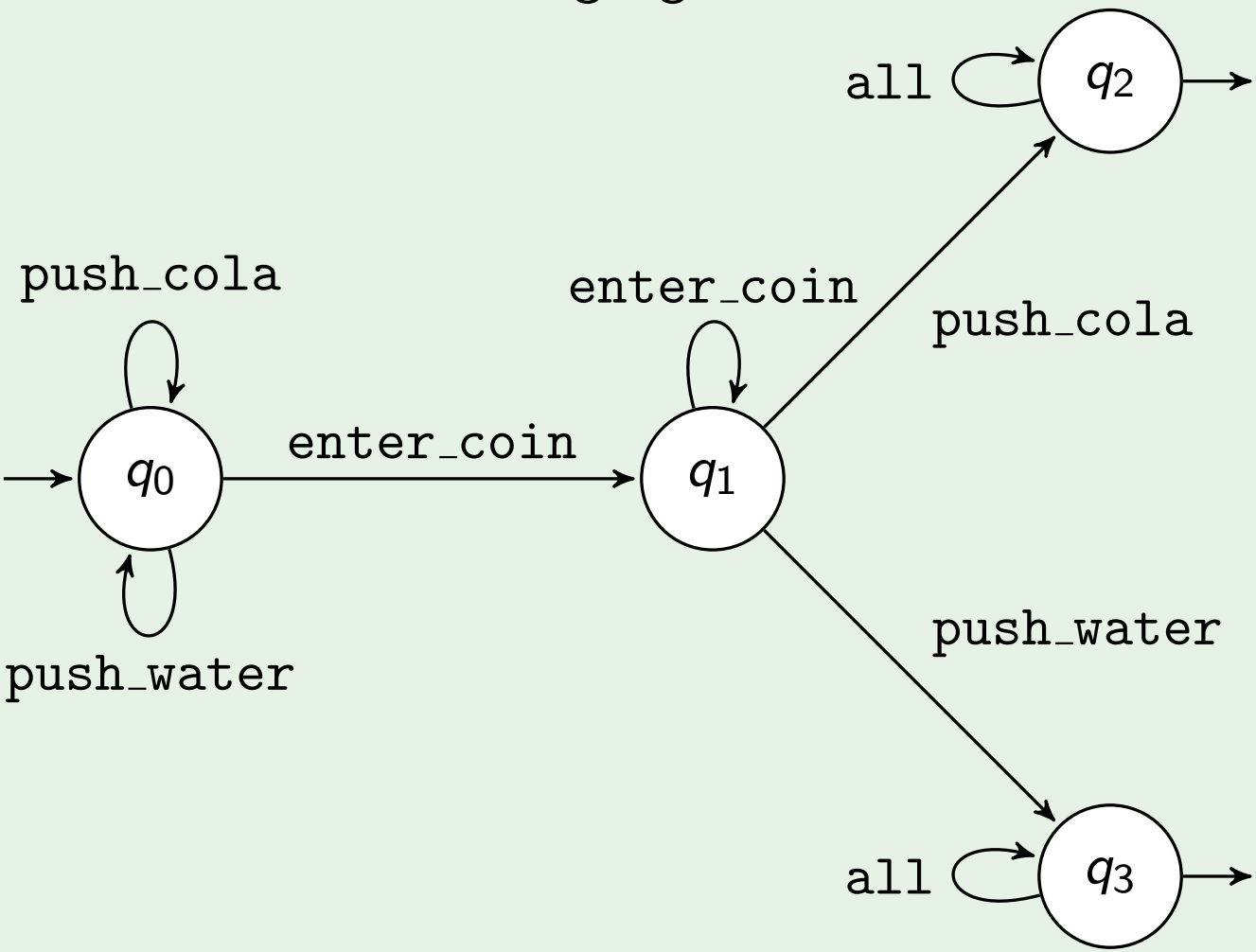
- Formal grammars
- Machines
- Logic

In this tutorial, we will focus on the last two, and we will mostly restrict to regular languages.

Automata

Examples

An automaton for the language BUY.

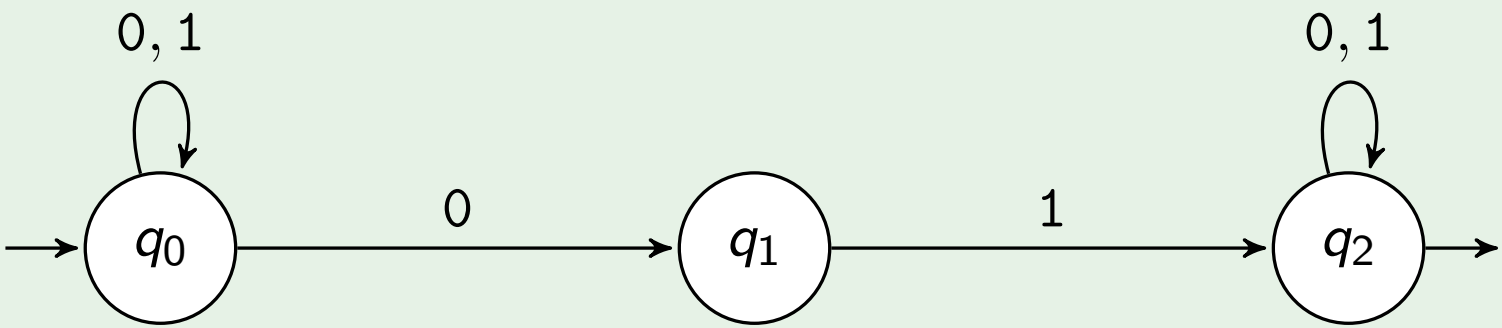


Automata

Examples

An automaton for the language

$\texttt{FACTOR01} = \{w \in \{0,1\}^* \mid w \text{ contains '01' as a factor}\}$.



Automata

- An *automaton* is a tuple $\mathcal{A} = (Q, \Sigma, \delta)$, where
  - Q is a finite set of *states*,
  - Σ is a finite *alphabet*,
  - $\delta\colon Q \times \Sigma \to \mathcal{P}(Q)$ is a *transition function*.
- *Notation:* $q \xrightarrow{a} q'$ means:
  q is a state in Q, a is a letter in Σ, and q' is a state in $\delta(q, a)$.
- Pick two sets of states, I and F, in Q.
- A word $a_1 \dots a_n \in \Sigma^*$ is *accepted by the automaton $\mathcal{A}$ with initial states I and final states F* if there exists a path $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$ such that $q_0 \in I$ and $q_n \in F$.
- The language $L_{\mathcal{A},I,F}$ of all accepted words is called the language *recognized* by $\mathcal{A}$ with initial states I and final states F.
- Full technical name: *non-deterministic finite automaton* (NFA).
- *Deterministic* (DFA): $\delta\colon Q \times \Sigma \to Q$.

Regular languages

- A language L is called *regular* if there exists an NFA that recognizes it.

Fact

There exists a non-deterministic finite automaton that recognizes L if, and only if, there exists a deterministic finite automaton that recognizes L.

Exercises

1. Describe an automaton that recognizes `EVENONES`.
2. Describe an automaton that recognizes `PW`.
3. Describe a *deterministic* automaton that recognizes `FACTOR01`.
4. (*) Is it possible to find an automaton that recognizes `NON1`? If no, explain why not.

Logic

Examples

- The language $\texttt{BUY}$ of action sequences for buying has logic description

$$\exists x \exists y \left[x < y \wedge \texttt{enter\_coin}(x) \wedge (\texttt{push\_cola}(y) \vee \texttt{push\_water}(y)) \right].$$

- The language $\texttt{EVENLENGTH}$ of $\{0,1\}$-words of even length has logic description

$$\texttt{empty} \vee \exists P \left[P(\texttt{first}) \wedge \neg P(\texttt{last}) \wedge \forall x \neq \texttt{last} \left(P(x) \leftrightarrow \neg P(\texttt{S}(x)) \right) \right].$$

Logic

- Syntax:
  - Basic propositional connectives: $\wedge$, $\neg$.
  - Quantification over first-order variables x, y, ... and monadic second-order variables P, Q, ...
  - Atomic formulas: $x < y$, $P \subseteq Q$, $P(x)$, $\mathrm{a}(x)$ for $a \in \Sigma$.
- Semantics: view a word $w = a_1 \dots a_n$ as a *structure* W, i.e.,
  - The underlying set of W is $\{1, \dots, n\}$.
  - The natural linear order $<^W$ interprets the binary predicate $<$.
  - For every letter $a \in \Sigma$, a^W is the set of positions i where $a_i = a$.
- For a sentence φ, $L_\varphi = \{w \in \Sigma^* \mid w \models \varphi\}$.
- Shortcuts such as $\mathrm{S}(x)$, `first`, `last`, `empty`, ... are definable.
- This is Monadic Second Order (MSO) Logic.
- First Order (FO) Logic is obtained by disallowing second order variables and second order quantifiers.

Exercises

1. Describe the language `FACTOR01` with MSO logic, or FO logic if possible.
2. Describe the language `PW` with MSO logic, or FO logic if possible.
3. Describe the language `EVENONES` with MSO logic, or FO logic if possible.
4. If you think it is impossible to find an FO logic definition in (1)–(3), explain why.
5. What is the lowest possible quantifier depth you need to describe `PW` and `EVENONES`? (*) Can you prove it?
6. (*) Is it possible to describe the language `NON1` with an MSO formula? If no, why not?

Logic and automata

Theorem (Büchi 1960)

Let $L \subseteq \Sigma^*$ be a language. Then L is regular if, and only if, L is definable in MSO logic.

Proof ingredients.

- The behavior of any automaton can be 'described' in MSO logic.
- MSO logic can be simulated by automata. □

From here on, regular = MSO-definable.

Problems

- Given an automaton, decide if it accepts any words?
- Given a regular language, decide if it is FO-definable?
- Given an FO-definable language, decide if it is definable in FO_k, i.e., FO logic of quantifier depth $\leq k$?
- Given two regular languages, decide if they are *separable* by an FO-definable language?
  - A language M *separates* L_1 from L_2 if $L_1 \subseteq M$ and $L_2 \cap M = \emptyset$.
- Given two regular languages, decide if they are *separable* by an FO_k-definable language?
- ...

Problems

- Given an automaton, decide if it accepts any words? See next slides
- Given a regular language, decide if it is FO-definable? See Part II
- Given an FO-definable language, decide if it is definable in FO_k, i.e., FO logic of quantifier depth $\leq k$? Open for $k \geq 3$
- Given two regular languages, decide if they are *separable* by an FO-definable language? See Part III
  - A language M *separates* L_1 from L_2 if $L_1 \subseteq M$ and $L_2 \cap M = \emptyset$.
- Given two regular languages, decide if they are *separable* by an FO_k-definable language? Open for $k \geq 3$
- ...

3. How to understand formal languages?
   - Boolean algebras with operators
   - Model theory
   - Monoids

Boolean algebras with operators

- The set of all Σ-languages, $\mathcal{P}(\Sigma^*)$, is a Boolean algebra with operations $\cup$ (union) and $()^c$ (complement).
- For any letter $a \in \Sigma$, the function

$$L \mapsto a^{-1}L = \{w \in \Sigma^* \mid aw \in L\}$$

is an *endomorphism* of the Boolean algebra, and so is

$$L \mapsto La^{-1} = \{w \in \Sigma^* \mid wa \in L\}.$$

Fact

- *If L_1, L_2 are regular, then $L_1 \cup L_2$ is regular.*
- *If L is regular, then L^c is regular.*
- *If L is regular, then $a^{-1}L$ and La^{-1} are regular.*

Quotient operators shift initial and final states

Proof of last item.

- Suppose that $\mathcal{A}$ is an NFA that recognizes L with initial states I and final states F.
- Then $a^{-1}L$ is recognized by $\mathcal{A}$ with final states F and initial states Ia, i.e., the set of states q which admit a transition $q_0 \xrightarrow{a} q$ for some $q_0 \in I$.
- Also, La^{-1} is recognized by $\mathcal{A}$ with initial states I and final states $a^{-1}F$, i.e., the set of states q which admit a transition $q \xrightarrow{a} q_F$ for some $q_F \in F$. □

Corollary

If L is regular, then the set $\{w^{-1}L, Lw^{-1} \mid w \in \Sigma^*\}$ is finite.

Boolean algebras with operators

- A Boolean subalgebra $B \leq \mathcal{P}(\Sigma^*)$ is *closed* if, for every L in B and a in Σ, both $a^{-1}L$ and La^{-1} are in B.
- The set of *regular* Σ -languages, $\mathrm{Reg}(\Sigma^*)$, is a closed subalgebra of $\mathcal{P}(\Sigma^*)$.
- For any automaton $\mathcal{A} = (Q, \Sigma, \delta)$, the set of Σ-languages which $\mathcal{A}$ can recognize is a finite closed subalgebra of $\mathrm{Reg}(\Sigma^*)$.
- Any Σ-language L *generates* a closed subalgebra, $B(L)$, i.e., the *smallest* closed subalgebra containing L.

Proposition

A language $L \in \mathcal{P}(\Sigma^*)$ is regular if, and only if, $B(L)$ is finite.

Exercises

1. Describe the closed subalgebra generated by the $\{0,1\}$-language $\texttt{EVENLENGTH}$.
2. Let $S \subseteq \mathbb{N}$. Describe the closed subalgebra generated by the $\{1\}$-language $\texttt{LENGTH}_S$ of $\{1\}$-words of length S.
3. (*) When is the algebra in (2) finite?

Model theory

- Let T_Σ be the MSO theory of finite Σ-words, i.e., the set of MSO sentences that are true in all finite Σ-words.
- Let $\mathcal{L}(T_\Sigma)$ be the Lindenbaum algebra of T, i.e., the algebra of MSO-sentences up to T_Σ-equivalence. This is a Boolean algebra.
- To any $[\varphi]_{T_\Sigma}$ in $\mathcal{L}(T_\Sigma)$, associate the regular language, $L(\varphi)$, described by φ.
- This assignment is a well-defined isomorphism between $\mathcal{L}(T_\Sigma)$ and $\mathrm{Reg}(\Sigma^*)$.
- Exercise: (*) Describe the operators $L \mapsto a^{-1}L$ and $L \mapsto La^{-1}$ directly on the Lindenbaum algebra $\mathcal{L}(T_\Sigma)$.
- Under this isomorphism, the subalgebra of FO-sentences corresponds to a subalgebra of $\mathrm{Reg}(\Sigma^*)$. Which? See Part II

Semigroups and monoids

- A *semigroup* is a pair $(S, \cdot)$, where $\cdot$ is an associative operation, i.e., $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ for all x, y, z in S.
- A *monoid* is a semigroup that contains an *identity element*, 1, i.e., $1 \cdot x = x \cdot 1$ for all x in S.

Monoids

Examples

- The set Σ^*, with multiplication $u \cdot v := uv$.
- For any set P, the set of functions from P to itself, $(P \to P)$, with multiplication $f \cdot g := f \circ g$.
- In particular, an NFA $\mathcal{A} = (Q, \Sigma, \delta)$ gives, for every $a \in \Sigma$, a function $\Diamond_a$ in $(\mathcal{P}(Q) \to \mathcal{P}(Q))$, defined by

$$\Diamond_a(R) := \{q \mid q \xrightarrow{a} q' \text{ for some } q' \in R\}.$$

Exercises

1. Show that Σ^* is a monoid.
2. Show that $(P \to P)$ is a monoid.
3. Show that Σ^* is the *free* monoid on Σ, i.e., that for any monoid M and any function $f : \Sigma \to M$, there is a unique homomorphism $\bar{f} : \Sigma^* \to M$ extending f.
4. Applying (3) to the function $\Diamond : \Sigma \to (\mathcal{P}(Q) \to \mathcal{P}(Q))$, give an explicit description of the function $\bar{\Diamond} : \Sigma^* \to (\mathcal{P}(Q) \to \mathcal{P}(Q))$.
5. (*) Show that $\mathcal{A}$ with initial states I and final states F accepts a word $w \in \Sigma^*$ if, and only if, $I \cap \bar{\Diamond}_w(F) \neq \emptyset$.

References for Part I

- An accessible textbook introduction to the field:

  P. Linz. *An introduction to formal languages and automata*. 5th ed. Jones & Bartlett, 2012

- A more advanced, but very readable introduction to logic on words:

  H. Straubing. *Finite automata, formal logic, and circuit complexity*. Progress in Theoretical Computer Science. Birkhäuser Boston Inc., Boston, 1994

- Our recent work on applications of model theory: (see also part II)

  S. J. v. Gool and B. Steinberg. "Pro-aperiodic monoids via saturated models". In: *STACS 2017*. Vol. 66. LIPIcs. https://arxiv.org/abs/1609.07736. 2017, 39:1–39:14

- A more category-theoretic view of formal language theory: (see also part II)

  M. Gehrke, D. Petrisan, and L. Reggio. "Quantifiers on languages and codensity monads". In: *TACL 2017*. https://arxiv.org/abs/1702.08841. 2017