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
Outline

1. Part I: Formal Languages, Automata, and Algebra
2. Part II: Duality and Varieties of Monoids
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.
Formal language theory

- A mathematical setting for analyzing computational problems.
- Or: ... formal grammars.
Formal language theory

- A mathematical setting for analyzing computational problems.
- Or: ... formal grammars.
- All definitions are elementary.
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.
What is a formal language?
- Alphabets and words
- Formal languages
Alphabets and words

- An alphabet is a finite set of symbols, $\Sigma$.
- A finite $\Sigma$-word is a finite sequence of elements of $\Sigma$. 
Alphabets and words

Examples

- If \( \Sigma = \{b, l, i, s, t, B, L, I, S, T\} \)
Alphabets and words

Examples

* If \( \Sigma = \{b, l, i, s, t, B, L, I, S, T\} \) then three examples of (distinct!) \( \Sigma \)-words are: tbilisi, Tbilisi, and TBILISI.

* If \( \Sigma = \{\text{enter coin}, \text{push cola}, \text{push water}\} \) then three examples of \( \Sigma \)-words are: (enter coin, push cola), (push cola, push water, push cola), and (push cola, push cola, push cola, push cola, push cola).
  The last one can be briefly denoted as: push cola \( 5 \).
Alphabets and words

Examples

- If $\Sigma = \{b, l, i, s, t, B, L, I, S, T\}$ then three examples of (distinct!) $\Sigma$-words are: tbilisi, Tbilisi, and TBILISI.

- If $\Sigma = \{\text{enter coin}, \text{push cola}, \text{push water}\}$
Alphabets and words

Examples

- If $\Sigma = \{b, l, i, s, t, B, L, I, S, T\}$ then three examples of (distinct!) $\Sigma$-words are: tbilisi, Tbilisi, and TBILISI.

- If $\Sigma = \{\text{enter\_coin, push\_cola, push\_water}\}$ then three examples of $\Sigma$-words are: $(\text{enter\_coin, push\_cola})$, $(\text{push\_cola, push\_water, push\_cola})$, and $(\text{push\_cola, push\_cola, push\_cola, push\_cola, push\_cola, push\_cola})$. The last one can be briefly denoted as: $\text{push\_cola}^5$. 
Alphabets and words

Examples

- If $\Sigma = \{b, l, i, s, t, B, L, I, S, T\}$ then three examples of (distinct!) $\Sigma$-words are: tbilisi, Tbilisi, and TBILISI.

- If $\Sigma = \{\text{enter\_coin}, \text{push\_cola}, \text{push\_water}\}$ then three examples of $\Sigma$-words are: $(\text{enter\_coin}, \text{push\_cola})$, $(\text{push\_cola}, \text{push\_water}, \text{push\_cola})$, and $(\text{push\_cola}, \text{push\_cola}, \text{push\_cola}, \text{push\_cola}, \text{push\_cola}, \text{push\_cola})$. The last one can be briefly denoted as: $\text{push\_cola}^5$.

- The empty word, $\epsilon$, is a word in any alphabet.
Formal Languages

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

Formal Languages

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

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

Examples

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

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

Examples

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

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

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

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

Examples

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

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

- Let $\Sigma = \{ \text{enter_coin}, \text{push_cola}, \text{push_water} \}$. An example of a $\Sigma$-language is

  $BUY = \{ w \in \Sigma^* \mid w \text{ contains an occurrence of enter_coin before an occurrence of push_cola or push_water} \}$.

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

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

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

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

$q_0$
Examples

An automaton for the language **BUY**.
Automata

Examples
An automaton for the language BUY.

\[ q_0 \xrightarrow{\text{enter coin}} q_1 \xrightarrow{\text{push cola}} q_2 \]
Automata

Examples

An automaton for the language BUY.

\[ q_0 \xrightarrow{\text{enter\_coin}} q_1 \xrightarrow{\text{push\_cola}} q_2 \]

\[ q_0 \xrightarrow{\text{enter\_coin}} q_1 \xrightarrow{\text{push\_water}} q_3 \]
Automata

Examples

An automaton for the language \textbf{BUY}.

\begin{tikzpicture}
    \node[state,initial] (q0) at (0,0) {$q_0$};
    \node[state] (q1) at (2,0) {$q_1$};
    \node[state] (q2) at (4,2) {$q_2$};
    \node[state] (q3) at (4,-2) {$q_3$};
    \draw[->] (q0) edge node {\text{enter\_coin}} (q1);
    \draw[->] (q1) edge node {\text{push\_cola}} (q2);
    \draw[->] (q1) edge node {\text{push\_water}} (q3);
    \draw[->] (q0) edge[loop above] node {\text{push\_cola}} (q0);
\end{tikzpicture}
Automata

Examples

An automaton for the language BUY.

\[q_0 \xrightarrow{\text{push cola}} q_1 \xrightarrow{\text{push cola}} q_2 \xrightarrow{\text{push cola}} q_3 \xrightarrow{\text{push water}} q_1 \xrightarrow{\text{push water}} q_3\]
Automata

Examples

An automaton for the language BUY.

\[
\begin{aligned}
q_0 &\xrightarrow{\text{push\_cola}} q_1 \\
q_0 &\xrightarrow{\text{push\_water}} q_0 \\
q_1 &\xrightarrow{\text{enter\_coin}} q_2 \\
q_1 &\xrightarrow{\text{push\_cola}} q_2 \\
q_2 &\xrightarrow{\text{push\_cola}} q_2 \\
q_2 &\xrightarrow{\text{push\_water}} q_3 \\
q_3 &\xrightarrow{\text{push\_water}} q_3
\end{aligned}
\]
Automata

Examples

An automaton for the language BUY.

q_0 \xrightarrow{\text{push}_\text{cola}} q_0 \xrightarrow{\text{push}_\text{water}} q_0

q_0 \xrightarrow{\text{enter}_\text{coin}} q_1

q_1 \xrightarrow{\text{push}_\text{cola}} q_2

q_1 \xrightarrow{\text{push}_\text{water}} q_3

q_2 \xrightarrow{\text{push}_\text{cola}} q_2

q_3 \xrightarrow{\text{push}_\text{water}} q_3

v. Gool (UvA & CCNY)  Machines, Models, Monoids, Modal logic  Logic Tutorial, TbiLLC 2017
Automata

Examples

An automaton for the language BUY.

\[ q_0 \xrightarrow{\text{push\_cola}} q_1 \xrightarrow{\text{enter\_coin}} q_2 \xrightarrow{\text{all}} q_3 \xrightarrow{\text{all}} q_2 \]

\[ q_0 \xrightarrow{\text{push\_water}} q_1 \xrightarrow{\text{enter\_coin}} q_2 \xrightarrow{\text{push\_cola}} q_3 \xrightarrow{\text{push\_water}} q_2 \]
Automata

Examples

An automaton for the language

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

An automaton for the language

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

![Diagram of an automaton for the language FACTOR01]

- $q_0$ to $q_1$ with input 0
Automata

Examples
An automaton for the language
\[ \text{FACTOR01} = \{ w \in \{0, 1\}^* \mid w \text{ contains '01' as a factor} \}. \]
Automata

Examples

An automaton for the language
\[ \text{FACTOR01} = \{ w \in \{0, 1\}^* \mid w \text{ contains ‘01’ as a factor} \} . \]
Automata

Examples

An automaton for the language

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

\[ 
\begin{align*}
q_0 & \twoheadrightarrow 0, 1 \\
q_0 & \rightarrow 0 \quad q_1 \\
q_1 & \rightarrow 1 \quad q_2 \\
q_2 & \twoheadrightarrow 0, 1 
\end{align*}
\]
Automata

Examples

An automaton for the language

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

- An automaton is a tuple $A = (Q, \Sigma, \delta)$, where
  - $Q$ is a finite set of states,
  - $\Sigma$ is a finite alphabet,
  - $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is a transition function.
Automata

- An *automaton* is a tuple \( A = (Q, \Sigma, \delta) \), where
  - \( Q \) is a finite set of *states*,
  - \( \Sigma \) is a finite *alphabet*,
  - \( \delta : Q \times \Sigma \rightarrow P(Q) \) is a *transition function*.

- **Notation:** \( q \xrightarrow{a} q' \) means:
  - \( q \) is a state in \( Q \), \( a \) is a letter in \( \Sigma \), and \( q' \) is a state in \( \delta(q, a) \).
Automata

- An *automaton* is a tuple \( A = (Q, \Sigma, \delta) \), where
  - \( Q \) is a finite set of *states*,
  - \( \Sigma \) is a finite *alphabet*,
  - \( \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q) \) is a *transition function*.

- *Notation*: \( q \xrightarrow{a} q' \) means:
  - \( q \) is a state in \( Q \), \( a \) is a letter in \( \Sigma \), and \( q' \) is a state in \( \delta(q, a) \).

- Pick two sets of states, \( I \) and \( F \), in \( Q \).
Automata

- An *automaton* is a tuple $\mathcal{A} = (Q, \Sigma, \delta)$, where
  - $Q$ is a finite set of *states*,
  - $\Sigma$ is a finite *alphabet*,
  - $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is a *transition function*.

- **Notation:** $q \xrightarrow{a} q'$ means:
  - $q$ is a state in $Q$,
  - $a$ is a letter in $\Sigma$,
  - $q'$ is a state in $\delta(q, a)$.

- Pick two sets of states, $I$ and $F$, in $Q$.

- A word $a_1 \ldots a_n \in \Sigma^*$ 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 $q_n \in F$. 
Automata

- An automaton is a tuple $A = (Q, \Sigma, \delta)$, where
  - $Q$ is a finite set of states,
  - $\Sigma$ is a finite alphabet,
  - $\delta: Q \times \Sigma \to \mathcal{P}(Q)$ is a transition function.

- Notation: $q \xrightarrow{a} q'$ means:
  $q$ is a state in $Q$, $a$ is a letter in $\Sigma$, and $q'$ is a state in $\delta(q, a)$.

- Pick two sets of states, $I$ and $F$, in $Q$.

- A word $a_1 \ldots a_n \in \Sigma^*$ is accepted by the automaton $A$ 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 $L_{A,I,F}$ of all accepted words is called the language recognized by $A$ with initial states $I$ and final states $F$. 
Automata

- An automaton is a tuple $\mathcal{A} = (Q, \Sigma, \delta)$, where
  - $Q$ is a finite set of states,
  - $\Sigma$ is a finite alphabet,
  - $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is a transition function.

- Notation: $q \xrightarrow{a} q'$ means:
  $q$ is a state in $Q$, $a$ is a letter in $\Sigma$, and $q'$ is a state in $\delta(q, a)$.

- Pick two sets of states, $I$ and $F$, in $Q$.

- A word $a_1 \ldots a_n \in \Sigma^*$ 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).
Automata

- An *automaton* is a tuple $\mathcal{A} = (Q, \Sigma, \delta)$, where
  - $Q$ is a finite set of *states*,
  - $\Sigma$ is a finite *alphabet*,
  - $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is a *transition function*.

- **Notation:** $q \xrightarrow{a} q'$ means:
  - $q$ is a state in $Q$,
  - $a$ is a letter in $\Sigma$,
  - $q'$ is a state in $\delta(q, a)$.

- Pick two sets of states, $I$ and $F$, in $Q$.

- A word $a_1 \ldots a_n \in \Sigma^*$ 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):** $\delta: Q \times \Sigma \rightarrow Q$. 
Regular languages

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


Examples

- The language \textsc{buy} of action sequences for buying has logic description

\[
\begin{array}{c}
x < y \\
\_ \text{enter coin}(x) \\
\_ \text{push cola}(y) \\
\_ \text{push water}(y)
\end{array}
\]
Logic

Examples

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

\[ \exists x \exists y [x < y \land \text{enter}_{-}\text{coin}(x) \land (\text{push}_{-}\text{cola}(y) \lor \text{push}_{-}\text{water}(y))]. \]
Logic

Examples

- The language **BUY** of action sequences for buying has logic description

\[ \exists x \exists y [x < y \land \text{enter\_coin}(x) \land (\text{push\_cola}(y) \lor \text{push\_water}(y))] \]

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

Examples

- The language BUY of action sequences for buying has logic description

  $$\exists x \exists y [x < y \land \text{enter\_coin}(x) \land (\text{push\_cola}(y) \lor \text{push\_water}(y))]$$.

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

  $$\text{empty} \lor \exists P [P(\text{first}) \land \neg P(\text{last}) \land \forall x \neq \text{last} (P(x) \leftrightarrow \neg P(S(x)))]$$.
Logic

- Syntax:

  - Basic propositional connectives: $\wedge$, $\neg$.
  - Quantification over first-order variables $x, y, \ldots$ and monadic second-order variables $P, Q, \ldots$.
  - Atomic formulas: $x < y$, $P \sqsubset Q$, $P(x)$, and $a(x)$ for $a \in \mathbb{N}$.

Semantics: view a word $w = a_1 \ldots a_n$ as a structure $W$, i.e.,

- The underlying set of $W$ is $\{1, \ldots, n\}$.
- The natural linear order $\prec_W$ interprets the binary predicate $<$.
- For every letter $a \in \mathbb{N}$, $a_W$ is the set of positions $i$ where $a_i = a$.

For a sentence $'L' = \{w \in \mathbb{N} \mid w | = 'L'\}$. Shortcuts such as $S(x)$, 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.
Syntax:

- Basic propositional connectives: \( \land, \neg \).
- Quantification over first-order variables \( x, y, \ldots \) and monadic second-order variables \( P, Q, \ldots \).
- Atomic formulas: \( x < y, P \subseteq Q, P(x), a(x) \) for \( a \in \Sigma \).
Logic

- **Syntax:**
  - Basic propositional connectives: $\land$, $\neg$.
  - Quantification over first-order variables $x, y, \ldots$ and monadic second-order variables $P, Q, \ldots$.
  - Atomic formulas: $x < y$, $P \subseteq Q$, $P(x)$, $a(x)$ for $a \in \Sigma$.

- **Semantics:** view a word $w = a_1 \ldots a_n$ as a *structure* $W$, i.e.,
Logic

- Syntax:
  - Basic propositional connectives: $\land$, $\neg$.
  - Quantification over first-order variables $x, y, \ldots$ and monadic second-order variables $P, Q, \ldots$.
  - Atomic formulas: $x < y$, $P \subseteq Q$, $P(x)$, $a(x)$ for $a \in \Sigma$.

- Semantics: view a word $w = a_1 \ldots a_n$ as a structure $W$, i.e.,
  - The underlying set of $W$ is $\{1, \ldots, n\}$.
  - The natural linear order $<^W$ interprets the binary predicate $<$.
  - For every letter $a \in \Sigma$, $a^W$ is the set of positions $i$ where $a_i = a$. 

Shortcuts such as $S(x)$, 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.
Logic

- **Syntax:**
  - Basic propositional connectives: $\land, \neg$.
  - Quantification over first-order variables $x, y, \ldots$ and monadic second-order variables $P, Q, \ldots$.
  - Atomic formulas: $x < y$, $P \subseteq Q$, $P(x)$, $a(x)$ for $a \in \Sigma$.

- **Semantics:** view a word $w = a_1 \ldots a_n$ as a *structure* $W$, i.e.,
  - The underlying set of $W$ is $\{1, \ldots, n\}$.
  - 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 $\varphi$, $L_\varphi = \{w \in \Sigma^* \mid w \models \varphi\}$. 

Shortcuts such as $S(x)$, 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.
Logic

- **Syntax:**
  - Basic propositional connectives: \(\land, \neg\).
  - Quantification over first-order variables \(x, y, \ldots\) and monadic second-order variables \(P, Q, \ldots\).
  - Atomic formulas: \(x < y, P \subseteq Q, P(x), a(x)\) for \(a \in \Sigma\).

- **Semantics:** view a word \(w = a_1 \ldots a_n\) as a *structure* \(W\), i.e.,
  - The underlying set of \(W\) is \(\{1, \ldots, n\}\).
  - 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 \(\varphi\), \(L_\varphi = \{w \in \Sigma^* \mid w \models \varphi\}\).

- Shortcuts such as \(S(x)\), *first*, *last*, *empty*, ... are definable.
Logic

- **Syntax:**
  - Basic propositional connectives: $\land$, $\neg$.
  - Quantification over first-order variables $x, y, \ldots$ and monadic second-order variables $P, Q, \ldots$.
  - Atomic formulas: $x < y$, $P \subseteq Q$, $P(x)$, $a(x)$ for $a \in \Sigma$.

- **Semantics:** view a word $w = a_1 \ldots a_n$ as a *structure* $W$, i.e.,
  - The underlying set of $W$ is $\{1, \ldots, n\}$.
  - 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 $\varphi$, $L_{\varphi} = \{w \in \Sigma^* \mid w \models \varphi\}$.

- Shortcuts such as $S(x)$, *first*, *last*, *empty*, ... are definable.
- This is Monadic Second Order (MSO) Logic.
Logic

**Syntax:**

- Basic propositional connectives: $\land, \neg$.
- Quantification over first-order variables $x, y, \ldots$ and monadic second-order variables $P, Q, \ldots$.
- Atomic formulas: $x < y, P \subseteq Q, P(x), a(x)$ for $a \in \Sigma$.

**Semantics:** view a word $w = a_1 \ldots a_n$ as a *structure* $W$, i.e.,

- The underlying set of $W$ is $\{1, \ldots, n\}$.
- 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 $\varphi$, $L_\varphi = \{w \in \Sigma^* | w \models \varphi\}$.

- Shortcuts such as $S(x)$, *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?
Exercises

1. Describe the language FACT0R01 with MSO logic, or FO logic if possible.
2. Describe the language PW with MSO logic, or FO logic if possible.
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?
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?
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?
Exercises

1. Describe the language FACT0R01 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.
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.
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?
Problems

- Given an automaton, decide if it accepts any words?
- Given a regular language, decide if it is FO-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 $\text{FO}_k$, 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 $L_1 \not\subseteq M$ and $L_2 \not\supseteq M$. 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?
- 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?
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 \(L_2 \cap M = \emptyset\).
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?
- 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?
- ...

v. Gool  (UvA & CCNY)  Machines, Models, Monoids, Modal logic  Logic Tutorial, TbiLLC 2017
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 $\text{FO}_k$, 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 $\text{FO}_k$-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?
- 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?
- ...

v. Gool (UvA & CCNY)    Machines, Models, Monoids, Modal logic    Logic Tutorial, TbiLLC 2017
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 $\text{FO}_k$, 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 $\text{FO}_k$-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?
  - 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?
- ...

v. Gool (UvA & CCNY)  Machines, Models, Monoids, Modal logic  Logic Tutorial, TbiLLC 2017
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?
- ...

v. Gool (UvA & CCNY)  Machines, Models, Monoids, Modal logic  Logic Tutorial, TbiLLC 2017 23 / 33
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 $k \geq 3$
- ...
How to understand formal languages?

- Boolean algebras with operators
- Model theory
- Monoids
Boolean algebras with operators

- The set of all $\Sigma$-languages, $\mathcal{P}(\Sigma^*)$, is a Boolean algebra with operations $\cup$ (union) and $(\cdot)^c$ (complement).
Boolean algebras with operators

- The set of all \( \Sigma \)-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.
Boolean algebras with operators

- The set of all $\Sigma$-languages, $\mathcal{P}(\Sigma^*)$, is a Boolean algebra with operations $\cup$ (union) and $(\cdot)^c$ (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

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

- The set of all $\Sigma$-languages, $\mathcal{P}(\Sigma^*)$, is a Boolean algebra with operations $\cup$ (union) and $(\cdot)^c$ (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.
Proof of last item.

Suppose that $A$ is an NFA that recognizes $L$ with initial states $I$ and final states $F$. 

Corollary

If $L$ is regular, then the set $\{w_1L, Lw_1|w|\in \Sigma^+\}$ is finite.
Quotient operators shift initial and final states

Proof of last item.

Suppose that $A$ is an NFA that recognizes $L$ with initial states $I$ and final states $F$.

Then $a^{-1}L$ is recognized by $A$ 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 $q_0 \in I$.  

Corollary

If $L$ is regular, then the set $\{w_1L, Lw \mid w \in \Sigma^*\}$ is finite.
Proof of last item.

- Suppose that $A$ is an NFA that recognizes $L$ with initial states $I$ and final states $F$.
- Then $a^{-1}L$ is recognized by $A$ 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 $A$ 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 $q_F \in F$. 

\[ \square \]
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 $\Sigma$, both $a^{-1}L$ and $La^{-1}$ are in $B$. 

---

v. Gool (UvA & CCNY)
Boolean algebras with operators

- A Boolean subalgebra $B \leq \mathcal{P}(\Sigma^*)$ is closed if, for every $L$ in $B$ and $a$ in $\Sigma$, both $a^{-1}L$ and $La^{-1}$ are in $B$.
- The set of regular $\Sigma$-languages, $\text{Reg}(\Sigma^*)$, is a closed subalgebra of $\mathcal{P}(\Sigma^*)$. 

Proposition: A language $L \in \mathcal{P}(\Sigma^*)$ is regular if, and only if, $B(L)$ 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 $\Sigma$, both $a^{-1}L$ and $La^{-1}$ are in $B$.
- The set of regular $\Sigma$-languages, $\text{Reg}(\Sigma^*)$, is a closed subalgebra of $\mathcal{P}(\Sigma^*)$.
- For any automaton $A = (Q, \Sigma, \delta)$, the set of $\Sigma$-languages which $A$ can recognize is a finite closed subalgebra of $\text{Reg}(\Sigma^*)$. 
Boolean algebras with operators

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

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

2. Let \( S \subseteq \mathbb{N} \). Describe the closed subalgebra generated by the \{1\}-language LENGTH\(_S\) of \{1\}-words of length \( S \).

3. (*) When is the algebra in (2) finite?
Model theory

- Let $T_\Sigma$ be the MSO theory of finite $\Sigma$-words, i.e., the set of MSO sentences that are true in all finite $\Sigma$-words.
Model theory

- Let $T_\Sigma$ be the MSO theory of finite $\Sigma$-words, i.e., the set of MSO sentences that are true in all finite $\Sigma$-words.
- Let $\mathcal{L}(T_\Sigma)$ be the Lindenbaum algebra of $T$, i.e., the algebra of MSO-sentences up to $T_\Sigma$-equivalence. This is a Boolean algebra.
Model theory

- Let $T_\Sigma$ be the MSO theory of finite $\Sigma$-words, i.e., the set of MSO sentences that are true in all finite $\Sigma$-words.
- Let $\mathcal{L}(T_\Sigma)$ be the Lindenbaum algebra of $T$, i.e., the algebra of MSO-sentences up to $T_\Sigma$-equivalence. This is a Boolean algebra.
- To any $[\varphi]_{T_\Sigma}$ in $\mathcal{L}(T_\Sigma)$, associate the regular language, $L(\varphi)$, described by $\varphi$. 

Exercise: (*) Describe the operators $L^1$ and $L^2$ directly on the Lindenbaum algebra $\mathcal{L}(T_\Sigma)$. Under this isomorphism, the subalgebra of FO-sentences corresponds to a subalgebra of $\mathcal{L}(\Sigma \star \star)$. Which? See Part II.
Model theory

- Let $T_\Sigma$ be the MSO theory of finite $\Sigma$-words, i.e., the set of MSO sentences that are true in all finite $\Sigma$-words.
- Let $\mathcal{L}(T_\Sigma)$ be the Lindenbaum algebra of $T$, i.e., the algebra of MSO-sentences up to $T_\Sigma$-equivalence. This is a Boolean algebra.
- To any $[\varphi]_{T_\Sigma}$ in $\mathcal{L}(T_\Sigma)$, associate the regular language, $L(\varphi)$, described by $\varphi$.
- This assignment is a well-defined isomorphism between $\mathcal{L}(T_\Sigma)$ and $\text{Reg}(\Sigma^*)$. 

Exercise: (*) Describe the operators $L\neg a_1 L$ and $L\neg La_1$ directly on the Lindenbaum algebra $\mathcal{L}(T_\Sigma)$.

Under this isomorphism, the subalgebra of FO-sentences corresponds to a subalgebra of $\text{Reg}(\Sigma^*)$. Which?

See Part II
Model theory

- Let $T_\Sigma$ be the MSO theory of finite $\Sigma$-words, i.e., the set of MSO sentences that are true in all finite $\Sigma$-words.

- Let $\mathcal{L}(T_\Sigma)$ be the Lindenbaum algebra of $T$, i.e., the algebra of MSO-sentences up to $T_\Sigma$-equivalence. This is a Boolean algebra.

- To any $[\phi]_{T_\Sigma}$ in $\mathcal{L}(T_\Sigma)$, associate the regular language, $L(\phi)$, described by $\phi$.

- This assignment is a well-defined isomorphism between $\mathcal{L}(T_\Sigma)$ and $\text{Reg}(\Sigma^*)$.

- Exercise: (*) Describe the operators $L \mapsto a^{-1}L$ and $L \mapsto La^{-1}$ directly on the Lindenbaum algebra $\mathcal{L}(T_\Sigma)$. 
Let $T_\Sigma$ be the MSO theory of finite $\Sigma$-words, i.e., the set of MSO sentences that are true in all finite $\Sigma$-words.

Let $\mathcal{L}(T_\Sigma)$ be the Lindenbaum algebra of $T$, i.e., the algebra of MSO-sentences up to $T_\Sigma$-equivalence. This is a Boolean algebra.

To any $[\varphi]_{T_\Sigma}$ in $\mathcal{L}(T_\Sigma)$, associate the regular language, $L(\varphi)$, described by $\varphi$.

This assignment is a well-defined isomorphism between $\mathcal{L}(T_\Sigma)$ and $\text{Reg}(\Sigma^*)$.

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 $\text{Reg}(\Sigma^*)$. Which?
Model theory

- Let $T_\Sigma$ be the MSO theory of finite $\Sigma$-words, i.e., the set of MSO sentences that are true in all finite $\Sigma$-words.
- Let $\mathcal{L}(T_\Sigma)$ be the Lindenbaum algebra of $T$, i.e., the algebra of MSO-sentences up to $T_\Sigma$-equivalence. This is a Boolean algebra.
- To any $[\varphi]_{T_\Sigma}$ in $\mathcal{L}(T_\Sigma)$, associate the regular language, $L(\varphi)$, described by $\varphi$.
- This assignment is a well-defined isomorphism between $\mathcal{L}(T_\Sigma)$ and $\text{Reg}(\Sigma^*)$.
- **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 $\text{Reg}(\Sigma^*)$. 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 \( S \).
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 \(S\).
The set $\Sigma^*$, 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 $A = (Q, \Sigma, \cdot)$ gives, for every $a \in \Sigma$, a function $\cdot a$ in $(P(Q) \to P(Q))$, defined by $\cdot a(R) := \{ q \mid q a \in R \}$ for some $q \in Q$. 

v. Gool (UvA & CCNY)  Machines, Models, Monoids, Modal logic  Logic Tutorial, TbiLLC 2017
Monoids

Examples

- The set $\Sigma^*$, with multiplication $u \cdot v := uv$.
- For any set $P$, the set of functions from $P$ to itself, $(P \rightarrow P)$, with multiplication $f \cdot g := f \circ g$. 
Monoids

Examples

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

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

1. Show that $\Sigma^*$ is a monoid.
Exercises

1. Show that $\Sigma^*$ is a monoid.
2. Show that $(P \rightarrow P)$ is a monoid.
Exercises

1. Show that $\Sigma^*$ is a monoid.
2. Show that $(P \rightarrow P)$ is a monoid.
3. Show that $\Sigma^*$ is the free monoid on $\Sigma$, i.e., that for any monoid $M$ and any function $f : \Sigma \rightarrow M$, there is a unique homomorphism $\bar{f} : \Sigma^* \rightarrow M$ extending $f$. 
4. Applying (3) to the function $\Sigma : \Sigma \rightarrow (P \rightarrow P) \rightarrow P(\rightarrow P)$, give an explicit description of the function $\bar{\Sigma} : \Sigma^* \rightarrow (P \rightarrow P) \rightarrow P(\rightarrow P)$.

5. (*) Show that $A$ with initial states $I$ and final states $F$ accepts a word $w \in \Sigma^*$ if, and only if, $I \not\subseteq \bar{\Sigma}w(F) = F$. 

v. Gool (UvA & CCNY)  Machines, Models, Monoids, Modal logic  Logic Tutorial, TbiLLC 2017 32 / 33
Exercises

1. Show that $\sum^*$ is a monoid.
2. Show that $(P \rightarrow P)$ is a monoid.
3. Show that $\sum^*$ is the free monoid on $\sum$, i.e., that for any monoid $M$ and any function $f : \sum \rightarrow M$, there is a unique homomorphism $\overline{f} : \sum^* \rightarrow M$ extending $f$.
4. Applying (3) to the function $\diamond : \sum \rightarrow (\mathcal{P}(Q) \rightarrow \mathcal{P}(Q))$, give an explicit description of the function $\overline{\diamond} : \sum^* \rightarrow (\mathcal{P}(Q) \rightarrow \mathcal{P}(Q))$. 

(*) Show that $A$ with initial states $I$ and final states $F$ accepts a word $w \in \sum^*$ if, and only if, $I \setminus \overline{\diamond}w(F) = \emptyset$. 

Exercises

1. Show that $\Sigma^*$ is a monoid.
2. Show that $(P \rightarrow P)$ is a monoid.
3. Show that $\Sigma^*$ is the free monoid on $\Sigma$, i.e., that for any monoid $M$ and any function $f : \Sigma \rightarrow M$, there is a unique homomorphism $\bar{f} : \Sigma^* \rightarrow M$ extending $f$.
4. Applying (3) to the function $\diamond : \Sigma \rightarrow (\mathcal{P}(Q) \rightarrow \mathcal{P}(Q))$, give an explicit description of the function $\bar{\diamond} : \Sigma^* \rightarrow (\mathcal{P}(Q) \rightarrow \mathcal{P}(Q))$.
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:
  
References for Part I

- **An accessible textbook introduction to the field:**
  

- **A more advanced, but very readable introduction to logic on words:**
  
References for Part I

- An accessible textbook introduction to the field:

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

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

- An accessible textbook introduction to the field:
  

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

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

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

1. Part I: Formal Languages, Automata, and Algebra
2. Part II: Duality and Varieties of Monoids
Recap from Part I

- Formal $\Sigma$-languages are subsets of $\Sigma^*$, the set of finite words over a finite alphabet $\Sigma$. 

Monadic second order logic also describes exactly the regular languages. First order logic describes a (strictly) smaller class of languages. The regular languages form a Boolean algebra with quotient operators. Every regular language $L$ defines a finite closed Boolean subalgebra $\mathcal{B}(L)$. Monoids are also somehow important (but why?)
Recap from Part I

- Formal $\Sigma$-languages are subsets of $\Sigma^*$, the set of finite words over a finite alphabet $\Sigma$.
- Finite-state automata (deterministic or not) describe the regular languages.
- Monadic second order logic also describes exactly the regular languages.
- First order logic describes a (strictly) smaller class of languages.
- The regular languages form a Boolean algebra with quotient operators.
- Every regular language $L$ defines a finite closed Boolean subalgebra $B(L)$.
- Monoids are also somehow important (but why?)
Recap from Part I

- Formal $\Sigma$-languages are subsets of $\Sigma^*$, the set of finite words over a finite alphabet $\Sigma$.
- Finite-state automata (deterministic or not) describe the regular languages.
- Monadic second order logic also describes exactly the regular languages.
Recap from Part I

- Formal $\Sigma$-languages are subsets of $\Sigma^*$, the set of finite words over a finite alphabet $\Sigma$.
- Finite-state automata (deterministic or not) describe the regular languages.
- Monadic second order logic also describes exactly the regular languages.
- First order logic describes a (strictly) smaller class of languages.
Recap from Part 1

- Formal $\Sigma$-languages are subsets of $\Sigma^*$, the set of finite words over a finite alphabet $\Sigma$.
- Finite-state automata (deterministic or not) describe the regular languages.
- Monadic second order logic also describes exactly the regular languages.
- First order logic describes a (strictly) smaller class of languages.
- The regular languages form a Boolean algebra with quotient operators.
Recap from Part I

- Formal $\Sigma$-languages are subsets of $\Sigma^*$, the set of finite words over a finite alphabet $\Sigma$.
- Finite-state automata (deterministic or not) describe the regular languages.
- Monadic second order logic also describes exactly the regular languages.
- First order logic describes a (strictly) smaller class of languages.
- The regular languages form a Boolean algebra with quotient operators.
- Every regular language $L$ defines a finite closed Boolean subalgebra $B(L)$. 
Recap from Part I

- Formal \( \Sigma \)-languages are subsets of \( \Sigma^* \), the set of finite words over a finite alphabet \( \Sigma \).
- Finite-state automata (deterministic or not) describe the regular languages.
- Monadic second order logic also describes exactly the regular languages.
- First order logic describes a (strictly) smaller class of languages.
- The regular languages form a Boolean algebra with quotient operators.
- Every regular language \( L \) defines a finite closed Boolean subalgebra \( B(L) \).
- Monoids are also somehow important.
Recap from Part I

- Formal $\Sigma$-languages are subsets of $\Sigma^*$, the set of finite words over a finite alphabet $\Sigma$.
- Finite-state automata (deterministic or not) describe the regular languages.
- Monadic second order logic also describes exactly the regular languages.
- First order logic describes a (strictly) smaller class of languages.
- The regular languages form a Boolean algebra with quotient operators.
- Every regular language $L$ defines a finite closed Boolean subalgebra $B(L)$.
- Monoids are also somehow important (but why?)
Examples

- The set $\Sigma^*$, with multiplication $u \cdot v := uv$. 
Monoids

Examples

- The set $\Sigma^*$, with multiplication $u \cdot v := uv$.
- For any set $P$, the set of functions from $P$ to itself, $(P \rightarrow P)$, with multiplication $f \cdot g := f \circ g$. 

v. Gool (UvA & CCNY)  
Machines, Models, Monoids, Modal logic  
Logic Tutorial, TbiLLC 2017
Monoids

Examples

- The set $\Sigma^*$, 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 $A = (Q, \Sigma, \delta)$ 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 \( \Sigma^* \) is a monoid.

2. Show that \((P \rightarrow P)\) is a monoid.

3. Show that \( \Sigma^* \) is the free monoid on \( \Sigma \), i.e., that for any monoid \( M \) and any function \( f : \Sigma \rightarrow M \), there is a unique homomorphism \( \bar{f} : \Sigma^* \rightarrow M \) extending \( f \).

4. Applying (3) to the function \( \lozenge : \Sigma \rightarrow (\mathcal{P}(Q) \rightarrow \mathcal{P}(Q)) \), give an explicit description of the function \( \bar{\lozenge} : \Sigma^* \rightarrow (\mathcal{P}(Q) \rightarrow \mathcal{P}(Q)) \).

5. (*) Show that \( A \) with initial states \( I \) and final states \( F \) accepts a word \( w \in \Sigma^* \) if, and only if, \( I \cap \bar{\lozenge}_w(F) \neq \emptyset \).
Proposition

A $\Sigma$-language $L$ is regular if, and only if, there exists a homomorphism $\eta: \Sigma^* \rightarrow M$, with $M$ a finite monoid, such that $L = \eta^{-1}(R)$ for some $R \subseteq M$. 

Proof ingredients.

The exercises on the previous slide show how to build a monoid homomorphism from an NFA. For the converse, notice that a homomorphism from $\Sigma^*$ to a monoid is a (deterministic) automaton.
Proposition

A $\Sigma$-language $L$ is regular if, and only if, there exists a homomorphism $\eta: \Sigma^* \rightarrow M$, with $M$ a finite monoid, such that $L = \eta^{-1}(R)$ for some $R \subseteq M$.

Proof ingredients.

- The exercises on the previous slide show how to build a monoid homomorphism from an NFA.
- For the converse, notice that a homomorphism from $\Sigma^*$ to a monoid ‘is’ a (deterministic) automaton.
Regular languages are:
Regular languages are:

- the languages recognized by finite non-deterministic automata.
Regular languages are:

- the languages recognized by finite non-deterministic automata.
- the languages recognized by finite deterministic automata.
Regular languages are:

- the languages recognized by finite non-deterministic automata.
- the languages recognized by finite deterministic automata.
- the languages definable in monadic second order logic.
Regular languages are:

- the languages recognized by finite non-deterministic automata.
- the languages recognized by finite deterministic automata.
- the languages definable in monadic second order logic.
- the inverse images of homomorphisms from the free monoid to a finite monoid.
Regular languages are:

- the languages recognized by finite non-deterministic automata.
- the languages recognized by finite deterministic automata.
- the languages definable in monadic second order logic.
- the inverse images of homomorphisms from the free monoid to a finite monoid.
- the unions of classes under finite index congruences on a free monoid.
Regular languages are:
- the languages recognized by finite non-deterministic automata.
- the languages recognized by finite deterministic automata.
- the languages definable in monadic second order logic.
- the inverse images of homomorphisms from the free monoid to a finite monoid.
- the unions of classes under finite index congruences on a free monoid.

Today, we will see how these characterizations are connected to each other through **Stone duality**.
Outline Part II

1 Finite Duality and Regular Languages
   - Boolean algebras
   - Finite Stone duality
   - Duality for regular languages

2 Full Duality and Varieties
   - First-order logic and aperiodic monoids
   - Full Stone duality
Stone duality

“In January last year I gave a course at the Indian Winter School in Logic and went on an excursion to Varanasi and Sarnath, the birthplace of Buddhism. Upon entering the amazing Archaeological Museum at Sarnath, our guide opened with: ‘Duality underlies the world.’ This is the kind of sweeping statement that every mathematician, at least secretly, would like to believe about their particular focus…”

M. Gehrke. *Duality*. Oratie (inaugural lecture) at Radboud University Nijmegen, 2009. URL: http://repository.ubn.ru.nl/bitstream/handle/2066/83300/83300.pdf
Stone duality

- Stone duality was introduced by mathematician M. H. Stone in the 1930’s.
Stone duality

- Stone duality was introduced by mathematician M. H. Stone in the 1930’s.
- In logic, it underpins the connection between syntax and semantics.
Stone duality

- Stone duality was introduced by mathematician M. H. Stone in the 1930's.
- In logic, it underpins the connection between syntax and semantics.
- The **dual** of a collection of formulas (syntax) is a space of possible worlds/states (semantics) interpreting the formulas, and vice versa.
Stone duality

- Stone duality was introduced by mathematician M. H. Stone in the 1930s.
- In logic, it underpins the connection between syntax and semantics.
- The dual of a collection of formulas (syntax) is a space of possible worlds/states (semantics) interpreting the formulas, and vice versa.
- A key idea, and the meaning of the term ‘duality’ (＝ dual categorical equivalence), is that the direction of morphisms is reversed.
Stone duality

- Stone duality was introduced by mathematician M. H. Stone in the 1930’s.
- In logic, it underpins the connection between syntax and semantics.
- The **dual** of a collection of formulas (syntax) is a space of possible worlds/states (semantics) interpreting the formulas, and vice versa.
- A key idea, and the meaning of the term ‘duality’ (≡ dual categorical equivalence), is that the direction of morphisms is reversed.
- **More information = Less** possible worlds.
Stone duality

- Stone duality was introduced by mathematician M. H. Stone in the 1930's.
- In logic, it underpins the connection between syntax and semantics.
- The dual of a collection of formulas (syntax) is a space of possible worlds/states (semantics) interpreting the formulas, and vice versa.
- A key idea, and the meaning of the term ‘duality’ (≈ dual categorical equivalence), is that the direction of morphisms is reversed.
- More information = Less possible worlds.
- More possible worlds = Less information.
Stone duality

- Stone duality was introduced by mathematician M. H. Stone in the 1930’s.
- In logic, it underpins the connection between syntax and semantics.
- The dual of a collection of formulas (syntax) is a space of possible worlds/states (semantics) interpreting the formulas, and vice versa.
- A key idea, and the meaning of the term ‘duality’ (= dual categorical equivalence), is that the direction of morphisms is reversed.
- More information = Less possible worlds.
- More possible worlds = Less information.
- Formulating duality theory precisely requires some algebra, and, for the non-finite case, topology.
Stone duality

- Stone duality was introduced by mathematician M. H. Stone in the 1930’s.
- In logic, it underpins the connection between syntax and semantics.
- The dual of a collection of formulas (syntax) is a space of possible worlds/states (semantics) interpreting the formulas, and vice versa.
- A key idea, and the meaning of the term ‘duality’ (= dual categorical equivalence), is that the direction of morphisms is reversed.
- More information = Less possible worlds.
- More possible worlds = Less information.
- Formulating duality theory precisely requires some algebra, and, for the non-finite case, topology.
- We will focus on the applications to regular languages.
1 Finite Duality and Regular Languages
- Boolean algebras
- Finite Stone duality
- Duality for regular languages
An (abstract) **Boolean algebra** is a tuple \((B, \lor, \neg, \bot)\), where

- \(B\) is a set,
- \(\lor\) is a binary operation,
- \(\neg\) is a unary operation,
- \(\bot\) is an element of \(B\),
- for any classical tautology \(\varphi(\bar{x}) \leftrightarrow \psi(\bar{x})\) and \(\bar{b}\) in \(B\), \(\varphi(\bar{b}) = \psi(\bar{b})\) in \(B\).
**Boolean algebras**

- An (abstract) **Boolean algebra** is a tuple \((B, \lor, \lnot, \bot)\), where
  - \(B\) is a set,
  - \(\lor\) is a binary operation,
  - \(\lnot\) is a unary operation,
  - \(\bot\) is an element of \(B\),
  - for any classical tautology \(\varphi(\bar{x}) \iff \psi(\bar{x})\) and \(\bar{b}\) in \(B\), \(\varphi(\bar{b}) = \psi(\bar{b})\) in \(B\).
- For example, \(a \lor b = b \lor a\), \(\lnot \lnot a = a\), \(a \lor \bot = a\), \ldots.
Boolean algebras

- An (abstract) **Boolean algebra** is a tuple \((B, \vee, \neg, \bot)\), where
  - \(B\) is a set,
  - \(\vee\) is a binary operation,
  - \(\neg\) is a unary operation,
  - \(\bot\) is an element of \(B\),
  - for any classical tautology \(\varphi(\overline{x}) \leftrightarrow \psi(\overline{x})\) and \(\overline{b}\) in \(B\), \(\varphi(\overline{b}) = \psi(\overline{b})\) in \(B\).

- For example, \(a \vee b = b \vee a\), \(\neg\neg a = a\), \(a \vee \bot = a\), \ldots.

- The last condition can be replaced by a finite list of axioms.
Boolean algebras

- An (abstract) *Boolean algebra* is a tuple \((B, \vee, \neg, \bot)\), where
  - \(B\) is a set,
  - \(\vee\) is a binary operation,
  - \(\neg\) is a unary operation,
  - \(\bot\) is an element of \(B\),
  - for any classical tautology \(\varphi(\bar{x}) \iff \psi(\bar{x})\) and \(\bar{b}\) in \(B\), \(\varphi(\bar{b}) = \psi(\bar{b})\) in \(B\).

- For example, \(a \vee b = b \vee a\), \(\neg\neg a = a\), \(a \vee \bot = a\), \ldots.

- The last condition can be replaced by a finite list of axioms.

- Boolean algebras are partially ordered: \(a \leq b\) iff \(a \vee b = b\).
Examples

- For any set $X$, $(\mathcal{P}(X), \cup, (\cdot)^c, \emptyset)$ is a Boolean algebra.
Boolean algebras: examples

Examples

- For any set $X$, $(\mathcal{P}(X), \cup, (\cdot)^c, \emptyset)$ is a Boolean algebra.
- The *Lindenbaum algebra* of classical propositional logic on a set of variables $V$ is the *free* Boolean algebra on $V$. 
Boolean algebras: examples

Examples

- For any set $X$, $(\mathcal{P}(X), \cup, (\cdot)^c, \emptyset)$ is a Boolean algebra.
- The *Lindenbaum algebra* of classical propositional logic on a set of variables $V$ is the *free* Boolean algebra on $V$.
- For any topological space $X$, the *clopen* (= closed and open) subsets are a Boolean subalgebra of $\mathcal{P}(X)$.
Finite Stone duality: algebras

Proposition

Every finite Boolean algebra $B$ is isomorphic to a Boolean algebra of the form $\mathcal{P}(X)$, for some set $X$. 

Proof.

Take $X = \text{At}(B)$, the set of atoms of $B$.

Identify $b \in B$ with the set, $\hat{b}$, of atoms below it.

Example

If $V = \{ p_1 \hookrightarrow \ldots \hookrightarrow p_n \}$, then the Lindenbaum algebra of classical propositional logic on $V$ is isomorphic to $\mathcal{P}(X)$, where $X = \{ 0 \hookrightarrow 1 \}$. 

In words: a formula of CPL can be identified with the set of valuations in which it is true. 

When $V$ is infinite, the situation is more subtle!
Finite Stone duality: algebras

**Proposition**

Every **finite** Boolean algebra $B$ is isomorphic to a Boolean algebra of the form $\mathcal{P}(X)$, for some set $X$.

**Proof.**

Take $X = \text{At}(B)$, the set of atoms of $B$. 

Example

If $V = \{p_1 \hookrightarrow \ldots \hookrightarrow p_n\}$, then the Lindenbaum algebra of classical propositional logic on $V$ is isomorphic to $\mathcal{P}(X)$, where $X = \{0 \hookrightarrow 1\}^V$. In words: a formula of CPL can be identified with the set of valuations in which it is true. When $V$ is infinite, the situation is more subtle!
Finite Stone duality: algebras

**Proposition**

Every finite Boolean algebra $B$ is isomorphic to a Boolean algebra of the form $\mathcal{P}(X)$, for some set $X$.

**Proof.**

Take $X = \text{At}(B)$, the set of atoms of $B$. Identify $b \in B$ with the set, $\hat{b}$, of atoms below it.
Finite Stone duality: algebras

Proposition

Every finite Boolean algebra $B$ is isomorphic to a Boolean algebra of the form $\mathcal{P}(X)$, for some set $X$.

Proof.

Take $X = \text{At}(B)$, the set of atoms of $B$.
Identify $b \in B$ with the set, $\hat{b}$, of atoms below it.

Example

If $V = \{p_1, \ldots, p_n\}$, then the Lindenbaum algebra of classical propositional logic on $V$ is isomorphic to $\mathcal{P}(X)$, where $X = \{0, 1\}^V$.
Finite Stone duality: algebras

**Proposition**

Every finite Boolean algebra $B$ is isomorphic to a Boolean algebra of the form $\mathcal{P}(X)$, for some set $X$.

**Proof.**

Take $X = \text{At}(B)$, the set of atoms of $B$. Identify $b \in B$ with the set, $\hat{b}$, of atoms below it.

**Example**

If $V = \{p_1, \ldots, p_n\}$, then the Lindenbaum algebra of classical propositional logic on $V$ is isomorphic to $\mathcal{P}(X)$, where $X = \{0, 1\}^V$. In words: a formula of CPL can be identified with the set of valuations in which it is true.
Finite Stone duality: algebras

**Proposition**

Every finite Boolean algebra $B$ is isomorphic to a Boolean algebra of the form $\mathcal{P}(X)$, for some set $X$.

**Proof.**

Take $X = \text{At}(B)$, the set of atoms of $B$. Identify $b \in B$ with the set, $\hat{b}$, of atoms below it.

**Example**

If $V = \{p_1, \ldots, p_n\}$, then the Lindenbaum algebra of classical propositional logic on $V$ is isomorphic to $\mathcal{P}(X)$, where $X = \{0, 1\}^V$. In words: a formula of CPL can be identified with the set of valuations in which it is true. When $V$ is infinite, the situation is more subtle!
Proposition

Every homomorphism between finite Boolean algebras $\mathcal{P}(Y) \rightarrow \mathcal{P}(X)$ is of the form $f^{-1}$ for some function $f : X \rightarrow Y$. 

In particular, any finite subalgebra of $\mathcal{P}(X)$ is of the form $\mathcal{P}(Y)$, where $q : X \rightarrow Y$ is a quotient of $X$. In other words, any finite subalgebra of $\mathcal{P}(X)$ is the collection of finite unions of equivalence classes of an equivalence relation on $X$. 

Finite Stone duality: homomorphisms
Finite Stone duality: homomorphisms

**Proposition**

*Every homomorphism between finite Boolean algebras* $\mathcal{P}(Y) \to \mathcal{P}(X)$ *is of the form* $f^{-1}$ *for some function* $f : X \to Y$.

- In particular, any finite subalgebra of $\mathcal{P}(X)$ has the form $q^{-1} : \mathcal{P}(Y) \hookrightarrow \mathcal{P}(X)$, where $q : X \twoheadrightarrow Y$ is a quotient of $X$.
- In other words, any finite subalgebra of $\mathcal{P}(X)$ is the collection of finite unions of equivalence classes of an equivalence relation on $X$. 
Subalgebras and equivalence relations

Example

- The closed subalgebra generated by the $\Sigma$-language $L = \text{EVENLENGTH}$ is

$$B(L) = \{\emptyset, L, L^c, \Sigma^*\} \hookrightarrow \text{Reg}(\Sigma^*).$$
Subalgebras and equivalence relations

Example

- The closed subalgebra generated by the \( \Sigma \)-language \( L = \text{EVENLENGTH} \) is

\[
B(L) = \{ \emptyset, L, L^c, \Sigma^* \} \hookrightarrow \text{Reg}(\Sigma^*). \]

- The dual of this subalgebra is a quotient \( q: \Sigma^* \rightarrow \text{At } B(L) \).

- This quotient is given by the equivalence relation \( w_1 \equiv_L w_2 \) if, and only if, the length of \( w_1 \) and \( w_2 \) have the same parity.
Finite Stone duality: regular languages

Let $L$ be a regular $\Sigma$-language.
Finite Stone duality: regular languages

- Let $L$ be a regular $\Sigma$-language.
- Let $B(L)$ be the finite closed subalgebra of $\text{Reg}(\Sigma^*)$ generated by $L$. 
Finite Stone duality: regular languages

- Let $L$ be a regular $\Sigma$-language.
- Let $B(L)$ be the finite closed subalgebra of $\text{Reg}(\Sigma^*)$ generated by $L$.
- Then $B(L)$ is the set of unions of equivalence classes under an equivalence relation $\equiv_L$ on $\Sigma^*$. 
Finite Stone duality: regular languages

- Let $L$ be a regular $\Sigma$-language.
- Let $B(L)$ be the finite closed subalgebra of $\text{Reg}(\Sigma^*)$ generated by $L$.
- Then $B(L)$ is the set of unions of equivalence classes under an equivalence relation $\equiv_L$ on $\Sigma^*$, which can be defined by

$$w_1 \equiv_L w_2 \iff \text{ for all } u, v \in \Sigma^*, uw_1v \in L \text{ iff } uw_2v \in L.$$
Finite Stone duality: regular languages

- Let $L$ be a regular $\Sigma$-language.
- Let $B(L)$ be the finite closed subalgebra of $\text{Reg}(\Sigma^*)$ generated by $L$.
- Then $B(L)$ is the set of unions of equivalence classes under an equivalence relation $\equiv_L$ on $\Sigma^*$, which can be defined by

\[ w_1 \equiv_L w_2 \iff \text{for all } u, v \in \Sigma^*, uw_1 v \in L \iff uw_2 v \in L. \]

- A language $L \subseteq \Sigma^*$ is regular if, and only if, $\equiv_L$ has finite index.
Duality and regular languages

- $B(L)$ is a closed subalgebra of $\text{Reg}(\Sigma^*)$. 
Duality and regular languages

- \( B(L) \) is a \textit{closed} subalgebra of \( \text{Reg}(\Sigma^*) \).

- It follows that the dual \( M(L) = \Sigma^*/\equiv_L \) of \( B(L) \) is a \textbf{monoid}.
Duality and regular languages

- $B(L)$ is a *closed* subalgebra of $\text{Reg}(\Sigma^*)$.
- It follows that the dual $M(L) = \Sigma^*/\equiv_L$ of $B(L)$ is a *monoid*.
- The monoid $M(L)$ is the *syntactic monoid* of $L$. 
Duality and regular languages

- $B(L)$ is a *closed* subalgebra of $\text{Reg}(\Sigma^*)$.
- It follows that the dual $M(L) = \Sigma^*/\equiv_L$ of $B(L)$ is a *monoid*.
- The monoid $M(L)$ is the *syntactic monoid* of $L$.
- The *homomorphism* $q : \Sigma^* \to M(L)$ recognizes $L$:
  
  $L = q^{-1}(R)$ where $R = q(L)$. 
Duality and regular languages

- $B(L)$ is a *closed* subalgebra of $\text{Reg}(\Sigma^*)$.
- It follows that the dual $M(L) = \Sigma^*/\equiv_L$ of $B(L)$ is a *monoid*.
- The monoid $M(L)$ is the *syntactic monoid* of $L$.
- The homomorphism $q: \Sigma^* \rightarrow M(L)$ recognizes $L$: $L = q^{-1}(R)$ where $R = q(L)$.
- Moreover, $M(L)$ is the *minimum* such monoid quotient of $\Sigma^*$: if $q': \Sigma^* \rightarrow M'$ recognizes $L$, then there exists $f: M' \rightarrow M(L)$ such that $fq' = q$. 
Example

Let $\Sigma = \{0, 1\}$ and $L = \text{EVENLENGTH}$.
Let $\Sigma = \{0, 1\}$ and $L = \text{EVENLENGTH}$. For $w_1, w_2 \in \Sigma^*$, $w_1 \equiv_L w_2$ iff the length of $w_1$ and of $w_2$ have the same parity. Therefore, $M(L) = \mathbb{Z}_2$, the two-element group. The quotient $q: \Sigma^* \to M(L)$ is defined by $q(w) := \text{parity of the length of } w$. Notice that $q(w_1w_2) = q(w_1) \cdot q(w_2)$, i.e., $q$ is a homomorphism.
Syntactic monoid: Example

Example

Let $\Sigma = \{0, 1\}$ and $L = \text{EVENLENGTH}$. For $w_1, w_2 \in \Sigma^*$, $w_1 \equiv_L w_2$ iff the length of $w_1$ and of $w_2$ have the same parity.
Example

Let $\Sigma = \{0, 1\}$ and $L = \text{EVENLENGTH}$. For $w_1, w_2 \in \Sigma^*$, $w_1 \equiv_L w_2$ iff the length of $w_1$ and of $w_2$ have the same parity. Therefore, $M(L) \cong \mathbb{Z}_2$, the two-element group.
Let $\Sigma = \{0, 1\}$ and $L = \text{EVENLENGTH}$.
For $w_1, w_2 \in \Sigma^*$, $w_1 \equiv_L w_2$ iff the length of $w_1$ and of $w_2$ have the same parity.
Therefore, $M(L) \cong \mathbb{Z}_2$, the two-element group.
The quotient $q: \Sigma^* \rightarrow M(L)$ is defined by
Example

Let $\Sigma = \{0, 1\}$ and $L = \text{EVENLENGTH}$. For $w_1, w_2 \in \Sigma^*$, $w_1 \equiv_L w_2$ iff the length of $w_1$ and of $w_2$ have the same parity. Therefore, $M(L) \cong \mathbb{Z}_2$, the two-element group. The quotient $q : \Sigma^* \rightarrow M(L)$ is defined by $q(w) := \text{parity of the length of } w$. 
Syntactic monoid: Example

Example

Let $\Sigma = \{0, 1\}$ and $L = \text{EVENLENGTH}$. For $w_1, w_2 \in \Sigma^*$, $w_1 \equiv_L w_2$ iff the length of $w_1$ and of $w_2$ have the same parity.

Therefore, $M(L) \cong \mathbb{Z}_2$, the two-element group.

The quotient $q: \Sigma^* \to M(L)$ is defined by $q(w) := \text{parity of the length of } w$.

Notice that $q(w_1w_2) = q(w_1) \oplus q(w_2)$, i.e., $q$ is a homomorphism.
Exercises

1. Find the syntactic monoid quotient $\Sigma^* \to M(L)$ when $L = \text{EVENONES}$.  
2. Find the syntactic monoid quotient $\Sigma^* \to M(L)$ when $L = \text{BUY}$.  
3. (*) Find the syntactic monoid quotient $\Sigma^* \to M(L)$ when $L = \text{PW}$.  

Conclude from the solutions to (1) – (3) what the closed subalgebras, $B(L)$, generated by $L$ are.

5. Use $\text{\$L}$ to show that $L$ is not regular when $L = \text{NON1}$.  

v. Gool (UvA & CCNY)  
Machines, Models, Monoids, Modal logic  
Logic Tutorial, TbiLLC 2017  
20 / 32
Exercises

1. Find the syntactic monoid quotient $\Sigma^* \rightarrow M(L)$ when $L = \text{EVENONES}$.
2. Find the syntactic monoid quotient $\Sigma^* \rightarrow M(L)$ when $L = \text{BUY}$.
3. (*) Find the syntactic monoid quotient $\Sigma^* \rightarrow M(L)$ when $L = \text{PW}$.
4. Conclude from the solutions to (1) – (3) what the closed subalgebras, $B(L)$, generated by $L$ are.

Use $\mathcal{L}$ to show that $L$ is not regular when $L = \text{NON1}$.
Exercises

1. Find the syntactic monoid quotient $\Sigma^* \to M(L)$ when $L = \text{EVENONES}$.

2. Find the syntactic monoid quotient $\Sigma^* \to M(L)$ when $L = \text{BUY}$.

3. (*) Find the syntactic monoid quotient $\Sigma^* \to M(L)$ when $L = \text{PW}$.

4. Conclude from the solutions to (1) – (3) what the closed subalgebras, $B(L)$, generated by $L$ are.

5. Use $\equiv_L$ to show that $L$ is not regular when $L = \text{NON1}$. 

2 Full Duality and Varieties

- First-order logic and aperiodic monoids
- Full Stone duality
In Part I, we asked: what is the subalgebra $\text{FO}(\Sigma^*)$ of $\text{Reg}(\Sigma^*)$?
FO and aperiodics

- In Part I, we asked: what is the subalgebra $\text{FO}(\Sigma^*)$ of $\text{Reg}(\Sigma^*)$?
- We now know that any regular language $L$ has a finite syntactic monoid $M(L)$. 

A monoid $M$ is aperiodic if it contains no non-trivial subgroups. For finite monoids, it is equivalent to say: the equation $x_n = x_{n+1}$ holds in $M$ for some $n$. It is also equivalent to say: $x! = x! x$, where $x!$ is the idempotent power of $x$. 

Theorem (Schützenberger, 1960s) A language $L$ is first-order definable if, and only if, the syntactic monoid $M(L)$ is finite and aperiodic. 

An algorithm for deciding if a regular language is FO-definable.
FO and aperiodics

- In Part I, we asked: what is the subalgebra $\text{FO}(\Sigma^*)$ of $\text{Reg}(\Sigma^*)$?
- We now know that any regular language $L$ has a finite syntactic monoid $M(L)$.
- A monoid $M$ is aperiodic if it contains no non-trivial subgroups.
FO and aperiodics

- In Part I, we asked: what is the subalgebra $\text{FO}(\Sigma^*)$ of $\text{Reg}(\Sigma^*)$?
- We now know that any regular language $L$ has a finite syntactic monoid $M(L)$.
- A monoid $M$ is aperiodic if it contains no non-trivial subgroups.
- For finite monoids, it is equivalent to say: the equation $x^n = x^{n+1}$ holds in $M$ for some $n$. 
FO and aperiodics

- In Part I, we asked: what is the subalgebra \( \text{FO}(\Sigma^*) \) of \( \text{Reg}(\Sigma^*) \)?
- We now know that any regular language \( L \) has a finite syntactic monoid \( M(L) \).
- A monoid \( M \) is **aperiodic** if it contains no non-trivial subgroups.
- For finite monoids, it is equivalent to say:
  the equation \( x^n = x^{n+1} \) holds in \( M \) for some \( n \).
- It is also equivalent to say: \( x^\omega = x^\omega x \),
  where \( x^\omega \) is the **idempotent power** of \( x \).
In Part I, we asked: what is the subalgebra FO(Σ*) of Reg(Σ*)?

We now know that any regular language L has a finite syntactic monoid M(L).

A monoid M is aperiodic if it contains no non-trivial subgroups.

For finite monoids, it is equivalent to say:
the equation \( x^n = x^{n+1} \) holds in M for some \( n \).

It is also equivalent to say: \( x^\omega = x^\omega x \),
where \( x^\omega \) is the idempotent power of \( x \).

**Theorem (Schützenberger, 1960s)**

A language \( L \) is first-order definable if, and only if, the syntactic monoid M(L) is finite and aperiodic.
FO and aperiodics

- In Part I, we asked: what is the subalgebra FO($\Sigma^*$) of $\text{Reg}(\Sigma^*)$?
- We now know that any regular language $L$ has a finite syntactic monoid $M(L)$.
- A monoid $M$ is aperiodic if it contains no non-trivial subgroups.
- For finite monoids, it is equivalent to say: the equation $x^n = x^{n+1}$ holds in $M$ for some $n$.
- It is also equivalent to say: $x^\omega = x^\omega x$, where $x^\omega$ is the idempotent power of $x$.

Theorem (Schützenberger, 1960s)

A language $L$ is first-order definable if, and only if, the syntactic monoid $M(L)$ is finite and aperiodic.

An algorithm for deciding if a regular language is FO-definable.
Example of Schützenberger’s Theorem

Example

- The syntactic monoid of \textit{EVENLENGTH} is $\mathbb{Z}_2$. 
Example of Schützenberger’s Theorem

Example

- The syntactic monoid of \textsc{EVENLENGTH} is $\mathbb{Z}_2$.
- This contains (in fact, is) a group.
Example of Schützenberger’s Theorem

Example

- The syntactic monoid of EVENLENGTH is $\mathbb{Z}_2$.
- This contains (in fact, is) a group.
- By Schützenberger’s theorem, EVENLENGTH is not first order definable.
Exercise

- Using the results from the previous exercise, determine which of the syntactic monoids for EVENONES, BUY, and PW are aperiodic.
- Conclude which of these languages are first order definable.
Varieties of monoids and languages

- A class of finite monoids $\mathbf{V}$ is a (pseudo)\textit{variety} if it is closed under homomorphic images (H), submonoids (S) and finite products ($P^\text{fin}$).
Varieties of monoids and languages

- A class of finite monoids $\mathbf{V}$ is a (pseudo)variety if it is closed under homomorphic images ($H$), submonoids ($S$) and finite products ($P_{\text{fin}}$).
- For a variety of monoids $\mathbf{V}$, define $\mathcal{V}(\Sigma^*)$ to be the class of $\Sigma$-languages $L$ such that $M(L) \in \mathbf{V}$.
Varieties of monoids and languages

- A class of finite monoids \( \mathbf{V} \) is a (pseudo)\textit{variety} if it is closed under homomorphic images (\( H \)), submonoids (\( S \)) and finite products (\( P_{\text{fin}} \)).

- For a variety of monoids \( \mathbf{V} \), define \( \mathbf{V}(\Sigma^*) \) to be the class of \( \Sigma \)-languages \( L \) such that \( M(L) \in \mathbf{V} \).

- Then \( \{ \mathbf{V}(\Sigma^*) \}_{\Sigma} \) is a \textit{variety of regular languages}: a collection of Boolean subalgebras of \( \text{Reg}(\Sigma^*) \) which is closed under inverse images of homomorphisms \( \Sigma_1^* \rightarrow \Sigma_2^* \).
Varieties of monoids and languages

A class of finite monoids $\mathbf{V}$ is a (pseudo)variety if it is closed under homomorphic images ($\mathbf{H}$), submonoids ($\mathbf{S}$) and finite products ($\mathbf{P}^{\text{fin}}$).

For a variety of monoids $\mathbf{V}$, define $\mathbf{V}(\Sigma^*)$ to be the class of $\Sigma$-languages $L$ such that $M(L) \in \mathbf{V}$.

Then $\{\mathbf{V}(\Sigma^*)\}_\Sigma$ is a variety of regular languages: a collection of Boolean subalgebras of $\mathbf{Reg}(\Sigma^*)$ which is closed under inverse images of homomorphisms $\Sigma^*_1 \rightarrow \Sigma^*_2$.

Theorem (Eilenberg)

The map $\mathbf{V} \leftrightarrow \mathbf{V}$ is an order-bijection between varieties of finite monoids and varieties of regular languages.
Equations?

- Birkhoff’s theorem: varieties of (arbitrary) algebras can be defined by (finite) equations.
Birkhoff’s theorem: varieties of (arbitrary) algebras can be defined by (finite) equations.

What about (pseudo)varieties of finite algebras?
Equations?

- Birkhoff’s theorem: varieties of (arbitrary) algebras can be defined by (finite) equations.
- What about (pseudo)varieties of finite algebras?
- We need profinite equations.
Equations?

- Birkhoff’s theorem: varieties of (arbitrary) algebras can be defined by (finite) equations.
- What about (pseudo)varieties of \textit{finite} algebras?
- We need \textit{profinite} equations.
- To explain what these are, and why we need them: \textbf{full} Stone duality.
## Stone duality: general case

**Proposition**

*Every Boolean algebra $B$ can be embedded into a Boolean algebra of the form $\mathcal{P}(X)$.*
Stone duality: general case

Proposition

Every Boolean algebra $B$ can be embedded into a Boolean algebra of the form $\mathcal{P}(X)$, and there is a unique such embedding for which the topology generated by the sets in the image of $B$ is compact and Hausdorff (and zero-dimensional).
Stone duality: general case

Proposition

Every Boolean algebra $B$ can be embedded into a Boolean algebra of the form $\mathcal{P}(X)$, and there is a unique such embedding for which the topology generated by the sets in the image of $B$ is compact and Hausdorff (and zero-dimensional).

‘Construction’ of the embedding.

Take $X$ to be the set of ultrafilters of $B$. 
Stone duality: general case

Proposition

Every Boolean algebra $B$ can be embedded into a Boolean algebra of the form $\mathcal{P}(X)$, and there is a unique such embedding for which the topology generated by the sets in the image of $B$ is compact and Hausdorff (and zero-dimensional).

‘Construction’ of the embedding.

Take $X$ to be the set of ultrafilters of $B$. Identify $b \in B$ with the set, $\hat{b}$, of ultrafilters containing it.
Stone duality: general case

Proposition

Every Boolean algebra $B$ can be embedded into a Boolean algebra of the form $\mathcal{P}(X)$, and there is a unique such embedding for which the topology generated by the sets in the image of $B$ is compact and Hausdorff (and zero-dimensional).

‘Construction’ of the embedding.

Take $X$ to be the set of ultrafilters of $B$. Identify $b \in B$ with the set, $\hat{b}$, of ultrafilters containing it.

- A Boolean space is a compact Hausdorff zero-dimensional space.
Stone duality: general case

Proposition

Every Boolean algebra $B$ can be embedded into a Boolean algebra of the form $\mathcal{P}(X)$, and there is a unique such embedding for which the topology generated by the sets in the image of $B$ is compact and Hausdorff (and zero-dimensional).

‘Construction’ of the embedding.

Take $X$ to be the set of ultrafilters of $B$.
Identify $b \in B$ with the set, $\hat{b}$, of ultrafilters containing it.

- A Boolean space is a compact Hausdorff zero-dimensional space.
- Equivalently, a Boolean space is a profinite object in the category of topological spaces.
Example

The dual space of the Lindenbaum algebra of CPL on a countable set $V = \{p_1, p_2, p_3, \ldots \}$ is the Cantor space $\{0, 1\}^V$. 
Exercises

1. What is the dual space of the Boolean algebra of finite subsets of the natural numbers and their complements?

2. Use what you know about classical propositional logic to prove that the Lindenbaum algebra of CPL on a countable set $V = \{p_1, p_2, p_3, \ldots \}$ can be embedded into $\mathcal{P}(\{0, 1\}^V)$.

3. (*) Show that the topology generated by the image of the embedding in (2) is compact and Hausdorff.

4. (*) Show that the topology generated by the image of the embedding in (2) coincides with the topology of the Cantor space.
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a \textit{continuous} function between the dual spaces.
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a *continuous* function between the dual spaces.
- The *categories* of Boolean algebras and Boolean spaces are dually equivalent.
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a continuous function between the dual spaces.
- The categories of Boolean algebras and Boolean spaces are dually equivalent.

<table>
<thead>
<tr>
<th>Algebras</th>
<th>dual to</th>
<th>Spaces</th>
</tr>
</thead>
</table>

Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a *continuous* function between the dual spaces.

- The *categories* of Boolean algebras and Boolean spaces are dually equivalent.

<table>
<thead>
<tr>
<th>Algebras</th>
<th>dual to</th>
<th>Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>subalgebras</td>
<td>$\leftrightarrow$</td>
<td>quotient objects</td>
</tr>
</tbody>
</table>
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a \textit{continuous} function between the dual spaces.
- The \textit{categories} of Boolean algebras and Boolean spaces are dually equivalent.

<table>
<thead>
<tr>
<th>Algebras</th>
<th>dual to</th>
<th>Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>subalgebras</td>
<td>$\leftrightarrow$</td>
<td>quotient objects</td>
</tr>
<tr>
<td>quotient algebras</td>
<td>$\leftrightarrow$</td>
<td>subobjects</td>
</tr>
</tbody>
</table>
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a *continuous* function between the dual spaces.
- The *categories* of Boolean algebras and Boolean spaces are dually equivalent.

<table>
<thead>
<tr>
<th>Algebras</th>
<th>dual to</th>
<th>Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>subalgebras</td>
<td>$\leftrightarrow$</td>
<td>quotient objects</td>
</tr>
<tr>
<td>quotient algebras</td>
<td>$\leftrightarrow$</td>
<td>subobjects</td>
</tr>
<tr>
<td>homomorphisms</td>
<td>$\leftrightarrow$</td>
<td>continuous functions</td>
</tr>
</tbody>
</table>
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a continuous function between the dual spaces.
- The categories of Boolean algebras and Boolean spaces are dually equivalent.

<table>
<thead>
<tr>
<th>Algebras</th>
<th>dual to</th>
<th>Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>subalgebras</td>
<td>$\leftrightarrow$</td>
<td>quotient objects</td>
</tr>
<tr>
<td>quotient algebras</td>
<td>$\leftrightarrow$</td>
<td>subobjects</td>
</tr>
<tr>
<td>homomorphisms</td>
<td>$\leftrightarrow$</td>
<td>continuous functions</td>
</tr>
<tr>
<td>algebraic operations</td>
<td>$\leftrightarrow$</td>
<td>co-algebraic operations</td>
</tr>
</tbody>
</table>
As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a \textit{continuous} function between the dual spaces.

The \textit{categories} of Boolean algebras and Boolean spaces are dually equivalent.

<table>
<thead>
<tr>
<th>Algebras</th>
<th>dual to</th>
<th>Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>subalgebras</td>
<td>$\leftrightarrow$</td>
<td>quotient objects</td>
</tr>
<tr>
<td>quotient algebras</td>
<td>$\leftrightarrow$</td>
<td>subobjects</td>
</tr>
<tr>
<td>homomorphisms</td>
<td>$\leftrightarrow$</td>
<td>continuous functions</td>
</tr>
<tr>
<td>algebraic operations</td>
<td>$\leftrightarrow$</td>
<td>co-algebraic operations</td>
</tr>
<tr>
<td>unions (directed colimits)</td>
<td>$\leftrightarrow$</td>
<td>projective limits</td>
</tr>
</tbody>
</table>
Stone duality: summary

- Finite Boolean algebras are power sets.
Stone duality: summary

- Finite Boolean algebras are power sets.
- Boolean algebras are subalgebras of power sets.
Stone duality: summary

- Finite Boolean algebras are power sets.
- Boolean algebras are subalgebras of power sets.
- Boolean algebra homomorphisms are inverse images.
Stone duality: summary

- Finite Boolean algebras are power sets.
- Boolean algebras are subalgebras of power sets.
- Boolean algebra homomorphisms are inverse images.
- Boolean algebras are algebras of clopen sets of a compact Hausdorff topological space, called the dual space.
Stone duality: summary

- Finite Boolean algebras are power sets.
- Boolean algebras are subalgebras of power sets.
- Boolean algebra homomorphisms are inverse images.
- Boolean algebras are algebras of clopen sets of a compact Hausdorff topological space, called the dual space.
- Subalgebras of the Boolean algebra correspond to quotient spaces of the dual space.
Stone duality: summary

- Finite Boolean algebras are power sets.
- Boolean algebras are subalgebras of power sets.
- Boolean algebra homomorphisms are inverse images.
- Boolean algebras are algebras of clopen sets of a compact Hausdorff topological space, called the dual space.
- Subalgebras of the Boolean algebra correspond to quotient spaces of the dual space.
- Quotients of the Boolean algebra correspond to closed subspaces of the dual space.
References for Part II

- Basics on duality theory for Boolean algebras: Chapter 11 in
  Cambridge University Press, May 6, 2002

- The duality-theoretic view on varieties:

- A proof of Schützenberger's Theorem: Chapter VI in
  H. Straubing. *Finite automata, formal logic, and circuit complexity*

- Our recent work on applications of model theory:

v. Gool (UvA & CCNY)

Machines, Models, Monoids, Modal logic
Logic Tutorial, TbiLLC 2017 32 / 32
References for Part II


- The duality-theoretic view on varieties:
References for Part II

- Basics on duality theory for Boolean algebras: Chapter 11 in

- The duality-theoretic view on varieties:

- A proof of Schützenberger’s Theorem: Chapter VI in
References for Part II

- Basics on duality theory for Boolean algebras: Chapter 11 in
  Cambridge University Press, May 6, 2002

- The duality-theoretic view on varieties:
  M. Gehrke. “Stone duality, topological algebra, and recognition”. In: *J.
  Pure Appl. Algebra* 220.7 (2016), pp. 2711–2747

- A proof of Schützenberger’s Theorem: Chapter VI in
  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:
  S. J. v. Gool and B. Steinberg. “Pro-aperiodic monoids via saturated
  models”. In: *STACS 2017.* Vol. 66. LIPIcs.
Outline

1. Part I: Formal Languages, Automata, and Algebra
2. Part II: Duality and Varieties of Monoids
Recap

- Any regular language $L$ comes with a finite closed Boolean subalgebra, $B(L)$, of $\text{Reg}(\Sigma^*)$ and a finite monoid quotient, $M(L)$, of $\Sigma^*$. 
Recap

- Any regular language $L$ comes with a finite closed Boolean subalgebra, $B(L)$, of $\text{Reg}(\Sigma^*)$ and a finite monoid quotient, $M(L)$, of $\Sigma^*$.
- The syntactic monoid $M(L)$ is the Stone dual (= atom set) of the closed Boolean subalgebra $B(L)$.
Recap

- Any regular language $L$ comes with a finite closed Boolean subalgebra, $B(L)$, of $\text{Reg}(\Sigma^*)$ and a finite monoid quotient, $M(L)$, of $\Sigma^*$.
- The syntactic monoid $M(L)$ is the Stone dual (= atom set) of the closed Boolean subalgebra $B(L)$.
- Properties of the syntactic monoid reflect properties of the language.
Recap

- Any regular language $L$ comes with a finite closed Boolean subalgebra, $B(L)$, of $\text{Reg}(\Sigma^*)$ and a finite monoid quotient, $M(L)$, of $\Sigma^*$.
- The syntactic monoid $M(L)$ is the Stone dual (= atom set) of the closed Boolean subalgebra $B(L)$.
- Properties of the syntactic monoid reflect properties of the language.
- The syntactic monoid $M(L)$ is aperiodic iff $L$ is first-order definable.
Recap

- Any regular language \( L \) comes with a finite closed Boolean subalgebra, \( B(L) \), of \( \text{Reg}(\Sigma^*) \) and a finite monoid quotient, \( M(L) \), of \( \Sigma^* \).
- The syntactic monoid \( M(L) \) is the Stone dual (\( = \) atom set) of the closed Boolean subalgebra \( B(L) \).
- Properties of the syntactic monoid reflect properties of the language.
- The syntactic monoid \( M(L) \) is aperiodic iff \( L \) is first-order definable.
- In general, varieties (\( HSP^{\text{fin}} \)-classes) of monoids correspond to varieties of languages.
Recap

- Any regular language $L$ comes with a finite closed Boolean subalgebra, $B(L)$, of $\text{Reg}(\Sigma^*)$ and a finite monoid quotient, $M(L)$, of $\Sigma^*$.
- The syntactic monoid $M(L)$ is the Stone dual (= atom set) of the closed Boolean subalgebra $B(L)$.
- Properties of the syntactic monoid reflect properties of the language.
- The syntactic monoid $M(L)$ is aperiodic iff $L$ is first-order definable.
- In general, varieties ($HSP^{\text{fin}}$-classes) of monoids correspond to varieties of languages.
- Full (non-finite) Stone duality can be used to explain and analyze this correspondence.
Recap

- Any regular language \( L \) comes with a finite closed Boolean subalgebra, \( B(L) \), of \( \text{Reg}(\Sigma^*) \) and a finite monoid quotient, \( M(L) \), of \( \Sigma^* \).
- The syntactic monoid \( M(L) \) is the Stone dual (= atom set) of the closed Boolean subalgebra \( B(L) \).
- Properties of the syntactic monoid reflect properties of the language.
- The syntactic monoid \( M(L) \) is aperiodic iff \( L \) is first-order definable.
- In general, varieties (\( HSP^{\text{fin}} \)-classes) of monoids correspond to varieties of languages.
- Full (non-finite) Stone duality can be used to explain and analyze this correspondence but how?
Outline Part III

1 Full Duality and Varieties
   - Varieties
   - Full Stone duality
   - Intuitionistic intermezzo
   - Profinite equations

2 Aperiodic pointlikes
   - Separation problem
   - Pointlike sets
   - Henckell’s Theorem

3 The Future
1 Full Duality and Varieties

- Varieties
- Full Stone duality
- Intuitionistic intermezzo
- Profinite equations
Varieties of monoids and languages

- A class of finite monoids $\mathbf{V}$ is a (pseudo)variety if it is closed under homomorphic images (H), submonoids (S) and finite products ($P^\text{fin}$).
Varieties of monoids and languages

- A class of finite monoids $\mathbf{V}$ is a (pseudo)\emph{variety} if it is closed under homomorphic images (H), submonoids (S) and finite products ($P^{\text{fin}}$).
- For a variety of monoids $\mathbf{V}$, define $\mathcal{V}(\Sigma^*)$ to be the class of $\Sigma$-languages $L$ such that $M(L) \in \mathbf{V}$. 

Theorem (Eilenberg) The map $\mathbf{V} \rightarrow \mathcal{V}$ is an order-bijection between varieties of finite monoids and varieties of regular languages.
Varieties of monoids and languages

- A class of finite monoids $\mathbf{V}$ is a (pseudo)variety if it is closed under homomorphic images (H), submonoids (S) and finite products ($P^{\text{fin}}$).
- For a variety of monoids $\mathbf{V}$, define $\mathbf{V}(\Sigma^*)$ to be the class of $\Sigma$-languages $L$ such that $M(L) \in \mathbf{V}$.
- Then $\{\mathbf{V}(\Sigma^*)\}_\Sigma$ is a variety of regular languages: a collection of Boolean subalgebras of $\text{Reg}(\Sigma^*)$ which is closed under inverse images of homomorphisms $\Sigma_1^* \rightarrow \Sigma_2^*$. 

Theorem (Eilenberg)

The map $\mathbf{V} \mapsto \mathbf{V}$ is an order-bijection between varieties of finite monoids and varieties of regular languages.
Varieties of monoids and languages

- A class of finite monoids $\mathbf{V}$ is a (pseudo)variety if it is closed under homomorphic images (H), submonoids (S) and finite products ($P^{\text{fin}}$).
- For a variety of monoids $\mathbf{V}$, define $\mathcal{V}(\Sigma^*)$ to be the class of $\Sigma$-languages $L$ such that $M(L) \in \mathbf{V}$.
- Then $\{\mathcal{V}(\Sigma^*)\}_\Sigma$ is a variety of regular languages: a collection of Boolean subalgebras of $\text{Reg}(\Sigma^*)$ which is closed under inverse images of homomorphisms $\Sigma_1^* \rightarrow \Sigma_2^*$.

**Theorem (Eilenberg)**

The map $\mathbf{V} \mapsto \mathcal{V}$ is an order-bijection between varieties of finite monoids and varieties of regular languages.
Equations?

- Birkhoff’s theorem: varieties of (arbitrary) algebras can be defined by (finite) equations.
Equations?

- Birkhoff’s theorem: varieties of (arbitrary) algebras can be defined by (finite) equations.
- What about (pseudo)varieties of finite algebras?
Equations?

- Birkhoff’s theorem: varieties of (arbitrary) algebras can be defined by (finite) equations.
- What about (pseudo)varieties of finite algebras?
- We need profinite equations.
Equations?

- Birkhoff’s theorem: varieties of (arbitrary) algebras can be defined by (finite) equations.
- What about (pseudo)varieties of finite algebras?
- We need profinite equations.
- To explain what these are, and why we need them: full Stone duality.
Stone duality: general case

Proposition

Every Boolean algebra $B$ can be embedded into a Boolean algebra of the form $\mathcal{P}(X)$.
Stone duality: general case

Proposition

Every Boolean algebra $B$ can be embedded into a Boolean algebra of the form $\mathcal{P}(X)$, and there is a unique such embedding for which the topology generated by the sets in the image of $B$ is compact and Hausdorff (and zero-dimensional).
Proposition

Every Boolean algebra $B$ can be embedded into a Boolean algebra of the form $\mathcal{P}(X)$, and there is a unique such embedding for which the topology generated by the sets in the image of $B$ is compact and Hausdorff (and zero-dimensional).

‘Construction’ of the embedding.

Take $X$ to be the set of ultrafilters of $B$. 
# Stone duality: general case

## Proposition

> Every Boolean algebra $B$ can be embedded into a Boolean algebra of the form $\mathcal{P}(X)$, and there is a unique such embedding for which the topology generated by the sets in the image of $B$ is compact and Hausdorff (and zero-dimensional).

## ‘Construction’ of the embedding.

Take $X$ to be the set of ultrafilters of $B$. Identify $b \in B$ with the set, $\hat{b}$, of ultrafilters containing it.
Stone duality: general case

Proposition

Every Boolean algebra $B$ can be embedded into a Boolean algebra of the form $\mathcal{P}(X)$, and there is a unique such embedding for which the topology generated by the sets in the image of $B$ is compact and Hausdorff (and zero-dimensional).

‘Construction’ of the embedding.

Take $X$ to be the set of ultrafilters of $B$. Identify $b \in B$ with the set, $\hat{b}$, of ultrafilters containing it.

- A Boolean space is a compact Hausdorff zero-dimensional space.
Stone duality: general case

Proposition

Every Boolean algebra $B$ can be embedded into a Boolean algebra of the form $\mathcal{P}(X)$, and there is a unique such embedding for which the topology generated by the sets in the image of $B$ is compact and Hausdorff (and zero-dimensional).

‘Construction’ of the embedding.

Take $X$ to be the set of ultrafilters of $B$.
Identify $b \in B$ with the set, $\hat{b}$, of ultrafilters containing it.

- A Boolean space is a compact Hausdorff zero-dimensional space.
- Equivalently, a Boolean space is a profinite object in the category of topological spaces.
Stone duality: example

Example
The dual space of the Lindenbaum algebra of CPL on a countable set \( V = \{ p_1, p_2, p_3, \ldots \} \) is the Cantor space \( \{0, 1\}^V \).
Exercises

1. What is the dual space of the Boolean algebra of finite subsets of the natural numbers and their complements?

2. Use what you know about classical propositional logic to prove that the Lindenbaum algebra of CPL on a countable set $V = \{p_1, p_2, p_3, \ldots \}$ can be embedded into $\mathcal{P}(\{0, 1\}^V)$.

3. (*) Show that the topology generated by the image of the embedding in (2) is compact and Hausdorff.

4. (*) Show that the topology generated by the image of the embedding in (2) coincides with the topology of the Cantor space.
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a *continuous* function between the dual spaces.
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a continuous function between the dual spaces.
- The categories of Boolean algebras and Boolean spaces are dually equivalent.
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a *continuous* function between the dual spaces.
- The *categories* of Boolean algebras and Boolean spaces are *dually equivalent*.

<table>
<thead>
<tr>
<th>Algebras</th>
<th>dual to</th>
<th>Spaces</th>
</tr>
</thead>
</table>
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a continuous function between the dual spaces.
- The categories of Boolean algebras and Boolean spaces are dually equivalent.

<table>
<thead>
<tr>
<th>Algebras</th>
<th>dual to</th>
<th>Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>subalgebras</td>
<td>⇔</td>
<td>quotient objects</td>
</tr>
</tbody>
</table>
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a \textit{continuous} function between the dual spaces.
- The \textit{categories} of Boolean algebras and Boolean spaces are \textit{dually equivalent}.

<table>
<thead>
<tr>
<th>Algebras</th>
<th>dual to</th>
<th>Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>subalgebras</td>
<td>$\leftrightarrow$</td>
<td>quotient objects</td>
</tr>
<tr>
<td>quotient algebras</td>
<td>$\leftrightarrow$</td>
<td>subobjects</td>
</tr>
</tbody>
</table>
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a *continuous* function between the dual spaces.
- The *categories* of Boolean algebras and Boolean spaces are *dually equivalent*.

<table>
<thead>
<tr>
<th>Algebras</th>
<th>dual to</th>
<th>Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>subalgebras</td>
<td>$\leftrightarrow$</td>
<td>quotient objects</td>
</tr>
<tr>
<td>quotient algebras</td>
<td>$\leftrightarrow$</td>
<td>subobjects</td>
</tr>
<tr>
<td>homomorphisms</td>
<td>$\leftrightarrow$</td>
<td>continuous functions</td>
</tr>
</tbody>
</table>
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a continuous function between the dual spaces.

- The categories of Boolean algebras and Boolean spaces are dually equivalent.

<table>
<thead>
<tr>
<th>Algebras</th>
<th>dual to</th>
<th>Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>subalgebras</td>
<td>$\leftrightarrow$</td>
<td>quotient objects</td>
</tr>
<tr>
<td>quotient algebras</td>
<td>$\leftrightarrow$</td>
<td>subobjects</td>
</tr>
<tr>
<td>homomorphisms</td>
<td>$\leftrightarrow$</td>
<td>continuous functions</td>
</tr>
<tr>
<td>algebraic operations</td>
<td>$\leftrightarrow$</td>
<td>co-algebraic operations</td>
</tr>
</tbody>
</table>
Duality: categorical level

- As in the finite case, all homomorphisms between Boolean algebras are of the form $f^{-1}$, for $f$ a \textit{continuous} function between the dual spaces.
- The \textit{categories} of Boolean algebras and Boolean spaces are \textit{dually equivalent}.

<table>
<thead>
<tr>
<th>Algebras</th>
<th>dual to</th>
<th>Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>subalgebras</td>
<td>$\leftrightarrow$</td>
<td>quotient objects</td>
</tr>
<tr>
<td>quotient algebras</td>
<td>$\leftrightarrow$</td>
<td>subobjects</td>
</tr>
<tr>
<td>homomorphisms</td>
<td>$\leftrightarrow$</td>
<td>continuous functions</td>
</tr>
<tr>
<td>algebraic operations</td>
<td>$\leftrightarrow$</td>
<td>co-algebraic operations</td>
</tr>
<tr>
<td>unions (directed colimits)</td>
<td>$\leftrightarrow$</td>
<td>projective limits</td>
</tr>
</tbody>
</table>
Stone duality: summary

- Finite Boolean algebras are power sets.
Stone duality: summary

- Finite Boolean algebras are power sets.
- Boolean algebras are subalgebras of power sets.
Stone duality: summary

- Finite Boolean algebras are power sets.
- Boolean algebras are subalgebras of power sets.
- Boolean algebra homomorphisms are inverse images.
Stone duality: summary

- Finite Boolean algebras are power sets.
- Boolean algebras are subalgebras of power sets.
- Boolean algebra homomorphisms are inverse images.
- Boolean algebras are algebras of clopen sets of a compact Hausdorff topological space, called the dual space.
Stone duality: summary

- Finite Boolean algebras are power sets.
- Boolean algebras are subalgebras of power sets.
- Boolean algebra homomorphisms are inverse images.
- Boolean algebras are algebras of clopen sets of a compact Hausdorff topological space, called the dual space.
- Subalgebras of the Boolean algebra correspond to quotient spaces of the dual space.
Stone duality: summary

- Finite Boolean algebras are power sets.
- Boolean algebras are subalgebras of power sets.
- Boolean algebra homomorphisms are inverse images.
- Boolean algebras are algebras of clopen sets of a compact Hausdorff topological space, called the **dual space**.
- Subalgebras of the Boolean algebra correspond to quotient spaces of the dual space.
- Quotients of the Boolean algebra correspond to closed subspaces of the dual space.
Intuitionistic Intermezzo
An open mapping theorem for Esakia spaces

- Stone duality generalizes to *Heyting algebras*, the structures for *intuitionistic* propositional logic analogous to Boolean algebras.
An open mapping theorem for Esakia spaces

- Stone duality generalizes to *Heyting algebras*, the structures for *intuitionistic* propositional logic analogous to Boolean algebras.
- The Boolean space is equipped with a *partial order* (= the Kripke accessibility relation in semantics for intuitionistic logic).

Heyting algebra homomorphisms also require special attention: their duals are *continuous p-morphisms*. Esakia duality is useful, for example, for proving interpolation properties of intermediate logics. In recent joint work with L. Reggio, we prove an open mapping theorem for Esakia spaces dual to finitely presented Heyting algebras. Our result in particular implies Pitts' Uniform Interpolation Theorem for IPC.
An open mapping theorem for Esakia spaces

- Stone duality generalizes to *Heyting algebras*, the structures for *intuitionistic* propositional logic analogous to Boolean algebras.
- The Boolean space is equipped with a *partial order* (= the Kripke accessibility relation in semantics for intuitionistic logic).
- The spaces dual to Heyting algebras were characterized by L. Esakia and are now called *Esakia spaces*.

- Heyting algebra homomorphisms also require special attention: their duals are *continuous p-morphisms*.
- Esakia duality is useful, for example, for proving interpolation properties of intermediate logics.

In recent joint work with L. Reggio, we prove an open mapping theorem for Esakia spaces dual to finitely presented Heyting algebras. Our result in particular implies Pitts' Uniform Interpolation Thoerem for IPC.
An open mapping theorem for Esakia spaces

- Stone duality generalizes to *Heyting algebras*, the structures for *intuitionistic* propositional logic analogous to Boolean algebras.
- The Boolean space is equipped with a *partial order* (= the Kripke accessibility relation in semantics for intuitionistic logic).
- The spaces dual to Heyting algebras were characterized by L. Esakia and are now called *Esakia spaces*.
- Heyting algebra homomorphisms also require special attention: their duals are *continuous p-morphisms*. 
An open mapping theorem for Esakia spaces

- Stone duality generalizes to *Heyting algebras*, the structures for *intuitionistic* propositional logic analogous to Boolean algebras.
- The Boolean space is equipped with a *partial order* (= the Kripke accessibility relation in semantics for intuitionistic logic).
- The spaces dual to Heyting algebras were characterized by L. Esakia and are now called *Esakia spaces*.
- Heyting algebra homomorphisms also require special attention: their duals are *continuous p-morphisms*.
- Esakia duality is useful, for example, for proving *interpolation* properties of intermediate logics.
An open mapping theorem for Esakia spaces

- Stone duality generalizes to Heyting algebras, the structures for intuitionistic propositional logic analogous to Boolean algebras.
- The Boolean space is equipped with a partial order (= the Kripke accessibility relation in semantics for intuitionistic logic).
- The spaces dual to Heyting algebras were characterized by L. Esakia and are now called Esakia spaces.
- Heyting algebra homomorphisms also require special attention: their duals are continuous p-morphisms.
- Esakia duality is useful, for example, for proving interpolation properties of intermediate logics.
- In recent joint work with L. Reggio, we prove an open mapping theorem for Esakia spaces dual to finitely presented Heyting algebras.
An open mapping theorem for Esakia spaces

- Stone duality generalizes to Heyting algebras, the structures for intuitionistic propositional logic analogous to Boolean algebras.
- The Boolean space is equipped with a partial order (= the Kripke accessibility relation in semantics for intuitionistic logic).
- The spaces dual to Heyting algebras were characterized by L. Esakia and are now called Esakia spaces.
- Heyting algebra homomorphisms also require special attention: their duals are continuous p-morphisms.
- Esakia duality is useful, for example, for proving interpolation properties of intermediate logics.
- In recent joint work with L. Reggio, we prove an open mapping theorem for Esakia spaces dual to finitely presented Heyting algebras.
- Our result in particular implies Pitts’ Uniform Interpolation Theorem for IPC.
Another hint of duality for interpolation?

- The use of duality for analyzing quantifiers is not limited to the context of regular languages.
Another hint of duality for interpolation?

- The use of duality for **analyzing quantifiers** is not limited to the context of regular languages.
- A classical example is Rasiowa & Sikorski’s proof of the completeness of classical predicate logic via Stone duality and the Baire category theorem.
Another hint of duality for interpolation?

- The use of duality for analyzing quantifiers is not limited to the context of regular languages.
- A classical example is Rasiowa & Sikorski’s proof of the completeness of classical predicate logic via Stone duality and the Baire category theorem.
- A recent example (I claim) is the counterexample to interpolation for constant domain intuitionistic predicate logic (Mints, Olkhovikov, Urquhart JSL 2013).
Another hint of duality for interpolation?

- The use of duality for analyzing quantifiers is not limited to the context of regular languages.
- A classical example is Rasiowa & Sikorski’s proof of the completeness of classical predicate logic via Stone duality and the Baire category theorem.
- A recent example (I claim) is the counterexample to interpolation for constant domain intuitionistic predicate logic (Mints, Olkhovikov, Urquhart JSL 2013).
- In the latter, and in Olkhovikov’s work on van-Benthem-style characterizations (2012-2015), the use of duality is not (yet) explicit.
End of Intuitionistic Intermezzo
Example

The dual space of the Boolean algebra $\text{Reg}(\Sigma^*)$ of regular $\Sigma$-languages is the projective limit of the diagram $(q: \Sigma^* \to M)$ of finite quotients of $\Sigma^*$. This is the space underlying the free profinite monoid $c\Sigma^*$ on $\Sigma$. Thus, the free profinite monoid over $\Sigma$ is the 'canonical Kripke model' for MSO on finite words (since the Lindenbaum algebra is $\text{Reg}(\Sigma^*)$).
Stone duality: crucial example for language varieties

Example

The dual space of the Boolean algebra $\text{Reg}(\Sigma^*)$ of regular $\Sigma$-languages is the projective limit of the diagram $(q: \Sigma^* \rightarrow M)$ of finite quotients of $\Sigma^*$. This is the space underlying the free profinite monoid, $\hat{\Sigma^*}$, on $\Sigma$. 

Thus, the free profinite monoid over $\Sigma$ is the ‘canonical Kripke model’ for MSO on finite words (since the Lindenbaum algebra is $\text{Reg}(\Sigma^*)$).
The dual space of the Boolean algebra $\text{Reg}(\Sigma^*)$ of regular $\Sigma$-languages is the projective limit of the diagram $(q : \Sigma^* \to M)$ of finite quotients of $\Sigma^*$. This is the space underlying the free profinite monoid, $\widehat{\Sigma^*}$, on $\Sigma$. Thus, the free profinite monoid over $\Sigma$ is the ‘canonical Kripke model’ for MSO on finite words (since the Lindenbaum algebra is $\text{Reg}(\Sigma^*)$).
“In the mid-1970s when I was at Oxford, it occurred to me that, using regular events in the free monoid on a finite alphabet as neighborhoods, one could make a completion to ‘infinite words’. I even suggested to one of my students to consider this idea for a thesis. Neither he nor I could make much progress in analyzing this algebra or applying this idea, however, and he went on to write a different thesis (fortunately).”
On the free profinite monoid

“In the mid-1970s when I was at Oxford, it occurred to me that, using regular events in the free monoid on a finite alphabet as neighborhoods, one could make a completion to ‘infinite words’. I even suggested to one of my students to consider this idea for a thesis. Neither he nor I could make much progress in analyzing this algebra or applying this idea, however, and he went on to write a different thesis (fortunately).”

[D. Scott, via e-mail, Nov. 9, 2016]
Describing varieties: profinite equations

- Let \( \mathbf{V} \) be a variety of finite monoids.
Describing varieties: profinite equations

- Let $\mathcal{V}$ be a variety of finite monoids.
- Let $\mathcal{V}$ be the corresponding variety of regular languages.
Describing varieties: profinite equations

- Let $\mathbf{V}$ be a variety of finite monoids.
- Let $\mathcal{V}$ be the corresponding variety of regular languages.
- For every alphabet $\Sigma$, $\mathcal{V}(\Sigma^*)$ is a closed subalgebra of $\text{Reg}(\Sigma^*)$. 
Let $\mathbf{V}$ be a variety of finite monoids.

Let $\mathcal{V}$ be the corresponding variety of regular languages.

For every alphabet $\Sigma$, $\mathcal{V}(\Sigma^*)$ is a closed subalgebra of $\text{Reg}(\Sigma^*)$.

The dual of this closed subalgebra is a continuous monoid quotient $\hat{\Sigma}^* \to \hat{F}_V(\Sigma)$, the free pro-$\mathbf{V}$-monoid on $\Sigma$. 

Describing varieties: profinite equations

- Let $\mathbf{V}$ be a variety of finite monoids.
- Let $\mathcal{V}$ be the corresponding variety of regular languages.
- For every alphabet $\Sigma$, $\mathcal{V}(\Sigma^*)$ is a closed subalgebra of $\text{Reg}(\Sigma^*)$.
- The dual of this closed subalgebra is a continuous monoid quotient $\widehat{\Sigma^*} \rightarrow \widehat{F}_{\mathbf{V}}(\Sigma)$, the free pro-$\mathbf{V}$-monoid on $\Sigma$.
- The fact that $\mathcal{V}$ is closed under inverse images of homomorphisms means that the quotients $\widehat{\Sigma^*} \rightarrow \widehat{F}_{\mathbf{V}}(\Sigma)$ are substitution-invariant.
Describing varieties: profinite equations

- Let \( \mathbf{V} \) be a variety of finite monoids.
- Let \( \mathcal{V} \) be the corresponding variety of regular languages.
- For every alphabet \( \Sigma \), \( \mathcal{V}(\Sigma^*) \) is a closed subalgebra of \( \text{Reg}(\Sigma^*) \).
- The dual of this closed subalgebra is a continuous monoid quotient \( \hat{\Sigma}^* \rightarrow \hat{F}_\mathbf{V}(\Sigma) \), the free pro-\( \mathbf{V} \)-monoid on \( \Sigma \).
- The fact that \( \mathcal{V} \) is closed under inverse images of homomorphisms means that the quotients \( \hat{\Sigma}^* \rightarrow \hat{F}_\mathbf{V}(\Sigma) \) are substitution-invariant.
- Thus, varieties can be described by profinite equations.
Example

The free pro-aperiodic monoid, $\hat{F}_A(\Sigma^*)$, is the quotient of $\hat{\Sigma}^*$ by the equivalence relation defined by the substitution-invariant equation $x^\omega = x^\omega x$. 
Profinite equations: example

Example

The free pro-aperiodic monoid, $\widehat{F_A}(\Sigma^*)$, is the quotient of $\widehat{\Sigma}^*$ by the equivalence relation defined by the substitution-invariant equation

$$x^\omega = x^\omega x.$$ 

Here, $(\cdot)^\omega : \widehat{\Sigma}^* \to \widehat{\Sigma}^*$ is the operation which sends any $x$ to the idempotent $x^\omega$ in $\{x^n \mid n \geq 1\}$.
Duality beyond the profinite

- Let $\mathcal{V}$ be any closed subalgebra of $\mathcal{P}(\Sigma^*)$. 


Duality beyond the profinite

- Let $\mathcal{V}$ be any closed subalgebra of $\mathcal{P}(\Sigma^*)$.
- It corresponds again to a topological quotient, but now of $\beta\Sigma^*$, the Stone-Cech compactification of the discrete free monoid.
Duality beyond the profinite

- Let $\mathcal{V}$ be any closed subalgebra of $\mathcal{P}(\Sigma^*)$.
- It corresponds again to a topological quotient, but now of $\beta\Sigma^*$, the Stone-Cech compactification of the discrete free monoid.
- The story for the monoid operation is a bit more complicated (internal monoid action).
Duality beyond the profinite

- Let $\mathcal{V}$ be any closed subalgebra of $\mathcal{P}(\Sigma^*)$.
- It corresponds again to a topological quotient, but now of $\beta\Sigma^*$, the Stone-Cech compactification of the discrete free monoid.
- The story for the monoid operation is a bit more complicated (internal monoid action).
- This idea, combined with methods from circuit complexity theory, leads to ‘ultrafilter equations’ for characterizing classes of non-regular languages.
Duality beyond the profinite

- Let $\mathcal{V}$ be any closed subalgebra of $\mathcal{P}(\Sigma^*)$.
- It corresponds again to a topological quotient, but now of $\beta\Sigma^*$, the Stone-Cech compactification of the discrete free monoid.
- The story for the monoid operation is a bit more complicated (internal monoid action).
- This idea, combined with methods from circuit complexity theory, leads to ‘ultrafilter equations’ for characterizing classes of non-regular languages.

Aperiodic pointlikes
- Separation problem
- Pointlike sets
- Henckell’s Theorem
Separation problem: language version

- From here on, we work with semigroups, and ‘$\Sigma$-language’ means subset of $\Sigma^+$. 

Let $V$ be a variety of finite semigroups with corresponding variety of languages $V$. 

Separation Problem: Given two disjoint regular $\Sigma$-languages $L_1$, $L_2$, is it possible to find a language, $K$, in $V(\Sigma)$ which separates $L_1$ from $L_2$?

Here, $K$ separates $L_1$ from $L_2$ if $L_1 \subseteq K$ and $L_2 \not\subseteq K$.

If $'1$ and $'2$ are MSO sentences defining $L_1$ and $L_2$, respectively, then disjointness means $'1 \not\Rightarrow '2$.

The logic formulation of the separation problem is: does there exist such that $'1 \not\Rightarrow '2$, with the language $K = L$ in $V(\Sigma)$.

In general, this problem can fail to be decidable, even when membership in $V$ is decidable.
Separation problem: language version

- From here on, we work with semigroups, and ‘\(\Sigma\)-language’ means subset of \(\Sigma^+\).
- Let \(\mathbf{V}\) be a variety of finite semigroups with corresponding variety of languages \(\mathcal{V}\).

Separation Problem:

Given two disjoint regular \(\Sigma\)-languages \(L_1, L_2\), is it possible to find a language, \(K\), in \(\mathbf{V}(\Sigma)\) which separates \(L_1\) from \(L_2\)?

Here, \(K\) separates \(L_1\) from \(L_2\) if \(L_1 \cap K \neq \emptyset\) and \(L_2 \cap K = \emptyset\).

If \(\varphi_1\) and \(\varphi_2\) are MSO sentences defining \(L_1\) and \(L_2\), respectively, then disjointness means \(\varphi_1 \not\equiv \neg \varphi_2\).

The logic formulation of the separation problem is: does there exist such that \(\varphi_1 \not\equiv \neg \varphi_2\), with the language \(K\) in \(\mathbf{V}(\Sigma)\).

In general, this problem can fail to be decidable, even when membership in \(\mathbf{V}\) is decidable.
Separation problem: language version

- From here on, we work with semigroups, and ‘$\Sigma$-language’ means subset of $\Sigma^+$. 
- Let $\mathbf{V}$ be a variety of finite semigroups with corresponding variety of languages $\mathcal{V}$.
- **Separation Problem:** Given two disjoint regular $\Sigma$-languages $L_1, L_2$, is it possible to find a language, $K$, in $\mathcal{V}(\Sigma)$ which *separates* $L_1$ from $L_2$?
Separation problem: language version

- From here on, we work with semigroups, and ‘$\Sigma$-language’ means subset of $\Sigma^+$. 
- Let $V$ be a variety of finite semigroups with corresponding variety of languages $\mathcal{V}$.
- **Separation Problem**: Given two disjoint regular $\Sigma$-languages $L_1, L_2$, is it possible to find a language, $K$, in $\mathcal{V}(\Sigma)$ which separates $L_1$ from $L_2$?
- Here, $K$ separates $L_1$ from $L_2$ if $L_1 \subseteq K$ and $L_2 \cap K = \emptyset$. 
Separation problem: language version

- From here on, we work with semigroups, and ‘$\Sigma$-language’ means subset of $\Sigma^+$.

- Let $\mathbf{V}$ be a variety of finite semigroups with corresponding variety of languages $\mathcal{V}$.

- **Separation Problem:** Given two disjoint regular $\Sigma$-languages $L_1$, $L_2$, is it possible to find a language, $K$, in $\mathcal{V}(\Sigma)$ which separates $L_1$ from $L_2$?

- Here, $K$ separates $L_1$ from $L_2$ if $L_1 \subseteq K$ and $L_2 \cap K = \emptyset$.

- If $\varphi_1$ and $\varphi_2$ are MSO sentences defining $L_1$ and $L_2$, respectively, then disjointness means $\varphi_1 \vdash \neg \varphi_2$. 


Separation problem: language version

- From here on, we work with semigroups, and ‘Σ-language’ means subset of $\Sigma^+$.
- Let $\mathcal{V}$ be a variety of finite semigroups with corresponding variety of languages $\mathcal{V}$.
- **Separation Problem**: Given two disjoint regular $\Sigma$-languages $L_1$, $L_2$, is it possible to find a language, $K$, in $\mathcal{V}(\Sigma)$ which separates $L_1$ from $L_2$?
- Here, $K$ separates $L_1$ from $L_2$ if $L_1 \subseteq K$ and $L_2 \cap K = \emptyset$.
- If $\varphi_1$ and $\varphi_2$ are MSO sentences defining $L_1$ and $L_2$, respectively, then disjointness means $\varphi_1 \vdash \neg \varphi_2$.
- The *logic formulation* of the separation problem is: does there exist $\psi$ such that $\varphi_1 \vdash \psi \vdash \neg \varphi_2$, with the language $K = L_\psi$ in $\mathcal{V}(\Sigma)$.
Separation problem: language version

- From here on, we work with semigroups, and ‘Σ-language’ means subset of Σ⁺.
- Let \( \mathbf{V} \) be a variety of finite semigroups with corresponding variety of languages \( \mathcal{V} \).
- **Separation Problem:** Given two disjoint regular Σ-languages \( L_1, L_2 \), is it possible to find a language, \( K \), in \( \mathcal{V}(\Sigma) \) which separates \( L_1 \) from \( L_2 \)?
- Here, \( K \) separates \( L_1 \) from \( L_2 \) if \( L_1 \subseteq K \) and \( L_2 \cap K = \emptyset \).
- If \( \varphi_1 \) and \( \varphi_2 \) are MSO sentences defining \( L_1 \) and \( L_2 \), respectively, then disjointness means \( \varphi_1 \models \neg \varphi_2 \).
- The **logic formulation** of the separation problem is: does there exist \( \psi \) such that \( \varphi_1 \models \psi \models \neg \varphi_2 \), with the language \( K = L_\psi \) in \( \mathcal{V}(\Sigma) \).
- In general, this problem can fail to be decidable, even when membership in \( \mathcal{V} \) is decidable.
Example of non-separable languages

Example (Place & Zeitoun 2016)

Let $\Sigma = \{0, 1\}$. Consider the automaton

\[
\begin{array}{c}
q_0 & \xrightarrow{1} & q_1 & \xrightarrow{0} & q_3 \\
& & & & \xleftarrow{0} & q_4 \\
& & & & \xleftarrow{1} & \xrightarrow{1} q_1 \\
q_4 & \xleftarrow{0} & q_2 & \xleftarrow{0} & q_2
\end{array}
\]

The language recognized with $q_1$ final is

$L_1 = (1(00)^*10(00)^*10)^*$(00)^*$. 
Example of non-separable languages

Example (Place & Zeitoun 2016)
Let $\Sigma = \{0, 1\}$. Consider the automaton

$$q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_3 \xrightarrow{0} q_2 \xrightarrow{0} q_4 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{0} q_0$$

The language recognized with $q_1$ final is $L_1 = (1(00)^*10(00)^*)^*1(00)^*$. 
Example of non-separable languages

Example (Place & Zeitoun 2016)

Let $\Sigma = \{0, 1\}$. Consider the automaton

The language recognized with $q_1$ final is $L_1 = (1(00)^*10(00)^*)^*1(00)^*$. The language recognized with $q_2$ final is
Example of non-separable languages

Example (Place & Zeitoun 2016)

Let $\Sigma = \{0, 1\}$. Consider the automaton

\[
\begin{array}{c}
q_0 \quad 1 \quad q_1 \\
\downarrow \quad 0 \quad \uparrow 0 \\
q_1 \quad 0 \quad q_3 \\
\downarrow 1 \quad \uparrow 1 \\
q_4 \quad 0 \quad q_2 \\
\end{array}
\]

The language recognized with $q_1$ final is $L_1 = (1(00)^*10(00)^*)1(00)^*$. The language recognized with $q_2$ final is $L_2 = (1(00)^*10(00)^*)^+$. 
Example (Place & Zeitoun 2016)

Let $\Sigma = \{0, 1\}$. Consider the automaton

![Automaton Diagram]

The language recognized with $q_1$ final is $L_1 = (1(00)^*10(00)^*)^*1(00)^*$. The language recognized with $q_2$ final is $L_2 = (1(00)^*10(00)^*)^+$. The languages $L_1$ and $L_2$ are disjoint,
Let $\Sigma = \{0, 1\}$. Consider the automaton

The language recognized with $q_1$ final is $L_1 = (1(00)^*10(00)^*)^*1(00)^*$.
The language recognized with $q_2$ final is $L_2 = (1(00)^*10(00)^*)^+$. The languages $L_1$ and $L_2$ are disjoint, but not FO-separable.
Example (Place & Zeitoun 2016)

Let $\Sigma = \{0, 1\}$. Consider the automaton

The language recognized with $q_1$ final is $L_1 = (1(00)^*10(00)^*)^*1(00)^*$. The language recognized with $q_2$ final is $L_2 = (1(00)^*10(00)^*)^+$. The languages $L_1$ and $L_2$ are *disjoint*, but not FO-separable.

**Exercise:** Use first-order logic games to prove this.
The separation problem can be formulated as a problem about semigroups.
Separation problem: semigroup version

- The separation problem can be formulated as a problem about semigroups.
- We may assume $L_1$ and $L_2$ are recognized by the same semigroup homomorphism $\eta: \Sigma^+ \to S$. 

$v. Gool \ (UvA \ & \ CCNY)$
Machines, Models, Monoids, Modal logic
Logic Tutorial, TbiLLC 2017 25 / 33
Separation problem: semigroup version

- The separation problem can be formulated as a problem about semigroups.
- We may assume $L_1$ and $L_2$ are recognized by the same semigroup homomorphism $\eta: \Sigma^+ \rightarrow S$.
- So $L_1 = \eta^{-1}(R_1)$ and $L_2 = \eta^{-1}(R_2)$, with $R_1$ and $R_2$ disjoint.
Separation problem: semigroup version

- The separation problem can be formulated as a problem about semigroups.
- We may assume $L_1$ and $L_2$ are recognized by the same semigroup homomorphism $\eta: \Sigma^+ \to S$.
- So $L_1 = \eta^{-1}(R_1)$ and $L_2 = \eta^{-1}(R_2)$, with $R_1$ and $R_2$ disjoint.
- Is there a semigroup homomorphism $\theta: \Sigma^* \to T \in \mathbf{V}$, and $P \subseteq T$, such that $\eta^{-1}(R_1)$ is contained in $\theta^{-1}(P)$, and $\eta^{-1}(R_2)$ is disjoint from $\theta^{-1}(P)$?

**Fact.** The answer is 'no' if, and only if, for every $r_1 \in R_1$ and $r_2 \in R_2$, the subset $\{r_1, r_2\}$ of $S$ is pointlike.
Separation problem: semigroup version

- The separation problem can be formulated as a problem about semigroups.
- We may assume $L_1$ and $L_2$ are recognized by the same semigroup homomorphism $\eta : \Sigma^+ \to S$.
- So $L_1 = \eta^{-1}(R_1)$ and $L_2 = \eta^{-1}(R_2)$, with $R_1$ and $R_2$ disjoint.
- Is there a semigroup homomorphism $\theta : \Sigma^* \to T \in \mathbf{V}$, and $P \subseteq T$, such that $\eta^{-1}(R_1)$ is contained in $\theta^{-1}(P)$, and $\eta^{-1}(R_2)$ is disjoint from $\theta^{-1}(P)$?
- **Fact.** The answer is ‘no’ if, and only if, for every $r_1 \in R_1$ and $r_2 \in R_2$, the subset $\{r_1, r_2\}$ of $S$ is pointlike.
Pointlike sets

- A relational morphism from a semigroup $S$ to a semigroup $T$ is a relation $\varphi \subseteq S \times T$ such that $s \varphi \cdot s' \varphi \subseteq ss' \varphi$ and $s \varphi \neq \emptyset$ for all $s, s' \in S$. 

For instance, if we can compute the two-element $V$-pointlike sets of $S$, then we can decide the $V$-separation problem: Given $L_1 = \mathcal{C}_1(R_1)$, $L_2 = \mathcal{C}_1(R_2)$ for $\mathcal{C}$: $\mathcal{U} \mathcal{V}$, check if $\{r_1, r_2\}$ is pointlike for all $r_1 \in R_1$, $r_2 \in R_2$. If so, $L_1$ and $L_2$ are non-separable. In particular, to decide FO-separation, we will compute the $A$-pointlike sets, where $A$ is the variety of aperiodic semigroups.
Pointlike sets

- A relational morphism from a semigroup $S$ to a semigroup $T$ is a relation $\varphi \subseteq S \times T$ such that $s\varphi \cdot s'\varphi \subseteq ss'\varphi$ and $s\varphi \neq \emptyset$ for all $s, s' \in S$.

- Equivalently, it is a relation of the form $\beta\alpha^{-1}$, where $\alpha : U \twoheadrightarrow S$ and $\beta : U \rightarrow T$ are homomorphisms from a semigroup $U$. 

Pointlike sets

- A relational morphism from a semigroup $S$ to a semigroup $T$ is a relation $\varphi \subseteq S \times T$ such that $s\varphi \cdot s'\varphi \subseteq ss'\varphi$ and $s\varphi \neq \emptyset$ for all $s, s' \in S$.

- Equivalently, it is a relation of the form $\beta\alpha^{-1}$, where $\alpha: U \rightarrow S$ and $\beta: U \rightarrow T$ are homomorphisms from a semigroup $U$.

- A subset $X \subseteq S$ is $V$-pointlike if, for every relational morphism $\varphi: S \rightarrow T$ such that $T \in V$, there exists a point $x \in T$ such that $X \subseteq \varphi^{-1}(x)$. 

Pointlike sets

- A relational morphism from a semigroup $S$ to a semigroup $T$ is a relation $\varphi \subseteq S \times T$ such that $s \varphi \cdot s' \varphi \subseteq ss' \varphi$ and $s \varphi \neq \emptyset$ for all $s, s' \in S$.

- Equivalently, it is a relation of the form $\beta \alpha^{-1}$, where $\alpha: U \to S$ and $\beta: U \to T$ are homomorphisms from a semigroup $U$.

- A subset $X \subseteq S$ is $\mathbf{V}$-pointlike if, for every relational morphism $\varphi: S \to T$ such that $T \in \mathbf{V}$, there exists a point $x \in T$ such that $X \subseteq \varphi^{-1}(x)$.

- If we can compute the (two-element) $\mathbf{V}$-pointlike sets of $S$, then we can decide the $\mathbf{V}$-separation problem:
Pointlike sets

- A relational morphism from a semigroup $S$ to a semigroup $T$ is a relation $\varphi \subseteq S \times T$ such that $s \varphi \cdot s' \varphi \subseteq ss' \varphi$ and $s \varphi \neq \emptyset$ for all $s, s' \in S$.

- Equivalently, it is a relation of the form $\beta \alpha^{-1}$, where $\alpha : U \rightarrow S$ and $\beta : U \rightarrow T$ are homomorphisms from a semigroup $U$.

- A subset $X \subseteq S$ is $\mathbf{V}$-pointlike if, for every relational morphism $\varphi : S \rightarrow T$ such that $T \in \mathbf{V}$, there exists a point $x \in T$ such that $X \subseteq \varphi^{-1}(x)$.

- If we can compute the (two-element) $\mathbf{V}$-pointlike sets of $S$, then we can decide the $\mathbf{V}$-separation problem:

- Given $L_1 = \eta^{-1}(R_1)$, $L_2 = \eta^{-1}(R_2)$ for $\eta : \Sigma^* \rightarrow M$, check if $\{r_1, r_2\}$ is pointlike for all $r_1 \in R_1$, $r_2 \in R_2$. If so, $L_1$ and $L_2$ are non-separable.
Pointlike sets

- A relational morphism from a semigroup \( S \) to a semigroup \( T \) is a relation \( \varphi \subseteq S \times T \) such that \( s\varphi \cdot s'\varphi \subseteq ss'\varphi \) and \( s\varphi \neq \emptyset \) for all \( s, s' \in S \).

- Equivalently, it is a relation of the form \( \beta\alpha^{-1} \), where \( \alpha: U \rightarrow S \) and \( \beta: U \rightarrow T \) are homomorphisms from a semigroup \( U \).

- A subset \( X \subseteq S \) is \( V \)-pointlike if, for every relational morphism \( \varphi: S \rightarrow T \) such that \( T \in V \), there exists a point \( x \in T \) such that \( X \subseteq \varphi^{-1}(x) \).

- If we can compute the (two-element) \( V \)-pointlike sets of \( S \), then we can decide the \( V \)-separation problem:
  
  - Given \( L_1 = \eta^{-1}(R_1) \), \( L_2 = \eta^{-1}(R_2) \) for \( \eta: \Sigma^* \rightarrow M \), check if \( \{r_1, r_2\} \) is pointlike for all \( r_1 \in R_1 \), \( r_2 \in R_2 \). If so, \( L_1 \) and \( L_2 \) are non-separable.

- In particular, to decide FO-separation, we will compute the \( A \)-pointlike sets, where \( A \) is the variety of aperiodic semigroups.
The monad of $\mathbb{V}$-pointlikes

- The collection of $\mathbb{V}$-pointlike sets, $\text{PL}_V(S)$, of a semigroup $S$ is a subset of the *power semigroup*, $2^S$, of $S$.  

---

**Fact** The collection, $\text{PL}_V(S)$, of $\mathbb{V}$-pointlike subsets of a finite semigroup $S$, is a downward closed subsemigroup of $2^S$ which contains all the singletons.

**Fact** The union of a $\mathbb{V}$-pointlike subset of the semigroup $\text{PL}_V(S)$ is $\mathbb{V}$-pointlike. That is, $S : \text{PL}_V(S) \to \text{PL}_V(S)$ is well-defined.
The monad of $\mathbf{V}$-pointlikes

- The collection of $\mathbf{V}$-pointlike sets, $\text{PL}_\mathbf{V}(S)$, of a semigroup $S$ is a subset of the *power semigroup*, $2^S$, of $S$.
- Elements of $2^S$ are subsets of $S$, i.e., as a set, $2^S = \mathcal{P}(S)$. 
The monad of $\mathbf{V}$-pointlikes

- The collection of $\mathbf{V}$-pointlike sets, $\text{PL}_\mathbf{V}(S)$, of a semigroup $S$ is a subset of the power semigroup, $2^S$, of $S$.
- Elements of $2^S$ are subsets of $S$, i.e., as a set, $2^S = \mathcal{P}(S)$.
- Multiplication on $2^S$ is given by: $X \cdot Y = \{xy \mid x \in X, y \in Y\}$.

Fact: The collection, $\text{PL}_\mathbf{V}(S)$, of $\mathbf{V}$-pointlike subsets of a finite semigroup $S$, is a downward closed subsemigroup of $2^S$ which contains all the singletons.

Fact: The union of a $\mathbf{V}$-pointlike subset of the semigroup $\text{PL}_\mathbf{V}(S)$ is $\mathbf{V}$-pointlike. That is, $S : \text{PL}_\mathbf{V}(S) \to \text{PL}_\mathbf{V}(S)$ is well-defined.
The monad of $\mathbf{V}$-pointlikes

- The collection of $\mathbf{V}$-pointlike sets, $\text{PL}_\mathbf{V}(S)$, of a semigroup $S$ is a subset of the power semigroup, $2^S$, of $S$.
- Elements of $2^S$ are subsets of $S$, i.e., as a set, $2^S = \mathcal{P}(S)$.
- Multiplication on $2^S$ is given by: $X \cdot Y = \{xy \mid x \in X, y \in Y\}$.

**Fact**

*The collection, $\text{PL}_\mathbf{V}(S)$, of $\mathbf{V}$-pointlike subsets of a finite semigroup $S$, is a downward closed subsemigroup of $2^S$ which contains all the singletons.*
The monad of $\mathbf{V}$-pointlikes

- The collection of $\mathbf{V}$-pointlike sets, $\text{PL}_V(S)$, of a semigroup $S$ is a subset of the \textit{power semigroup}, $2^S$, of $S$.
- Elements of $2^S$ are subsets of $S$, i.e., as a set, $2^S = \mathcal{P}(S)$.
- Multiplication on $2^S$ is given by: $X \cdot Y = \{xy \mid x \in X, y \in Y\}$.

**Fact**

The collection, $\text{PL}_V(S)$, of $\mathbf{V}$-pointlike subsets of a finite semigroup $S$, is a downward closed subsemigroup of $2^S$ which contains all the singletons.

**Fact**

The union of a $\mathbf{V}$-pointlike subset of the semigroup $\text{PL}_V(S)$ is $\mathbf{V}$-pointlike. That is, $\bigcup : \text{PL}_V(\text{PL}_V(S)) \to \text{PL}_V(S)$ is well-defined.
Generating aperiodic-pointlike sets

- For $X \in 2^S$, define $X^{\omega+*} = \bigcup_{n \geq 0} X^\omega X^n$. 
Generating aperiodic-pointlike sets

- For $X \in 2^S$, define $X^{\omega+*} = \bigcup_{n \geq 0} X^\omega X^n$.
- **Fact.** If $X$ is $A$-pointlike, then so is $X^{\omega+*}$.
Generating aperiodic-pointlike sets

- For $X \in 2^S$, define $X^{\omega+*} = \bigcup_{n \geq 0} X^\omega X^n$.
- **Fact.** If $X$ is $A$-pointlike, then so is $X^{\omega+*}$.
- Singletons are $A$-pointlike.
Generating aperiodic-pointlike sets

- For $X \in 2^S$, define $X^{\omega+*} = \bigcup_{n \geq 0} X^\omega X^n$.
- **Fact.** If $X$ is $A$-pointlike, then so is $X^{\omega+*}$.
- Singletons are $A$-pointlike.
- Products of $A$-pointlike sets are $A$-pointlike.
Generating aperiodic-pointlike sets

- For $X \in 2^S$, define $X^{\omega+*} = \bigcup_{n \geq 0} X^\omega X^n$.
- **Fact.** If $X$ is $A$-pointlike, then so is $X^{\omega+*}$.
- Singletons are $A$-pointlike.
- Products of $A$-pointlike sets are $A$-pointlike.
- Subsets of $A$-pointlike sets are $A$-pointlike.
Generating aperiodic-pointlike sets

- For $X \in 2^S$, define $X^{\omega+*} = \bigcup_{n \geq 0} X^\omega X^n$.
- Fact. If $X$ is $A$-pointlike, then so is $X^{\omega+*}$.
- Singletons are $A$-pointlike.
- Products of $A$-pointlike sets are $A$-pointlike.
- Subsets of $A$-pointlike sets are $A$-pointlike.

**Theorem (Henckell)**

*For any finite semigroup $S$, the set of $A$-pointlikes of $S$ is the smallest downward closed subsemigroup of $2^S$ which contains the singletons and is closed under the operation $X \mapsto X^{\omega+*}$.***
Generating aperiodic-pointlike sets

- For $X \in 2^S$, define $X^{\omega++} = \bigcup_{n \geq 0} X^\omega X^n$.
- **Fact.** If $X$ is $A$-pointlike, then so is $X^{\omega++}$.
- Singletons are $A$-pointlike.
- Products of $A$-pointlike sets are $A$-pointlike.
- Subsets of $A$-pointlike sets are $A$-pointlike.

**Theorem (Henckell)**

*For any finite semigroup $S$, the set of $A$-pointlikes of $S$ is the smallest downward closed subsemigroup of $2^S$ which contains the singletons and is closed under the operation $X \mapsto X^{\omega++}$.***

In particular, the $A$-pointlikes of any finite semigroup are computable.
Recent progress

- Henckell, Rhodes and Steinberg (2010) improved on Henckell’s original proof and extended his methods to varieties of semigroups that avoid specific subgroups.
Recent progress

- Henckell, Rhodes and Steinberg (2010) improved on Henckell’s original proof and extended his methods to varieties of semigroups that avoid specific subgroups.
- Place and Zeitoun (2016) gave a logic proof of Henckell’s Thm.
- Steinberg and I (2017) gave a semigroup proof of Henckell’s Thm.

To do so, we construct a ‘merge decomposition’ of homomorphisms. This is an algebraic version of ‘quantifying over first and last occurrences’.

In addition to a short elementary proof of Henckell’s Theorem, we also give a short proof of the two-sided Krohn-Rhodes theorem. The latter, in a slogan, says: ‘semigroup theory = semilattice theory + group theory’.
Recent progress

- Henckell, Rhodes and Steinberg (2010) improved on Henckell’s original proof and extended his methods to varieties of semigroups that avoid specific subgroups.
- Place and Zeitoun (2016) gave a logic proof of Henckell’s Thm.
- Place and Zeitoun (2014-17) computed $\text{FO}_2$-pointlikes.
Recent progress

- Henckell, Rhodes and Steinberg (2010) improved on Henckell’s original proof and extended his methods to varieties of semigroups that avoid specific subgroups.
- Place and Zeitoun (2016) gave a logic proof of Henckell’s Thm.
- Place and Zeitoun (2014-17) computed FO₂-pointlikes.
- Steinberg and I (2017) gave a semigroup proof of Henckell’s Thm.
Recent progress

- Henckell, Rhodes and Steinberg (2010) improved on Henckell’s original proof and extended his methods to varieties of semigroups that avoid specific subgroups.
- Place and Zeitoun (2016) gave a logic proof of Henckell’s Thm.
- Place and Zeitoun (2014-17) computed $\text{FO}_2$-pointlikes.
- Steinberg and I (2017) gave a semigroup proof of Henckell’s Thm.
- To do so, we construct a ‘merge decomposition’ of homomorphisms.
Recent progress

- Henckell, Rhodes and Steinberg (2010) improved on Henckell’s original proof and extended his methods to varieties of semigroups that avoid specific subgroups.
- Place and Zeitoun (2016) gave a logic proof of Henckell’s Thm.
- Place and Zeitoun (2014-17) computed FO$_2$-pointlikes.
- Steinberg and I (2017) gave a semigroup proof of Henckell’s Thm.
- To do so, we construct a ‘merge decomposition’ of homomorphisms.
- This is an algebraic version of ‘quantifying over first and last occurrences’.

In addition to a short elementary proof of Henckell’s Theorem, we also give a short proof of the two-sided Krohn-Rhodes theorem. The latter, in a slogan, says: ‘semigroup theory = semilattice theory + group theory’.
Recent progress

- Henckell, Rhodes and Steinberg (2010) improved on Henckell’s original proof and extended his methods to varieties of semigroups that avoid specific subgroups.
- Place and Zeitoun (2016) gave a logic proof of Henckell’s Thm.
- Place and Zeitoun (2014-17) computed FO$_2$-pointlikes.
- Steinberg and I (2017) gave a semigroup proof of Henckell’s Thm.
- To do so, we construct a ‘merge decomposition’ of homomorphisms.
- This is an algebraic version of ‘quantifying over first and last occurrences’.
- In addition to a short elementary proof of Henckell’s Theorem, we also give a short proof of the two-sided Krohn-Rhodes theorem.
Recent progress

- Henckell, Rhodes and Steinberg (2010) improved on Henckell’s original proof and extended his methods to varieties of semigroups that avoid specific subgroups.
- Place and Zeitoun (2016) gave a logic proof of Henckell’s Thm.
- Place and Zeitoun (2014-17) computed $\text{FO}_2$-pointlikes.
- Steinberg and I (2017) gave a semigroup proof of Henckell’s Thm.
- To do so, we construct a ‘merge decomposition’ of homomorphisms.
- This is an algebraic version of ‘quantifying over first and last occurrences’.
- In addition to a short elementary proof of Henckell’s Theorem, we also give a short proof of the two-sided Krohn-Rhodes theorem.
- The latter, in a slogan, says: ‘semigroup theory = semilattice theory + group theory’.
References for Part III

References for Part III


- The duality-theoretic view on varieties:
References for Part III


References for Part III

- Basics on duality theory for Boolean algebras: Chapter 11 in 
  Cambridge University Press, May 6, 2002

- The duality-theoretic view on varieties: 
  M. Gehrke. “Stone duality, topological algebra, and recognition”. In: *J.
  Pure Appl. Algebra* 220.7 (2016), pp. 2711–2747

- On the relation between pointlikes and separation: 
  J. Almeida. “Some algorithmic problems for pseudovarieties”. In: *Publ.

- Our recent preprint on Henckell’s and Krohn-Rhodes theorems: 
  S. J. v. Gool and B. Steinberg. “Merge decompositions, two-sided 
  Aug. 2017
3 The Future
Seven questions

- How far can the decidability of pointlikes be stretched?
Seven questions

- How far can the decidability of pointlikes be stretched?
- How far do duality-theoretic methods reach beyond the regular?
Seven questions

- How far can the decidability of pointlikes be stretched?
- How far do duality-theoretic methods reach beyond the regular?
- How does our semigroup-theoretic work fit with the category/duality approach?

(Partial answers in joint work with Ghilardi)

- Can the relationship with modal logic be made more tight?
- Is anything I’ve said relevant for (formal) linguistics?
Seven questions

- How far can the decidability of pointlikes be stretched?
- How far do duality-theoretic methods reach beyond the regular?
- How does our semigroup-theoretic work fit with the category/duality approach?
- Is there a topos-theoretic interpretation of ‘logic on words’?
- What can be said about regular languages with model theory?
- Can the relationship with modal logic be made more tight?
- Is anything I’ve said relevant for (formal) linguistics?
Seven questions

- How far can the decidability of pointlikes be stretched?
- How far do duality-theoretic methods reach beyond the regular?
- How does our semigroup-theoretic work fit with the category/duality approach?
- Is there a topos-theoretic interpretation of ‘logic on words’?
- What can be said about regular languages with model theory?  
  (Partial answers in joint work with Ghilardi)
Seven questions

- How far can the decidability of pointlikes be stretched?
- How far do duality-theoretic methods reach beyond the regular?
- How does our semigroup-theoretic work fit with the category/duality approach?
- Is there a topos-theoretic interpretation of ‘logic on words’?
- What can be said about regular languages with model theory? (Partial answers in joint work with Ghilardi)
- Can the relationship with modal logic be made more tight?
Seven questions

- How far can the decidability of pointlikes be stretched?
- How far do duality-theoretic methods reach beyond the regular?
- How does our semigroup-theoretic work fit with the category/duality approach?
- Is there a topos-theoretic interpretation of ‘logic on words’?
- What can be said about regular languages with model theory? (Partial answers in joint work with Ghilardi)
- Can the relationship with modal logic be made more tight?
- Is anything I’ve said relevant for (formal) linguistics?
XKCD 208: Regular Expressions (https://xkcd.com/208/)