Büchi's Logical Characterisation of Regular Languages

Deepak D'Souza

Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

14 August 2018

Outline

Background

- Büchi's motivation: Decision procedure for deciding truth of first-order logic statements about natural numbers and their ordering. Eg.

$$\forall x \exists y (x < y).$$

- Used finite-state automata to give a decision procedure.
- By-product: a logical characterisation of regular languages.

Theorem (Büchi 1960)

L is regular iff L can be described in Monadic-Second Order Logic.

First-Order Logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x < y, \quad \exists x \varphi, \ \forall x \varphi, \ \neg, \wedge, \vee.$$

- Examples:
  1. $\forall x \exists y (x < y)$.

First-Order Logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x < y, \quad \exists x \varphi, \ \forall x \varphi, \ \neg, \wedge, \vee.$$

- Examples:
  1. $\forall x \exists y (x < y)$.
  2. $\forall x \exists y (y < x)$.

First-Order Logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x < y, \quad \exists x \varphi, \ \forall x \varphi, \ \neg, \wedge, \vee.$$

- Examples:
  1. $\forall x \exists y (x < y)$.
  2. $\forall x \exists y (y < x)$.
  3. $\exists x (\forall y (y \leq x))$.

First-Order Logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x < y, \quad \exists x \varphi, \ \forall x \varphi, \ \neg, \wedge, \vee.$$

- Examples:
  1. $\forall x \exists y (x < y)$.
  2. $\forall x \exists y (y < x)$.
  3. $\exists x (\forall y (y \leq x))$.
  4. $\exists x (\forall y (x \leq y))$.

First-Order Logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x < y, \quad \exists x\varphi, \; \forall x\varphi, \; \neg, \wedge, \vee.$$

- Examples:
  1. $\forall x \exists y (x < y)$.
  2. $\forall x \exists y (y < x)$.
  3. $\exists x (\forall y (y \leq x))$.
  4. $\exists x (\forall y (x \leq y))$.
  5. $\forall x \forall y ((x < y) \implies \exists z (x < z < y))$.

First-Order Logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x < y, \quad \exists x\varphi, \ \forall x\varphi, \ \neg, \wedge, \vee.$$

- Examples:
  1. $\forall x \exists y (x < y)$.
  2. $\forall x \exists y (y < x)$.
  3. $\exists x (\forall y (y \leq x))$.
  4. $\exists x (\forall y (x \leq y))$.
  5. $\forall x \forall y ((x < y) \implies \exists z (x < z < y))$.
- Sentences 1 and 4 are true while the others are not.

First-Order Logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x < y, \quad \exists x\varphi, \ \forall x\varphi, \ \neg, \wedge, \vee.$$

- Examples:
  1. $\forall x \exists y (x < y)$.
  2. $\forall x \exists y (y < x)$.
  3. $\exists x (\forall y (y \leq x))$.
  4. $\exists x (\forall y (x \leq y))$.
  5. $\forall x \forall y ((x < y) \implies \exists z (x < z < y))$.
- Sentences 1 and 4 are true while the others are not.
- Question: Is there an algorithm to decide if a given $\mathrm{FO}(\mathbb{N}, <)$ sentence is true or not?

Monadic Second-Order logic over alphabet A: $\mathrm{MSO}(A)$

- Interpreted over a string $w \in A^*$.

w	$=$	a	a	b	a	b	a	b	a	b
		0	1	2	3	4	5	6	7	8

- Domain is set of positions in w: $\{0, 1, 2, \ldots, |w| - 1\}$.
- "$<$" is interpretated as usual $<$ over numbers.
- What we can say in the logic:
  - $Q_a(x)$: "Position x is labelled a".
  - $x < y$: "Position x is strictly less than position y".
  - $\exists x \varphi$: "There exists a position x ..."
  - $\forall x \varphi$: "For all positions x ..."
  - $\exists X \varphi$: "There exists a set of positions X ..."
  - $\forall X \varphi$: "For all sets of positions X ..."
  - $x \in X$: "Position x belongs to the set of positions X".

Example $\mathrm{MSO}(\{a, b\})$ formulas

Consider the alphabet $\{a, b\}$.

What language do the sentences below define?

1. $\exists x(\neg \exists y(y < x) \wedge Q_a(x))$.
2. $\exists y(\neg \exists x(y < x) \wedge Q_b(y))$.
3. $\exists x \exists y \exists z(succ(x, y) \wedge succ(y, z) \wedge last(z) \wedge (Q_b(x))$.

Example $\mathrm{MSO}(\{a, b\})$ formulas

Consider the alphabet $\{a, b\}$.

What language do the sentences below define?

1. $\exists x(\neg \exists y(y < x) \wedge Q_a(x))$.
2. $\exists y(\neg \exists x(y < x) \wedge Q_b(y))$.
3. $\exists x \exists y \exists z(succ(x, y) \wedge succ(y, z) \wedge last(z) \wedge (Q_b(x))$.

Give sentences that describe the following languages:

1. Every a is immediately followed by a b.
2. Strings of odd length.

MSO sentence for strings of odd length

Language $L \subseteq \{a, b\}^*$ of strings of odd length.

	a	a	b	a	b	a	b	a	b
X_e	1	0	1	0	1	0	1	0	1
X_o	0	1	0	1	0	1	0	1	0

$$\begin{aligned}
\exists X_e \exists X_o (\exists x (x \in X_e) \wedge (\forall x ((x \in X_e \implies \neg x \in X_o) \wedge \\
(x \in X_o \implies \neg x \in X_e) \wedge \\
(x \in X_e \vee x \in X_o) \wedge \\
(zero(x) \implies x \in X_e) \wedge \\
(\forall y ((x \in X_e \wedge succ(x, y)) \implies y \in X_o)) \wedge \\
(\forall y ((x \in X_o \wedge succ(x, y)) \implies y \in X_e)) \wedge \\
(last(x) \implies x \in X_e)))).
\end{aligned}$$

First-Order Logic

- A First-Order Logic usually has a signature comprising the constants, and function/relation symbols. Eg. $(0, <, +)$.
- Terms are expressions built out of the constants, variables and function symbols. Eg. 0, $x + y$, $(x + y) + 0$. They are interpreted as elements of the domain of interpretation.
- Atomic formulas are obtained using the relation symbols on terms of the logic. Eg. $x < y$, $x = 0 + y$, $x + y < 0$.
- Formulas are obtained from atomic formulas using boolean operators, and existential quantification $(\exists x)$ and universal quantification $(\forall x)$. Eg. $\neg(x < y)$, $(x < 0) \wedge (x = y)$, $\exists x(\forall y(x < y) \wedge (z < x))$.

First-Order Logic

- Given a "structure" (i.e. a domain, a concrete interpretation for each constant and function/relation symbol) and an assignment for variables to values in the domain) to interpret the formulas in, each formula is either true or false.
- A formula is called a sentence if it has no free (unquantified) variables.

Second-Order Logic

- In Second-Order logic, one allows quantification over relations over the domain (not just elements of the domain). Eg:

$$\exists R^{(2)}(R^{(2)}(x, y) \implies x < y).$$

- In Monadic second-order logic, one allows quantification over monadic relations (i.e. relations of arity one, or equivalently, subsets of the domain). Eg:

$$\exists X(x \in X \implies 0 < x).$$

Formal Semantics of MSO

- An interpretation for the logic will be a pair $(w, \mathbb{I})$ where $w \in A^*$ and $\mathbb{I}$ is an assignment of "individual" variables to a position in w, and "set" variables to a set of positions in w.

$$\mathbb{I} : Var \to pos(w) \cup 2^{pos(w)}.$$

- $\mathbb{I}[i/x]$ denotes the assignment which maps x to i and agrees with $\mathbb{I}$ on all other individual and set variables.
- Similarly for $\mathbb{I}[S/X]$.

Formal Semantics of MSO

The satisfaction relation $w, \mathbb{I} \models \varphi$ is given by:

$$
\begin{array}{lll}
w, \mathbb{I} \models Q_a(x) & \text{iff} & w(\mathbb{I}(x)) = a \\
w, \mathbb{I} \models x < y & \text{iff} & \mathbb{I}(x) < \mathbb{I}(y) \\
w, \mathbb{I} \models x \in X & \text{iff} & \mathbb{I}(x) \in \mathbb{I}(X) \\
w, \mathbb{I} \models \neg\varphi & \text{iff} & w, \mathbb{I} \not\models \varphi \\
w, \mathbb{I} \models \varphi \vee \varphi' & \text{iff} & w, \mathbb{I} \models \varphi \text{ or } w, \mathbb{I} \models \varphi' \\
w, \mathbb{I} \models \exists x \varphi & \text{iff} & \text{exists } i \in pos(w) \text{ s.t. } w, \mathbb{I}[i/x] \models \varphi \\
w, \mathbb{I} \models \exists X \varphi & \text{iff} & \text{exists } S \subseteq pos(w) \text{s.t } w, \mathbb{I}[S/X] \models \varphi
\end{array}
$$

Example to illustrate semantics

Consider the word $w = aaba$ and the formula

$$\exists x (Q_a(x) \wedge \neg \exists y (y < x)).$$

MSO sentences

- A sentence is a formula with no free variables.
- For example $\exists X(y \in X \implies 0 < y)$ is not a sentence since y occurs free.
- $\exists X(0 \in X \implies \exists y(0 < y \wedge y \in X))$ is a sentence.
- If φ is a sentence, then we don't need an interpretation for variables to say if φ is true or false of a given word w:

$$w \models \varphi.$$

- For a sentence φ, we can define the language of words that satisfy φ:

$$L(\varphi) = \{w \in A^* \mid w \models \varphi\}.$$

Languages definable by MSO

- We say that a language $L \subseteq A^*$ is definable in $\mathrm{MSO}(A)$ if there is a sentence φ in $\mathrm{MSO}(A)$ such that $L(\varphi) = L$.

Theorem (Büchi 1960 (also Elgot '61 and Traktenbrot 62))

$L \subseteq A^*$ is regular iff L is definable in $\mathrm{MSO}(A)$.

From automata to MSO sentence

- Let $L \subseteq A^*$ be regular. Let $\mathcal{A} = (Q, s, \delta, F)$ be a DFA for L.
- To show L is definable in $\mathrm{MSO}(A)$.
- Idea: Construct a sentence $\varphi_{\mathcal{A}}$ describing an accepting run of A on a given word.
  That is: $\varphi_{\mathcal{A}}$ is true over a given word w precisely when $\mathcal{A}$ has an accepting run on w.

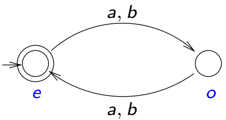
Let $Q = \{q_1, \ldots, q_n\}$, with $q_1 = s$.
Define $\varphi_{\mathcal{A}}$ as

$$\begin{aligned}\exists X_1 \cdots \exists X_n (\forall x(\quad & (\textstyle\bigwedge_{i \neq j}(x \in X_i \implies \neg x \in X_j) \wedge \bigvee_i x \in X_i) \wedge \\ & (zero(x) \implies x \in X_1) \wedge \\ & (\textstyle\bigwedge_{a \in A,\ i,j \in \{1,\ldots,n\},\ \delta(q_i,a)=q_j}((x \in X_i \wedge Q_a(x) \wedge \neg last(x)) \implies \\ & \qquad \exists y(succ(x,y) \wedge y \in X_j))) \wedge \\ & (last(x) \implies \textstyle\bigvee_{a \in A,\ \delta(q_i,a) \in F}(Q_a(x) \wedge x \in X_i)))).\end{aligned}$$

Example

Consider language $L \subseteq \{a, b\}^*$ of strings of even length.

DFA $\mathcal{A}$ for L:



	a	a	b	a	b	a	b	a	b
X_e	1	0	1	0	1	0	1	0	1
X_o	0	1	0	1	0	1	0	1	0

$\varphi_{\mathcal{A}}$:

$$\begin{aligned}
\exists X_e \exists X_o (\forall x(\quad & (x \in X_e \implies \neg x \in X_o) \wedge (x \in X_o \implies \neg x \in X_e) \wedge \\
& (x \in X_e \vee x \in X_o) \wedge \\
& (zero(x) \implies x \in X_e) \wedge \\
& ((x \in X_e \wedge Q_a(x) \wedge \neg last(x)) \implies \exists y(succ(x,y) \wedge y \in X_o)) \wedge \\
& ((x \in X_e \wedge Q_b(x) \wedge \neg last(x)) \implies \exists y(succ(x,y) \wedge y \in X_o)) \wedge \\
& ((x \in X_o \wedge Q_a(x) \wedge \neg last(x)) \implies \exists y(succ(x,y) \wedge y \in X_e)) \wedge \\
& ((x \in X_o \wedge Q_b(x) \wedge \neg last(x)) \implies \exists y(succ(x,y) \wedge y \in X_e)) \wedge \\
& (last(x) \implies ((Q_a(x) \wedge x \in X_o) \vee (Q_b(x) \wedge x \in X_o))))).
\end{aligned}$$

From MSO sentence to automaton

- Idea: Inductively describe the language of extended models of a given MSO formula φ by an automaton $\mathcal{A}_\varphi$.
- Extended models wrt set of first-order and second-order variables $T = \{x_1, \ldots, x_m, X_1, \ldots, X_n\}$: $(w, \mathbb{I})$
- Can be represented as a word over $A \times \{0,1\}^{m+n}$.

	a	a	b	a	b	a	b	a	b
x_1	0	1	0	0	0	0	0	0	0
x_2	0	0	0	0	1	0	0	0	0
X_1	1	0	1	0	1	0	1	0	1
X_2	0	1	0	1	0	1	0	1	0

- For example, the extended word above satisfies the formula

$$Q_a(x_1) \wedge (x_2 \in X_1).$$

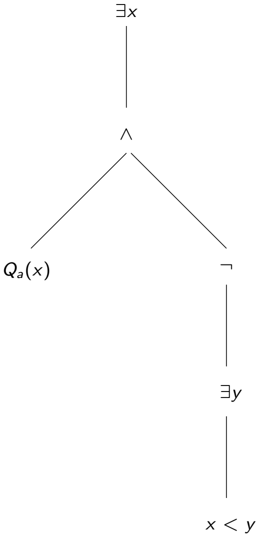
Inductive construction of $\mathcal{A}_{\varphi}^{T}$.

- If φ is a formula whose free variables are in T, then we have the notion of whether $w' \models \varphi$ based on whether the $(w, \mathbb{I})$ encoded by w' satisfies φ or not.
- Let the set of valid extended words wrt T be $valid^T(A)$.
- We can define an automaton $\mathcal{A}_{val}^{T}$ which accepts this set.
- Claim: with every formula φ in $\mathrm{MSO}(A)$, and any finite set of variables T containing at least the free variables of φ, we can construct an automaton $\mathcal{A}_{\varphi}^{T}$ which accepts the language $L^T(\varphi)$.
- Proof: by induction on structure of φ.

$$Q_a(x), \; x < y, \; x \in Y, \; \neg\varphi, \; \varphi \vee \psi, \; \exists x \varphi, \; \exists X \varphi.$$

Example formula

$$\exists x(Q_a(x) \wedge \neg\exists y(x < y))$$



Back to First-Order logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x < y, \quad \exists x\varphi, \ \forall x\varphi, \ \neg, \wedge, \vee.$$

- Examples:
  1. $\forall x \exists y (x < y)$.
  2. $\forall x \exists y (y < x)$.
  3. $\exists x (\forall y (y \leq x))$.
  4. $\forall x \forall y ((x < y) \implies \exists z (x < z < y))$.
- Question: Is there an algorithm to decide if a given $\mathrm{FO}(\mathbb{N}, <)$ sentence is true or not?

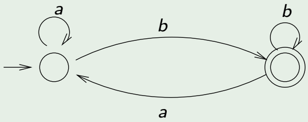
Büchi's decision procedure for $\mathrm{MSO}(\mathbb{N}, <)$

- Büchi considered finite automata over infinite strings (so called ω-automata).
- An infinite word is accepted if there is a run of the automaton on it that visits a final state inifinitely often.
- Büchi showed that ω-automata have similar properties to classical automata: are closed under boolean operations, projection, and can be effectively checked for emptiness.
- MSO characterisation works similarly for ω-automata as well.
- Given a sentence φ in $\mathrm{MSO}(\mathbb{N}, <)$ we can now view it as an $\mathrm{MSO}(\{a\})$ sentence.
- Construct an ω-automaton $\mathcal{A}_\varphi$ that accepts precisely the words that satisfy φ.
- Check if $L(\mathcal{A}_\varphi)$ is non-empty.
- If non-empty say "Yes, φ is true", else say "No, it is not true."

Büchi automata

- Finite state automata that run over infinite words.
- How do we accept an *infinite* word? Acceptance mechanism proposed by Büchi: see if run visits a final state infinitely often.

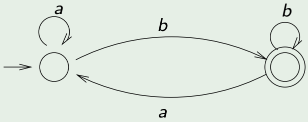
Büchi automaton for infinitely many b's



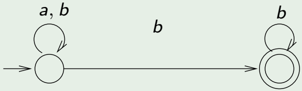
Büchi automata

- Finite state automata that run over infinite words.
- How do we accept an *infinite* word? Acceptance mechanism proposed by Büchi: see if run visits a final state infinitely often.

Büchi automaton for infinitely many b's



Büchi automaton for finitely many a's



Checking non-emptiness of Büchi automata

- Büchi automata have similar closure properties to classical FSA's: closed under union, intersection, and complement.
- Non-emptiness is efficiently decidable: Look for a path from initial state to a final state that can reach itself.
- Can be checked efficiently: in time linear in the number of states and transitions of automaton.

Checking non-emptiness

Summary

- We saw another characterisation of the class of regular languages, this time via logic:

Theorem (Büchi 1960)

$L \subseteq A^*$ is regular iff L is definable in $\mathrm{MSO}(A)$.

- We saw an application of automata theory to solve a decision procedure in logic:

Theorem (Büchi 1960)

The Monadic Second-Order (MSO) logic of $(\mathbb{N}, <)$ is decidable.

Related seminar topics

- Büchi automata, closure properties, decision procedures.
- Characterization of FO-definable langauges via counter-free automata.