

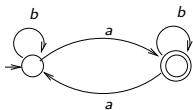
Overview of E0 222: Automata and Computability

Deepak D'Souza

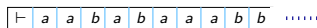
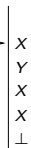
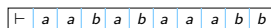
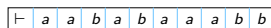
Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

August 6, 2015

Different Kinds of “Automata” or “State Machines”



- Finite-State Automata
- Pushdown Automata
- Turing Machines



Kind of results we study in Automata Theory

- Expressive power of the models in terms of the class of languages they define.
 - Characterisations of this class of languages
 - Myhill-Nerode theorem.
 - Büchi's logical characterisation.
 - Necessary conditions these classes satisfy
 - Pumping Lemma and ultimate periodicity (for Regular/CFL).
 - Parikh's Theorem (for Context-Free Languages).
- Decision procedures
 - Emptiness problem
 - Language inclusion problem
 - Configuration reachability problem.
- Computability (most compelling notion of computable function is via Turing Machines), Rice's Theorem.

Why study automata theory?

Corner stone of many subjects in CS:

- ① Compilers
 - Lexical analysis, parsing, regular expression search
- ② Digital circuits (state minimization, analysis).
- ③ Complexity Theory (algorithmic hardness of problems)
- ④ **Mathematical Logic**
 - Decision procedures for logical problems.
- ⑤ **Formal Verification**
 - Configuration reachability
 - Is $L(\mathcal{A}) \subseteq L(\mathcal{B})$?

Uses in Verification

- 1 System models are natural extensions of automata models
 - Programs with no dynamic memory allocation, no procedures = Finite State systems.
 - No dynamic memory allocation = Pushdown systems.
 - General program = Turing machine.
 - Programs with integer variables = Counter machines.

Decision procedures for emptiness, configuration reachability, etc, directly translate to decision procedures for programs.

- 2 To solve “model-checking” problem for logics that talk about infinite behaviour.

Uses in Logic

- Obtain decision procedure for satisfiability of a logic by translating a formula to an automaton and checking emptiness.
- Argue undecidability/incompleteness of a proof system.

What this course is about

What we study

- Connections between Logic and Automata
 - Büchi's logical characterization of regular languages
 - Decision procedures for logic (Büchi, Presburger logic, Godel).
- Pushdown Systems
 - Parikh's theorem on semi-linearity of CFL's
 - Reachability in pushdown systems
 - Deterministic PDA's and complementation
 - Visibly Pushdown Automata
- Automata on infinite words
- Automata on Trees

Büchi's logical characterisation of automata

- Describe properties of strings in a logical language
Eg. "For all positions x in a word which are labelled a , there is a later position labelled b "

$$\forall x(Q_a(x) \Rightarrow \exists y(y > x \ \& \ Q_b(y))).$$

- Büchi's result:
A language is regular iff it is definable by a sentence in this logic.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

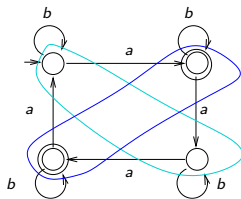
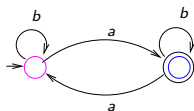
- Examples:
 - 1 $\forall x\exists y(x < y)$.
 - 2 $\forall x\exists y(y < x)$.
 - 3 $\exists x(\forall y(y \leq x))$.
 - 4 $\forall x\forall y((x < y) \implies \exists z(x < z < y))$.
- Question: Is there an **algorithm** to decide if a given $\text{FO}(\mathbb{N}, <)$ sentence is true or not?

Büchi used automata to give such an algorithm.

Myhill-Nerode Theorem

Myhill-Nerode Theorem:

Every regular language has a *canonical* DFA accepting it.



Some consequences:

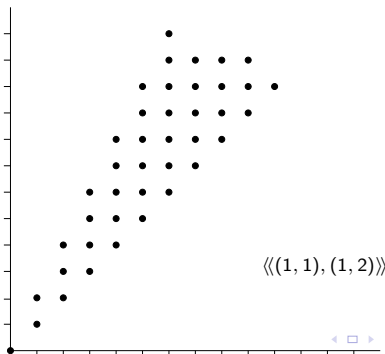
- Any DFA for L is a *refinement* of its canonical DFA.
- “minimal” DFA’s for L are isomorphic.

Parikh's Theorem for CFL's

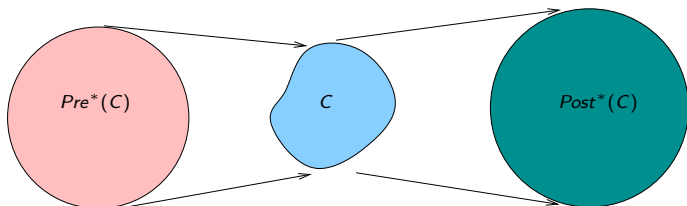
$\psi(w)$: "Letter-count" of a string w :

$$\text{Eg : } \psi(aabab) = (3, 2).$$

*If L is a context-free language, then $\psi(L)$ is semi-linear
(Every CFL is letter-equivalent to a regular language).*



Reachable configurations of a Pushdown automaton



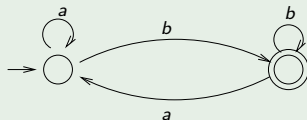
The set of reachable configurations of a Pushdown automaton is regular.

Useful for program analysis and verification of pushdown systems.

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**.

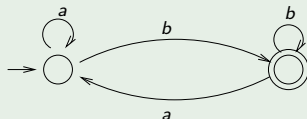
Büchi automaton for infinitely many b 's



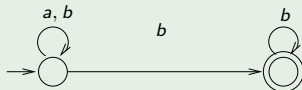
Büchi automata

- 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



Gödel's Incompleteness result

There cannot be a sound and complete proof system for first-order arithmetic.

What we can say in $\text{FO}(\mathbb{N}, +, \cdot)$

- “Every number has a successor”

$$\forall n \exists m (m = n + 1).$$

- “Every number has a predecessor”

$$\forall n \exists m (n = m + 1).$$

- “There are only finitely many primes”

$$\exists n \forall p (\text{prime}(p) \implies p < n).$$

- “There are infinitely many primes”

$$\forall n \exists p (\text{prime}(p) \ \& \ p > n).$$

Gödel's Incompleteness result

There cannot be a sound and complete proof system for first-order arithmetic.

Formal language-theoretic proof: $\text{Th}(\mathbb{N}, +, \cdot)$ is not even recursively enumerable.

Course details

- Weightage: 40% assignments + seminar, 20% midsem exam, 40% final exam.
- Assignments to be done on your own.
- Dishonesty Policy: Any instance of copying in an assignment will fetch you a 0 in that assignment + one grade reduction.
- Seminar (in pairs) can be chosen from list on course webpage or your own topic.
- Course webpage:
www.csa.iisc.ernet.in/~deepakd/atc-2015/
- Teaching assistants for the course: To be announced.
- Those interested in crediting/auditing please send me an email so that I can add you to the course mailing list.