Finite automata and formal languages (DIT322, TMV028)

Nils Anders Danielsson

2020-03-09

Today

- A summary of the course.

Proofs and induction

Proofs

Throughout the course we have talked about:

- How to attack a problem.
- How to prove something.

Proofs

Some examples:

- One way to prove $(p \Rightarrow q) \Rightarrow r$ is to assume that you are given a method for proving q given p, and use that to prove r.
- You can prove $\neg p$ by finding a counterexample to p, i.e. showing that p leads to a contradiction.

Induction

- Mathematical induction.
- Complete induction.
- Mutual induction.
- Inductively defined sets:
  - Primitive recursion.
  - Structural induction.
- Inductively defined subsets.

One way to structure a proof by induction

If you want to prove something by induction on the structure of a list of natural numbers:

- State what you want to prove, and how you intend to prove it:

  Let us prove $\forall xs \in List(\mathbb{N}).P(xs)$, where $P(xs) = ...$, by induction on the structure of the list.

- Prove each case:

  We have two cases:

  - $P(\text{nil})$ holds because...
  - Given $x \in \mathbb{N}$, $xs \in List(\mathbb{N})$ and $P(xs)$, we can prove $P(\text{cons}(x, xs))$ by...

Regular languages

Automata

Terminology, notation:

- Alphabets.
- Strings.
- Languages.
- Concatenation.
- Exponentiation.
- Kleene star.
- ...

DFAs

- Deterministic.
- 5-tuples.
- Transition diagrams.
- Transition tables.
- Transition functions for strings ($\hat{\delta}$).
- The language of a DFA.

DFAs

States can be:

- Accessible.
- Equivalent to each other.
- Distinguishable from each other.

NFAs

- Nondeterministic.
- 5-tuples.
- Transition diagrams.
- Transition tables.
- Transition functions for strings ($\hat{\delta}$).
- The language of an NFA.

DFAs and NFAs

- DFAs can easily be turned into NFAs.
- NFAs can be turned into DFAs:
  - The subset construction.
  - Optimisation: Skip inaccessible states.
  - Potential problem: Exponential blowup.

ε-NFAs

- Nondeterministic and with ε-transitions.
- 5-tuples.
- Transition diagrams.
- Transition tables.
- ε-closure.
- Transition functions for strings ($\hat{\delta}$).
- The language of an ε-NFA.

DFAs, NFAs and ε-NFAs

- NFAs can easily be turned into ε-NFAs.
- ε-NFAs can be turned into DFAs:
  - The subset construction with ε-closure.
  - Optimisation: Skip inaccessible states.

Regular expressions

- Syntax.
- The language of a regular expression.
- Proving that two regular expressions denote the same language:
    - Using known equalities and equational reasoning.
    - Using known inequalities, inequational reasoning and antisymmetry.
    - By converting to DFAs and proving that the DFAs denote the same language.

ε-NFAs and regular expressions

Translating regular expressions to equivalent ε-NFAs:

- Easy.

Translating ε-NFAs to equivalent regular expressions:

- By eliminating states.
- By using Arden's lemma:
  The equation $X = AX \cup B$ has the least solution $X = A^*B$.

Regular languages

- Definition in terms of DFAs, NFAs, ε-NFAs or regular expressions.
- The pumping lemma.
- Closure properties:
  - Union.
  - Concatenation.
  - Kleene star/plus.
  - Intersection (product construction).
  - Complement.

The pumping lemma

For every alphabet Σ and *regular* language $L \subseteq \Sigma^*$.

$$
\begin{aligned}
&\exists m \in \mathbb{N}. \\
&\quad \forall w \in L.\ w \geq m \Rightarrow \\
&\qquad \exists t, u, v \in \Sigma^*. \\
&\qquad\quad w = tuv \wedge u \neq \varepsilon \wedge tu \leq m \wedge \\
&\qquad\quad \forall n \in \mathbb{N}.\ tu^n v \in L
\end{aligned}
$$

- The pumping lemma can be used to prove that a language is not regular.

The pumping lemma

For every alphabet Σ and *regular* language $L \subseteq \Sigma^*$.

$$
\begin{array}{l}
\exists m \in \mathbb{N}. \\
\quad \forall w \in L.\ |w| \geq m \Rightarrow \\
\quad\quad \exists t, u, v \in \Sigma^*. \\
\quad\quad\quad w = tuv \wedge u \neq \varepsilon \wedge |tu| \leq m \wedge \\
\quad\quad\quad \forall n \in \mathbb{N}.\ tu^n v \in L
\end{array}
$$

- The last five lines are a necessary, but not a sufficient, condition for being regular:
  there is at least one non-regular language for which they hold.

The pumping lemma

For every alphabet Σ and *regular* language $L \subseteq \Sigma^*$.

$$\exists m \in \mathbb{N}.$$
$$\forall w \in L.\ |w| \geq m \Rightarrow$$
$$\exists t, u, v \in \Sigma^*.$$
$$w = tuv \wedge u \neq \varepsilon \wedge |tu| \leq m \wedge$$
$$\forall n \in \mathbb{N}.\ tu^n v \in L$$

- Do not give "the pumping lemma holds, so the language is regular" as an exam answer.

Regular languages

Algorithms:

- Conversions between different formats.
- Is the language empty?
- Is a given string a member of the language?
- Are two regular languages equal?
  - Are two states equivalent?
- Minimisation of DFAs.

Context-free languages

Context-free grammars

4-tuples:

- Nonterminals.
- Terminals.
- Productions.
- Start symbol.

Context-free grammars

The language of a CFG can be defined in several equivalent ways:

- Derivations.
- Leftmost (rightmost) derivations.
- Recursive inference.
- Parse trees.

Context-free grammars

- Ambiguous grammars.
- Associativity.
- Precedence.

Context-free grammars

- Chomsky normal form:
  $A \to a$ or $A \to BC$.
- BIN, DEL, UNIT, TERM.

Pushdown automata

- A kind of finite automaton with a single stack.
- 7-tuples.
- Instantaneous descriptions.
- Transition relation ($\vdash$).
- The languages of a PDA P: $L(P)$ and $N(P)$.

Context-free languages

- Definition in terms of CFGs or PDAs, which define the same class of languages.
- The pumping lemma.
- Closure properties:
  - Substitution.
  - Union.
  - Concatenation.
  - Kleene star/plus.
  - Homomorphism.
  - Intersection with a regular language.

Only 32% answered the following quiz question correctly. Try to use closure properties.

Which of the following languages, if any, are context-free?

1. $\{uuvv \mid u \in \{0\}^+, v \in \{1\}^+\} \cup \{uvvu \mid u \in \{0\}^+, v \in \{1\}^+\}$
2. $\{uuvv \mid u \in \{0\}^+, v \in \{1\}^+\} \cap \{uvvu \mid u \in \{0\}^+, v \in \{1\}^+\}$
3. $\{ssttuvvu \mid s, u \in \{0\}^+, t, v \in \{1\}^+\}$
4. $\{uuvvuvvu \mid u \in \{0\}^+, v \in \{1\}^+\}$
5. $\{(uvvu)^n \mid u \in \{0\}^+, v \in \{1\}^+, n \in \mathbb{N}\}$
6. $\{(ab)^m c^{2n} (ab)^m \mid m, n \in \mathbb{N}\}$
7. $\{uvu \mid u \in \{0, 1\}^*, v \in \{2, 3\}^*\}$

Context-free languages

Algorithms:

- Generating symbols.
- Is the language empty?
- Nullable symbols.
- Is the empty string a member of the language?
- Is a nonempty string a member of the language?
  - The CYK algorithm.

Recursive or recursively enumerable languages

Turing machines

- A kind of simple computer.
- Read/write head, unbounded tape, finite set of states.
- 7-tuples.
- Instantaneous descriptions.
- Transition relation ($\vdash$).
- The language of a TM.
- Halting.
- Undecidable problems.

Recursive languages

- Definition in terms of (halting) TMs, or lambda expressions, or recursive functions, or...
- The Church-Turing thesis.

Recursively enumerable languages

- Definition in terms of TMs, or lambda expressions, or recursive functions, or...

A hierarchy

A hierarchy of languages over the alphabet Σ (if $|\Sigma| \geq 2$):

$$
\begin{array}{ll}
\text{Finite} & \subsetneq \\
\text{Regular} & \subsetneq \\
\text{Context-free} & \subsetneq \\
\text{Recursive} & \subsetneq \\
\text{Recursively enumerable} & \subsetneq \\
\wp(\Sigma^*) &
\end{array}
$$

A hierarchy

A hierarchy of languages over the alphabet Σ (if $|\Sigma| \geq 2$):

Finite	$\subsetneq$
Regular	$\subsetneq$
Context-free	$\subsetneq$
Recursive	$\subsetneq$
Recursively enumerable	$\subsetneq$
$\wp(\Sigma^*)$	

It might not be a good idea to give "the language is context-free, but not regular" as an exam answer.

Discuss what you have learnt in this course.

- What has been most interesting?
- What has been least interesting?
- What would you like to know more about?
- ...

Coming up

- Next lecture:
  - Old exam questions.
- Deadline for the seventh assignment:
  2020-03-13, 23:59.
  (Only one exercise, five points.)