Finite automata and formal languages (DIT322, TMV028)

Nils Anders Danielsson,
partly based on slides by Ana Bove

2020-01-27

Today

- Structural induction.
- Some concepts from automata theory.
- Inductively defined subsets
  (if we have enough time).

The last quiz from the previous lecture

Discuss how you would prove
$\forall n \in \mathbb{N}. even(n) = nots(n, \mathsf{true})$.

$$nots \in \mathbb{N} \times Bool \to Bool$$
$$nots(\mathsf{zero}, \quad b) = b$$
$$nots(\mathsf{suc}(n), b) = nots(n, not(b))$$

$$odd, even \in \mathbb{N} \to Bool$$

$$odd(\mathsf{zero}) \quad = \mathsf{false}$$
$$odd(\mathsf{suc}(n)) = even(n)$$

$$even(\mathsf{zero}) \quad = \mathsf{true}$$
$$even(\mathsf{suc}(n)) = odd(n)$$

The last quiz from the previous lecture

One possibility is to use mathematical induction to prove $\forall n \in \mathbb{N}. P(n)$, with

$$\begin{aligned} P(n) := {} & even(n) = nots(n, \mathsf{true}) \land \\ & odd(n) = nots(n, \mathsf{false}). \end{aligned}$$

Structural induction

Structural induction

- For a given inductively defined set we have a corresponding induction principle.
- Example:

$$\frac{}{\mathsf{zero} \in \mathbb{N}} \qquad \frac{n \in \mathbb{N}}{\mathsf{suc}(n) \in \mathbb{N}}$$

In order to prove $\forall n \in \mathbb{N}.\ P(n)$:

  - Prove $P(\mathsf{zero})$.
  - For all $n \in \mathbb{N}$, prove that $P(n)$ implies $P(\mathsf{suc}(n))$.

Structural induction

- For a given inductively defined set we have a corresponding induction principle.
- Example:

$$\frac{}{\mathsf{true} \in Bool} \qquad \frac{}{\mathsf{false} \in Bool}$$

In order to prove $\forall b \in Bool.\ P(b)$:

- Prove $P(\mathsf{true})$.
- Prove $P(\mathsf{false})$.

Structural induction

- For a given inductively defined set we have a corresponding induction principle.
- Example:

$$\frac{}{\textsf{nil} \in List(A)} \qquad \frac{x \in A \qquad xs \in List(A)}{\textsf{cons}(x, xs) \in List(A)}$$

In order to prove $\forall xs \in List(A).\ P(xs)$:

- Prove $P(\textsf{nil})$.
- For all $x \in A$ and $xs \in List(A)$, prove that $P(xs)$ implies $P(\textsf{cons}(x, xs))$.

Pattern

- An inductively defined set:

$$\ldots \qquad \frac{x \in A \qquad \ldots \qquad d \in D(A)}{\mathsf{c}(x, \ldots, d) \in D(A)} \qquad \ldots$$

  Note that x is a non-recursive argument, and that d is recursive.
- In order to prove $\forall d \in D(A).\ P(d)$:
  - $\vdots$
  - For all $x \in A$, ..., $d \in D(A)$, prove that ... and $P(d)$ imply $P(\mathsf{c}(x, \ldots, d))$.
  - $\vdots$

  One inductive hypothesis for each *recursive* argument.

What is the induction principle for

$$\frac{n \in \mathbb{N}}{\mathsf{leaf}(n) \in \mathit{Tree}} \qquad \frac{l, r \in \mathit{Tree}}{\mathsf{node}(l, r) \in \mathit{Tree}}?$$

1. $(\forall n \in \mathbb{N}.\ P(\mathsf{leaf}(n))) \land$
   $(\forall l, r \in \mathit{Tree}.\ P(l) \land P(r) \Rightarrow P(\mathsf{node}(l, r)))$.
2. $(\forall n \in \mathbb{N}.\ P(\mathsf{leaf}(n))) \land$
   $(\forall l, r \in \mathit{Tree}.\ P(l) \land P(r) \Rightarrow P(\mathsf{node}(l, r))) \Rightarrow$
   $(\forall t \in \mathit{Tree}.\ P(t))$.
3. $(\forall n \in \mathbb{N}.\ P(\mathsf{leaf}(n))) \land$
   $(\forall t \in \mathit{Tree}.\ P(t) \Rightarrow P(\mathsf{node}(t, t))) \Rightarrow$
   $(\forall t \in \mathit{Tree}.\ P(t))$.

Some functions

Recall from last lecture:

$$
\begin{aligned}
&\mathit{length} \in \mathit{List}(A) \to \mathbb{N} \\
&\mathit{length}(\mathsf{nil}) = \mathsf{zero} \\
&\mathit{length}(\mathsf{cons}(x, xs)) = \mathsf{suc}(\mathit{length}(xs))
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{append} \in \mathit{List}(A) \times \mathit{List}(A) \to \mathit{List}(A) \\
&\mathit{append}(\mathsf{nil}, \; ys) = ys \\
&\mathit{append}(\mathsf{cons}(x, xs), \; ys) = \mathsf{cons}(x, \mathit{append}(xs, ys))
\end{aligned}
$$

Lemma

$\forall xs, ys \in List(A).$
$length(append(xs, ys)) = length(xs) + length(ys).$

Proof.

Let us prove the property

$$P(xs) := \forall ys \in List(A).\\ length(append(xs, ys)) =\\ length(xs) + length(ys)$$

by induction on the structure of the list.

Lemma

$\forall xs, ys \in List(A).$
$length(append(xs, ys)) = length(xs) + length(ys).$

Proof.

Case nil:

$$
\begin{aligned}
& length(append(\mathsf{nil}, ys)) = \\
& length(ys) = \\
& 0 + length(ys) = \\
& length(\mathsf{nil}) + length(ys)
\end{aligned}
$$

Lemma

$\forall xs, ys \in List(A).$
$length(append(xs, ys)) = length(xs) + length(ys).$

Proof.

Case $\mathsf{cons}(x, xs)$:

$$
\begin{aligned}
& length(append(\mathsf{cons}(x, xs), ys)) = \\
& length(\mathsf{cons}(x, append(xs, ys))) = \\
& 1 + length(append(xs, ys)) = \{\text{By the IH, } P(xs).\} \\
& 1 + (length(xs) + length(ys)) = \\
& (1 + length(xs)) + length(ys) = \\
& length(\mathsf{cons}(x, xs)) + length(ys)
\end{aligned}
$$

Prove $\forall xs \in List(A).append(xs, \mathsf{nil}) = xs$ and $\forall xs \in List(A).append(\mathsf{nil}, xs) = xs$. Which proof is “easiest”?

1. The first.
2. The second.

Induction/recursion

- Inductively defined sets:
  inference rules with constructors.
- Recursion (primitive recursion):
  recursive calls only for recursive arguments
  $(f(\mathsf{c}(x, d)) = ...f(d)...)$.
- Structural induction:
  inductive hypotheses for recursive arguments
  $(P(d) \Rightarrow P(\mathsf{c}(x, d)))$.

Some concepts from automata theory

Alphabets and strings

- An *alphabet* is a finite, nonempty set.
  - $\{ a, b, c, ..., z \}$.
  - $\{ 0, 1, ..., 9 \}$.
- A *string (or word) over the alphabet* Σ is a member of $List(\Sigma)$.

Notation

- Σ^* instead of $List(\Sigma)$.
- ε instead of nil or $[\,]$.
- aw instead of cons(a, w).
- a instead of cons$(a,$ nil$)$ or $[a]$.
- abc instead of $[a, b, c]$.
- uv instead of $append(u, v)$.
- $|w|$ instead of $length(w)$.
- Σ^+: Nonempty strings, $\{\, w \in \Sigma^* \mid w \neq \varepsilon \,\}$.

Exponentiation

- Σ^n: Strings of length n, $\{ w \in \Sigma^* \mid |w| = n \}$.
- Alternative definition of $\Sigma^n \subseteq \Sigma^*$:

$$\begin{aligned} \Sigma^0 &= \{ \varepsilon \} \\ \Sigma^{n+1} &= \{ aw \mid a \in \Sigma, w \in \Sigma^n \} \end{aligned}$$

- Similarly, $-^n \in \Sigma^* \to \Sigma^*$:

$$\begin{aligned} w^0 &= \varepsilon \\ w^{n+1} &= ww^n \end{aligned}$$

Which of the following propositions are valid? The alphabet is $\{ a, b, c \}$.

1. $|uv| = |u| + |v|$.
2. $|uv| = |u||v|$.
3. $|w^n| = n$.
4. $uv = vu$.
5. $\varepsilon v = v\varepsilon$.

Languages

A *language* over an alphabet Σ is a set $L \subseteq \Sigma^*$.

- Typical programming languages.
- Typical natural languages?
  (Are they well-defined?)
- Other examples, for instance the even natural numbers expressed in binary notation, which is a language over $\{ 0, 1 \}$.

Operations

- Concatenation: $LM = \{ uv \mid u \in L, v \in M \}$.
- Exponentiation:

$$L^0 = \{ \varepsilon \}$$
$$L^{n+1} = LL^n$$

- The Kleene star $L^* = \bigcup_{n \in \mathbb{N}} L^n$.
- These definitions are consistent with previous ones for alphabets:
  - $\Sigma^n = \{ w \in \Sigma^* \mid |w| = 1 \}^n$.
  - $\Sigma^* = \{ w \in \Sigma^* \mid |w| = 1 \}^*$.

Which of the following propositions are valid? The alphabet is $\{ 0, 1, 2 \}$.

1. $\forall w \in L^n.\ |w| = n$.
2. $LM = ML$.
3. $L(M \cup N) = LM \cup LN$.
4. $LM \cap LN \subseteq L(M \cap N)$.
5. $L^* L^* \subseteq L^*$.

Inductively defined subsets

Inductively defined subsets

- One can define subsets of (say) Σ^* inductively.
- For instance, for $L \subseteq \Sigma^*$ we can define $L^* \subseteq \Sigma^*$ inductively:

$$\frac{}{\varepsilon \in L^*} \qquad \frac{u \in L \qquad v \in L^*}{uv \in L^*}$$

- Note that there are no constructors.

Inductively defined subsets

- What about recursion?

$$f \in L^* \to Bool$$
$$f(\varepsilon) = \text{false}$$
$$f(uv) = not(f(v))$$

- If $\varepsilon \in L$, do we have

$$f(\varepsilon) = f(\varepsilon\varepsilon) = not(f(\varepsilon))?$$

Inductively defined subsets

- Induction works
  (assuming "proof irrelevance").
- $P(\varepsilon) \wedge (\forall u \in L, v \in L^*.\ P(v) \Rightarrow P(uv)) \Rightarrow \forall w \in L^*.\ P(w)$.

Today

- Structural induction.
- Some concepts from automata theory.
- Inductively defined subsets.

Next lecture

- Deterministic finite automata.

- Deadline for the next quiz: 2020-01-28, 8:00.
- Deadline for the first assignment: 2020-02-02, 23:59.