CSC B36 Additional Notes

structural induction for regular expressions

⋆ Introduction

Recall that the set $\mathcal{RE}$ of all regular expressions (in this course we abbreviate *regular expression* to *regex*) over some alphabet Σ can be defined by structural induction as follows.

> Let $\mathcal{RE}$ be the smallest set such that
> Basis: $\emptyset, \epsilon \in \mathcal{RE}$, and $c \in \mathcal{RE}$ for each $c \in \Sigma$.
> Induction Step: If $S, T \in \mathcal{RE}$, then $(S + T), (ST), S^* \in \mathcal{RE}$.

Since $\mathcal{RE}$ is defined by structural induction, for any predicate $P(R)$ on regexes, we can (try to) prove that $P(R)$ holds for every regex R with a structural induction proof on the structure of $\mathcal{RE}$.

Example:

Suppose we want to prove the following.

> Every regex without the symbol $*$ denotes a finite language.

Put another way, for regex R, suppose we define a predicate $P(R)$ as follows.

> $P(R)$: If R does not contain the symbol $*$, then $\mathcal{L}(R)$ is finite.

Then we want to prove that $P(R)$ holds for all regexes $R \in \mathcal{RE}$.

Aside: $P(R)$ is an implication. So it is (vacuously) true whenever R contains the symbol $*$.
If R does not contain the symbol $*$, then $P(R)$ holds iff $\mathcal{L}(R)$ is finite.

Here is our proof.

Basis: There are 3 cases.

(i) Let $R = \emptyset$.
Then R does not contain the symbol $*$ and $\mathcal{L}(R) = \emptyset$, which is finite.
So $P(R)$ holds as wanted.

(ii) Let $R = \epsilon$.
Then R does not contain the symbol $*$ and $\mathcal{L}(R) = \{\epsilon\}$, which is finite.
So $P(R)$ holds as wanted.

(iii) Let $R = c$, where $c \in \Sigma$.
Then R does not contain the symbol $*$ and $\mathcal{L}(R) = \{c\}$, which is finite.
So $P(R)$ holds as wanted.

Induction Step: Let S, T be regexes.
Suppose $P(S)$ and $P(T)$ hold. [IH]
I.e., if S does not contain the symbol $*$, then $\mathcal{L}(S)$ is finite, and
if T does not contain the symbol $*$, then $\mathcal{L}(T)$ is finite.

There are 3 cases.

(a) Let $R = (S + T)$.
Suppose R does not contain the symbol $*$.
That means neither S nor T contains the symbol $*$.
So by IH, both $\mathcal{L}(S)$ and $\mathcal{L}(T)$ are finite.
Thus $\mathcal{L}(R) = \mathcal{L}(S + T)$ $[R = S + T]$

$= \mathcal{L}(S) \cup \mathcal{L}(T)$. [definition of $\mathcal{L}(S+T)$]

Since the union of two finite languages is finite, therefore $\mathcal{L}(R)$ is finite as wanted.

(b) Let $R = (ST)$.
Suppose R does not contain the symbol $*$.
That means neither S nor T contains the symbol $*$.
So by IH, both $\mathcal{L}(S)$ and $\mathcal{L}(T)$ are finite.
Thus $\mathcal{L}(R) = \mathcal{L}(ST)$ $[R = ST]$
$= \mathcal{L}(S)\mathcal{L}(T)$. [definition of $\mathcal{L}(ST)$]
Since the concatenation of two finite languages is finite, therefore $\mathcal{L}(R)$ is finite as wanted.

(c) Let $R = S^*$.
Then R contains a symbol $*$.
So $P(R)$ is (vacuously) true as wanted. □

⋆ Proofs of closure properties for regular languages

Sometimes structural induction (on structure of regexes) can be used to prove that regular languages are closed under certain language operations. Here is how.

Suppose we want to prove that regular languages are closed under language operation, say $f(L)$.
First we define, for a regex R, the predicate $P(R)$ as follows.

$P(R)$: There exists a regex R' such that $\mathcal{L}(R') = f(\mathcal{L}(R))$.
I.e., some regex R' denotes the language obtained by applying f to the language $\mathcal{L}(R)$.

Then we prove that $P(R)$ holds for every regex R (using structural induction).

Here is why this gives us our desired result.

Suppose A is an arbitrary regular language.
Then $A = \mathcal{L}(R)$ for some regex R.
Applying what was proved by induction above, there is a regex R' such that
$\mathcal{L}(R') = f(\mathcal{L}(R)) = f(A)$.
So $f(A)$ is denoted by some regex. Thus it is regular.

Note: For some language operations, this method of proof works very well.
For others, it does not work at all.

Example:

Let $\Sigma = \{0, 1\}$. Consider the following language operation.

$\text{Ins0}(L) = \{y : \text{for some } u, v \in \Sigma^*,\ uv \in L \text{ and } y = u0v\}$.

Informally, we get a string in $\text{Ins0}(L)$ by taking any string in L and inserting a 0 anywhere in it.
Using structural induction (on the structure of regexes), we will prove that if L is any regular language, then so is $\text{Ins0}(L)$.

We start by defining a predicate $P(R)$ for any regex R.

$P(R)$: There is a regex R' such that $\mathcal{L}(R') = \text{Ins0}(\mathcal{L}(R))$.

Now we prove that $P(R)$ holds for all regexes R, and our result follows.

Basis: There are 3 cases.
(i) For $R = \emptyset$, let $R' = \emptyset$.

Then $\mathcal{L}(R) = \emptyset$.
So $\text{Ins0}(\mathcal{L}(R)) = \emptyset$ [definition of Ins0]
$= \mathcal{L}(R')$
as wanted.

(ii) For $R = \epsilon$, let $R' = 0$.
Then $\mathcal{L}(R) = \{\epsilon\}$.
So $\text{Ins0}(\mathcal{L}(R)) = \{0\}$ [definition of Ins0]
$= \mathcal{L}(R')$
as wanted.

(iii) For $R = b$, where $b \in \Sigma$, let $R' = (0b + b0)$.
Then $\mathcal{L}(R) = \{b\}$.
So $\text{Ins0}(\mathcal{L}(R)) = \{0b, b0\}$ [definition of Ins0]
$= \mathcal{L}(R')$
as wanted.

INDUCTION STEP: Let S, T be regexes.
Suppose $P(S)$ and $P(T)$ hold. [IH]
I.e., there are regexes S', T' such that $\mathcal{L}(S') = \text{Ins0}(\mathcal{L}(S))$ and $\mathcal{L}(T') = \text{Ins0}(\mathcal{L}(T))$.

There are 3 cases.

(a) For $R = (S + T)$, let $R' = (S' + T')$. Then

$$\begin{aligned}
\text{Ins0}(\mathcal{L}(R)) &= \text{Ins0}(\mathcal{L}(S + T)) && [R = (S + T)] \\
&= \text{Ins0}(\mathcal{L}(S) \cup \mathcal{L}(T)) && [\text{definition of } \mathcal{L}(S + T)] \\
&= \text{Ins0}(\mathcal{L}(S)) \cup \text{Ins0}(\mathcal{L}(T)) && [\text{the Ins0 of the union of 2 languages equals} \\
&&& \text{the union of the Ins0 of those languages}] \\
&= \mathcal{L}(S') \cup \mathcal{L}(T') && [\text{IH}] \\
&= \mathcal{L}(S' + T') && [\text{definition of } \mathcal{L}(S' + T')] \\
&= \mathcal{L}(R')
\end{aligned}$$

as wanted.

(b) For $R = (ST)$, let $R' = (S'T + ST')$. Then

$$\begin{aligned}
\text{Ins0}(\mathcal{L}(R)) &= \text{Ins0}(\mathcal{L}(ST)) && [R = (ST)] \\
&= \text{Ins0}\Big(\mathcal{L}(S) \cdot \mathcal{L}(T)\Big) && [\text{definition of } \mathcal{L}(ST)] \\
&= \Big(\text{Ins0}(\mathcal{L}(S)) \cdot \mathcal{L}(T)\Big) \cup \Big(\mathcal{L}(S) \cdot \text{Ins0}(\mathcal{L}(T))\Big) && [\text{insert into either } \mathcal{L}(S) \text{ or } \mathcal{L}(T)] \\
&= \Big(\mathcal{L}(S') \cdot \mathcal{L}(T)\Big) \cup \Big(\mathcal{L}(S) \cdot \mathcal{L}(T')\Big) && [\text{IH}] \\
&= \mathcal{L}(S'T + ST') && [\text{definition of } \mathcal{L}(S'T + ST')] \\
&= \mathcal{L}(R')
\end{aligned}$$

as wanted.

(c) For $R = S^*$, let $R' = (0 + S^*S'S^*)$. Then

$$\begin{aligned}
\text{Ins0}(\mathcal{L}(R)) &= \text{Ins0}(\mathcal{L}(S^*)) && [R = S^*] \\
&= \{0\} \cup \Big(\mathcal{L}(S^*) \cdot \text{Ins0}(\mathcal{L}(S)) \cdot \mathcal{L}(S^*)\Big) && [\text{consider 2 cases:} \\
&&& \text{- inserting into } \epsilon \text{ yields 0,} \\
&&& \text{- inserting into a nonempty string means} \\
&&& \text{inserting into one of one or more} \\
&&& \text{strings from } \mathcal{L}(S) \text{ concatenated together}] \\
&= \{0\} \cup \Big(\mathcal{L}(S^*) \cdot \mathcal{L}(S') \cdot \mathcal{L}(S^*)\Big) && [\text{IH}] \\
&= \mathcal{L}(0 + S^*S'S^*) && [\text{definition of } \mathcal{L}(0 + S^*S'S^*)] \\
&= \mathcal{L}(R')
\end{aligned}$$

as wanted. □