CS420

Introduction to the Theory of Computation

Lecture 9: Power, Kleene star, equivalence

Tiago Cogumbreiro

Today we will learn...

- Void
- All
- Power
- Kleene star
- Language equivalence

The void language

Void

- The language that rejects all strings.

Void

> The language that rejects all strings.

```
Definition Void w := False.
```

Correction properties

1. Show every word is rejected by `Void`

The all language

All

- Language that accepts all strings

`All`

Language that accepts all strings

```
Definition All (w:word) := True.
```

Correction properties

1. Show that any word is accepted by `All`.

Exercises

Coq	Notation	Math
`Nil`		$\{\epsilon\}$
`Char c`	`c`	$\{c\}$
`Union L1 L2`	`L1 U L2`	$L_1 \cup L_2$
`App L1 L2`	`L1 >> L2`	$L_1 \cdot L_2$
`Void`		$\emptyset$
`All`		$\Sigma^\star$

- $L_1 = \{[0], [1], [2]\}$
- $L_2 = \{[3], [4]\}$

Solve the following exercises

1. $L_1 \cup \{\epsilon\} =$

Exercises

Coq	Notation	Math
`Nil`		$\{\epsilon\}$
`Char c`	`c`	$\{c\}$
`Union L1 L2`	`L1 U L2`	$L_1 \cup L_2$
`App L1 L2`	`L1 >> L2`	$L_1 \cdot L_2$
`Void`		$\emptyset$
`All`		$\Sigma^\star$

- $L_1 = \{[0], [1], [2]\}$
- $L_2 = \{[3], [4]\}$

Solve the following exercises

1. $L_1 \cup \{\epsilon\} = \{[0], [1], [2], \epsilon\}$
2. $L_1 \cup L_2 =$

Exercises

Coq	Notation	Math
`Nil`		$\{\epsilon\}$
`Char c`	`c`	$\{c\}$
`Union L1 L2`	`L1 U L2`	$L_1 \cup L_2$
`App L1 L2`	`L1 >> L2`	$L_1 \cdot L_2$
`Void`		$\emptyset$
`All`		$\Sigma^\star$

- $L_1 = \{[0], [1], [2]\}$
- $L_2 = \{[3], [4]\}$

Solve the following exercises

1. $L_1 \cup \{\epsilon\} = \{[0], [1], [2], \epsilon\}$
2. $L_1 \cup L_2 = \{[0], [1], [2], [3], [4]\}$
3. $L_1 \cdot L_2 =$

Exercises

Coq	Notation	Math
`Nil`		$\{\epsilon\}$
`Char c`	`c`	$\{c\}$
`Union L1 L2`	`L1 U L2`	$L_1 \cup L_2$
`App L1 L2`	`L1 >> L2`	$L_1 \cdot L_2$
`Void`		$\emptyset$
`All`		$\Sigma^\star$

- $L_1 = \{[0], [1], [2]\}$
- $L_2 = \{[3], [4]\}$

Solve the following exercises

1. $L_1 \cup \{\epsilon\} = \{[0], [1], [2], \epsilon\}$
2. $L_1 \cup L_2 = \{[0], [1], [2], [3], [4]\}$
3. $L_1 \cdot L_2 = \{[0, 3], [0, 4], [1, 3], [1, 4], [2, 4]\}$
4. $L_2 \cdot \{\epsilon\} =$

Exercises

Coq	Notation	Math
`Nil`		$\{\epsilon\}$
`Char c`	`c`	$\{c\}$
`Union L1 L2`	`L1 U L2`	$L_1 \cup L_2$
`App L1 L2`	`L1 >> L2`	$L_1 \cdot L_2$
`Void`		$\emptyset$
`All`		$\Sigma^\star$

- $L_1 = \{[0], [1], [2]\}$
- $L_2 = \{[3], [4]\}$

Solve the following exercises

1. $L_1 \cup \{\epsilon\} = \{[0], [1], [2], \epsilon\}$
2. $L_1 \cup L_2 = \{[0], [1], [2], [3], [4]\}$
3. $L_1 \cdot L_2 = \{[0, 3], [0, 4], [1, 3], [1, 4], [2, 4]\}$
4. $L_2 \cdot \{\epsilon\} = L_2$
5. $L_1 \cup \Sigma^\star =$

Exercises

Coq	Notation	Math
`Nil`		$\{\epsilon\}$
`Char c`	`c`	$\{c\}$
`Union L1 L2`	`L1 U L2`	$L_1 \cup L_2$
`App L1 L2`	`L1 >> L2`	$L_1 \cdot L_2$
`Void`		$\emptyset$
`All`		$\Sigma^\star$

- $L_1 = \{[0], [1], [2]\}$
- $L_2 = \{[3], [4]\}$

Solve the following exercises

1. $L_1 \cup \{\epsilon\} = \{[0], [1], [2], \epsilon\}$
2. $L_1 \cup L_2 = \{[0], [1], [2], [3], [4]\}$
3. $L_1 \cdot L_2 = \{[0, 3], [0, 4], [1, 3], [1, 4], [2, 4]\}$
4. $L_2 \cdot \{\epsilon\} = L_2$
5. $L_1 \cup \Sigma^\star = \Sigma^\star$
6. $L_2 \cup \emptyset =$

Exercises

Coq	Notation	Math
`Nil`		$\{\epsilon\}$
`Char c`	`c`	$\{c\}$
`Union L1 L2`	`L1 U L2`	$L_1 \cup L_2$
`App L1 L2`	`L1 >> L2`	$L_1 \cdot L_2$
`Void`		$\emptyset$
`All`		$\Sigma^\star$

- $L_1 = \{[0], [1], [2]\}$
- $L_2 = \{[3], [4]\}$

Solve the following exercises

1. $L_1 \cup \{\epsilon\} = \{[0], [1], [2], \epsilon\}$
2. $L_1 \cup L_2 = \{[0], [1], [2], [3], [4]\}$
3. $L_1 \cdot L_2 = \{[0, 3], [0, 4], [1, 3], [1, 4], [2, 4]\}$
4. $L_2 \cdot \{\epsilon\} = L_2$
5. $L_1 \cup \Sigma^\star = \Sigma^\star$
6. $L_2 \cup \emptyset = L_2$
7. $L_2 \cdot \emptyset =$

Exercises

Coq	Notation	Math
`Nil`		$\{\epsilon\}$
`Char c`	`c`	$\{c\}$
`Union L1 L2`	`L1 U L2`	$L_1 \cup L_2$
`App L1 L2`	`L1 >> L2`	$L_1 \cdot L_2$
`Void`		$\emptyset$
`All`		$\Sigma^\star$

- $L_1 = \{[0], [1], [2]\}$
- $L_2 = \{[3], [4]\}$

Solve the following exercises

1. $L_1 \cup \{\epsilon\} = \{[0], [1], [2], \epsilon\}$
2. $L_1 \cup L_2 = \{[0], [1], [2], [3], [4]\}$
3. $L_1 \cdot L_2 = \{[0, 3], [0, 4], [1, 3], [1, 4], [2, 4]\}$
4. $L_2 \cdot \{\epsilon\} = L_2$
5. $L_1 \cup \Sigma^\star = \Sigma^\star$
6. $L_2 \cup \emptyset = L_2$
7. $L_2 \cdot \emptyset = \emptyset$

The power operator for languages

The power operator for languages

- $L^{n+1} = L \cdot L^n$
- $L^0 = \{\epsilon\}$

Example

- $L = \{[0], [1], [2]\}$
- $L^0 = \{\epsilon\}$
- $L^1 = L \cdot \{\epsilon\} = L$
- $L^2 = L \cdot L = \{[0, 0], [0, 1], [0, 2], [1, 0], [1, 1], [1, 2], [2, 0], [2, 1], [2, 2]\}$

Implementing power

```
Inductive Pow (L:language) : nat → word → Prop :=
| pow_nil:
  Pow L 0 nil
| pow_cons:
  forall n w1 w2 w3,
  In w2 (Pow L n) →
  In w1 L →
  w3 = w1 ++ w2 →
  Pow L (S n) w3.
```

Rule pow_nil:

$$\epsilon \in L^0$$

Rule pow_cons:

$$\frac{w_2 \in L^n \qquad w_1 \in L}{w_1 \cdot w_2 \in L^{n+1}}$$

Rules in the form of:

$$\frac{P_1 \qquad P_2 \qquad P_3}{Q}$$

Are read as: If P_1 **and** P_2 **and** P_3 all hold, **then** we have Q.

Exercise

```
Require Import Coq.Strings.Ascii.
Require Import Coq.Lists.List.
From Turing Require Import Lang.
From Turing Require Import Util.
Import Lang.Examples.
Import LangNotations.
Import ListNotations.
Open Scope lang_scope.
Open Scope char_scope.

Lemma in_aaa:
  In ["a"; "a"; "a"] (Pow "a" 3).
Proof.
Qed.

Lemma pow_char_in_inv:
  forall c n w,
  In w (Pow (Char c) n) →
  w = Util.pow1 c n.
Proof.
Qed.
```

Kleene operator

Kleene operator

$L^\star = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \cdots$

Inductive definition

$$\frac{w \in L^n}{w \in L^\star}$$

Wait, what is n?

Any n will do. If you can build a proof object such that $w \in L^n$, then $w \in L^\star$.

Does this mean that there is only one n? Say, $L^\star = L^{1000}$?

NO it does not. Each word membership will have its possibly distinct n.

Example: $L = [a]$, we have that $\epsilon \in L^0$ and that $[a, a] \in L^2$, thus $\epsilon \in L^\star$ and $[a, a] \in L^\star$.

Exercise

```
Lemma in_aaa_2:
  In ["a"; "a"; "a"] (Star "a").
Proof.
```

Language Equivalence

Language equivalence (equality)

- Mathematically, we write $L_1 = L_2$ to mean that two languages are equal.
- How do you prove language equality?

Language equivalence (equality)

- Mathematically, we write $L_1 = L_2$ to mean that two languages are equal.
- How do you prove language equality?
- You have to show that all words in L_1 are also in L_2 and vice-versa.

Language equivalence in Coq

```
Definition Equiv (L1 L2:language) := forall w, L1 w <-> L2 w.
```

Show that Vowel is equivalent to previous example

```
Lemma vowel_eq:
  Vowel == (Char "a" U Char "e" U Char "i" U Char "o" U Char "u").
Proof.
```

Language equivalence in Coq

```
Definition Equiv (L1 L2:language) := forall w, L1 w <-> L2 w.
```

Show that `Vowel` is equivalent to previous example

```
Lemma vowel_eq:
  Vowel == (Char "a" U Char "e" U Char "i" U Char "o" U Char "u").
Proof.

  apply vowel_iff.
Qed.
```

Exercise

Show that `Void` is a neutral element in union.

```
Lemma union_l_void:
  forall L,
  L ∪ Void == L.
```

Exercise

Show that `Void` is a neutral element in union.

```
Lemma union_l_void:
  forall L,
  L U Void == L.

Proof.
  split; intros.
  - destruct H. {
      assumption.
    }
    apply not_in_void in H.
    contradiction.
  - left.
    assumption.
Qed.
```

Exercise

Show that `Void` is an absorbing element in concatenation.

```
Lemma app_l_void:
  forall L,
  L >> Void == Void.
```

Exercise

Show that `Void` is an absorbing element in concatenation.

```
Lemma app_l_void:
  forall L,
  L >> Void == Void.

Proof.
  unfold App; split; intros.
  - destruct H as (w1, (w2, (Ha, (Hb, Hc)))).
    subst.
    apply not_in_void in Hc.
    contradiction.
  - apply not_in_void in H.
    contradiction.
Qed.
```

Exercise

- A language that accepts any words that consists of two vowels

Exercise

- A language that accepts any words that consists of two vowels

```
Definition TwoVowels := Vowel >> Vowel.
```

Show that `["a"; "e"]` is in `TwoVowels`

Exercise

A language that accepts any words that consists of two vowels

```
Definition TwoVowels := Vowel >> Vowel.
```

Show that `["a"; "e"]` is in `TwoVowels`

```
Goal In ["a"; "e"] (Vowel >> Vowel).
Proof.
```

Exercise

A language that accepts any words that consists of two vowels

```
Definition TwoVowels := Vowel >> Vowel.
```

Show that `["a"; "e"]` is in `TwoVowels`

```
Goal In ["a"; "e"] (Vowel >> Vowel).
Proof.

  unfold App.
  exists ["a"], ["e"]. (* Existential in the goal *)
  split. { reflexivity. }
  split. { left. reflexivity. }
  right. left. reflexivity.
Qed.
```

Exercise

What words are accepted by L2?

```
Definition L2 := All >> Char "a".
```

Exercise

- Rewrite `Vowels` to use only language operators.

Exercise

Rewrite `Vowels` to use only language operators.

```
Definition Vowels2 := Char "a" U Char "e" U Char "i" U Char "o" U Char "u".
```

Exercise

```
Lemma vowel_length:
  forall w,
  Vowel w ->
  length w = 1.
```

Exercise

```
Lemma vowel_length:
  forall w,
  Vowel w ->
  length w = 1.

Proof.
  intros.
  destruct H as [H|[H|[H|[H|H]]]]; subst; reflexivity.
Qed.
```

Exercise

```
Goal forall w, (Vowel >> Vowel) w -> length w = 2.
```

Exercise

```
Goal forall w, (Vowel >> Vowel) w -> length w = 2.
```

```
Proof.
  intros.
  unfold App in *.
  destruct H as (w1, (w2, (Ha, (Hb, Hc)))). (* Existential in hypothesis *)
  subst. apply vowel_length in Hb. apply vowel_length in Hc.
  SearchAbout (length(_ ++ _)).              (* Search for lemmas *)
  rewrite app_length. rewrite Hb. rewrite Hc. reflexivity.
Qed.
```

Exercise

Show that all strings are rejected by `Void`.

Exercise

Show that all strings are rejected by `Void`.

```
Lemma not_in_void:
  forall w,
  ~ In w Void.
Proof.
  intros.
  intros N.
  inversion N.
Qed.
```