Formal Languages: Concepts

- Alphabet (A) - any finite non-empty set of letters (used to write the input)
  e.g. $A = \{0, 1\}$, $E = \{a, b, c, \ldots, z\}$
- Word (w) (akka string) - finite sequence of letters (elements of the alphabet A)
  $w \in A^*$ (here A^* is the set of all finite sequences of elements of A)
  $A^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$ (all words)
  We write sequence denoting a word by just writing one letter after another
  ε is the word of length zero (empty string)
  Length of the word $|w|$ is the number of symbols (repetitions count): $|01011| = 5$
- Language (L) - a set of words (possibly empty, possibly infinite)
  $L \subseteq A^*$
  e.g. $L_1 = \{1, 11, 111, \ldots\}$ (words of length one or more, containing only 1-s)
  $L_2 = \{\varepsilon, 00, 01, 10, 11, 0000, 0001, 0010, \ldots\}$ (words of even length)
  $L_3 = \{0, 101, 111, 00000\}$ (finite language with these specific four words)

Definition of Words in Set Theory

Let A be the alphabet. We define words of length n, denoted A^n

Definition: $A^0 = \{\varepsilon\}$ (only one word of length zero, always denoted ε)
For $n > 0$, $A^n = \{f \mid f : \{0, \ldots, n-1\} \to A\}$
A non-empty word is just a function that tells us what the letters are and in which order.
For $w = \mathbf{1011}$ we thus have:
$w(0) = \mathbf{1} \quad w(1) = \mathbf{0} \quad w(2) = \mathbf{1} \quad w(3) = \mathbf{1}$
(We also write the pretty $w_{(i)}$ instead of $w(i)$)
Set of all words:

$$A^* = \bigcup_{n \geq 0} A^n$$

which means: $w \in A^*$ if and only iff there exists n such that $w \in A^n$.
Note: sometimes people represent e.g. 1011 as $(1,0,1,1)$, but we can think of n-tuple as a function $\{0, \ldots, n-1\} \to A$, so that is equivalent.

Word Equality

Words are equal when they have same length and same letters in the same order:

Let $u, v \in A^*$. Then

$u = v$ if and only if both

1. $|u| = |v|$ and
2. for all i where $0 \leq i < |u|$ we have $u_{(i)} = v_{(i)}$

Words as Scala Lists

```scala
sealed abstract class List[A] { // A is the alphabet
  def ::(t:A): List[A] = Cons(t, this)
  def length: BigInt = this match {
    case Nil() ⇒ BigInt(0)
    case Cons(h, t) ⇒ 1 + t.length }
  def apply(index: BigInt): A = {
    this match {
      case Cons(h,t) ⇒
        if (index == BigInt(0)) h
        else t(index-1) } }
}
case class Nil[A]() extends List[A]
case class Cons[A](h: A, t: List[A]) extends List[A]

val w = 1 :: 0 :: 1 :: 1 :: Nil[Int]() // 1011
val n = w.length // 4
val z = w(1) // 0
```

Words as Inductive Structures

If $a \in A$ and $u \in A^*$, let $a \cdot u$ denote the word that starts with a and then follows with symbols from u (like Cons).

Theorem (Decomposing a word)

Given $w \in A^*$, either $w = \varepsilon$ or $w = a \cdot v$ where $a \in A$ and $v \in A^*$.

Theorem (Equality)

Given $u, v \in A^*$ we have $u = v$ if and only if one of the following conditions hold:

- $u = \varepsilon$ *and* $v = \varepsilon$.
- *there exists* $a \in A$ *and* $u', v' \in A^*$ *such that* $u = a \cdot u'$, $v = a \cdot v'$ *and* $u' = v'$.

Theorem (Structural induction for words)

Given a property of words $P : A^* \to \{true, false\}$, if (1) $P(\varepsilon)$ and, (2) for every letter $a \in A$ and every u, $P(u)$ implies $P(a \cdot u)$, then: $\forall u \in A^*. P(u)$.

Each Word is Finite. The Set of All of Them is Infinite

Each word has a finite length, and each symbol is an element from a finite set. Thus, each word is a finite object that can be written down using finitely many bits.

That set of all words is countably infinite: it is as big as the set of natural numbers. For example, if $A = \{1\}$ then each word is of the form $1 \dots 1$ and is uniquely given by its length n. Thus, there is a bijection between such words and non-negative integers n, which, by definition, means that these two sets have the same cardinality. Similarly, if $A = \{0,1\}$, we have a bijection between positive integers and words over A: given a word of length n of the form $k_1 \dots k_n$ we can assign it to a strictly positive integer whose binary number representation is

$$\overline{1k_1 \dots k_n}$$

Such mapping establishes a bijection between A^* and postitive integers. More generally, we can show that, for any alphabet A the set of all words A^* is a countably infinite set. Intuitively, we can take any total ordering on A and use it to sort all words as in a dictionary. This defines a bijection with non-negative integers.

Concatenation

Concatenation is a fundamental operation on words, and denotes putting the words of one word after another. For example, concatenating words 01 and 10, denoted $01 \cdot 10$, results in the word 0110.
Concatenation of $u = u_{(0)} \dots u_{(n-1)}$ and $v = v_{(0)} \dots v_{(m-1)}$, denoted $u \cdot v$, or uv for short, is the word

$$u_{(0)} \dots u_{(n-1)} v_{(0)} \dots v_{(m-1)}$$

Definition

$u \cdot v$ is the unique word w such that $|w| = |u| + |v|$ and for all i where $0 \leq i < |w|$,

$$w_{(i)} = \begin{cases} u_{(i)}, & \text{if } 0 \leq i < |u| \\ v_{(i-|u|)}, & \text{if } |u| \leq i < |u| + |v| \end{cases}$$

Note that it follows: $w \cdot \varepsilon = w$ and $\varepsilon \cdot w = w$

Associativity of Concatenation

Theorem

For all $u, v, w \in A$,

$$u \cdot (v \cdot w) = (u \cdot v) \cdot w$$

First, we show that the two words have the same length. Indeed, $|u \cdot (v \cdot w)| = |u| + |v \cdot w| = |u| + |v| + |w|$ and likewise $|(u \cdot v) \cdot w| = |u \cdot v| + |w| = |u| + |v| + |w|$.
Next, we show that the letters are same at all positions i where $0 \leq i < |u| + |v| + |w|$.
Pick any such i. There are three cases, depending on the interval to which i belongs.
Case $i < |u|$. We have $(u \cdot (v \cdot w))_{(i)} = u_{(i)}$ by the definition of concatenation.
Similarly, because $i < |u \cdot v|$, we have that likewise $((u \cdot v) \cdot w)_{(i)} = (u \cdot v)_{(i)} = u_{(i)}$.
Case $|u| \leq i < |u| + |v|$. We have $(u \cdot (v \cdot w))_{(i)} = (v \cdot w)_{i-|u|} = v_{i-|u|}$ and also $((u \cdot v) \cdot w)_{(i)} = (u \cdot v)_i = v_{i-|u|}$.
Case $|u| + |v| \leq i$. We have $(u \cdot (v \cdot w))_{(i)} = (v \cdot w)_{i-|u|} = w_{i-|u|-|v|}$ and also $((u \cdot v) \cdot w)_{(i)} = w_{i-|u \cdot v|} = w_{i-|u|-|v|}$.

Free Monoid of Words

The neutral element and associativity law imply that the structure $(A^*, \cdot, \varepsilon)$ is an algebraic structure called *monoid*. The monoid of words is called the *free monoid*. Word monoid satisfies, among others, the following additional properties (which do not hold in all monoids).

Theorem (Left cancellation law)

For every three words $u, v, w \in A^*$, *if* $wu = wv$, *then* $u = v$.

Theorem (Right cancellation law)

For every three words $u, v, w \in A^*$, *if* $uw = vw$, *then* $u = v$.

Reversal

Reversal of a word is a word of same length with symbols but in the reverse order.
Example: the reversal of the word 011, denoted $(011)^{-1}$, is the word 110.

Definition

Given $w \in A^*$, its reversal w^{-1} is the unique word such that $|w^{-1}| = |w|$ and $w^{-1}_{(i)} = w_{(|w|-1-i)}$ for all i where $0 \leq i < |w|$.

From definition it follows that $\varepsilon^{-1} = \varepsilon$ and that $a^{-1} = a$ for all $a \in A$.

Theorem

For all $u, v \in A^*$, $(u^{-1})^{-1} = u$ and $(uv)^{-1} = v^{-1}u^{-1}$.

Every law about words has a dual version.
Here is the dual of induction principle, where we peel off last elements.

Theorem (Structural induction for words (dual))

Given a property of words $P : A^* \to \{true, false\}$, if (1) $P(\varepsilon)$ and, (2) for every letter $a \in A$ and every u, $P(u)$ implies $P(u \cdot a)$, then: $\forall u \in A^*.P(u)$.

Prefix, Postfix, and Slice

Definition

Let $u, v, w \in A^*$ such that $uv = w$. We then say that u *is a prefix of* w and that v *is a suffix of* w.

Definition

Given a word $w \in A^*$ and two integers p, q such that $0 \leq p \leq q \leq |w|$, the $[p, q)$*-slice of* w, denoted $w_{[p,q)}$, is the word u such that $|u| = q - p$ and $u_{(i)} = w_{(p+i)}$ for all i where $0 \leq i < q - p$.

Theorem

Let $w \in A^*$ and $u = w_{[p,q)}$ where $0 \leq p \leq q \leq |w|$. Then the exist words $x, y \in A^*$ such that $|x| = p$, $|y| = |w| - q$, and $w = xuy$.

Theorem

Let $w, u, x, y \in A^*$ and $w = xuy$. Then $x = w_{[0,|x|)}$, $u = w_{[|x|,|x|+|u|)}$ and $v = w_{[|x|+|u|,|w|)}$.

Slice in Scala

$w \in A^*$, $0 \leq p \leq q \leq |w|$, $[p,q)$-*slice of* w, denoted $w_{[p,q)}$, is u such that $|u| = q - p$ and $u_{(i)} = w_{(p+i)}$ for all i where $0 \leq i < q - p$.

```scala
def slice(i: BigInt, j: BigInt): List[T] = {
  require(0 <= i && i <= j && j <= length)
  this match {
    case Nil() => Nil()
    case Cons(h,t) =>
      if (i == 0 && j == 0) Nil()
      else if (i == 0) Cons(h, t.slice(0, j-1))
      else t.slice(i-1, j-1)
  }
} ensuring(_.size == j - i)
```