1.3 (Part A) Tuples

Sets are useful for unordered, possibly infinite collections of elements.

A **tuple** is a finite, ordered collection of *elements* (aka *members*, *components*).

We’ll denote a tuple by writing its elements, in order, separated by commas, beginning with “(“ and ending with “)“.

For example, the tuple

$$(12, R, -9)$$

has three elements: the first is $12$, the second $R$ and the third $-9$.

If a tuple has $n$ elements, we say it has *length $n$*, and call it an **$n$-tuple**.

There is a unique 0-tuple, which can be written $( )$, called the **empty tuple**.

Sometimes tuples are called “vectors” or (finite) “sequences”.

2-tuples are usually called **ordered pairs**.
Tuple equality, Cartesian product

Two $n$-tuples $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ are equal if

$$x_i = y_i \quad \text{for all } i \ (1 \leq i \leq n).$$

For sets $A, B$, the Cartesian product of $A$ and $B$, written $A \times B$, is defined as follows.

$$A \times B = \{ (x, y) \mid x \in A \text{ and } y \in B \}$$

For example, if $A = \{0, 1\}$ and $B = \{1, 2\}$, then

$$A \times B = \emptyset \times \mathbb{N} =$$

We extend the Cartesian product as follows. For sets $A_1, \ldots, A_n$,

$$A_1 \times \cdots \times A_n = \{ (x_1, \ldots, x_n) \mid \text{for all } i \ (1 \leq i \leq n), x_i \in A_i \}.$$  

If all sets $A_i$ are the same set $A$, we also write $A^n$ for their Cartesian product.

$$A^1 =$$  

$$A^0 =$$
Each element of $A^n$ is an $n$-tuple, which is essentially an array of length $n$.

And we often use array-like notations to give “direct access” to the components of an $n$-tuple $x$:

$$(x_1, x_2, \ldots, x_n)$$

or

$$(x(1), x(2), \ldots, x(n))$$

or

$$(x[1], x[2], \ldots, x[n]).$$

And when we “implement” $n$-tuples, we typically implement them as arrays.

Each element of $(A^m)^n$ is an $n$-tuple, each of whose members is an $m$-tuple.

We can think of an element of $(A^m)^n$ as a two-dimensional array, with $n$ rows and $m$ columns.

And similar remarks apply about notation and implementation.

Along the same lines, elements of $A \times B$ are analogous to records or structures with two fields.
Counting tuples

If $A$ and $B$ are finite sets, with $|A| = m$ and $|B| = n$, then what is $|A \times B|$?

If $A$ is a finite set, how can we express $|A^n|$ in terms of $|A|$ and $n$?

For finite set $A$, what is the relationship between $|\text{power}(A)|$ and $|\{0, 1\}^{|A|}|$?
Relations

If $R$ is a subset of $A_1 \times \cdots \times A_n$, then $R$ is said to be an \textit{n-ary relation} on (or over) $A_1 \times \cdots \times A_n$.

If $R$ is an \textit{n-ary relation} on $A^n$, we also say, more simply, that $R$ is an \textit{n-ary relation} on $A$.

Instead of 1-ary we typically say \textit{unary}, instead of 2-ary \textit{binary} and instead of 3-ary \textit{ternary}.

Notice: For any set $A$, $A^n$ is the largest \textit{n-ary relation} on $A$.

What is the smallest \textit{n-ary relation} on $A$?

If $|A| = k$, how many binary relations on $A$?

How many \textit{n-ary relations} on $A$?
Lists

A list is a finite sequence of elements.

So a list is essentially a tuple, except that with lists we do not typically assume that we are working with a tuple of a particular given length, and so we take a different view of how to access the elements of a list.

Instead, we access list elements by taking either the “head” of the list (its first element) or the “tail” of the list...

Accordingly, we use slightly different notation for lists:

“⟨” and “⟩”, instead of “(“ and “)”.

The empty list is written ⟨ ⟩.

The length of a list

\[ L = ⟨x_1, x_2, \ldots, x_n⟩ \]

is \( n \), with

\[ \text{head}(L) = x_1 \]

and

\[ \text{tail}(L) = ⟨x_2, \ldots, x_n⟩. \]

Notice that \( \text{head}(⟨ ⟩) \) and \( \text{tail}(⟨ ⟩) \) are undefined.
Constructing lists

We have a convenient notation for constructing a new list by adding a new element $h$ at the head of a list $L$ — we write

$$\text{cons}(h, L).$$

For example, if

$$L = \langle 1, 2 \rangle$$

then

$$\text{cons}(0, L) = \langle 0, 1, 2 \rangle$$

and

$$\text{cons}(1, \text{cons}(0, L)) = \langle 1, 0, 1, 2 \rangle.$$

As you know from prior study of list implementation via linked lists, the operations cons, head and tail have efficient computer implementations.

Observe that, for any nonempty list $L$,

$$\text{cons}(\text{head}(L), \text{tail}(L)) = L.$$
Of courses lists can have lists as elements:

\[
\text{head}(\langle\langle a, b\rangle, \langle \rangle, c, d\rangle) = \\
\text{tail}(\langle\langle a, b\rangle, \langle \rangle, c, d\rangle) =
\]

We’ll eventually see how to represent trees as lists (whose elements may be lists. . .).

If all the elements of a list \( L \) belong to a set \( A \), we say \( L \) is a list \textit{over} \( A \).

It is convenient to let

\[
\text{lists}(A)
\]

denote the set of all lists over \( A \).
Counting lists

If \(|A| = n\), how many lists over \(A\) of length \(m\)?

How many lists over \(A\) of length at most \(m\)?

How many lists of length \(m\) over \(A \times A\)?