12. The union types

A compound object may come from different types. For example, a component of a world wide web home page may contain normal text, pictures, audio data, and so on, each belonging to a different category. The union types allow us to define such compound objects.

The outline of this part:

- Union type declaration
- Is function
- A specification with union type
12.1 Union type declaration

Let $T_1, T_2, ..., T_n$ denote $n$ types. Then a union type $T$ constituted by these types is declared in the format:

$$T = T_1 \mid T_2 \mid ... \mid T_n$$

A value of $T$ can come from one of the types $T_1$, $T_2$, ..., $T_n$.

It is important to keep $T_1$, $T_2$, ..., $T_n$ disjointed so that any value of type $T$ can be precisely determined to belong to which constituent type.
Example:
Color = \{<Red>, <Blue>, <Yellow>\}
Key = char
Digits = set of nat

the union type Hybrid can then be declared as:
Hybrid = Color | Key | Digits

the following values belong to type Hybrid:
<Red>
<Blue>
'b'
'5'
\{3, 5, 8\}
\{10, 20, 100\}
No operators can be built on a union type except the equality (\(=\)) and inequality (\(<\>)). For example,

\[
<\text{Red}> = <\text{Blue}> \iff \text{false} \\
<\text{Red}> \not= \{3, 5, 8\} \iff \text{true} \\
'b' = 'b' \iff \text{true}
\]
12.2 Is function

When writing specifications there may be a situation that requires a precise type of a given value. Such a type can be determined by applying built-in function known as is function:

\[ \text{is}_T(x) \]

This function is a predicate that yields true when the type of value \( x \) is \( T \) (any type is possible); otherwise, it yields false.

Examples:

\[
\begin{align*}
\text{is}_\text{Color(<Red>)} & \iff \text{true} \\
\text{is}_\text{Hybrid(<Red>)} & \iff \text{true}
\end{align*}
\]
12.3 A specification with union type

We take the identifier of SOFL as an example to illustrate the use of union types.

Identifier = EnglishLetter | StringOfChar
StringOfChar = EnglishLetter * IdentifierBody
IdentifierBody = EnglishLetter | Digit | Underscore | Identifier * Identifier
EnglishLetter = {<a>, <b>, ..., <Y>, <Z>}
Digit = {<0>, <1>, <2>, <3>, <4>, <5>, <6>, <7>, <8>, <9>}
Underscore = {<_>
Examples:

```
<a>
  mk_SringOfChar(<b>, <1>)
  mk_SringOfChar(<b>, mk_SringOfChar(<2>, <3>))
  mk_SringOfChar(<x>, mk_SringOfChar(<_>, <3>))
```

These values represent the strings of characters, disregarding their syntax. So they can be interpreted as the following strings more intuitively:

```
a
b1
b23
x_3
```
1. Define a union type School with the constituent types ElementarySchool, JuniorHighSchool, HighSchool, and University, assuming that all the constituent types are given types.

2. Let s1 and s2 be two variables of the type set of Hybrid. Let s1 = \{<Red>, 3, 'b'\} and s2 = \{<Blue>, 'a', 'b', 9\}. Evaluate the expressions:
   a. card(s1) = card(s2) \iff ?
   b. union(s1, s2) = ?
   c. inter(s1, s2) = ?
   d. diff(s1, s2) = ?
3. Let a, b, c: Identifier. Evaluate the expressions:
   a. is.Identifier(a) = ?
   b. is.Digit(b) = ?
   c. is.EnglishLetter(c) = ?