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 types

Union type declaration

Let T1, T2, ..., Tn be n types. Then, a union type T constituted from these types is declared in the format:

T = T1 | T2 | ... | Tn

A value of T can come from one of the types T1, T2, ..., Tn.

It is important to keep T1, T2, ..., Tn disjoint 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 the type Hybrid:

<Red>

<Blue>

'b'

'5'

{10, 20, 100}

No operators can be built on a union type except the equality (=) and inequality (<>).

For example,

<Red> = <Blue> <=> false

<Red> <> {3, 5, 8} <=> true

'b' = 'b' <=> true

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 a built-in function known as is function:

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:

is_Color(<Red>) <=> true

is_Hybrid(<Red>) <=> true

Specification with a union type

Suppose we want to write a program that scans a a specification in SOFL and records different kinds of tokens in different tables. We first declare Token as a union type:

Token = EnglishLetter | Identifier |SpecialCharacter

where EnglishLetter, Identifier, and SpecialCharacter are supposed to have been declared before.

We then build a process Record_Token to record different tokens obtained by scanning the current text in different tables.

```
process Record_Token(token: Token)
ext wr english_char_table: seq of EnglishLetter
      wr identifier_table: seq of Identifier
      wr special_char_table: seq of SpecialCharacter
post (is_EnglishLetter(token) =>
english_char_table = conc(~english_char_table, [token])) and
        (is_Identifier(token) =>
identifier_table = conc(~identifier_table, [token])) and
        (is_SpecialCharacter(token) =>
special_char_table = conc(~special_char_table, [token]))
comment
 The token is recorded in the corresponding table.
end_process
```

Class exercise 9

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) <=> ?
   b. union(s1, s2) = ?
   c. inter(s1, s2) = ?
   d. diff(s1, s2) = ?