31.2.6.1 rf_scalar

This struct *like*-inherits from **rf_type** (see 31.2.1.2).

a) **rf_scalar.get_size_in_bits()**: int

Returns the size of this scalar type in bits.

b) **rf_scalar.get_range_string()**: string

Returns a string representation of the scalar range of values in the format of range modifiers (e.g., the string `"[1..4,7,9..10]"`).

31.2.6.2 rf_numeric

This struct *like*-inherits from **rf_scalar** (see 31.2.6.1).

a) **rf_numeric.is_signed()**: bool

Returns TRUE if the numeric type is signed. Otherwise, returns FALSE.

b) **rf_numeric.get_set_of_values()**: set

Returns a set that contains all the legal values of this numeric type, similar to using the **set_of_values()** operator (see 5.8.4). If this type is an unbounded integer type without a range restriction, an exception is thrown.

c) **rf_numeric.get_full_set_of_values()**: set

Returns a set that contains all possible values of this numeric type, similar to using the **full_set_of_values()** operator (see 5.8.5). If this type is an unbounded integer type, an exception is thrown.

31.2.6.3 rf_enum

This struct *like*-inherits from **rf_scalar** (see 31.2.6.1).

a) **rf_enum.get_items()**: list of rf_enum_item

Returns the set of named values for this type. The legal values of an **enum** type (see 4.3.2.3) are not restricted by a range declaration, e.g., the type introduced by the statement `type my_color: color [red..blue]` has the same items as `type color`. Such declarations only affect generation properties of the type.

b) **rf_enum.get_item_by_value(***value*: int**)**: rf_enum_item

Returns the named value object for *value* or NULL if no such value exists in this type's range.

c) **rf_enum.get_item_by_name(***name*: string**)**: rf_enum_item

Returns the named value object for *name* or NULL if no value by such name exists in this type's range.

The following method of **rf_enum** is described in 31.3.2.2:

rf_enum.get_layers()

31.2.6.4 rf_enum_item

The following method of **rf_named_entity** is described in 31.2.1.1:

Enum items are pairs of identifier-integer, which are the possible values of a variable of that **enum** type. The integer values of **enum** items are the numbers assigned to them explicitly in the declaration (e.g., `[red = 3, green = 17]`) or the default (consecutive) numbers.

a) **rf_enum_item.get_defining_type()**: rf_enum