Aurox language specification

Jakub Grobelny

21 maja 2019

1 Syntax

Character classification:

- <*whitespace*> — HT, LF, CR, SPACE
- <*digit*> — 0-9
- <*lowercase*> — underscore or any other lowercase Unicode characters [1]
- <*uppercase*> — any uppercase Unicode characters [2]
- <*special*> — ’-’, ’+’, ’*’, ’/’, ’=’, ’>’, ’<’, ’.’, ’!’, ’@’, ’%’, ’ˆ’, ’ ’, ’&’, ’$’, ’|’

Any character sequence beginning with character `#` ending with LF are *comments.*

$\langle$*operator*$\rangle$ ::= $\langle$*special*$\rangle$
  | $\langle$*special*$\rangle$ $\langle$*operator*$\rangle$

$\langle$*keyword*$\rangle$ ::= **let** | **in** | **if** | **and** | **or** | **then**
  | **match** | **else** | **with** | **type** | **import**
  | **define** | **_** | **defop** | **end** | **case**

[1]All characters `X` which satisfy `char_type(X, lower) predicate in SWI Prolog`
[2]All characters `X` which satisfy `char_type(X, upper) predicate in SWI Prolog`

⟨*identifier*⟩ ::= ⟨*lowercase*⟩ ⟨*alphanum*⟩

⟨*type name*⟩ ::= ⟨*uppercase*⟩ ⟨*alphanum*⟩

⟨*alphanum*⟩ ::= ⟨*alphanum char*⟩ ⟨*alphanum*⟩ | ϵ

⟨*alphanum char*⟩ ::= ⟨*lowercase*⟩ | **?** | ⟨*digit*⟩

⟨*integer*⟩ ::= ⟨*digit*⟩ | ⟨*digit*⟩ ⟨*integer*⟩

⟨*float*⟩ ::= ⟨*integer*⟩ . ⟨*digit sequence*⟩ ⟨*exponent*⟩ **e**
  | ⟨*integer*⟩ ⟨*expontent*⟩

⟨*digit sequence*⟩ ::= ⟨*digit*⟩ | ⟨*digit*⟩ ⟨*digit sequence*⟩

⟨*e*⟩ ::= **e** | **E**

⟨*exponent*⟩ ::= ⟨*e*⟩ **-** ⟨*integer*⟩
  | ⟨*e*⟩ ⟨*integer*⟩

⟨*boolean*⟩ ::= **false** | **true**

⟨*string*⟩ ::= **"** ⟨*char sequence*⟩ **"** | **""**

⟨*char*⟩ ::= **'** ⟨*character*⟩ **'**

⟨*char sequence*⟩ ::= ⟨*character*⟩ | ⟨*character*⟩ ⟨*char sequence*⟩

⟨*character*⟩ ::= Unicode | \\ | **b** | **n** | **f** | **a** | **r** | **t** | **0** | **"** | **'**

⟨*program*⟩ ::= ⟨*operator declaration*⟩ ⟨*program*⟩
  | ⟨*import*⟩ ⟨*program*⟩
  | ⟨*expression sequence*⟩ ⟨*program*⟩
  | ⟨*definition*⟩ ⟨*program*⟩
  | ϵ

⟨*operator declaration*⟩ ::= **defop** ⟨*operator*⟩ ⟨*integer*⟩ ⟨*associativity*⟩

⟨*associativity*⟩ ::= **left** | **right** | **none** | **prefix** | **postfix**

⟨*import*⟩ ::= **import** ⟨*import list*⟩ **end**

⟨*import list*⟩ ::= ϵ | ⟨*string*⟩ ⟨*import list*⟩
| ⟨*type name*⟩ ⟨*import list*⟩

⟨*definition*⟩ ::= **define** ⟨*function name*⟩ ⟨*formal parameters*⟩ **:**
⟨*type*⟩ = ⟨*expression sequence*⟩ **end**

⟨*function name*⟩ ::= ⟨*identifier*⟩ | **(** ⟨*operator*⟩ **)**

⟨*formal parameters*⟩ ::= ⟨*variable name*⟩ ⟨*formal parameters*⟩ | ϵ

⟨*variable name*⟩ ::= ⟨*identifier*⟩ | **_**

⟨*type*⟩ ::= ⟨*function type*⟩
| ⟨*function type*⟩ **,** ⟨*tupe*⟩

⟨*function type*⟩ ::= ⟨*algebraic data type*⟩
| ⟨*function type*⟩ **(->)** ⟨*algebraic data type*⟩

⟨*algebraic data type*⟩ ::= ⟨*type name*⟩ ⟨*atomic type sequence*⟩
| ⟨*atomic type*⟩

⟨*atomic type sequence*⟩ ::= ⟨*atomic type*⟩ ⟨*atomic type sequence*⟩ | ϵ

⟨*atomic type*⟩ ::= ⟨*identifier*⟩ | ⟨*type name*⟩
| **[** ⟨*type*⟩ **]** | **(** ⟨*type*⟩ **)**

⟨*type definition*⟩ ::= **type** ⟨*type name*⟩ ⟨*formal parameters*⟩ **with**
⟨*type constructors*⟩ **end**

⟨*type constructors*⟩ ::= **case** ⟨*type name*⟩ ⟨*atomic type*⟩
| **case** ⟨*type name*⟩

⟨*expression sequence*⟩ ::= ⟨*expression*⟩
| ⟨*expression*⟩ **;** ⟨*expression sequence*⟩

⟨*expression*⟩ ::= ⟨*pattern matching*⟩ | ⟨*let definition*⟩
| ⟨*conditional expression*⟩ | ⟨*tuple expression*⟩

⟨*let definition*⟩ ::= **let** ⟨*variable name*⟩ **:** ⟨*type*⟩ =
⟨*expression sequence*⟩ **in** ⟨*expression sequence*⟩ **end**

⟨*conditional expression*⟩ ::= **if** ⟨*expression sequence*⟩ **then**
  ⟨*expression sequence*⟩ **else** ⟨*expression sequence*⟩ **end**

⟨*pattern matching*⟩ ::= **match** ⟨*expression sequence*⟩ **with**
  ⟨*pattern matching cases*⟩ **end**

⟨*pattern matching cases*⟩ ::= ⟨*pattern case*⟩ ⟨*pattern matching cases*⟩
  | ϵ

⟨*pattern case*⟩ ::= **case** ⟨*pattern*⟩ => ⟨*expression sequence*⟩

⟨*pattern*⟩ ::= ⟨*deconstructor pattern*⟩
  | ⟨*deconstructor pattern*⟩ **,** ⟨*pattern*⟩

⟨*deconstructor pattern*⟩ ::= ⟨*type name*⟩ ⟨*atomic pattern*⟩
  | ⟨*atomic pattern*⟩

⟨*atomic pattern*⟩ ::= ⟨*variable name*⟩ | ⟨*type name*⟩
  | **(** ⟨*pattern*⟩ **)**
  | ⟨*list pattern*⟩
  | ⟨*constant*⟩

⟨*list pattern*⟩ ::= **[** ⟨*pattern*⟩ | ⟨*variable name*⟩ **]**
  | **[** ⟨*pattern*⟩ **]**
  | **[ ]**

⟨*constant*⟩ ::= ⟨*integer*⟩ | ⟨*boolean*⟩ | ⟨*float*⟩
  | **( )** | ⟨*string*⟩ | ⟨*char*⟩

2 Semantics

3 Type system