The Language davinci

BNF-converter

July 9, 2014

This document was automatically generated by the *BNF-Converter*. It was generated together with the lexer, the parser, and the abstract syntax module, which guarantees that the document matches with the implementation of the language (provided no hand-hacking has taken place).

The lexical structure of davinci

Literals

String literals $\langle String \rangle$ have the form `"`x`"`, where x is any sequence of any characters except `"` unless preceded by `\`.

Integer literals $\langle Int \rangle$ are nonempty sequences of digits.

Double-precision float literals $\langle Double \rangle$ have the structure indicated by the regular expression $\langle digit \rangle + \text{`.'}\langle digit \rangle + (\text{`e'`-'}?\langle digit \rangle+)?$ i.e. two sequences of digits separated by a decimal point, optionally followed by an unsigned or negative exponent.

UIdent literals are recognized by the regular expression $\langle upper \rangle(\langle letter \rangle \mid \langle digit \rangle \mid \text{`\_'})*$

LIdent literals are recognized by the regular expression $\langle lower \rangle(\langle letter \rangle \mid \langle digit \rangle \mid \text{`\_'})*$

Wild literals are recognized by the regular expression '_'

Reserved words and symbols

The set of reserved words is the set of terminals appearing in the grammar. Those reserved words that consist of non-letter characters are called symbols, and they are treated in a different way from those that are similar to identifiers. The lexer follows rules familiar from languages like Haskell, C, and Java, including longest match and spacing conventions.

The reserved words used in davinci are the following:

`case`	`false`	`for`
`switch`	`true`	

The symbols used in davinci are the following:

{	}	`*`
`(`	`)`	|
`!?`	`!!`	=>
<−	`??`	`?!`
`@`	`'`	`[`
`]`	`;`	`,`

Comments

Single-line comments begin with `//`.
Multiple-line comments are enclosed with `/*` and `*/`.

The syntactic structure of davinci

Non-terminals are enclosed between ⟨ and ⟩. The symbols ::= (production), | (union) and ϵ (empty rule) belong to the BNF notation. All other symbols are terminals.

⟨*Agent*⟩	::=	{ ⟨*ListAgent*⟩ }
	|	`switch` { ⟨*ListGuardedAgent*⟩ }
	|	`*` ⟨*Channel*⟩
	|	`for` `(` ⟨*ListBinding*⟩ `)` ⟨*Agent*⟩
	|	`for` `(` ⟨*ListBinding*⟩ | ⟨*ListPattern*⟩ `)` ⟨*Agent*⟩
	|	⟨*Variation*⟩ `!?` `(` ⟨*Concretion*⟩ `)`
	|	⟨*Variation*⟩ `!!` `(` ⟨*Concretion*⟩ `)`
	|	⟨*Value*⟩

⟨*GuardedAgent*⟩	::=	`case` ⟨*Pattern*⟩ => ⟨*Agent*⟩

⟨*Abstraction*⟩	::=	`(` ⟨*Variation*⟩ `)` ⟨*Agent*⟩

⟨*Concretion*⟩	::=	`(` ⟨*Information*⟩ `)`

⟨*Binding*⟩	::=	⟨*Pattern*⟩ <− ⟨*Channel*⟩ `??` `(` ⟨*Pattern*⟩ `)`
	|	⟨*Pattern*⟩ <− ⟨*Channel*⟩ `?!` `(` ⟨*Pattern*⟩ `)`

```
⟨Pattern⟩  ::=  ⟨Symbol⟩ ( ⟨ListPattern⟩ )
            |   ⟨Variation⟩
            |   ⟨Value⟩
            |   ⟨Lyst⟩

⟨Channel⟩  ::=  ⟨LIdent⟩
            |   ⟨Variation⟩
            |   @ ' ⟨Agent⟩ '

⟨Symbol⟩  ::=  ⟨LIdent⟩

⟨Variation⟩  ::=  ⟨UIdent⟩

⟨Information⟩  ::=  ⟨Variation⟩
                |   ⟨Agent⟩

⟨Lyst⟩  ::=  [ ]
         |   [ ⟨ListPattern⟩ ]
         |   [ ⟨ListPattern⟩ | ⟨Lyst⟩ ]
         |   [ ⟨ListPattern⟩ | ⟨Variation⟩ ]

⟨Value⟩  ::=  ⟨Duality⟩
          |   ⟨String⟩
          |   ⟨Integer⟩
          |   ⟨Double⟩
          |   @ ' ⟨Agent⟩ '

⟨Duality⟩  ::=  true
            |   false

⟨ListAgent⟩  ::=  ϵ
              |   ⟨Agent⟩
              |   ⟨Agent⟩ ; ⟨ListAgent⟩

⟨ListGuardedAgent⟩  ::=  ϵ
                     |   ⟨GuardedAgent⟩
                     |   ⟨GuardedAgent⟩ ; ⟨ListGuardedAgent⟩

⟨ListPattern⟩  ::=  ⟨Pattern⟩
                |   ⟨Pattern⟩ , ⟨ListPattern⟩

⟨ListBinding⟩  ::=  ϵ
                |   ⟨Binding⟩
                |   ⟨Binding⟩ ; ⟨ListBinding⟩
```