Syntax of the ALPHA language

Florent de Dinechin, Tanguy Risset, Patrice Quinton, Doran K. Wilde

May 18, 1995

1 Meta Syntax

*phrase**	===	zero or more repetitions of *phrase*.
phrase1 | *phrase2*	===	alternation, either *phrase1* or *phrase2*.
[...]	===	optional phrase.
(...)	===	syntactic grouping.
`bold`	===	a terminal.
Italic	===	a non-terminal.

2 Systems

Program	::	*PDecl* *PDecl* `*`
PDecl	::	*SystemDecl*
SystemDecl	::	`system` *Name* *[* `:` *ParamDecl* *]* `(` *InputDeclList* `)` `returns` `(` *OutputDeclList* `)` `;` *[* `var` *LocalDeclList* `;` *]* *Equationblock* `;`
Name	::	*Identifier*
ParamDecl	::	*Domain*
InputDeclList	::	*VarDeclList*
OutputDeclList	::	*VarDeclList*
LocalDeclList	::	*VarDeclList*

3 Declarations of variables

VarDeclList	::	*VarDeclList* `*`

```
VarDeclaration    ::  IdentifierList : [ Domain of ] ScalarType ;
ScalarType        ::  integer | real | boolean
```

4 Domains

```
Domain            ::  { IndexList | ConstraintList }
                    | Domain | Domain
                    | Domain & Domain
                    | Domain .AffineFunction
                    | ~ Domain
                    | Domain .convex
                    | ( Domain )

IndexList         ::  [ IndexList , ] Identifier

ConstraintList    ::  [ ConstraintList ; ] Constraint
Constraint        ::  IncreasingSeq | DecreasingSeq | EqualitySeq
IncreasingSeq     ::  ( IncreasingSeq | IndexExpList ) ( < | <= ) IndexExpList
DecreasingSeq     ::  ( DecreasingSeq | IndexExpList ) ( > | >= ) IndexExpList
EqualitySeq       ::  ( EqualitySeq | IndexExpList ) = IndexExpList
```

5 Equations

```
Equationblock     ::  let EquationList tel
EquationList      ::  [ EquationList ] Equation
Equation          ::  Identifier [ IndexList ] = Expression ;
                    | Identifier = Expression ;
                    | use [ ExtensionDomain ] Identifier [ .ParamAssignation ]
                        ( InputList )
                        returns ( IdentifierList ) ;

ParamAssignation  ::  AffineFunction

InputList         ::  [ InputList , ] Expression

ExtensionDomain   ::  Domain
```

6 Expressions

```
Expression        ::  case ExpressionList esac
                    | if Expression then Expression else Expression
```

```
                   | Domain :Expression
                   | Expression .AffineFunction
                   | Expression [ IndexExpList ]
                   | Expression BinaryOp Expression
                   | BinaryOp ( Expression , Expression )
                   | UnaryOp Expression
                   | reduce ( CommutativeOp , AffineFunction , Expression )
                   | ( Expression )
                   | Identifier
                   | Constant

ExpressionList  ::   [ ExpressionList ] Expression ;

BinaryOp        ::   CommutativeOp | RelativeOp | - | div | mod
CommutativeOp   ::   + | * | and | or | xor | min | max
RelativeOp      ::   = | <> | < | <= | > | >=
UnaryOp         ::   - | not | sqrt

Constant        ::   IntegerConstant | RealConstant | BooleanConstant
```

7 Dependance Functions and Index Expressions

```
AffineFunction   ::   ( IndexList -> IndexExpList )
IndexExpList     ::   [ IndexExpList , ] IndexExpression | IndexExpression
IndexExpression  ::   IndexExpression ( + | - ) IndexTerm | [ - ] IndexTerm
IndexTerm        ::   IntegerConstant Identifier | IntegerConstant | Identifier
```

8 Terminals

```
IntegerConstant  ::   [ - ] Number
RealConstant     ::   [ - ] Number .Number
BooleanConstant  ::   true | false |True | False
Number           ::   Digit Digit *
Digit            ::   0 | 1 |...| 9
Identifier       ::   Letter ( Letter | Digit ) *
Letter           ::   a |...| z | A |...| Z | _
```