5. ALF language construction principles and overview

Add lead-in text

This section discusses the ALF language construction principles and gives an overview of the language features.

5.1 ALF meta-language

The following syntax construction establishes an ALF meta-language.

```
ALF_statement ::=
      ALF_type [ALF_name ] [ = ALF_value ] ALF_statement_termination
ALF_statement_termination ::=
      ;
    | { { ALF_value | : | ; } }
    | { { ALF_statement } }
ALF_type ::=
      non_escaped_identifier [ index ]
    | @
    | :
ALF_name ::=
      identifier [ index ]
    | control_expression
ALF_value ::=
      identifier
    | number
    | arithmetic_expression
    | boolean_expression
    | control_expression
```

Syntax 1—syntax construction for ALF meta-language

An *ALF statement* uses the delimiters "**;**", "**{**" and "**}**" to indicate its termination.

The *ALF type* is defined by a *keyword* (see section ...) eventually in conjunction with an *index* (see section ...) or by the delimiter "**@**" or "**:**". The usage of keyword, index, or delimiter as ALF type is defined by ALF language rules concerning the particular ALF type.

The *ALF name* is defined by an *identifier* (see section ...) eventually in conjunction with an index or by a *control expression* (see section ...). Depending on the ALF type, the ALF name is mandatory or optional or not applicable. The usage of identifier, index, or control expression as ALF name is defined by ALF language rules concerning the particular ALF type.

The *ALF value* is defined by an identifier, a *number* (see section ...), an *arithmetic expression* (see section ...), a *boolean expression* (see section ...), or a control expression. Depending on the type of the ALF statement, the ALF value is mandatory or optional or not applicable. The usage of identifier, number, arithmetic expression, boolean expression or control expression as ALF value is defined by ALF language rules concerning the particular ALF type.

An ALF statement can contain one or more other ALF statements. The former is called *parent* of the latter. Conversely, the latter is called *child* of the former. An ALF statement with child is called a *compound* ALF statement.

An ALF statement containing one or more ALF values, eventually interspersed with the delimiters "**;**" or "**:**", is called a *semi-compound* ALF statement. The items between the delimiters "**{**" and "**}**" are called *contents* of the

ALF statement. The usage of the delimiters "**;**" or "**:**" within the contents of an ALF statement is defined by ALF language rules concerning the particular ALF statement.

An ALF statement without child is called an *atomic* ALF statement. An ALF statement which is either compound or semi-compound is called a *non-atomic* ALF statement.

Examples

a) ALF statement describing an unnamed object without value:

```
ARBITRARY_ALF_TYPE {
    // put children here
}
```

b) ALF statement describing an unnamed object with value:

```
ARBITRARY_ALF_TYPE = arbitrary_ALF_value;
```

or

```
ARBITRARY_ALF_TYPE = arbitrary_ALF_value {
    // put children here
}
```

c) ALF statement describing a named object without value:

```
ARBITRARY_ALF_TYPE arbitrary_ALF_name;
```

or

```
ARBITRARY_ALF_TYPE arbitrary_ALF_name {
    // put children here
}
```

d) ALF statement describing a named object with value:

```
ARBITRARY_ALF_TYPE arbitrary_ALF_name = arbitrary_ALF_value;
```

or

```
ARBITRARY_ALF_TYPE arbitrary_ALF_name = arbitrary_ALF_value {
    // put children here
}
```

5.2 Categories of ALF statements

In this section, the terms *statement*, *type*, *name*, *value* are used for shortness in lieu of *ALF statement*, *ALF name*, *ALF value*, respectively.

Statements are divided into the following categories: *generic object*, *library-specific object*, *arithmetic model*, *arithmetic submodel*, *arithmetic model container*, *geometric model*, *annotation*, *annotation container*, and *auxiliary statement*.

Table 2—Categories of ALF statements

category	purpose	syntax particularity
generic object	provide a definition for use within other ALF statements	Statement is atomic, semi-compound or compound. Name is mandatory. Value is either mandatory or not applicable.

Table 2—Categories of ALF statements

category	purpose	syntax particularity
library-specific object	describe the contents of a IC technology library	Statement is atomic or compound. Name is mandatory. Value does not apply. Category of parent is exclusively *library-specific object*
arithmetic model	describe an abstract mathematical quantity that can be calculated and eventually measured within the design of an IC	Statement is atomic or compound. Name is optional. Value is mandatory, if atomic.
arithmetic submodel	describe an arithmetic model under a specific measurement condition	Statement is atomic or compound. Name does not apply. Value is mandatory, if atomic. Category of parent is exclusively *arithmetic model*
arithmetic model container	provide a context for an arithmetic model	Statement is compound. Name and value do not apply. Category of child is exclusively *arithmetic model*
geometric model	describe an abstract geometrical form used in physical design of an IC	Statement is semi-compound or compound. Name is optional. Value does not apply.
annotation	provide a qualifier or a set of qualifiers for an ALF statement	Statement is atomic, semi-compound or compound. Name does not apply. Value is mandatory, if atomic or compound. Value does not apply, if semi-compound. Category of child is exclusively *annotation*
annotation container	provide a context for an annotation	Statement is compound. Name and value do not apply. Category of child is exclusively *annotation*
auxiliary statement	provide an additional description within the context of a library-specific object, an arithmetic model, an arithmetic submodel, geometric model or another auxliary statement	dependent on subcategory

The following figure illustrates the parent/child relationship between categories of statements.

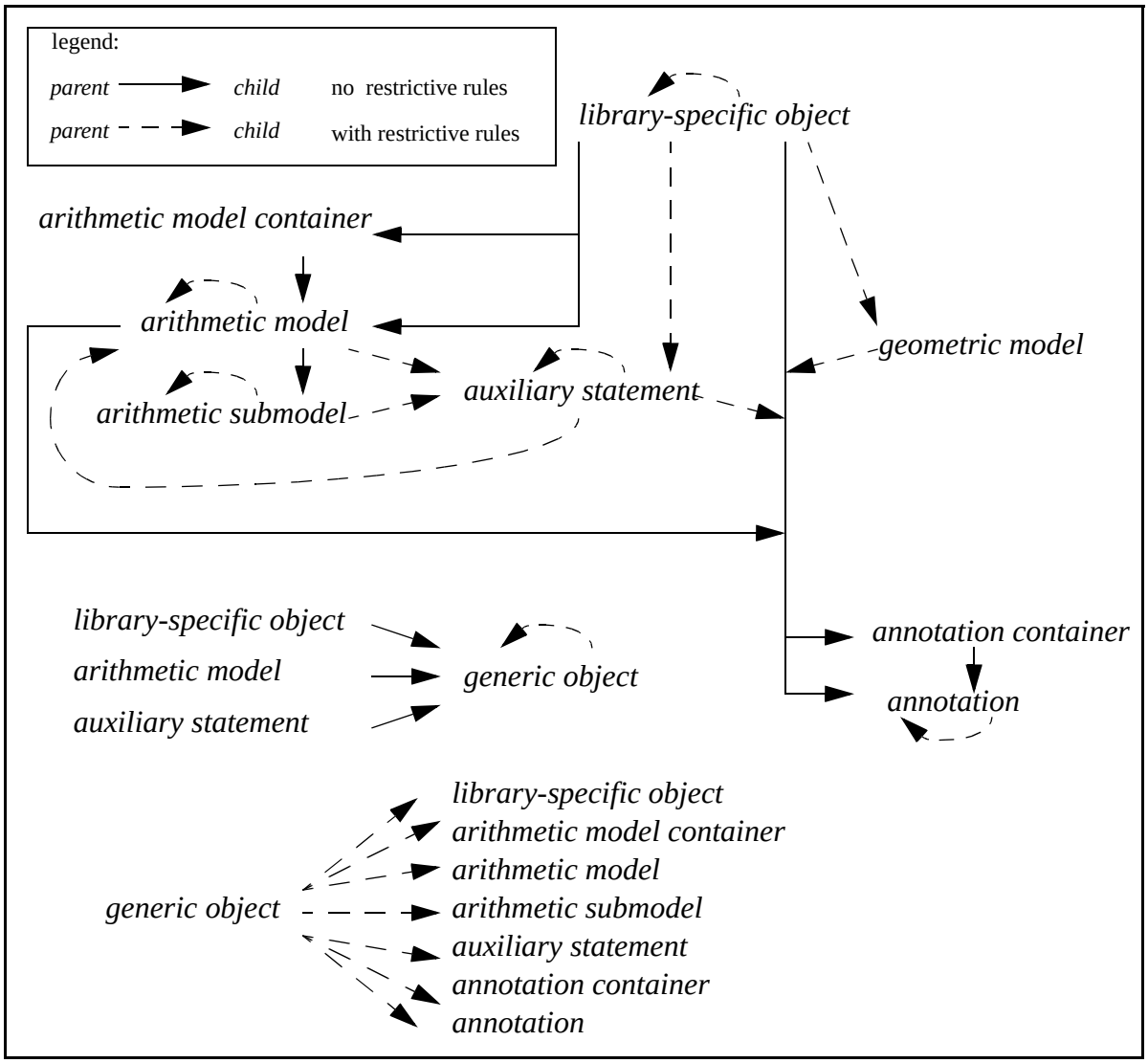

Figure 2—Parent/child relationship between ALF statements

More detailed rules for parent/child relationships for particular types of statements apply.

5.3 Types and subcategories of ALF statements

This section provides an overview of the types of ALF statements and rules for parent/child relationships between types. Most of the types are associated with predefined keywords. The keywords in ALF are case-insensitive. However, uppercase is used for keywords throughout this section for clarity.

5.3.1 Statements within the category “generic object” and “library-specific object”

Statements with mandatory name are called *objects*, i.e., *generic object* and *library-specific object*.

The following table lists the keywords and items in the category *generic object*. The keywords used in this category are called *generic keywords.*

Table 3—Generic objects

keyword	item	section
ALIAS	alias	
CONSTANT	constant	
INCLUDE	include	
CLASS	class declaration	
GROUP	group declaration	
KEYWORD	keyword declaration	
TEMPLATE	template declaration	

The following table lists the keywords and items in the category *library-specific object.* The keywords used in this category are called *library-specific keywords*.

Table 4—Library-specific objects

keyword	item	section
LIBRARY	library	
SUBLIBRARY	sublibrary	
CELL	cell	
PRIMITIVE	primitive	
WIRE	wire	
PIN	pin	
PINGROUP	pin group	
VECTOR	vector	
NODE	node	
LAYER	layer	
VIA	via	
RULE	rule	
ANTENNA	antenna	

Table 4—Library-specific objects

keyword	item	section
SITE	site	
ARRAY	array	
BLOCKAGE	blockage	
PORT	port	
PATTERN	pattern	
REGION	region	new proposal for IEEE

The following figure illustrates the parent/child relationship between statements within the category *library-specific object*.

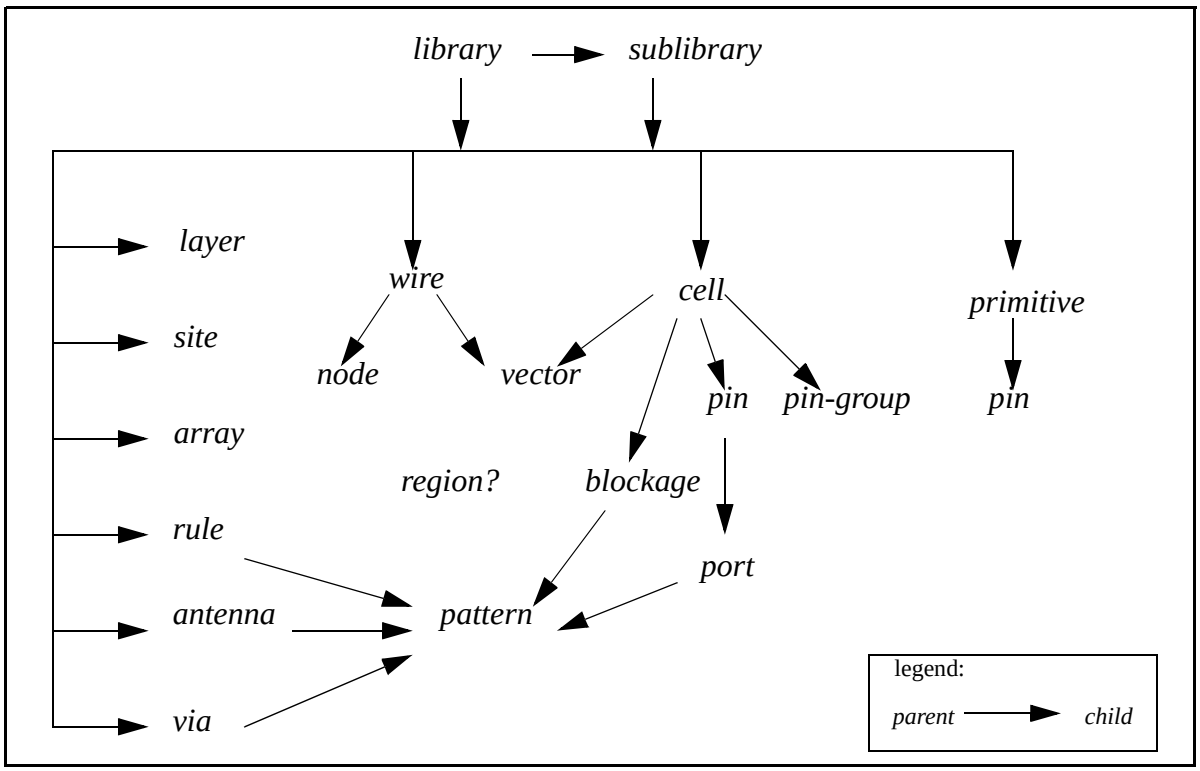

Figure 3—Parent/child relationship amongst library-specific objects

A parent can have multiple library-specific objects of the same type as children. Each child is distinguished by name.

5.3.2 Auxiliary statements with predefined keywords

Auxiliary statements with predefined keywords are devided in the following subcategories: *singular statement* and *plural statement*.

Auxiliary statements with predefined keywords and without name are called *singular statements.* Auxiliary statements with predefined keywords and with name, yet without value, are called *plural statements*.

The following table lists the singular statements.

Table 5—Singular statements

keyword	item	value	complexity	section
FUNCTION	function	N/A	compound	
TEST	test	N/A	compound	
RANGE	range	N/A	semi-compound	
FROM	from	N/A	compound	
TO	to	N/A	compound	
VIOLATION	violation	N/A	compound	
HEADER	header	N/A	compound (or semi-compound?)	
TABLE	table	N/A	semi-compound	
EQUATION	equation	N/A	semi-compound	
BEHAVIOR	behavior	N/A	compound	
STRUCTURE	structure	N/A	compound	
NON_SCAN_CELL	non-scan cell	optional	compound or semi-compound	
ARTWORK	artwork	mandatory	compound or atomic	
PULL	pull	optional	compound or atomic	

The following table lists the plural statements.

Table 6—Plural statements

keyword	item	name	complexity	section
STATETABLE	state table	optional	semi-compound	
@	control statement	mandatory	compound	

Table 6—Plural statements

keyword	item	name	complexity	section
:	alternative control statement	mandatory	compound	

The following figure illustrates the parent/child relationship involving singular statements and plural statements.

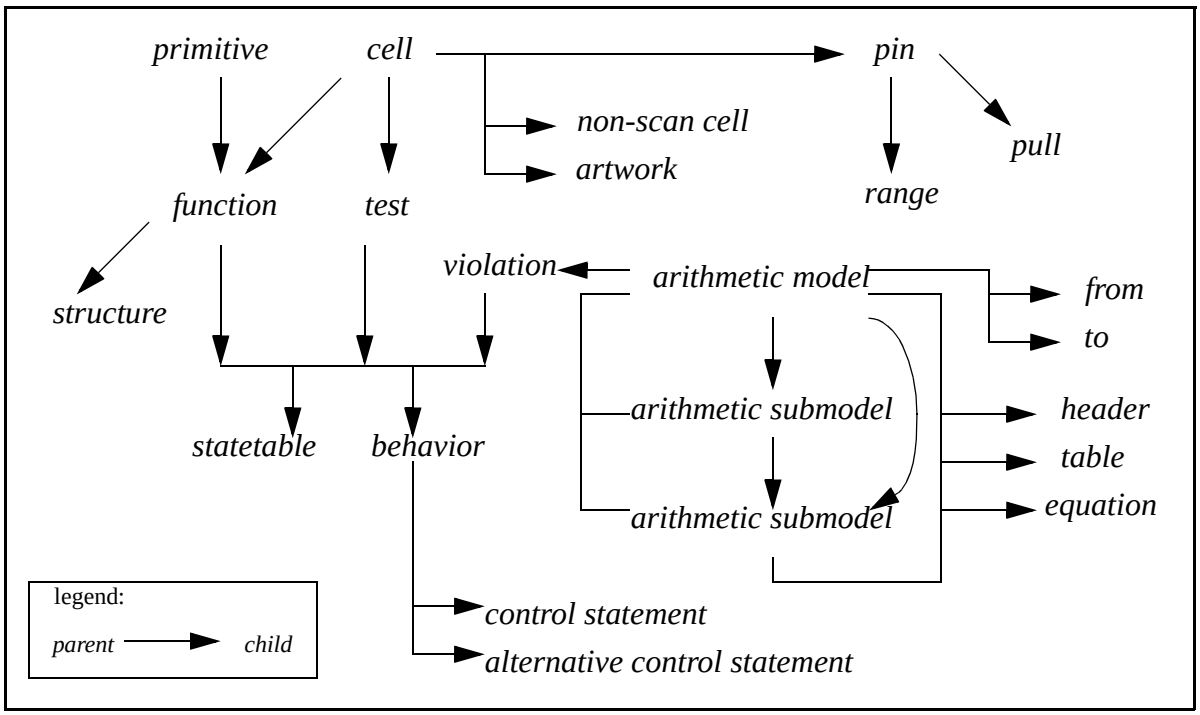

Figure 4—Parent/child relationship involving singular statements and plural statements

A parent can have at most one child of a particular type in the category singular statements, but multiple children of a particular type in the category plural statements.

5.3.3 Auxiliary statements without predefined keywords

Auxiliary statements without predefined keywords use the name of an object as keyword. Such statements are devided in the following subcategories: *instantiation statement* and *assignment statement*.

Compound or semi-compound statements using the name of an object as keyword are called *instantiation statements*. Their purpose is to specify an instance of the object.

The following table lists the instantiation statements.

Table 7—Instantiation statements

item	name	value	section
cell instantiation	optional	N/A	
primitive instantiation	optional	N/A	
template instantiation	N/A	optional	
wire instantiation	mandatory	N./A	proposed for IEEE

Atomic statements without name using an identifier as keyword which has been defined within the context of another object are called assignment statements. A value is mandatory for assignment statements, as their purpose is to assign a value to the identifier. Such an identifier is called a *variable*.

The following table lists the assignment statements.

Table 8—Assignment statements

item	section
pin assignment	
boolean assignment	
arithmetic assignment	

The following figure illustrates the parent/child relationship involving instantiation and assigment statements.

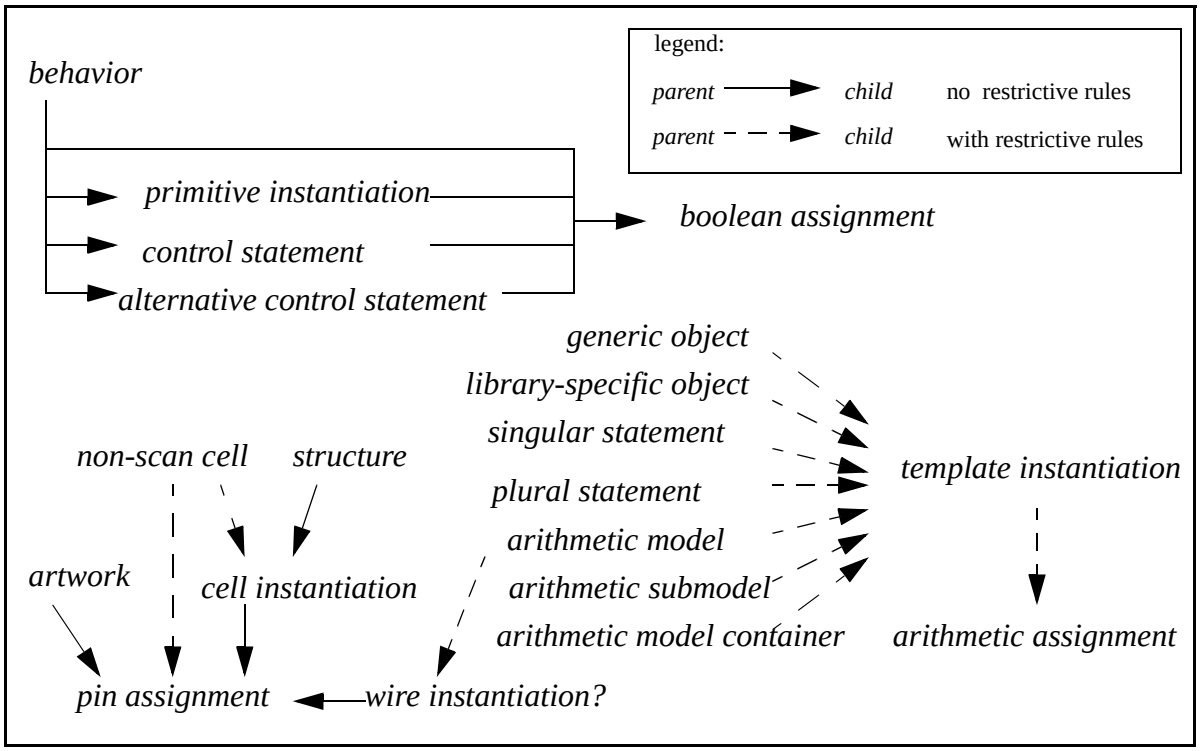

Figure 5—Parent/child relationship involving instantiation and assignment statements

A parent can have multiple children using the same keyword in the category instantiation statement, but at most one child using the same variable in the category assignment statement.

5.3.4 Statements within other categories

Multiple keywords are predefined in the categories *arithmetic model*, *arithmetic model container*, *arithmetic submodel*, *annotation*, *annotation container*, and *geometric model*. Their semantics are established within the context of their parent. Therefore they are called *context-sensitive keywords*. In addition, the ALF language allows additional definition of keywords in these categories.

The following table provides a reference to sections where more definitions about these categories can be found.

Table 9—Other categories of ALF statements

item	section
arithmetic model	
arithmetic submodel	
arithmetic model container	
annotation	
annotation container	

Table 9—Other categories of ALF statements

item	section
geometric model	

There exist predefined keywords with no predefined context-sensitive semantics in the category *annotation* and *annotation container.* They are called *generic keywords*, like the keywords for *generic objects*.

The following table lists the generic keywords in the category annotation and annotation container.

Table 10—Annotations and annotation containers with generic keyword

keyword	item / subcategory	section
PROPERTY	one_level_annotation_container	
ATTRIBUTE	multi_value_annotation	
INFORMATION	one_level_annotation_container	