1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

Concrete syntax

/Reference manual

1. Contents of this page

- Introduction
- ISO Standard concrete syntax
  - Formal definition of concrete syntax
  - User-defined operators
  - Additional syntactic restrictions
- CADiZ-specific concrete syntax

2. Introduction

The concrete syntax defines which sequences of tokens (or lexemes) are acceptable as sentences of the Z language.

The concrete syntax is formalized below using syntactic metalanguage. *MixedCase* is used for the names of syntactic rules. *CAPITALISED* names are tokens of the lexis, as are mathematical symbols and symbols ending in $-tok$.

1 Contents of this page

2 Introduction

3 ISO Standard . . .

3.1 Formal . . .

3.2 User-defined . . .

3.3 Additional . . .

4 CADiZ-specific . . .

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

3. ISO Standard concrete syntax

3.1. Formal definition of concrete syntax

This formal definition is public domain material, and appears as it appears in ISO/IEC 13568:2002 (the Z standard).

$$
\begin{array}{lcll}
Specification & = & \{Section\} & \text{(* sectioned specification *)} \\
 & | & \{Paragraph\} & \text{(* anonymous specification *)} \\
 & ; & &
\end{array}
$$

$$
\begin{array}{lcl}
Section & = & ZED, section, NAME, parents, [NAME, \{, -tok, NAME\}], END, \{Paragr \\
 & | & ZED, section, NAME, END, \{Paragraph\} \\
 & ; &
\end{array}
$$

$$
\begin{array}{lcll}
Paragraph & = & ZED, [-tok, NAME, \{, -tok, NAME\},] - tok, END & \\
 & | & AX, SchemaText, END & \\
 & | & SCH, NAME, SchemaText, END & \\
 & | & GENAX, [-tok, Formals,] - tok, SchemaText, END & \text{(* ger} \\
 & | & GENSCH, NAME, [-tok, Formals,] - tok, SchemaText, END & \text{(*} \\
 & | & ZED, DeclName, ==, Expression, END & \\
 & | & ZED, NAME, [-tok, Formals,] - tok, ==, Expression, END & \text{(* ge} \\
 & | & ZED, GenName, ==, Expression, END & \text{(*} \\
 & | & ZED, FreeType, \{ampersand, FreeType\} & \\
 & | & ZED, \vdash ?\ , Predicate, END & \\
 & | & ZED, [-tok, Formals,] - tok, \vdash ?\ , Predicate, END & \\
 & | & ZED, OperatorTemplate, END & \\
 & ; & &
\end{array}
$$

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

$$Freetype \quad = \quad NAME, ::=, Branch, \{| -tok, Branch\}; \quad \text{(* free type *)}$$

$$Branch \quad = \quad DeclName, [\langle\!\langle, Expression, \rangle\!\rangle]; \quad \text{(* element or injection *)}$$

$$Formals \quad = \quad NAME, \{, -tok, NAME\}; \quad \text{(* generic parameters *)}$$

$$\begin{array}{rcll}
Predicate & = & Predicate, NL, Predicate & \text{(* newline conjunction *)} \\
& | & Predicate, ; \ -tok, Predicate & \text{(* semicolon conjunction *)} \\
& | & \forall, SchemaText, \bullet, Predicate & \text{(* universal quantification *)} \\
& | & \exists, SchemaText, \bullet, Predicate & \text{(* existential quantification *)} \\
& | & \exists_1, SchemaText, \bullet, Predicate & \text{(* unique existential quantification *)} \\
& | & Predicate, \Leftrightarrow, Predicate & \text{(* equivalence *)} \\
& | & Predicate, \Rightarrow, Predicate & \text{(* implication *)} \\
& | & Predicate, \vee, Predicate & \text{(* disjunction *)} \\
& | & Predicate, \wedge, Predicate & \text{(* conjunction *)} \\
& | & \neg \ , Predicate & \text{(* negation *)} \\
& | & Relation & \text{(* relation operator application *)} \\
& | & Expression & \text{(* schema predicate *)} \\
& | & true & \text{(* truth *)} \\
& | & false & \text{(* falsity *)} \\
& | & (-tok, Predicate,) - tok & \text{(* parenthesized predicate *)} \\
& ; & &
\end{array}$$

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

$$
\begin{array}{rcl}
Expression & = & \forall, SchemaText, \bullet, Expression \quad (* s \\
& | & \exists, SchemaText, \bullet, Expression \quad (* sc \\
& | & \exists_1, SchemaText, \bullet, Expression \quad (* schema u \\
& | & \lambda, SchemaText, \bullet, Expression \\
& | & \mu, SchemaText, \bullet, Expression \\
& | & \mathrm{let}, DeclName, ==, Expression, \{;\ -tok, DeclName, ==, Expression\}, \bullet, Expression \\
& | & Expression, \Leftrightarrow, Expression \\
& | & Expression, \Rightarrow, Expression \\
& | & Expression, \vee, Expression \\
& | & Expression, \wedge, Expression \\
& | & \neg\ , Expression \\
& | & \mathrm{if}, Predicate, \mathrm{then}, Expression, \mathrm{else}, Expression \\
& | & Expression, \fatsemi, Expression \\
& | & Expression, \gg, Expression \\
& | & Expression, \upharpoonright, (-tok, DeclName, \{, -tok, DeclName\},) - tok \\
& | & Expression, \uparrow, Expression \\
& | & \mathrm{pre}, Expression \\
& | & Expression, \times, Expression, \{\times, Expression\} \\
& | & \mathbb{P}, Expression \\
& | & Application \quad (* function \\
& | & Expression, Expression \\
& | & Expression, STROKE \\
& | & Expression, [-tok, DeclName, /, DeclName, \{, -tok, DeclName, /, DeclName\},] - tok \\
& | & Expression, ., RefName \\
& | & Expression, ., NUMERAL \\
& | & \theta, Expression, \{STROKE\} \\
& | & RefName \\
& | & RefName, [-tok, Expression, \{, -tok, Expression\},] - tok \\
& | & NUMERAL \\
& | & \{-tok, [Expression, \{, -tok, Expression\}], \} - tok \\
& | & \{-tok, SchemaText, \bullet, Expression, \} - tok \\
& | & ((\{-tok, SchemaText, \} - tok) - (\{-tok, \} - tok)) - (\{-tok, Expression, \} - tok) \quad (* ch \\
& | & ([-tok, SchemaText,] - tok) - ([-tok, Expression,] - tok) \\
& | & \langle\!|, [DeclName, ==, Expression, \{, -tok, DeclName, ==, Expression\}], |\!\rangle \\
& | & (-tok, Expression, , -tok, Expression, \{, -tok, Expression\},) - tok \\
& | & (-tok, \mu, SchemaText,) - tok \quad (* ch \\
& | & (-tok, Expression,) - tok \\
; &&
\end{array}
$$

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

$$SchemaText = [DeclPart], [| -tok, Predicate];$$

$$DeclPart = Declaration, \{(; -tok \mid NL), Declaration\};$$

$$\begin{aligned} Declaration = {} & DeclName, \{, -tok, DeclName\}, :, Expression \\ \mid {} & DeclName, ==, Expression \\ \mid {} & Expression \\ ; & \end{aligned}$$

$$\begin{aligned} OperatorTemplate = {} & \text{relation}, Template \\ \mid {} & \text{function}, CategoryTemplate \\ \mid {} & \text{generic}, CategoryTemplate \\ ; & \end{aligned}$$

$$\begin{aligned} CategoryTemplate = {} & PrefixTemplate \\ \mid {} & PostfixTemplate \\ \mid {} & Prec, Assoc, InfixTemplate \\ \mid {} & NofixTemplate \\ ; & \end{aligned}$$

$$Prec = NUMERAL;$$

$$\begin{aligned} Assoc = {} & \text{leftassoc} \\ \mid {} & \text{rightassoc} \\ ; & \end{aligned}$$

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

$$
\begin{array}{lcl}
Template & = & PrefixTemplate \\
& | & PostfixTemplate \\
& | & InfixTemplate \\
& | & NofixTemplate \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
PrefixTemplate & = & (-tok, PrefixName,) - tok \\
& | & (-tok, \mathbb{P}, \_,) - tok \\
& ; &
\end{array}
$$

$$PostfixTemplate = (-tok, PostfixName,) - tok;$$

$$InfixTemplate = (-tok, InfixName,) - tok;$$

$$NofixTemplate = (-tok, NofixName,) - tok;$$

$$
\begin{array}{lcl}
DeclName & = & NAME \\
& | & OpName \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
RefName & = & NAME \\
& | & (-tok, OpName,) - tok \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
OpName & = & PrefixName \\
& | & PostfixName \\
& | & InfixName \\
& | & NofixName \\
& ; &
\end{array}
$$

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

$$
\begin{array}{lcl}
PrefixName & = & PRE, \_ \\
& | & PREP, \_ \\
& | & L, \{\_, ES \mid , , , SS\}, (\_, ERE \mid , , , SRE), \_ \\
& | & LP, \{\_, ES \mid , , , SS\}, (\_, EREP \mid , , , SREP), \_ \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
PostfixName & = & \_, POST \\
& | & \_, POSTP \\
& | & \_, EL, \{\_, ES \mid , , , SS\}, (\_, ER \mid , , , SR) \\
& | & \_, ELP, \{\_, ES \mid , , , SS\}, (\_, ERP \mid , , , SRP) \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
InfixName & = & \_, I, \_ \\
& | & \_, IP, \_ \\
& | & \_, EL, \{\_, ES \mid , , , SS\}, (\_, ERE \mid , , , SRE), \_ \\
& | & \_, ELP, \{\_, ES \mid , , , SS\}, (\_, EREP \mid , , , SREP), \_ \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
NofixName & = & L, \{\_, ES \mid , , , SS\}, (\_, ER \mid , , , SR) \\
& | & LP, \{\_, ES \mid , , , SS\}, (\_, ERP \mid , , , SRP) \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
GenName & = & PrefixGenName \\
& | & PostfixGenName \\
& | & InfixGenName \\
& | & NofixGenName \\
& ; &
\end{array}
$$

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

$$
\begin{aligned}
PrefixGenName &= PRE, NAME \\
&\mid L, \{NAME, (ES \mid SS)\}, NAME, (ERE \mid SRE), NAME \\
&;
\end{aligned}
$$

$$
\begin{aligned}
PostfixGenName &= NAME, POST \\
&\mid NAME, EL, \{NAME, (ES \mid SS)\}, NAME, (ER \mid SR) \\
&;
\end{aligned}
$$

$$
\begin{aligned}
InfixGenName &= NAME, I, NAME \\
&\mid NAME, EL, \{NAME, (ES \mid SS)\}, NAME, (ERE \mid SRE), NAME \\
&;
\end{aligned}
$$

$$
NofixGenName = L, \{NAME, (ES \mid SS)\}, NAME, (ER \mid SR);
$$

$$
\begin{aligned}
Relation &= PrefixRel \\
&\mid PostfixRel \\
&\mid InfixRel \\
&\mid NofixRel \\
&;
\end{aligned}
$$

$$
\begin{aligned}
PrefixRel &= PREP, Expression \\
&\mid LP, ExpSep, (Expression, EREP \mid ExpressionList, SREP), Expression \\
&;
\end{aligned}
$$

$$
\begin{aligned}
PostfixRel &= Expression, POSTP \\
&\mid Expression, ELP, ExpSep, (Expression, ERP \mid ExpressionList, SRP) \\
&;
\end{aligned}
$$

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

$$
\begin{array}{lcl}
InfixRel & = & Expression, (\in\!|= -tok \mid IP), Expression, \{(\in\!|= -tok \mid IP), Expression\} \\
& \mid & Expression, ELP, ExpSep, (Expression, EREP \mid ExpressionList, SREP), \\
; & &
\end{array}
$$

$$
NofixRel \quad = \quad LP, ExpSep, (Expression, ERP \mid ExpressionList, SRP);
$$

$$
\begin{array}{lcl}
Application & = & PrefixApp \\
& \mid & PostfixApp \\
& \mid & InfixApp \\
& \mid & NofixApp \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
PrefixApp & = & PRE, Expression \\
& \mid & L, ExpSep, (Expression, ERE \mid ExpressionList, SRE), Expression \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
PostfixApp & = & Expression, POST \\
& \mid & Expression, EL, ExpSep, (Expression, ER \mid ExpressionList, SR) \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
InfixApp & = & Expression, I, Expression \\
& \mid & Expression, EL, ExpSep, (Expression, ERE \mid ExpressionList, SRE), Expr \\
& ; &
\end{array}
$$

$$
NofixApp \quad = \quad L, ExpSep, (Expression, ER \mid ExpressionList, SR);
$$

$$
ExpSep \quad = \quad \{Expression, ES \mid ExpressionList, SS\};
$$

$$
ExpressionList \quad = \quad [Expression, \{, -tok, Expression\}];
$$

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

3.2. User-defined operators

In a *Template*, each operator token shall be a name with no *STROKE*s. That name can take part in the templates of several operators that are in scope together, so long as it has the same operator token in all of them and all of them have the same precedence. Every template shall differ in some way from all other templates that are in scope together.

All prefix operators are right associative. All postfix operators are left associative. Different associativities shall not be used at the same precedence level by operator template paragraphs in the same scope.

The following table defines the relative precedences of the productions of *Expression* and *Predicate*. The rows in the table are ordered so that the entries with higher precedence (and so that bind more strongly) appear nearer the top of the table than those with lower precedence (that bind more weakly). Associativity has significance only in determining the nesting of applications involving non-associative operators of the same precedence. Explicitly-defined function and generic operator applications have a range of precedences specified numerically in the corresponding operator template paragraph. Cartesian product expressions have precedence value 8 within that numeric range.

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

Productions	Associativity
binding construction	
binding selection, tuple selection	
schema renaming	
schema decoration	
application	left
postfix function and generic operator application	
powerset, prefix function and generic operator application	
Cartesian product, infix function and generic operator application	
schema precondition	
schema projection	left
schema hiding	
schema piping	
schema composition	
conditional	
substitution expression	
definite description	
function construction	
relation operator application	
negation	
conjunction	
disjunction	
implication	right
equivalence	
universal, existential and unique existential quantifications	
newline conjunction, semicolon conjunction	

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

3.3. Additional syntactic restrictions

STROKE is used in three contexts: within *NAME*s, in binding construction expressions, and in schema decoration expressions. The condition for a *STROKE* to be considered as part of a *NAME* was given in the lexis. Other *STROKE*s are considered to be parts of binding construction expressions if they can be when interpreted from left to right. The schema decoration expression interpretation is considered last.

A predicate can be just an expression, yet the same logical operators ($\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$, $\forall$, $\exists$, $\exists_1$) can be used in both expressions and predicates. Where a predicate is expected, and one of these logical operators is used on expressions, there is an ambiguity: either the whole logical operation is an expression and that expression is used as a predicate, or the whole logical operation is a predicate involving expressions each used as a predicate. This ambiguity is benign, as both interpretations have the same precedence, associativity and meaning.

4. CADiZ-specific concrete syntax

Cadiz permits the declaration part of a substitution expression to be an arbitrary *SchemaText*, just like a definite description (μ) expression, rather than only constant ($==$) declarations.

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

$$\begin{array}{lcl} Expression & = & \mathrm{let}, SchemaText, \bullet, Expression \\ & | & (-tok, \mathrm{let}, SchemaText,) - tok \\ & ; & \end{array}$$

Cadiz offers exclusive or ($\veebar$) as an additional logical operator in *Predicate* and *Expression*. It has the same precedence and associativity as inclusive or.

$$\begin{array}{lcl} Predicate & = & Predicate, \veebar, Predicate; \\ Expression & = & Expression, \veebar, Expression; \end{array}$$

A string literal may appear as an *Expression*.

$$Expression \quad = \quad STRING;$$

A comment to be typeset as part of the specification can be included as a predicate.

$$Predicate \quad = \quad comment, STRING;$$

A constant declaration can declare more than one name for the constant.

$$Declaration \quad = \quad DeclName, \{, -tok, DeclName\}, ==, Expression;$$

The syntax of conjectures is replaced by a syntax of sequents, which includes conjectures as a special case.

$$Paragraph \quad = \quad [[-tok, Formals,] - tok], [DeclPart, \{\dagger, DeclPart\}], [| -tok, Predicate,$$

Cadiz adds the following schema operators: overriding, postcondition and undecoration.

1 Contents of this page
2 Introduction
3 ISO Standard . . .
3.1 Formal . . .
3.2 User-defined . . .
3.3 Additional . . .
4 CADiZ-specific . . .

$$\begin{array}{rcl} Expression & = & Expression, \oplus, Expression \\ & | & post, Expression \\ & | & undecor, STROKE, Expression \\ & ; & \end{array}$$

A generic parameter list may distinguish parameters that may be instantiated only with schemas. These are used in the definition of type-constrained generics.

$$Formals \quad = \quad [NAME, \{, -tok, NAME\}], [\dagger, NAME, \{, -tok, NAME\}];$$

Branch is redefined to permit mixfix syntax for injections in free types.

$$\begin{array}{rcl} Branch & = & DeclName, \langle\!\langle, Expression, \rangle\!\rangle \\ & | & Expression \\ & ; & \end{array}$$

Multiple unboxed paragraphs may be merged into a single unboxed paragraph.

1 Contents of this page

2 Introduction

3 ISO Standard . . .

3.1 Formal . . .

3.2 User-defined . . .

3.3 Additional . . .

4 CADiZ-specific . . .

$$
\begin{array}{lcl}
\mathit{Paragraph} & = & \mathit{ZED}, \mathit{UnboxedDef}, \{\mathit{UnboxedDef}\}, \mathit{END} \\
& | & \mathit{ZED}, \mathit{OperatorTemplate}, \{\mathit{OperatorTemplate}\}, \mathit{END} \\
& | & \text{axiomatic description} \\
& | & \text{schema definition} \\
& | & \text{generic axiomatic description} \\
& | & \text{generic schema definition} \\
& | & \text{conjecture} \\
& | & \text{generic conjecture} \\
& ; & \\
\mathit{UnboxedDef} & = & \text{given set} \\
& | & \text{free types} \\
& | & \text{horizontal definition} \\
& | & \text{generic horizontal definition} \\
& | & \text{generic operator definition} \\
& ; &
\end{array}
$$

IT 22-Jan-2002