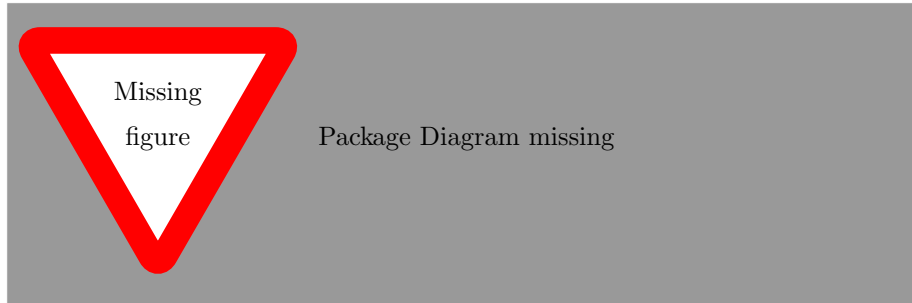


1 Package uppaal

Overview Contains Uppaal-specific sub-packages.



1.1 Class NTA

Overview A 'Network of Timed Automata' as basic input to Uppaal.

Super Types of NTA

`NamedElement` see Section 2.2 on Page 3 ,

`CommentableElement` see Section 2.1 on Page 3

References of NTA

`bool` : `PredefinedType` [1..1] see Section 9.4 on Page 38

The predefined type 'bool'.

`chan` : `PredefinedType` [1..1] see Section 9.4 on Page 38

The predefined type 'chan'.

`clock` : `PredefinedType` [1..1] see Section 9.4 on Page 38

The predefined type 'clock'.

`globalDeclarations` : `GlobalDeclarations` see Section 3.12 on Page 8

The global declarations for the NTA.

`int` : `PredefinedType` [1..1] see Section 9.4 on Page 38

The predefined type 'int'.

`systemDeclarations` : `SystemDeclarations` [1..1] see Section 3.17 on Page 10

The declarations of process instantiations.

`template` : `Template` [1..*] see Section 8.9 on Page 36

The Timed Automata templates of the NTA.

`void` : PredefinedType [1..1] see Section 9.4 on Page 38
The predefined dummy type 'void'.

OCL Constraints of NTA

MatchingIntDetails

```
(not self.int.ocIsUndefined())  
implies  
((self.int.type = types::BuiltInType::INT) and (  
  self.int.name.equalsIgnoreCase('int')))
```

MatchingBoolDetails

```
(not self.bool.ocIsUndefined())  
implies  
((self.bool.type = types::BuiltInType::BOOL) and  
  (self.bool.name.equalsIgnoreCase('bool')))
```

MatchingClockDetails

```
(not self.clock.ocIsUndefined())  
implies  
((self.clock.type = types::BuiltInType::CLOCK)  
  and (self.clock.name.equalsIgnoreCase('clock'))  
))
```

MatchingChanDetails

```
(not self.chan.ocIsUndefined())  
implies  
((self.chan.type = types::BuiltInType::CHAN) and  
  (self.chan.name.equalsIgnoreCase('chan')))
```

MatchingVoidDetails

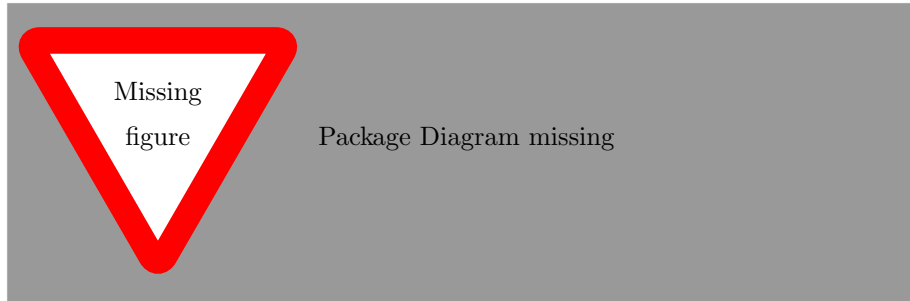
```
(not self.void.ocIsUndefined())  
implies  
((self.void.type = types::BuiltInType::VOID) and  
  (self.void.name.equalsIgnoreCase('void')))
```

UniqueTemplateName

```
self.template->isUnique(name)
```

2 Package uppaal::core

Overview Contains abstract general purpose classes.



2.1 Abstract Class CommentableElement

Overview Abstract base class for commentable model elements.

Attributes of CommentableElement

`comment : EString`
The comment for the model element.

CG says: Change cardinality to 1..1?

2.2 Abstract Class NamedElement

Overview Abstract base class for named model elements.

Attributes of NamedElement

`name : EString [1..1]`
The name of the model element..

OCL Constraints of NamedElement

NoWhitespace

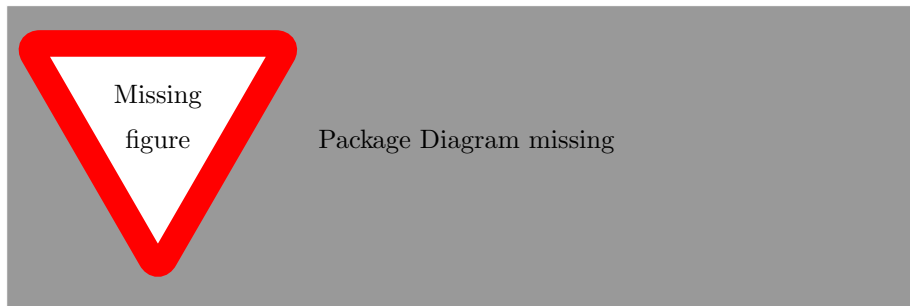
`self.name.characters()->excludes(' ')`

NoDigitStart

`Set{0..9}->excludes(self.name.characters()->first())`

3 Package uppaal::declarations

Overview Support for all kinds of declarations, e.g. types, functions, or variables.



3.1 Class ArrayInitializer

Overview An initializer for array variables, referring to multiple sub-initializers.

Super Types of ArrayInitializer

`Initializer` see Section 3.14 on Page 9

References of ArrayInitializer

`initializer : Initializer [1..*]` see Section 3.14 on Page 9

A number of sub-initializers, each one representing the initial value for one array index.

3.2 Enumeration CallType

Overview Represents call-by-value or call-by-reference parameters.

Literals of CallType

`CALL_BY_VALUE = 0`

`CALL_BY_REFERENCE = 1`

3.3 Class ChannelVariableDeclaration

Overview A declaration of synchronization channel variables.

Super Types of ChannelVariableDeclaration

`VariableDeclaration` see Section 3.23 on Page 13

Attributes of ChannelVariableDeclaration

broadcast : EBoolean [1..1]
 Specifies whether the declared synchronization channels use broadcast.

urgent : EBoolean [1..1]
 Specifies the urgency of the declared synchronization channels.

OCIL Constraints of ChannelVariableDeclaration

MatchingType

```
(not self.typeDefinition.ocIsUndefined())
implies
self.typeDefinition.baseType = types::BuiltInType
::CHAN
```

3.4 Class ClockVariableDeclaration

Overview A declaration of clock variables.

Super Types of ClockVariableDeclaration

VariableDeclaration see Section 3.23 on Page 13

OCIL Constraints of ClockVariableDeclaration

MatchingType

```
(not self.typeDefinition.ocIsUndefined())
implies
self.typeDefinition.baseType = types::BuiltInType
::CLOCK
```

3.5 Class DataVariableDeclaration

Overview A declaration of data variables.

Super Types of DataVariableDeclaration

VariableDeclaration see Section 3.23 on Page 13

Attributes of DataVariableDeclaration

prefix : DataVariablePrefix [1..1] see Section 3.6 on Page 6
 The prefix of the data variable declaration.

OCIL Constraints of DataVariableDeclaration

MatchingType

```

(not self.typeDefinition.ocIsUndefined())
implies
(self.typeDefinition.baseType <> types::
  BuiltInType::CHAN
and
self.typeDefinition.baseType <> types::
  BuiltInType::CLOCK)

```

3.6 Enumeration DataVariablePrefix

Overview Prefixes for data variables with base type 'int' or 'bool'.

Literals of DataVariablePrefix

```

NONE = 0
CONST = 1
META = 2

```

3.7 Abstract Class Declaration

Overview Abstract base class representing a variable, function, or type declaration.

3.8 Abstract Class Declarations

Overview Represents a set of variable, type, function, or template declarations, that are either global, local to a template, local to a block, or system declarations.

References of Declarations

```

declaration : Declaration [0..*]    see Section 3.7 on Page 6
    The single declarations.

```

OCL Constraints of Declarations

UniqueFunctionNames

```

self.declaration->select(oclIsKindOf(
  FunctionDeclaration)).oclAsType(
  FunctionDeclaration)->collect(function)->
  isUnique(name)

```

UniqueVariableNames

```

self.declaration->select(oclIsKindOf(
  VariableDeclaration)).oclAsType(
  VariableDeclaration)->collect(variable)->
  isUnique(name)

```

UniqueTypeNames

```
self.declaration->select(oclIsKindOf(
    TypeDeclaration)).oclAsType(TypeDeclaration)->
    collect(type)->isUnique(name)
```

3.9 Class ExpressionInitializer

Overview An initializer that represents a single initial value by means of an expression.

Super Types of ExpressionInitializer

Initializer see Section 3.14 on Page 9

References of ExpressionInitializer

expression : Expression [1..1] see Section 6.13 on Page 22
The expression representing the initial value.

3.10 Class Function

Overview A function with a return type and optional parameters.

Super Types of Function

NamedElement see Section 2.2 on Page 3

References of Function

block : Block [1..1] see Section 7.1 on Page 28
The block of statements representing the function body.

parameter : Parameter [0..*] see Section 3.16 on Page 9
The function's parameters.

returnType : TypeDefinition [1..1] see Section 9.9 on Page 40
The return type of this function.

OCL Constraints of Function

ReturnStatementExistsIfRequired

```
((not self.returnType.oclIsUndefined()) and
self.returnType.baseType <> types::BuiltInType::
    VOID)
implies
((not self.block.oclIsUndefined()) and
self.block.statement->exists(oclIsKindOf(
    statements::ReturnStatement)))
```

ValidReturnType

```
(not returnType.ocIsUndefined())  
implies  
(returnType.baseType = types::BuiltInType::VOID  
  or  
  returnType.baseType = types::BuiltInType::INT or  
  returnType.baseType = types::BuiltInType::BOOL)
```

UniqueParameterNames

```
self.parameter->collect(variableDeclaration)->  
  collect(variable)->isUnique(name)
```

3.11 Class FunctionDeclaration

Overview Declaration of a single function.

Super Types of FunctionDeclaration

Declaration see Section 3.7 on Page 6

References of FunctionDeclaration

function : **Function** [1..1] see Section 3.10 on Page 7
The return type of this function.

3.12 Class GlobalDeclarations

Overview Global declarations of an NTA.

Super Types of GlobalDeclarations

Declarations see Section 3.8 on Page 6

References of GlobalDeclarations

channelPriority : **ChannelPriority** see Section 4.2 on Page 14
The declaration of the synchronization channel priorities.

OCL Constraints of GlobalDeclarations

NoTemplateDeclarations

```
not self.declaration->exists(oclIsKindOf(system::  
  TemplateDeclaration))
```

3.13 Abstract Class Index

Overview Abstract base-class for indexing variables or types.

3.14 Abstract Class Initializer

Overview An initializer specifies a variable's initial value.

3.15 Class LocalDeclarations

Overview Local declarations inside a template or block of statements.

Super Types of LocalDeclarations

Declarations see Section 3.8 on Page 6

OCL Constraints of LocalDeclarations

NoTemplateDeclarations

```
not self.declaration->exists(oclIsKindOf(system::
    TemplateDeclaration))
```

NoChannelDeclarations

```
not self.declaration->exists(oclIsKindOf(
    ChannelVariableDeclaration))
```

3.16 Class Parameter

Overview A parameter of a function or template.

Attributes of Parameter

callType : **CallType** see Section 3.2 on Page 4
Specifies whether call-by-value or call-by-reference semantics should be applied.

References of Parameter

variableDeclaration : **VariableDeclaration** [1..1] see Section 3.23 on Page 13
A variable declaration containing the variable that represents the parameter.

OCL Constraints of Parameter

SingleVariable

```
(not self.variableDeclaration.oclIsUndefined())
implies
self.variableDeclaration.variable->size() <= 1
```

3.17 Class SystemDeclarations

Overview System declarations consisting of process instantiations.

Super Types of SystemDeclarations

`Declarations` see Section 3.8 on Page 6

References of SystemDeclarations

`progressMeasure : ProgressMeasure` see Section 5.2 on Page 17
The optional progress measure section.

`system : System [1..1]` see Section 5.3 on Page 17
The system section describing the process instantiations.

OCL Constraints of SystemDeclarations

UniqueTemplateName

```
self.declaration->select(oclIsKindOf(system::
    TemplateDeclaration)).oclAsType(system::
    TemplateDeclaration)->collect(declaredTemplate
)->isUnique(name)
```

NoChannelDeclarations

```
not self.declaration->exists(oclIsKindOf(
    ChannelVariableDeclaration))
```

3.18 Class TypeDeclaration

Overview A declaration of one or more types.

Super Types of TypeDeclaration

`Declaration` see Section 3.7 on Page 6

References of TypeDeclaration

`type : DeclaredType [1..*]` see Section 9.2 on Page 37
The types declared by this type declaration.

`typeDefinition : TypeDefinition [1..1]` see Section 9.9 on Page 40
The type definition for declared types.

OCL Constraints of TypeDeclaration

UniqueTypeNames

```
self.type->isUnique(name)
```

3.19 Class TypeIndex

Overview An index specified by a bounded integer-based type.

Super Types of TypeIndex

Index see Section 3.13 on Page 8

References of TypeIndex

typeDefinition : TypeDefinition [1..1] see Section 9.9 on Page 40
An integer-based type representing size and range of the indexed type or variable.

OCIL Constraints of TypeIndex

IntegerBasedIndex

```
(not self.typeDefinition.ocIsUndefined())  
implies  
self.typeDefinition.baseType = types::BuiltInType  
::INT
```

3.20 Class ValueIndex

Overview An index specified by an expression value.

Super Types of ValueIndex

Index see Section 3.13 on Page 8

References of ValueIndex

sizeExpression : Expression [1..1] see Section 6.13 on Page 22
An integer-based expression representing size and range of the indexed type or variable.

3.21 Class Variable

Overview A typed variable.

Super Types of Variable

NamedElement see Section 2.2 on Page 3

References of Variable

container : VariableContainer [1..1] see Section 3.22 on Page 12
The container of this variable.
index : Index [0..*] see Section 3.13 on Page 8
A set of array indexes for the variable.

initializer : **Initializer** see Section 3.14 on Page 9
 Represents the variable's initial value.

/typeDefinition : **TypeDefinition** [1..1] see Section 9.9 on Page 40
 The type definition of this variable.

derivation

```

if self.container.oclIsUndefined()
then null
else
  self.container.typeDefinition
endif

```

OCIL Constraints of Variable

NoInitializerForClockAndChannelVariables

```

((not self.typeDefinition.oclIsUndefined()) and
 (self.typeDefinition.baseType = types::
   BuiltInType::CHAN or
   self.typeDefinition.baseType = types::
     BuiltInType::CLOCK))
implies self.initializer.oclIsUndefined()

```

3.22 Abstract Class VariableContainer

Overview Abstract base class for objects containing variables like variable declarations, iterations, quantifications or selections.

References of VariableContainer

typeDefinition : **TypeDefinition** [1..1] see Section 9.9 on Page 40
 The type definition for the contained variables.

variable : **Variable** [1..*] see Section 3.21 on Page 11
 The contained variables.

OCIL Constraints of VariableContainer

NoVoidVariables

```

(not self.typeDefinition.oclIsUndefined())
implies
  self.typeDefinition.baseType <> types::
    BuiltInType::VOID

```

UniqueVariableNames

```

self.variable->isUnique(name)

```

3.23 Abstract Class VariableDeclaration

Overview A declaration of one or more variables.

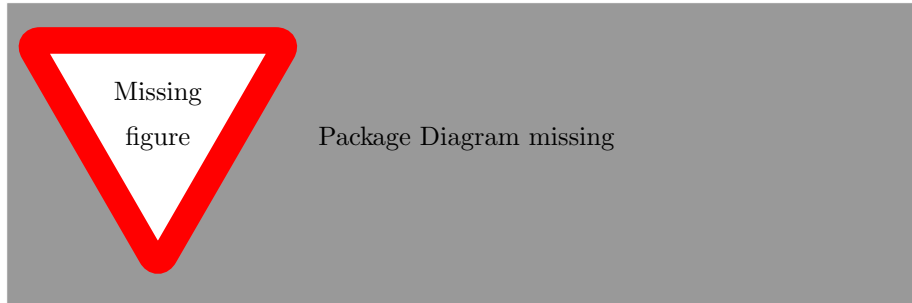
Super Types of VariableDeclaration

Declaration see Section 3.7 on Page 6 ,

VariableContainer see Section 3.22 on Page 12

4 Package uppaal::declarations::global

Overview Contains special classes that are relevant for the global declarations.



4.1 Class ChannelList

Overview A list of synchronization channel variables, used to assign these channels a common priority.

Super Types of ChannelList

ChannelPriorityItem see Section 4.3 on Page 15

References of ChannelList

channelExpression : IdentifierExpression [1..*] see Section 6.15 on Page 23

The variable expressions representing the synchronization channels inside the channel list.

OCL Constraints of ChannelList

ChannelVariablesOnly

```
self.channelExpression->forAll(  
    (not identifier.typeDefinition.  
        oclIsUndefined()) implies identifier.  
        typeDefinition.baseType = types::  
        BuiltInType::CHAN  
)
```

4.2 Class ChannelPriority

Overview A priority ordering for synchronization channels.

References of ChannelPriority

`item : ChannelPriorityItem [1..*]` see Section 4.3 on Page 15
The items of the channel priority ordering.

OCL Constraints of ChannelPriority

AtMostOneDefaultItem

```
self.item->select(oclIsKindOf(
    DefaultChannelPriority))>size() <= 1
```

EachChannelContainedAtMostOnce

```
self.item->select(oclIsKindOf(ChannelList)).
    oclAsType(ChannelList)->collect(
    channelExpression)->isUnique(variable)
```

4.3 Abstract Class ChannelPriorityItem

Overview Abstract base class for items inside a channel priority.

4.4 Class DefaultChannelPriority

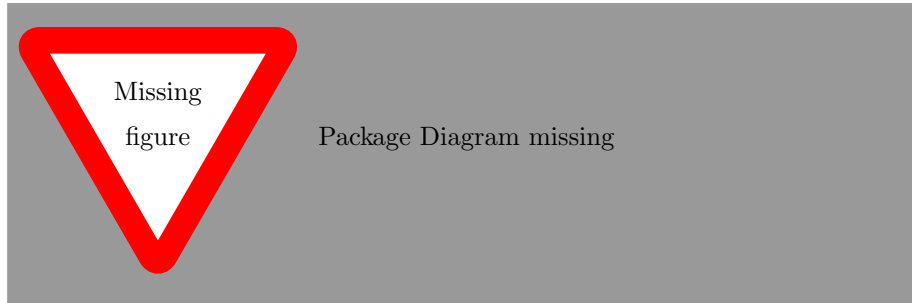
Overview A 'default' item inside a channel priority, representing all channels not listed explicitly.

Super Types of DefaultChannelPriority

`ChannelPriorityItem` see Section 4.3 on Page 15

5 Package uppaal::declarations::system

Overview Contains special classes that are relevant for the system declarations.



5.1 Class InstantiationList

Overview Represents a list of templates to be instantiated using a common priority.

References of InstantiationList

template : AbstractTemplate [1..*] see Section 8.1 on Page 32
The list of instantiations.

OCL Constraints of InstantiationList

OnlyLegalParamsForPartialInstantiation

```
self.template->forAll(  
    parameter->forAll(  
        callType = declarations::CallType  
            ::CALL\BY\VALUE  
        and  
        ((not variableDeclaration.  
            oclIsUndefined()))  
            implies  
        (variableDeclaration.  
            typeDefinition.ocIsKindOf(  
                types::RangeTypeSpecification  
            ) or  
            variableDeclaration.  
                typeDefinition.ocIsKindOf(  
                    types::  
                        ScalarTypeSpecification)))
```


)
)

5.2 Class ProgressMeasure

Overview A progress measure consisting of monotonically increasing expressions.

References of ProgressMeasure

`expression` : `Expression` [1..*] see Section 6.13 on Page 22
The progress measure expressions.

5.3 Class System

Overview A system contains declarations of template instantiations.

References of System

`instantiationList` : `InstantiationList` [1..*] see Section 5.1 on Page 16
A list of process instantiation sublists, ordered by decreasing priority.
The templates referenced inside the sublists are instantiated to be part of the system at runtime.

OCL Constraints of System

EachTemplateReferencedAtMostOnce

```
self.instantiationList->collect(template)->
  isUnique(t : templates::AbstractTemplate | t)
```

5.4 Class TemplateDeclaration

Overview A declaration of a template redefinition.

Super Types of TemplateDeclaration

`Declaration` see Section 3.7 on Page 6

References of TemplateDeclaration

`argument` : `Expression` [0..*] see Section 6.13 on Page 22
A number of arguments that describe how the referred template's parameters should be mapped towards the declared template's parameters.

`declaredTemplate : RedefinedTemplate [1..1]` see Section 8.5
on Page 34

The template being declared.

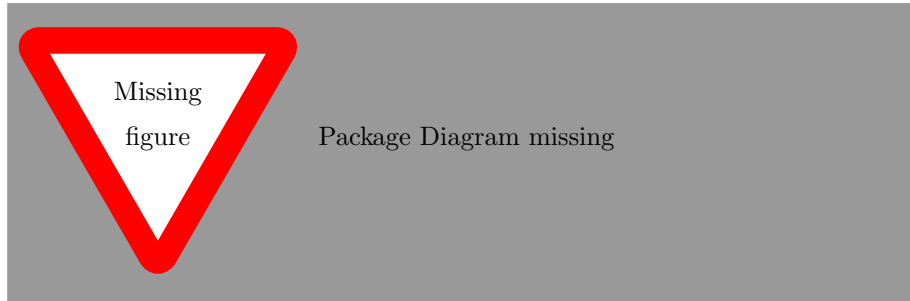
OCL Constraints of TemplateDeclaration

NumberOfArgumentsMatchesDeclaration

`(not self.declaredTemplate.ocIsUndefined() and
not self.declaredTemplate.referredTemplate.
ocIsUndefined())`
implies
`self.argument->size() = self.declaredTemplate.
referredTemplate.parameter->size()`

6 Package uppaal::expressions

Overview Introduces all kinds of expressions.



6.1 Class ArithmeticExpression

Overview A binary expression representing an arithmetic operation.

Super Types of ArithmeticExpression

BinaryExpression see Section 6.5 on Page 20

Attributes of ArithmeticExpression

operator : ArithmeticOperator [1..1] see Section 6.2 on Page 19
The arithmetic operator to be applied.

6.2 Enumeration ArithmeticOperator

Overview Representing all arithmetic operators.

Literals of ArithmeticOperator

ADD = 0
SUBTRACT = 1
MULTIPLICATE = 2
DIVIDE = 3
MODULO = 4

6.3 Class AssignmentExpression

Overview A binary assignment expression using a specific assignment operator.

Super Types of AssignmentExpression

BinaryExpression see Section 6.5 on Page 20

Attributes of AssignmentExpression

operator : AssignmentOperator [1..1] see Section 6.4 on Page 20
The operator for the assignment.

6.4 Enumeration AssignmentOperator

Overview Representing all assignment operators.

Literals of AssignmentOperator

EQUAL = 0
PLUS_EQUAL = 1
MINUS_EQUAL = 2
TIMES_EQUAL = 3
DIVIDE_EQUAL = 4
MODULO_EQUAL = 5
BIT_AND_EQUAL = 6
BIT_OR_EQUAL = 7
BIT_LEFT_EQUAL = 8
BIT_RIGHT_EQUAL = 9
BIT_XOR_EQUAL = 10

6.5 Abstract Class BinaryExpression

Overview Abstract base class for all binary expressions connecting two sub-expressions.

Super Types of BinaryExpression

Expression see Section 6.13 on Page 22

References of BinaryExpression

firstExpr : Expression [1..1] see Section 6.13 on Page 22
The first sub-expression.
secondExpr : Expression [1..1] see Section 6.13 on Page 22
The second sub-expression.

6.6 Class BitShiftExpression

Overview A binary expression representing an arithmetic operation.

Super Types of BitShiftExpression

BinaryExpression see Section 6.5 on Page 20

Attributes of BitShiftExpression

operator : BitShiftOperator [1..1] see Section 6.7 on Page 21
The arithmetic operator to be applied.

6.7 Enumeration BitShiftOperator

Overview Representing all arithmetic operators.

Literals of BitShiftOperator

LEFT = 0
RIGHT = 1

6.8 Class BitwiseExpression

Overview A binary expression representing an arithmetic operation.

Super Types of BitwiseExpression

BinaryExpression see Section 6.5 on Page 20

Attributes of BitwiseExpression

operator : BitwiseOperator [1..1] see Section 6.9 on Page 21
The arithmetic operator to be applied.

6.9 Enumeration BitwiseOperator

Overview Representing all arithmetic operators.

Literals of BitwiseOperator

AND = 0
XOR = 1
OR = 2

6.10 Class CompareExpression

Overview A comparison between two expression values using a specific comparison operator.

Super Types of CompareExpression

BinaryExpression see Section 6.5 on Page 20

Attributes of CompareExpression

operator : CompareOperator [1..1] see Section 6.11 on Page 22
The comparison operator to be applied.

6.11 Enumeration CompareOperator

Overview Representing all comparison operators.

Literals of CompareOperator

EQUAL = 0
GREATER = 1
GREATER_OR_EQUAL = 2
LESS = 3
LESS_OR_EQUAL = 4
UNEQUAL = 5

6.12 Class ConditionExpression

Overview An expression representing a conditional redirection to one of the sub-expressions. Uses tokens '?' and ':' for delimitation.

Super Types of ConditionExpression

Expression see Section 6.13 on Page 22

References of ConditionExpression

elseExpression : Expression [1..1] see Section 6.13 on Page 22
The else-expression.
ifExpression : Expression [1..1] see Section 6.13 on Page 22
The boolean if-expression.
thenExpression : Expression [1..1] see Section 6.13 on Page 22
The then-expression.

6.13 Abstract Class Expression

Overview Abstract base class for all kinds of expressions.

6.14 Class FunctionCallExpression

Overview An expression representing a call to a function.

Super Types of FunctionCallExpression

Expression see Section 6.13 on Page 22

References of FunctionCallExpression

argument : Expression [0..*] see Section 6.13 on Page 22
A set of expressions representing the argument values for the function call. Must conform to the parameters of the function declaration.

function : Function [1..1] see Section 3.10 on Page 7
The function to be called.

OCL Constraints of FunctionCallExpression

NumberOfArgumentsMatchesDeclaration

```
(not self.function.ocIsUndefined())  
implies  
self.argument->size() = self.function.parameter->  
size()
```

6.15 Class IdentifierExpression

Overview An expression referring to a variable.

Super Types of IdentifierExpression

Expression see Section 6.13 on Page 22

References of IdentifierExpression

identifier : NamedElement [1..1] see Section 2.2 on Page 3
The referred variable.

index : Expression [0..*] see Section 6.13 on Page 22
A set of expressions that refer to the array indexes of the variable.

6.16 Class IncrementDecrementExpression

Overview An expression describing increment (++) or decrement (--) of an integer-based expression.

Super Types of IncrementDecrementExpression

Expression see Section 6.13 on Page 22

Attributes of IncrementDecrementExpression

operator : IncrementDecrementOperator [1..1] see Section 6.17
 on Page 24
 Specifies increment or decrement.

position : IncrementDecrementPosition [1..1] see Section 6.18
 on Page 24
 Specifies pre- or post-evaluation.

References of IncrementDecrementExpression

expression : Expression [1..1] see Section 6.13 on Page 22
 The expression to be incremented or decremented.

6.17 Enumeration IncrementDecrementOperator

Overview Representing increment and decrement operators.

Literals of IncrementDecrementOperator

INCREMENT = 0
 DECREMENT = 1

6.18 Enumeration IncrementDecrementPosition

Overview Representing pre- or post-processing inside increment/decrement expressions.

Literals of IncrementDecrementPosition

PRE = 0
 POST = 1

6.19 Class LiteralExpression

Overview An expression referring to a literal of any type.

Super Types of LiteralExpression

Expression see Section 6.13 on Page 22

Attributes of LiteralExpression

text : EString [1..1]
 The textual description of the literal.

6.20 Class LogicalExpression

Overview A logical expression.

Super Types of LogicalExpression

BinaryExpression see Section 6.5 on Page 20

Attributes of LogicalExpression

operator : LogicalOperator [1..1] see Section 6.21 on Page 25

ecore2latex: Documentation missing (GenModel is not defined)

6.21 Enumeration LogicalOperator

Overview

ecore2latex: Documentation missing (GenModel is not defined)

Literals of LogicalOperator

AND = 0

OR = 1

IMPLY = 2

6.22 Class MinMaxExpression

Overview A binary expression representing an arithmetic operation.

Super Types of MinMaxExpression

BinaryExpression see Section 6.5 on Page 20

Attributes of MinMaxExpression

operator : MinMaxOperator [1..1] see Section 6.23 on Page 25
The arithmetic operator to be applied.

6.23 Enumeration MinMaxOperator

Overview Representing all arithmetic operators.

Literals of MinMaxOperator

MIN = 0

MAX = 1

6.24 Class MinusExpression

Overview An inversion of an integer-based expression using the '-' token.

Super Types of MinusExpression

Expression see Section 6.13 on Page 22

References of MinusExpression

invertedExpression : Expression [1..1] see Section 6.13 on Page 22
The expression negated by this negation.

6.25 Class NegationExpression

Overview A negation of an expression.

Super Types of NegationExpression

Expression see Section 6.13 on Page 22

References of NegationExpression

negatedExpression : Expression [1..1] see Section 6.13 on Page 22
The expression negated by this negation.

6.26 Class PlusExpression

Overview A confirmation of an integer-based expression using the '+' token.

Super Types of PlusExpression

Expression see Section 6.13 on Page 22

References of PlusExpression

confirmedExpression : Expression [1..1] see Section 6.13 on
Page 22
The expression negated by this negation.

6.27 Class QuantificationExpression

Overview A quantification expression introducing a quantified variable.

Super Types of QuantificationExpression

Expression see Section 6.13 on Page 22 ,
VariableContainer see Section 3.22 on Page 12

Attributes of QuantificationExpression

quantifier : **Quantifier** [1..1] see Section 6.28 on Page 27

The quantifier to be applied.

References of QuantificationExpression

expression : **Expression** [1..1] see Section 6.13 on Page 22

The quantified expression.

OCL Constraints of QuantificationExpression

Single Variable

`self.variable->size() <= 1`

6.28 Enumeration Quantifier

Overview Representing existential and universal quantification.

Literals of Quantifier

`EXISTENTIAL = 0`

`UNIVERSAL = 1`

6.29 Class ScopedIdentifierExpression

Overview An expression used to access a scoped identifier. Uses a dot for delimitation between scope and identifier.

Super Types of ScopedIdentifierExpression

Expression see Section 6.13 on Page 22

References of ScopedIdentifierExpression

identifier : **IdentifierExpression** [1..1] see Section 6.15 on Page 23

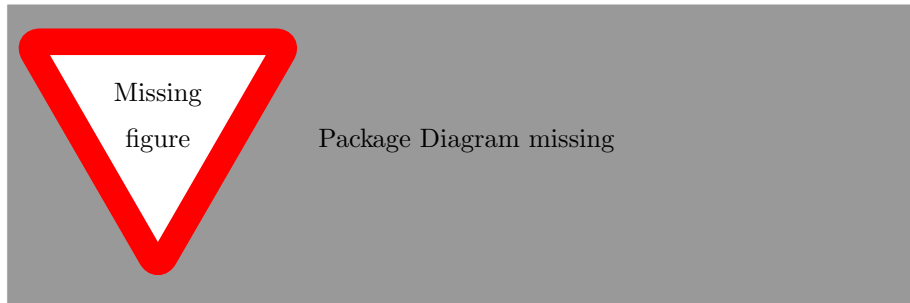
An expression that refers to a member of the scope.

scope : **Expression** [1..1] see Section 6.13 on Page 22

An expression that refers to a certain identifier scope.

7 Package uppaal::statements

Overview Support for statements inside functions.



7.1 Class Block

Overview A block of one or more statements.

Super Types of Block

Statement see Section 7.9 on Page 31

References of Block

declarations : `LocalDeclarations` see Section 3.15 on Page 9

The local declarations for the function's body.

statement : `Statement` [1..*] see Section 7.9 on Page 31

The statements inside the function's body.

OCL Constraints of Block

DataVariableDeclarationsOnly

```
(not self.declarations.ocIsUndefined())  
implies  
(self.declarations.declaration->forAll(  
  ocIsKindOf(declarations::  
    DataVariableDeclaration)))
```

7.2 Class DoWhileLoop

Overview A do-while-loop statement.

Super Types of DoWhileLoop

Statement see Section 7.9 on Page 31

References of DoWhileLoop

expression : Expression [1..1] see Section 6.13 on Page 22
A boolean expression for the while loop.

statement : Statement [1..1] see Section 7.9 on Page 31
The statement to be evaluated for every value.

7.3 Class EmptyStatement

Overview An empty statement represented by a semicolon only.

Super Types of EmptyStatement

Statement see Section 7.9 on Page 31

7.4 Class ExpressionStatement

Overview A statement that refers to an arbitrary expression.

Super Types of ExpressionStatement

Statement see Section 7.9 on Page 31

References of ExpressionStatement

expression : Expression [1..1] see Section 6.13 on Page 22
The expression this statement refers to.

7.5 Class ForLoop

Overview A for-loop statement.

Super Types of ForLoop

Statement see Section 7.9 on Page 31

References of ForLoop

condition : Expression [1..1] see Section 6.13 on Page 22
The condition of the for loop, represented by a boolean expression.

initialization : Expression [1..1] see Section 6.13 on Page 22
The initialization expression of the for loop.

iteration : Expression [1..1] see Section 6.13 on Page 22
The iteration statements of the for loop.

statement : Statement [1..1] see Section 7.9 on Page 31
The statement to be evaluated for every value.

7.6 Class IfStatement

Overview An if-then-else statement.

Super Types of IfStatement

Statement see Section 7.9 on Page 31

References of IfStatement

elseStatement : **Statement** see Section 7.9 on Page 31

The else-statement.

ifExpression : **Expression** [1..1] see Section 6.13 on Page 22

The boolean if-expression.

thenStatement : **Statement** [1..1] see Section 7.9 on Page 31

The then-statement.

7.7 Class Iteration

Overview An iteration over all possible values of a bounded type using the 'for' keyword.

Super Types of Iteration

Statement see Section 7.9 on Page 31 ,

VariableContainer see Section 3.22 on Page 12

References of Iteration

statement : **Statement** [1..1] see Section 7.9 on Page 31

The statement to be evaluated for every value.

OCL Constraints of Iteration

SingleVariable

`self.variable->size() <= 1`

7.8 Class ReturnStatement

Overview A statement used to return from a function's body, optionally carrying a return value.

Super Types of ReturnStatement

Statement see Section 7.9 on Page 31

References of ReturnStatement

returnExpression : **Expression** see Section 6.13 on Page 22

The expression representing the return value.

7.9 Abstract Class Statement

Overview Abstract base-class for all statements inside a function's body.

7.10 Class WhileLoop

Overview A while-loop statement.

Super Types of WhileLoop

Statement see Section 7.9 on Page 31

References of WhileLoop

expression : **Expression** [1..1] see Section 6.13 on Page 22

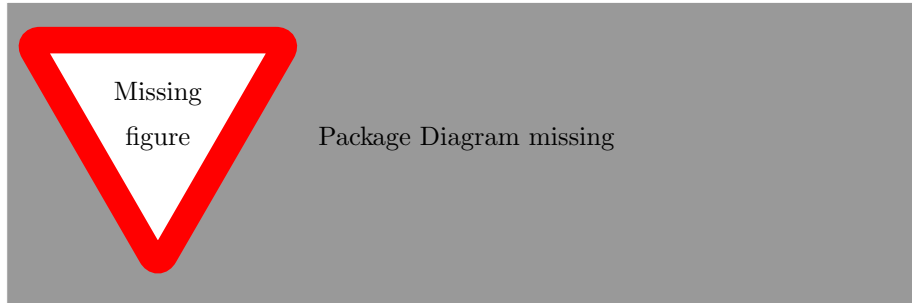
A boolean expression for the while loop.

statement : **Statement** [1..1] see Section 7.9 on Page 31

The statement to be evaluated for every value.

8 Package uppaal::templates

Overview Support for Timed Automata templates consisting of locations and edges.



8.1 Abstract Class AbstractTemplate

Overview Abstract base class for ordinary timed automata templates as well as redefined templates.

Super Types of AbstractTemplate

`NamedElement` see Section 2.2 on Page 3 ,

`CommentableElement` see Section 2.1 on Page 3

References of AbstractTemplate

`parameter` : `Parameter` [0..*] see Section 3.16 on Page 9
The parameter declarations of the template.

OCL Constraints of AbstractTemplate

UniqueParameterNames

```
self.parameter->collect(variableDeclaration)->  
collect(variable)->isUnique(name)
```

8.2 Class Edge

Overview An edge connecting two locations inside a template.

Super Types of Edge

`LinearElement` see Section 10.3 on Page 43 ,

`CommentableElement` see Section 2.1 on Page 3 ,

`ColoredElement` see Section 10.2 on Page 43

References of Edge

- guard** : **Expression** see Section 6.13 on Page 22
The guard expression of the edge.
- parentTemplate** : **Template** [1..1] see Section 8.9 on Page 36
The parent template containing the edge.
- selection** : **Selection** [0..*] see Section 8.6 on Page 34
A set of non-deterministic value selections.
- source** : **Location** [1..1] see Section 8.3 on Page 33
The source location of the edge.
- synchronization** : **Synchronization** see Section 8.7 on Page 35
A synchronization performed when the edge fires.
- target** : **Location** [1..1] see Section 8.3 on Page 33
The target location of the edge.
- update** : **Expression** [0..*] see Section 6.13 on Page 22
A set of update expressions for the edge, evaluated if the edge fires.

OCL Constraints of Edge

UniqueParentTemplate

```
(not (self.source.oclIsUndefined() or self.target
      .oclIsUndefined()))
implies
self.source.parentTemplate = self.target.
parentTemplate
```

8.3 Class Location

Overview A location inside a template.

Super Types of Location

- NamedElement** see Section 2.2 on Page 3 ,
- CommentableElement** see Section 2.1 on Page 3 ,
- PlanarElement** see Section 10.4 on Page 43 ,
- ColoredElement** see Section 10.2 on Page 43

Attributes of Location

- locationTimeKind** : **LocationKind** [1..1] see Section 8.4 on Page 34
Specifies the kind of location (default, urgent, or committed).

References of Location

invariant : **Expression** see Section 6.13 on Page 22
 A boolean expression representing the location's invariant.

parentTemplate : **Template** [1..1] see Section 8.9 on Page 36
 The parent template containing the location.

8.4 Enumeration LocationKind

Overview Location types.

Literals of LocationKind

NORMAL = 0
URGENT = 1
COMMITTED = 2

8.5 Class RedefinedTemplate

Overview A template resulting from redefinition of another referred template, altering its name and parametrization.

Super Types of RedefinedTemplate

AbstractTemplate see Section 8.1 on Page 32

References of RedefinedTemplate

declaration : **TemplateDeclaration** [1..1] see Section 5.4 on Page 17
 The declaration of this template.

referredTemplate : **AbstractTemplate** [1..1] see Section 8.1 on Page 32
 The template that serves as basis for redefinition.

8.6 Class Selection

Overview A non-deterministic selection of a value from a range. The range is specified by a bounded type.

Super Types of Selection

VariableContainer see Section 3.22 on Page 12

OCL Constraints of Selection

Single Variable

`self.variable->size() <= 1`

IntegerBasedType

```
(not self.typeDefinition.ocIsUndefined())  
implies  
self.typeDefinition.baseType = types::BuiltInType  
::INT
```

8.7 Class Synchronization

Overview A sent or received synchronization between two templates using a specific synchronization channel.

Attributes of Synchronization

kind : SynchronizationKind [1..1] see Section 8.8 on Page 35
The kind of synchronization (sent or received).

References of Synchronization

channelExpression : IdentifierExpression [1..1] see Section 6.15
on Page 23
An expression representing the channel variable used for synchronization.

OCL Constraints of Synchronization

ChannelVariablesOnly

```
(not self.channelExpression.ocIsUndefined())  
and  
(not self.channelExpression.identifier.  
    ocIsUndefined())  
and  
(self.channelExpression.identifier.ocIsKindOf(  
    declarations::Variable))  
and  
(not self.channelExpression.identifier.ocAsType(  
    declarations::Variable).typeDefinition.  
    ocIsUndefined())  
implies  
self.channelExpression.identifier.ocAsType(  
    declarations::Variable).typeDefinition.  
    baseType = types::BuiltInType::CHAN
```

8.8 Enumeration SynchronizationKind

Overview Representing the type of synchronization.

Literals of SynchronizationKind

```
RECEIVE = 0  
SEND = 1
```

8.9 Class Template

Overview An Uppaal template representing a single timed automaton.

Super Types of Template

`AbstractTemplate` see Section 8.1 on Page 32

References of Template

`declarations` : `LocalDeclarations` see Section 3.15 on Page 9
The local declarations of the template.

`edge` : `Edge` [0..*] see Section 8.2 on Page 32
The edges inside this template.

`init` : `Location` [1..1] see Section 8.3 on Page 33
The initial location of this template.

`location` : `Location` [1..*] see Section 8.3 on Page 33
The locations inside this template.

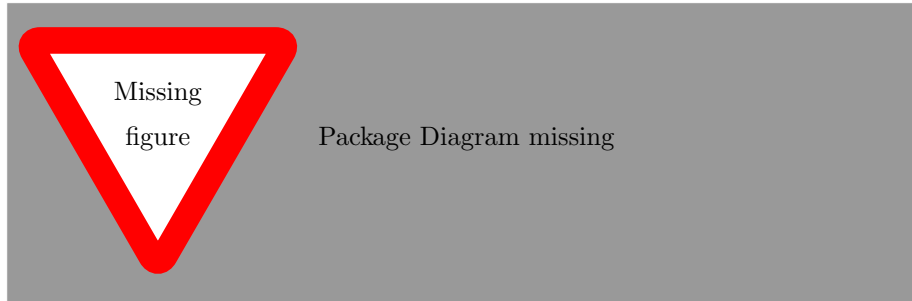
OCL Constraints of Template

UniqueLocationNames

```
self.location->isUnique(name)
```

9 Package uppaal::types

Overview Provides support for built-in and user-defined types.



9.1 Enumeration BuiltInType

Overview All built-in types.

Literals of BuiltInType

```
INT = 0
CLOCK = 1
CHAN = 2
BOOL = 3
VOID = 4
```

9.2 Class DeclaredType

Overview A user-declared type.

Super Types of DeclaredType

Type see Section 9.8 on Page 39

References of DeclaredType

`typeDeclaration` : `TypeDeclaration` [1..1] see Section 3.18 on Page 10

The declaration that declares this type.

`/typeDefinition` : `TypeDefinition` [1..1] see Section 9.9 on Page 40

The definition of the declared type. Usually a type specification, but can also be a type reference to a "renamed" type.

derivation

```

if self.typeDeclaration.oclIsUndefined()
then null
else self.typeDeclaration.typeDefinition
endif

```

9.3 Class IntegerBounds

Overview Used to restrict the 'int' type to a range of values.

References of IntegerBounds

```

lowerBound : Expression [1..1]    see Section 6.13 on Page 22
    An integer-based expression representing the lower bound.
upperBound : Expression [1..1]    see Section 6.13 on Page 22
    An integer-based expression representing the upper bound.

```

9.4 Class PredefinedType

Overview One of the predefined types 'int', 'bool', 'chan', 'clock' or 'void'.

Super Types of PredefinedType

Type see Section 9.8 on Page 39

Attributes of PredefinedType

```

type : BuiltInType [1..1]    see Section 9.1 on Page 37
    Stores the concrete literal that represents the predefined type.

```

9.5 Class RangeTypeSpecification

Overview A type specification restricting the 'int' type to a range of values.

Super Types of RangeTypeSpecification

TypeSpecification see Section 9.11 on Page 41

References of RangeTypeSpecification

```

bounds : IntegerBounds [1..1]    see Section 9.3 on Page 38
    The bounds that restrict the type specification.

```

9.6 Class ScalarTypeSpecification

Overview A specification of a 'scalar' type.

Super Types of ScalarTypeSpecification

TypeSpecification see Section 9.11 on Page 41

References of ScalarTypeSpecification

sizeExpression : Expression [1..1] see Section 6.13 on Page 22
An integer-based expression that represents the size of the scalar type.

9.7 Class StructTypeSpecification

Overview A specification of a 'struct' type.

Super Types of StructTypeSpecification

TypeSpecification see Section 9.11 on Page 41

References of StructTypeSpecification

declaration : DataVariableDeclaration [1..*] see Section 3.5
on Page 5
The variable declarations representing the fields of the 'struct' type.

OCL Constraints of StructTypeSpecification

UniqueFieldNames

```
self.declaration->collect(variable)->isUnique(  
    name)
```

9.8 Abstract Class Type

Overview Abstract base class for all types.

Super Types of Type

NamedElement see Section 2.2 on Page 3

Attributes of Type

/baseType : BuiltInType see Section 9.1 on Page 37

ecore2latex: Documentation missing (GenModel is not defined)

derivation

```

if self.ocIsKindOf(DeclaredType)
then
    if self.ocAsType(DeclaredType).
        typeDefinition.ocIsUndefined()
    then null
    else self.ocAsType(DeclaredType).
        typeDefinition.baseType
    endif
else
    if self.ocIsKindOf(PredefinedType)
    then self.ocAsType(PredefinedType).
        type
    else null
    endif
endif

```

References of Type

index : Index [0..*] see Section 3.13 on Page 8
 A set of array indexes for the type.

9.9 Abstract Class TypeDefinition

Overview Abstract base class for type definitions of all typed elements. Type definitions are either references to types defined elsewhere, or in place specifications of new types.

Attributes of TypeDefinition

/baseType : BuiltInType see Section 9.1 on Page 37
 The built-in base type this type definition relies on. Can be 'null' in case of a 'struct' type definition involved.

derivation

```

if self.ocIsKindOf(TypeReference)
then
    if self.ocAsType(TypeReference).
        referredType.ocIsUndefined()
    then null
    else self.ocAsType(TypeReference).
        referredType.baseType
    endif
else
    if self.ocIsKindOf(
        ScalarTypeSpecification) or self.
        ocIsKindOf(RangeTypeSpecification)
    then BuiltInType::INT

```



```
        else null
        endif
    endif
```

9.10 Class TypeReference

Overview A reference to a type defined elsewhere.

Super Types of TypeReference

TypeDefinition see Section 9.9 on Page 40

References of TypeReference

referredType : Type [1..1] see Section 9.8 on Page 39
The referred type.

9.11 Abstract Class TypeSpecification

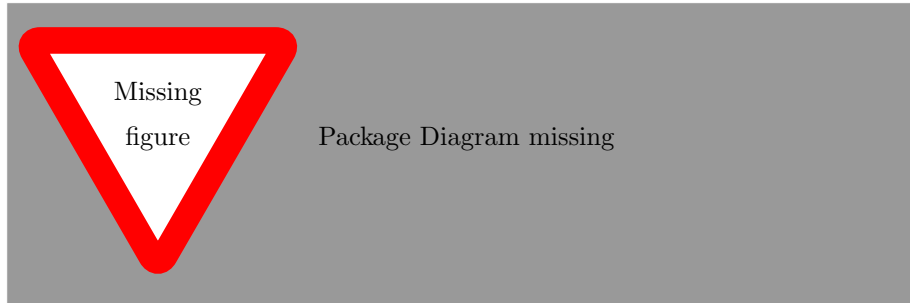
Overview Abstract base class for the specification of new types, using either the 'struct' or 'scalar' keywords, or restricting a type to a range of values.

Super Types of TypeSpecification

TypeDefinition see Section 9.9 on Page 40

10 Package uppaal::visuals

Overview Provides support for the visual representation of model elements.



10.1 Enumeration ColorKind

Overview The color kinds of an element. They are the standard colors of uppaal or a self-defined color.

Literals of ColorKind

```
DEFAULT = 0
WHITE = 1
LIGHTGREY = 2
DARKGREY = 3
BLACK = 4
BLUE = 5
CYAN = 6
GREEN = 7
MAGENTA = 8
ORANGE = 9
PINK = 10
RED = 11
YELLOW = 12
SELF_DEFINED = 13
```

10.2 Abstract Class ColoredElement

Overview A model element that has an optional color.

Attributes of ColoredElement

`color` : `ColorKind` see Section 10.1 on Page 42

The color of the model element. It is either a standard uppaal color (default, white, light grey, dark grey, black, blue, cyan, green, magenta, orange, pink, red, yellow) or a self-defined color. Edges should not be white.

SD says: We need an OCL-Constraint: Edges should not be white.

SD says: We need an OCL-Constraint: If self defined is choosen then a color code must be specified.

`colorCode` : `EString`

The hexadecimal color code of the model element that must be defined if a self-defined color should be used.

10.3 Abstract Class LinearElement

Overview A linear model element that has a set of bend points.

References of LinearElement

`bendPoint` : `Point` [0..*] see Section 10.5 on Page 43

The bend points of the linear model element.

10.4 Abstract Class PlanarElement

Overview A planar model element that has an optional position.

References of PlanarElement

`position` : `Point` see Section 10.5 on Page 43

The planar position of the model element.

10.5 Class Point

Overview Represents a point in the two-dimensional space.

Attributes of Point

`x` : `EInt` [1..1]

The horizontal component of the point.

`y` : `EInt` [1..1]

The vertical component of the point.