

# *Conceptual Modeling Language* Specification

Version 1.0

Quenio Cesar Machado dos Santos

Universidade Federal de Santa Catarina\*

July 2017

\* Initially developed as part of the author's Bachelor Technical Report in Computer Sciences

---

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Concepts</b>	<b>2</b>
<b>3</b>	<b>Associations</b>	<b>3</b>
<b>4</b>	<b>Values</b>	<b>4</b>
<b>5</b>	<b>Expressions</b>	<b>5</b>
<b>6</b>	<b>Targets</b>	<b>6</b>
<b>7</b>	<b>Modules and Libraries</b>	<b>7</b>
<b>A</b>	<b>Concrete Syntax (Grammar)</b>	<b>8</b>
A.1	ANTLR Grammar . . . . .	9
<b>B</b>	<b>Abstract Syntax (Metamodel)</b>	<b>11</b>
<b>C</b>	<b>Abstract Syntax Tree (Instantiation)</b>	<b>12</b>

---

# List of Figures

---

---

# List of Tables

---

# One

---

## Introduction

---

**Two**

---

Concepts

---

# Three

---

# Associations

---

**Four**

---

**Values**

---



# Five

---

# Expressions

---

**Six**

---

**Targets**

---

# Seven

---

## Modules and Libraries

---

# A

---

## Concrete Syntax (Grammar)

---

## A.1 ANTLR Grammar

```
grammar CML;

@header
{
import cml.language.foundation.*;
import cml.language.features.*;
}

modelName returns [Model model]:
    modelElementNode*;

modelElementNode:
    conceptNode | targetNode;

conceptNode returns [Concept concept]:
    ABSTRACT? 'concept' NAME
    (':' ancestorListNode)?
    ( ';' | propertyListNode);

targetNode returns [Target target]:
    'target' NAME propertyListNode;

propertyListNode:
    '{' (propertyNode ';'*) '}' ;

propertyNode returns [Property property]:
    NAME (':' typeNode)? ('=' STRING)?;

ancestorListNode:
    NAME (',' NAME)*;

typeNode returns [Type type]:
    NAME CARDINALITY?;

// Reserved words must precede names. Otherwise, they will be recognized as names.
ABSTRACT:
    'abstract';
```

NAME:

```
('A'..'Z' | 'a'..'z')  
( 'A'..'Z' | 'a'..'z' | '0'..'9' | '_' )*;
```

STRING:

```
'"' . * ? "' ;
```

CARDINALITY:

```
('?' | '*');
```

// Ignoring whitespace:

WS:

```
( ' ' | '\t' | '\f' | '\n' | '\r' )+ -> skip;
```

// Ignoring comments:

COMMENT:

```
('//' . * ? '\n' | '(*' . * ? '*)' ) -> skip;
```

# B

---

## Abstract Syntax (Metamodel)

---

**C**

---

# Abstract Syntax Tree (Instantiation)

---