AACHENER VERFAHRENSTECHNIK

Technical Report LPT-2008-26

Supporting Concepts

J. Morbach, A. Yang, A. Wiesner, W. Marquardt

July 2008

Enquiries should be addressed to:

RWTH Aachen University
Aachener Verfahrenstechnik
Process Systems Engineering
52056 Aachen

Tel.: +49 (0) 241 80 - 94668
Fax: +49 (0) 241 80 - 92326
E-Mail: secretary.pt@avt.rwth-aachen.de

RWTH

Contents

List of Figures

1 Introduction

The partial model **supporting_concepts** provides fundamental notions, such as spatial and temporal coordinate systems, geometrical concepts, mathematical relations, as well as commonly used physical dimensions and SI-units. The concepts defined in this partial model do not belong to the core of the CAPE domain, but are merely utilized by the other partial models of OntoCAPE for defining and completing domain concepts. For that reason, **supporting_concepts** is only rudimentarily developed, as it is not the objective of OntoCAPE to conceptualize areas that are beyond the scope of the CAPE domain. For example, the partial model **mathematical_relation** does not attempt to establish a full-fledged algebraic theory, as does the EngMath ontology (Gruber and Olsen, 1994); rather, it provides a simple but pragmatic mechanism for the representation of mathematical relations, which serves the current needs of the other partial models of OntoCAPE. In the future, the ontology modules of **supporting_concepts** could be replaced by generic ontologies developed and tested by others in a more systematic manner.

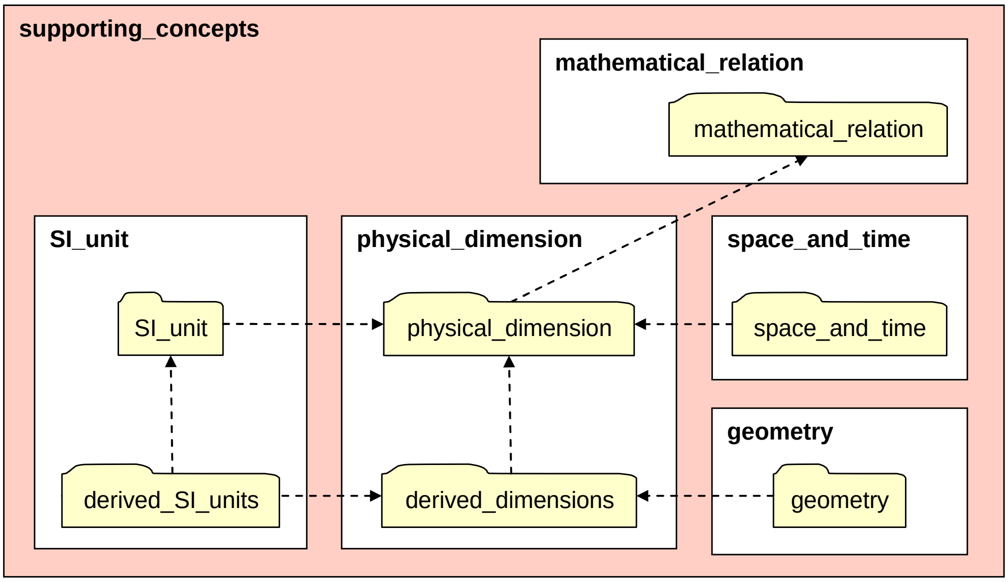

Fig. 1: Overview on partial model **supporting_concepts**

As depicted in Fig. 1, **supporting_concepts** comprises five subordinate partial models, which are **mathematical_relation**, **physical_dimension**, **SI_unit**, **space_and_time**, and **geometry**.

2 Mathematical Relation

The ontology module *mathematical_relation* introduces concepts to represent mathematical expressions. However, it is not the objective of this module to describe mathematical models. Rather, this module provides auxiliary concepts, which are utilized by other ontology modules (e.g., for the definition of units, cf. Sect. 4.1).

Mathematical relations and expressions are represented by means of binary trees[1]. . The nodes of the tree represent the operands and the operators, and the structure of the tree specifies the order of evaluation. For example, the relation $a + b = 2$ can be represented through the binary tree depicted in Fig. 2.

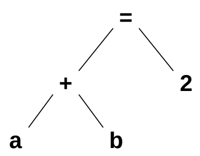

Fig. 2: Tree representation of the equation $a + b = 2$

The leaves of the tree represent the *operands* (here: a, b, 2), the internal nodes represents the *functional operators* (here: +), and the root node denotes either a *relational operator* (here: =) or another functional operator.

The ontological representation of binary trees is adopted from the design pattern *binary tree*, as defined in the Meta Model (cf. [meta_model/binary_tree]): A *node* may have a left child and a right child; special *nodes* are *root node*, *leaf*, and *internal node*.

Additionally, the class *node value* is introduced, which assigns a *node* to a mathematical concept (cf. Fig. 3). A node value is either an *operand* or an *operator*. The *operand* class represents the variables, parameters, or numbers of a mathematical expression. An *operator* is either a *relational operator* or a *functional operator*. A *relational operator* represents mathematical relations, such as equality or greater than. A *functional operator* denotes mathematical operations, such as addition, exponentiation, or logarithm. Two types of *functional operators* are distinguished: A *binary operator*, which takes two arguments, and a *unary operator*, taking only a single argument[2].

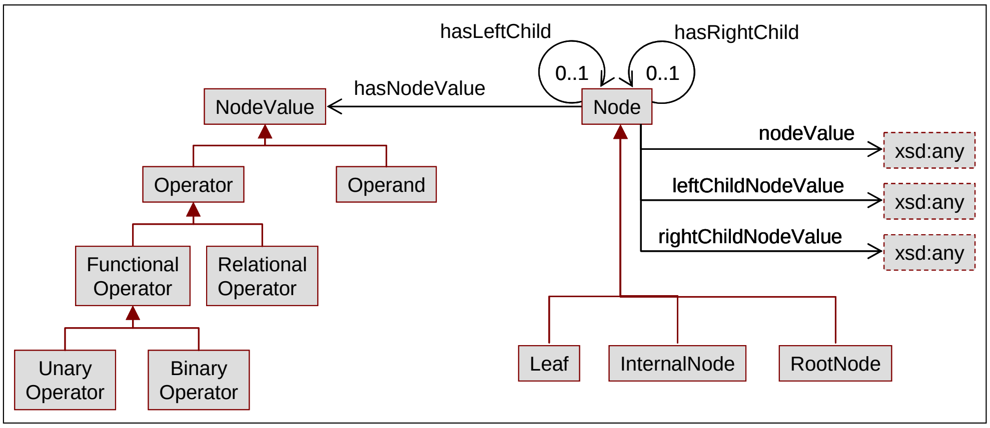

Fig. 3: Class diagram of the ontology module *mathematical relation*

Depending on the type of *node*, some restrictions are imposed on the type of its *node value*:

- A *root node* has only *operators* as *node values*.

[1] Currently, it is applied to algebraic expressions only. However, the approach can probably be extended to complex PDE systems.

[2] In this case, the corresponding *node* has only a single child *node*, which represents the *operand*.

- An *internal node* has only *functional operators* as node values.
- A *leaf* has only *operands* as node values.
- If a *unary operator* is used, the corresponding *node* has exactly one child.
- If a *binary operator* is used, the corresponding *node* has exactly two children.

It is not compulsory that the operands of a mathematical expression are represented as instances of the class *operand*; instead, they may be represented as values of the attribute nodeValue. An example is presented in Fig. 4, which shows the OntoCAPE representation of the equation $a + b = 2$. The variables a and b are values of type ‘string’, the number 2 is of type ‘float’ (both ‘string’ and ‘float’ are built-in XML schema datatypes; cf. Biron et al. 2004).

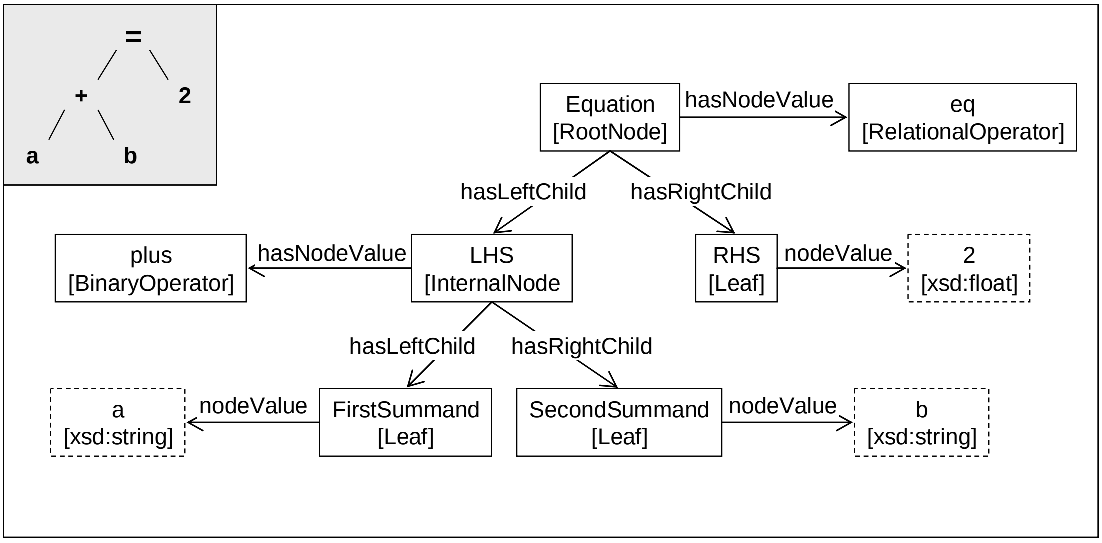

Fig. 4: OntoCAPE representation of $a + b = 2$; operands are represented through attribute values.

Moreover, a shorthand notation may be used to substitute child *nodes* that apply the nodeValue attribute. These *nodes* must be *leaves* that have no other function but to carry the value of the attribute nodeValue. In this case, they may be replaced by the attributes leftChildNodeValue and rightChildNodeValue, which are applied instead of the relations hasLeftChild and hasRightChild. Fig. 5 shows again the representation of the equation $a + b = 2$, this time in shorthand notation. Compared to Fig. 4, the child *nodes* RHS, FirstSummand, and SecondSummand have been pruned, and their respective node values are represented through the attributes leftChildNodeValue and rightChildNodeValue.

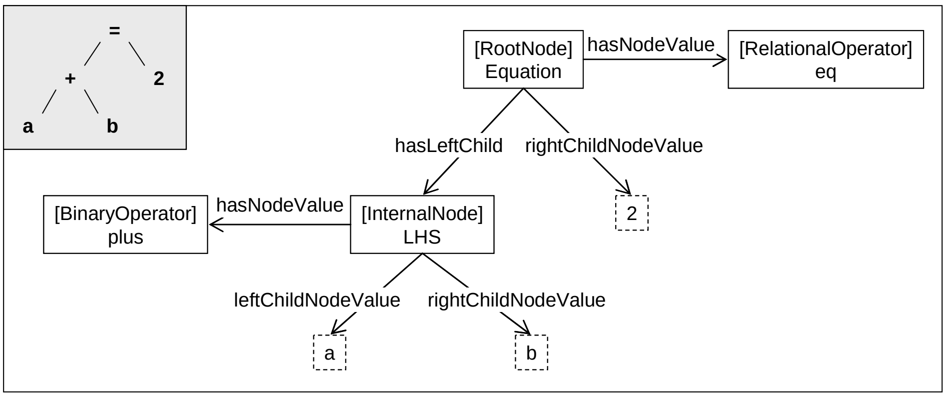

Fig. 5: Shorthand representation of the equation $a + b = 2$.

Usage

In the module *physical_dimension*, the concepts of *mathematical_relation* are utilized to establish mathematical relations between physical dimensions: (e.g., *velocity* = *length / time*). Similarly, in the ontology module *SI_unit*, mathematical relations can be established between different units (e.g., *Newton = kilogram * meter / second^2*). In these applications, a single *operand* is frequently involved in multiple mathematical expressions. Take for instance the linear equation system

$$a + b = 2 \qquad (1)$$

$$a - b = 0 \qquad (2)$$

Here, the *operand* a appears both in equation (1) and in equation (2); the same is true for *operand* b. There are two possible representation patterns for such a situation:

Representation pattern 1 is shown in Fig. 6. Here, separate *nodes* (*leaves*) are created for each occurrence of an *operand*; for instance, the *leaf* Var1_Eq1 represents the occurrence of the *operand* a in equation (1), Var1_Eq2 represents the occurrence of a in equation (2).

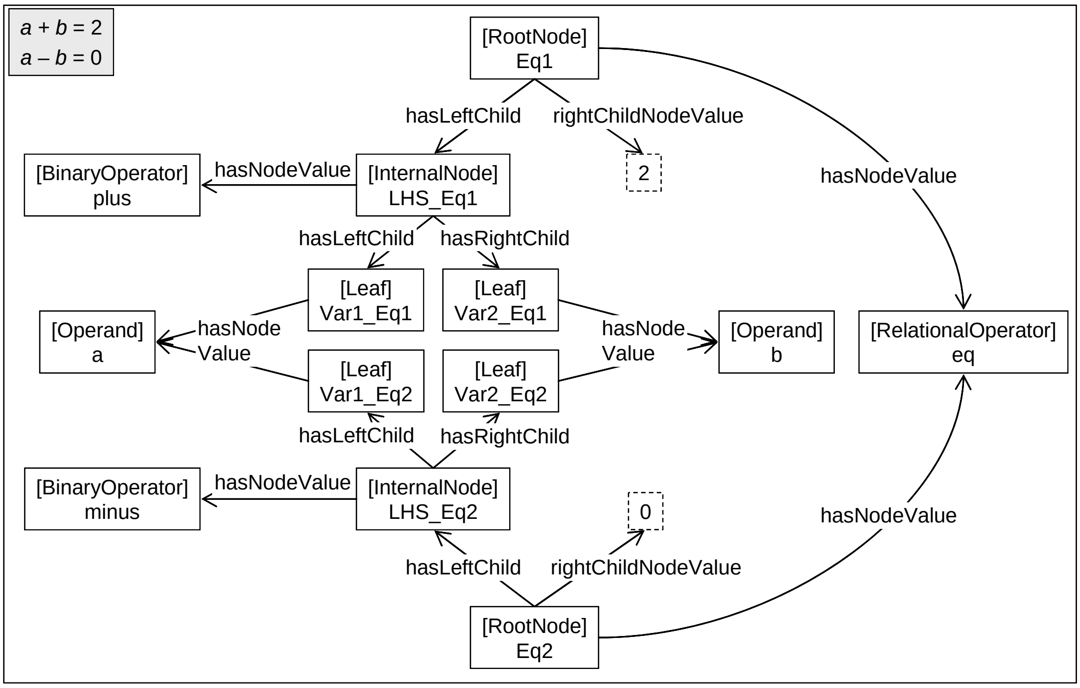

Fig. 6: Representation pattern 1

Representation pattern 1 is rather straight-forward. However, it may lead to a significant overhead, as a new *node* must be created for each occurrence of an *operand*. This can be avoided by representation pattern 2, which is exemplarily presented in Fig. 7: Here, only one *node* (*leaf*) is created for each *operand*; this *node* forms part of multiple trees. For instance, Var1 is a *leaf* of both the tree that represents equation (1) and of the tree that represents equation (2).

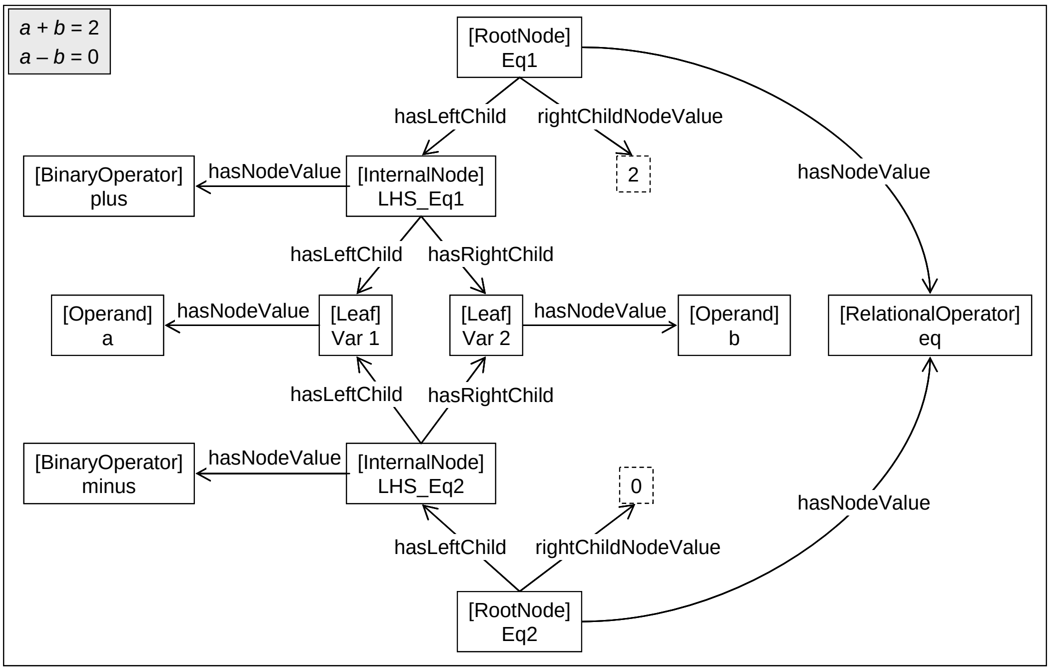

Fig. 7: Representation pattern 1

Pattern 2 reduces the representation effort, which is a considerable relief when modeling large equation systems or nested definition expressions, such as those in the ontology module *SI_unit*. On the other hand, pattern 2 is not as straight-forward as representation pattern 1. The user must choose which one he/she prefers.

Note that representation pattern 1 can be enforced by adding the constraint that each *node* must have only a single parent.

Concept Descriptions

Individual concepts of the module *mathematical relations* are defined below.

Classes

Binary operator

Description

A *binary operator* denotes a binary operation between two expressions. Typical binary operations are addition, subtraction, multiplication, division, and exponentiation.

Relations

- A *binary operator* is a specialization of a functional *operator*.

Usage

The following *binary operators* are predefined: plus (addition), minus (subtraction), times (multiplication), divide (division), power (exponentiation), and root (n^{th} root). Further *binary operators* may be added by the user. Labels and definitions of *binary operators* should be adopted from the Mathematical Markup Language, MathML (Carlisle et al., 2003).

Functional operator

Description

A *functional operator* denotes a mathematical function.

Definition

A *functional operator* is either a *unary operator* or a *binary operator*.

Relations

- A *functional operator* is a specialization of *operator*.

Internal node

Description

An *internal node* is a *node* that has at least one parent and at least one child.

Definiton

A *node* that has a parent *node* as well as a child. The child may be represented either as a *node* or through the attribute leftChildNodeValue or rightChildNodeValue.

Relations

- *Internal node* is a specialization of *node*.
- *Internal node* is a specialization of the meta class *internal node*.
- An *internal node* has at least one parent *node*.
- An *internal node* has at least one child *node* or one leftChildNodeValue or one rightChild NodeValue.
- An *internal node* has exactly one *functional operator* as node value.
- If the node value is a *unary operator*, then the *internal node* has exactly one child *node* (or, if shorthand notation is applied, exactly one leftChildNodeValue or rightChildNodeValue).
- If the node value is a *binary operator* or a *relational operator*, then the *internal node* has exactly two child *nodes* (or alternatively one leftChildNodeValue and one rightChildNodeValue).

Leaf

Description

A *leaf* is a *node* without any children.

Definiton

A *node* that has neither a child *node*, nor a leftChildNodeValue, nor a rightChildNodeValue.

Relations

- *Leaf* is a specialization of a *node*.
- A *leaf* has at least one parent *node*.
- A *leaf* has no child *node*.
- A *leaf* has no leftChildNodeValue.
- A *leaf* has no rightChildNodeValue.
- A *leaf* has only *operands* as node values.

Node

Description

A *node* is the basic element of a binary tree. It can be linked to up to two child *nodes*.

Definition

A *node* is either a *leaf* or a *root node* or an *internal node*.

Relations

- *Node* is a specialization of the meta class *binary_tree:node*.
- A *node* has zero or one left child, which is either represented as a *node* or (if shorthand notation is applied) represented through the attribute leftChildNodeValue.
- A *node* has zero or one right child *node* , which is either represented as a *node* or, alternatively, through the attribute rightChildNodeValue.
- A node may have some parent *node*.
- A *node* has exactly one node value, which is either represented as a *node value* or through the attribute nodeValue.

Node value

Description

A *node value* represents a component part of mathematical expression. It can be either an *operator* or an *operand*.

Definition

A *node value* is either an *operand* or an *operator*.

Relations

- *Node value* is a specialization of the meta class *non-exhaustive value set*.

Operand

Description

An *operand* is one of the inputs of a *functional operator*.

Relations

- An *operand* is a specialization of *node value*.

Operator

Description

An *operator* is either a *relational operator* or a *functional operator*.

Definition

An *operator* is either a *relational operator* or a *functional operator*.

Relations

- An *operator* is a specialization of *node value*.

Relational operator

Description

A *relational operator* denotes a mathematical relation, such as equality or greater than, between two expressions.

Relations

- A *relational operator* is a specialization of an *operator*.

<u>Usage</u>

The following *relational operators* are predefined: eq (equal), geq (greater or equal), gt (greater than), leq (less or equal), lt (less than), and neq (not equal). Further *binary operators* may be added by the user. Labels and definitions of *relational operators* should be adopted from the Mathematical Markup Language, MathML (Carlisle et al., 2003).

Root node

<u>Description</u>

A *root node* is the root element of a binary tree. All other *nodes* are descendents of the *root node*.

<u>Definiton</u>

A *node* without any parent *node*.

<u>Relations</u>

- A *root node* is a specialization of a *node*.
- A *root node* has at least one child *node*.
- A *root node* has exactly one *operator* as a node value.
- If the node value is a *unary operator*, then the *root node* has exactly one child *node* (or, in shorthand notation, exactly one leftChildNodeValue or rightChildNodeValue).
- If the node value is a *binary operator* or a *relational operator*, then the *root node* has exactly two child *nodes* (or alternatively one leftChildNodeValue and one rightChildNodeValue).

Unary operator

<u>Description</u>

A *unary operator* denotes a mathematical operation which takes a single argument. Typical binary operations are squaring, root extraction, or factorial.

<u>Relations</u>

- A *unary operator* is a specialization of a *functional operator*.

<u>Usage</u>

The following *unary operators* are predefined: cos (cosinus function), exp (exponentiation function), factorial (factorials), ln (natural logarithm), sin (sinus function). Further *binary operators* may be added by the user. Labels and definitions of *unary operators* should be adopted from the Mathematical Markup Language, MathML (Carlisle et al., 2003).

Relations

hasAncestor

<u>Description</u>

The ancestors of a *node* are the *nodes* that precede the current *node* in the tree (i.e., the *node*'s parent, grandparent, etc.).

<u>Characteristics</u>

- Specialization of the meta relation hasAncestor.

- Domain: *node*
- Range: *node*
- Inverse: hasDescendent
- Transitive

hasChild

Description

The relation hasChild points to the children of a *node*; it subsumes the relations hasLeftChild and hasRightChild.

Characteristics

- Specialization of the meta relation hasChild.
- Specialization of hasDescendent.
- Domain: *node*
- Range: *node*
- Inverse: hasParent

hasDescendent

Description

The descendents of a *node* are the *nodes* that succeed the current *node* in the tree (i.e., the *node*'s children, grandchildren, etc.).

Characteristics

- Specialization of the meta relation hasDescendent.
- Domain: *node*
- Range: *node*
- Inverse: hasAncestor
- Transitive

hasLeftChild

Description

The relation hasLeftChild links a parent *node* to its left child *node*.

Characteristics

- Specialization of the meta relation hasLeftChild.
- Specialization of hasChild
- Domain: *node*
- Range: *node*

hasNodeValue

Description

The relation hasNodeValue links a *node value* to a *node*.

Characteristics

- Specialization of the meta relation object-featureRelation.

- Domain: *node*.
- Range: *node value*.

hasParent

Description

The relation hasParent denotes the parent of a *node*.

Characteristics

- Specialization of the meta relation hasParent.
- Specialization of hasAncestor
- Domain: *node*
- Range: *node*
- Inverse: hasChild

hasRightChild

Description

The relation hasRightChild links a parent *node* to its right child *node*.

Characteristics

- Specialization of the meta relation hasRightChild.
- Specialization of hasChild
- Domain: *node*
- Range: *node*

isLeftChildOf

Description

The relation isLeftChildOf points from the left child *node* to its parent *node*.

Characteristics

- Specialization of the meta relation isLeftChildOf.
- Specialization of hasParent
- Domain: *node*
- Range: *node*

isRightChildOf

Description

The relation isRightChildOf points from the right child *node* to its parent *node*.

Characteristics

- Specialization of the meta relation isRightChildOf.
- Specialization of hasParent
- Domain: *node*
- Range: *node*

Attributes

leftChildNodeValue

Description

The attribute leftChildNodeValue can be used as a shorthand to substitute a left child *node*, the node value of which is represented through the attribute nodeValue.

Characteristics

- Domain: *node*
- Datatype: any (built-in XML Schema Datatype)

rightChildNodeValue

The attribute leftChildNodeValue can be used as a shorthand to substitute a left child *node*, the node value of which is represented through the attribute nodeValue.

Characteristics

- Domain: *node*
- Datatype: any (built-in XML Schema Datatype)

nodeValue

Description

The attribute nodeValue indicates an operand (usually a number) in a mathematical expression.

Characteristics

- Domain: *node*
- Datatype: any (built-in XML Schema Datatype)

Individuals

The labels and definitions of the following individuals have been adopted from the Mathematical Markup Language, MathML (Carlisle et al., 2003).

cos

Description

The individual cos is the unary operator that represents the cosine function.

Characteristics

- Instance of *unary operator*.

divide

Description

The individual divide represents the binary division operator; it indicates the mathematical operation A "divided by" B.

Characteristics

- Instance of *binary operator*.

eq

Description

The individual eq represents the relational operator "equals".

Characteristics

- Instance of *relational operator*.

exp

Description

The individual exp is the unary operator that represents the exponentiation function.

Characteristics

- Instance of *unary operator*.

factorial

Description

The individual factorial is the unary operator used to construct factorials. Factorials are defined by n! = n*(n-1)* ... *1.

Characteristics

- Instance of *unary operator*.

geq

Description

The individual geq represents the relational operator "greater than or equal to".

Characteristics

- Instance of *relational operator*.

gt

Description

The individual gt represents the relational operator "greater than".

Characteristics

- Instance of *relational operator*.

leq

Description

The individual leq represents the relational operator "lesser than or equal to".

Characteristics

- Instance of *relational operator*.

ln

Description

The individual ln is the unary operator that represents the ln function (natural logarithm).

Characteristics

- Instance of *unary operator*.

lt

Description

The individual lt represents the relational operator "lesser than".

Characteristics

- Instance of *relational operator*.

minus

Description

The individual minus represents the binary subtraction operator. It constructs the mathematical operation A "minus" B.

Characteristics

- Instance of *binary operator*.

neq

Description

The individual neq represents the relational operator "not equal to".

Characteristics

- Instance of *relational operator*.

plus

Description

The individual plus represents the binary addition operator.

Characteristics

- Instance of *binary operator*.

power

Description

The individual power represents the binary powering operator that is used to construct expressions such as A "to the power of" B. In particular, it is the operation for which A "to the power of" 2 is equivalent to A*A.

Characteristics

- Instance of *binary operator*.

root

Description

The individual root represents the binary operator that is used to construct the n^{th} root of an expression. The first argument "a" is the expression and the second object "n" denotes the root, as in (a) ^ (1/n)

Characteristics

- Instance of *binary operator*.

sin

Description

The individual sin is the unary operator that represents the sine function.

Characteristics

- Instance of *unary operator*.

times

Description

The individual times represents the binary multiplication operator.

Characteristics

- Instance of *binary operator*.

3 Physical Dimension (Partial Model)

The partial model **physical_dimension** comprises two ontology modules. The main module, *physical_dimension*, defines a set of base dimensions and establishes the proceedings to derive further physical dimensions from these base dimensions. It is extended by the ontology module *derived_dimensions*, which introduces a number of frequently used derived dimensions.

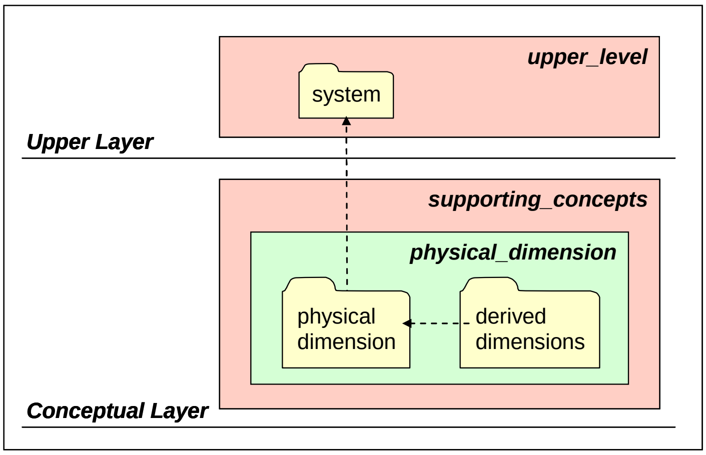

Fig. 8: Overview on partial model **physical_dimension**

3.1 Physical dimension (Ontology module)

The concept of a *physical dimension* has been introduced in the *system* module. Basically, it serves as a characteristic that can be associated with *physical properties*, *physical quantities*, and *units* for the purpose of classification or differentiation.

The relations between *physical dimensions* can be described through mathematical equations, such as

Joule = Newton * *meter*.

Multiplication and exponentiation to a real power are the permissible operations on physical dimensions. Exploiting this property, the majority of *physical dimensions* can be mathematically derived from a small set of dimensions that we call *fundamental* or *base dimensions* by means of multiplication and exponentiation operations; *physical dimensions* that can be defined in terms of the base dimensions are called *composite* or *derived dimensions*. For example, the derived dimension 'velocity' can be defined as the product of the base dimension 'length' and the base dimension 'time' raised to the power of minus one.

There is no intrinsic property of a dimension that makes it fundamental (Gruber and Olsen, 1994); hence, the choice of the base dimensions is a matter of convention. While the definition of the *physical dimension* concept in the *system* module still allows for arbitrary conventions (or 'systems of dimensions', as they will be called hereafter), the *physical_dimension* module establishes the SI system of dimensions, which comprises the base dimensions of *length*, *time*, *thermodynamic temperature*, *mass*, a*mount of substance*, *electric current*, and *luminous intensity* (BIPM, 2006).

Fig. 9 presents the major concepts of the *physical_dimension* module. The class *physical dimension*, introduced in the ontology module *system*, is exhaustively partitioned into three subclasses:

- The class *base dimension* is defined as the exhaustive enumeration of the individuals length, time, thermodynamic_temperature, mass, amount_of_substance, electric_current_strength, and luminous_ intensity.
- The class *supplementary dimensions* subsumes further fundamental dimensions that do not form part of the SI system of units and are therefore not classified as *base dimensions*. Currently, the class holds two individuals: amount_of_money characterizes monetary *physical quantities* and *units*. For the characterization of dimensionless *physical quantities*, the concept of an

identity_dimension is introduced. Prominent examples of dimensionless *physical quantities* are dimensionless numbers like the Reynolds number, or counting quantities like the partition function in statistical thermodynamics or the number of trays in a distillation column. According to Gruber and Olsen (1994), the identity_dimension represents the identity element for multiplication over physical dimensions. That means that the product of the identity_dimension and any other *physical dimension* is that other *physical dimension*.

- All other *physical dimensions* are *derived dimensions*. These can be subdivided by further subclasses that, for example, group physical dimensions by fields of science. A possible classification is provided by the ontology module *derived_dimensions*: (cf. Sect. 3.2).

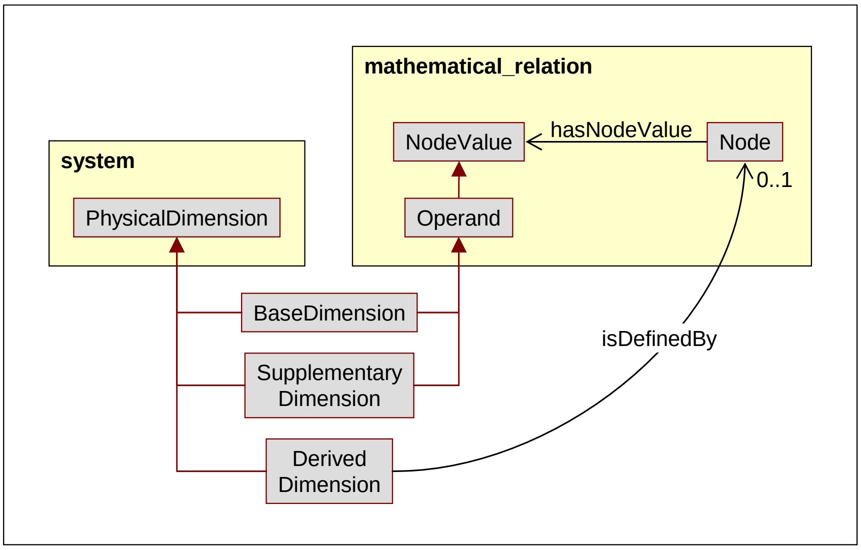

Fig. 9: Class diagram of the ontology module *physical_dimension*

The mathematical definition of a *derived dimension*[3] can be specified using the concepts defined in ontology module *mathematical_relation*: A *derived dimension* is defined by a *node*, which represents the root of a definition tree; the leaves of the definition tree have either *base dimensions* or *fundamental dimensions* as *node values*. To this end, *base dimension* and *supplementary dimension* are declared as subclasses of *operand*; that way, their instances may appear as operands in the definition equation. An example is given in Fig. 10: velocity is declared as an instance of *derived dimension*; it is defined by Velocity *node*, which is in turn defined by its two child *nodes*, Length and Time, which are concatenated by the *binary operator* divide. Length and Time have the *base dimensions* length and time, respectively, as node values.

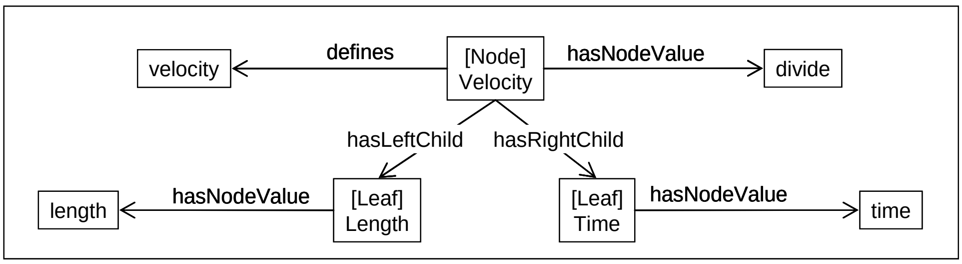

Fig. 10: Definition of the *derived dimension* ‘velocity’

[3] Implementation advice: Unfortunately, the definition of *derived dimensions* through the above mechanism scales badly with current reasoners. Therefore, such definitions should be omitted in order to enhance the reasoning performance, unless they are definitely required by the respective application.

Concept Definitions

Individual concepts of the module *physical_dimension* are defined below.

Classes

Base dimension

Description

Most *physical dimensions* can be mathematically derived from a small set of dimensions that we call *base dimensions*. Such a set of *base dimensions* is chosen by convention. In OntoCAPE, we adopt the base dimensions of the SI system of units (BIPM, 2006), which are *length*, *time*, *thermodynamic temperature*, *mass*, a*mount of substance*, *electric current*, and *luminous intensity*.

Definition

A *base dimension* is one of the following individuals: amount_of_substance, electric_current, length, luminous_intensity, mass, thermodynamic_temperature, or time.

Relations

- *Base dimension* is a specialization of a *physical dimension*.
- *Base dimension* is a subclass of *operand*.

Derived dimension

Description

A *derived dimension* is a *physical dimension* that can be defined as a product of powers of the *base dimensions*. For example, the *derived dimension* velocity can be defined as the ratio of the base dimensions length and time.

Relations

- *Derived dimension* is a specialization of a *physical dimension*.
- A *derived dimension* can only be defined by a *node*.
- A *derived dimension* can be defined by at most one *node*.

Physical dimension (continued)

Relations

A *physical dimension* is either a *base dimension* or a *supplementary dimension* or a *derived dimension*.

Supplementary dimension

Description

This class subsumes fundamental dimensions that do not form part of the SI system of units and are therefore not classified under the *base dimension* class.

Relations

- *Supplementary dimension* is a specialization of a *physical dimension*.
- *Supplementary dimension* is a subclass of *operand*.

Relations

isDefinedBy

Description

The relation defines links a *note* to a *derived dimension*, which represents the right hand side of a definition equation for the *derived dimension*.

Characteristics

- Domain: *derived dimension*
- Range: *node*

Individuals

amount_of_money

Description

An amount of money in an arbitrary currency

Characteristics

- Instance of *fundamental dimension*
- Different from identity_dimension

amount_of_substance

Description

The number of elementary entities contained in a body, or a in system of bodies (Chertov, 1997).

Characteristics

- Instance of *supplementary dimension*
- Different from electric_current, length, luminous_intensity, mass, thermodynamic_temperature, time

electric_current

Description

The time derivative from an electric charge sustained by a charge carrier through an observed surface (Chertov, 1997).

Characteristics

- Instance of *base dimension*
- Different from amount_of_substance, length, luminous_intensity, mass, thermodynamic_temperature, time

identity_dimension

Description

According to Gruber & Olsen (1994) the identity_dimension is defined as the identity element for multiplication over *physical dimensions*. That means that the product of the identity_dimension and any other *physical dimension* is that other *physical dimension*.

Characteristics

- Instance of *supplementary dimension*
- Different from amount_of_money

length

Description

The physical dimension which characterizes the space and distance traveled by bodies or their parts along a given line (Chertov, 1997).

Characteristics

- Instance of *base dimension*
- Different from amount_of_substance, electric_current, luminous_intensity, mass, thermodynamic_ temperature, time

luminous_intensity

Description

The radiant flux emitted by a source of radiation in a given direction inside a small solid angle in relation to this solid angle (Chertov, 1997).

Characteristics

- Instance of *base dimension*
- Different from amount_of_substance, electric_current, length, mass, thermodynamic_ temperature, time

mass

Description

The physical dimension which characterizes the inert and gravitational properties of material objects (Chertov, 1997).

Characteristics

- Instance of *base dimension*
- Different from amount_of_substance, electric_current, length, luminous_intensity, thermodynamic_temperature, time

thermodynamic_temperature

Description

The temperature calculated according to a thermodynamic temperature scale from absolute zero (Chertov, 1997).

Characteristics

- Instance of *base dimension*
- Different from amount_of_substance, electric_current, length, luminous_intensity, mass, time

time

Description

The physical dimension characterizing the successive change in phenomena and the states of matter which determines the duration of phenomenal being (Chertov, 1997).

Characteristics

- Instance of *base dimension*
- Different from amount_of_substance, electric_current, length, luminous_intensity, mass, thermodynamic_temperature

3.2 Derived Dimensions

The ontology module *derived_dimensions* provides a number of frequently used *derived dimensions*. These predefined *derived dimensions* are classified into categories which have been suggested by Chertov (1997):

- The class *space and time* subsumes those *physical dimensions* that can be derived from the *base dimensions* length and time.
- *Periodic phenomena* assembles *derived dimensions* with a periodic character, such as frequency or period.
- *Mechanics* subsumes *derived dimensions* that are relevant for the field of mechanics.
- Similarly, the instances of the class *thermodynamics* are of relevance in the area of thermodynamics and transport phenomena.
- Finally, *electricity and magnetism* is intended to subsume *derived dimensions* that are connected with the phenomena of electricity or magnetism.

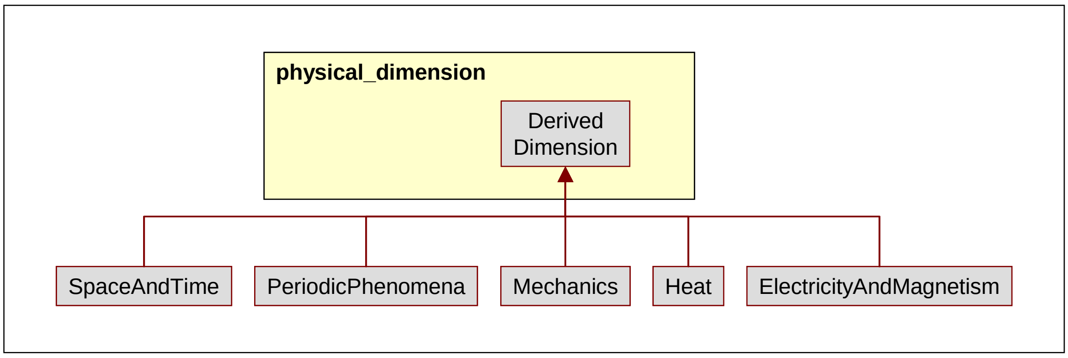

Fig. 11: Classification of *derived dimensions*, according to Chertov (1997)

Concept Definitions

Individual concepts of the module *derived_dimensions* are defined below.

Classes

Electricity and magnetism

<u>Description</u>

The class *electricity and magnetism* subsumes *derived dimensions* that are connected with the phenomena of electricity or magnetism.

<u>Relations</u>

- *Electricity and magnetism* is a subclass of *derived dimension*.

Thermodynamics

<u>Description</u>

The class *thermodynamics* subsumes *derived dimensions* that are of relevance in the area of heat transfer and thermodynamics.

Relations

- *Heat* is a subclass of *derived dimension*.

Mechanics

Description

The class *mechanics* subsumes *derived dimensions* that are of relevance for the field of mechanics.

Relations

- *Mechanics* is a subclass of *derived dimension*.

Periodic phenomena

Description

The class *periodic phenomena* subsumes *derived dimensions* with a periodic character, such as frequency or period.

Relations

- *Periodic phenomena* is a subclass of *derived dimension*.

Space and time

Description

The class *space and time* subsumes the *physical dimensions* that can be derived from the *base dimensions* length and time.

Relations

- *Space and time* is a subclass of *derived dimension*.

Individuals

Acceleration

Description

Acceleration is the first time derivative from velocity (Chertov, 1997).

Characteristics

- Instance of *space and time*.
- Different from angular_acceleration, angular_velocity, area, plane_angle, solid_angle, velocity, volume.

Angular Acceleration

Description

Angular Acceleration is the first time derivative from the angular velocity (Chertov, 1997).

Characteristics

- Instance of *space and time*.
- Different from acceleration, angular_velocity, area, plane_angle, solid_angle, velocity, volume.

Angular Velocity

Description

Angular Velocity is the first time derivative from the deflection angle (Chertov, 1997).

Characteristics

- Instance of *space and time*.
- Different from acceleration, angular_acceleration, area, plane_angle, solid_angle, velocity, volume.

Area

Description

Characterizes the plane and curved surfaces of a physical body (Chertov, 1997).

Characteristics

- Instance of *space and time*.
- Different from acceleration, angular_acceleration, angular_velocity, plane_angle, solid_angle, velocity, volume.

Damping Coefficient

Description

The inverse to the time interval during which the amplitude decreases *e* (Euler number, i.e., 2.718…) times (Chertov, 1997).

Characteristics

- Instance of *periodic phenomena*.
- Different from frequency, oscillation_phase, period, rotational_frequency.

Density

Description

Density is the ratio of the mass to the volume of a body's element (Chertov, 1997).

Characteristics

- Instance of *Mechanics*.
- Different from dynamic_viscosity, force, kinematic_viscosity, moment_of_force, moment_of_inertia, momentum, power, pressure, specific-volume, surface_tension, work.

Dynamic Viscosity

Description

Dynamic viscosity is the coefficient of proportionality in the internal friction force formula (Chertov, 1997).

Characteristics

- Instance of *Mechanics*.
- Different from density, force, kinematic_viscosity, moment_of_force, moment_of_inertia, momentum, power, pressure, specific-volume, surface_tension, work.

Electric charge

Description

Electronic charge is the measure equal to the product of electric current into time during which the current flows (Chertov, 1997).

Characteristics

- Instance of *Electricity and Magnetism*

Entropy

Description

In thermodynamics, entropy is a measure of the unavailability of a system's energy to do work (Daintith, 2005).

Characteristics

- Instance of *Heat*.
- Different from heat_capacity, heat_flow_rate, internal_energy, quantity_of_heat, specific_entropy, specific_heat_capacity, temperature_coefficient, thermal_conductivity.

Frequency

Description

Frequency is the inverse of the period (Chertov, 1997).

Characteristics

- Instance of *periodic phenomena*.
- Different from damping_coefficient, oscillation_phase, period, rotational_frequency.

Force

Description

The degree of mechanical influence exerted on a body by other bodies (Chertov, 1997).

Characteristics

- Instance of *Mechanics*.
- Different from density, dynamic_viscosity, kinematic_viscosity, moment_of_force, moment_of_inertia, momentum, power, pressure, specific-volume, surface_tension, work.

Heat Capacity

Description

The ratio of heat required to warm a physical body to the difference in temperature (Chertov, 1997).

Characteristics

- Instance of *Heat*.
- Different from entropy, heat_flow_rate, internal_energy, quantity_of_heat, specific_entropy, specific_heat_capacity, temperature_coefficient, thermal_conductivity.

Heat Flow Rate

Description

Heat Flow Rate is the ratio of the quantity of heat which passes a surface to time (Chertov, 1997).

Characteristics

- Instance of *Heat*.
- Different from entropy, heat_capacity, internal_energy, quantity_of_heat, specific_entropy, specific_heat_capacity, temperature_coefficient, thermal_conductivity.

Internal Energy

Description

Internal Energy is the energy from the random thermal movement of all micro particles in a system (Chertov, 1997).

Characteristics

- Instance of *Heat*.
- Different from entropy, heat_capacity, heat_flow_rate, quantity_of_heat, specific_entropy, specific_heat_capacity, temperature_coefficient, thermal_conductivity.

Kinematic Viscosity

Description

Kinematic viscosity is the ratio of the dynamic viscosity of a medium to its density (Chertov, 1997).

Characteristics

- Instance of *Mechanics*.
- Different from density, dynamic_viscosity, force, moment_of_force, moment_of_inertia, momentum, power, pressure, specific-volume, surface_tension, work.

Moment of Force

Description

In relation to a certain point, the product of the force and the arm, i.e. the distance between the direction of the force and the point (Chertov, 1997).

Characteristics

- Instance of *Mechanics*.
- Different from density, dynamic_viscosity, force, kinematic_viscosity, moment_of_inertia, momentum, power, pressure, specific-volume, surface_tension, work.

Moment of Inertia

Description

Moment of inertia is the mass of a material point per square distance to the axis of rotation (Chertov, 1997).

Characteristics

- Instance of *Mechanics*.
- Different from density, dynamic_viscosity, force, kinematic_viscosity, moment_of_force, momentum, power, pressure, specific-volume, surface_tension, work.

Momentum

Description

Momentum is the mass of a physical body at its velocity (Chertov, 1997).

Characteristics

- Instance of *Mechanics*.
- Different from density, dynamic_viscosity, force, kinematic_viscosity, moment_of_force, moment_of_inertia, power, pressure, specific-volume, surface_tension, work.

Oscillation Phase

Description

Oscillation phase is an independent variable of a function describing the quantity which changes according the law of harmonic vibrations (Chertov, 1997).

Characteristics

- Instance of *periodic phenomena*.
- Different from damping_coefficient, frequency, period, rotational_frequency.

Period

Description

The time interval during which a cycle of periodic process is completed (Chertov, 1997).

Characteristics

- Instance of *periodic phenomena*.
- Different from damping_coefficient, frequency, oscillation_phase, rotational_frequency.

Plane Angle

Description

Geometric figure formed by two rays (sides of the angle) extending from one point (Chertov, 1997).

Characteristics

- Instance of *space and time*.
- Different from acceleration, angular_acceleration, angular_velocity, area, solid_angle, velocity, volume.

Power

Description

Power is the ratio of work to time interval during which the work is completed (Chertov, 1997).

Characteristics

- Instance of *Mechanics*.
- Different from density, dynamic_viscosity, force, kinematic_viscosity, moment_of_force, moment_of_inertia, momentum, pressure, specific-volume, surface_tension, work.

Pressure

Description

Pressure is the ratio between the perpendicular force acting on a surface element to the area of the element (Chertov, 1997).

Characteristics

- Instance of *Mechanics*.
- Different from density, dynamic_viscosity, force, kinematic_viscosity, moment_of_force, moment_of_inertia, momentum, power, specific-volume, surface_tension, work.

Quantity of Heat

Description

Quantity of heat is the part of internal energy which is transferred spontaneously, with no external influence from more to less heated physical body (Chertov, 1997).

Characteristics

- Instance of *Heat*.
- Different from entropy, heat_capacity, heat_flow_rate, internal_energy, specific_entropy, specific_heat_capacity, temperature_coefficient, thermal_conductivity.

Rotational Frequency

Description

The number of rotations completed per unit time.

Characteristics

- Instance of *periodic phenomena*.
- Different from damping_coefficient, frequency, oscillation_phase, period.

Specific Entropy

Description

Specific Entropy is the ratio of entropy to the mass of a system (Chertov, 1997).

Characteristics

- Instance of *Heat*.
- Different from entropy, heat_capacity, heat_flow_rate, internal_energy, quantity_of_heat, specific_heat_capacity, temperature_coefficient, thermal_conductivity.

Specific Heat Capacity

Description

Specific heat capacity is the ratio of the heat capacity of a uniform physical body to its mass (Chertov, 1997).

Characteristics

- Instance of *Heat*.
- Different from entropy, heat_capacity, heat_flow_rate, internal_energy, quantity_of_heat, specific_entropy, temperature_coefficient, thermal_conductivity.

Specific Volume

Description

Specific volume is the ratio of the volume to the mass of a body's element (Chertov, 1997).

Characteristics

- Instance of *Mechanics*.
- Different from density, dynamic_viscosity, force, kinematic_viscosity, moment_of_force, moment_of_inertia, momentum, power, pressure, surface_tension, work.

Solid Angle

Description

The part of space enclosed within one cavity of conical surface with a closed directrix (Chertov, 1997).

Characteristics

- Instance of *space and time*.
- Different from acceleration, angular_acceleration, angular_velocity, area, plane_angle, velocity, volume.

Surface Tension

Description

Surface tension is the ratio of the force, acting on a segment of the free surface contour perpendicular to the contour and tangentially to the surface, and the length of the segment (Chertov, 1997).

Characteristics

- Instance of *Mechanics*.
- Different from density, dynamic_viscosity, force, kinematic_viscosity, moment_of_force, moment_of_inertia, momentum, power, pressure, specific-volume, work.

Temperature Coefficient

Description

The ratio of the relative change in a physical quantity to the temperature change reckoned from the initial temperature (Chertov, 1997).

Characteristics

- Instance of *Heat*.
- Different from entropy, heat_capacity, heat_flow_rate, internal_energy, quantity_of_heat, specific_entropy, specific_heat_capacity, thermal_conductivity.

Thermal Conductivity

Description

Thermal conductivity is the density of heat flux produced by heat conductivity at temperature gradient (Chertov, 1997).

Characteristics

- Instance of *Heat*.
- Different from entropy, heat_capacity, heat_flow_rate, internal_energy, quantity_of_heat, specific_entropy, specific_heat_capacity, thermal_conductivity.

Velocity

Description

Velocity is the first time derivative from movement (Chertov, 1997).

Characteristics

- Instance of *space and time*.
- Different from acceleration, angular_acceleration, angular_velocity, area, plane_angle, solid_angle, volume.

Volume

Description

The amount of space occupied by a three-dimensional object or region of space. (Chertov, 1997).

Characteristics

- Instance of *space and time*.

- Different from acceleration, angular_acceleration, angular_velocity, area, plane_angle, solid_angle, velocity.

Work

Description

Work is the scalar product of force into an elementary displacement (Chertov, 1997).

Characteristics

- Instance of *Mechanics*.
- Different from density, dynamic_viscosity, force, kinematic_viscosity, moment_of_force, moment_of_inertia, momentum, power, pressure, specific-volume, surface_tension.

4 SI Unit (Partial Model)

The partial model **SI_unit** comprises the ontology modules *SI_unit* and *derived_SI_units* (cf. Fig. 12). The former module introduces the set of base units of the SI system, and it establishes a mechanism to derive further units from these. The latter module defines a number of frequently used derived SI units by applying this mechanism.

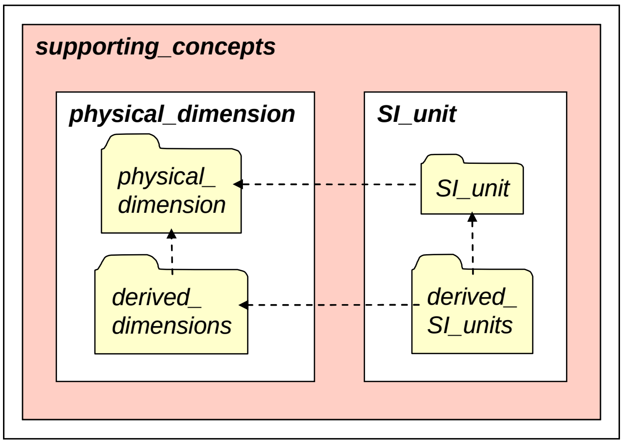

Fig. 12: Overview on the partial model **SI_unit**

4.1 SI Unit (Ontology Module)

The ontology module *SI_unit* establishes the International System of Units, which is also known as the SI system of units (cf. BIPM, 2006). According to Gruber and Olsen (1994), a *system of units* is “a class of units defined by composition from a base set of units, such that every instance of the class is "standard" unit for a physical dimension and every physical dimension has an associated unit.”

Some units are designated as *fundamental units* or *base units*, meaning that all other units can be derived from them. There is no intrinsic property that makes a unit fundamental; rather, a system of units defines a set of orthogonal base dimensions and assigns a base unit to each of them. Fig. 13 shows a listing of the base dimensions and the corresponding base units of the SI system.

base dimension	**base unit**
lenght	meter
mass	kilogram
time	second
electric current	ampere
thermodynamic temperature	kelvin
amount of substance	mole
Luminous intensity	candela

Fig. 13: The base dimensions and the corresponding base units of the SI system

Fig. 14 shows how the SI system of units is implemented in OntoCAPE. The *unit of measure* class has already been established in the ontology module *system*. Now, an *SI unit* is introduced as a special type of *unit of measure* that complies with the SI system; formally, an *SI unit* is defined as the exhaustive enumeration of its subclasses *SI base unit* and *SI derived unit*.

- The class *SI base unit* subsumes the seven base units of the SI system, namely A (ampere), cd (candela), K (kelvin), kg (kilogram), m (meter), mol (mole), and s (second).
- A *derived SI unit* is an *SI unit* that can be derived from one or several of the *SI base units* by means of multiplication and exponentiation operations.

Additionally, the classes *SI prefix* and *SI prefixed unit* are introduced: An *SI prefix* represents a decimal power by which an *SI unit* is multiplied; that way, one obtains a *prefixed derived unit* , which is a multiple or submultiple of the original *unit of measure*. So far, the following 20 prefixes have been approved by the General Conference on Weights and Measures: yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto (cf. BIPM, 2006).

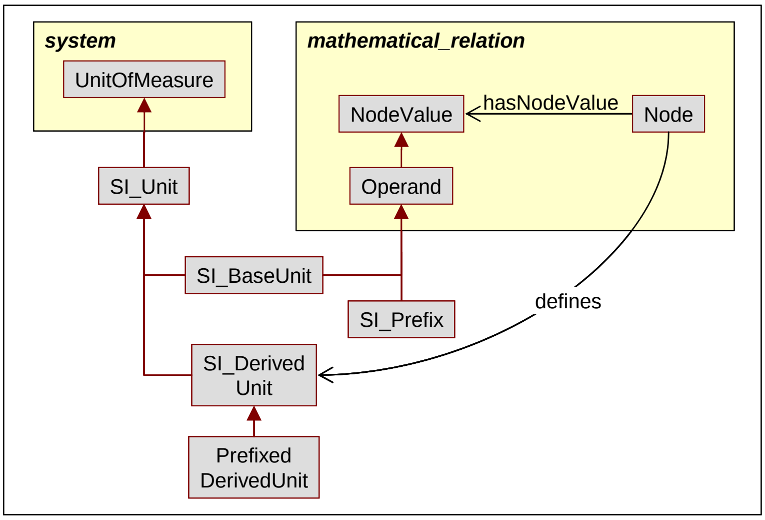

Fig. 14: Overview on the ontology module *SI_unit*

The definition equation for a particular *derived SI unit* can be indicated through the same mechanism that has already been used for the definition of *derived dimensions* (cf. Sect. 3.2): A *derived SI unit* is defined by a *node* (cf. ontology module *mathematical_relation*, Sect. 2), which represents the root of a definition tree; the leaves of the definition tree have *SI base units* or *SI prefixes* as *node values*. To this end, *SI base unit* and *SI prefix* are declared to be subclasses of *operand*; that way, their instances may appear as operands in the definition equation. An example is given in Fig. 15: The *derived SI unit* m_per_s is defined by the *node* Meter_Per_ Second. In turn, Meter_Per_Second is further specified through its two child *nodes*, Meter and Second, which are concatenated by the *binary operator* divide. Meter and Second have the *SI base units* m and s, respectively, as node values. A more complex example, the definition tree for a Newton (N), is shown in Fig. 16. Note that these derivation as shown in **Fehler! Verweisquelle konnte nicht gefunden werden.** and Fig.1 are defined in *derived_units*.

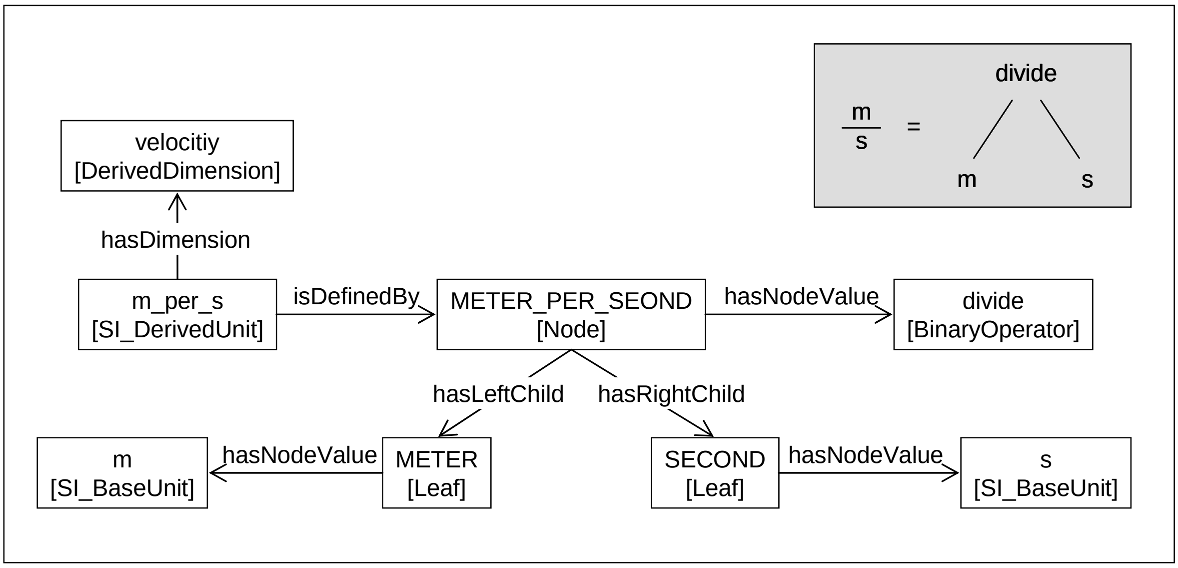

Fig. 15: Definition of the *derived SI unit* ‘m_per_s’

As explained before, the suggested mechanism for the indication of definition equations scales badly with current reasoners. Therefore, one should abstain from such definitions, unless they are definitely required by the respective application.

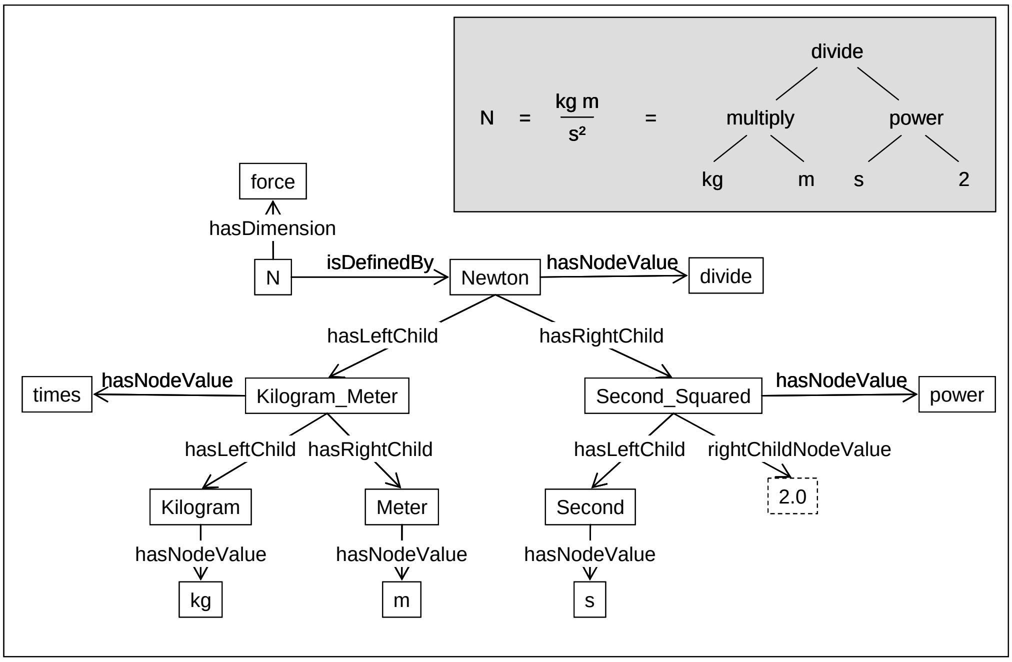

Fig.1: Definition of the *derived SI unit* ‘**N**’

Concept Descriptions

Individual concepts of the module *SI unit* are defined below.

Classes

Prefixed derived unit

Description

A *prefixed derived unit* is an *SI unit* with an *SI prefix*. Examples are kJ (kilo-joule), hPa (hecto-pascal), or mm (milli-meter).

Definition

A *prefixed derived unit* defines a *node*, the left child *node* of which has an *SI prefix* as a node value.

Relations

- A *prefixed derived unit* is a specialization of an *SI derived unit*.

SI base unit

Description

The seven *base units* of the SI system are: ampere, candela, kelvin, kilogram, meter, mole, and second (BIPM, 2006).

Definition

An *SI base unit* is one of the following individuals: A, cd, K, kg, m, mol, s.

Relations

- *SI prefix* is a subclass of *SI unit*.
- *SI prefix* is a subclass of *leaf value*.

SI derived unit

Description

"Derived units are units which may be expressed in terms of base units by means of the mathematical symbols of multiplication and division. Certain derived units have been given special names and symbols, and these special names and symbols may themselves be used in combination with those for base and other derived units to express the units of other quantities" (BIPM, 2006).

Relations

- *SI derived unit* is a subclass of an *SI unit*.
- An *SI derived unit* can only be defined by a *node*.
- An *SI derived unit* cannot be defined by more than a single *node*.

SI prefix

Description

An *SI prefix* can be used to prefix any *SI unit* to produce a multiple or submultiple of the original unit (BIPM, 2006). So far (as of 2006), the following 20 prefixes have been approved by the General Conference on Weights and Measures: yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

Definition

An *SI prefix* is one of the following individuals: yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

Relations

- An *SI prefix* is a subclass of *leaf value*.

SI unit

Description

An *SI unit* is *unit of measure* that complies with the SI system of units (cf. BIPM, 2006).

Definition

An *SI unit* is either an *SI base unit* or an *SI derived unit*.

Relations

- *SI unit* is a subclass of *unit*.

Relations

isDefinedBy

Description

The relation defines links a *note* to an *SI derived unit*, which represents the right hand side of a definition equation for the *SI derived unit*.

Characteristics

- Specialization of the meta relation feature-objectRelation.
- Domain: *SI derived unit*
- Range: *node*

Individuals

A

Description

The official definition of the ampere (symbol: A), given in Sec. 2.1.1.4 of (BIPM, 2006), is as follows:

"The ampere is that constant current which, if maintained in two straight parallel conductors of infinite length, of negligible circular cross-section, and placed 1 metre apart in vacuum, would produce between these conductors a force equal to 2 x 10–7 newton per metre of length."

Characteristics

- A is an instance of *SI base unit*.
- A hasDimension current_strength.
- Different from cd, K, kg, m, mol, s.

atto

Description

atto (symbol: a) is an SI prefix that stands for the factor 1e–18.

Characteristics

- atto is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, zepto, yocto.

cd

Description

The official definition of the candela (symbol: cd), given in Sec. 2.1.1.7 of (BIPM, 2006), is as follows:

"The candela is the luminous intensity, in a given direction, of a source that emits monochromatic radiation of frequency 540 x 1012 hertz and that has a radiant intensity in that direction of 1/683 watt per steradian."

Characteristics

- cd is an instance of *SI base unit*.
- cd hasDimension luminous_intensity.
- Different from A, K, kg, m, mol, s.

centi

Description

centi (symbol: c) is an SI prefix that stands for the factor 1e-2.

Characteristics

- centi is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, milli, micro, nano, pico, femto, atto, zepto, yocto.

deca

Description

deca (symbol: da) is an SI prefix that stands for the factor 10e1.

Characteristics

- deca is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

deci

Description

deci (symbol: d) is an SI prefix that stands for the factor 1e-1.

Characteristics

- deci is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

exa

Description

exa (symbol: E) is an SI prefix that stands for the factor 1e18.

Characteristics

- exa is an instance of *SI prefix*.
- Different from yotta, zetta, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

femto

Description

femto (symbol: f) is an SI prefix that stands for the factor 1e–15.

Characteristics

- femto is an instance of *SI prefix*.

- Different from yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, atto, zepto, yocto.

giga

Description

giga (symbol: g) is an SI prefix that stands for the factor 1e9.

Characteristics

- giga is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

hecto

Description

hecto (symbol: h) is an SI prefix that stands for the factor 1e2.

Characteristics

- hecto is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, mega, kilo, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

K

Description

The official definition of the Kelvin (symbol: K), given in Sec. 2.1.1.5 of (BIPM, 2006), is as follows:

"The kelvin, unit of thermodynamic temperature, is the fraction 1/273.16 of the thermodynamic temperature of the triple point of water."

Characteristics

- K is an instance of *SI base unit*.
- K hasDimension thermodynamic_temperature.
- Different from A, cd, kg, m, mol, s.

kg

Description

The official definition of the kilogram (symbol: kg), given in Sec. 2.1.1.2 of (BIPM, 2006), is as follows:

"The kilogram is the unit of mass; it is equal to the mass of the international prototype of the kilogram."

Characteristics

- kg is an instance of *SI base unit*.
- kg hasDimension mass.
- Different from A, cd, K, m, mol, s.

kilo

Description

kilo (symbol: k) is an SI prefix that stands for the factor 1e3.

Characteristics

- kilo is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, mega, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

m

Description

The official definition of the meter (symbol: m), given in Sec. 2.1.1.1 of (BIPM, 2006), is as follows:

"The metre is the length of the path travelled by light in vacuum during a time interval of 1/299,792,458 of a second."

Characteristics

- m is an instance of *SI base unit*.
- m hasDimension length.
- Different from A, cd, K, kg, mol, s.

mega

Description

mega (symbol: M) is an SI prefix that stands for the factor 1e6.

Characteristics

- mega is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

micro

Description

micro (symbol: μ) is an SI prefix that stands for the factor 1e-6.

Characteristics

- micro is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, nano, pico, femto, atto, zepto, yocto.

milli

Description

milli (symbol: m) is an SI prefix that stands for the factor 1e-3.

Characteristics

- milli is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, micro, nano, pico, femto, atto, zepto, yocto.

mol

Description

The official definition of the mole (symbol: mol), given in Sec. 2.1.1.6 of (BIPM, 2006), is as follows:

“1. The mole is the amount of substance of a system which contains as many elementary entities as there are atoms in 0.012 kilogram of carbon 12 […].

2. When the mole is used, the elementary entities must be specified and may be atoms, molecules, ions, electrons, other particles, or specified groups of such particles.”

Characteristics

- mol is an instance of *SI base unit*.
- mol hasDimension amount_of_substance.

nano

Description

nano (symbol: n) is an SI prefix that stands for the factor 1e-9.

Characteristics

- nano is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, pico, femto, atto, zepto, yocto.

peta

Description

peta (symbol: P) is an SI prefix that stands for the factor 1e15.

Characteristics

- peta is an instance of *SI prefix*.
- Different from yotta, zetta, exa, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

pico

Description

pico (symbol: p) is an SI prefix that stands for the factor 1e–12.

Characteristics

- pico is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, femto, atto, zepto, yocto.

s

Description

The official definition of the second (symbol: s), given in Sec. 2.1.1.3 of (BIPM, 2006), is as follows:

“The second is the duration of 9,192,631,770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the caesium 133 atom.”

Characteristics

- s is an instance of *SI base unit*.
- s hasDimension time.
- Different from A, cd, K, kg, m, mol.

tera

Description

tera (symbol: T) is an SI prefix that stands for the factor 1e12.

Characteristics

- tera is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

yocto

Description

yocto (symbol: y) is an SI prefix that stands for the factor 1e–24.

Characteristics

- yocto is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto.

yotta

Description

yotta (symbol: Y) is an SI prefix that stands for the factor 1e24.

Characteristics

- yotta is an instance of *SI prefix*.
- Different from zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

zepto

Description

zepto (symbol: z) is an SI prefix that stands for the factor 1e-21.

Characteristics

- zepto is an instance of *SI prefix*.
- Different from yotta, zetta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, yocto.

zetta

Description

zetta (symbol: Z) is an SI prefix that stands for the factor 1e21.

Characteristics

- zetta is an instance of *SI prefix*.
- Different from yotta, exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto, zepto, yocto.

METER

Description

Auxiliary individual for the definition of *derived SI units* that are derivable from the *base unit* of m.

Characteristics

- Instance of *leaf*
- hasNodeValue: m
- Different from AMPERE, CANDELA, KELVIN, KILOGRAM, MOLE, SECOND, YOTTA-, ZETTA-, EXA-, PETA-, TERA-, GIGA-, MEGA-, KILO-, HECTO-, DECA-, DECI-, CENTI-, MILLI-, MICRO-, NANO-, PICO-, FEMTO-, ATTO-, ZEPTO-, YOCTO-.

ZETTA-

Description

Auxiliary individual for the definition of *prefixed derived units*, the definition of which includes the *SI prefix* of zetta.

Characteristics

- Instance of *leaf*
- hasNodeValue: zetta
- Different from AMPERE, CANDELA, KELVIN, KILOGRAM, METER, MOLE, SECOND, YOTTA-, ZETTA-, EXA-, PETA-, TERA-, GIGA-, MEGA-, KILO-, HECTO-, DECA-, DECI-, CENTI-, MILLI-, MICRO-, NANO-, PICO-, FEMTO-, ATTO-, ZEPTO-, YOCTO-.

4.2 Derived SI Units

The ontology module *derived_SI_units* establishes a number of frequently used *derived SI units* and provides the corresponding definition trees. For details, we refer to the formal specification of OntoCAPE.

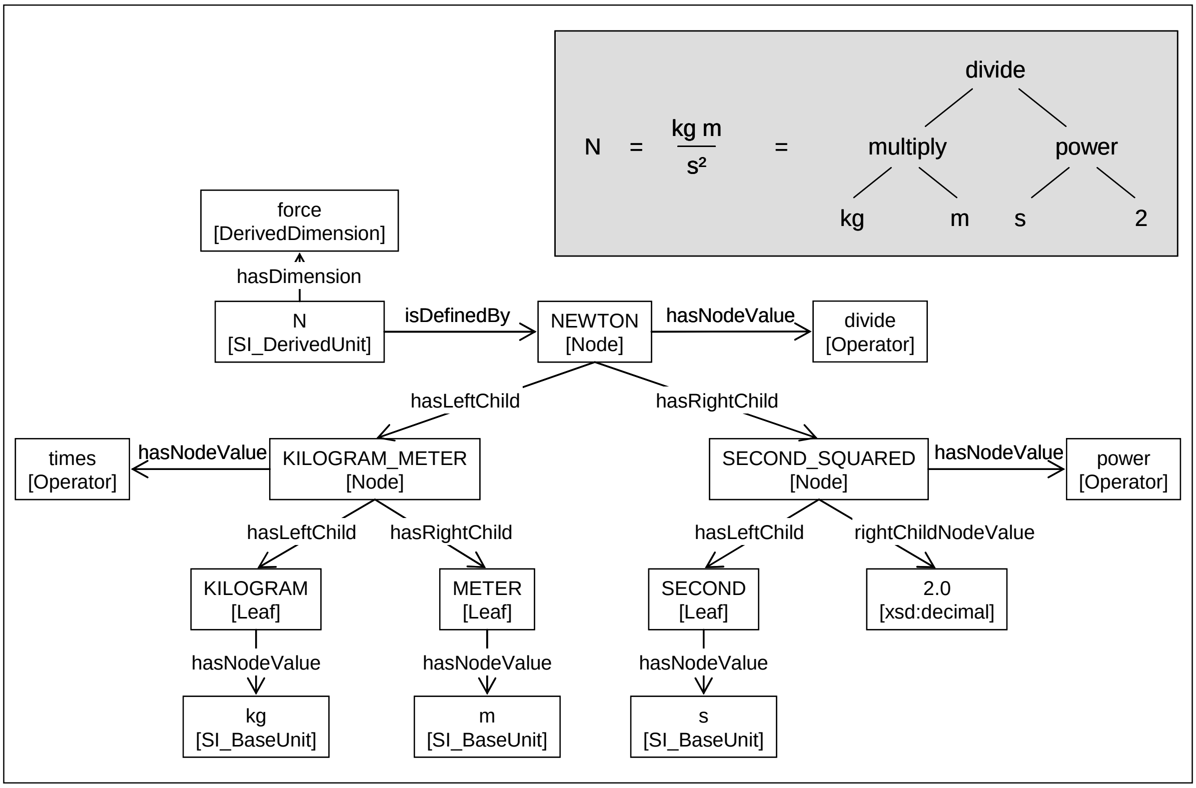

Fig. 16: Definition of the *derived SI unit* 'N'

5 Space and Time

The ontology module *space_and_time* introduces spatial and temporal coordinate systems and provides concepts for the representation of spatial and temporal points as well as periods of time..

The concept of a *spatial coordinate system* is introduced as a special type of *coordinate system* (cf. Fig. 17). A *spatial coordinate system* has *spatial coordinate system axes*, which may be either *Cartesian* or *curvilinear coordinate system axes*. The ontology module provides some predefined axes, like the x-, y-, and z-axis of a Cartesian coordinate system. Moreover, a *spatial coordinate system* has up to three *spatial coordinates* (depending on whether the system is intended for 1D, 2D, or 3D space); these are either *straight coordinates* (representing a distance) or *angular coordinates* (representing an angle). A *spatial point* is represented through up to three *spatial coordinates*, again depending on the dimensionality of the considered space.

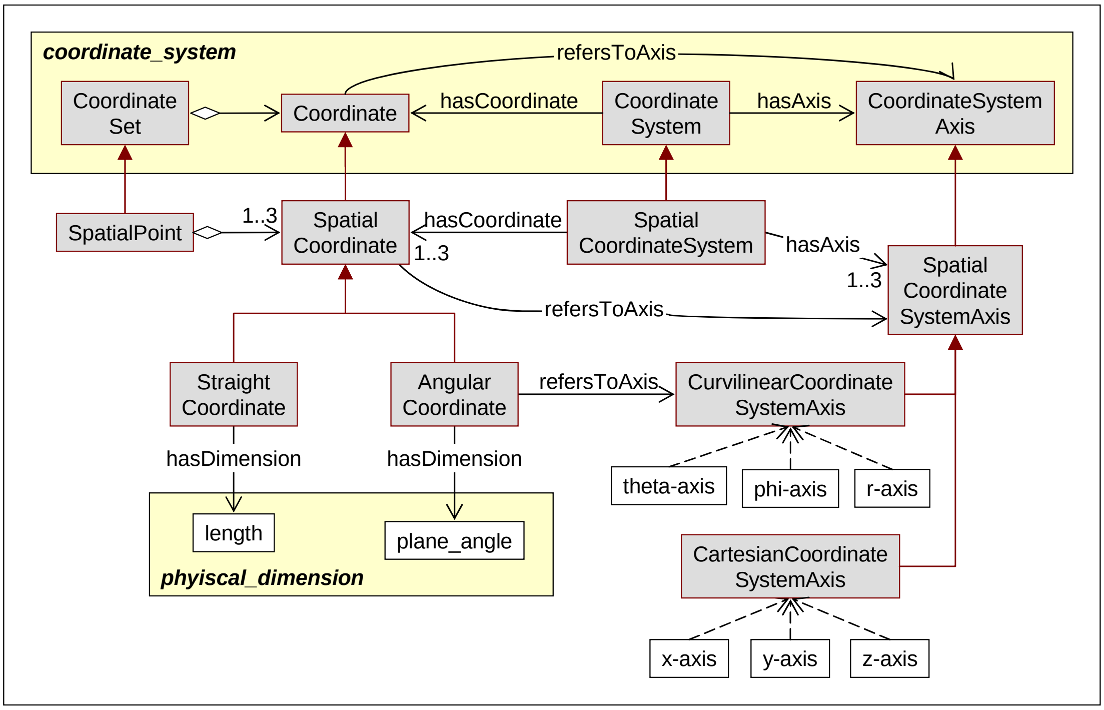

Fig. 17: Spatial coordinates

Some special types of *spatial coordinate systems* are introduced in Fig. 18; each type is assigned its respective *coordinate system axes*. The *spatial coordinate systems* are classified from two perspectives: The first perspective differentiates coordinate systems for 2D and 3D space, while the second perspective distinguishes curvilinear and Cartesian coordinate systems.

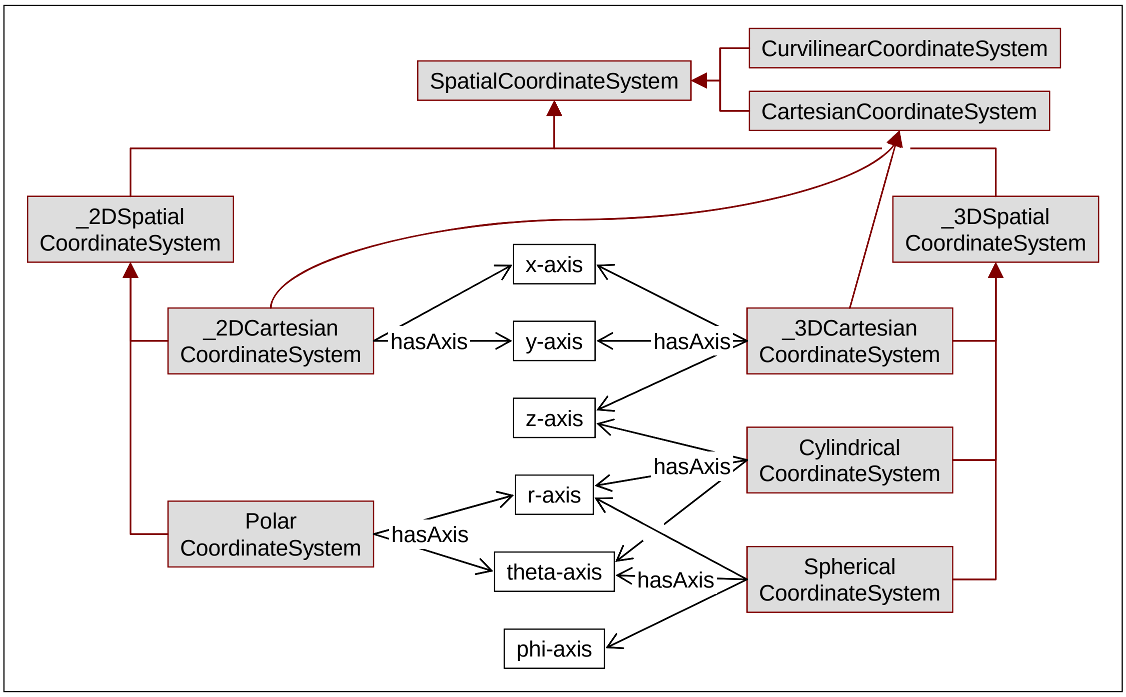

Fig. 18: Types of spatial coordinate systems

An application example for a *polar coordinate system* is given in Fig. 19.

The *spatial coordinate systems* are explicitly classified along the first dimension, i.e., they are categorized as either *2D* or *3D spatial coordinate systems*[4] as shown in Fig. 19.

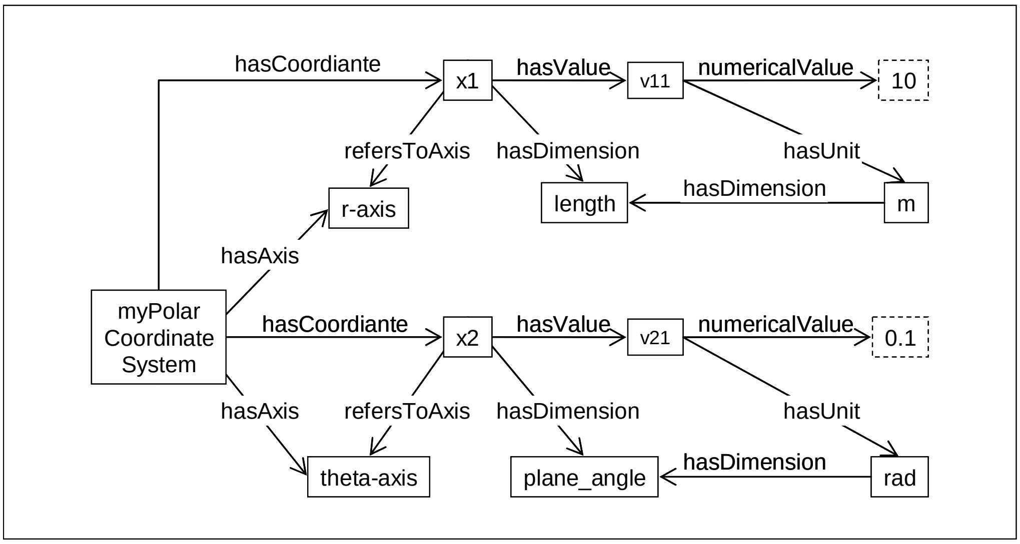

Fig. 19: Application example for a *polar coordinate system*

[4] Implementation advise: According to the principle of ontology normalization (cf. Meta Model), the affiliation to the second dimension should be indirectly defined via necessary and sufficient conditions. This principle is followed in the case of *curvilinear coordinate systems* (i.e., the class *curvilinear coordinate system* is defined through necessary and sufficient conditions, such that the subclasses of *curvilinear coordinate system* can be automatically inferred by a reasoner). However, it is not appropriate to define the class *Cartesian coordinate system* through necessary and sufficient conditions, since such a definition would severely deteriorate the reasoner performance. Thus, deviating from the principle of ontology normalisation, multiple classification is applied here.

As shown in Fig. 20, the class *temporal coordinate system* is defined analogously to *spatial coordinate system*: A *temporal coordinate system* has one *temporal coordinate*, which refers to a *temporal coordinate system axis*. The t-axis is defined as the default axis of a *temporal coordinate system*.

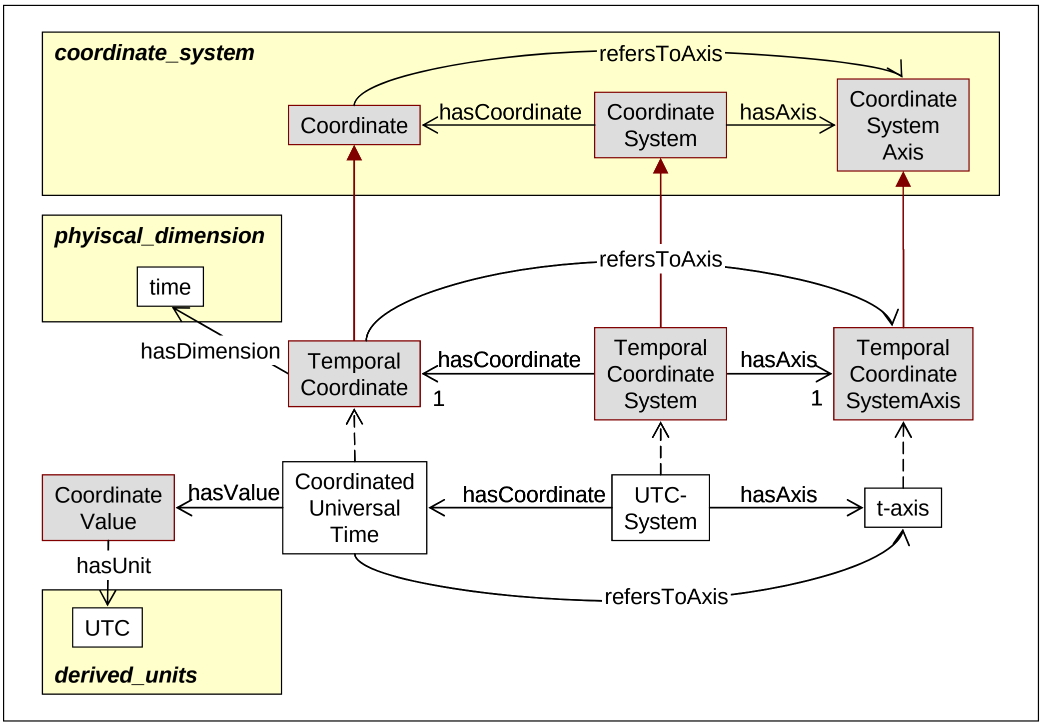

Fig. 20: Temporal coordinates

An important *temporal coordinate system* is the UTC-System. UTC stands for Coordinated Universal Time and denotes an international time standard, which is disseminated by the International Bureau of Weights and Measures (BIPM, 2007). The *temporal coordinate* of the UTC-System is named CoordinatedUniversalTime. Its *coordinate value* has the *unit* UTC, and the numericalValue attribute of the *coordinate value* should be specified in the format of the XML datatype *dateTime* (Biron and Malhotra, 2004).

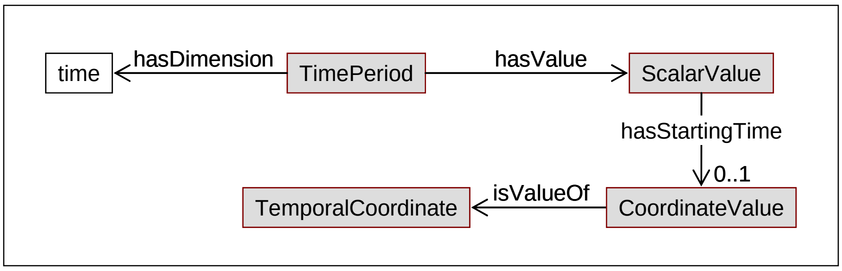

Fig. 21: Representation of a time period with a definite starting time

A *time period* is a *scalar quantity* whose *scalar value* denotes the temporal duration of a period of time. Optionally, the starting time of the *time period* can be indicated; to this end, the *scalar value* of the *time period* refers to the *value* of a *temporal coordinate* via the relation hasStartingTime (cf. Fig. 21). An application example is shown in Fig. 22.

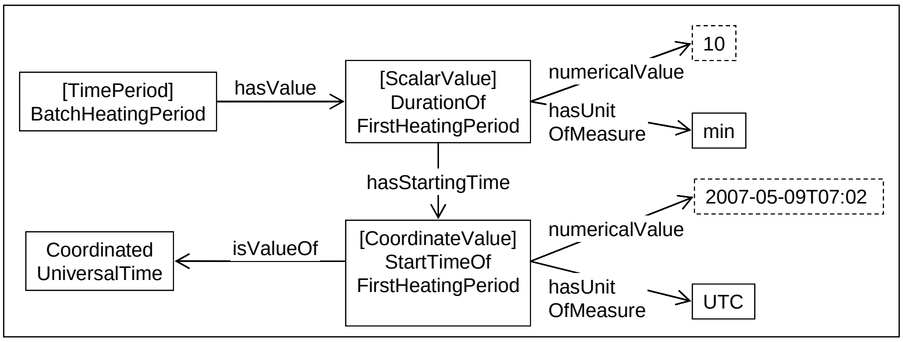

Fig. 22: Application of the concepts *time period* and *temporal coordinate*

Finally, the concept of a *spatio-temporal coordinate system* is introduced, which denotes a *coordinate system* that has both *spatial* and *temporal coordinates* (Fig. 23).

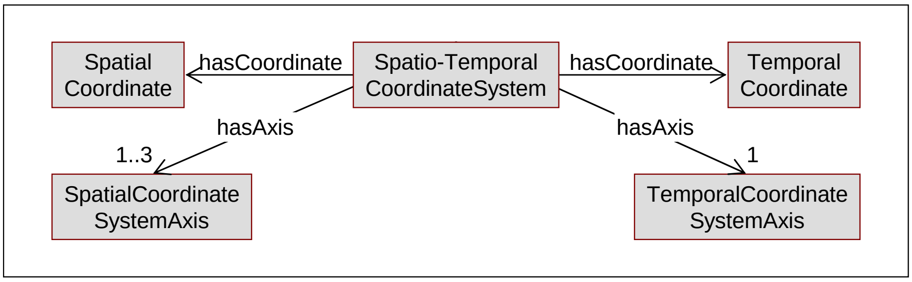

Fig. 23: Spatio-temporal coordinate system

Concept Descriptions

Individual concepts of the module *space_and_time* are defined below.

Classes

2D Cartesian coordinate system

Description

A *2D Cartesian coordinate system* is an orthogonal *planar coordinate system* that has two straight, perpendicular axes: the x-axis (a.k.a. abscissa) and the y-axis (a.k.a ordinate). A *2D Cartesian coordinate system* has a positive orientation (i.e., the x-axis points right and the y-axis points up).

Relations

- *2D Cartesian coordinate system* is a subclass of *planar coordinate system*.
- *2D Cartesian coordinate system* is a subclass of *Cartesian coordinate system*.
- A *2D Cartesian coordinate system* has some *Cartesian coordinate system axes*.
- A *2D Cartesian coordinate system* can only have *Cartesian coordinate system axes*.
- A *2D Cartesian coordinate system* has an x-axis.
- A *2D Cartesian coordinate system* has a y-axis.

3D Cartesian coordinate system

Description

A *3D Cartesian coordinate system* is an orthogonal *3D spatial coordinate system* that has three straight, perpendicular axes: the x-axis, the y-axis, and the z-axis. A *3D Cartesian coordinate system* has a positive (right-handed) orientation; that is, the *xy*-plane is horizontal, the z-axis points up, and the x-axis and the y-axis form a positively oriented *2D Cartesian coordinate system* in the *xy*-plane if observed from above the *xy*-plane.

Relations

- *3D Cartesian coordinate system* is a subclass of *3D spatial coordinate system*.
- *3D Cartesian coordinate system* is a subclass of *Cartesian coordinate system*.
- A *3D Cartesian coordinate system* has some *Cartesian coordinate system axes*.
- A *3D Cartesian coordinate system* can only have *Cartesian coordinate system axes*.
- A *3D Cartesian coordinate system* has an x-axis.
- A *3D Cartesian coordinate system* has a y-axis.
- A *3D Cartesian coordinate system* has a z-axis.

3D spatial coordinate system

Description

A *3D spatial coordinate system* is a *spatial coordinate system* for describing positions in 3D space.

Relations

- *3D spatial coordinate system* is a subclass of *spatial coordinate system*.
- A *3D spatial coordinate system* has three *spatial coordinate system axes*.
- A *3D spatial coordinate system* has three *spatial coordinates*.

Angular coordinate

Description

An *angular coordinate* is an angle that acts as a *spatial coordinate*.

Definition

An *angular coordinate* is a *spatial coordinate* that has the *physical dimension* of plane_angle.

Relations

- *Angular coordinate* is a subclass of *spatial coordinate*.
- An *angular coordinate* has the dimension plane_angle.
- An *angular coordinate* can only refer to an *curvilinear coordinate system axis*.

Cartesian coordinate system

Description

A *Cartesian coordinate system* is a *spatial coordinate system*, the coordinate surfaces of which are planes (in 3D) or straight lines (in 2D).

Definition

A *Cartesian coordinate system* is a *spatial coordinate system* that has (1) some *Cartesian coordinate system axes* and (2) only *Cartesian coordinate system axes*.

Relations

- *Cartesian coordinate system* is a subclass of *spatial coordinate system*.
- A *Cartesian coordinate system* has some *Cartesian coordinate system axes*.
- A *Cartesian coordinate system* can only have *Cartesian coordinate system axes*.

- A *Cartesian coordinate system* has some *straight coordinates*.
- A *Cartesian coordinate system* has only *straight coordinates*.

Cartesian coordinate system axis

Description

A *Cartesian coordinate system axis* is an axis of a *Cartesian coordinate system*.

Relations

- *Cartesian coordinate system axis* is a subclass of *spatial coordinate system axis*.

Curvilinear coordinate system

Description

A *curvilinear coordinate system* is a *spatial coordinate system* the coordinate surfaces of which are curved surfaces (in 3D) or curved lines (in 2D).

Definition

A *curvilinear coordinate system* is a *spatial coordinate system* that has some *curvilinear coordinate system axes*.

Relations

- *Curvilinear coordinate system* is a subclass of *spatial coordinate system*.
- A *curvilinear coordinate system* has some *curvilinear coordinate system axes*.
- A *curvilinear coordinate system* has some *angular coordinate*.
- A *curvilinear coordinate system* has some *straight coordinate*.

Curvilinear coordinate system axis

Description

A *curvilinear coordinate system axis* is an axis of a *curvilinear coordinate system*.

Relations

- *Curvilinear coordinate system axis* is a subclass of *spatial coordinate system axis*.

Cylindrical coordinate system

Description

A *cylindrical coordinate system* is an orthogonal *3D spatial coordinate system* that has cylindrical coordinates (i.e., *radius*, *height*, and *azimuth angle*). It is especially suited to describe positions on rotationally symmetrical shapes like cylinders or cones.

Relations

- *Cylindrical coordinate system* is a subclass of *3D spatial coordinate system*.
- A *cylindrical coordinate system* has some *curvilinear coordinate system axis*.
- A *cylindrical coordinate system* has some *Cartesian coordinate system axis*.
- A *cylindrical coordinate system* has an r-axis.
- A *cylindrical coordinate system* has a theta-axis.
- A *cylindrical coordinate system* has a z-axis.

Planar coordinate system

Description

A *planar coordinate system* is a *spatial coordinate system* for describing positions located on a two-dimensional plane.

Relations

- *Planar coordinate system* is a subclass of *spatial coordinate system*.
- A *planar coordinate system* has two *spatial coordinate system axes*.
- A *planar coordinate system* has two *spatial coordinates*.

Polar coordinate system

Description

A *polar coordinate system* is a *planar coordinate system* that has polar coordinates (i.e., *radius* and *polar angle*). It is especially suited for describing positions on a circle or ellipse.

Relations

- *Polar coordinate system* is a subclass of *coordinate system*.
- A *polar coordinate system* has some *curvilinear coordinate system axes*.
- A *polar coordinate system* can only have *curvilinear coordinate system axes*.
- A *polar coordinate system* has an r-axis.
- A *polar coordinate system* has a theta-axis.

Spatial coordinate

Description

A *spatial coordinate* is a *coordinate* that denotes a spatial position.

Definition

A *spatial coordinate* is either a *straight coordinate* or an *angular coordinate*.

Relations

- *Spatial coordinate* is a subclass of *coordinate*.
- A *spatial coordinate* refers to one *spatial coordinate system axis*.

Spatial coordinate system

Description

A *spatial coordinate system* is a *coordinate system* for describing spatial positions.

Relations

- *Spatial coordinate system* is a subclass of *coordinate system*.
- A *spatial coordinate system* has some *spatial coordinates*.
- A *spatial coordinate system* has only *spatial coordinates*.
- A *spatial coordinate system* has up to three *spatial coordinates*.
- A *spatial coordinate system* has some *spatial coordinate system axes*.
- A *spatial coordinate system* has only *spatial coordinate system axes*.
- A *spatial coordinate system* has up to three *spatial coordinate system axes*.

Spatial coordinate system axis

Description

A *spatial coordinate system axis* is the *coordinate system axis* of some *spatial coordinate system*.

Relations

- *Spatial coordinate system axis* is a subclass of *coordinate system axis*.

Spatio-temporal coordinate system

Description

A *spatio-temporal coordinate system* denotes positions in space and time.

Relations

- *Spatio-temporal coordinate system* is a subclass of *coordinate system*.
- A *spatio-temporal coordinate system* has some *spatial coordinates*.
- A *spatio-temporal coordinate system* has some *temporal coordinates*.
- A *spatio-temporal coordinate system* has only *spatial* and *temporal coordinates*.
- A *spatio-temporal coordinate system* has up to four *coordinates*.
- A *spatio-temporal coordinate system* has some *spatial coordinate system axes*.
- A *spatio-temporal coordinate system* has some *temporal coordinate system axis*.
- A *spatio-temporal coordinate system* has only *spatial* and *temporal coordinate system axes*.
- A *spatio-temporal coordinate system* has up to four *coordinate system axes*.

Spatial point

Description

A *spatial point* is a point in space; it is represented through a *coordinate set* comprising up to 3 *spatial coordinates*.

Relations

- *Spatial point* is a subclass of *coordinate set*.
- A *spatial point* comprises some *spatial coordinates*.
- A *spatial point* comprises only *spatial coordinates*.
- A *spatial point* comprises up to 3 *spatial coordinates*.

Spherical coordinate system

Description

A *spherical coordinate system* is an orthogonal *3D spatial coordinate system* that has spherical coordinates (i.e., *radius*, *azimuth angle*, and *zenith angle*). It is especially suited for describing positions on a sphere or spheroid.

Relations

- *Spherical coordinate system* is a subclass of *3D spatial coordinate system*.
- A *spherical coordinate system* has an r-axis.
- A *spherical coordinate system* has a theta-axis.
- A *spherical coordinate system* has a phi-axis.

Straight coordinate

Description

A *straight coordinate* is a distance that acts as a *spatial coordinate*.

Definition

A *straight coordinate* is a *spatial coordinate* that has the *physical dimension* of length.

Relations

- *Straight coordinate* is a subclass of *spatial coordinate*.
- A *straight coordinate* has the dimension length.

Temporal coordinate

Description

A *temporal coordinate* is a *coordinate* that denotes a temporal position.

Relations

- *Temporal coordinate* is a subclass of *coordinate*.
- A *temporal coordinate* refers to one *temporal coordinate system axis*.
- A *temporal coordinate* has the dimension time.

Temporal coordinate system

Description

A *temporal coordinate system* is a *coordinate system* for describing temporal positions.

Relations

- *Temporal coordinate system* is a subclass of *coordinate system*.
- A *temporal coordinate system* has some *temporal coordinates*.
- A *temporal coordinate system* has only *temporal coordinates*.
- A *temporal coordinate system* has exactly one *temporal coordinates*.
- A *temporal coordinate system* has some *temporal coordinate system axis*.
- A *temporal coordinate system* has only *temporal coordinate system axes*.
- A *temporal coordinate system* has exactly one *temporal coordinate system axes*.

Temporal coordinate system axis

Description

A *temporal coordinate system axis* is the *coordinate system axis* of some *temporal coordinate system*.

Relations

- *Temporal coordinate system axis* is a subclass of *coordinate system axis*.

Time period

Description

A *time period* is a *scalar quantity* that denotes the temporal duration of a period of time. Optionally, the starting time of the *time period* can be indicated.

Relations

- *Time period* is a subclass of *scalar quantity*.
- A *time period* has the *physical dimension* of time.
- The *scalar value* of a *time period* may refer to the *coordinate value* of a *temporal coordinate* via the hasStartingTime relation to indicate the starting time of the *time period*.

Relations

hasStartingTime

Description

Indicates the starting time of a *time period*.

Characteristics

- Specialization of isObservedAgainstBackdrop
- Domain: *scalar value*
- Range: *coordinate value* of some *temporal coordinate*
- Functional

Individuals

CoordinatedUniversalTime

Description

A CoordinatedUniversalTime is a *temporal coordinate* of an UTC-System; it measures the date-time according to the international time standard UTC.

Relations

- CoordinatedUniversalTime is an instance of *temporal coordinate*.
- CoordinatedUniversalTime is a property of the UTC-System.
- CoordinatedUniversalTime refers to the t-axis.

phi-axis

Description

The individual denotes the *phi*-axis of a spherical coordinate system. *Phi* is the *zenith angle* between the z-axis and the r-axis. Its value range lies between 0 and π .

Characteristics

- Instance of *curvilinear coordinate system axis*
- Different from theta-axis

r-axis

Description

The r-axis corresponds to the *radial coordinate*, which denotes the distance (i.e., radius) between a point and the origin of a *spatial coordinate system*.

Characteristics

- Instance of *curvilinear coordinate system axis*
- Different from x-axis, y-axis, and z-axis.

t-axis

Description

The default axis of a *temporal coordinate system*.

Characteristics

- Instance of *temporal coordinate system axis*

theta-axis

Description

The individual denotes the *theta*-axis of a polar coordinate system, spherical coordinate system, or cylindrical coordinate system. *Theta* is the *azimuth* or *polar angle* located in the *xy*-plane of a positive *Cartesian coordinate system*; it is defined as the angle between the polar axis (which is equivalent to the x-axis of a *Cartesian coordinate system*) and the projection of the r-axis onto the *xy*-plane. The value range of *theta* lies between 0 and 2π .

Characteristics

- Instance of *curvilinear coordinate system axis*
- Different from phi-axis

UTC-System

Description

The UTC-System is a *temporal coordinate system* that measures the date-time according to the international time standard UTC (Coordinated Universal Time), disseminated by the International Bureau of Weights and Measures (BIPM, 2007).

Relations

- UTC-System is an instance of *temporal coordinate system*.
- UTC-System has CoordinatedUniversalTime as a *temporal coordinate*.
- UTC-System has a t-axis.

x-axis

Description

The individual denotes the *x*-axis of a positive *Cartesian coordinate system*.

Characteristics

- Instance of *Cartesian coordinate system axis*
- Different from y-axis, z-axis, and r-axis

y-axis

Description

The individual denotes the *y*-axis of a positive *Cartesian coordinate system*.

Characteristics

- Instance of *Cartesian coordinate system axis*
- Different from x-axis, z-axis, and r-axis

z-axis

Description

The individual denotes the *z*-axis of a positive *Cartesian coordinate system* or *cylindrical coordinate system*.

Characteristics

- Instance of *Cartesian coordinate system axis*
- Different from x-axis, y-axis, and r-axis

6 Geometry

The module **geometry** provides the concepts for describing the shapes and main dimensions of simple geometric figures. Two major classes of figures are introduced, which are both defined as subclasses of *system*: *solids* and *surfaces*.

- A *solid* (a.k.a. geometric solid or solid geometric figure) is a bounded three-dimensional geometric figure in Euclidean space.
- A *surface* is a bounded geometric figure in a two-dimensional submanifold of three-dimensional Euclidean space.

Solids and *surfaces* may have certain geometric *properties*. Some of them are *scalar quantities*, such as, *diameter*, *radius*, or *volume* (the latter is only defined for *solids*, cf. Fig. 24); others, such as *edge length*, *height*, and *surface area*, may alternatively be a *scalar quantity* or a *vector quantity*.

- As a *scalar quantity*, these quantities simply indicate a size – *surface area*, for instance, indicates either the area of a *surface* or of (one of) the exterior surface(s) of a *solid*.
- As a *vector quantity*, the quantities additionally indicate the orientation of the respective line or surface – in case of the *surface area*, the vector would have the same orientation as the surface normal, while the Euclidean norm of the vector would equal the area of the surface.

There are two further specializations of the *surface area* concept:

- *Side area*, which is only defined for *solids*, corresponds to one particular exterior surface of a *solid*. This concept should be applied only if the *solid* has several distinguishable exterior surfaces (as, e.g., in the case of a *cuboid*).
- The *total surface area* indicates the total area of either a *surface* or of (all) the exterior surface(s) of a *solid*. A *total surface area* is always a scalar.

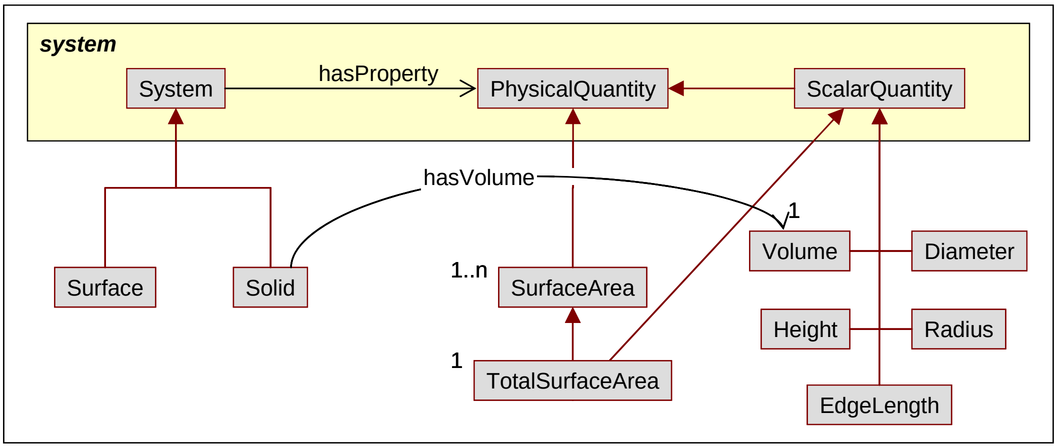

Fig. 24: Key concepts of *geometry*

The relations has_area, has_length, and has_volume are introduced as specializations of hasProperty (Fig. 25). However, these relations are merely auxiliary constructs used as replacements for qualified number restrictions. They will drop out again, as soon as qualified number restrictions are made available in OWL.

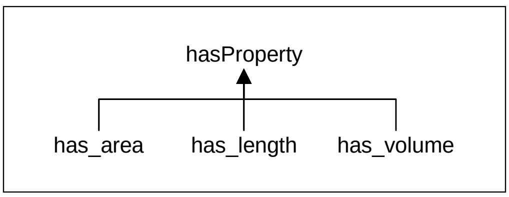

Fig. 25: Specializations of hasProperty as workarounds for missing QNR

In the following, some special types of *solids* and *surfaces* and their geometric properties will be introduced. Note that most of the terminology as well as the textual definitions for these concepts have been adopted from the interactive mathematics dictionaries *Mathwords* (Simmons, 2007) and *MathWorld* (Weisstein, 2007).

Disk and *rectangle* are two special kinds of *surfaces*, which have the following properties: a *disk* has exactly one length of type *diameter* or *radius*, whereas a rectangle has exactly two lengths of type *edge length* (Fig. 26).

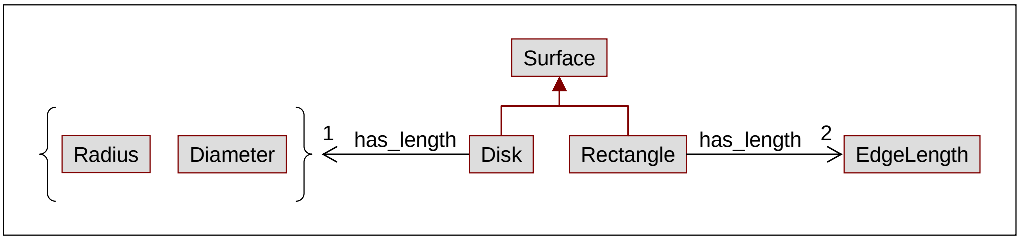

Fig. 26: Disk and Rectangle

Cuboid and *sphere* are special types of *solids*, for which the following properties are defined (cf. Fig. 27):

- A *sphere* has exactly one length of type *radius* or *diameter* as well as one area of type *total surface area*.
- A *cuboid* has exactly three lengths of type *edge length* as well as three areas of type *side area*.

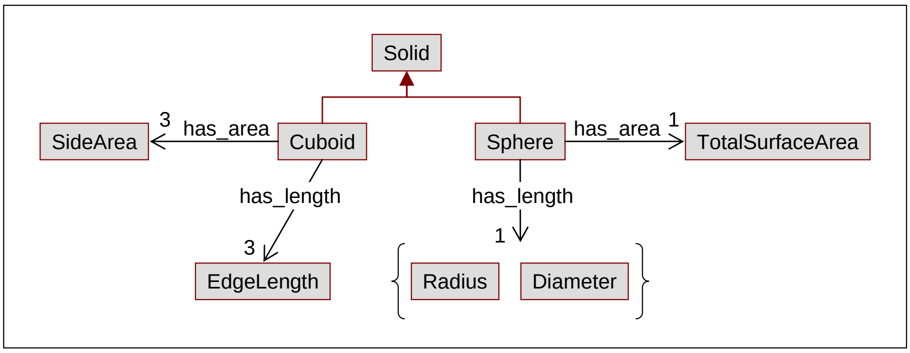

Fig. 27: *Cuboid* and *sphere*

A *cylinder* is a *solid* with parallel congruent bases. The bases can be shaped like any closed plane figure (not necessarily a circle) and must be oriented identically. In order to differentiate the bases from the other exterior surfaces of the *cylinder*, two specializations of *side area* are introduced, namely *base area* and *lateral surface area* (Fig. 28). Then it can be stated that a *cylinder* has exactly two *base areas* and at least one *lateral surface area*. Moreover, a *right circular cylinder*, which is a *cylinder* with circular bases that are aligned one directly above the other, has exactly one *lateral surface area*.

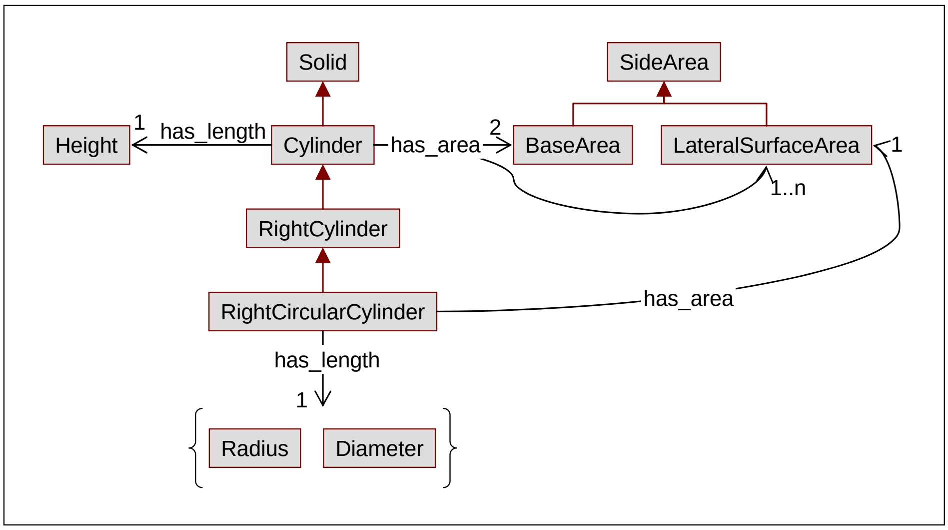

Fig. 28: Types of *cylinders*

The last type of *solid* introduced here is the *cone*, which is a figure with a single base tapering to an apex. If the apex is aligned directly above the center of the base, the cone can be classified as a *right cone*. Furthermore, if the base of a *right cone* takes the form of a circle, it can be classified as a *right circular cone*, which has exactly one *lateral surface area* (Fig. 29).

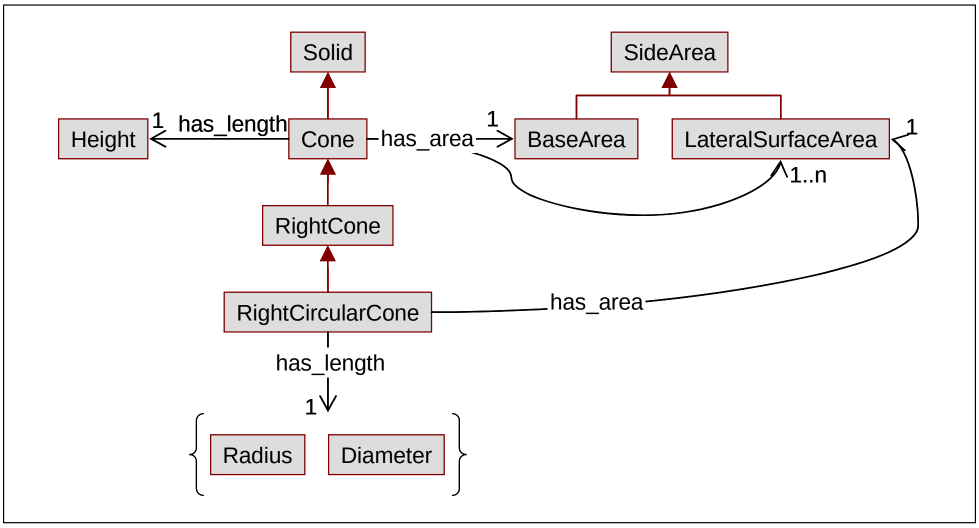

Fig. 29: Types of *cones*

Usage

There are two alternative ways to specify the geometry of a *system*. The first alternative is to summarize all the geometric aspects of the *system* in a separate *aspect system*. To this end, the *geometry* module provides the relations hasShapeRepresentation and hasSurfaceGeometry, which are specializations of the relation hasAspectSystem. These relations may be used to link a *solid* or a *surface* to a *system* (Fig. 30); a reasoner will then infer that the respective *solid* or *surface* is an *aspect system* of the main *system*.

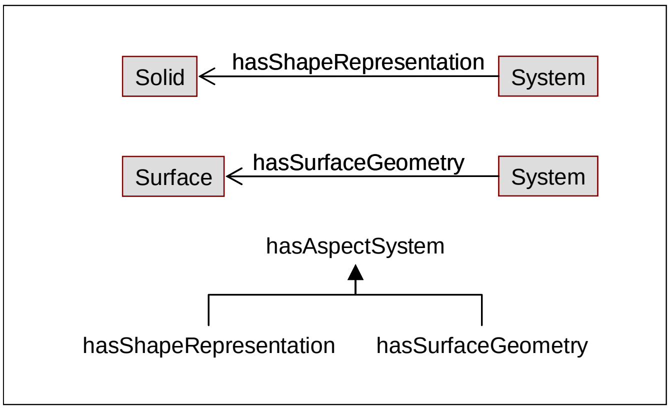

Fig. 30: Representation alternative 1: The geometric properties are summarized in an *aspect system*

An example is shown in Fig. 31: The shape of the distillation column Column C1 is represented by the individual Shape of C1. As indicated by the brackets in Fig. 31, Column C1 is an instance of *system*, while Shape of C1 is an instance of *right circular cylinder*. The two individuals are linked via the relation hasShapeRepresentation; thus, it can be inferred that Shape of C1 is an *aspect system* of the main *system* Column C1.

Shape of C1 is further characterized through its properties and their values. Exemplarily shown is its *height* H_C1, which has a value of 10 m.

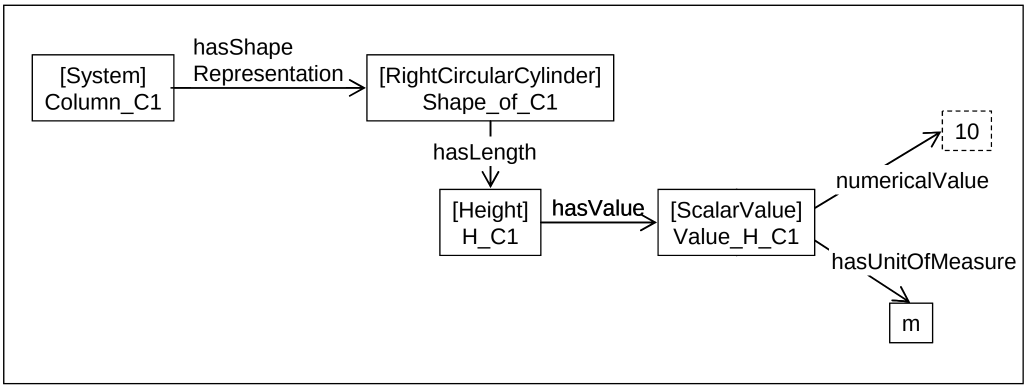

Fig. 31: Application example of representation alternative 1

Depending on the application, the above representation alternative may be too complicated for practical use. In the following, a more simple alternative is presented. This alternative can be applied if the shape of the respective *system* is obvious from the context or a matter of common knowledge – for instance, one may safely assume that a distillation column, unless otherwise indicated, has the shape of a *right circular cylinder*. In such a case, the description of the *system*'s geometry is simply a matter of specifying its main dimensions, which can be done by assigning the geometric *properties* directly to the *system*. This is exemplarily shown in Fig. 32, where the *height* H_C1 is directly assigned to the *system* Column C1.

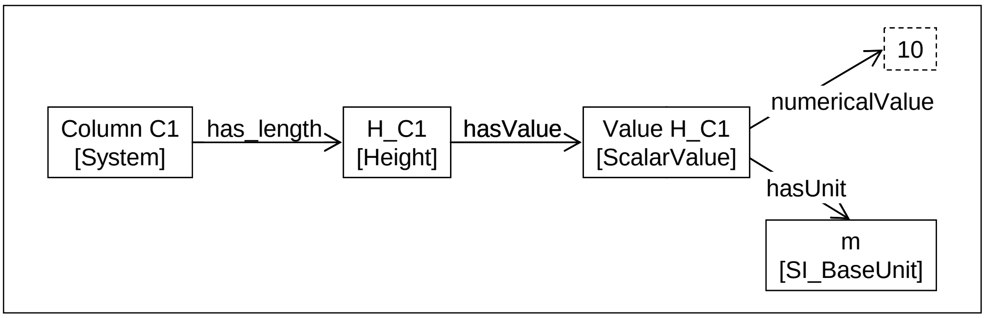

Fig. 32: Application example of representation alternative 2

As a further simplification, the geometric *properties* may be replaced by specializations of the hasCharacteristic relation (cf. [upper-level]). This approach is taken in module [chemical_process_system/plant].

With respect to information sharing, the users certainly favorable to agree on one convention (i.e. alternative1 or alternative 2) to avoid misconceptions.

Concept Descriptions

Individual concepts of the module *geometry* are defined below.

Classes

Base area

Description

The base is the bottom of a *solid*. If the top is parallel to the bottom (as in a trapezoid or prism), both the top and bottom are called bases. The *base area* is the *surface area* of (one of) the base(s).

Relations

- *Base area* is a subclass of *side area*.

Cone

Description

A three-dimensional figure with a single base tapering to an apex. The base can be any simple closed curve (Simmons, 2007).

Relations

- *Cone* is a subclass of *solid*.
- A *cone* has some length of type *height*.
- A *cone* has some *base area*.
- A *cone* has some *lateral surface area*.
- A *cone* can only have areas of type *base area* or *lateral surface area*.

Cuboid

Description

A closed box composed of three pairs of rectangular faces placed opposite each other and joined at right angles to each other, also known as a rectangular parallelepiped (Weisstein, 2007).

Relations

- *Cuboid* is a subclass of *solid*.
- A *cuboid* has exactly three lengths of type *edge length*.
- A *cuboid* has exactly three areas of type *side area*.

Cylinder

Description

A *solid* with parallel congruent bases. The bases can be shaped like any closed plane figure (not necessarily a circle) and must be oriented identically (Simmons, 2007).

Relations

- *Cylinder* is a subclass of *solid*.

- A *cylinder* has some length of type *height*.
- A *cylinder* has at least three *side areas*.
- A *cylinder* has some *base area*.
- A *cylinder* has some *lateral surface area*.
- A *cylinder* can only have areas of type *base area* or *lateral surface area*.

Diameter

Description

The length of a line segment between two points on a circle or *sphere* which passes through the center of the circle or *sphere*.

Relations

- *Diameter* is a subclass of *scalar quantity*.
- A *diameter* has the dimension of length.

Disk

Description

A *disk* is the union of a circle and its interior (Simmons, 2007). A circle is given by the set of points in a plane that are equidistant from a given point.

Relations

- *Disk* is a subclass of *surface*.
- A *disk* has exactly one length of type *diameter* or *radius*.

Edge length

Description

The length of a (straight) edge of a *surface* or *solid*.

Relations

- *Edge length* is a subclass of *scalar quantity*.
- *Edge length* has the dimension of length.

Height

Description

The shortest distance between the base of a geometric figure and its top, whether that top is an opposite vertex, an apex, or another base (Simmons, 2007).

Relations

- *Height* is a subclass of *scalar quantity*.
- A *height* has the dimension of length.

Lateral surface area

Description

The *surface area* of a single lateral surface of a *solid* (i.e., any *side area* that is not a *base area*).

Relations

- *Lateral surface area* is a subclass of *side area*.

Radius

Description

The length of the line segment between the center and a point on a circle or *sphere*.

Relations

- *Radius* is a subclass of *scalar quantity*.
- A *radius* has the dimension of length.

Rectangle

Description

A *rectangle* is a box shape on a plane. Formally, a *rectangle* is a quadrilateral with four congruent angles (all 90°) (Simmons, 2007).

Relationsr

- *Rectangle* is a subclass of *surface*.
- A *rectangle* has exactly two lengths of type *edge length*.

Right circular cone

Description

A *right cone* with a base that is a circle.

Relations

- *Right circular cone* is a subclass of *right cone*.
- A *right circular cone* has exactly two areas.
- A *right circular cone* has exactly one length of type *height*.

Right circular cylinder

Description

A *right cylinder* with bases that are circles (Simmons, 2007).

Relations

- *Right circular cylinder* is a subclass of *right cylinder*.
- A *right circular cylinder* has exactly three areas.
- A *right circular cylinder* has exactly two lengths.
- A *right circular cylinder* some length of type *radius* or *diameter*.
- A *right circular cylinder* can only have lengths of type *height* or *radius* or *diameter*.

Right cone

Description

A *cone* that has its apex aligned directly above the center of its base (Simmons, 2007).

Relations

- *Right cone* is a subclass of *cone*.

Right cylinder

Description

A *cylinder* which has bases aligned one directly above the other (Simmons, 2007).

Relations

- *Right cylinder* is a subclass of *cylinder*.

Side area

Description

The area of one particular exterior surface of a *solid*. This concept should be applied only if the *solid* has several distinguishable exterior surfaces. Otherwise (e.g., for a *sphere*) use *total surface area*. A *side area* can either be a *scalar quantity* or a *vector quantity*. In case of the latter, the vector is perpendicular to the surface (i.e., it has the same orientation as the surface normal), and its Euclidean norm equals the area of the surface.

Relations

- *Side area* is a subclass of *surface area*.
- A *side area* can only be a property of a *solid*.

Sphere

Description

A *solid* consisting of all points equidistant from a given point. This point is the center of the *sphere* (Weisstein, 2007).

Relations

- *Sphere* is a subclass of *solid*.
- A *sphere* has exactly one length of type *radius* or *diameter*.
- A *sphere* has exactly one area of type *total surface area*.

Solid

Description

A *solid* (a.k.a. geometric solid or solid geometric figure) is a collective term for all bounded three-dimensional geometric figures. This includes polyhedra, pyramids, prisms, cylinders, cones, spheres, ellipsoids, etc. (Simmons, 2007).

Relations

- *Solid* is a subclass of *system*.
- A *solid* has some *surface area*.
- A *solid* has exactly one *volume*.

Surface

Description

A *surface* is a two-dimensional submanifold of three-dimensional Euclidean space.

Relations

- *Surface* is a subclass of *system*.

- A *surface* has exactly one *surface area*.

Surface area

Description

The area of a *surface* or of (one of) the exterior surface(s) of a *solid*. More precisely, the class alternatively denotes

- the area of a *surface*, or
- the area of a single exterior surface of a *solid*, or
- the total area of the exterior surface(s) of a *solid*.

Definition

A *surface area* is either a *side area* or a *total surface area*.

Relations

- *Surface area* is a subclass of *physical quantity*.
- A *surface area* has the dimension of area.

Total surface area

Description

The total area of a *surface* or of (all) the exterior surface(s) of a *solid*.

Relations

- *Total surface area* is a subclass of *surface area*.
- *Total surface area* is a subclass of *scalar quantity*.

Volume

Description

The total amount of space enclosed in a *solid* (Simmons, 2007).

Relations

- *Volume* is a subclass of *scalar quantity*.
- A *volume* has the dimension of volume.

Relations

has_area

Description

workaround for Qualified Cardinality Restriction (QCR) which is a feature of modeling currently not available from OWL.

Characteristics

- Specialization of has_property
- Domain: *surface* or *solid*
- Range: *surface area*

has_length

Description

workaround for Qualified Cardinality Restriction (QCR) which is a feature of modeling currently not available from OWL.

Characteristics

- Specialization of has_property
- Domain: *surface* or *solid*
- Range: *radius* or *diameter* or *edge length* or *height*

has_volume

Description

workaround for for Qualified Cardinality Restriction (QCR) which is a feature of modeling currently not available from OWL.

Characteristics

- Specialization of has_property
- Domain: *surface* or *solid*
- Range: *volume*

hasShapeRepresentation

Description

The relation hasShapeRepresentation points from a *system* to the *solid* that represents its geometry.

Characteristics

- Specialization of hasAspectSystem
- Domain: *system*
- Range: *solid*
- Inverse: representsShapeOf

hasSurfaceGeometry

Description

The relation hasSurfaceGeometry points from a *system* to the *surface* that represents its geometry.

Characteristics

- Specialization of hasAspectSystem
- Domain: *system*
- Range: *surface*
- Inverse: representsSurfaceGeometryOf

representsShapeOf

Description

The relation representsShapeOf points from a *solid* to the *system* whose geometry the *solid* represents.

Characteristics

- Specialization of representsAspectOf
- Domain: *solid*

- Range: *system*
- Inverse: hasShapeRepresentation

representsSurfaceGeometryOf

Description

The relation representsSurfaceGeometryOf points from a *surface* to the *system* whose geometry the *surface* represents.

Characteristics

- Specialization of representsAspectOf
- Domain: *surface*
- Range: *system*
- Inverse: hasSurfaceGeometry

References

BIPM (2006) *The International System of Units (SI)*, 8th edition. SI brochure, published by the International Committee for Weights and Measures (Bureau International des Poids et Measures, BIPM). Online available at http://www.bipm.fr/en/si/si_brochure/. Accessed September 2007.

BIPM (2007) *BIPM: Bureau International des Poids et Mesures*. Website, available at www.bipm.org. Accessed October 2007.

Biron PV, Permanente K, Malhotra A (2004) *XML Schema Part 2: Datatypes Second Edition.* W3C Recommendation. Online available at http://www. w3.org/TR/xmlschema-2/. Accessed January 2007.

Carlisle D, Ion P, Miner R, Poppelier N, eds. (2003) *Mathematical Markup Language (MathML) Version 2.0 (Second Edition)*. W3C Recommendation, online available at http://www.w3.org/TR/MathML2/. Accessed September 2007.

Chertov AG (1997) Units of physical measure. In: Grigoriev IS, Meilikhov EZ (eds.): *Handbook of Physical Quantities*, CRC Press.

Daintith J (2005) *Oxford Dictionary of Physics.* Oxford University Press.

Gruber TR, Olsen GR (1994) An Ontology for Engineering Mathematics. In: Doyle J, Torasso P, Sandewall E (eds.): *Proceedings of Fourth International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann. Online available at http://www-ksl.stan-ford.edu/knowledge-sharing/papers/engmath.html. Accessed September 2007.

Simmons B (2007) *Mathwords* – website. Online available at http://www.math- words.com/. Accessed October 2007.

Weisstein E (2007) *MathWorld* – website. Online available at http://mathworld. wolfram.com/. Accessed October 2007.

References

BIPM (2006) *The International System of Units (SI)*, 8th edition. SI brochure, published by the International Committee for Weights and Measures (Bureau International des Poids et Measures, BIPM). Online available at http://www.bipm.fr/en/si/si_brochure/. Accessed September 2007.

BIPM (2007) *BIPM: Bureau International des Poids et Mesures*. Website, available at www.bipm.org. Accessed October 2007.

Biron PV, Permanente K, Malhotra A (2004) *XML Schema Part 2: Datatypes Second Edition.* W3C Recommendation. Online available at http://www. w3.org/TR/xmlschema-2/. Accessed January 2007.

Carlisle D, Ion P, Miner R, Poppelier N, eds. (2003) *Mathematical Markup Language (MathML) Version 2.0 (Second Edition)*. W3C Recommendation, online available at http://www.w3.org/TR/MathML2/. Accessed September 2007.

Chertov AG (1997) Units of physical measure. In: Grigoriev IS, Meilikhov EZ (eds.): *Handbook of Physical Quantities*, CRC Press.

Daintith J (2005) *Oxford Dictionary of Physics.* Oxford University Press.

Gruber TR, Olsen GR (1994) An Ontology for Engineering Mathematics. In: Doyle J, Torasso P, Sandewall E (eds.): *Proceedings of Fourth International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann. Online available at http://www-ksl.stan-ford.edu/knowledge-sharing/papers/engmath.html. Accessed September 2007.

Simmons B (2007) *Mathwords* – website. Online available at http://www.math- words.com/. Accessed October 2007.

Weisstein E (2007) *MathWorld* – website. Online available at http://mathworld. wolfram.com/. Accessed October 2007.

Appendix A Documentation Format

Classes

Classes are characterized by the following categories:

Description: A lexical description of the class, for example “A chemical reactor is an apparatus for holding substances that are undergoing a chemical reaction.” The description explains the meaning of the class to the user.

Definition: Unlike a description, a definition can be transcribed into a formal ontology language, where it establishes the set of necessary and sufficient conditions from which the membership of an ontological concept (class or individual) to the class can be inferred. Classes for which such a definition can not be indicated are called primitive classes.

Relations: The following characteristics are indicated, if existent:

- *Specialization*. A list of parent classes from which the current class is derived via specialization.
- *Disjointness*. A list of classes which are disjoint with the present class. Disjointness between classes means that an instance of the first class cannot simultaneously be an instance of the second class.
- *Restrictions*. Restrictions of binary relations (or attributes) specify the existence of a relation (or attribute) as well as its cardinality and value range with respect to the current class.

Usage: Some recommendations for the use of the class may be given if such advice is required.

Relations

Binary relations are characterized by the following categories:

Description: Similar to that of classes mentioned above.

Characteristics: The following characteristics are listed, if existent:

- *Specialization*. A listing of the relations from which the relation is derived via specialization.
- *Domain*. The domain of the relation.
- *Range*. The value range of the relation.
- *Inverse*. The inverse of a relation.
- Further characteristics, such as if the relation is *transitive*, *symmetric*, or (*inverse*) *functional*.

Usage: As above.

Attributes

Attributes are characterized by the following categories:

Description: As above.

Characteristics: The following characteristics are listed, if existent:

- *Specialization*. A listing of the attributes from which the attribute is derived via specialization.
- *Domain*. The domain of the attribute.

- *Range* or *datatype*. The value range of the attribute, which is usually indicated by referring to a built-in XML Schema Datatype (Biron et al., 2004).
- Further characteristics, such as if the attribute is *functional*.

Usage: As above.

Individuals

Predefined individuals are characterized by the following categories:

Description: As above.

Characteristics: The following characteristics are indicated, if existent:

- *Instance of*. The classes from which the individual is instantiated.
- *Different from*. A list of individual which are explicitly declared to be different from the present individual.
- *Relations*. Instances of binary relations the individual is involved in.
- *Attributes*. Attribute values of the individual.

Usage: As above.

Notation Conventions

Classes and relations of the Meta Model are named according to the CamelCase[5] naming convention: UpperCamelCase notation is used to denote identifiers of classes, while relation identifiers are represented in lowerCamelCase notation. No particular naming convention is followed for identifiers of individuals (i.e., instances of classes).

In this document, class identifiers are highlighted by *italicized sans-serif font*; for better readability, the UpperCamelCase notation is not applied in the text, but the individual words that constitute the class identifiers are written separately and in lowercase (e.g., *class identifier*). If relations are explicitly referred to in the text, they are written in lowerCamelCase notation and are additionally highlighted by sans-serif font. Individuals are accentuated by **bold sans-serif font**. Partial models are denoted **bold serif font**, *italicized serif font* refers to ontology modules.

In figures, a graphical notation in the style of UML class diagrams is used; the basic elements are depicted in Fig. 33. Grey shaded boxes represent *classes*, white boxes represent *individuals*. *Attributes* are denoted by grey shaded boxes with dashed boundary lines, *attribute values* by white boxes with dashed boundary lines. *Specialization* is depicted through a solid line with a solid arrowhead that points from the subclass to the superclass. A dashed line with an open arrowhead denotes *instantiation*. *Binary relations* are depicted though solid lines. Three basic relation types are distinguished: a line with one open arrowhead represents a *unidirectional* relation; a line with two open arrowheads represents a *symmetric* relation; a line without any arrowheads represents a bidirectional relation[6]. Finally, graphic elements for two special types of relation are introduced: an *aggregation* relation is depicted through a line with a white diamond-shaped arrowhead pointing towards the aggregate class. Similarly, a black diamond-shaped arrowhead indicates a *composition* relation.

---

[5] CamelCase is the practice of writing compound words joined without spaces; each word is capitalized within the compound. While the UpperCamelCase notation also capitalizes the initial letter of the compound, the lowerCamelCase notation leaves the first letter in lowercase.

[6] In OWL, a bidirectional relation is modeled through a unidirectional relation and its inverse.

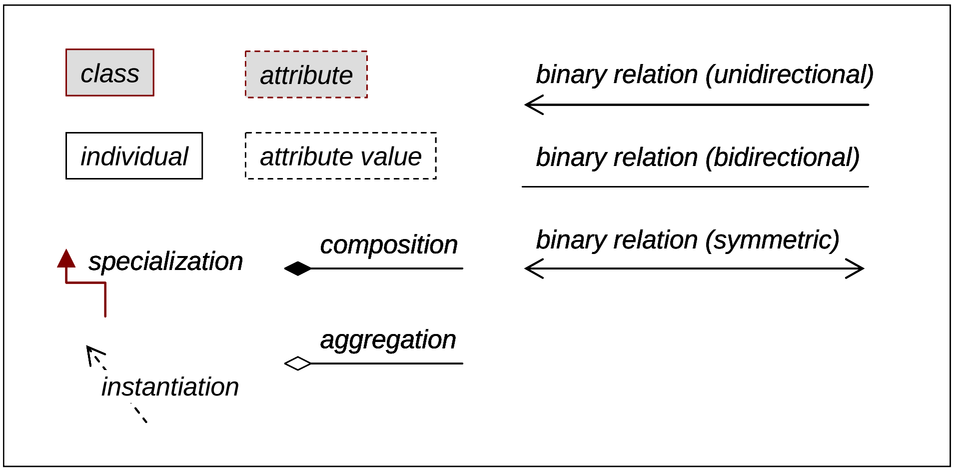

Fig. 33: Basic elements of graphical notation

Index of Concepts