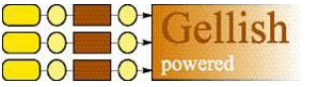


Gellish
A Formalized Natural Language for Semantic Modeling

a language defining ontology
facilitating
information exchange and interoperability of systems

Edition 2

Andries van Renssen

Gellish.net

Copyright © 2024: Dr. Ir. Andries van Renssen.
Gellish.net, Zoetermeer, The Netherlands
www.gellish.net
All rights reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the Author.

ISBN nr: 978-1-304-51359-5

Table of content

1 Introduction

Semantic modeling is a methodology for expressing information in such a way that its meaning can be interpreted from the data itself. The Gellish semantic modeling methodology creates expressions in a computer interpretable and formalized natural language. Interpretation of expressions in that language do not require a data model nor additional (meta) data. The methodology intends to facilitate interoperability of systems and parties and data exchange between systems of various parties without the need for data conversions. It can also facilitate standardization of terminology and reuse of software.

The Gellish syntax is defined in the 'Gellish Syntax and contextual facts' document whereas the Gellish semantics is defined in the Gellish taxonomic dictionary files, especially in the Gellish expressions in the 'Formal language definition base' file. Both are free of charge downloadable from the Gellish website. This book is an elucidation of that language definition.

Semantic modeling in Gellish English results in collections of expressions in formalized English. Such collections of expressions are called semantic networks or semantic models. Such networks can be about individual things and occurrences and/or about kinds of things. The latter typically express possibilities or knowledge, requirements and/or definitions. Semantic modeling can be applied in numerous domain disciplines, including manufacturing, technology, commerce, as well as for modeling human and organizational relationships.

This book intent to elucidate the definition of the Gellish family of formalized natural languages, as every natural language in principle has a Gellish variant: Gellish English, Gellish Dutch, etc.. Each of those family members is made as close as possible to the respective natural language, while maintaining being unambiguous and computer interpretable. The scope of the languages aims to cover in principle any application area and is not limited to a particular universe of discourse (UoD). All natural language variants of

Gellish share the same concepts, whereas those concepts are denoted in various languages by different terms and phrases. Although each concept has the same unique identifier (UID) in each language variant. For example, the concept 'person' has a Gellish unique identifier (990010), independent of the various terms by which that concept is denoted in the various languages and independent of the number and kind of 'attributes' by which the concept is described.

This book describes the core concepts in the Gellish dictionary-ontology and the structure of expressions in Gellish. The definitions of the concepts and their relations as well as the vocabulary of Formalized English is specified in the electronic Gellish formalized English Taxonomic Dictionary. Gellish formalized languages are user extensible. The Communicator reference application software that demonstrates how Gellish can be applied is available on GitHub in two versions. A tutorial and wiki guide is available on the Gellish.net website.

1.1 Semantic versus conventional modeling

Conventional data modeling is based on the conviction that domain specific data models are necessary for the design and construction of databases. Such data models define the storage capabilities and constraints of the databases or data exchange interfaces for a particular application domain or Universe of Discourse (UoD). It typically includes defining 'classes' or 'entity types' and their 'data elements' or 'attribute types' and 'relationship types' in their application domain. Entering data in a defined database is basically a matter of instantiating its data model. However, every developer is free in defining his 'classes' etc. so that the same concept is defined differently in every data model and as a consequence such databases are mutually incompatible. Thus as a consequence, data exported from one database need to be converted before it can be imported in other databases. *Instances* of data models are conventionally not called data models, although they can be regarded as being information models in their own right.

The Gellish semantic modeling methodology uses a different approach. It applies the predefined formalized natural language

Gellish, which includes a syntax as well as an extensive dictionary for the semantics. The Gellish dictionary contains not only definitions of concepts and the vocabulary of the language, as in ordinary dictionaries, but also a taxonomy (a subtype-supertype hierarchy) and a language defining ontology. This makes that the Gellish language can be used for expressing knowledge and requirements as well as for the expression of information about individual things. Gellish is intended to be universal, just as natural languages are, thus enabling expressing nearly any knowledge, requirements, ideas, facts, queries and responses. The language is the same for many application areas. Gellish is more flexible than (fixed) data models and has nearly unlimited capabilities for expressing, storing and exchanging information. Thus for a new database in another application domain it is needless to define a new language. It may be required only to enrich the dictionary. This enables reuse of software and reduces the modeling effort for database design. Furthermore, the logic constructs that are included in Gellish enable the application of logic for reasoning and arriving at logic conclusions. For example by applying inheritance rules via the built-in taxonomy and transitive relations. When information is expressed in Gellish, it can be stored directly as a semantic network in a Gellish enabled database. A Gellish enabled database definition does not need to be modified when the business requirements grow or change. Thus semantic databases can serve wide application areas.

Data model developers typically use tools for the definition of data models that are based on dedicated data definition languages (DDL's). Examples of such languages are XML-Schema, SQL/DDL, EXPRESS, etc. or their graphical equivalents, such as UML and IDEFx. The data models act as meta-models (meta-languages) for their content. This means that conventional data modeling distinguishes three separate languages:

- The highest level languages are the data modeling languages in which data models are written.
- The medium level languages are the data models themselves, because they act as meta-languages for their content. Their

vocabularies consist of names of entity types and attribute types or similar things.

- The lowest level is the user language in which the database content is written. This user language typically consists of terms (data) that don't have their own syntax, because the data are stored in the syntactic structure of the data model (the meta-language).

The first two (meta) languages are formalized languages, as they are precisely defined. The user language is usually only partly formalized and does not belong to the domain of the data modelers, apart from the definitions of 'allowed values' lists or 'pick lists'. This means that the definition of data requirements and storage capabilities is done in another language than the actual user language. However those languages are not independent from each other, because the content of a database in user language can only be interpreted correctly by knowing the semantics of the data model definitions and its syntax.

Furthermore, it should be noted that data models each covers a limited Universe of Discourse. Thus they fixate and limit the data storage capabilities of databases by allowing only instantiations of their entity types/classes. Thus data models in fact are restrictive (meta) languages without flexibility to store other information than the scope of the models allow. This may prevent that unwanted data is entered, but has as disadvantage that costly database conversions are required when the scope of the system is extended.

The Gellish language does not make such distinctions in (meta)languages and UoDs. It is flexible and does not fixate nor limit data storage capabilities of semantic databases. It is a single formalized language that enables the specification of data requirements and data storage capabilities as well as enables the storage of any expression of information, knowledge or requirement in universal databases and messages. This means that there is no meta language or data model required for guiding the expression of information or for the interpretation of expressions.

Semantic modeling is a potential successor of conventional data modeling, and does more than that, as it also standardizes natural

language user terminology by introducing the use of a common dictionary, which enables harmonization of terminology in database systems and data exchange messages. Gellish formalized languages are user extensible and enable the definition and use of company specific terminology and synonyms.

1.2 Terminology

This document uses the following terms:

Topic — Something about which expressions of ideas can be communicated.

Fact — Something that is the case.

Possibility — Something that might be the case and maybe is the case. People can communicate about it by expressing an idea about it with a particular communicative intention.

Idea — Something that is the case, is assumed to be the case, or is wanted to be the case or that was the case, either in a real or in an imaginary world. It may be uttered with a communicative intention, such as a statement, denial, promise or question, etc. about a topic.

Expression — A formulation of an idea about a topic, including an intention with which it is formulated and optional contextual facts about the expression.

Intention — A purpose with which an idea is expressed and communicated, which purpose typically can be derived from the way in which the idea is expressed in natural language. For example, the intention to make a statement, to ask a question, etc.

Elementary idea — A basic idea from which atomic ideas can be composed. There are two basic kinds of elementary ideas, which are expressed by two kinds of relations. The first one is a relation between a role and a relation, which expresses that the relation requires

that role, whereas that role shall be played by one of the related things. The second one is a relation between a role in a relation and something that plays that role in the relation, which expresses that the player plays the role.

Atomic idea
: A binary idea about a role that is played by something in a relation. An atomic idea can be composed of two elementary ideas about the same role and it can be expressed as a participation relation between something and a relation, which expresses that the thing plays a particular role in the relation.

Binary idea
: An idea in which two things play their role.

Unary relation
: A relation that expresses one atomic idea. This means that the relation expresses that one thing plays a role in the relation. This does not exclude that other things also play a role in the relation.

Binary relation
: A relation that expresses two atomic ideas about the same relation. This means that the relation expresses that there are two things that each plays its role in the relation.

Higher order idea
: An idea in which more than two things play their role.

Higher order relation
: A relation that expresses more than two atomic ideas about the same topic. This means that the relation expresses that there are more than two things that each plays its role in the relation.

Variable order relation
: A relation in which the number of things that are involved varies or can vary over time.

Unit of communication

: An expression of an idea about a topic that comprises only one relation and its communicative intention. Such a relation may be composed of atomic and elementary relations.

1.3 Nomenclature

This book uses conventions for graphical models as explained below.

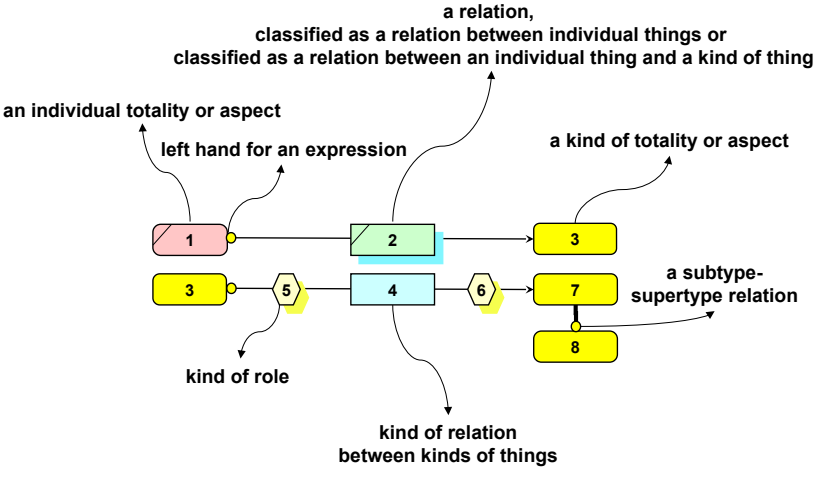


Figure 1, Conventions for graphical models

The graphical elements in Figure 1 have a meaning as follows:

1 = A box with rounded corners represents a totality or aspect or a high level concept and can represent an individual thing as well as a kind of thing.

1 and 2 = A line in the top left corner of a box indicates that the box represents an individual thing.

2 and 4 = A rectangular box with an arrow passing behind the box represents a relation or a kind of relation that is an expression of an idea.

- A term or phrase in a rectangular box that denotes a kind of relation requires by definition a particular kind of left hand object and a right hand object.
- The circle at one end of the arrow indicates the left hand object in the expression. The arrow point indicates the right hand object in the expression.

Note: In this document the unqualified term ‘object’ is used as synonym for the term ‘anything’. The terms left hand object and right hand object refer to the things denoted by terms at the left hand and right hand in formalized language expressions.

2 = A rectangular box with a line in the top left corner indicates that the relation expresses an idea about an individual thing, being either a relation between individual things or a relation between an individual thing and a kind of thing.
Furthermore:
- A shaded rectangular box represents a relation and a classification relation between that relation and a kind of relation.
- A term or phrase in a shaded rectangular box is a name of the kind of relation that classifies the relation.

For example, if the phrase in box 2 would be: ‘is classified as a’, then the arrow behind relation 2 indicates that 1 is related to 3 by relation 2.
Furthermore, the line in the top left corner indicates that 2 is an individual relation, whereas the shade indicates that relation 2 is classified as a classification relation (an <is classified as a> relation).

3 = A box with rounded corners without a line in the top left corner represents a particular concept (kind of thing).

4 = A rectangular box without a line in the top left corner represents a relation between concepts.

5 = A hexagonal box in an arrow at the side of the circle represents a first role (role-1) in a relation that is played by a role player. For example, the role that is played by object (3) in relation (4).
If the hexagonal box is shaded, then the term in the box denotes the kind of role that classifies the individual role.
Often the roles are not graphically represented as their type can be derived from the definition of the kind of relation.

6 = A hexagonal box in an arrow at the side of the arrow point represents a second role (role-2) that is played by a role player. For example, the role that is played by object (7) in relation (4).

7 and 8 = A thick line with a circle at one end is an equivalent of a specialization relation.

- The circle indicates the subtype (8) and the other connected box (7) represents the supertype.
- Thus box (8) represents a particular concept (kind of thing) that is a subtype of (7).
- The inverse means: (7) is the supertype of (8).

2 Semantic modeling

Semantics is the study of meaning and its expressions by humans.

Meaning is about any sensible ideas, including definitions, knowledge, requirements and opinions as well as information about individual things that are in people's mind. And such meaning can be about possible as well as about real and imaginary states of affairs.

In the natural language semantics discipline it is common practice that the actual usage of natural languages is taken as basic material for studying meaning. This makes that the pragmatics of natural language expressions are the basic subject of the natural language semantics study. Such a study derives the apparent rules that apply for the expressions and interpretations from the practice of the language usage. The study also interprets meaning(s) from the expressions in various languages.

However, the meanings themselves, the things that are expressed in natural languages, are language independent, because the same meaning can be expressed in different ways and in different languages or even in artificial languages. So, if there is one meaning then there can be many expressions of that meaning.

2.1 What is Semantic modeling

Semantic modeling use as basic materials (language independent) meanings and then develops a methodology for expressing those meanings in the form of a collection of expressions in a formalized language, whereas those expressions constitute a semantic network (also called an information model) that is interpretable by software in computers.

> A *semantic network of expressions* (i.e. a semantic model) is an information model in which the meaning of data can be interpreted from the expressions themselves, without the need to consult a meta-model or external documentation.

The statement by which a particular meaning is expressed in a semantic formalized language implies that the expressions should include everything that is necessary to interpret the meaning from the expressions. This makes that the language definition is actually a part of the semantic network. It also means that a semantic network shall include the expression of context. The formalized language therefore has to use a formal vocabulary, as well as formal kinds of expressions (kinds of relations between concepts) and a formal syntax (a structure of expression components that make up a sentence). The formality implies that the terms or phrases (the vocabulary) that are used in writing in that formalized language denote concepts that are defined in a computer interpretable formal dictionary. Furthermore, proper definitions of concepts imply that the concepts in that dictionary are arranged in a subtype-supertype structure, also called a taxonomy. The reason why that taxonomic structure is required is explained in chapter 9. And finally the definition of a formalized language requires the expression of generally valid knowledge about the valid combinations of concepts in expressions in the formalized language. Such a formalized language definition forms a collection of expressions that is called a language defining ontology. It is called an 'ontology' because it is a knowledge model that uses relations of various kinds to define the concepts. And it is called 'language defining' because the ontology is limited to expressions that define the formalized language and thus does not include other categories of knowledge. Thus the language defining ontology is a distinct basis for a knowledge representation ontology.

Semantic models in Gellish use natural language terminology, which makes the expressions and models natural language dependent. However, Because Gellish uses language independent unique identifiers (UIDs) to represent the concepts in the expressions, the models become natural language independent. This is possible, because semantic models reflect general human information and knowledge, which is not dependent on its expression in a particular language. By combining natural language terminology with language independent UIDs we get the best of both worlds: human

readability as well as language independent computer interpretability.

Natural language expressions are not a very suitable means to unambiguously express information, as natural languages allow for too much freedom for making expressions, so that unambiguous interpretation of natural language expressions is not achievable with the current generation of computers and software. Unambiguous computer interpretability can be achieved by defining a *formalized language*, which is a formalized subset of natural language in which the ambiguity is eliminated and the degrees of freedom are reduced.

In natural languages, meaning or information is typically expressed as statements, questions, commands, etc. about topics or ideas. According to the Gellish Semantic Modeling Methodology, information is expressed as collections of formalized language expressions, in such a way that software can interpret the meaning (semantics) from the expressions, without the need for using a separate meta-model. The inclusion of the definition of a number of contextual facts (meta-data or data about the ideas) supports an unambiguous interpretation. A standard format enables that database systems can import and export messages that contain expressions that are structured conform the standard data structure/format. Therefore, a native universal data structure (syntax), the Gellish Expression Format is defined, although the language can also be expressed using equivalent formats such as a particular implementation of RDF and triple stores or in object oriented network (graph) databases. The formalized language definition is itself also expressed in that standard format. This allows that database systems can be provided with capabilities for communicating in the formalized language by loading an initial vocabulary and taxonomic dictionary-ontology that defines the formalized language and thus provide the system with a language definition for expressing and interpreting information for data storage and exchange with other systems. The common use of a formalized language also enables that software can interpret the semantic expressions from multiple databases and it enables that different databases can interoperate or be treated as if they are one

distributed database. Interoperation of information about requirements as well as about deliverables enables verification of deliverables and management of the consistency possibly in multiple databases.

This differs from conventional information modeling. In Software Engineering it is a widespread convention to create semantic *meta* models as a basis for database designs and designs for exchanging messages (usually called interfaces). (A meta model is a model about an instance model) Such a meta model defines the database structure or message structure and acts as its documentation. Typically the meta model remains separate from the database instances (its content). To interpret the meaning of the data instances in a database or message the software uses the meaning that is contained in the semantic meta model. Typically, each database and message uses its own meta model, thus the data structures of most conventional databases and messages are different. The different meta models for different databases are the root cause of the costly and time consuming process of integrating data from different databases and developing new interfaces.

Semantic modeling thus means that meaning is included in and can be inferred from the created semantic models. To enable this, it is required that not only objects and aspects are defined and represented in the expressions, but the kinds of the relations between the things shall also be defined and explicitly be represented in the expressions. Therefore, semantic models are relation oriented or expression oriented (although they can be implemented in an object oriented manner).

2.2 Formalized languages

The above description of semantic modeling illustrates that a semantic model or semantic network is a collection of expressions in a computer interpretable formalized language. For the definition of such a formalized language we need to distinguish between:

- Language definition, comprising the language defining ontology and the syntax definition.

- Rules and guidelines for creating valid expressions
- Language usage, comprising the creation of semantic information models
- Verification of the correctness and consistency of expressions

The language definition requires the definition of at least three components:

- The Syntax, which consists of
  - Unambiguously defined syntactic structures and rules on how to express what needs to be communicated, which implies the rules on how to interpret the expressions.
- The Lexicon, which consists of
  - Unambiguously defined concepts and individual things and their denotations by terms and phrases, being the components from which expressions in syntactic structures can be formed. This includes also unambiguously defined concepts for relations (kinds of relations).
- Semantic patterns, which consists of
  - A specification of the minimum amount of information and context that shall be expressed in order to enable unambiguous interpretation of meaning.

In other words: a language definition consists of a definition of words and sentences (statements, questions, etc.), their structure and their context.

There are formal languages that are defined by using artificial terminology, but Gellish formalized languages are based on natural language terminology. Artificial terminology is practiced for example in formal logic notation systems that use for example formulae and parameters to make expressions. The Gellish methodology provides a means for making formal expressions using formalized natural language terminology in universal semantic patterns. This means that formal expressions are made using components that are taken from terms and phrases that are also used and defined in natural languages.

In summary: Formalization of English defines Formalized English as a semantic language that can be used for creating computer interpretable semantic models, being collections of expressions in Formalized English. Similarly, translated terms and phrases provide the vocabulary of other formalized languages (Formalized Dutch, Formalized German, etc.).

2.3 The communication cycle

Communication is an interaction or dialogue between an information creator and one or more addressees that act as information user or replier. An information creator typically is a speaker, an author (writer) or a user of a computer system who enters ‘data’ or creates drawings, whereas an information user typically is a receiver, hearer, a reader or a user of a computer system who searches and retrieves information (data and documents, including drawings) or (re)uses information in a business process in which additional information is generated and which is possibly part of a reply. In some cases this information exchange is a one way traffic, in other cases there is real communication in the form of a dialogue. Data exchange between computers is traditionally primarily one way traffic, but is transforming more and more into dialogues, in which receiving systems are expected to respond on the content of the messages they receive. This communication process is illustrated in Figure 2.

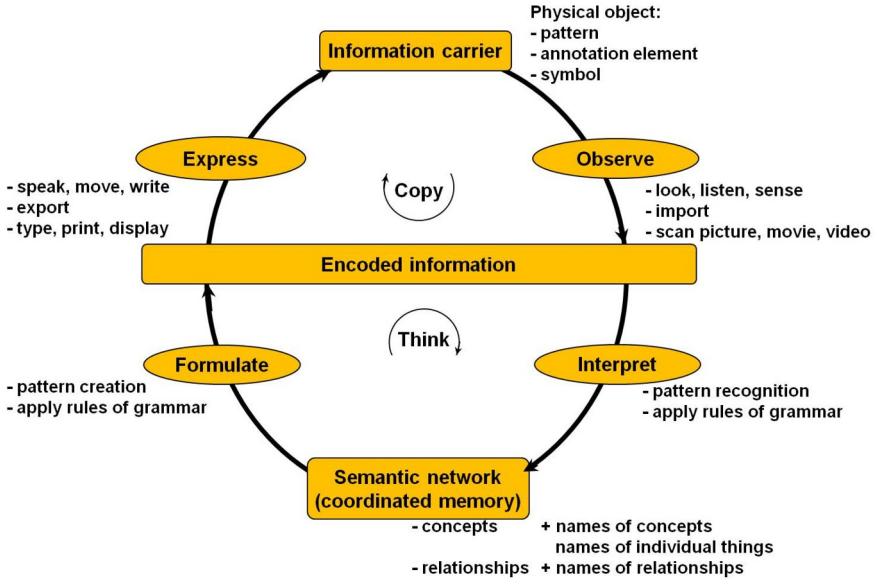


Figure 2, Communication cycle

The picture in Figure 2 illustrates that a two way communication or dialogue between different parties is a continuous sequence of actions in a 'Communication cycle'. The cycle always starts in a 'semantic network' in a human brain or in software in a computer or robot with the production of a thought or idea. The idea is first formulated in (or translated to) a coding system or language (the encoded information) and then expressed as aspects of a physical information carrier. This results in an expression or message (as output) which is transported to a receiving party. That party observes the information carrier and either copies it to create another information carrier, or the party interprets the message (as input) and stores it in its memory (semantic network), whereas the party may produce another thought, which starts the next cycle.

For example, assume that a person has an idea about something that is the case in the real world or in an imaginary world, which he wants to communicate with somebody. In the formulation phase he uses the rules of some language to formulate his idea, for example in English. This formulation process will result in information that is encoded, in this example in English. In the expression phase this

encoded information guides his speaking mechanism in uttering the words or it guides his fingers in typing characters on a keyboard. The expression phase results in a physical carrier of information, for example in the form of ink on paper or sound waves or modulated signals, such as radio waves. A similar process takes place at the interpretation side, where the same definition of the language should be used. When an idea is not expressed, but internally interpreted and further processed, then it is called 'thinking' or 'processing' and when an expression is reproduces without interpretation, it is called copying. This information cycle illustrates the importance of the common use of a formalized language by all communicating parties for a correct interpretation of the expressions, which is fundamental for interoperability of systems.

2.4 Formalization of languages

The Gellish formalized language was initially developed by generalizing limited conventional data models and by addition of flexibility. After discovering the equivalency between generalized data models and natural languages with their general applicability and flexibility the development of a formalized language became regarded as the formalization of natural languages through simplifying and reducing the rich expression capabilities of natural languages. In the following paragraphs we will therefore discuss the main simplifications of natural language that are applied for the development of the Gellish family of formalized languages. These simplifications are:

1. Decomposing meaning into 'basic semantic units'
2. Separating concepts from terminology
3. Separating intentions from topics
4. Separating timing from time independent expressions
5. Use of singulars and numbers to denote plurals
6. Expressing contexts

2.4.1 Decomposing meaning into 'basic semantic units'

There are many ways in which meaning can be expressed. Examples are not only through sentences of spoken and written words, but also through technical drawings, 3D models, (standard) forms such as data sheets and (standard) tables, such as database tables. Each of such a way of expressing meaning uses conventions or rules for expressing and interpreting expressions. Consistent collections of such conventions form coding systems that can also be called 'languages'. The art (or science) of converting such ways of expressing into data models is called information analysis and data modeling. Semantic modeling is the methodology to express such information in a formalized natural language, or in other words in semantic information models.

A Gellish semantic modeling process begins with decomposing meaning into 'basic semantic units' which are expressions in the form of one or more (binary) relations between things, whereas the relations are chosen from the standard kinds of relations in the Gellish dictionary. As part of this process, indirect references to things are replaced by direct references and implied relations are replaced by explicit relations of explicit kinds. This process converts complex and long sentences into collections of simple short expressions. For example, a sentence such as

- the Erasmus bridge in Rotterdam which is red

is converted into four basic semantic units:

- the Erasmus bridge is located in Rotterdam
- the Erasmus bridge has as aspect CE
- CE is classified as a color
- CE is qualified as red

In the latter collection of expressions CE denotes an individual aspect of the Erasmus bridge and the phrases <is located in>, <has as aspect>, <is classified as a> and <is qualified as> denote implied standardized kinds of relations in Gellish. The collection of expressions can be visualized as a network in which the nodes

represent concepts and individual things and the edges represent binary relations of specified standardized kinds between them. The nodes in this mini network are related with other information that is available about the concepts such as Erasmus bridge, color, etc. This makes that the network forms a part of the total network that also includes the language definition. Thus natural language sentences are converted in collections of binary relations in Gellish.

But what about higher order relations, being relations that relate more than two things? As will be explained later, the higher order relations are not represented by edges, but are represented by nodes in the networks and are modeled as a collection of binary relations with their involved things. This simplified the structure of Gellish by forming networked collections of only *binary* relations.

2.4.2 Separating concepts from terminology

Gellish deals with synonyms, abbreviations, codes and translations for the same concepts and homonyms for different concepts by distinction between the concepts themselves and the multiple names by which they may be denoted.
As said before, concepts and relations represent meaning that is language independent. Therefore, each concept and individual thing, including each relation and kind of relation is represented in Gellish not by a language dependent name, term or phrase, but by its own language independent unique identifier (UID). This means that a language defining taxonomy and ontology which include semantic models that define concepts and kinds of relations can be language independently represented by a network of relations between identifiers of things. For users in a particular language community it is required that the concepts are related to terms and synonyms that form the vocabulary of their language. Thus the unique identifiers of the concepts as well as the network of relations between the concepts are language independent and need not be redefined for other formalized languages in the Gellish family. As a consequence the Gellish family of formalized languages share the same concepts and structure.

This enables the use of synonyms and homonyms and multi lingual dialogs as well as automated translation of expressions that are formulated in any formalized language of the family. This separation between concepts and terminology is further discussed in chapter 5.

2.4.3 Separating intentions from topics

Natural languages use different word sequences, dependent on the intention of the expressions. This variety of expressions is simplified in Gellish by adding explicit intentions to the expression of topics while using uniformity in such expressions, as explained below.
Human speaking or writing produces a stream of expressions by means of a sequence of what can be called expression production acts, also called 'speech acts' (Ref. 5, John R. Searle, Speech Acts). Each expression production act produces an expression that is intended to express what its creator means. A resulting expression is a physical object, such as an information carrier, typically in the form of an audible or electronic signal, or written ink on a physical document. That physical object is a bearer of that meaning, but it is not the meaning itself. Different persons can express the same meaning, resulting in different physical expressions. For example, the same meaning can be expressed in different languages. The interpretation of those various physical expressions should result in one common meaning (a common content). Such a common content can be given a unique identifier (UID) and a description. We will call such a piece of common content (a piece of) 'qualitative information'.

When different persons express information about the same topic, they may exchange ideas about the truth of an idea, or they may negotiate about the execution of an act. In such cases they have a dialogue in which they create various expressions about the same topic. For example, consider the following dialogue about whether the Euromast is located in Rotterdam as follows:

- Where is the Euromast located?
- The Euromast is located in Rotterdam.
- The Euromast is not located in Rotterdam.
- Is the Euromast (really) located in Rotterdam?
- Yes, the Euromast is located in Rotterdam.

All these five expressions are about the same topic[1]. A topic is a state that may be the case, either in a real or in an imaginary world. The topic in this example can be expressed as:

- whether the Euromast is located in Rotterdam.

It was Searle's semantic analysis that demonstrated that the freedom of expressions in natural language that leads to the variety as in those five expressions can be transformed into a much simpler and uniform structure. First it should be noted that the five expressions differ in the *intention* with which each expression is communicated. The above five intentions are:

- Question
- Statement
- Denial
- Request for confirmation
- Confirmation

In other words, the speaker (creator of the expression) has the intention to communicate a question, a statement, etc. In linguistics this intention is called the illocutionary force.
We can now transform the above four expressions in one simplified uniform structure by using these intentions as separate classifications of the idea, while leaving the expression of the idea unchanged. This transformation results in the following expressions:

1 A topic may seem similar to what is called a proposition in logic. However, a proposition may be a statement (which is true or untrue), whereas a topic is not an expression, but something about which an opinion may exist and about which an expression may be uttered.

- Question: the Euromast is located in where
- Statement: the Euromast is located in Rotterdam
- Denial: the Euromast is located in Rotterdam

- Request for confirmation: the Euromast is located in Rotterdam.
- Confirmation: the Euromast is located in Rotterdam.

The combination of a possibility and an intention forms a more complete expression of an idea.

There are many more kinds of intentions. For example, we can distinguish for example between a statement, a question, a request, a promise, a command, a refusal, a denial, withdrawals of each of them, etc. By default we can assume that the intention for an expression is making a statement.

By identification of all those kinds of intentions it becomes possible to simplify the large variety of natural language expression significantly by transforming them into this structure in the Gellish formalized language.

The above semantic analysis demonstrates that statements and queries have the same semantic pattern. This suggests that there is no semantic need to create a separate language for the formulation of queries. In conventional information technology practice there is a distinction between data definition languages for the expression and storage of statements in databases and separate query languages for the selection and retrieval of information from those databases. This appears to be unnecessary.

The expression of queries will be further discussed in paragraph 2.6.

2.4.4 Expressing past, present and future

In natural languages we use various grammatically different expressions to express whether something was the case in the past, is the case in the actual world or will be the case in the future. This variety in expressions can be simplified by explicitly specifying a validity period and a status of each expression. Something is only an expression of a historic fact if the status is ‘history’, ‘deleted’ or

‘replaced’. This enables not to use inflections, such as is, was, be, being, make, making and made, nor is it necessary to define such inflections as synonyms. The validity period plus the specified time determines whether something is the case in the past, present or future.

This enables that Gellish adopted the convention of usually using only the present tense, while adding explicit validity periods to the expressions.

Another option is to model the time explicitly, not by the validity period of the expression, but by including the time in the expression as illustrated below.

Intention	Topic	Status	Date of start of validity	Date of change of expression
statement	**The Erasmus bridge <is located in> Rotterdam**	accepted	1995	2011
statement	**Temperature of R1 <is on scale> 1 degC**	history	2012-12-8 17.22	2012-12-8 17.22
statement	**Act-1 <has as start date> 15 January 2015**	proposed	1 dec 2012	2 dec 2012

Table 1, Validity period

Table 1 presents some examples of statements that are all expressed in the present tense, but only the second statement has a validity that is in the past, because the status is ‘history’. The first and third statements have statuses that do not denote a history, so that the dates of (latest) change of the expression (including a change of status) denote changes of the expressions and not termination of its validity.

The status and timing columns in the above table are a representation of some contextual facts. The contextual facts that represent a validity period are illustrated in Figure 3.

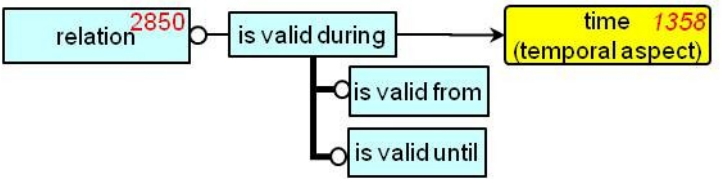


Figure 3, Validity period for a relation

These and other contextual facts are further discussed in chapter 13.

2.4.5 Use of singulars and numbers to denote plurals

In natural language we can use plural words when we denote collections of single items. For example we can use the plural 'books' when we state that collection B consists of (only) books. If we would accommodate for that in Gellish it would mean that the dictionary should nearly be doubled in size with a double specification of its taxonomic hierarchy or software should be able to unambiguously deduce singular terms from the plural names in all languages. This necessity is eliminated in Gellish by making the collections explicit and by including special kinds of relations that classify relations with collections that imply a relation with each component in the collections. For example, we can express the same meaning as above while using the singular word 'book' by stating that every element in collection B is a book. By modeling in that way Gellish does not need plural words in most cases. This is achieved by applying the following rules (see table 2):

- Denote a concept (a kind) always in single form, except when the concept (kind) is a kind of collection. Thus do not specialize kinds of collections by the kinds of components in the collection.
- Allow that individual collections have names that include a plural term.
- Use explicit minimum and maximum simultaneous cardinalities to denote constraints on numbers of individual things of this kind that have a corresponding relation with a single exemplar of the other kind.

- For a relation with a collection that implies a relation with each component in the collection use a kind of relation that makes that explicit. For example use a kind of relation which name starts with 'each of which'.

Intention	Left hand object	Kind of relation	Cardi nalities	Right hand object
Statement	**book**	**has by definition**	**2,n**	**page**
Statement	**Stock of books B**	**is classified as a**		**collection**
Statement	**Stock of books B**	**each of which is classified as a**		**book**

Table 2, Eliminating the need for using plurals

Such a simplification enables that a formalized language dictionary need to include nearly only terms in single form.

2.4.6 Expressing contexts

Communication always has a creator and one or more (potential) addressees. These and other facts make that communication always functions in a context. If the same expression is uttered by different persons at different occasions and different times, then it may have different meanings. Therefore, the context in which an expression is used is usually relevant for its interpretation. However, context is often not an explicit content of an expression, but it is implied with the expression. In Gellish we want to capture all aspects of expressions that are relevant for a proper interpretation. Therefore, any expression should include relevant elements that express the context. We will call such contextual elements ´contextual facts´ about the expressions.

The first contextual facts that we already encountered are answers to the following questions:

- Who is the creator of the expression
- Who is the addressee of the expression
- When was the expression created

These contextual facts illustrate that the above expressions, such as the question 'Is the Euromast located in Rotterdam?' needs a context for a proper interpretation and response. If you read the question in a book (such as this), then you don't need to respond, but if the question is addressed to you, then you need to know who has raised the question (the 'author'). Thus a full semantic model should expand the expression as follows:

> The information creator John addresses addressee Mary on time t_1 with the question: 'Is the Euromast located in Rotterdam?'

Or simplified and formalized in tabular form:

Expression id	Creator	Addressee	Date-time of creation	Intention	Expression of idea
101	John	Mary	t_1	question	the Euromast is located in Rotterdam
102	Mary	John	t_2	assertion	the Euromast is located in Rotterdam

Table 3, Expressions with some contextual facts

A large list of kinds of optional contextual facts in Gellish is discussed in chapter 13.

2.5 Commands

A message in a formalized language, typically consisting of a collection of expressions, may be preceded by a command that indicates the start and the nature of the collection and may be followed by a command to terminate the execution of the command. For example an insertion command has the following pattern:

Intention	Name of left hand object	Name of kind of relation	Name of right hand object
command	insert	the following expressions into	uri/filename
statement	...	...	...
statement	...	...	...
command	terminate	the execution of	insert

A query command has the following pattern:

Intention	Name of left hand object	Name of kind of relation	Name of right hand object
command	select	the following expressions from	uri/filename
question	...	...	...
condition	...	...	...
command	terminate	the execution of	select

Depending on how a sender qualifies the 'intention' of the expressions, the interpretation may result in an insertion of new expressions in existing expressions, or as a modification (delete, historicize or replace) of expressions, or as a query that requires a response to the sender.
A receiving system (an information user or addressee or hearer) that receives such a message should start the interpretation of expressions in such a message by validating it in order to verify that:

- The expressions are semantically correct (see par. 3.7).
  For example, UIDs should be unique and in the correct range, kinds of things should be known according to the dictionary or they are properly defined within the message, the left hand and right hand objects shall be of the correct kind, etc.
- The message is internally consistent and without duplicates,

An *insertion* or *modification* command then requires to verify whether the message is consistent with or duplicates existing expressions. This requires comparing each expression in the message with the expressions about the things that are mentioned in the message and that are already included in the receiving system. Therefore, the interpretation continues with the execution of a query on the expressions that are already included in the receiving system. The result of such a query will include requirements, constraints, possibilities and definitions of concepts. This enables the receiving system to verify the consistency of the new expressions when compared with the existing expressions and enables to further process the insertion.

A query command or ‘select’ command can be executed without further validation. The results can be returned in a message that is preceded by a report command as follows:

Intention	Name of left hand object	Name of kind of relation	Name of right hand object
command	report	the following expressions from	uri/filename
answer	...	...	...
confirmation	...	...	...
denial	...	...	...
command	terminate	the execution of	report

The report can then be post-processed and converted into other layouts, using a report generator.

2.6 Queries and insertions

A query typically consists of a collection of one or more expressions about one or more unknowns. In Gellish, the expression(s) form a mini semantic network or search pattern that includes the unknown(s) and that is intended for finding known objects that have a pattern that corresponds with the search pattern. A query is answered by reporting the found knowns together with information about them.

Queries that are expressed in conventional query languages for databases such as SQL and SPARQL have structures that are quite different from expressions of statements (propositions) in those languages. This differs from natural languages in which queries and statements, as well as denials and confirmations have nearly the same structure and use the same terminology. This raises the question whether the Gellish language that is used to express statements (propositions) can also be used for the expression of questions.

Some differences between kinds of expressions, such as changes in word sequences and the use of question marks, can be eliminated completely when the ‘Speech act’ theory of John Searl is applied. As described in the previous paragraphs, Searl demonstrated that the expressions can be made identical by explicit mentioning a separate

‘intention’ for each expression. For example, the following three expressions only differ in their intention:

Intention	Topic
statement	book B-1 has a price of 110 dollar
question	book B-1 has a price of 110 dollar
confirmation	book B-1 has a price of 110 dollar

The intention on the second row indicates that the second expression of the topic shall be interpreted as a question. As there are no unknowns included in the expression, it implies that the question asks for a confirmation or a denial, which is equivalent to the answer ‘yes/no’ or ‘true/false’.
This illustrates why the ‘Speech act’ theory and other conventions open up the possibility for using the same formalized language for statements (in exchanges messages or data stores) as well as for insertion commands, for queries and for responses (answers), promises, etc. as is described below.

Therefore, a general model of a *query* about an object, when expressed in Gellish, is the same as a general model with *assertions* about any object, apart from the fact that expressions in a query have the following characteristics:

- The intention of an expression in a query has the value ‘question’ or ‘condition’. A condition provides a further specification of the conditions that should be satisfied by the unknown target object(s).
- The unknowns and collections of unknowns in a query are represented by UIDs that should be in the reserved range 1-99, possibly preceded by the prefix ‘unkn:’. And application software may generate such UIDs automatically from names that are preceded by question marks.
- The names of the unknowns are free text.
  If nothing about the name(s) is known, then it is recommended to take a name from the list of reserved names as specified below or to use names that start with a question mark (?) as is the same as the convention for SPARQL. This freedom for names enables searching

on string commonalities (see below).
For example, what, what-1, which person, or ?person.

- It is allowed to specify a part of a name of a searched object in combination with a specification of 'string commonalities' as defined below.

2.6.1 Names and UIDs of unknowns

Thus objects with a UID in the range 1-99 are by definition interpreted as unknowns that are searched for on the basis of expressions that specify the search criteria. This should be consistent with the 'intention'. For example, an unknown with UID '1' might be denoted by a name that is just a question mark ('?') or a question mark followed by a dash and a sequence number ('?-1'), or any other string that is not a name of a known thing. To facilitate that, the following terms are reserved and automatically interpreted as unknowns:

- what
- who
- which xx, whereas xx stands for some kind. For example 'which person', 'which pump', etc.
- where
- when
- how many
- how much

The use of any of these terms or of terms that start with a question mark (such as ?person) *supports* for a human user that the expression should be interpreted as a question. For software that interprets the formalized language expressions the name as well as the UID in the range 1-99 (possibly with prefix 'unkn:' can be used to determine that the expression contains an unknown and thus should be interpreted as (part of) a question.
The above reserved terms may be used more than once for different unknowns provided that different UIDs are used to identify the unknowns. However, it is recommended that the unknowns have

different names, for example by extending a standard name with a dash and a sequence number. For example what-1, what-2, etc.

2.6.2 Simple questions

One pattern for a question is: “what are the object(s) that have particular types of relations with other specified objects?” For example, the question ‘what is classified as a pump’. This question has no UIDs yet. This it is not excluded that a first search may find different concepts with the same name (homonyms). Therefor, a system should reflect the question by asking for verification that the question is properly interpreted as is expressed in Table 4:

43	2	101	1	3	15	201
Intention	**UID of left hand object**	**Name of left hand object**	**UID of idea**	**Name of kind of relation**	**UID of right hand object**	**Name of right hand object**
question	1	what	301	is classified as a	130206	pump

Table 4, A Query

The question that is expressed in Table 4 asks for the object(s) that have a relation of type <is classified as a> with the object 130206 (pump). This expression can be unambiguously interpreted by a computer as a question, because the intention ‘question’ expresses that it is requested to identify the unknown object(s) that satisfy the kind of relation, and it is defined for the family of Gellish languages that a UID in the range 1-99 is an identifier of an unknown, whereas for a human reader the term ‘what’ (or e.g. ‘?pump’) denotes an unknown.

Furthermore it is common logic that the inheritance rules define that the question: “what is classified as a pump?” implies: “what is classified as pump or is classified as one of the subtypes of pump?”. It even also implies: “what has a relation with the concept ‘pump’ or its subtypes whereas the relation is a classification relation or a subtype of classification relation”.

Advanced software should be able to use this as a basis to automatically generate the answer as a list of pumps with their characteristics. Such a list could for example consist of one bicycle pump and two centrifugal pumps, all three being a subtype of pump

and a water pump, not being a subtype of pump, but a role of a pump. Thus the response could be as follows:

P-1	is classified as a	bicycle pump
P-101	is classified as a	centrifugal pump
P-102	is classified as a	centrifugal pump
P-301	has a role as a	water pump

For such an implementation see the Communicator reference application on GitHub.

2.6.3 String commonalities

Software should enable searching for objects by character strings that only partially match with the name of the objects. A specification of such conditions is called a specification of the string commonality. Thus a string commonality is a specification of the conditions under which a search string (term) at the left hand or right hand of an expression should be considered as matching with one or more target strings (names of things). For example, it can be specified how to find all things that are denoted by a term (have a name) that contains a capital P. This requires the specification of string commonality value 'case sensitive partially identical', which specifies that a search string shall be identical to a part of the target term, whereas the case (upper case or lower case) of the part shall match.

An Expression for a query contains two additional components (component ID 80 and 81) in which the commonality criteria for the left hand and the right hand term can be specified. The allowed values for string commonalities are:

- csi: case sensitive identical
- cii: case insensitive identical
- cspi: case sensitive partially identical
- cipi: case insensitive partially identical
- csfi: case sensitive front end identical
- cifi: case insensitive front end identical
- csd: case sensitive different
- cid: case insensitive different

- e: equal
- u: unequal
- le: less than or equal
- ge: great than or equal

2.6.4 Conditions in queries

A query may contain additional conditions on the unknown(s). For example, Table 5 presents the expression of the query 'which P-1 is classified in some way', whereas P-1 may be preceded and/or succeeded by additional characters, while that '?P-1?' also has an aspect that is classified as a height.

43	2	101	60	3	15	201	80
Intention	**UID of left hand object**	**Name of left hand object**	**UID of kind of relation**	**Name of kind of relation**	**UID of right hand object**	**Name of right hand object**	**Left hand string commonality**
question	1	P-1	1225	is classified as a	2	how	case sensitive partially identical
question	1	P-1	1727	has as aspect	3	what	
condition	3	what	1225	is classified as a	550126	height	

Table 5, A Query for objects with aspects

The Query Table 5 includes three unique identifiers (UIDs) in the range 1-99. The first one (1) indicates that P-1 is a string that denotes an unknown object. The first line also includes a 'left hand string commonality' that indicates that search string P-1 may be only a part of the target string(s), whereas the P in the target string(s) shall be in uppercase. Furthermore, only the expressions are requested in which the target object is classified, although that may be in any way ('how'). The results of this query may well deliver the same three pumps as in the above example.

The second line asks for aspects of the resulting objects, whereas the third line specifies the condition that only the aspects are requested that are classified as a height. For querying aspects, Gellish uses the following rule:

When a query asks for an individual aspect, then the response should not only provide a possible name of the aspect, but should also provide its classification, its value and if applicable the scale for the value.

Therefore, a possible response on the above query could be:

P-1	is classified as a	centrifugal pump	
P-1	has as aspect	h-1	
h-1	is classified as a	height	
h-1	has on scale a value equal to	600	mm

Note that Table 5 contains two questions. This means that P-1 should be reported anyway, also when the searched dataset would not contain its height. If the intention on the second line would have been a condition instead of a question, then P-1 should not be reported as it does not satisfy the condition to have a height.

2.6.5 SQL equivalent inserts and queries

Data manipulation languages (DMLs) or Query languages such as SQL are designed to be independent of database structures and of languages that can be used for their content, as long as the database structure is tabular. They do not put any requirements on the tables in the database, neither on their names nor on their columns and column names. This freedom is required because a lack of standardization in database creation methodologies causes that, even when different databases contain the same information about the same kinds of things, those databases are generally composed of different tables, with different table names and different numbers of columns with different column names. Thus INSERTs as well as SELECT statements (for queries) will be different for each database.

This freedom implies that these languages presuppose that authors of inserts and queries have knowledge about the internal structure (syntax) of the queried database as well as of the used terminology for table columns and table content. Thus, SQL and other query languages themselves do not deal with any meaning (semantics) of the columns of the tables and are also independent of the terminology that is used for the content of the tables. They are generic query languages that contain a minimum of semantics.

Insertion

This can be illustrated on the insertion of some data in a relational database. For example, the following simple insertion of prices, titles and classification of books in a database table called 'Book' is written in SQL as follows (this example is adapted from http://en.wikipedia.org/wiki/SQL):

INSERT INTO Book
*(*title*,* price, type*)*
VALUES
('B-1', 110)
('B-2', 120, 'paperback') ;

Apparently the author of this insert knows (and must know) that there exists already a table, called 'Book', that has at least three columns, called title, price, and type, whereas he also knows that a title is the title of a book and a price on the same row is the net selling price (in dollar) of a single copy of a book with that title and that a type denotes a subtype of the concept book that classifies the book on that same row. The insert statement also introduces a new free term 'paperback' in the vocabulary of the content, unless the term 'paperback' is a predefined allowed value for 'type'. (Note that a column 'type' in another table in the same database may have another meaning.)

The same content could however be stored in a databases that have different definitions. For example, the database 'Product', with columns that have names such as 'name', 'net price', 'product type'. Then the INSERT would have been different, although the content would be the same, and the queries on those two databases will also be different.

Selection

Data can be retrieved from such database tables with queries that are expressed in such query languages and those expressions are also dependent on the database structure. This can be illustrated on an example select statement from a 'Book' table, which is expressed in SQL as follows:

```
SELECT *
        FROM  Book
        WHERE price > 100.00 ;
```

This simple query apparently assumes the same knowledge from its author as is required for an insertion.
Also for queries holds that the same question should be formulated in a different way when it was a query on the 'Product' database. It should also be noted that the expressions for insertion of information are significantly different from expressions of a query about the same information and those expressions are again different from expressions that present the results of a query.

Equivalent inserts and selects in Gellish

If the Gellish formalized language is adopted for expressing inserts and queries, results and messages, then the structure of expressions as well as their content are all the same. This is achieved by the rule that all Gellish formalized language expressions have the same expression components (thus they all can be stored in one collection of standard expression components) and that all Gellish expressions use the same (extensible) taxonomic dictionary with predefined concepts. This also hold for example for terms such as book, title, price and paperback. Furthermore, the expressions for insertion are similar to the expressions of queries and to stored and exchanged information.

Thus formalized language expressions of insertions and queries are only determined by the semantics and are not determined by the many possible database structures, and they are independent on the variety of terminology that is used in current practices for names of entity types and names of attribute types.

This means that information that is expressed in a formalized language can be inserted in any database that is based on the formalized language or that has import and export mapping to the formalized language expressions (within access and requirements constraints). Thus the system independent expressions don't need to be rewritten for other databases. Table 6 presents an example of an

insert command for information about the prices of two books with expressions in a formalized language that are database independent.

Intention	UID of left hand object	Name of left hand object	Name of kind of relation	UID of right hand object	Name of right hand object	UoM
command	195070	insert	the following expressions into	101	ABC	
statement	102	B-1	is classified as a	490023	book	
statement	102	B-1	has as aspect	103	P-1 of B-1	
statement	103	P-1 of B-1	is classified as a	550742	price	
statement	103	P-1 of B-1	has on scale a value equal to	920366	110	$
statement	104	B-2	is classified as a	493755	paperback	
statement	104	B-2	has as aspect	105	P-1 of B-2	
statement	105	P-1 of B-2	is classified as a	550742	price	
statement	105	P-1 of B-2	has on scale a value equal to	920376	120	$
command	193423	terminate	the execution of	195070	insert	

Table 6, Insert product data in database ABC

Note: Each row in Table 6 represents an expression. The names of objects are repeated for readability and UoM means Unit of Measure or scale. Table 6 only shows a subset of the components of Gellish expressions and thus of the columns in a Gellish Expression Format table. In a full Gellish Expression Format each line has more UIDs and contextual facts, such as the validity period, status, originator, etc. This enables for example adding multiple prices in various currencies and each with its own validity time period, if the cardinality constraints allow for that.

The body of Table 6, without the first and the last line can be copied exactly into a database table, such as ABC, because the storage table has an identical expression structure.

The above query in SQL is expressed in a formalized language as follows:

Intention	UID of left hand object	Name of left hand object	Name of kind of relation	UID of right hand object	Name of right hand object	UoM
command	193617	select	the following expressions from	101	ABC	
question	1	?Book-1	is classified as a	490023	book	
question	1	?Book-1	has as aspect	2	?Price-1	
question	2	?Price-1	is classified as a	550742	price	
question	2	?Price-1	has on scale a value greater than	920053	100	$
command	193423	terminate	the execution of	193617	select	

Table 7, Query on ABC in the form of a product model

Comparison of Tabel 6 with Tabel 7 shows the similarity of the two models, which demonstrates that the expression of information and the expression of queries can be done in the same language. Thus there is no need for a dedicated query language.

Note: A query may search for and select from more than one table at the same time. Because the various tables have the same definition, tables do not need to be JOINed; only the search results should then be presented to the user as a combined result.

The query that is expressed in Table 7 illustrates that software should take the taxonomy of concepts into account. For example, the taxonomic dictionary specifies that the concept paperback is a subtype of book. If the software processes that information correctly, then a query on book will also find the paperbacks. This hierarchy enables to simply modify the query to search e.g. on paperbacks only or on any other subtype. This would be more complicated in an SQL search in the above table ‘Books’.

In SQL and asterisk (*) can be used to specify that ‘all’ attributes from a table should be reported. This assumes that the authors knows what ‘all’ means, thus which attributes are in the table. However, ‘all’ does not mean ‘all information about the selected books’. Because, when there is information about the books in other tables the query becomes more complicated. In Gellish modeling approach the kinds of relations that are queried can be specified

more precisely. For example the query in Table 7 uses kind of relation <has as aspect> and thus it only asks for aspects, whereas on the following line it is specified that only aspects are required for which holds that the aspect <is classified as a> price. The query can easily be extended with additional requests for other information, such as:

question	What-1	is located in	Some location-1
question	Some location-1	is classified as a	building

Or with the very generic question:

question	What-1	is related to	A-1
question	A-1	is classified as a	anything

This latest question asks for everything that is known about the books.

2.6.6 SPARQL and RDF

SPARQL is a query language that is especially made for querying databases that are formatted conform RDF, also called 'triple stores'. As shown above, the semantics of questions and other expressions require more than just triples, such as units of measure and contextual facts. That is the reason why many implementation specify extensions of RDF to represent collections of triples, which are called 'named graphs' as is also applied in ISO 15926-11, which standardizes an RDF implementation of Gellish Formal English.

Extended RDF implementations of a formalized language can use SPARQL directly. However, RDF itself defined a syntax and a minimum of semantics (it only defined a few concepts), just as SQL. This enables that in RDF expressions any kind of relation ('predicates' in RDF) and any left hand and right hand term ('subject' and 'object' in RDF) can be used. Thus everybody can use his or her own 'namespace' and own ontology. This powerful flexibility at the same time reveals the weakness towards interoperability, because RDF does not standardize the language in

which databases, messages and query contents can or shall be expressed.

The expression of inserts and queries can be made database system independent only when an extended RDF is combined with a semantically rich formalized language, such as Formalized English. Such a combination provides a language that includes semantics as well as syntax (format).

Another question is whether the SPARQL syntax is to be preferred above the tabular Gellish Expression Format syntax as is used in Table 6 and Table 7. The commonalities and differences between these two formats can be illustrated on the SPARQL example query for a 'foaf' (friend of a friend) database http://en.wikipedia.org/wiki/SPARQL:

```
PREFIX foaf: http://xmlns.com/foaf/spec/
SELECT ?name ?email
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:mbox ?email.
}
```

The above example shows that SPARQL also presupposes knowledge about the particular structure of the queried database. Although the structure of RDF expressions is database (data model) independent, this example demonstrates that this query is dependent on the structure of the foaf database and relies on the understanding of the content of the foaf ontology (http://xmlns.com/foaf/spec/), which includes a database structure (table definitions) with definitions of 'classes' (entity types) that have pre-defined 'properties' (attribute types). For example the class foaf:Person is not the same as the generic concept 'person', because the foaf ontology defines a foaf:Person as a person that has a number of predefined 'properties' (attributes) with specific names. Thus a foaf:Person is factually defined as a particular collection of 'properties'. For example the foaf ontology pre-defines that a foaf:Person can have or has a surname, as well as e.g. publications and a currentProject, and inherits an mbox. Apparently the foaf

ontology defines a very specific ‘language’ that cannot be merged with other ontologies/languages and thus the query will only work on a foaf database and shall be rewritten for any other database.

This demonstrates why the neutral form of expressions in Table 6 and Table 7 has advantages.

3 Semantic models in formalized languages

> *Definition:* A Semantic Model is a chain or network of formalized expressions (also called an information model) in which the meaning of the content (data) can be interpreted from the model itself, without the need to consult separate information about that data.

The formalization implies that meaning can be expressed as a collection of connected expressions, called units of communication, that are syntactically and semantically comply with the language definition, and are computer interpretable. The definition furthermore states that the expressions include all information that is required for their interpretation, without the need to consult system documentation, such as meta-data, data models (database definition schemas) or program code.

Expressions in semantic models comprise terms as well as phrases or verbs that relate the terms. The terms represent things about which something is expressed and the phrases or verbs represent relations of particular kinds between the things that are represented by the terms. Expressions therefore consist of relations between terms, which relations are classified by explicitly defined kinds of relations.

To some extent it is possible to regard natural languages also as being semantic models. The allowed sentence structures form the possible structures of 'messages' and the allowed terms and phrases define the remainder of the language. The enormous freedom to make sentences in natural languages mean that the underlying semantic patterns can be very complicated and flexible, whereas different users apply different parts of the allowed structures when they make sentences. This makes it very difficult for computers to interpret and to generate natural language expressions. This is the reason to formalize and predefine the allowed sentence structures and the allowed terms and phrases in the definition of a formalized language.

The above definition also states that a Semantic Model is a collection of expressions in a *formalized* language. A formalized language means that the grammar (the syntactical structure of the expressions and the lexicon) of the language is explicitly defined (and thus limited) and that the allowed components that are used in the expressions are also explicitly defined. Such explicit definitions can be pre-defined in the dictionary of the formalized language, or they can be user defined as part of the model.

Simple semantic models can be presented graphically as networks of nodes and vertices, although such networks usually ignore intentions, classifications and contextual facts. Such networks are often called graphs. Nodes in such networks represent anything that can be thought and communicated about, real as well as imaginary. The vertices represent (binary) relations between the things that are represented by the nodes. These binary relations represent expressed ideas, such as statements, opinions or questions about topics or states of affairs.

Semantic databases that support the storage of information that is expressed in a formalized language should have data structures that support storage and retrieval of any semantic expression in that formalized language. Therefore, they should enable storing a definition of the formalized language as their common initial content, followed by storing expressions of ideas that are expressed in that language. Thus, software should be able to interpret any semantic model in that formalized language. Preferably it should also enable treating different semantic models as if being one distributed model.

Interoperability or integration of semantic databases that apply the same formalized language thus should only require verification and management of the consistency of their content.

Semantic modeling thus means that a sending party creates semantic models in a formalized language that include sufficient expression of meaning such that the meaning can be inferred from the models by the computer of a receiving party that can interpret the formalized language.

3.1 Scope: information, knowledge and requirements

The scope of a conventional information model is limited by the definition of the data model for which the model enables making instances. The scope of a Gellish semantic model is in principle unlimited, because the formalized language has no scope limitations. For example, a conventional Building Information Model (BIM) will have a scope that is defined and limited by the scope of its data model (for example the IFC data model of ISO 16739). However a BIM in the form of a 'semantic model' does not have constraints on its scope. Therefore, the required scope of a semantic model can be defined explicitly by the party that requires the model.

Information that is exchanged between parties or is stored in databases can be very different. It includes models of information about individual thing, but it also includes knowledge models, which consists of expressions of what can or might be the case about kinds of things in general. Furthermore it may include generic requirement models with the expression of what is required for kinds of things or requirements for the realization of individual things, such as is typically expressed in designs and specifications. It may also include definitions of things that are by definition the case. All such information can be covered by Gellish semantic models and can be expressible in Gellish formalized languages.

The scope of the expressions that are considered in this document thus covers expression of things that are the case, or that may, can or shall be the case, and that the expressions are communicated with any of the possible intentions, such as statements, questions, answers, promises, etc.

3.2 Expression capabilities of formalized languages

The expression capabilities of a formalized language are mainly determined by three components:

- The number of different kinds of things that can be 'said'. In other words: the number of different kinds of expressions (sentences) that can be made.

- The amount of information about the context in which the things are said.
- The extent of the vocabulary that is available in the language.

It appears that natural languages can be formalized because of the following observations and experience:

- Ideas as well as questions can be formulated in the form of networks of relations between things.
- Relations can be classified by kinds of relations, whereas it appears possible to predefine a collection of (standardized) kinds of relations that provide a rich semantic expression capability.
- Meaning of expressions is mainly determined by the definitions of the kinds of relations and the definitions of the related things and the intention with which the expressions are made.
- Each relation has a context that can be expressed as a collection of contextual facts.

With the above observations in mind, the expression capabilities of a formalized language are mainly determined by the following three components:

- The variety of kinds of relations that are defined as part of the language definition (the language defining ontology).
- The richness of the number and kind of contextual facts that express the context for interpretation.
- The number and richness of concepts and their vocabulary that are defined in the taxonomic dictionary.

The kinds of relations for the Gellish family of formalized languages are described in this book. Definitions, terms and phrases that denote those standard kinds of relations are provided in the Upper Ontology section (base ontology) of the Gellish Formalized English Taxonomic Dictionary-Ontology [Ref. 4]. Additional definitions of kinds of higher order relations are provided in the mathematics and activities and processes domains of that dictionary.

This book addresses mainly the components:

- kinds of relations,
- the kinds of roles that are required by those kinds of relations,
- the kinds of objects that by definition have such roles (role players)
- the kinds of contextual facts that can express the context of expressions

The vocabulary of the formalized language is formed by further subtypes of the concepts and are not discussed in this book, but can be found in the dictionary. This book is intended mainly as a clarification on why and how ideas and questions can be expressed in a formalized language, using the available kinds of relations. It is especially intended to clarify the logic that helps finding the proper kinds of relations for making expressions. Once a kind of relation is found, the definition of the relation, as provided in the dictionary-ontology, will further specify the required roles and allowed roles players in such a relation. This document also describes what can be done in case a new kind of relation or a new concept seems to be required.

The way in which the interpretation context is expressed and corresponding kinds of contextual facts are defined is described in chapter 13.

3.3 Models of expressions of ideas

To facilitate users in making expressions in a formalized language and thus in building semantic models, we intent to develop generic patterns for expressions in semantic models. Each pattern should be a pattern for a collection of expressions that is minimally required to express a unit of communication, thus making a statement or expressing an idea about any topic. We define a unit of communication as an expression of an idea about a topic that comprises basically only one relation. Such a relation may be composed of atomic and elementary relations.

Before we discuss those patterns, we first need to discuss how to express an idea about a topic.

Topics are possible states or possibilities in a real or imaginary world. Any topic may be the subject in an expression of an idea about the topic. Consider the following topic about something in the real world:

'whether the Euromast is located in Rotterdam'. (1)

A semantic analysis of this topic will result in the conclusion that it is about a possible state, expressed as relation of a particular kind, which kind of relation is defined as a relation between two single individual objects. This conclusion is reflected and captured in the classification of the things and the classification of the relation. From the expression we can infer that 'Euromast' refers to (denotes) a possible object in the real world.

The meaning of this expression thus is:

- Probably there exists something called 'Euromast', and
- There is some 'predicate'[2] about it, namely the possibility of its 'being located in Rotterdam'.

The second part of expression (1) allocates a 'predicate' to the referred object [Searle, Ref. 5]. Our interpretation of the predicate 'being located in Rotterdam' reveals that it contains a reference to another possible object, which suggests:

- Probably there exists something called 'Rotterdam'.

Finally the predicate contains the phrase 'is located in' which must be the specification of the reason why the two referred possible objects are related according to the expression. The phrase 'is located in' is a phrase that we can recognize as a repeating pattern in similar expressions about things that 'are located in' other things. This 'standard' phrase apparently denotes a general concept or idea of 'being located in something'. The phrase does not directly denote an individual relation, but it is a general phrase for a kind of relation

2 A predicate is sometimes called a property. However, we will use the term property only, with the connotation of some kind of facet and ownership, for quantifiable aspects that are intrinsic to a possessor of the aspect. For example length, color, etc.

that is used in expressions about individual relations between pairs of individual things. This means that it is a kind of relation that is used for classification of the individual relations between two referred individual objects.

This leads us to the conclusion that an idea is usually expressed by the intention 'whether', followed by a sequence of three terms or phrases. For the above example this expression is:

- whether the Euromast is located in Rotterdam

This means that the model of Table 3 can be extended into the model of Table 8.

Expression id	Creator	Addressee	Date-time of creation	Intention	First referred object	Kind of relation	Second referred object
1	John	Mary	t_1	question	the Euromast	is located in	Rotterdam
2	Mary	John	t_2	statement	the Euromast	is located in	Rotterdam

Table 8, Model of an expression with some contextual facts

Linguistic analysis of phrases, such as 'is located in', will conventionally start from the grammatical components, being the words, such as 'is', 'located' and 'in'. This usually leads to a discussion of the roles of categories of words, such as verbs, like 'being' and 'locating', etc. as basic concepts. However, our semantic analysis is searching for semantic concepts[3] that represent distinct meanings. Such a semantic concept is primarily a concept in people's minds that is used in thinking about the real world or about an imaginary world. Our hypothesis is that concepts, such as 'being located in', are generic semantic concepts in human thinking. If that

3 Semantics deal with concepts rather than words, because the concept 'word' is a grammatical concept and not a semantic concept. Words may denote anything and they are natural language dependent. The concept 'verb' is not very helpful either as a semantic concept as it may indicate a static relation as well as a dynamic occurrence. Furthermore, a kind of activity may be denoted by a verb (e.g. in W1 is classified as walking) as well as by a noun (e.g. W1 is classified as a walk), whereas those classification express practically the same meaning.

is the case, then it is basically irrelevant how such concepts are expressed in various languages. The concept itself is language independent. Only the grammar of the language and the available vocabulary determine whether the concept is denoted by a sequence of three or four words in English or by one or more character in Japanese (written Japanese only recognizes characters and no words) or in whatever way.

Semantic analysis has the task to reveal the variety of semantic concepts that are represented by phrases that are in use to express the various kinds of states of affairs in the real and imaginary world. For example, it should clarify what the relation is between concepts such as 'being located in something' and concept 'being located somewhere'. The identification and definition of those semantic concepts should result in a dictionary of semantic concepts that are used to express why referred objects are related according to expressions. The availability and maintenance of such a dictionary is an essential component of semantic modeling.

Expressions of ideas

Expressions of ideas in a formalized language have the following characteristics:

- They contain kinds of relations and their denotations by terms or phrases that have unambiguous definitions.

Those kinds of relations:

- Define the nature of the relations that relate concepts and individual things in syntactic structures by classifying those relations
- Define the roles that are by definition involved in those relations
- Define the requirements for role players in those relations.

3.4 On denoting

The above analysis of the topic 'whether the Euromast is located in Rotterdam' was basically an analysis of various expressions (1-5). From that analysis we concluded that people apparently use a generic concept of 'being located in' something, in a generic pattern

to create expressions about individual things that are located in other individual things.

As said before, semantic modeling typically starts with a state of affairs in the real world or in an imaginary world. A basic assumption of semantic modeling is:

> human beings share notions of individual things and concepts (kinds of things), although they may denote them with different terms or phrases.

In other words: human beings generally recognize the same individual things and concepts in the real world or if they create things in a realistic imaginary world, then those things can be recognized by others.
When we want to communicate about the shared recognized things and concepts, we cannot use the things and concepts themselves, unless the things are present so that we can point to the things. Therefore, instead of the things themselves, we should use shared unique identifiers (UIDs) that unambiguously refer to the things and concepts, so that those identifiers can be arranged in syntactic structures to create universal expressions. Then we can replace the UIDs by terms and phrases with which we denote the things and concepts in various languages to make natural language expressions.

We will illustrate this process by modeling a real world situation (state) and derive the semantic expression from that. Assume that the Japanese Okada was on holidays and crossed a border of what appears to be a city, where he sees a sign with the name of the city ‘Rotterdam’. Thus, since then he is aware of an individual thing (UID 1) which he denotes as ‘Rotterdam’ and which he classifies as ‘city’. Then he saw a building from which he took the picture as shown (UID 2):

Then Okada was told by a guide that the building is called ‘the Euromast’ (UID 3).

When Okada came home in Japan, he documented his trip for his wife Oni with the following statements:

the Euromast	is located in	Rotterdam

This information is related to his knowledge about these objects, such as:

the Euromast	is represented on	picture-1
picture-1	is classified as a	photo
the Euromast	is classified as a	building
Rotterdam	is classified as a	city
Rotterdam	is located in	Netherlands

According to our assumption, human beings are familiar with shared concepts, so that we can allocate universal UIDs to them. Examples of such shared concepts are: ‘photo’ (4), ‘being represented on’ (5), ‘being located in something’ (6), ‘city’ (7), ‘building’ (8) and ‘classification’ (9), ‘Netherlands’ (10). (Note: these UIDs are not official Gellish UIDs.)

When the English terms and phrases are replaced by the UIDs to denote the generally shared things and concepts, then the expressions become as is presented in the following table:

3	6	1
3	5	2
2	9	4
3	9	8
1	9	7
1	6	10

This table is a language independent representation of the expressions, which can be presented to a user in Japanese as well as in any language, provided that a dictionary is available that links numbers that represent concepts and individual things to terms.

3.5 Syntactic and semantic patterns

A syntactic structure is a linguistic structure, which means that it is a structure of spoken or written or computer encoded communication.

For the purpose of this document we ignore other kinds of communication, such as body language.

> *Definition:* A syntactic structure is a linguistic structure in which linguistic representatives of concepts are or may be arranged in order to express meaning.

A human being or computer can only communicate about ideas about what can be the case in practice by using representatives of the things about which is communicated. Such representatives of concepts are typically linguistic components that are also called terms, such as words (lexemes) and phrases, including also abbreviations, codes and names. For example, the fact that the Eiffel tower is located in Paris, can be communicated only by expressing that idea using words and phrases that are representatives of the concepts Eiffel tower and Paris, but also a representative of the idea of 'being located somewhere'.

Syntactic structures are studied extensively by linguists. See for example Noam Chomsky's Syntactic Structures (Ref. 2). Every language has its own Grammar, which consists of the allowed structures for its expressions (syntactic rules) and its terminology (lexicon), and they are all different, unfortunately. But the meanings, the semantics that are expressed in those languages are language independent. So, there is one meaning and many possible expressions.

Valid expressions in a particular language shall be syntactically correct, semantically correct and shall express the intention of the originator of the expression.

An expression in a particular language is syntactically correct when it consist of terms that are arranged in positions (slots) in syntactic structures that are valid for that language, whereas each term represents a concept which kind is allowed for the slot it occupies. Syntactic rules describe which kind of concept is allowed for which position in a syntactic structure of a particular language. These rules indirectly determine which terms may appear in which position in a particular structure.

For example, assume that an artificial language L is defined by the simple syntactic structure for expressions in L that consists of an Object–RelationType–Object (O-RT-O) structure. Assume that L is also defined by the rules (constraints) that any linguistic representative of anything may take an O position and only a linguistic representative of a kind of relation may take the RT position in an expression. Then every expression that consists of a sequence of a term, a kind of relation and another term, is a syntactically correct expression in L.
Assume further that the definition of L is extended with the additional grammatical rule that only linguistic representative terms and phrases may be used that are selected from an English Lexicon. Then the expression 'the Eiffel tower <is located in> Paris' is syntactically and grammatically correct in L.
However, also 'John <is author of> Paris' is syntactically and grammatically correct in L. Thus the grammatical rules allow for nonsense expressions as well as sensible expressions.

The syntactic and grammatical rules are not suitable to determine whether the expressions are *semantically* correct. This requires semantic rules and expressed knowledge.

Semantic patterns consist of syntactic structures that are accompanied by additional semantic rules about the kinds of roles that are by definition involved in relations of various kinds and about kinds of players of those roles in those relations.

These semantic rules can only be adhered to and verified when the nature of the related things and the roles they play are known. The nature of some thing is expressed by the category or kind to which the thing belongs. The relation between some thing and such a category or kind is called a classification relation. Therefore, the related things and the roles shall be classified explicitly by the kinds of things and kinds of roles respectively in order to record the nature of the things.

An expression in a language L is semantically correct when each thing (object) that is related to another thing in a syntactic structure is of a kind that complies with the allowed kind of role player that is

defined for the kind of role that is by definition involved in the relation of the specified kind.

For example, assume that the definition of language L is extended with the semantic rule (constraint) that a kind of relation that is denoted by the phrase <is author of> by definition involves a first role of the kind 'author' and a second role of the kind 'written' and that the role 'author' may be played by an individual thing of the kind 'person' whereas the role 'written' may be played by a thing that is a qualitative document. Then the above expression John <is author of> Paris can be semantically verified. If Paris is classified as a city, then the verification will report that the expression is semantically incorrect, because city is not a subtype of document.

3.6 Semantic principles

Semantic modeling is based on a number of principles. The main ones are described below. These principles are derived from the theory that is developed in the book 'Formalized Natural Languages' [Ref. 1].

Semantic principle 1: Elementary binary ideas

Every idea can be decomposed into one or more *elementary binary* ideas. This holds also for second order[4] ideas as well as for higher order ideas (ideas in which more than two things play a role), which can also be composed of a collection of elementary binary ideas.

Semantic principle 2: Elementary binary relations

An elementary or second order idea can be expressed by and unambiguously interpreted in a known context from the meaning of an expression in the form of a relation between two things (a binary relation) and one or two explicit classifications of that relation. These classifications shall relate the relation with an earlier defined (standardized) kind of relation.

4 First order ideas, such as 'John walks', are always expressed in Gellish as binary relations. For example as John <is involved in> or <is performer of> walking.

Semantic principle 3: Ideas and intentions

Communication requires the expression of an idea together with the expression of a communicative intention for that idea.

Semantic principle 4: Expression of context – main idea and contextual facts

The interpretation of an idea requires not only expression of the idea as such, but also information about the context in which the idea is expressed. That contextual information requires that each main idea is accompanied by the expressions of a number of contextual facts.

> Because of the above principles, each line in an Expression table contains the expression of one "main" atomic idea and various contextual facts.

Semantic principle 5: Unique identifiers

An idea is represented by a relation between unique identifiers (UIDs), which are representatives of things themselves, and not between *names* of things. The reason for that is that names of things are ambiguous, because of the phenomenon of synonyms, homonyms, codes, abbreviations, translations, etc. The related things as well as the relations and both their kinds are therefore represented in the formalized language by UIDs.

Semantic principle 6: Taxonomy of concepts

Each concept (including also kinds of relations) shall be defined as a subtype of another (supertype) concept.

> The principles 5 and 6 together make that each idea is classified by a kind of relation that is a subtype of 'relation' (UID 2850).

Semantic principle 7: Kinds of relations

The definition of the formalized language shall include all the semantics that is necessary for the unambiguous interpretation of expressions, which includes kinds of relations. In other words, semantic expressions can be interpreted by a computer in an unambiguous way when the meaning of kinds of relations, such as

<is classified as> and <is a kind of> as well as the meaning of the other concepts, such as ‘pump’ and ‘capacity’, are denoted by a unique identifier (UID) and are predefined in an electronic taxonomic dictionary (the Lexicon of the formalized language).

Semantic principle 8: Extensibility

The semantic language shall be extensible dynamically with the addition of new concepts, including also new kinds of relations, by adding specialization relations, which incorporate the new concepts in the taxonomy.

With such extension capabilities it is possible that individual objects are even classified by new concepts that do not yet exist in the taxonomic dictionary. For example, the individual object P-1 can be classified as a bicycle pump, by also adding a definition of the concept bicycle pump as an ad-hoc extension of the taxonomic dictionary. This makes that a receiver system can unambiguously interpret even such kinds of ideas.

3.7 Correct formalized language expressions

A formalized language comprises a collection of formalized language expressions, presented in a Gellish Expression Format or an equivalent format. Correct formalized language expressions are expressions that comply with the following rules and guidelines.

Rule 1: Each expression of a main idea relates only individual things and/or kinds of things (concepts or classes) that are:

- Either selected from the formalized language dictionary,

  or are

- Properly defined subtypes of those concepts,

  or are

- Individual things that are properly classified by those kinds of things or their subtypes (via individual classification relations),

  whereas the used individual things and kinds of things are related by individual relations that are classified by kinds of

relations (relation types) that are also selected from the formalized language dictionary.

Rule 2: A subtype of a concept is properly defined if the definition satisfies the requirements that are expressed in chapter 9, which describes how a formalized language dictionary can be developed and extended.

Rule 3: An individual thing is properly classified if it has at least one classification relation with a kind of thing that is selected from the formalized language dictionary or its subtypes.

Rule 4: Any relation shall relate things that play roles of kinds that are required by the kind of relation that classifies the relation and that are defined in the Upper Ontology section of the Dictionary.
Example 1: assume that A is a performer of B. This implies that A shall be a physical object, because the 'is a performer of' kind of relation requires a first role (performer) that can only be played by a physical object. N.B. The second role (performed) is a role that can only be played by an occurrence. So B must be an occurrence.
Example 2: assume that C is a kind of D. This implies that C is a concept, because the <is a kind of> relation requires a first role that is a subtype, which can only be played by a concept (or subtype of concept).

Rule 5: A relation between an aspect and a number may be classified by a scale, in which case the qualitative scale (unit of measure) shall be selected from the formalized language dictionary or from an extension that is defined in correct Formal English.

Rule 6: Concepts and individual things shall have UIDs in the correct range and names of them (being terms or phrases to denote them) shall only be used in expressions when they are allocated in a defining statement (classification or specialization relation) or in an alias relation (or one of its subtypes, such as a synonym relation).

3.8 Expression of meaning

Meaning can be expressed in various natural languages. That is done in expressions or sentences that should satisfy the rules of the language. Such an expression or sentence consists of components ('things') that are denoted by terms and that are arranged in a linguistic structure. Therefore, an expression or sentence can be regarded as a relation that involves (relates) one or more related 'things', whereas each 'thing' has its own role in the relation, which role determines its position in the structure.

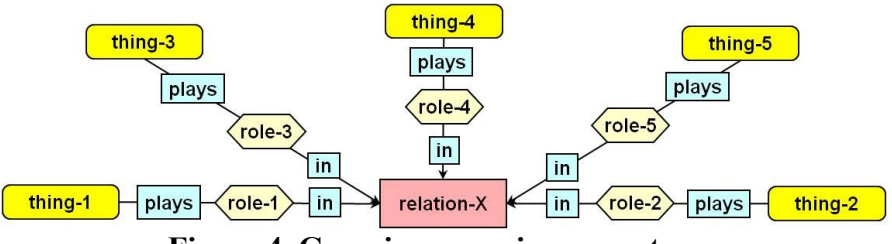


Figure 4, Generic expression or sentence

Meaning can also be expressed in formalized languages. Formalization language such as into Formal English or Formal Dutch (Formeel Nederlands) is based on a standardization of the structure for expressions and sentences and on the sole use of explicitly defined concepts and defined individual things as elements in the expression. This mean that expressions should use either concepts from a formal dictionary or should be defined by a language user conform the rules of the formalized language.

A generally usable formalized language should be an open language, which means that it should include a mechanism to define new concepts 'on the fly' and to add those concepts to the language and immediately use them. Gellish Formal English is such an open formalized language.

The standardized structure of expressions enables to present expressions in tabular form, whereas such tables are suitable for databases as well as exchange messages. Formal English can thus use a single generic Expression table that is suitable for containing any collection of relations that represents any expression or sentence that fits a fundamental semantic pattern as discussed above. Figure 4

illustrates those structures, whereas Table 9 illustrates how those structures can be represented in tabular form. Table 9 shows the core of an Expression table, filled with an example of the expression of a higher order (5th order) relation. It also illustrates that an individual relation (relation-X) need to be classified explicitly in order to enable proper interpretation.

Left hand	Left hand kind of role	Elementary kind of relation	Right hand kind of role	Right hand
thing-1	player	plays role of kind-1 in	involver	relation-X
thing-2	player	plays role of kind-2 in	involver	relation-X
thing-3	player	plays role of kind-3 in	involver	relation-X
thing-4	player	plays role of kind-4 in	involver	relation-X
thing-5	player	plays role of kind-5 in	involver	relation-X
relation-X	classified	is classified as a	classifier	kind of relation Y

Table 9, Tabular representation of a relation

For binary relations it is possible to simplify the tabular expression by combining the two lines and the classification line in one row in the same table (without loss of explicit meaning). Table 10 is an Expression table in which the first row shows in general how a binary relation is expressed, and the second row provides an example of a binary relation.

Left hand	Left hand kind of role	Kind of relation	Right hand kind of role	Right hand
thing-1	role of kind-1	kind of relation Y	role of kind-2	thing-2
thing-1	part	is a part of	whole	thing-2

Table 10, Tabular representation of a binary relation

Note that the kind of relation on the second row in Table 10 is denoted by a phrase <is a part of>, which phrase represents a composition relation.

The definition of a kind of relation specifies the kinds of roles that are required for such a relation as well as the allowed kinds of role players. For example the definition of a composition relation includes that it requires two roles of different kinds: a part and a whole, whereas the definition also includes that each role may be

played by any individual thing. Such definitions of kinds of relations therefore enable that for classified relation it becomes possible that the kinds of roles can be derived from the definition of the kinds of relations. Because of that the kinds of roles do not need to be repeated for every usage of a kind of relation.

Binary relations

Thus the definitions of kinds of relations include the definitions of the kinds of roles. This fact makes it possible that the representation of binary relations can be further simplified by eliminating the explicit kinds of roles. Table 11 is an Expression table in which the first row shows a simplified way of expressing a binary relation in general, and the second row provides an example of a simplified expression of a binary relation.

Left hand	Left hand kind of role	Kind of relation	Right hand kind of role	Right hand
thing-1		kind of relation Y		thing-2
thing-1		is a part of		thing-2

Table 11, Tabular representation of a binary relation with implied roles

The definition of kinds of relations also include the specification of the allowed kinds of role players. This fact enables software to verify whether the related things (or their classifiers) satisfy the requirements for role players (using the taxonomy hierarchy). In that way the software can verify whether an expression is semantically allowed.

The above illustrates why a basic assumption of Formal English is that meaning can be expressed as a collection of relations between things.
To provide sufficient information for a correct interpretation of such a collection of relations it is required that is explicitly specified

- For each relation it is explicitly specified of which defined kind of relation it is, and

- For each related thing it is explicitly specified of which defined kind it is, and
- For each role that is played by a related thing it is explicitly specified of which defined kind it is.

Furthermore, such a collection of relations shall consist of a minimum number of relations of particular kinds. Requirements for collections of relations are further described in the documentation of the semantic patterns that are discussed in the next chapter. Finally, each relation requires the explicit expression of a context. Such a context can be expressed as a collection of ‘contextual facts’. Those kinds of contextual facts are defined in par. 13.4.

4 Universal semantic patterns

This chapter is about universal semantic patterns for the expression of ideas of any kind. The structures specify what is minimally required to be expressed in order to enable interpretation of the meaning of basic units of communication. This minimum comprises the expression main ideas and the expression of the context of the expressions. The semantic patterns for the expression of main ideas are discussed in this chapter. The expression of context is discussed in chapter 13.

The semantic patterns are natural language independent, although the terms that may be arranged in those structures are taken from natural languages.

4.1 Expressions of ideas by binary relations

The way in which meaning is modeled in a semantic model builds on the principle that semantic expressions are idea oriented (or relation oriented although it may be implemented in an object oriented database). Knowledge as well as information about individual things is modeled as expressions of ideas.

Expressions of ideas are composed of relations of particular kinds between related things, each of which playing a role of a particular kind in the relation.

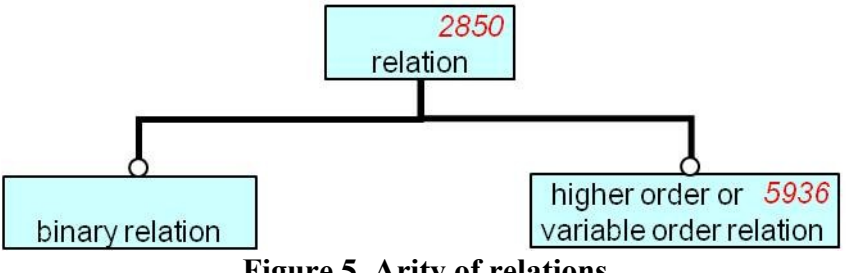


Figure 5, Arity of relations

A relation in principle can relate any number of related things (role players), each with its own role of a particular kind in the relation. The number of roles (and role players) is called the rank or 'arity' of the relation. For many kinds of relations their rank or 'arity' is fixed,

but some relations have a variable arity. Relations with a variable arity are for example interactions of multiple things in activities and processes, because the number of things that participate may be increasing or decreasing during the activity or process. Figure 5 illustrates that kinds of relations can be distinguished by their arity.

An example of an expression (or sentence) that is modeled by one higher order relation, thus with multiple roles and thus multiple role players, is given in Figure 6.

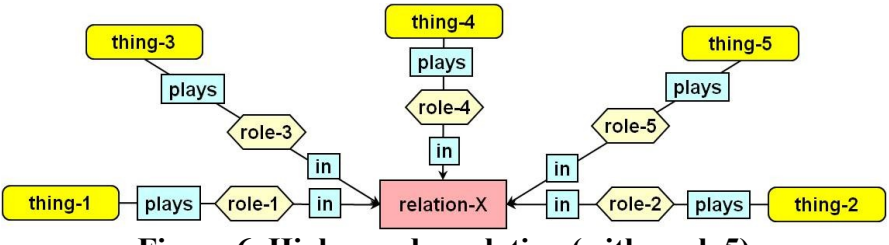


Figure 6, Higher order relation (with rank 5)

Such a relation-X can be expressed in tabular form as in Table 12:

thing-1	**plays**	**role-1**	**in**	**relation-X**
thing-2	**plays**	**role-2**	**in**	**relation-X**
thing-3	**plays**	**role-3**	**in**	**relation-X**
thing-4	**plays**	**role-4**	**in**	**relation-X**
thing-5	**plays**	**role-5**	**in**	**relation-X**

Table 12, Tabular form of a higher order relation

This illustrates that relations of any rank n can be expressed by a collection of n elementary expressions, each of which consisting of two elementary (level 1) binary relations. The general form of each of the elementary expressions is given in Figure 7.

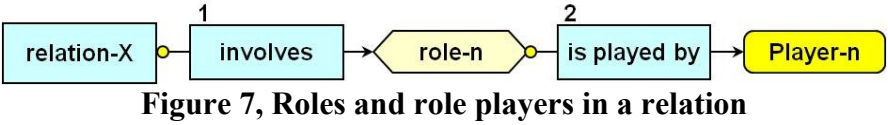


Figure 7, Roles and role players in a relation

The same content can be expressed by an inverse expression as is illustrated in Figure 8.

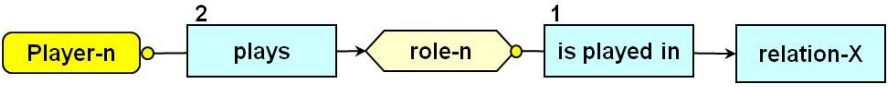


Figure 8, Inverse generic expression by two elementary relations

Relation 1 and relation 2 in the elementary expression and the inverse elementary expression are the same relations, although they are denoted by different terms or phrases. Those differences are caused by the natural language conventions in case of a difference in reading direction. For a semantic model this reading direction is of secondary importance and semantically such different terms and phrases can be regarded as equivalent.

Figure 7 and Figure 8 illustrate the position of roles and role players in a semantic model.

Table 13 expresses in a generalized tabular form that each of the above elementary expression is composed of two elementary level 1 binary relations.

1.	**relation-X**	**involves**	**role-1**
2.	**role-1**	**is played by**	**player-1**

Table 13, Generic elementary relations (level 1)

The two elementary relations specify how a relation is related to a role of a role player and how such a role is related to a role player.

For example, a particular project can be represented by a higher order relation (Project-X), which involves various roles, one role of being the manager and several other roles of being a participant. The fact that somebody is the manager of the project can be expressed by an elementary relation as follows:

Project-X involves role-1 is played by John

Whereas in this expression role-1 is classified as follows:

role-1 is classified as a manager.

The model of Figure 7 and

can be simplified, while maintaining the semantic richness. This can be done by using an equivalent *atomic* level 2 kind of relation between the role player and the relation. Such a level 2 kind of relation is only equivalent when it is defined such that it implies by definition the existence of a role and its classification, as well as the two elementary relations ('involves' and 'is played by') in which the role is involved. The general form of a phrase that denotes such a level 2 relation is <is playing a role of kind-1 in>. This results in expressions at level 2, which have as general form:

Player-1 is playing a role of kind-1 in relation-X

In natural languages we shorten such phrases, while maintaining their meaning, by simply saying for example:

John is manager of project-1 (level 2)

The explicitly modeled definition of such a level 2 kind of relation consists basically of six statements. Table 14 presents a pattern for such a definition.

Name of left hand object	**Name of kind of relation**	**Name of right hand object**	**Definition of left hand object**
relation type-1	has by definition as first role a	role of kind-1	
relation type-1	has by definition as second role a	role of kind-2	
role of kind-1	is (defined as) a kind of	role	is a role that….
role of kind-2	is (defined as) a kind of	role	is a role that….
role of kind-1	can be played by a	kind-1	
role of kind-2	can be played by a	kind-2	

Table 14, Pattern for the definition of a kind of relation

An example of the definition of the <is manager of> relation is:

is manager of	has by definition as first role a	managed	
is manager of	has by definition as second role a	manager	
managed	is (defined as) a kind of	role	is a role that….
manager	is (defined as) a kind of	role	is a role that….
managed	can be played by a	activity	
manager	can be played by a	person	

Table 15, Example of a definition of a kind of relation

Based on this definition a computer should be able to deduce from expression (2) that John plays a role as manager in project-1.
Furthermore, when John is classified as a man or as a person and Project-X is classified as a project or as an activity, then a computer should be able to verify whether the role players are of the correct type. This means that the computer can perform a verification of the semantic correctness of such expressions.

Thus in summary: relations with any number of involvements of role players with their roles of particular kinds can be expressed in a semantic model as a collection of elementary binary relations (level 1). This makes that any relation (binary as well as a variable order or higher order relations) can be represented by a collection of elementary binary relations. Each elementary binary relation expresses that a related thing plays a particular role in the relation. Thus there are as many elementary binary relations as there are roles played in a relation. This makes that (collections of) elementary binary relations are a sufficient basis for the expression of any kind of idea.
Furthermore, pairs of elementary binary relations can be replaced by atomic binary relations (level 2) that are defined by the explicit definition of implied kinds of roles. Each atomic binary relation specifies the involvement of one thing in the relation. This simplifies the expressions and enables to represent higher order and variable order relations by a collection of binary relations.

Using binary relations also for the expression of variable and higher order relations enabled the development of a standard universal Expression table that is suitable for the recording of any expression

in Formal English. Each Expression table has the same structure (table column definition). That universal structure is defined in chapter 15.

The expression of any idea in a semantic model follows one of the two fundamental universal semantic patterns: one pattern for ideas that represents information about individual things and another pattern for ideas that represents knowledge or general requirements or definitions. Both patterns are explained below.

4.2 Individual things and kinds of things

There is an essential difference between *kinds of things* and *individual things*. Individual things typically have a location in time and space. They include not only individual physical objects in the real world, but also their individual aspects (facets), individual occurrences and individual relations as well as individual realistic (and unrealistic) imaginary things. Kinds of things, also called concepts (or classes of things), are abstract categories that can be used to classify things using criteria for inclusion or exclusion that define the concepts.

To enable computer interpretation it is required that a user of the language defines each new *individual* thing by at least two actions:

- Representing the individual thing uniquely in the formalized language by allocating a UID conform the rules for allocating UIDs and by denoting it by a name.
- Specifying at least one explicit *classification relation* between the UID and a UID of a properly defined kind of thing (concept or qualitative aspect, also called a value) that is defined in a formal taxonomic dictionary.

4.2.1 Classification relations

Such a classification relation adds (the definition of) the individual thing to the vocabulary of Formal English and the statement about the classification should therefore be communicated to other parties when information about that individual thing is exchanged.

Thus, a *classification relation* indicates that something is an individual thing that is classified by the classifying concept or by that qualitative aspect. For example,

P-1	is classified as a	pump
h-1	is classified as a	height

Note that in addition to classifying individual things it is also possible to classify kinds, as will be discussed later.

4.2.2 Specialization and qualitative subtyping

Each new user defined *kind of thing* (or concept) shall also be added to the vocabulary of Formal English. This shall be done by allocating a UID conform the rules and by relating it to at least one other UID of a properly defined concept by a *specialization relation* or by one of its subtype kinds of relations, such as a qualification relation (denoted by the phrase <is a qualitative subtype of> or <is a type of>.

A *specialization relation* indicates that the defined object is a kind of thing that is a subtype of the related supertype concept. The specialization relation is typically denoted by the phrase <is a kind of> or <is a specialization of>. For example,

thermometer	is a kind of	meter
repairing	is a kind of	activity

A *qualification relation* indicates that the defined thing is a type of thing or is a qualitative aspect (also called a value) that is a qualitative or quantitative subtype of an aspect. A qualification relation is typically denoted by the phrase <is a qualitative subtype of> or <is a type of>. For example, a particular number and color can be defined as follows:

3.141592	is a qualitative subtype of	number
red	is a qualitative subtype of	color

Whereas a type of physical object can be defined as follows:

M6 bolt	is a type of	bolt

The use of these types of relations between new defined concepts and the standard Formal English concepts, together with an optional textual description, are required to define the meaning of new user defined concepts.
For example, Table 16 is an Expression table that states that my bicycle pump, called P-1, is defined by a classification relation.

54	2	101	43	1	60	3	15	201	65
Language	**UID of left hand object**	**Name of left hand object**	**Intention**	**UID of idea**	**UID of kind of relation**	**Name of kind of relation**	**UID of right hand object**	**Name of right hand object**	**Partial definition**
English	101	P-1	assertion	201	1225	is classified as a	102	bicycle pump	
English	102	bicycle pump	assertion	202	1146	is a kind of	130206	pump	that is intended to inflate bicycle tires.

Table 16, Description of User Objects in an Expression table.

Assuming that the concept bicycle pump is not present in the taxonomic dictionary, it is added as a user defined concept. This is done by representing it by a UID (102) and by using the specialization relation <is a kind of> to relate it to the existing concept ‘pump’. If the concept bicycle pump would have been available in the dictionary, the second line would be superfluous and the UID should have been taken from the dictionary.

Each line in an Expression table denotes a ‘main atomic idea’, represented by a ‘UID of idea’, and includes an expression of that ‘main atomic idea’ (shortly called ’main idea’), together with a number of contextual facts. The main idea on the first line in Table 16 has UID 201. It denotes the ‘assertion’ that P-1 is classified as a bicycle pump. That ‘idea’ is communicated as an assertion (the intention of the expression). Basically, the idea is expressed as a relation of kind 1225 (a classification relation) between object 101 and object 102. The following contextual facts are shown in Table 16:

- The fact that the expression is in English

- The fact that 101 is called P-1
- The fact that the expression is communicated with the intention of being an assertion
- The fact that kind of relation 1225 is called ‘is classified as a’ (in English)
- Etc.

So, an expression is a representation of an atomic idea, or more precise: of a proposition about an atomic fact in a context, whereas with a atomic idea is meant: something that is the case or might be the case.

The semantics of a line in an Expression table is defined by the kinds of the relations between the columns in the Expression table. The relations between the columns in the table define the main idea and the contextual facts on the line. This is further discussed in par. 13 and 13.4.3.

For example, line 1 in Table 16 expresses the following ideas (in English):

1. User object 101 is an individual object with the name “P-1”.

   The fact that the object is an individual thing is inferred from the kind of relation <is classified as a>, because a classification relation of this kind is defined as being a relation between an individual thing and a kind of thing. This can be inferred from a Gellish formalized language dictionary, which also contains the language definition, because the upper ontology section of that dictionary contains two relations (with UIDs 1.003.840 and 1.003.573 that express the ideas that a classification relation requires two roles, (1) a “classified individual thing” and (2) a “classifier of an individual thing”. It also contains two other relations that express that the classified role can be played by an individual thing and the classifier role can be played by a kind of thing (a class).

2. There is ‘main atomic idea’ with UID 201 that is expressed as a relation between object 101 and object 102.
3. The idea with UID 201 is classified by 1225, being a standard classification relation concept, found in a formal dictionary. This defines the meaning of the main atomic idea, being in this case that idea 201 is qualified as a classification relation.
4. Relation type 1225 is denoted as <is classified as a>.
   Note that this denotation is already specified in a Gellish formalized language dictionary and therefore the kind of relation name is semantically superfluous in a formalized English expression. However, the name is useful for human readability of the expression.

1. User object 102 is a kind of thing and has the name “bicycle pump”.
   It is a general rule in formal English that a name is formally allocated to an object only at the left hand side on a line where the object is defined by a classification, a specialization or a qualification relation, or on a line where the name is defined as an alias (synonym, abbreviation, etc.) or as a translation of an existing object name. Therefore, the fact that object 102 is called “bicycle pump” is formally specified on the next line in the above example Expression table and is referred to on this line only (a verification of consistency between such multiple usage of names is recommended to be done by software).

Line 2 is required to ensure that the right hand term of line 1 is a defined thing. Line 2 in Table 16 defines similar ideas as in line 1. Note the following ideas:

1. User Object 102 is a kind of thing (class) with the name “bicycle pump”.
   The fact that it is a kind of thing was already concluded from the kind of relation in line 1, but can also be concluded from the kind of relation on line 2 (see below)

2. User Object 102 is a specialization of the existing Gellish UID 130.206 with the name “pump”. From a Gellish formalized language dictionary it can be inferred that the specialization relation is a relation with two roles: a subtype and a supertype, each of which is played by a kind of thing So, both the left hand object and the right hand object is a kind of thing, which is consistent with the fact that the right hand object of line 1 is a kind of thing.
3. User Object 102 has an additional relation with the textual definition in column 4, which specifies in natural language text in what respect a bicycle pump distinguishes itself from the general concept of a pump and from its ‘brother’ types of pumps.

Any other idea can be described in Formal English by usage of other kinds of relations. A large variety of available standard kinds of relations are defined in the upper ontology section of a Gellish formalized language dictionary.

4.3 Expression of ideas about individual things

In this paragraph we give a semantic analysis of expressions of ideas about individual things. We start with a statement about how such ideas are expressed in semantic models:

> *In a semantic model any idea that represents information about an individual thing is expressed by a relation of a particular kind together with a specification of the related things and the roles that they play in the relation.*

An example of a statement about an idea that is represented by a single binary relation is:

> assertion: the Eiffel tower <is located in> Paris

Note: Conventional linguistics treat such an expression (sentence) as a 'model of (seven) words', whereas a semantic model treats the expression as a model of (three) concepts. A semantic model is not a model of words, but a model of concepts. Therefore, the phrase 'is located in' as a whole is used as a name of a kind of relation. The

individual words in the phrase are irrelevant, which is opposed to conventional linguistic analysis of natural languages.

In the above example expression, the assertion that the Eiffel Tower is located in Paris is expressed by a location relation that implies two roles. The first one is a role of 'being located'. This role is played by the Eiffel Tower. The second one is a role of 'locator', which role is played by Paris. This is graphically depicted in Figure 9.

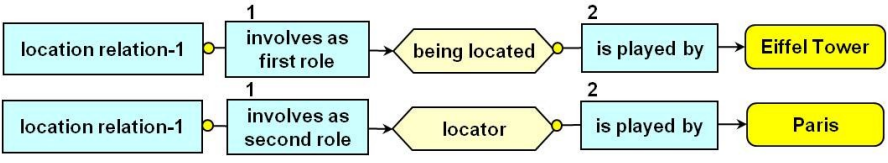


Figure 9, Example relation with two roles and two role players

Thus, such an idea is expressed by two elementary expressions. Each consisting of two (level 1) elementary relations: an involvement relation and an 'is played by' relation.

The meaning of an expression is provided through the specification of the nature of each of the individual components of the expression. Those natures are provided by the definitions of the kinds of things that classify the individual things (and not through the names of the individual things). Therefore it is a requirement that each individual thing is explicitly classified by at least one kind of thing, whereas that classifier shall be defined in the dictionary of the formalized language.

Thus, in order to capture the meaning of an expression, so that a software application can interpret the meaning, it is required to add classification relations that classify the role players, the roles and the relation. For example, in addition to the expressions in Table 12 we should add:

The Eiffel Tower	is classified as a	tower
Role-1	is classified as a	being located
Relation-1	is classified as a	is located at (or 'location relation')

Note that the individual role players typically have a name, whereas their (individual) roles and relations have no name, because their

classification is sufficient for the interpretation of the meaning of expressions.

The combination of level 1 relations and these classification relations in one model results in a fundamental universal semantic pattern comprising five binary relations that are required for expressing the meaning of an idea about an individual thing. That structure is illustrated in Figure 10.

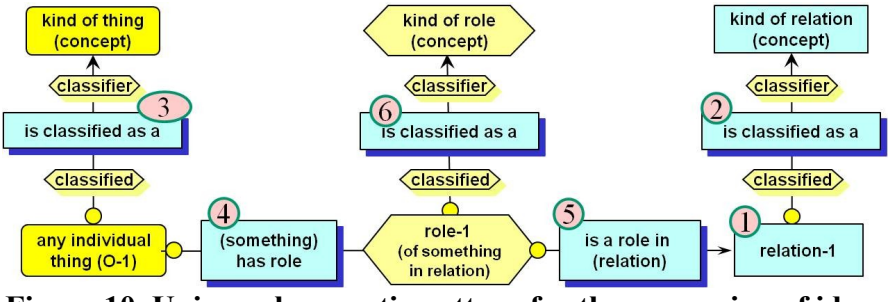


Figure 10, Universal semantic pattern for the expression of ideas about individual things

The five binary relations in such a pattern are:

- The classification of the relation (2)

And for every role in the relation there are four elementary relations, which are:

- The assertion that the relation requires a role (5)
- The classification of that role (6)
- The assertion that the role is played by an individual thing (4)
- The classification of that individual thing (3)

The semantic pattern of Figure 10 illustrates that each idea about individual things can be expressed as a classified relation (1) between the involved classified individual things, whereas each individual thing plays a particular classified role in that relation. According to the pattern, the relation between an individual thing and its role as well as the relation between that role and the relation are expressed by *elementary* (level 1) relations. This can also be

expressed by inverse phrases for those relations. The inverse for <is played by> is <is player of> and the inverse of <is involving> is <is involved in>. Then the expression of the role of the Eiffel tower in the relation (1) becomes:

4	The Eiffel tower	is player of	role-1
5	role-1	is a role in	relation-1

A similar collection of elementary relations can be created for the role of Paris in this relation.

For a correct interpretation of the expression the related objects, the roles as well as the relation need to be classified. This results in the following expressions of ideas:

3	The Eiffel tower	is classified as a	tower
4	role-1	is classified as a	located (object)
2	relation-1	is classified as a	is located in (relation)

The above universal semantic pattern is usually not directly applied, because it can be simplified while still keeping its semantic expression richness. There are two relevant simplifications.

4.3.1 Pattern for relations between individual things

As described in paragraph 4.1 the above pattern of Figure 10 can be simplified for the expression of ideas that require higher order relations, such as correlations and models of occurrences. This can be done by replacing the two binary *elementary* level 1 relations (4) and (5) by one binary *atomic* level 2 relation (7), which is classified by a kind of level 2 relation (8), which leaves the roles implicit. This pattern is illustrated in Figure 11.

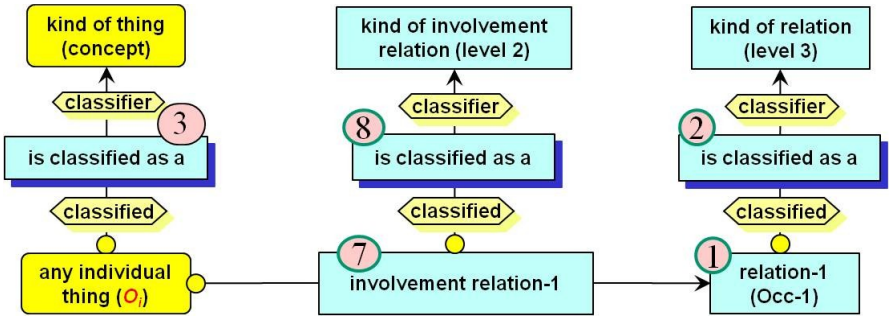


Figure 11, Simplified universal semantic pattern for higher order relations between individual things

The kinds of involvement relations at level 2, as depicted in Figure 11, are defined such that they have by definition implied roles of particular kinds. One of those kinds of roles is the kind of role that is used to classify role-1 in Figure 10.

A tabular representation of this pattern is given in Table 17.

UID of idea	Name of left hand object	Name of kind of relation	Name of right hand object
4	Object-1	is involved (as…) in	Occ-1
2	Occ-1	is classified as a	kind of occurrence
3	Object-1	is classified as a	kind of physical object

Table 17, Pattern for higher order relations between individual things

For example, assume that the pattern of Figure 10 is used to express that Object-1 (O-1) is involved as performer in higher order Relation-1 (representing Occ-1). This means that role-1 in Figure 10 is classified (6) as 'performer'. We can then simplify the pattern by replacing the two elementary relations (4) and (5) by one atomic relation (7), which is classified (8) as a <is performer of> kind of relation. The <is performer of> kind of relation is defined as a relation that requires by definition a first role as 'performer'.

Definitions of such kinds of relations will be expressed by using the second semantic pattern, as will be discussed in paragraph 4.4. For the time being it is sufficient to assume that the definition of such a kind of relation defines which kinds of roles are by definition involved (and which kinds of role players can play such roles). For example, the kind of relation <is performer of> is defined by the requirement to have as its first role a <performer> kind of role and as its second role a <performed> kind of role, whereas the definition also specifies which kinds of things can play roles of such kinds. Thus, the kinds of roles are already defined by the definition of the kind of relation, which makes the classification (6) of the individual roles superfluous.

A semantic interpretation of an expression that uses such a kind of level 2 relation can thus derive the kinds of roles from the definition of the kind of relation and therefore it is neither necessary to make the individual roles explicit, nor to classify them.

Figure 11 presents the simplified universal semantic pattern for expression of relations of any arity. This structure comprises one classification relation that classifies the (higher order) relation (the occurrence) (2) and three relations for every involved role player (3), (7) and (8). This semantic pattern is especially useful for the expression of unary, higher order, or variable order relations, such as occurrences, correlations and if-then-else relations.

When applied for the specification of an activity, this structure leads to new *atomic kind of relations* to distinguish the players of the various roles. For example:

7	John	is performer of	act-1
3	John	is classified as a	inspector

Whereas

2	act-1	is classified as a	inspecting (relation)

And similarly:

7	P-101	is subject of	act-1
3	P-101	is classified as a	pump

The kinds of roles, performer and subject, in the inspection activity can be inferred from the definition of the kinds of relations by which the atomic relations (7) are classified.

Thus, higher order relations, variable order relations and unary relations are expressed as collections of one or more of this kind of classified *atomic binary* relations.

Unary relations

Semantic modeling usually distinguishes between unary, binary and higher order relations. A unary relation is an expression of an idea in which only one object is involved. A binary relation is a relation between two objects, etc.

A typical example of the expression of a unary idea is a statement that expresses that a person is performing an activity of a particular kind. For example: 'John is inspecting'. The semantic model of this example is given above.
However, this atomic relation is not the complete representation of what is occurring. The complete idea would also specify which object is being inspected and possibly also e.g. which tools are used during the inspection. This illustrates that unary relations are usually, if not always, only atomic relations and thus they are incomplete expressions of higher order ideas. Unary relations are typically relations between an object and an activity or process, whereas the object plays a role as performer or as subject.

4.3.2 Pattern for binary relations between individual things

Many ideas appear to be ideas that can be expressed as binary relations, being units of communication in which exactly two things are involved, each with their own role of a particular kind. Such binary relations can be expressed by using a further simplified semantic pattern. This simplification can be achieved because such ideas require always two atomic level 2 relations. Such *pairs of atomic* relations can be replaced by one *molecular* level 3 relation. This simplification results in the simplified universal semantic pattern of Figure 12.

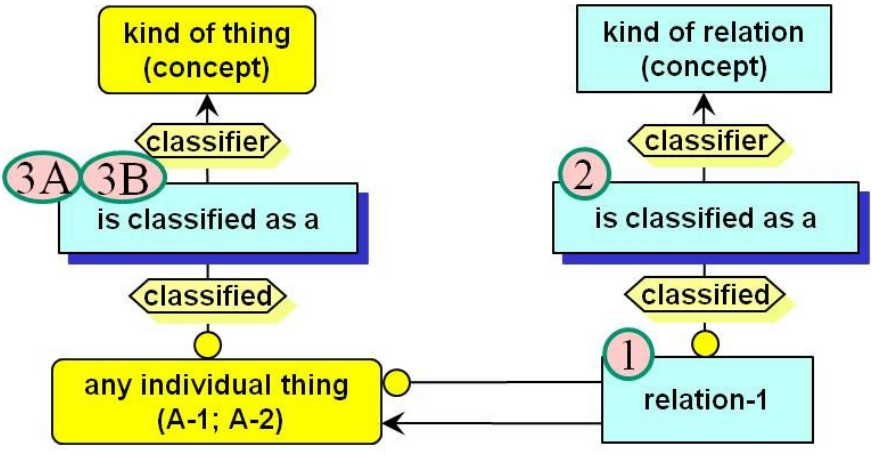


Figure 12, Simplified universal semantic pattern for binary relations between individual things

Figure 12 presents graphically a pattern for the expression of *binary* ideas about individual things. Such an idea is expressed as a binary relation between 'any individual thing' and another individual thing. That second individual thing is represented in Figure 12 by the same box, called 'any individual thing', as the first individual thing. Thus the expression of a complete *molecular* binary idea comprises a specification of two role players and a binary relation at level 3, each with its classification.

Each level 3 kind of binary relation according to the pattern of Figure 12 is defined such that it requires by definition two roles of kinds that are the kinds of roles that classify two roles played by two role players of the pattern of Figure 10.

The pattern of Figure 12 does not define for each of the individual role players which of the two roles it actually plays. The definition of the kind of relation only specifies the kind of role player that may play the first role and the kind of the one that may play the second role. In natural languages the role player that plays the first role and the one that plays the second role is determined by the grammar of the language. However, those grammatical conventions are natural language dependent. According to the English language grammar the roles are determined by the phrase that denotes the kind of

relation and the corresponding sequence of the role players from left to right. The syntax of the formalized language therefore defines which role player is specified to play the first and which one the second role. This is further discussed in chapter 13.

For example, a part-whole relation is a binary relation, with two involved individual things, one with a role as a part and the other with a role as a whole. Those kinds of roles would be explicit when we would use the pattern of Figure 10. According to the pattern of Figure 11 the model would be simplified as:

P-1	has a role as part in	assembly relation-1
W-1	has a role as whole in	assembly relation-1

To enable to directly relate the two involved things, P-1 and W-1, we need to define and use a *molecular* kind of binary relation at level 3 that implies by definition the two atomic involvement relations at level 2 and the four elementary relations and the two roles with their classification at level 1.

For example, we can define a molecular level 3 kind of relation called <is a part of>, which by definition requires two roles, the first one classified as a ‘part’ and the second one classified as a ‘whole’. This enables to express a binary part-whole relation as follows:

P-1 is a part of W-1

The semantics of such a binary expression is included in the definition of the kind of relation, which includes an explicit specification of the kinds of roles that are played by the related role players and the explicit specification of which kinds of things may play roles of such kinds. The latter specification can be used to semantically verify whether the classifiers of the role players (e.g. the classifiers of P-1 and W-1) comply with the allowed kinds of role players.

The definition of such a binary relation uses the same pattern of Table 14 that was developed for the simplification from level 1 into

level 2. For example, by using that pattern, the definition of a part-whole relation becomes as is presented in Table 18:

Name of left hand object	Name of kind of relation	Name of right hand object	Definition of left hand object
is a part of	has by definition as first role a	part	
is a part of	has by definition as second role a	whole	
part	is defined as a kind of	role	is a role that….
whole	is defined as a kind of	role	is a role that….
part	can be played by a	individual thing	
whole	can be played by a	individual thing	

Table 18, Definition of a part-whole relation between individual things

Figure 12 uses one box to represent two related things in a binary relation, just as Figure 11 uses one box to represent any number of related things in a higher order relation. For clarity an alternative representation of the pattern of Figure 12 is given in Figure 13, in which each role player is represented by its own box.

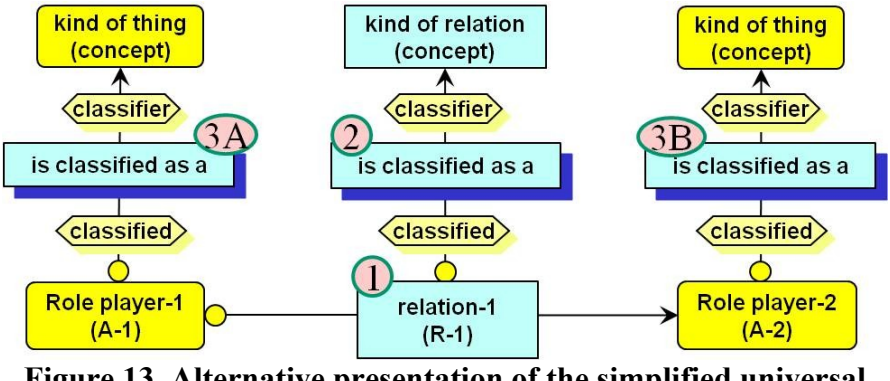


Figure 13, Alternative presentation of the simplified universal semantic pattern for binary relations between individual things

The relation R-1 (1) and its classification (2) can be expressed on one line. Therefore, a general pattern for the expression of binary ideas about individual things becomes:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
1	A-1	(2) is related to	A-2
3A	A-1	is classified as a	kind (1)
3B	A-2	is classified as a	kind (2)

Table 19, Pattern for expressing binary ideas about individual things

For example, assume we want the computer to interpret the expression of a statement about a binary idea, such as:

101 the Eiffel tower is located in Paris

This expression shows only three components: the first and last one, the Eiffel tower and Paris, are both individual things that are represented by the boxes Role player-1 and Role player-2 in Figure 13. The phrase <is located in> represents a kind of relation that classifies (2) the individual relation (1). Thus that kind of relation is an example of a concept at the upper side of the figure. For a complete interpretation it is required that a computer also knows of what kind the two related individual things are. The individual things are defined by their explicit classification relations (3A and 3B). For example this results in the following additional expressions:

102 the Eiffel tower is classified as a building
103 Paris is classified as a city

The roles that are played by the Eiffel tower and Paris are individual roles. The kinds of roles are always the same for all relations of the same kind. Therefore, the kinds of roles that are played in a relation of a particular kind are defined as part of the definition of the kind of relation. This implies that the classification relations for the roles can be deduced from the definition of the kind of relation and don't need to be included explicitly in each expression for binary ideas about individual things. In the above example those roles are 'located' and 'locator' respectively.

The explicit classification of the related individual things enable a computer to verify the semantic correctness of the expressions, because the definition of the kind of relations define which kinds of things may play roles of the required kind. In this example, a computer can conclude that both related thing are individual things, because the definition of a classification relation specifies that the first role is played by an individual thing. Furthermore, the definition of a relation, such as the <is located in> relation, specifies that both role players should be physical objects. This means that the classifiers, being building and city, should both be subtypes of physical object. Whether that semantic requirement is satisfied should be verified via the relations in the taxonomic dictionary of the formalized language.

4.4 Expression of ideas about kinds of things

Any idea that represents knowledge about a kind of things or a general requirement or a concept definition is expressed in a semantic model as a relation between kinds of things. To enable the interpretation of the expression it is insufficient to only record the relation itself, because it is required to include additional relations that define the used concepts.

A concept definition model includes at least one specialization relation with a direct supertype of the concept and a description of the relation(s) that specify an additional constraint with regard to the supertype concept, which constraint distinguishes the subtype from other subtypes of the same supertype. The specialization relation expresses that the defined concept is a proper subtype of that supertype concept. Such a specialization relation means that the defined concept inherits the definition and all other ideas about the supertype concept, including its possible roles in relations (except for its names). The specification of the distinguishing constraint may be specified as free text and/or may be defined by one or more ideas about the defined concept that are by definition the case. The specialization relation is further discussed in par. 9.1

Ideas that represent *knowledge* include knowledge about what *can be* the case as well as knowledge about what is by definition the

case. General *requirements* include states of affairs that *shall be* the case in a particular context. Each of such a category of ideas is expressed by using its own kind of relation.

For example, expressions of knowledge about what is by definition the case are:

(any) house	has by definition as part a	roof
flat roof	is (by definition) a specialization of	roof

The ideas about what is by definition the case only define what the essential characteristics of such a kind of thing is. This means that a thing that does not comply with that essertion is not a thing of that kind. Thus, this definition states that something without a roof is not a (well formed) house (yet).

An example of a general requirement is that *in a particular context* it may be specified as a requirement that:

(any) house	shall have as part a	flat roof

For a correct interpretation of the meaning of 'house', 'roof' and the various kinds of relations (relation types) shall be defined in the formal dictionary.

Thus, because of their definition, each kind of thing has at least one specialization relation with a supertype kind of thing. This means that a subtype-supertype hierarchy of relations between kinds of things is resulting. In other words the concepts are arranged in a Taxonomy hierarchy. This also holds for the kinds of relations. This has additional benefits, because each kind of relation is defined in such a way that it constrains the related objects to objects of particular kinds. For example, only physical objects can be located in physical objects. Therefore, the <is located in> relation requires that each located thing and each locator thing is a physical object. To enable the verification of that constraint it is required that concepts such as 'building' and 'city', etc. all are defined in the Dictionary as subtypes of the concept 'physical object'. Only then the subtype-supertype hierarchy of concepts (the Taxonomy) enables automated verification whether an expression is semantically allowed.

These general requirements on expressions are the cause that a universal semantic pattern is discovered that specifies what is minimally required for the expression of the meaning of a unit of communication about kinds of things. That pattern is presented in Figure 14.

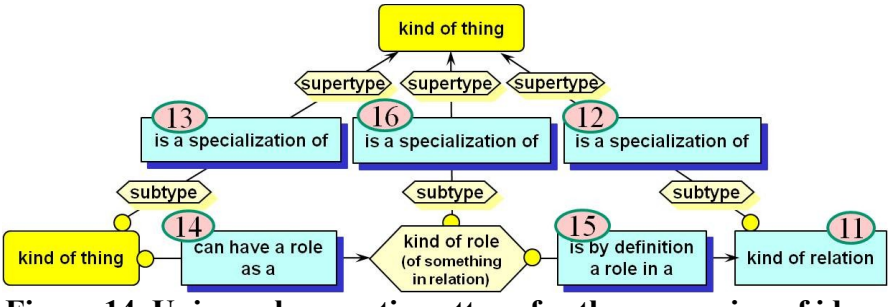


Figure 14, Universal semantic pattern for the expression of ideas about kinds of things

This pattern has similarity with the universal pattern for expressions about individual things as presented in Figure 10, but the concepts and the kinds of relations are different. The semantic pattern of Figure 14 illustrates how ideas that represents knowledge or general requirements about a kind of thing are expressed. A complete expression comprises:

- A kind of relation (11) and its definition as specialization of a supertype kind of relation (12)

And for each involved kind of thing it comprises:

- A pair of elementary relations (14) and (15) that the particular kind of thing is involved in a role of a particular kind
- Two a specialization relations, one to define the nature of the kind of thing (13) and another to define the kind of role (16).

This pattern is used for the definition of kinds of relations. The definition of a kind of relation includes:

- A specification of the kinds of roles that are by definition played in such a kind of relation.
- For each kind of role a specification of the kinds of things that can be player of such a kind of role.

The pattern for the definition of a kind of relation with a specification of one of its kinds of roles becomes in tabular form as is presented in Table 20.

UID of idea	Name of left hand object	Name of kind of relation	Name of right hand object
12	kind of relation-1	is a kind of	kind of relation-2
17	kind of relation-1	has by definition as first role a	kind of role-1
18	kind of role-1	is a kind of	kind of role-2
16	kind of role-1	can be played by a	kind-1
13	kind-1	is a kind of	kind-2

Table 20, Patter for the definition of kinds of relations

Each component in an expression is a kind of thing or concept that shall be included in the dictionary of the formalized language; together with its definition (i.e. it shall be included in a Domain Dictionary or in a proprietary extension). As a proper definition of a concept should be computer interpretable, the definition of a concept shall be modeled and may not be just human readable free text.

4.4.1 Pattern for relations between kinds of things

Constraints can be specified on kinds of roles that are played by particular kinds of role players. The specification of such constraints requires making such kinds of roles explicit. When such constraints are not applicable on roles in higher order relations, then for proper interpretation it is sufficient to know the kind of role that can be deducted from the definition of the kind of relation or that is inherited from the definition of a kind of relation that is higher in the taxonomy hierarchy. Therefore for most applications it is not required to make the kind of role explicit. This enables to simplify

the semantic pattern for higher order relations as presented in Figure 15.

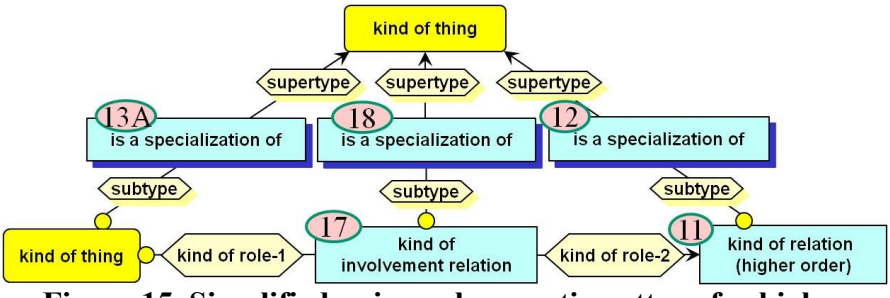


Figure 15, Simplified universal semantic pattern for higher order relations between kinds of things

A tabular representation of this pattern is presented in Table 21.

UID of idea	Name of left hand object	Name of kind of relation	Name of right hand object
14	kind-1	can be involved (as a…) in a	kind of occurrence-1
15	can be involved (as a…) in a	is a kind of	kind of involvement in an occurrence
12	kind of occurrence-1	is a kind of	kind of occurrence-2
13	kind-1	is a kind of	kind-3

Table 21, Pattern for higher order relations between kinds of things

There are many kinds of involvement relation and kinds of occurrences defined in the Dictionary as will be discussed later. They are not only kinds of relation for expressing knowledge, such as <can be a performer of a> kind of relation, but also for the expression of requirements and definitions, such as <shall be performer of a> and <is by definition a performer of a>.

4.4.2 Pattern for binary relations between kinds of things

For binary relations between kinds of things both kinds of roles can be derived from the definition of the kind of binary relation. This enables a further simplification of the semantic pattern as follows:

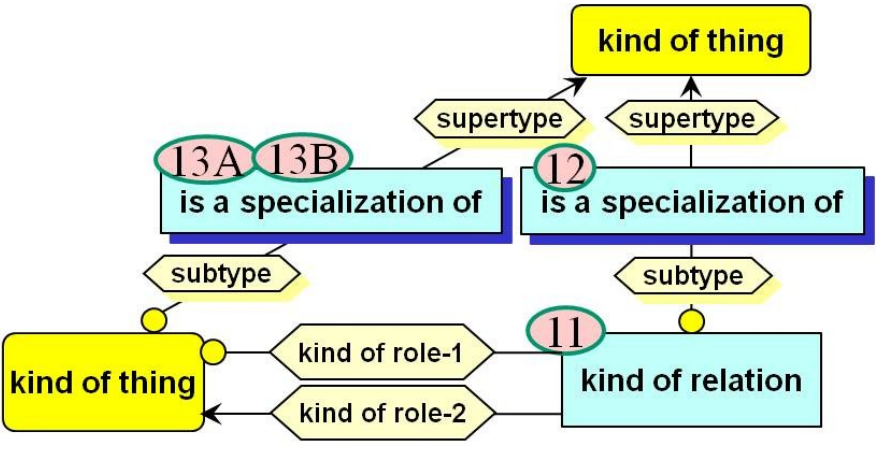


Figure 16, Simplified universal semantic pattern for binary relations between kinds of things

Thus a general expression of a binary relation between two kinds of things becomes:

- Some kind of thing plays a kind of role in a kind of relation in which another kind of role is involved that is played by another kind of thing,

In other words:

- Something of a particular kind plays a role of a particular kind in a relation of a particular kind in which another role of a particular kind is involved that is played by another thing of a particular kind.

The box at the left hand corner of Figure 16 represents two different related kinds of things. For clarity an alternative representation of the pattern of Figure 16 is given in Figure 13, in which each related kind of thing is represented by its own box.

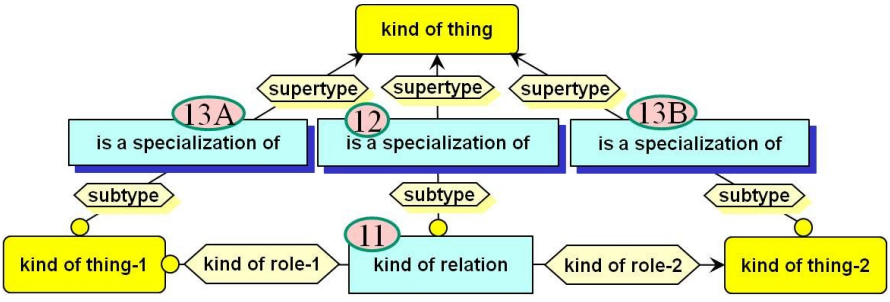


Figure 17, Alternative presentation of the simplified universal semantic pattern for binary relations between kinds of things

A tabular representation of this pattern is presented in Table 22.

UID of idea	Name of left hand object	Name of kind of relation	Name of right hand object
11	kind-1	can be related to a	kind-2
12	can be related to a	is a kind of	binary relation between kinds
13A	kind-1	is a kind of	kind-3
13b	kind-2	is a kind of	kind-4

Table 22, Pattern for binary relations between kinds of things

The various kinds of relations that are defined in the Dictionary are discussed in chapter 9 and 10. Those kinds of relations include kinds of relations to express knowledge, requirements and definitions.

4.5 Integrated semantic patterns

The correct interpretation of expressions about individual things in a formalized language always requires the classification of the individual things, as well as the definition of the classifying concepts by means a taxonomic dictionary of concepts. Therefore it requires the combination of the semantic pattern for the expression of ideas about individual things and the semantic pattern for the expression of ideas about kinds of things. This combination of the universal

semantic patterns of Figure 10 and Figure 14 is presented in Figure 18.

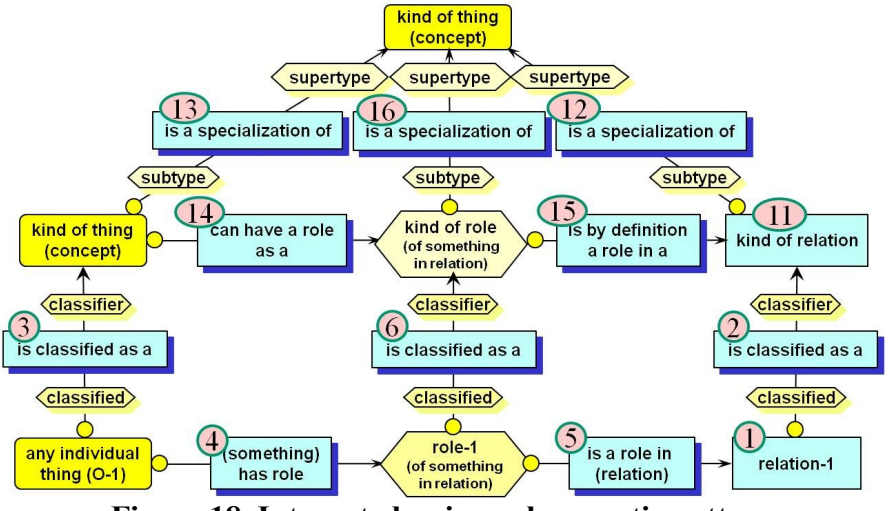


Figure 18, Integrated universal semantic pattern

The lower part of the pattern is universally applicable for the expression of information about individual things; the upper part is applicable for the expression of definitions of concepts as well as for the expression of knowledge and requirements.

The simplified version of the integrated pattern for higher order relations, such as occurrences, correlations and conditional consequence (if-then) relations is given in Figure 19.

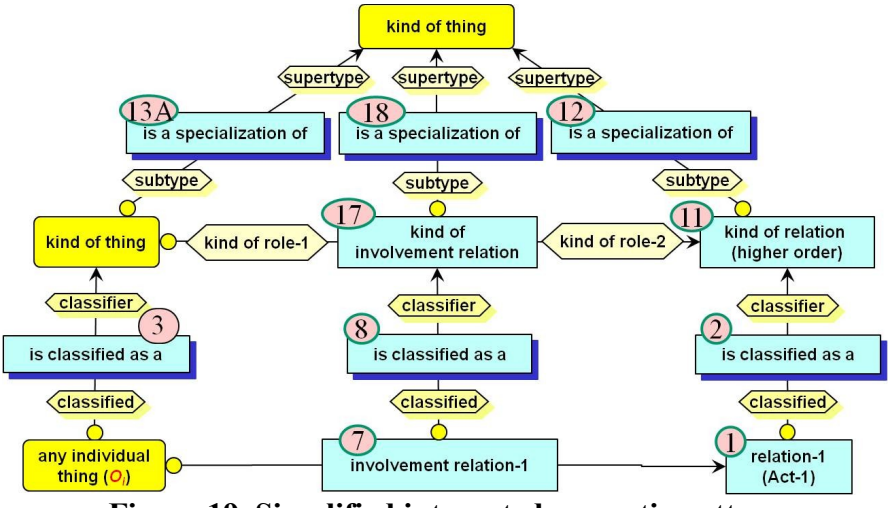


Figure 19, Simplified integrated semantic pattern for higher order relations

The simplified version of the integrated pattern for binary relations is given in Figure 20.

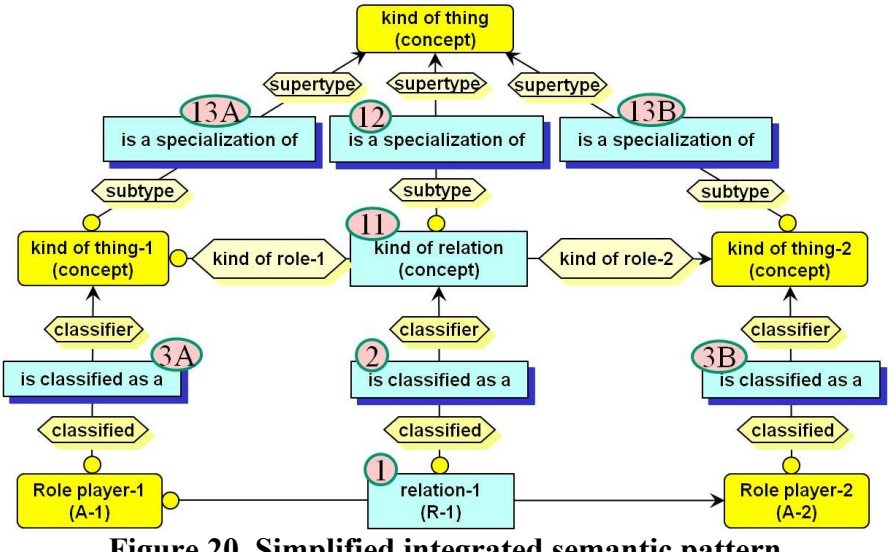


Figure 20, Simplified integrated semantic pattern for binary relations

Note that the term ‘simplified’ indicates that the roles are implicit.

4.6 Bootstrapping the formal definitions

Defining concepts in a formalized language requires the use of already pre-defined concepts. Avoiding circular definitions implies that an initial list of 'bootstrapping' concepts should be available that form the basis for the definition of further concepts. Semantic analysis of formalized language definition resulted in the discovery that formal expressions follow the above described universal semantic patterns. Furthermore it was concluded that the patterns that are required for concept definitions and the allocation of terms to concepts requires only the bootstrapping kinds of relations as given in Table 23.

Name of kind of relation	**Phrase that denotes the kind of relation**
specialization relation	is a kind of
by definition having a first role	has by definition as first role a
by definition having a second role	has by definition as second role a
possibly playing a role	can play a role as a
naming relation	is called

Table 23, Bootstrapping kinds of relations

The bootstrapping kinds of relations themselves can be defined in the same way as defining other concepts, although their definition models make use of the bootstrapping kinds of relations themselves.

Phrases that denote kinds of relations have a reading direction that specifies which role player is located at the left hand side of the phrase and which one is located at the right hand side. To enable the use of inverse phrases and thus for distinguishing phrases from phrases that require an inverse position of the role players it is necessary to distinguish to ways of naming (two subtypes of the naming relation). For practical reasons and for remaining close to the natural language way of denoting synonyms, the following two kinds of relations are added to the list of bootstrapping relations

synonymy relation	is a synonym of
inverse synonymy relation	is an inverse of

The only additional required concept is the top concept of the taxonomy of concepts, called ‘anything’ and possibly its counterpart ‘nothing’. The concept ‘anything’ can be defined as ‘that what can be thought of’.

On the basis of these bootstrapping concepts it is possible to build the whole language defining ontology. These bootstrapping kinds of relations are therefore the only kinds of relations that are used in the upper ontology to define the basic concepts of the formalized language.

5 Vocabulary and identification

In natural language things are usually denoted by names, but also by other terms, such as abbreviations, codes, translations, drawing numbers, document titles, phrases, symbols, etc. Those names, etc. form the vocabulary of the language. In practice the same thing is often denoted by multiple different terms. Each denotation typically originated or is based in some language community (also called 'speech community'), although it may be used also outside its 'native' language community. Sometimes things are denoted by concatenated terms, such as addresses or concatenated codes and sometimes things are denoted indirectly, whereas they may be nameless. Finally in systems and on the Internet, things are denoted by identifiers that include resource names and paths to directories. This chapter discusses those various kinds of denotations and thus it discusses the vocabulary of Formal English.

5.1 UIDs, names and synonyms

In a formalized language everything needs an unambiguous identification, but there is also a requirement for multiple denotation of something in different contexts. For example, things may be denoted by a term or name as well as by a synonym and by a code and by different names in other languages. On the other hand the same term may be a denotation for different things, when used in different contexts, thus used as a homonym. These are reasons to distinguish between the representation of something 'itself' and various terms to denote it.

5.1.1 Unique identification

Everything is represented in a formalized language by a 'unique identifier' (UID). Although there is a subset of the formalized language defined that allows for the use of 'names' as identifiers, but in that subset the use of homonyms is not allowed.

> Note: Without homonyms the (preferred) terms can be used as UIDs. However, such terms do not uniquely identify things, because of the existence of synonyms, abbreviations, codes, translations, etc. Therefore in such cases software should verify for

every term whether there are possibly relations in which (explicitly declared) synonyms, etc. are specified and whether there are relations specified for such aliases.

A unique UID in Gellish forms an unambiguous identification of a unique identity within the Gellish family of formalized languages. A UID is in fact an arbitrary symbol that is meaningless and thus does not contain any information about the thing it represents. A UID is also natural language independent. Thus a UID that is used in Formal English represents the same thing as that UID when used in Formal Dutch, etc. A UID is not a key in the IT sense, because a UID does not represent a combination of aspects that together uniquely identify something.

External identifiers from coding systems other than Gellish that identify things that are also known within the Gellish language are normally treated as alias names or codes in Gellish, whereas the coding system has a role as language community. Other external identifiers may be used as Gellish identifiers provided that they satisfy specific conditions. This is further discussed in paragraph 5.2.

Everything shall have a UID; as it represents the identity of something in the formalized language. Something without a UID is not represented and cannot be talked about. A UID may not change during the life of the thing it represents. There may not be another thing that has the same UID. This is independent of the contexts in which the UID is used.

The UID of anything that is standard in Gellish is a natural number (a positive integer), except for strings, numbers and user defined identifiers. Strings are unique in themselves, numbers are prefixed by a hash character (#) as is explained in detail in appendix A and user defined identifiers are (preferrably natural numbers) preceded by a prefix that is chosen by that user, followed by a colon (:). This enables software to automatically generate new UIDs provided that the allowed range is specified. The range of UIDs below 3.000.000.000 is reserved. UIDs above that number are free, as described below. Within the Formal English language definition the

reserved range is subdivided in the following ranges (the upper values are not included in the ranges):

1-100: Unknowns, optionally preceded by the prefix ‘unkn:’
This range is reserved for the identification of objects (or collections of objects) that are ‘unknowns’. Thus, the use of a UID in this range means that the object is unknown. Typically such unknowns are used in Formal English Queries in which the identity of such a thing or things is requested. For example, the UID of the left hand object in the following question is 1 with the name ‘what’. That ‘object’ or collection of objects stands for an unknown collection of pumps (with their real UIDs) as will become known in the answer to the question.

UID of left hand object	Name of left hand object	UID of kind of relation	Name of kind of relation	UID of right hand object	Name of right hand object
1	what	1225	is classified as a	130206	pump

100-1000 Testing.
This range is reserved for testing and demonstration purposes.

1000-15.000.000
Language definition.
This range is reserved for the definition of concepts in the Taxonomic Dictionary. UIDs in this range are allocated by the formalized language manager.

dd:yyyymmdd:hh:mm:ss.decimal
Date-Time.
This notation has a prefix dd: followed by a date. For example the UID 20111126 shall be interpreted as 2011 11 26, or 26th of November 2011. If the two digits for the day are missing, then a whole month period is meant. If the two digits for the month are also missing, then a whole year is meant. For example: 201111 indicates the month November in the year 2011. Optionally the date is followed by a colon (:) that is followed by an hour,

optionally followed by a colon and a minute, optionally followed by a colon and a number that represents a second and optionally a dot (.) and a decimal part of a second. For example, UID dd:20111126:12:1:1.5 denotes 1 minute and 1.5 second past noon at the above date.

The relation between UID and number is described in chapter Fout: Bron van verwijzing niet gevonden, Fout: Bron van verwijzing niet gevonden.

Non-numeric identifiers

User defined UIDs should be preceded by a prefix, except when they are allocated a numeric range by the formalized language manager, provided that they obey the rules as described in par. 5.2.

Names and phrases

Names and phrases (terms) are character strings. Such a character string is a unique sequence of characters in its own right and therefore does not need an additional UID. Note that a character string may have various roles to denote different things (homonyms). Therefore a character string that is used as a name or phrase is not a Gellish UID.

Literals

It is a widely used practice to allow for values that are free text strings, typically denoted as ‘literals’. However, the concept of literal is not a semantic concept and it does not bear any meaning. Typically a ‘literal’ is expressed in a particular language and can only be interpreted by a human who understands that language. In Gellish formalized languages, all concepts should be a defined meaning, so that most ‘literals’ should be defined as qualitative subtypes of generic concepts. For example, instead of a literal for denoting a color such as ‘red’, the concept ‘red’ shall have a UID and shall be defined as a qualitative subtype of the concept ‘color’.

Uniqueness of UIDs

The formalized language is defined such that each concept (whether an individual or a kind) is only represented by only one UID. Thus, if something is searched for, and its UID is found, then in principle there is no need to continue searching in order to find another UID that represents the same thing. Thus, in principle it is not allowed that different UIDs represent the same thing in the formalized language. Therefore, any two different UIDs will always represent different things, except for possible errors in the language definition. However, for non-numeric identifiers and identifiers with prefixes, the responsibility for allocating UIDs is distributed to various independent organizations; and thus the responsibility for maintaining consistency and for avoiding that multiple identifiers identify the same thing is the responsibility of the organization that uses such UIDs. Therefore, it is recommended that organization reserve one or more ranges for their proprietary concepts and verify the uniqueness of those concepts and coordinate the addition of new concepts with the formalized language manager.

Examples of UIDs

Standard Gellish concepts have for example the following UIDs:

1.225	is classified as a
130.206	pump
551.564	capacity (mass flow rate)
927.838	5.0
570.449	dm3/s

User defined things of organizations that have provided with a reserved range might have for example the following UIDs:

40.000.001	P-6501
40.000.002	capacity of P-6501
101.001.001	valve type A
501.001.002	nominal diameter of valve type A
601.001.003	idea 1501001002

Examples of non-numeric UIDs

Examples of external UIDs that may be used as UIDs in Gellish are CAS registration numbers for chemicals. For example the Gellish UID that identifies 1,1,2-trichloroethane is CAS:99-00-5.

Gellish UID	Name in English
CAS:99-00-5	1,1,2-trichloroethane
CAS:7732-18-5	water

Notes:

1. Usage of external UIDs does not guarantee unambiguous UIDs. For example, water also has a Gellish UID (430266). Thus two things that are classified using different UIDs are not recognized as being of the same kind, unless there is an explicit statement that states that they identify the same thing and software ensures its implications. For example:
   CAS:7732-18-5 <is the same thing as> 430266
2. Systems such as the CAS Registry have their own purpose and conventions. For example, the CAS registration numbers identify pure chemical substances or molecules. Gellish allows for a classification as water, while the purity is not 100%.
3. CAS only distinguishes substances on their chemical structure and thus it does not distinguish between e.g. liquid water, ice, and water vapor or steam, because they are all chemically identical, whereas the Gellish dictionary includes subtypes of water depending on their state, even including concepts such as high pressure steam, saturated steam, etc.

Sameness

When the uniqueness of UIDs cannot be expected, there may be a need to specify that two different UIDs nevertheless represent the same thing, even when their names are different.

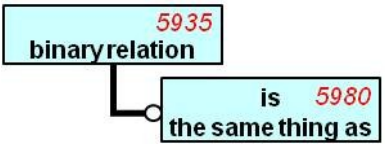


Figure 21, Sameness of something represented by two UIDs

Figure 21 illustrates that there is a standard kind of relation defined that specifies that two different UIDs represent the same thing, even if the names are different An example of the use of such an expression is:

Intention	UID of idea	UID of left hand object	Name of left hand object	Name of kind of relation	UID of right hand object	Name of right hand object
statement	201	20001	name-1	is the same thing as	31002	name-2

Note that the above kind of relation is not applicable in the standard Gellish formalized languages, but only in extended Gellish in which it is not guarantee that UIDs are really unique identifiers of things.

5.1.2 Naming and description

The various denotations of something (denotations by terms such as names, codes and abbreviations) and their descriptions or definitions may vary in various language communities and in various languages.

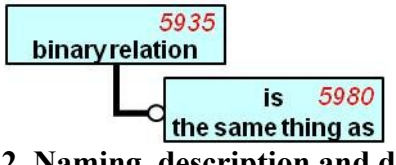


Figure 22, Naming, description and definition

Therefore, as illustrated in Figure 22, multiple description relations are required to relate the multiple terms and other text strings to the UIDs, whereas the kind of relation indicates whether a text string is a (has a role as a) name, description of definition. Each term is allocated with a basis in its own 'native' language community. A

language community can be either a discipline, or an organization, an application system in which particular codes are used, etc.

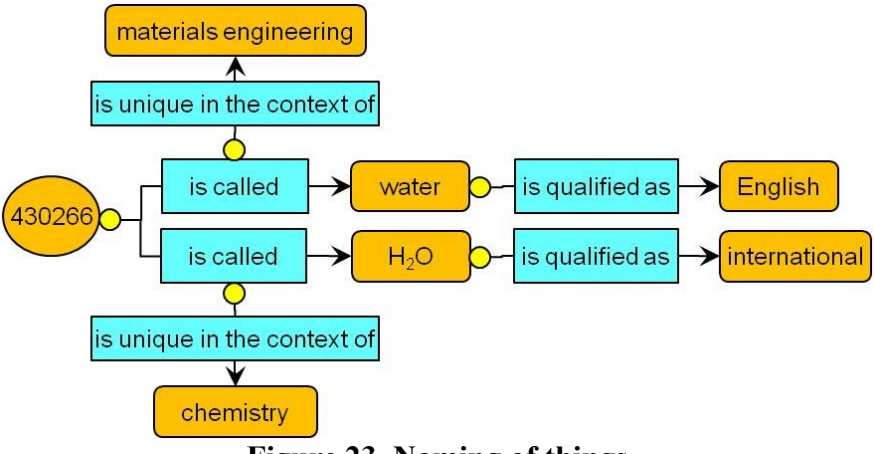


Figure 23, Naming of things

Figure 23 illustrates that one concept can be denoted by various terms in different language communities (that act as naming contexts). For example, the concept represented in Formal English by UID 430266, is denoted in English as water and is denoted internationally in chemistry as H_2O. Thus the term water belongs to the English vocabulary and the term H_2O belongs to an international vocabulary. Both terms are allocated in Formal English to concept 430266, the first one in a materials engineering language community context, the second one in a chemistry language community context.

Note that in general it is a rule that, if a term is allocated to a concept in a particular language community, then that term is the 'preferred term' for that concept in that language community.

It is a uniqueness rule in Formal English that within a combination of a language and a language community context a concept shall be named by only one term. (In principle multiple unique terms are allowed, but then it would be unclear which term is the preferred term). In other words, within a language community in a particular language something is uniquely identified by the (then preferred) term. Thus the combination [language, language community, term] is a '*unique key*' for anything in Formal English.

The allocation of names (terms) to concepts, including their language and language communities can be expressed in a simplified form as a table as follows:

UID	Language	Language community	Name of kind of relation	Name of thing
430266	English	materials engineering	is called	water
430266	International	chemistry	is called	H2O

Table 24, Naming table with synonyms

The columns in a Naming table have relations that represent a meaning. The kinds of relations that classify the relations between the cells in the columns are illustrated in Figure 24.

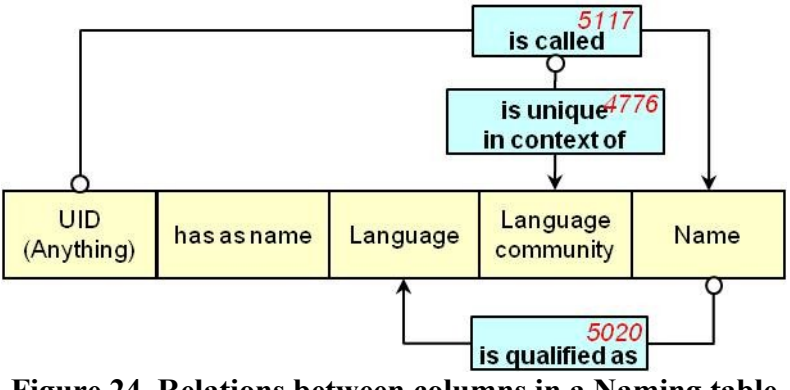


Figure 24, Relations between columns in a Naming table

These multi-naming possibilities enable companies to act as language communities within Formal English. Thus they can use Formal English and nevertheless use their own terminology to denote things for concepts that are known to others under different names. They can even add their proprietary concepts and allocate terms that have a specific meaning that is only valid within that company. Large companies and governments sometimes have their own terminology management organized to avoid ambiguity.

5.1.3 Aliases and synonyms

If aliases are used, then it is desired to specify whether an alias term is a synonym or an abbreviation or a code, etc. for another term for the same thing. To enable specifying the relation between an alias term and a base term there are a large number of subtypes of the alias relation defined that can be used to specify the precise role of a term in relation to another term. Most of those subtypes are presented in Figure 25.

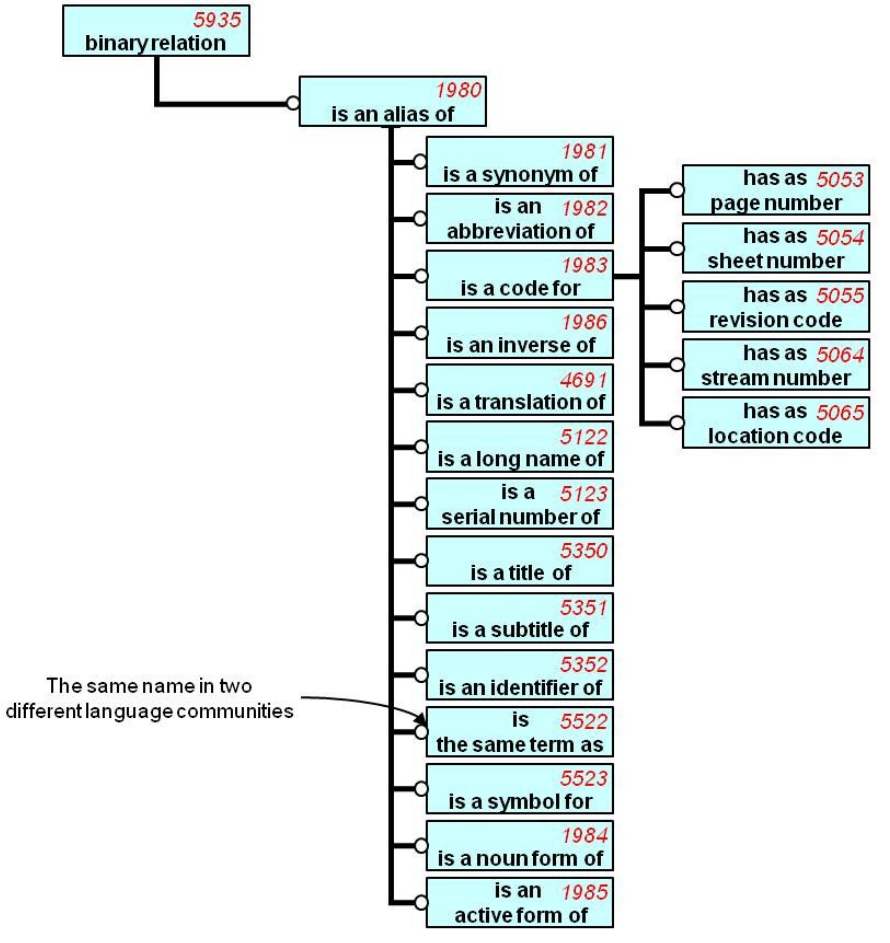


Figure 25, Subtypes of the alias relation

Some codes or terms are alternative denotations (alias terms) for things that have already another term as its preferred name in some language community (context). For example, a technical drawing is typically primarily denoted by a drawing identifier, such as 'T-123.456 Sheet 3 Rev 2'. However, such a drawing usually also has a title and may have a subtitle. Both title and subtitle are alternative 'names' for the drawing that may be used for searching the drawing. Some systems may record even the sheet number of a drawing as a separate item (field in a database). The drawing may also be denoted by a page number, when being part of a volume. All such denotations denote the same drawing which is represented by a UID. For example, assume that the UID is 123. Then these alternative denotations are expressed as follows:

UID of left hand object	Name of left hand object	Name of kind of relation	UID of right hand object	Name of right hand object
123	T-123.456 Sheet 3 Rev 2	is qualified as a	490196	drawing
123	T-123.456 Sheet 3 Rev 2	has as title	123	A-1 layout
123	T-123.456 Sheet 3 Rev 2	has as sheet number	123	3

Table 25, Alternative denotations for a drawing

Note that all the alternative denotations are treated as names of the same UID. When a denoted object is recorded as being a part of a larger whole, then the larger whole acts as a natural context within which a name of a part should be interpreted. Therefore, alternative denotations usually do not need language community contexts. In the above example, T-123.456 might be recorded as a (name of) a collection of sheets and revisions of sheets that acts as a (language community) context for the sheet number. Nevertheless, it is possible to add an explicit language community in a formalized English expression, especially when inverse relations of the above ones are used (further details.

Occurrences, such as activities, processes and events can be denoted by nouns, as well as by other forms, such as active forms. For example, the terms activity, acting and action are aliases, although one is a noun and the other is a verb[5].

5 This illustrates that the concepts 'noun' and 'verb' don't denote a semantic distinction, but only a grammatical distinction. The same holds

5.1.4 Mapping of organization's terminology

A mapping between terms used in a particular organization and terms in the Formal Dictionary is a mapping between different language communities. The organization should use existing UIDs from the Dictionary for existing concepts even when the names are different.

For existing concepts a mapping table thus uses expressions with the same UID at the left hand side as at the right hand side, but with different terms. This is illustrated in Table 26.

UID of left hand object	Language	Language community	Name of left hand object	Name of kind of relation	UID of right hand object	Name of right hand object
430266	English	medical	aqua	is a synonym of	430266	water
430266	English	Company X	water	is the same as	430266	water
430266	German	materials engineering	Wasser	is a translation of	430266	water
430266	International	chemistry	H2O	is a code for	430266	water

Table 26, Mapping table

A mapping table uses primarily the <is a synonym of> relation to denote that a term that is used in the company (or other language community) is a synonym of a term that is used in the Dictionary for the same concept. The fact that it is the same concept is indicated that the same UID is used on the left hand side as on the right hand side of the expression.

If the two terms are identical, then the <is the same as> relation can be used. It denotes that the name is identical to an already allocated name for a particular concept, but the language communities are different. Thus in a particular language community, such as Company X, the same term is used as preferred term as the term that is specified for another language community. In principle this statement is superfluous, but for completeness it is recommended to include such equalities.

---

for the concepts 'subject' and 'object', because the same (semantic) meaning can be expressed in an expression as well as in its inverse expression, whereas the thing that has a role as subject in an expression has a role as object in the inverse expression, without a change of meaning.

Only if a concept or individual thing is not yet included in the Dictionary, neither the same name, nor under an alias name, then an organization should introduce a UID for the new thing and then provide a definition for the new thing. The new things shall be created as extensions of the Dictionary. How that is done in the formalized language is described for individual things in par. 8.1.1. and for kinds of things in par. 9.

5.2 Coding systems and namespaces

There are many organizations and standards in the world that manage classification systems or coding systems in order to allocate unique identifiers or codes to widely known concepts and individual things. If the concepts that are denoted by such identifiers are not (yet) part of the Gellish language, then those identifiers can be used in Gellish by adding a prefix followed by a colon (:) to the idetifier of the organization. For example, the ISO 3166-1 standard numeric code for countries allocates numeric codes to all countries in the world. For example, the number 840 is allocated to the USA. Thus, the USA can be identified in Gellish e.g. by the identifier ISO1366-1:840.

However, countries, such as the USA, are already included in the Gellish dictionary. The USA has UID 2700347. The use of both as UIDs would mean that the two identifiers would not be recognized as the same country. In order to harmonize that, such third party uids can be included in Gellish formalized languages by treating the codes as synonym codes (‘names’) for things that may have also other names, whereas they are identified by their Gellish UID. The coding system is typically indicated as the language community in which the code find its home base.

Thus users of the formalized language may include namespaces of other organizations and may define some or all of them as synonyms of names of things that are already in the dictionary. Such namespaces are specified in the same way as mapping tables, as was described in par. 5.1.4.
For example, the alternative denotation of the USA by code 840 can be included in the Gellish dictionary as follows:

Language community	UID of left hand object	Name of left hand object	Name of kind of relation	UID of right hand object	Name of right hand object
ISO 3166-1	2700347	840	is a code for	2700347	United States of America

Table 27, Example code in a coding system

Note that in Table 27 the UID of the left hand and right hand are identical and identify the country within the formalized language. The ISO 3166-1 standard thus forms a 'language community' of its users in which the OID codes have a meaning and can be interpreted. Therefore 'ISO 3166-1' is used as the language community

Furthermore, ISO 9834 standardized the joint ISO, IEC and ITU-T defined Object Identifier's (OID's) in principle for any concept in the world. Those OID's are recorded in a repository http://www.oid-info.com/index.htm that acts as a reference for concepts defined and managed by a hierarchy of 'Registration Authorities'. For example, the United States of America has OID: 2.16.840. In Gellish Formal English the concept with UID 2700347 and name USA has already a code name 840, being the code from ISO 3166-1, as mentioned above. But in addition to that this OID code can be added as another synonym. This is illustrated in Table 28.

Language community	UID of left hand object	Name of left hand object	Name of kind of relation	UID of right hand object	Name of right hand object
ISO 9843	2700347	OID:2.16.840	is a code for	2700347	United States of America[6]

Table 28, Coding system synonyms

6 Specifying the OID 2.16.840 as a (synonym) code for UID 2700347 ensures that in Gellish both refer to the same thing. Allowing both codes as identifiers (UIDs) in Gellish is not acceptable, because it would require a separate equivalence relation, whereas the possible presence of equivalence relations has the drawback that it is continuously required to search for possible equivalences. On the other hand Gellish allows for adding non-mapped things in a free number range above 1.000.000.000.

The ISO 9834 standard thus created a ‘language community’ in which the OID codes have a meaning and can be interpreted. Therefore ‘ISO 9834’ should be used as the language community for OID’s.

An alternative way, although not recommended, is to use the foreign code with a prefix as UID in Gellish and define that they denote the same object as follows:

Language community	UID of left hand object	Name of left hand object	Name of kind of relation	UID of right hand object	Name of right hand object
ISO 9843	OID:2.16.840	USA	is the same as	2700347	United States of America

A consequence of this alternative is that software should be programmed such that these are two nodes that the semantic network which relations should be superposed.

An example of an ISO standard for coding systems for kinds of things is ISO 81346. Its part 2 defines the general code structure for individual things and ISO 81346-10 contains names for kinds of things and their codes in Power Plants and ISO 81346-12 (draft) contains names for kinds of functions (processes) and physical objects and their codes in Buildings and Building Services. For example, the concept sewerage system has code TT.

The process to relate other identifiers and codes to Formal English concepts is similar as for individual things.

Note that identifiers from coding systems may also be used as Gellish UIDs without being declared identical things to existing Gellish UIDs. For a proper functioning of Gellish based systems it is then important that those external concepts are arranged in the taxonomy of Gellish concepts. Therefore they should be defined as being subtypes of existing Gellish concepts. Thus the definition of such concepts should obey the same rules as are applicable for defining new concepts in Gellish and its taxonomic dictionary. Those rules are further discussed in Ref. 2.

5.3 Homonyms

A homonym is a name that is used in different contexts to denote different concepts (represented by different UIDs). Homonyms can be allowed in a formalized language, provided that the allocation of a name to a concept is always defined in a language community within a language such that it makes the combination (language, language community, name) a unique and unambiguous identification of the concept. For example, in English the term building denotes multiple concepts. Expression Table 29 shows that (in English) the combination of a language community and a name enable to distinguish a building activity from a building construction that is the result of a building activity.

Language community	UID of left hand object	Name of left hand object	Name of kind of relation	Name of right hand object
activity	194173	building	is a specialization of	forming
building engineering	40018	building	is a specialization of	construction

Table 29, Example of a homonym

5.4 Addresses

An address is a name or denotation of an area that acts as a location where a physical object may reside. There are various kinds of addresses, such as home address, postal address, telephone address, e-mail address, etc. This definition of an address shows that allocating an address to a physical object should be done by relating a physical object to an area where the physical object is located, whereas the address is the name of that area. For example, the allocation of an address may be expressed as a relation between a house and an area on which the house is built, or the relation between a person and an area where he resides officially.

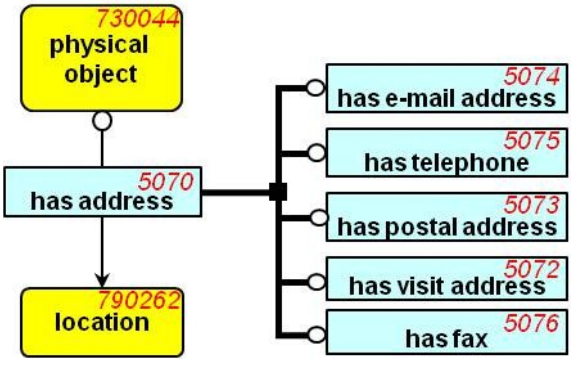


Figure 26, Addresses

Home addresses (visit addresses) are usually allocated and registered officially by a government agency. They determine the boundaries of a parcel and allocate a registration number as well as an address. For example, in The Netherlands, the Kadaster is the registration authority that allocates the 'Kadastrale' registration number to a parcel; whereas the local government adds a street name and parcel number within the street. These authorities should ensure that those numbers and names are unique within the context of the local community (which behaves as some kind of language community).

Home addresses are often decomposed as text and then form a concatenated code, but in a semantic model, the area shall be related to stepwise larger areas, each with its own UID and denotation. For example, I live on a place (a parcel) which I can give a UID 101. Assume that the place has registration number 123.456 and is locally in the street denoted by a number, such as '1'. The parcel is located within a larger postal area, called '2724 VR'. Now the parcel registration number and zip code are related as is given in Table 30.

Language community for left hand object name	UID	Name of left hand object	Name of kind of relation	UID	Name of right hand object
Kadaster	101	123.456	is classified as a	731.018	parcel
Zoetermeer government	101	1	is an alias for	101	123.456
Zoetermeer government	101	1	is a part of	102	2724 VR

Table 30, Parcel registration number and zip code

Note that the UID is a unique identifier for the area within Formal English, but the number 1 is only unique within the context of the postal area.

The parcel is also located in a street in Zoetermeer in The Netherlands. That can be specified as in Table 31.

Language community for left hand object name	UID	Name of left hand object	Name of kind of relation	UID	Name of right hand object
Zoetermeer government	101	1	is a part of	103	Violiervaart
Zoetermeer government	103	Violiervaart	is a part of	104	Zoetermeer
geography	104	Zoetermeer	is a part of	270021	The Netherlands

Table 31, Geographically composed address

The above examples illustrate how addresses can be modeled semantically. The language communities illustrate that for coding systems the base language community is typically the organization that is responsible for allocating the codes.

5.5 Denotation by code and classification

Sometimes things are not denoted by a name, but they are given a code or number, which is allocated by an authority that determines the context for interpretation. Such an authority is the core of a language community that makes use of the code. Within that language community such a code, usually in combination with the classification of the thing, it is unambiguously defined which object is meant.

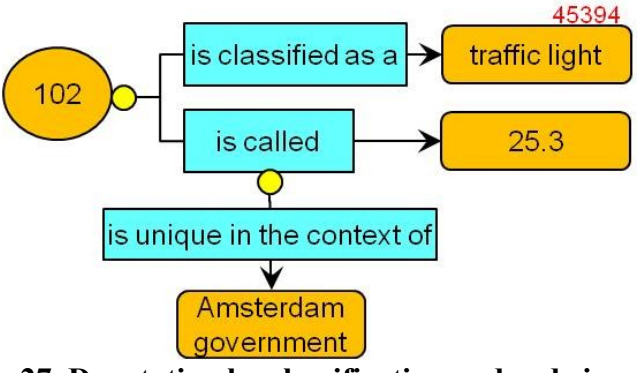


Figure 27, Denotation by classification and code in context

In tabular form this is expressed as follows:

Language community	UID of left hand object	Name of left hand object	Name of kind of relation	UID of right hand object	Name of right hand object
Amsterdam government	102	25.3	is classified as a	45394	traffic light

For example, in many modern cities traffic lights and lamp posts are coded, often by a number, such as 25.3, being the third traffic light on road crossing 25 in Amsterdam. This example illustrates that there are two additional pieces of information required in order to interpret the code 25.3: firstly 'Amsterdam government' is the language community context for the name, because this particular code is allocated by the Amsterdam government and secondly 'traffic light' is the kind that classifies the individual thing. (We ignore the fact that the code itself is a concatenation of two codes).

5.6 Nameless things

Sometimes, things have no name, nor a code. They have an identity and are represented by a UID, but they are nameless. However they may be indirectly denoted by their relation(s) to other things. For example, they may be denoted by their classification and/or by the name of a kind of role which they play and possibly by the name of the assembly from which they are a component.

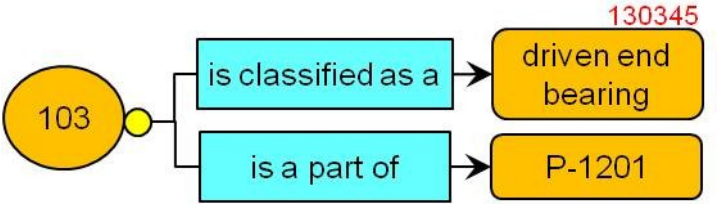


Figure 28, Denotation by classification and composition

Figure 28 illustrates and example of a nameless object identified by its UID as '103'. From its classification relation with a kind of thing and a composition relation with its higher assembly we can interpret that it is the driven end bearing of P-1201. Thus it is an individual thing that is classified as a bearing, which is located at the driven end (the end of the shaft at the side where a motor drives P-1201) and that is a part of P-1201. In tabular form this can be presented as:

Intention	UID	Name of left hand object	Name of kind of relation	UID	Name of right hand object
statement	103	Nameless	is classified as a	130345	driven end bearing
statement	103	Nameless	is a part of	104	P-1201
statement	104	P-1201	is classified as a	130206	pump

From the third line it can be interpreted that P-1201 is the tag name of a pump. We may also interpret that the combination of words 'driven end bearing' is a name of a kind of role for a bearing, because that concept is defined as such in the dictionary and is widely used in the rotating equipment engineering language community to classify a bearing in such a position.

Thus, anything needs a unique identifier (UID), but not necessarily a name, code, etc. In order to know that something is nameless on purpose, it is a recommended convention for Formalized English to give nameless things the name 'Nameless'.

5.7 Identification on the Internet

On Internet things are typically denoted by Uniform Resource Identifiers (URIs) as by Uniform Resource Names (URNs) as described in http://en.wikipedia.org/wiki/Uniform_resource_identifier. A URN is a concatenation of three components: the string 'urn:', an explicitly specified namespace that is followed by a colon (:), followed by a 'character string' that acts as an identifier. The namespace 'Gellish' is explicitly specified as http://www.formalenglish.net/dictionary. The concept water has a Gellish UID of 430266. Thus in the namespace 'Gellish' the concept water is unambiguously referenced on the Internet as:

urn:Gellish:430266

When URIs are preferred (as with RDF) this should be replaced by:

http://www.formalenglish.net/dictionary#430266.

Homonyms are distinguished from each other in Gellish by their language community. Therefore the name for denoting a concept in an unambiguous way needs to be preceded by a language community that denotes a domain dictionary, such as 'materials' in the case of water. Thus references to the concept 430266 by one of its names in English becomes:

urn:Gellish:materials:water

http://www.formalenglish.net/dictionary/materials#water

And as a second example, the United States of America is unambiguously referenced on the Internet as:

urn:Gellish:2700347

5.8 Descriptions and textual definitions

Terms, such as names, codes, abbreviations and descriptions or textual definitions are unique by their unique combination of characters and therefore they are not given separate UIDs in Gellish. UIDs are only required to represent the things that are *denoted by* the terms.

Definitions may be expressed in a computer interpretable form as ‘definition models’. A definition model of a concept is a network of relations between the concept and other concepts, each of which relation is by definition the case, and each of which related concept is defined as well. Definition models are further discussed in par. 10.1.

5.9 Textual and graphical information and documents

Text, other than terms (names) and textual definitions, can either be fully incorporated within a semantic model as a textual object, or it can be expressed in an external document on a carrier, such as a paper document or an electronic file, which can be linked to a semantic model. Before we explain how these two options are modeled, we need to clarify the relation between text, graphics, information and documents.

5.9.1 Information versus documents

Physical documents (e.g. on paper) and electronic files are both real physical objects. The ink and bits are also physical objects. The *content* of physical documents and files however (not the physical ink or bits, but the informative aspect), which is interpreted from the shapes and patterns, is ‘information’. Such information can be expressed as text or spoken expressions in natural language or it can be expressed (modeled) in a formalized language, such as formalized English.

Qualitative information

The term ‘document’ is ambiguous. In some contexts the term document denotes a physical copy, for example in the form of ink on paper. But more often the term document is used to denote a piece of information, which may be a single content that is common to multiple physical copies. To eliminate the ambiguity we will use the term document then we denote a piece of information, which may qualify the content of multiple physical documents. For physical manifestations we will use the term physical document, file or web page.

Thus, a particular content (a piece of information, such as Requirement-5) can be expressed in multiple physical copies and it may be present even in multiple different formats. For example, the same piece of information can be expressed on paper as well as in a Word file (doc format), as well as in pdf format files on various locations. Thus with 'a piece of information', such as 'Requirement 5', we do not mean one individual (physical) text, but the common content that is contained in and is interpreted from each of the possibly multiple copies of some text. The individual content of each of the physical copies is qualified by that same common qualitative content. Therefore, that common 'same' content is called 'qualitative information'. In fact it classifies the content of multiple copies and therefore it appears to be a (qualitative) kind of information. That is the reason why it can be stated that Requirement 5 <is a qualitative (subtype of)> requirement (whereas, according to the taxonomic dictionary, a requirement is defined as a subtype of information).

When a piece of information incorporated in a semantic model, then the complete text can be treated as an object in its own right, whereas that text remains in natural language and thus is not expressed in Formal English This implies that the text shall have its own UID and may have a name or remain nameless.

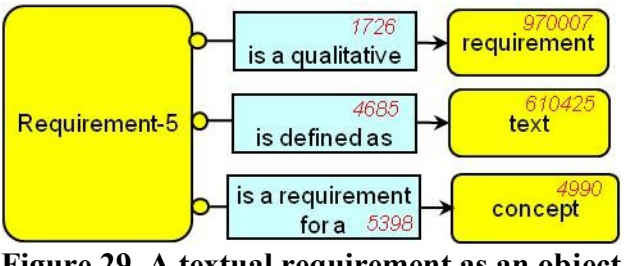


Figure 29, A textual requirement as an object in a semantic model

For example, as illustrated in Figure 29, a paragraph in a standards document may contain requirements about compressors. Each such requirement is a piece of information (qualitative information) that should be given a UID and might have a name, such as 'Requirement 5'. That particular Requirement-5 then has to be

defined as being a ‘requirement’. The definition of Requirement-5 is then expressed as follows:

Requirement-5 is a qualitative requirement

The content of such a paragraph (the actual text) defines the requirement. Therefore that text is specified as the definition of the requirement. That textual definition is related to the UID of the piece of information with name Requirement-5 as follows:

Requirement-5 is defined as A compressor shall ...

The paragraph can then be related to the concept about which it provides information by stating for example

Requirement-5 is a requirement for a compressor

The content (the textual definition) of a paragraph as well as its definition as being a qualitative requirement can be recorded in an Expression table (see chapter 15) on the same line. For that purpose that standard table has a separate column, called ‘Description’.

UID of LHO	Name of left hand object	UID of idea	Name of kind of relation	UID of RHO	Name of right hand object	Description
101	Requirement 5	1726	is a qualitative	970007	requirement	A compressor shall ...
101	Requirement 5	5398	is a requirement for a	130069	compressor	

Table 32, Piece of information as an object in a semantic model

Thus the example model of Figure 29 can be expressed as is illustrates in Table 32.

External documents

Information that is expressed in an external document is typically available in one or more electronic files (copies) in one or more different formats. For example, the same information can be available in a Word document (doc-file), as well as in a pdf-file format. In such a case the content (text) is not provided in the semantic model itself, but the file is incorporated as an object in the semantic model and the requirement is related to that file.

Furthermore, the location of the file (the path to the file in a directory) and the format of the file may be included in the model. The model may also include a specification that allows for the launching of suitable application software to retrieve and display the content of the file. For example, the above 'Requirement 5' text may be stored in two files as 'Requirement_5.doc' and as 'Requirement_5.pdf'. Then the relation between UID 101 and the two files can be as illustrated in Table 33

UID of LHO	Name of left hand object	Name of kind of relation	UID of RHO	Name of right hand object	Description
101	Requirement 5	is presented in	102	Requirement_5.doc	
101	Requirement 5	is presented in	103	Requirement_5.pdf	
102	Requirement_5.doc	is an element of	104	C:\my documents	
103	Requirement_5.pdf	is an element of	105	www.examples.com	
104	C:\my documents	is classified as a	492017	directory	
105	www.examples.com	is classified as a	970274	url address	

Table 33, Information in referred external files

This can be interpreted as a definition of a hyperlink, so that the file extensions (doc or pdf, etc.) may be sufficient for application software to determine which software should be launched to retrieve the file and to display its content when a user activates the hyperlink.

Figure 30 presents the example for a reference to two external files that contain different physical expressions of the same qualitative information.

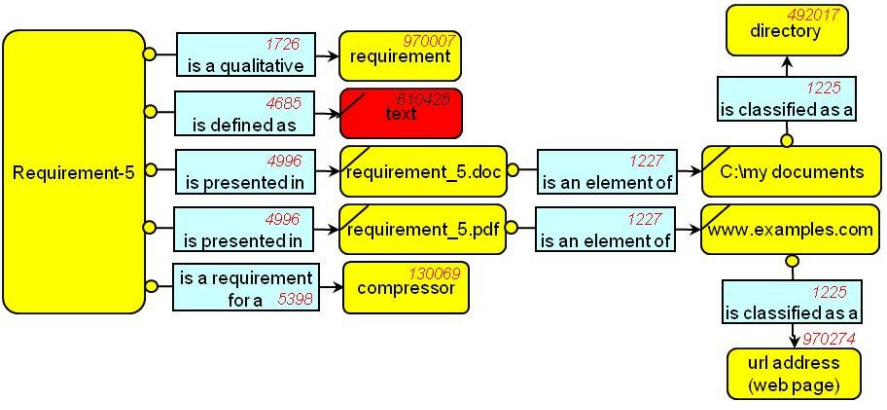


Figure 30, Text stored in external document

Note that the (qualitative) information is presented in (or <is presented on>) the file, whereas the file is an element in a collection of files, being either a directory in a file management system, or an internet address of a 'page' on the Internet.

In the above example, the requirement is applicable for compressors in general. Other requirements are applicable only to specific individual things. For example, a (textual) requirement may be applicable for compressor C-351. For the specification of such a requirement (slightly) different kind of relation is required as is shown in Figure 31.

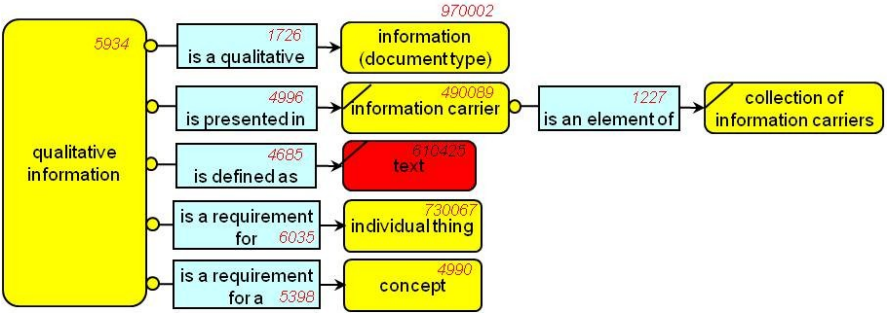


Figure 31, Pattern for textual requirements

5.9.2 Information about physical objects

The fact that particular information *is about a* physical object is typically modeled as a relation between some 'qualitative information' and the physical object.

For example, A P&ID about process unit 1400 with drawing nr T-123.456 contains information about P-1401. This is modeled as follows:

T-123.456	contains information about	P-1401
T-123.456	is defined as a	P&ID

Note that in this example T-123.456 is not an individual copy, but is the 'qualitative information' that is common to all copies of that drawing. The kind of relation <is defined as a> is a synonym of <is a qualitative>

5.9.3 Location of information in files

The relation between a particular piece of information and an individual physical file at a particular location on the internet is modeled as follows:

T-123.456	is presented on	Unit-1400.dwg
Unit-1400.dwg	is an element	http:/www.gellish.net/examples
Unit-1400.dwg	is classified as a	electronic data file
T-123.456	is presented on	Unit-1400.pdf
Unit-1400.pdf	is classified as a	electronic data file
Unit-1400.pdf	is an element of	http:/www.gellish.net/examples

Table 34, Location of information in a file in a directory

This example illustrates that both files are stored at the same location on the internet.
In a similar way a figure or picture is presented on a document file. For example:

Picture-1	is presented on/in	Report-1.doc

This latter example illustrates that there does not need to be a 1:1 relation between a piece of information and a physical file.

A physical electronic document is usually a blob (a binary large object) that is a 'black box' for the model. In other words the content

of a document is not interpreted by according to the rules of the modeling language, unless the document content happens to be in the form of a computer interpretable language. For example, a picture (png-file), a pdf-file, a Word document (doc-file) or an Excel spreadsheet (xls-file) is intended to be displayed by an application but (semantically) interpreted only by human beings. Such files are included in the model by a reference to the blob, as described above.

However, there are exceptions. One exception is when the file contains an information model that is computer interpretable. An example of an interpretable information model is an information model in the form of an Expression table in Excel. Such a document is an xls-file with a spreadsheet that has a predefined tabular structure with content that complies with the rules of Gellish Formal English and therefore it can be interpreted by application software. The content of such a file can be part of the information model in which the file as a whole is a blob.

5.9.4 Dynamic standard forms & data sheets

Empty data sheets are a kind of standard forms. Just as other standard forms they consist of predefined text and empty fields in a particular lay-out. Such empty standard forms can be re-used and then filled with data, either for display only or the filled-in form can be treated as a new document.

Sometimes a standard form (or data sheet) is created in spreadsheet form, such as in Excel (as an xls-file). That means that a particular field on a sheet in a file is reserved for a particular value. This enables to create ‘dynamic’ standard forms and data sheets, where the empty or partially filled in form is static, whereas other fields are dynamically filled by application software with data from the information model. The static fields are part of the blob that is not part of the information model. The fields that are intended to be dynamically filled are related to the information model by relating the field to one of the related objects in an expression in the information model that ‘contains’ the value that need to be filled-in.

6 Categories of kinds of relations

The concept ‘relation’ or ‘relationship’ (2850) is the top of the taxonomy hierarchy of a large number of standardized kinds of relations. The hierarchy of kinds of relations has the structure of a hierarchical network and does not have a pure tree structure, because some kinds of relations have more than one supertype. Nevertheless, the explanation below typically follows the branches of a tree structure.

Formal English intents to enable to make expression about anything. This means that the language should provide kinds of relations for expressions about

- individual things
- kinds of things
- collections of individual things
- collections of kinds of things

This is illustrated in Figure 32.

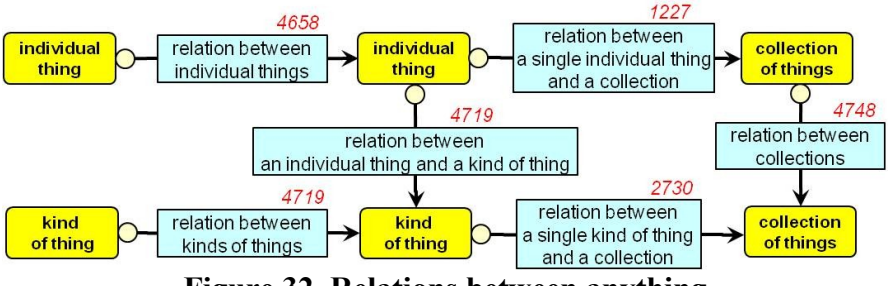


Figure 32, Relations between anything

The related things are ordered in a taxonomy structure (a subtype-supertype hierarchy) and the kinds of relations are also arranged in a taxonomy structure. Figure 33 illustrates the top structure of the taxonomy of kinds of relations.

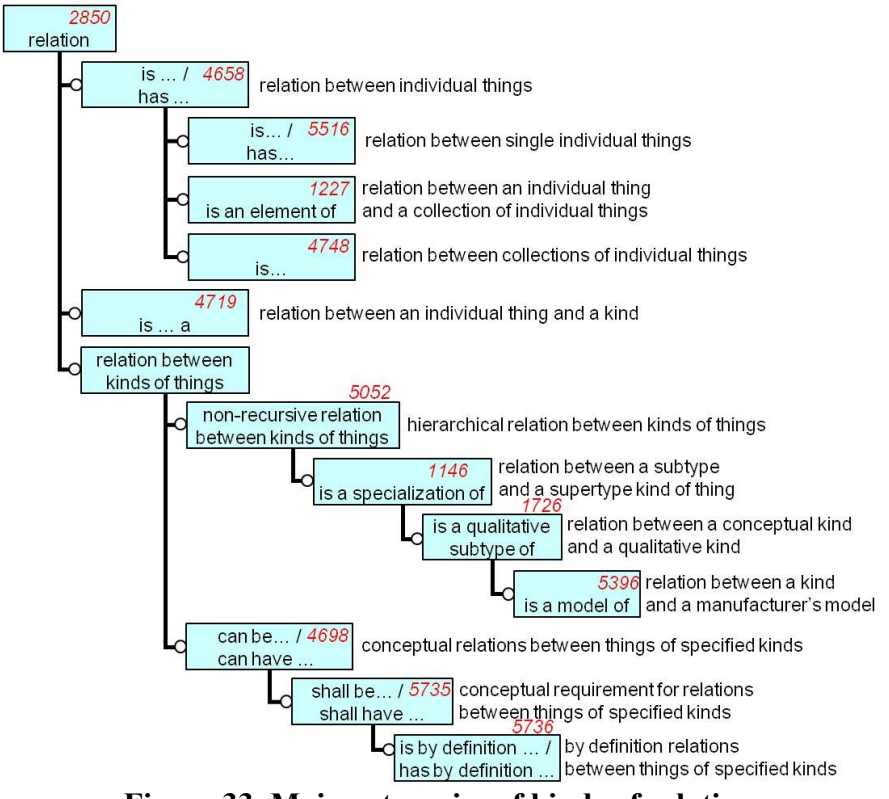


Figure 33, Main categories of kinds of relations

Each kind of relation can be used to classify relations of such kinds. Kinds of relations are typically denoted in Formal English by standard phrases and inverse phrases. Those phrases and inverse phrases have a reading direction that determines the role of the left hand and right hand object in an expression. The top structure defines the following categories of kinds of relations.

- Naming and identification relations

  These are relations that are suitable for being used to identify and name anything (individual things as well as kinds of things) by relating it to a name or identifier.

- Relations between individual things

These are relations that for the expression of information about real as well as imaginary individual things.

The phrases that denote these kinds of relations typically begin with <is...> or with <has...> and do not end with <...a>. For example: <is a part of> or <has as part>.

Within this category of relations we distinguish between:

- Relations between single individual things
- Relations between a single individual thing and a collection of individual things
- Relations between collections of individual things
- Relations between individual things and kinds of things.

These are relations for the classification of individual things.

The phrases that denote these kinds of relations typically begin with <is...> and end with <...a>. For example: <is classified as a>.

- Relations between kinds of things.

These are relations for the expression of knowledge, requirements and definitions of things of particular kinds.

This category has two subcategories:

- Hierarchical relations between kinds of things

  These are relations that are not self referential, i.e. the concepts can be used for forming hierarchical networks of relations without recursion. This means that in such a hierarchy a related concept has as constraint that it may not appear more than once in a chain of related concepts.

  The phrases that denote these kinds of relations typically begin with <is a...> and do not end with <...a>, because they state that something is by definition the case for a kind. For example: <is a kind of>, which means <is by definition a subtype of>.

- Conceptual relations between individual things of specified kinds

These are relations that specify things that can be or are the case for categories of individual things. They do not have the non-recursivity constraint. Thus the related concepts may appear more than once in a chain of concepts that uses a conceptual relation. They specify possibilities in general, requirements in specific contexts or what is by definition the case for all things of the specified kinds.

The phrases that denote these kinds of relations typically begin with

- <can be...> or <can have...>,
- <shall be...> or <shall have...> or
- <is by definition...> or <has by definition...>.

Within this category of relations we distinguish between:

- Relations between single kinds of things
- Relations between a single kind of thing and a collection of kinds of things
- Relations between collections of kinds of things

Note that for sake of clarity of this document sometimes there are some details not show in the hierarchies and figures. For example some kinds of relations have multiple supertypes whereas only one supertype is shown and there are some intermediate concepts defined in the upper ontology that are not show.

Each of the above categories is further described in the following chapters.

7 Relations between individual things

Individual things are not only real or imaginary physical objects, but also aspects of physical objects, such as properties, qualities, abilities, etc. as well as individual relations, activities, processes, etc. Thus an individual thing can be anything that is distinguished from something else and that is not a classifier for multiple individual things.

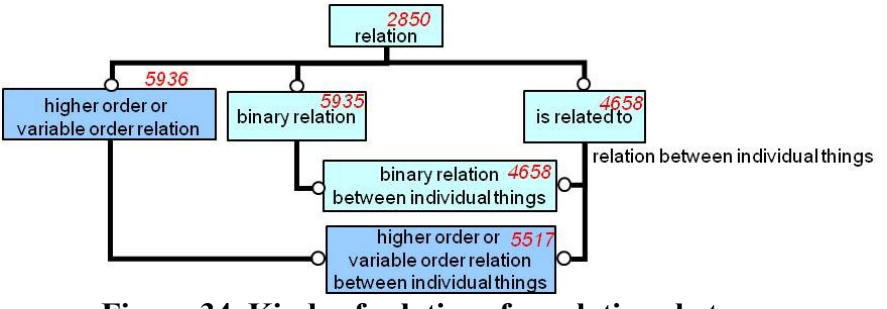


Figure 34, Kinds of relations for relations between individual things

Figure 34 illustrates the main kinds of relations in the top of the taxonomy of kinds of relations that are intended for the classification of relations between individual things. Relations between individual things can be *binary relations* or *variable order* or *higher order relations*.

Binary relations

There are many kinds of binary relations between individual things.

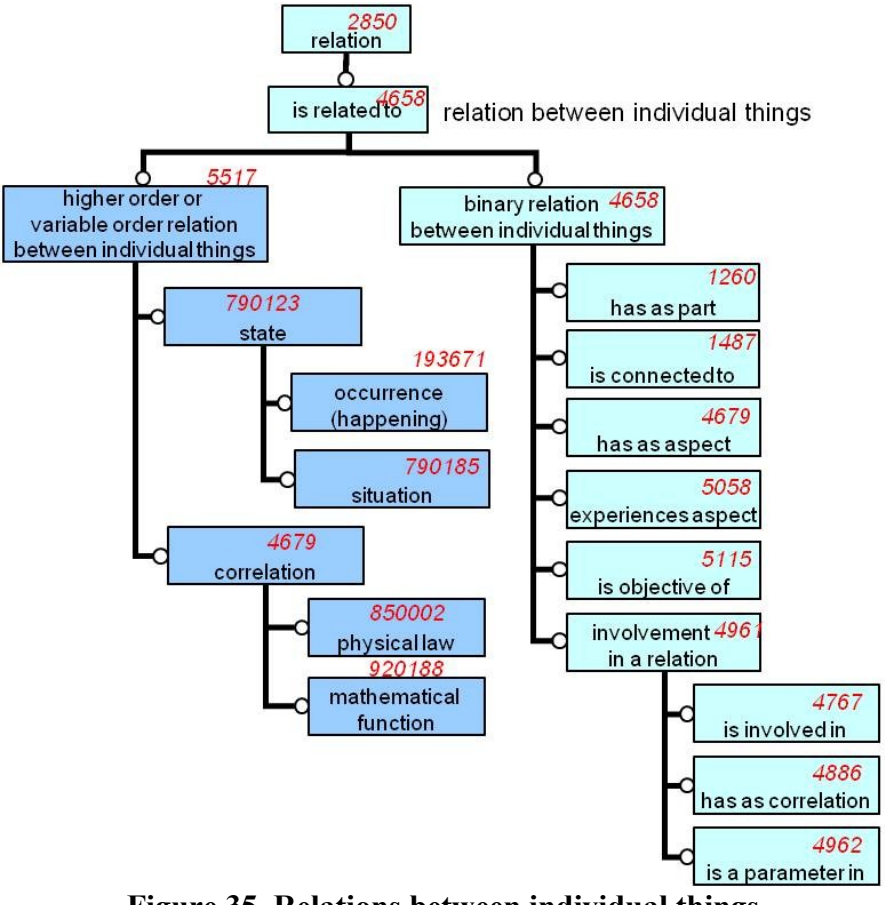


Figure 35, Relations between individual things

The right hand side of Figure 35 presents for example:

- Composition relation (1260) between two physical objects. The composition relation and its subtypes are further discussed in par. 7.1.
- Connection relation (1487) between two physical objects. Connections can be rather complicated as various kinds of connections are possible, including also various kinds of connection materials. Those subjects are discussed in par. 7.2.
- Possession of aspect relation (4679) between a physical object and an aspect that is possessed by that physical object (aspects that are possessed), such as its temperature, color, skill, etc. That kind of relation has various subtypes as is discussed in par. 7.4.

- Experience of aspect relations (5058) between a physical object and an aspect of another physical object (aspects that are experienced), such as a sensor that senses a temperature of another object.

- Objectives relations (5115) between a relation or state and a purpose or objective, such as the reason why an activity exists.

- Involvement relations (4961) between a variable order or higher order relation and an individual thing that plays a role (is involved) in that higher order relation, such as person who is a performer of an activity. The involvement relation (4767) has a number of subtypes which express the kind of role in which something is involved. Those subtypes are discussed in par. 7.5.1.

Each of the two things that are related by a binary relation plays a role of a particular kind.

> *The kind of relation that is to be used for classification of a binary relation can usually be found by searching for the combination of the term <is> or <has> and the name of the role that one of the related things is playing.*

For example, searching for a phrase that denotes a composition relation between two physical objects can be done in various ways. For example, using the Gellish Search Engine [Ref. 3] for searching on the combination of 'beginning with <is> and including <part>' will deliver various kinds of relations in which one thing plays a role of part. Among them is the relation denoted by the phrase <is a part of> (1260). However, the combination of <has> and <part> will point to the same relation, via the phrase <has as part> (1260). But it is also possible to find that relation by searching via the role of the other role player. Thus the combination of <is> and <whole> will point to the phrase <is a whole for>, which is an alternative phrase for (1260).

Higher order relations

States of affairs are typically defined by a collection of coherent aspects that have a particular value at a particular moment in time and that may be correlated and dependent on each other. Such states can be static over time or they can be dynamic, thus changing over

time, such as occurrences, activities and processes. Static states, dynamic states, physical and mathematical correlations relate multiple aspects that are possessed by one or more things. Such a relation is therefore called a higher order relation.

Higher order relations between individual things are expressed in Expression tables as a collection of elementary involvement relations (as described above). This enables that the number of involved things is flexible and may even vary over time. For example, the participants in an activity or process may vary over time, whereas the participation of each participant in the process is recorded as a separate involvement relation between that participant and the process.

Constraints on recursivity and incompatible roles

A self referential relation (also called a recursive relation) is a relation that relates something to itself. Typically relations between individual things are not recursive, because usually individual things are not related to themselves. For example, a temporal sequence of occurrences (<occurs after>, 1388) relation relates an individual occurrence to another individual occurrence. For example, given that A-1 and A-2 are individual activities we can specify:

A-1 occurs after A-2

However, when A-1 and A-2 would be the same activity, then the statement would be incorrect, as something cannot occur after itself.

A constraint that the player of a role of some kind may not also play a role of another kind is expressed as a constraining relation between the kinds of roles. The general definition of a self referential kind of relation and a constraint on its role players is illustrated by the example in Figure 36.

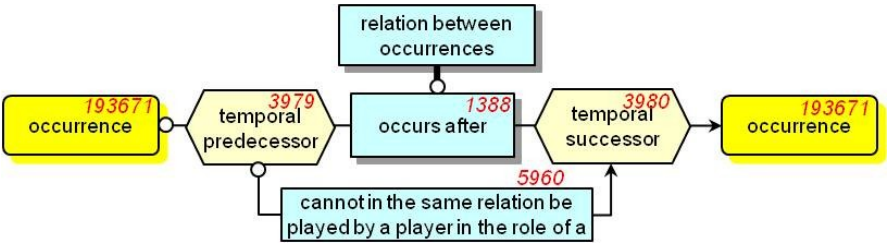


Figure 36, Constraint on role players

The definition of <occurs after> specifies that a relation of this kind requires a first role as a temporal predecessor and a second role as a temporal successor. Each of those roles is by definition played by an individual thing, under the constraint that those individual things must be classified by the concept 'occurrence' or one of its subtypes, and as additional constraint that the two individual occurrences may not be the same. The latter constraint is a constraint on the allowed players of the roles of the specified kinds. This constraint can be modeled explicitly by using a kind of relation that expresses that an individual thing that plays a role as 'temporal predecessor' cannot simultaneously play a role as 'temporal successor' in the same relation (which is by definition the <occurs after> relation). Formally this is expressed as:

temporal predecessor	cannot be played by the player of	temporal successor

Similar constraints on role players can be expressed in general as:

kind of role	cannot be played by the player of	kind of role

7.1 Composition relations

In general individual things are composed of components or parts. This holds e.g. for physical objects and fluids, systems, spaces, routes as well as for organizations, occurrences, aspects such as distances and time, networks, etc. This is opposed to elements that are combined in a collection. The elements in a collection do not have roles of different kinds in the collection, apart from possibly

being arranged in a list sequence. There are various ways in which things can be (de)composed.

Figure 37 illustrates the various subtypes of composition relations that enable the expression of various different ways in which wholes can be decomposed.

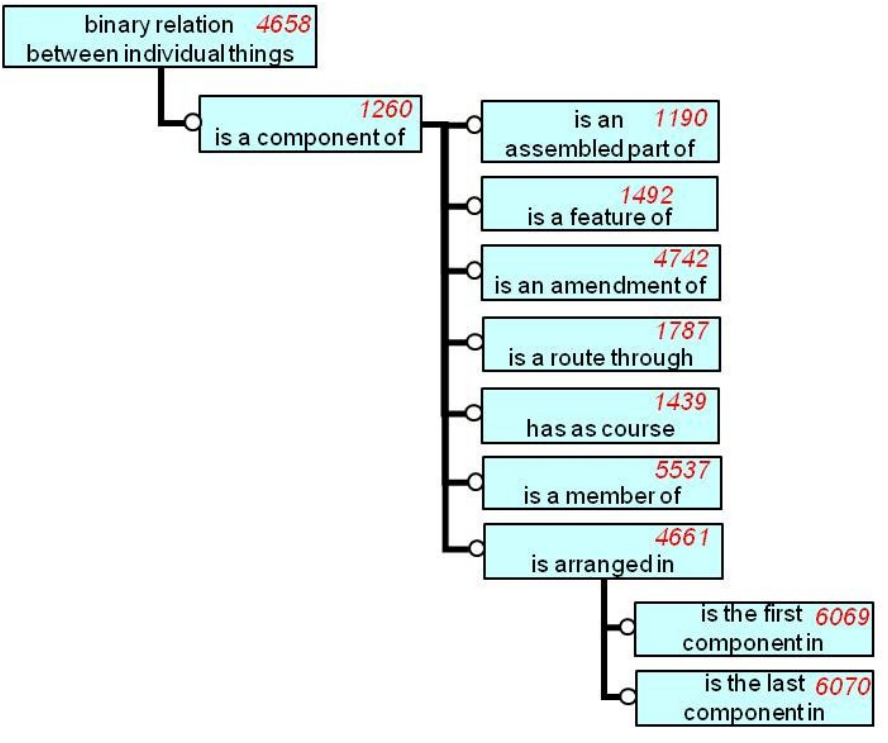


Figure 37, Composition relations

The various kinds of composition relations correlate with the various ways in which components are related to each other. The kinds of relations can express:

- A component is assembled in an assembly (1190). This implies that multiple components are connected, bound or welded together.
- Something is a physical feature of another thing (1492). A physical feature is an integral part of something, whereas it can be distinguished as a separate item, although the boundary between the

feature and the whole may not be well defined. An example of a physical feature is a rim or a lifting lug.

- A physical document is an amendment of another document (4742).
- A route is a part of a network (1787).
- A course is a part of a route (1439)
- A person or organization is a member of an organization (5537).
- A individual thing is component in a composed whole (4661) whereas the component is arranged in a pattern in a particular position. For example, an arrangement of components that are connected in a sequence, such as in a chain. The first and last component in a chain have a special position, which can be specified using the following subtypes:
- An individual thing is arranged in a first position in a sequence of components in a composed whole (6069).
- An individual thing is arranged in a last position in a sequence of components in a composed whole (6070).

There are separate kinds of relations for specifying relations between pairs of components in an arrangement, such as the arrangement of items in a sequence (see par. 7.10).

7.1.1 Extents, concentrations and recipes

A component or portion is always a fraction of the whole of which it is a part. A composition relation only indicates *that* the component is a part, but does not specify to what extent that is the case. For example, it might be the stated that a fluid consists of x %wt of a particular component, thus specifying the concentration of the component in the fluid. In fact this expression uses a more precise composition relation, expressing that the fluid consists of (has as part) a component that comprises x % of the mass of the whole fluid. In other words it is an 'x % part-whole relation'. This meaning can be expressed accurately by classifying the relation twice: first classifying it as a composition relation and secondly classifying its extent by a fraction or percentage on a scale. For example, the concentration of salt in a sample of seawater can be expressed in this

way and also the mass of a component relative to the mass of whole item, as is illustrated in the following table.:

Left hand object	Relation type	Extent	Unit of measure	Right hand object
Batch of seawater	has as part	3	wt%	Batch of salt
my car	has as part	10	wt%	my engine

Thus the extent, possibly quantified on a scale indicated by a unit of measure, expresses the extent that a composition relation is the case.
Also classification relations can be only partially the case, especially for a classification by substance, e.g. when classifying materials of construction or ingredients in a mixture. Then it may be that in fact only a part of the whole is classified that way. For example, when a material is classified by substance as being water, then usually it is not 100% pure water. The purity can be specified by stating e.g. that it is classified as 98% water. Furthermore, the impurities can be expressed by specifying e.g. that the material is classified as 1% salt. The following table presents such expressions in Formal English.

Left hand object	Relation type	Extent	Unit of measur e	Right hand object
Batch of seawater	is classified by substance as	98	wt%	water
Batch of seawater	is classified by substance as	1	wt%	salt

This mechanism can be used to specify e.g. recipes. When the components need to have separate identities, for example because there can be additional information about those components, then the composition relations with specified extent should be used. If the components are pure substances or natural ingredients, then it may be sufficient only specifying a number of classifications by substance.
The value can also be an upper or lower limit value for an extent. In such cases a subtype of the kind of relation should be used to express whether the value should be interpreted as a lower limit or as an upper limit value. For example the usage of the subtype kinds

of relations <has with a minimum ratio as part> and <is classified by substance as at least> indicate that a specified extent is a lower limit value.

7.2 Connections

Connections between physical objects can be specified in a simple way by a binary relation as:

A <is connected to> B

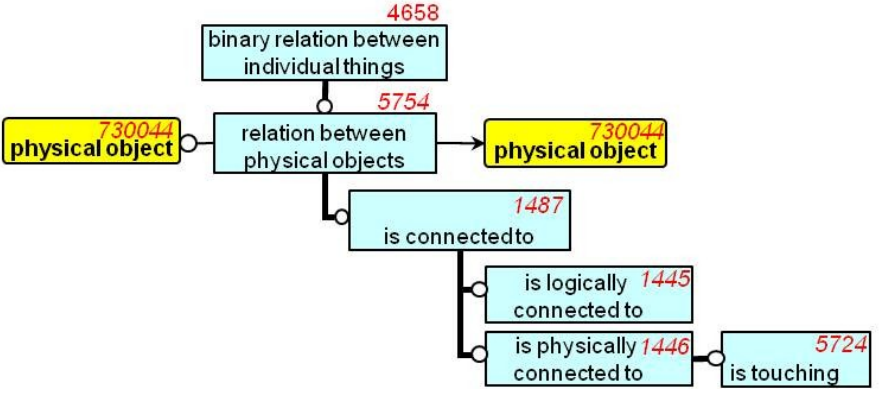


Figure 38, Connection relations

It may be expressed explicitly that physical objects are physically connected, but a connection may be logical, such as for example a funnel that is logically 'connected' to a drain, as it should be located under the drain, whereas the connected items may not be in physical contact. Also in telecommunication there are connections without physical contact. Thus connected items may physically be connected, but nevertheless they may not have a direct contact, such as when isolation material is used to avoid direct contact. On the other hand it may be required to explicitly express that two physical surfaces touch each other.

7.2.1 Connection assemblies

A specification that two physical objects are connected does not specify *how* those things are connected. How things are connected can be 730044
very complicated, because various kinds of connection materials

may be involved, such as fittings, flanges, gaskets, nuts and bolts or welds. The modeling of such connections typically requires the creation of a ‘connection assembly’.

A connection assembly is an assembly (an assembled physical object) that has as parts the ends of the items that are connected, together with the connection material(s). By making the parts and ends or physical features of the components explicit it becomes possible to specify details. For example, it can be specified which end is connected to which other end or which surface is touching which other surface. The parts, ends and surfaces must then be specified as distinct components of the connection assembly. For example, two pipelines are connected by a flanged connection that uses a gasket and a bolt set. A typical example of a specification of the details of such a connection is presented in the following connection model:

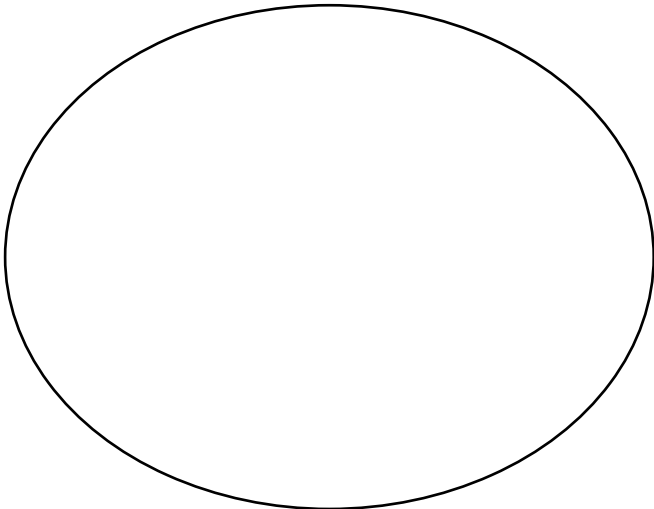
Pipeline-1	is connected to	Pipeline-2
Flange-1	is a part of	Pipeline-1
Flange-2	is a part of	Pipeline-2
Conn-1	is classified as a	flanged connection
Flange-1	is a part of	Conn-1
Gasket-1	is a part of	Conn-1
Flange-2	is a part of	Conn-1
Boltset-1	is a part of	Conn-1
Surface-1	is a feature of	Flange-1
Surface-G1	is a feature of	Gasket-1
Surface-G2	is a feature of	Gasket-1
Surface-1	is a feature of	Flange-1
Surface-1	is touching	Surface-G1
Surface-2	is touching	Surface-G2

Note that each flange is specified to be a part of a pipeline as well as a part of the flanged connection. Furthermore, a flanged connection is defined in Formal English as a subtype of a connection assembly (and not as a subtype of a connection relation). This enables to specify the parts of that assembly and features of those parts. Finally

that makes it possible to specify which surface is touching which other surface to make the real physical connection. Typically this decomposition is accompanied by a specification of the geometry of such a connection, which is usually presented as a drawing. The specification of geometry is beyond the (current) scope of this book.

7.3 Routes through networks

Routes and paths are physical branches that connect physical terminals or nodes in a physical network. For example a road has a role as a branch that connects its begin or ‘source’ physical object with its termination or ‘destination’ physical object and a fluid may follow a route through a network in which piping connects equipment. Each source or destination or intermediate node is a physical object in its own right similar to a connection assembly. Examples of a node are a roundabout or a power distribution terminal.



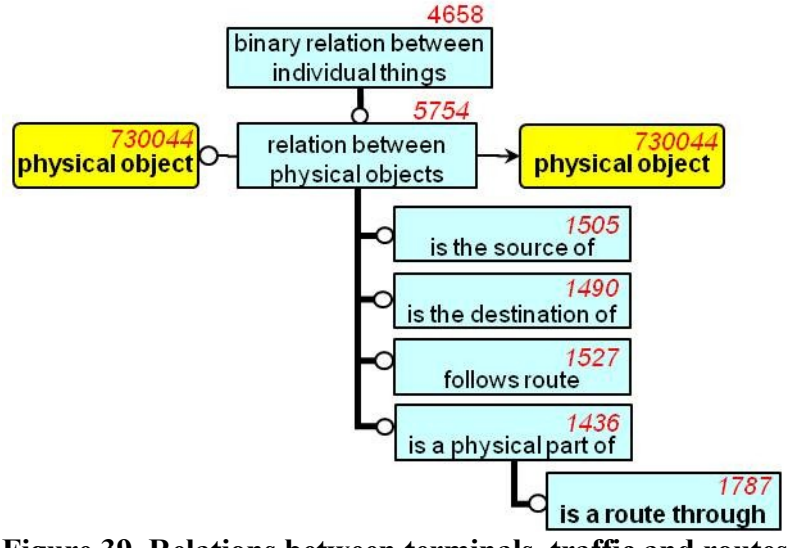


Figure 39, Relations between terminals, traffic and routes

Figure 39 illustrates the kinds of relations that can be used to specify the relation of a source and a destination physical object with a route, or part of a route. The figure also mentions a kind of relation to relate traffic to a route, by specifying that some object, such as a vehicle, follows a particular route from source to destination. Note that it may be required to define the retour route as a separate route.

Routes can be composed of sections. Each section behaves as a smaller route (see 'composition of a route').

7.4 Aspects of individual things

Physical objects have aspects, such as characteristics, qualities and properties. Most aspects are intrinsic to the object, such as its color, length, temperature or material of construction. Some aspects are extrinsic, which means that the existence of the aspect depends on the existence of a relation with another physical object. For example, it may depend on a role in a relation with something else.

Individual aspects are related to individual physical objects by a <possession of aspect> relation, which is usually denoted by the phrase <has as aspect>.

It should be noted that related objects are not aspects of objects. For example, many data models and systems define entities that have 'attributes' which often appear to be independently existing other things. Examples of such things are locations, manufacturers, parts, etc. Such things shall be related to the physical object by other relations than the 'possession of aspect' relation or one of its subtypes. Thus attribute is not a synonym of aspect.

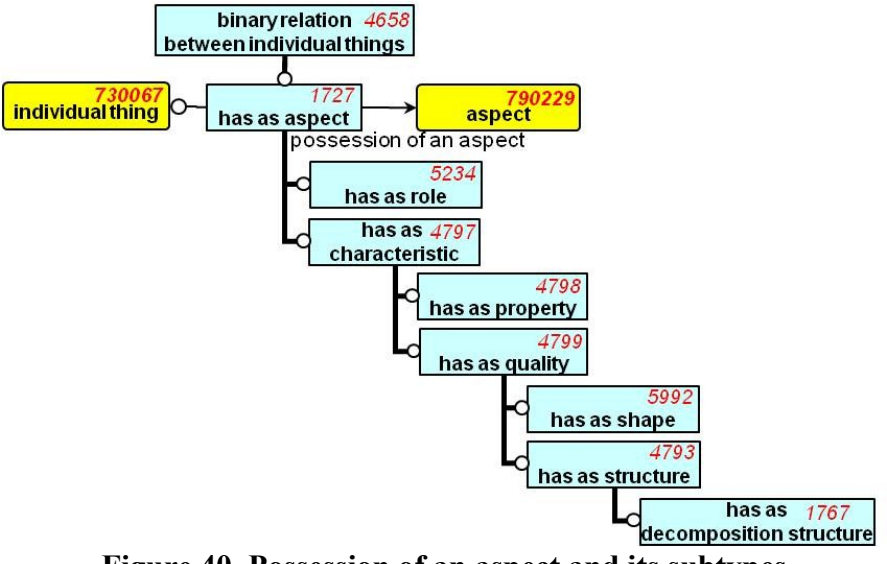


Figure 40, Possession of an aspect and its subtypes

Figure 40 shows that individual things can have individual aspects and are generally related by an expression, such as:

Individual thing-1 has as aspect A-1

Depending on the subtype of aspect it is possible to use a semantically more precise specification of the kind of relation. These kinds of relations are subtypes of the possession of aspect relation (1727) and thus form a hierarchy of kinds of relations.

The first subtype specifies that the individual object has a particular individual role. Such a role is an extrinsic aspect, which means that its existence is dependent on the existence of a relation with another object and it also means that the role is only possessed by the object during the time that it has the role.

The distinction between the subtypes of characteristic, being (physical) *properties* and *qualities* is based on the distinction between quantifiable properties and non-quantifiable qualities. For example, properties, such as diameter and temperature, can be quantified by a numeric value on a scale. For example, a property may be quantified as 30 mm or 37 degC. On the other hand, there are qualities, such as color, toxicity, flammability, being pregnant, etc. which are not (directly) quantifiable, but are qualified by values that are typically denoted by terms, such as red, green, toxic, non-toxic, etc. or by a category code, such as an RGB-code. Nevertheless some qualities can be correlated to quantifiable properties. For example, color can be correlated to frequency of light, which is a (quantifiable) property. Furthermore, properties may (also) be qualified by values that are denoted by terms provide a (rough) indication of their value. For example, a temperature may be qualified as hot or cold, without specifying the precise value.

> Note: in information science the qualitative values for qualities are often called 'literals', which usually means that users of systems are allowed to use free text to denote them. However, they are qualitative concepts that shall be (and are) defined in the dictionary of Formal English. Thus 'free text' is not allowed in a formalized language.

There are some relations for qualities for which in natural language dedicated terminology is used, which are also be adopted in Formal English. This holds especially for relations that express that something can have a particular kind of structure or that it can be made of a particular kind of substance, such as plastic, wood or steel (which is in fact is a statement that it can have a particular kind of atomic or subatomic structure).

The above described subtype kinds of relations can be used in a model for example as follows:

Object-1	has as role	Role-1
Object-1	has as property	Temp-1
Object-1	has as quality	Color-1
Object-1	has as shape	Shape-1

Object-1	has as aspect	Material of construction-1
Act-1	has as property	Duration-1

The above statements only specify that the object or occurrence possesses an individual aspect with a certain name, but not the nature of the aspect nor its quality or size. The latter requires that each of the individual aspects need to be classified and qualified or quantified on a scale (as is discussed in par. 8.1 and 8.2 respectively).

Instead of specifying that an individual object possesses an individual aspect, it is also possible to use a short-cut relation to relate an individual object directly to a qualitative aspect (which leaves the individual aspect implicit). For example, this can be done by stating that an individual object is cylindrical or that it is made of stainless steel. Such short-cut relations are further discussed in par. 8.1.5. The advantage of explicit individual aspects is that there is a smaller chance of ambiguity. Explicit individual aspects are explicitly classified, they allow for multiple individual aspects of the same kind and they enable that the same aspect has multiple qualification and quantifications in the course of time.

The latest example in the above table (about Act-1) illustrates that not only physical objects can have aspects, but also occurrences can have some aspects. In the example, the activity has a duration (a time aspect). An occurrence can also have a location in time, possibly indicated by a date of begin or end.

Note that it is a common mistake to erroneously state that occurrences have aspects, whereas in fact objects that are involved in the occurrences possess those aspects.

Furthermore, note that aspects cannot possess aspects. When aspects are related, then they are related by a correlation between aspects, which is a different kind of relation. For example, the temperature that marks the end of a temperature range is correlated to the range by an ‘end of range’ correlation. Similarly a color does not possess a frequency of light, but a color and a frequency of light may be correlated. Correlations are discussed in par. .

7.5 Occurrences and states

States are situations that last for some time. There are static states and dynamic states. In static states nothing changes from some macroscopic perspective, whereas dynamic states are states in which a change takes place. Therefore the latter are called occurrences or happenings.

Occurrences include human activities, business processes as well as physical and chemical processes, and events. Events are typically short during processes. Sometimes the duration of an event is very small and can be neglected, but by nature something cannot happen within a duration of zero. Nevertheless the period of occurrence (duration) of an event can sometimes be denoted by a single value, pointing to a duration within a second or part of a second.

7.5.1 Objects involved in occurrences

An occurrence, such as an activity, a process, an event or a happening, is a state that changes a pre-state situation into a post-state situation, whereas the change implies an interaction of involved things within the duration of the (dynamic) state.

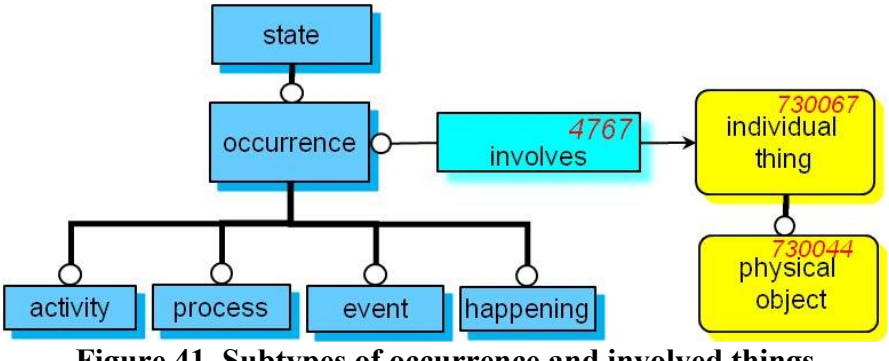


Figure 41, Subtypes of occurrence and involved things

Figure 41 illustrates some subtypes of occurrence, whereas there are numerous further subtypes of occurrence. Those subtypes are usually denoted by verbs in various forms, which all refer to the same kind of occurrence. For example, the terms acting, act, to act and action all denote the same concept. The ‘activities’ domain

Taxonomic dictionary provides a large taxonomy of kinds of occurrences.

Occurrences involve individual things, especially individual physical objects. Each involvement of an individual thing in an individual occurrence in a role of some kind can be expressed by a binary involvement relation, which is denoted by the phrase <involves> or its inverse phrase <is involved in>. For expressing the difference in roles that can be played by the involved things it is required that the involvement relation is specialized by defining a number of subtype kinds of relations.

Figure 42 illustrates a large number of standard subtypes according to roles that can be played by involved things.

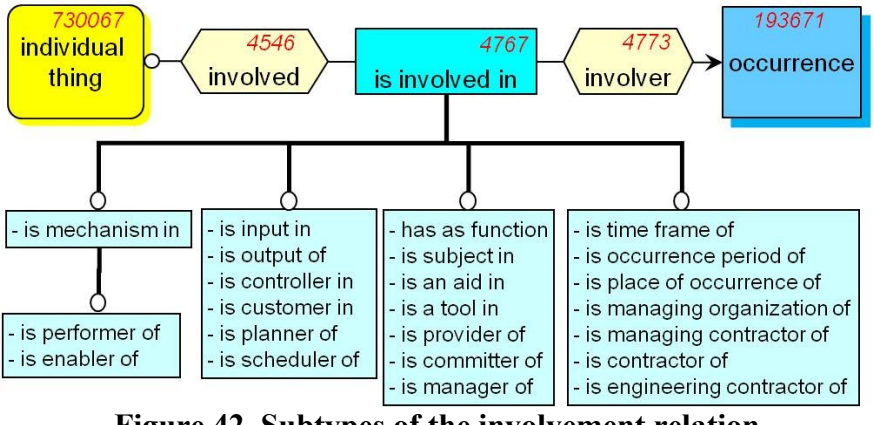


Figure 42, Subtypes of the involvement relation

Figure 42 also shows the 'involved' and 'involver' roles that are played by definition in such a relation. Each subtype of the involvement relation requires by definition a role of a particular kind that is played by the involved thing. Those kinds of roles are subtypes of the 'involved' kind of role. For example, mechanism, performer, enabler, input, output, manager and contractor are kinds of roles that have their own definition and position in a taxonomy of roles.

The kinds of roles can be used to find the proper kind of involvement relation. Further subtypes, other than the ones defined

in the in the TOPini section of the Taxonomic dictionary, can be defined as and when required.

7.5.2 Relations between occurrences and between states

Physical objects, solid, liquid as well as gaseous ones, can be in various states. This can be recorded by a binary 'being in a state' relation as is illustrated in Figure 43.

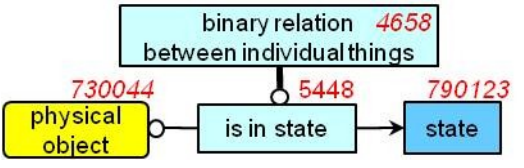


Figure 43, A physical object that is in a state

This also holds for aggregations of physical objects, such as systems and networks. Such states are typically temporal states during a particular period in time. The life of physical objects might be partitioned in a sequence of such states, also called lifecycle phases[7].

States and occurrences may be related to each other in various ways. Such relations may be between states of the same object, or may be occurrences in which the same object is involved, or it may be between states and occurrences in which different objects are involved. For example, during planning and scheduling of activities the sequence of activities is typically constrained by the persons or resources that are available and that cannot be involved in different activities at the same time.

Note that a state and an occurrence cannot occur twice, nor can it occur after itself. This is expressed in the definition of the relations between states and relations between occurrences by specifying a constraining relation (5960) that expresses that a state that has a role as relator in a relation between states cannot also have a role as being related in that same relation.

7 A physical object in a state is sometimes called a 'temporal part' of the physical object (for example in ISO 15926-2). A physical object in a particular state denotes the same physical object as the partitioned physical object. The aspects and relations of a temporal part are only applicable during the period of existence of the state (the lifecycle phase).

Figure 44 illustrates some relations between states and occurrences.

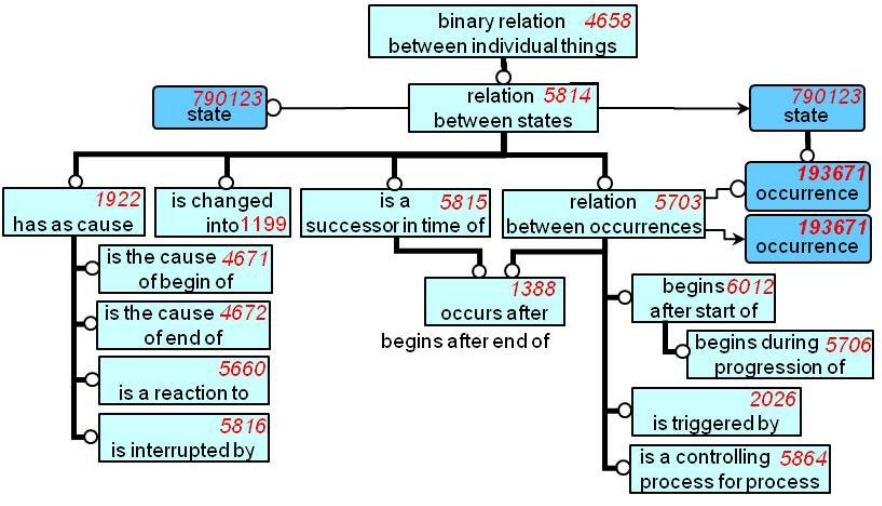


Figure 44, Relations between states and between occurrences

The following relations between states can be distinguished:

- State transitions: <is changed into> (1199).
  States may change over time. This is often described as state transitions, which are changes from a pre-state to a post-state situation.

- State sequences: <is a successor in time of> (5815).
  A succession of states in time describes a chain of states.

As occurrence is a subtype of state, Relations between occurrences (5703) are also subtypes of relation between states (5814), because occurrence is a subtype of state (an occurrence is a dynamic state). Therefore, the following kinds of relations are also subtypes of relations between states:

- Sequences of occurrences: <occurs after> (1388).
  Occurrences may appear in a particular sequence, where the succeeding occurrence starts after the termination of the preceding occurrence.

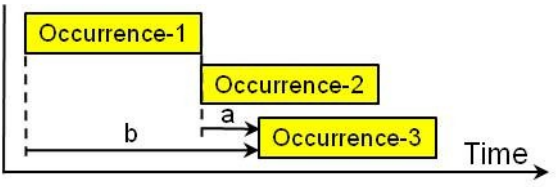


Figure 45, Sequences of occurrences

Figure 45 illustrates two situations: Occurrence-2 as well as Occurrence-3 starts after Occurrence-1 is terminated. To distinguish these two situations, a time gap (a) between the end of the preceding and the begin of the succeeding occurrence can be specified as a property of the relation. For example:

- Rel-1 has as aspect T-1
  T-1 is classified as a time gap
  T-1 has on scale a value equal to 5 min

Note that a negative value of the time gap means that the succeeding occurrence and the preceding occurrence occur partly in parallel.

This is equivalent to defining two subtypes as follows:

- <occurs immediately after>

This relation implies a time gap of zero duration. It represents a relations such as between Occurrence-1 and Occurrence-2 in Figure 45.

- <occurs some time after>

This relation expresses that the succeeding occurrence starts some time after the termination of the preceding occurrence. This represents a relation such as between Occurrence-1 and Occurrence-3 with time gap (a) in Figure 45.

- Relation between starts of occurrences: <begins after start of> .

The begin of a succeeding occurrence might be related to the start of a preceding occurrence. This corresponds with the relation between Occurrence-1 and Occurrence-3 with time gap (b) in Figure 45. This time gap can be specified in the same way as in the previous example.

Note that a time gap of zero in this case means that the two occurrences start at the same time and that a time gap greater than the duration of the preceding occurrence means that there is no overlap.

The constraint that the time gap is less than the duration of the preceding occurrence corresponds with the following subtype relation:

- <begins during progression of> (5706).

This kind of relation enables to specify that a succeeding occurrence begins during progression of the preceding occurrence, possibly without explicitly specifying the time gap.

- <is triggered by> (2026).

This kind of relation expresses that an occurrence is triggered by another occurrence. Typically the triggering occurrence is an event, being a short during occurrence. Typically the triggered occurrence is in fact only a creation process or a termination process as is illustrated in Figure 46 below. Such a creation or termination is a part of an occurrence. However, in many cases it is states that a whole occurrence is triggered by a triggering occurrence.

- <is a controlling process for process> (5864).

This specifies that a control process controls a controlled process. The details of how such a control is effectuated should be specified by a specification of the various sensing, controlling and actuating actions that are part of the control process, together with the signals and objects that are involved.

- Cause and effect: <has as cause> (1922).

An occurrence can be a cause of a beginning (creation) or a continuation after an interruption, or a termination of a state, which can be another occurrence.

- This is illustrated in Figure 46.

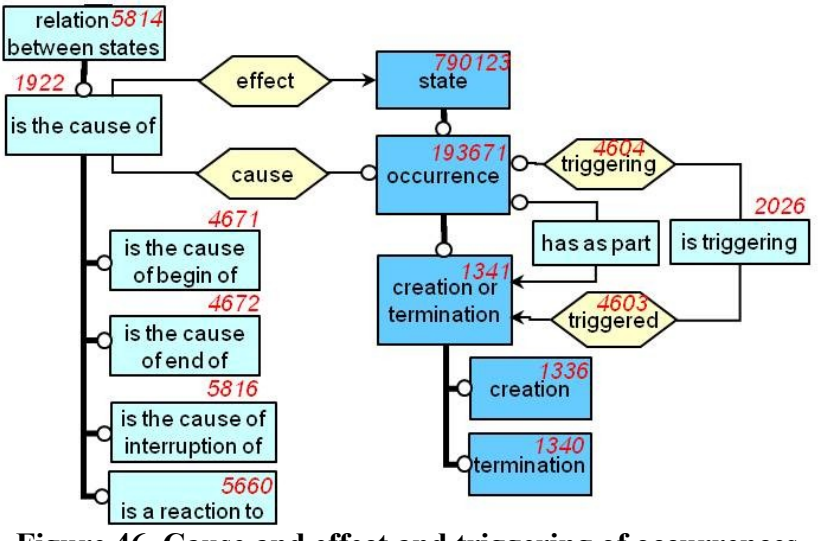


Figure 46, Cause and effect and triggering of occurrences

As a creation or a termination is a subtype of occurrence, which is a subtype of state, this kind of relation can be used to specify either that a whole state is an effect, or that a creation or a termination is an effect. The cause and effect relation has a number of subtypes that specify more precisely what the relation is between a causing occurrence and a state that is an effect:

- <is the cause of begin of> (4671)
- <is the cause of end of> (4672)
- <is the cause of interruption of> (5816)
- <is a reaction to> (5860)

These kinds of relations express that a state is caused to begin or caused to be terminated or caused to be interrupted by or is in some way a reaction to an occurrence.

7.5.3 Time of occurrences and states

States can be the case or occurrences can take place at a certain moment or within a period in time and that state or occurrence has a particular duration. A moment or period in time should be

distinguished from a duration. A duration is a property of a state or occurrence, which can be specified independent of when the state or occurrence takes place. On the other hand, a moment or period in time is always located in the course of time, thus refers to a calendar time.

Thus a state or occurrence has a duration. The duration can be expressed as a numeric value on a time scale (unit of measure). This is illustrated in Figure 47.

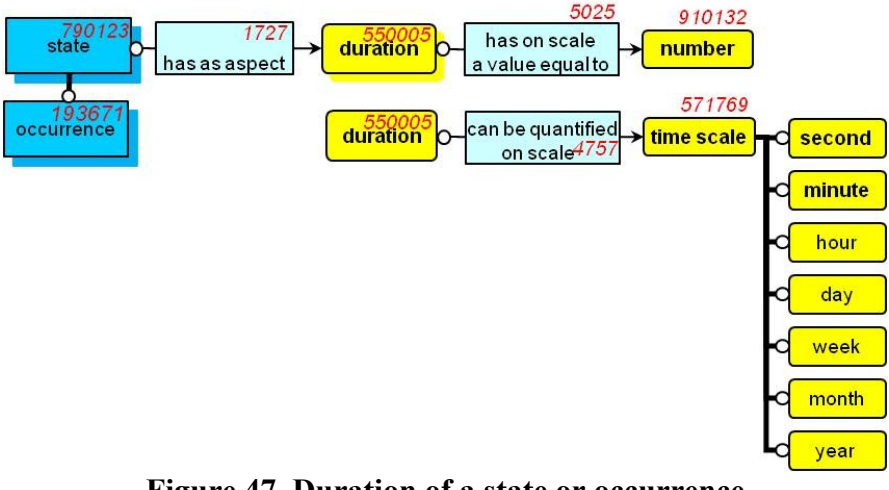


Figure 47, Duration of a state or occurrence

The quantification of the duration by a numeric value is discussed in paragraph 8.2.

For example, a duration of an occurrence, such as a particular repair activity can be expressed as:

Repair-1	has as aspect	D-1	
D-1	is classified as a	duration	
D-1	has on scale a value equal to	3.5	hour

Note that a duration can be quantified on a scale just as any other property. The lower part of Figure 48 therefore illustrates the specification of the fact that a duration can be quantified on a time scale and it provides a number of subtypes of time scale as are included in a formalized language dictionary.

A state or occurrence takes place within some period in time or at a point in time, which is located in a calendar time. This is illustrated in Figure 48.

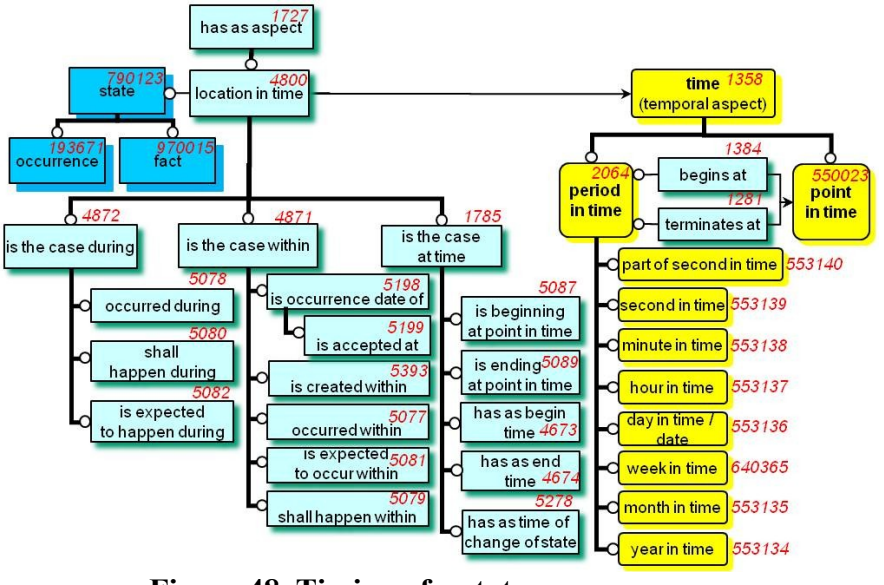


Figure 48, Timing of a state or occurrence

A point in time is a short period in time, referred to by a single value. A point in time typically has an undefined duration, whereas it is typically represented by a period within which something takes or took place or started to take place, such as second in time. A period in time has in principle a defined duration, which duration may be defined as the duration between two points in time.

There are three possibilities of how an occurrence or state relates to a period in time:

- The occurrence happens precisely during the period (4872). This happening can be a historic fact from the past (5078) or it can be a required happening during a period (partly or completely) in future (5080) or it can be an expected happening in a future period (5082), seen from the perspective of the moment of registration of the idea.
- The occurrence happens completely within the period (4871), whereas the period is larger than the duration of the occurrence. For example an occurrence that starts and stops within a day.

- The occurrence started before the beginning of the period and/or terminated after the end of the period (1785). Thus it is happening at least partly during the period.

The latter relation has two subtypes:

- The occurrence begins at a point in time and continues after that.
- The occurrence terminates at a point in time, and was ongoing before that.

A period in time or a point in time is always an individual period. For example, an individual period can be a particular day, such as 23 December 2013 at the Gregorian calendar. Such a date can have various roles in multiple relations with occurrences that take place on or during such a day. To facilitate that not every user of Formal English needs to allocate UIDs to dates, a generic rule is included in the language definition that states:

Dates and times

Thus dates are individual periods, denoted typically relative to a *time scale* which is normally called a calendar. Dates may be denoted in different notations, such as in a dd mmm yyyy or yyyy:mm:dd notation, etc. in calendar time, such as the Gregorian calendar. Various standard notations for times and dates are defined in ISO 8601.

Date:	2013-12-22
Combined date and time in UTC:	2013-12-22T21:08:07+00:00 2013-12-22T21:08Z
Week:	2013-W51
Date with week number:	2013-W51-7
Ordinal date:	2013-356

Table 35, Example date and time according to ISO 8601

Time periods are typically denoted relative to a local time zone in which an occurrence takes place. Within such a time zone, the time has a standard offset relative to the time scale called Greenwich Mean Time (GMT) or to a time scale called Coordinated Universal Time (UTC). For example, Nepal has a standard offset of +5:45

UTC hours, which means that a particular moment in time, the time is denoted in Nepal by a value to which 5:45 hours should be added to get the UTC time (which is nearly identical to the time in Greenwich during winter). This means that the date value should be accompanied by the time scale UTC+5:45.

To facilitate unambiguous references to the same dates in Formal English, each date should represent by a UID. This is done by defining the following rules:

- Dates (periods of a day) between the year 1500 and 3000 in the Gregorian calendar (also called the Western calendar) have by definition a UID between 15.0000.000 and 30.000.000, such that the first four digits denote the year, the second two denote the month in the year and the last two digits denote the day within the month.

  For example UID 20131223 represents the 23rd of December 2013.

- Furthermore, a month in a year is represented by a UID that is a number that ends with two zero's (as if it is day zero). For example, UID 20131200 represents the period December 2013.

- Finally, a year is represented by a UID that is a number that ends with four zero's. For example, UID 20130000 represents the year 2013.

The standard offset relative to GMT or UTC is in fact using another time scale that is defined by a shift relative to a standard time scale. To explicitly model the time scale, it can be recorded that a time value is a quantification on a particular time scale, such as UTC, UTC+1, UTC+5:45, etc. For example:

A-1 is beginning at point in time 2013:12:23:16:08 UTC+1

A date-time value may include not only seconds, but also parts of seconds. A convenient time scale for that is an epoch date system, which uses a rational number for the number of seconds (and parts of seconds) since the epoch date. Thus the date value (in seconds) has as unit of measure the epoch date system. For example the '2000

date system' [8] An example of the quantification of a moment in time in seconds since 1 January 2000 is:

T-1has on scale a value equal to 441806615,3 s since 2000

Note that the unit of measure, such as 's since 2000' should be defined in the dictionary.

8 See: http://en.wikipedia.org/wiki/Epoch_%28reference_date%29

7.5.4 Time of occurrences about physical objects

The period within which an occurrence about a physical object takes place, or the moment at which it takes place, is often not recorded as a timing of an occurrence, but as a relation between a period in time or date and the physical object that is subjected to the occurrence (or as an 'attribute' of the physical object), as is illustrated in Figure 49.

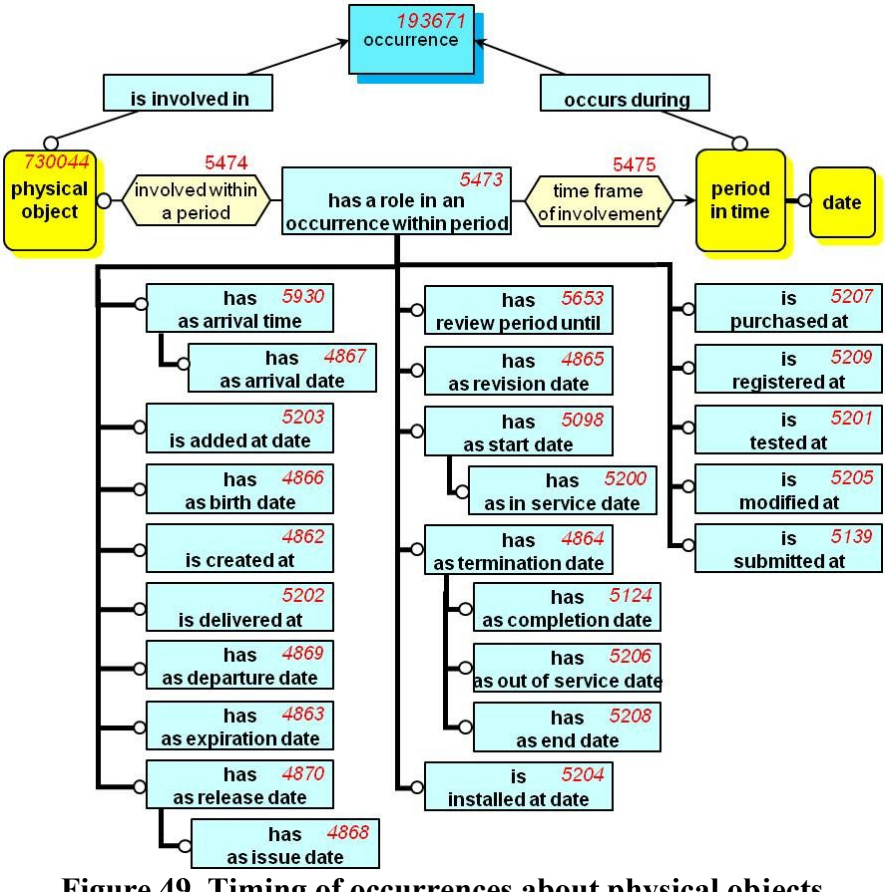


Figure 49, Timing of occurrences about physical objects

However, semantically a particular period in time or date can have various roles at the same time. For example, a particular date can be (have a role as) arrival date and birth date and release date, etc. of the same or different things together. Therefore, the timing of such

an occurrence in which a physical object is involved is expressed as a relation between the physical object and a period in time.

Examples of dates at which something about a physical object takes place are:

Train A has as arrival time 10.38 h

Thus, the role of the period in time (or of the date) can be used to find the proper kind of relation, and additional kinds of relation can be added as and when required.

Note that each kind of relation can be denoted by an inverse phrase, when the left hand and right hand objects in the expression are inversed. For example, the kind of relation <has as completion date> can be denoted by an inverse phrase as is used in the following expression:

3-3-2000 is completion date of Project X

7.5.5 Location of occurrences and states

States are the case and occurrences happen at a particular place or location. Such a place or location can be denoted as at a physical object or at a space. The relation that is used to specify that an occurrence takes place at a location that is denoted by a physical object is illustrated in Figure 50.

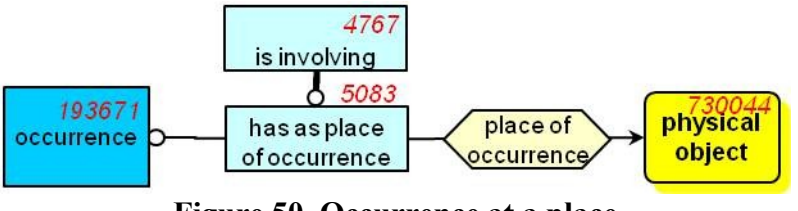


Figure 50, Occurrence at a place

The relation that is used to specify that a state is the case or an occurrence takes place at a location that is denoted by a space is illustrated in Figure 51.

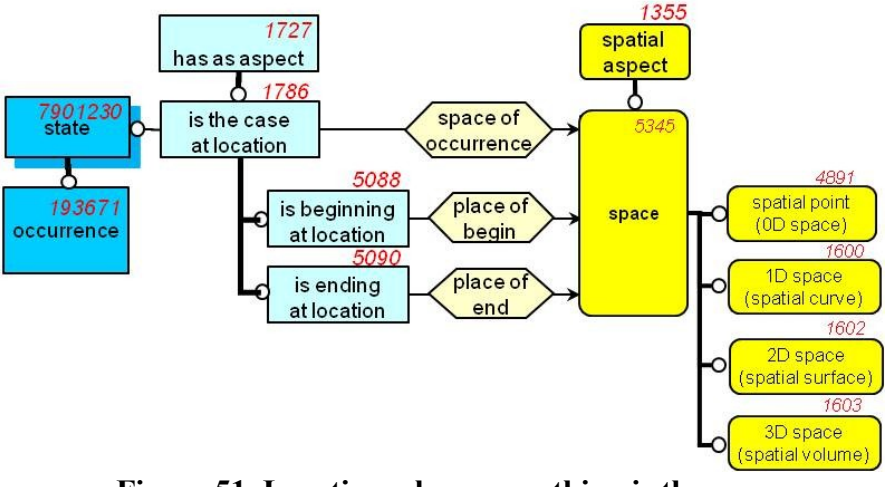


Figure 51, Location where something is the case

To denote that an occurrence takes place at a location it is more natural to use the synonym phrase <occurs at location>. The space can be denoted as being at a spatial point or along a line (a path), on a surface or in a volume.

7.6 Relations between objects involved in occurrences

Physical objects that are involved in an occurrence in different roles are often directly related to each other, without explicitly mentioning the occurrence. For example, it can be expressed that a particular company <is the manufacturer of> a particular individual physical object. This can be done by using such a phrase for the involvement relation or its inverse phrase as follows:

My car is manufactured by Volvo Sweden

Such a the relation implies that there has been some production process in which the party and the object were involved as manufacturer and as manufactured respectively. Thus although from such a relation it can be deduced by logic that there is or has been an

occurrence, that occurrence may remain implicit in the model. Thus such kinds of relations are short-cut relations for models in which the occurrence is explicit. Figure 52 illustrates a number of subtypes of such a kind of short-cut relation. The subtype kinds of relations relate objects that are both involved in the same occurrence, each in its own role.

Note that the roles can be played by physical objects, including social entities, such as organizations and families as well as individual persons. Furthermore, some kinds of relations, for example a custodianship may be regarding an individual thing that is not a physical object.

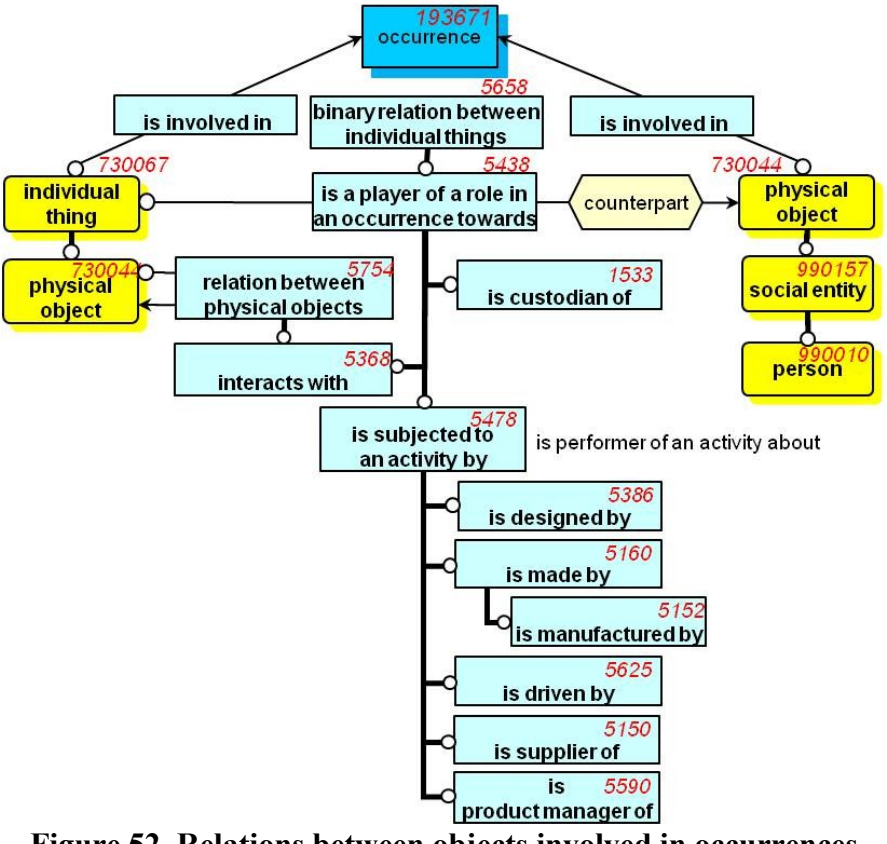


Figure 52, Relations between objects involved in occurrences

Short-cut relations are sometimes denoted by a phrase that is expressed from the perspective of the first role player in the relation and sometimes by an inverse phrase that is expressed from the perspective of the second role player in the relation. Therefore, the left hand object is either a performer or is subjected to the activity.

Further subtypes, other than the ones already defined in the upper ontology section of the Taxonomic dictionary, can be defined as and when required. It should be noted that these kinds of short-cut relations imply occurrences that may not appear explicit in the information model. However, if such an occurrence is made explicit, possibly at a later stage, then the objects that are involved in that occurrence are not automatically recognized as being involved in that occurrence.

7.7 Facts caused by acts

The short-cut relations of the kinds that are mentioned in the previous paragraph are by definition caused by an occurrence of particular kinds. The definitions of the kinds of short-cut relations define by what kind of occurrences they are caused. In a modeled definition this is specified by a relation of the kind <is by definition caused by a>. For example, the fact that my car is manufactured by an organization by definition implies that that fact is caused by a manufacturing activity. This can be expressed as follows:

Name of left hand object	Name of kind of relation	Name of right hand object
is manufactured by	is by definition caused by a	manufacturing

A number of examples of such relations between facts and acts are given in Figure 53.

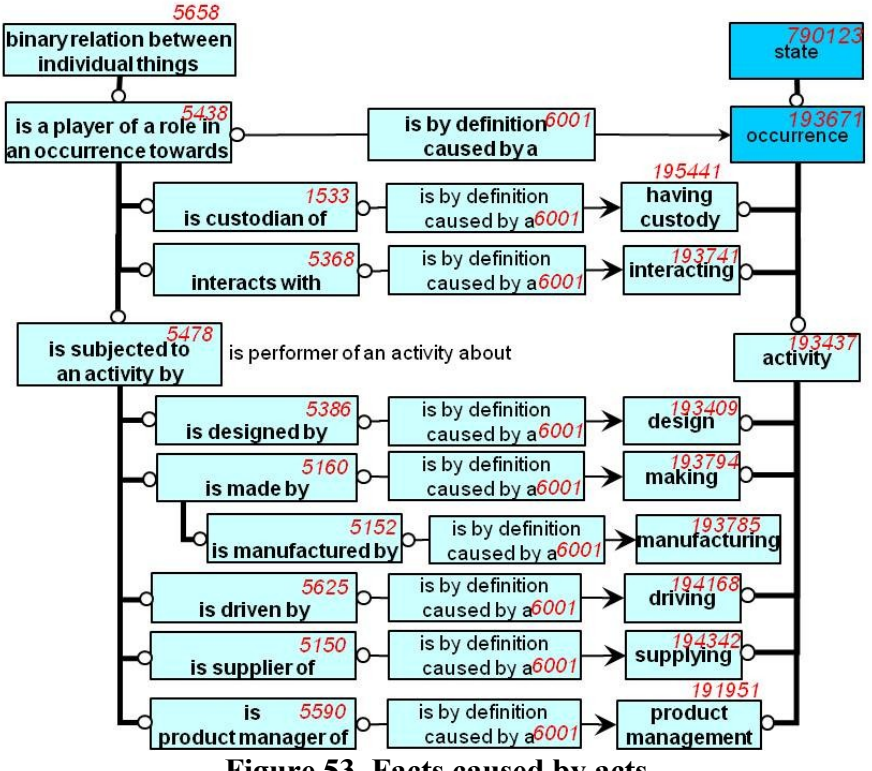


Figure 53, Facts caused by acts

Relations between the kinds of relations and the causing kinds of occurrences are included in the upper ontology section of the Formal English Taxonomic dictionary to provide a basis for logic reasoning either to derive an occurrence and its classification, or to verify the consistency between short-cut relations and occurrences that are explicitly included in an information model.

7.8 Purposes and objectives

The reasons why people create things are the purposes for the existence of things. There are also objectives why occurrences should take place. This objective can be in order to achieve a particular state or in order to prevent a particular state.

The kinds of relations that can be used to express such purposes are presented in Figure 54.

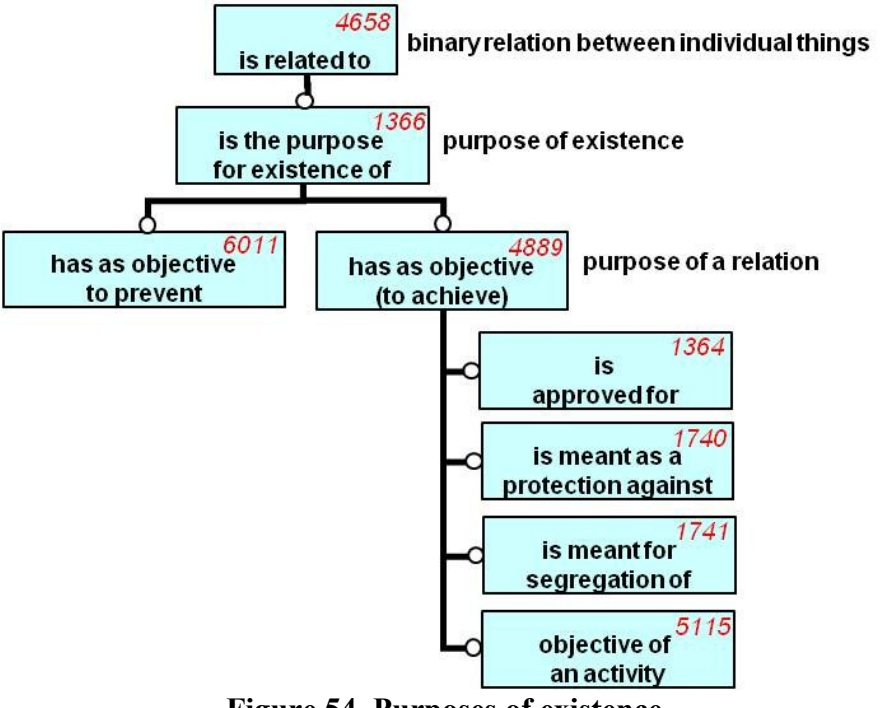


Figure 54, Purposes of existence

Note that the 'possession of purpose' is a relation between something and a state that has a role as purpose. The thing that has a purpose of existence can also be a relation, or an occurrence (which is modeled as a higher order relation). For example a connection relation A-B has as objective to achieve that 'A is connected to B' (which is a state).

7.9 Collections

A collections of things is a plural object (with a UID) in which the collected things have no particular role, apart for possibly being arranged in a sequence. The collected things are assumed not being connected. There are various kinds of collections, dependent of the kinds of things in the collection: collections of individual things, collections of kinds and mixed collections. Different kinds of relations are to be used to express relations with different kind of collections. This illustrated in Figure 55.

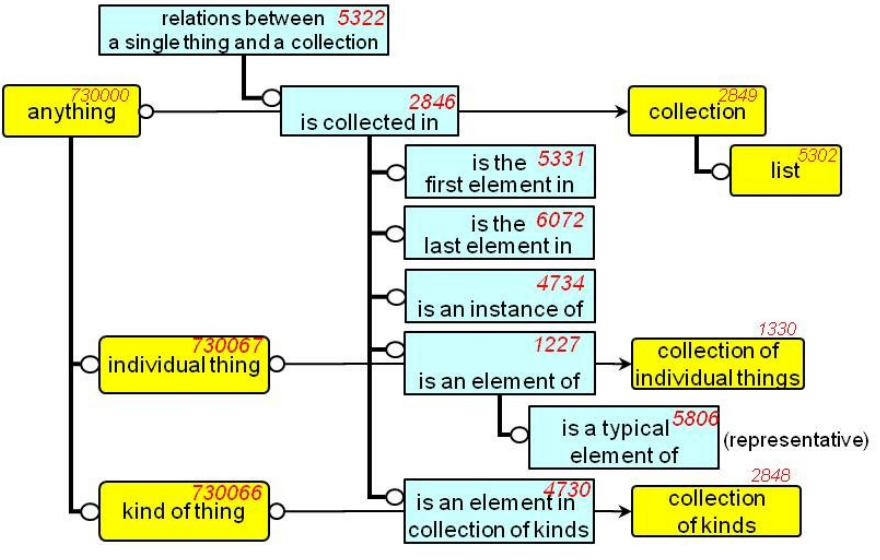


Figure 55, Specifying a collection

Figure 55 illustrates how things can be explicitly declared to be elements in a collection. The generic collection relation (2846) can be used for specifying elements in any kind of collection (with individuals, kinds or both). This also holds for its first three subtypes. The first and the second subtype express a particular position of an element in a collections in which the elements are arranged in a sequence, such as in a list. Note that any particular collection needs to be classified as a collection or as one of its subtypes, such as list, row, stack, etc. and may be classified as a collection of individual things or as a collection of kinds.

By using the <is an element of> relation (1227) it can be specified that the collected thing is an individual things and that the collection is constrained by being a collection of individual things. If one of the two is not consistent with other information about the related things, the software should generate an error message.

Note that a collection of individual things has an identity (UID) and may consist of a number of physical objects, each with their own identity. Those physical objects might be put in some kind of bag or package. Then the filled bag or package is a new whole with another identity. This bag or package is not a collection, but it is a composed object that is a composition of the bag and (the elements in) the collection.

Usage of an <is an element in collection of kinds> relation (4730) implies the constraint that the collected thing should be a kind and the collection is a collection of kinds only.

Apart from definition collections by their elements, there are various other relevant kinds of relations between single things and collections. A number of them are presented in Figure 56.

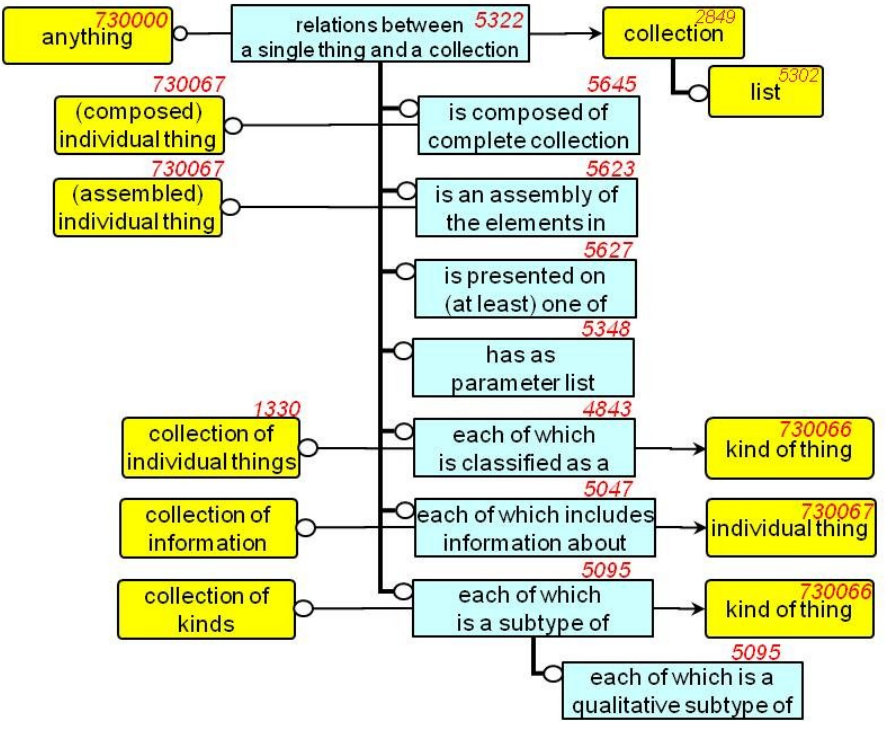


Figure 56, Relations between single things and collections

Sometimes it is required to specify that a collection is complete for the composition of a particular composed object. This means that the necessary and sufficient elements for a particular purpose are present in such a collection. This can be expressed using relation 5645 which is a relation between a complete collection and an object that is or can be composed from the collection. For example a complete inventory may specify that a collection consists of all the components that are necessary to make a particular assembly.

The relation <is an assembly of the elements in> (5623) specifies that each element in the collection is or is intended as a part of the indicated assembly.

The relation <is presented on one of> (5627) expresses that something is presented on at least one of the elements in a collection. For example it may be expressed that some object is

presented on at least one of the documents in a collection of documents.

The relation <has as parameter list> (5348) expresses that a particular list is a parameter list for some object. Typically a list for some mathematical function or software module.

For the definition of the other kinds of relations, see the Gellish taxonomic dictionary.

7.10 Sequences and location of things in space

The position or location of an individual thing in space, such as the location of a physical object or an aspect, can be expressed with a location relation between the located thing and the physical object that acts as a reference. This is illustrated in Figure 57.

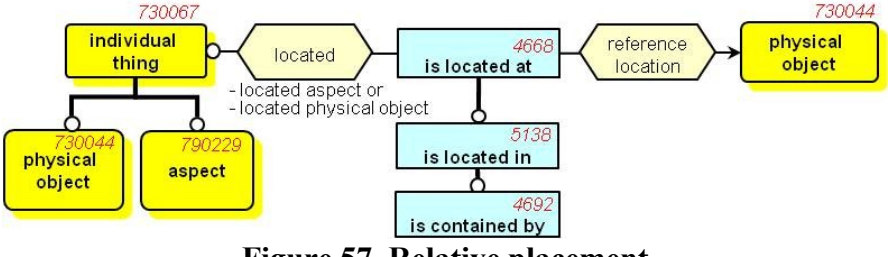


Figure 57, Relative placement

The reference location can be a large physical object or a small one, such as a physical point. This relative placement is specified by using a simple binary relation. The accuracy of the location depends on the size of the reference object. The relation does not specify the precise position within the reference object, not does it specify the orientation of a located physical object. The location of an object in a coordinate system is described in the next paragraph.

Elements in a sequence

Another kind of arrangement includes that a location is specified as a relative position in a sequence, without explicitly specifying an absolute position. For example, it can be stated that individual things (A through N) are at particular positions in a sequence. This implies that those individual things that are arranged are elements in a collection (comprising the things in the sequence). The latter can be

made explicit by declaring for each thing that it is an element in the collection. This is described in par. 7.9. The combination of being element of a collection and being arranged in a sequence is illustrated in Figure 58.

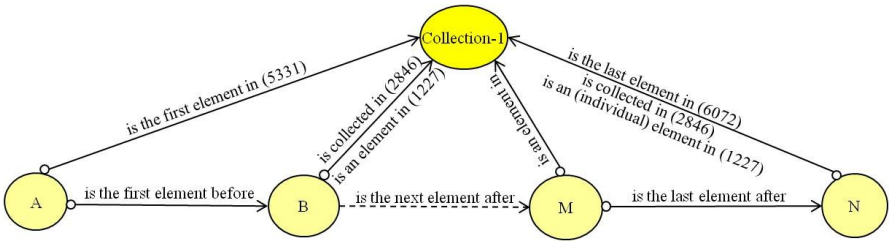


Figure 58, Elements of a collection arranged in a sequence

To specify the complete sequence, in principle it is sufficient to specify for each pair in the sequence which one is next to the previous one. However, to be specific on the completeness of the chain it can be valuable to specify which one is the first one in the chain and which one is the last one after the one but last one. Therefore, as shown in Figure 59, the sequence relation (5332) has subtypes for the first and last element.

Thus a sequency relation specifies that:

- A component is the next component after another component (5332).

The first and the last component in a sequence requires the use of the following subtypes of a sequency relation:

- Being a first element before a specified other element in a sequence (5932).
- Being the last element after a specified other element in a sequence (5338).

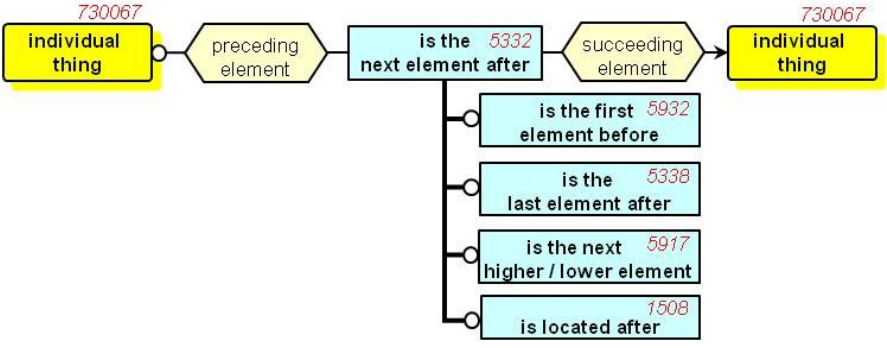


Figure 59, Arrangement of individual things in a sequence

Figure 59 also shows two other subtypes, one indicating that a sequence is ordered from low to high and a relation that specifies that a sequence is an arrangement (a location) in space.

A specification of a sequence can be accompanied by a specification that the elements in the row belong to a particular collection. This can be specified by declaring for each thing that it <is collected in> a particular collection as described in par. 7.9.

7.11 Binary logic relations

Figure 60 presents a number of binary relations that originated in logic and mathematical set theory. They are applicable to semantic modeling as they define specific characteristics of various categories of binary relations. Such characteristics can be used to formally deduce consequences from statements that use kinds of relations that are a subtype of one or more of these kinds of binary relations.

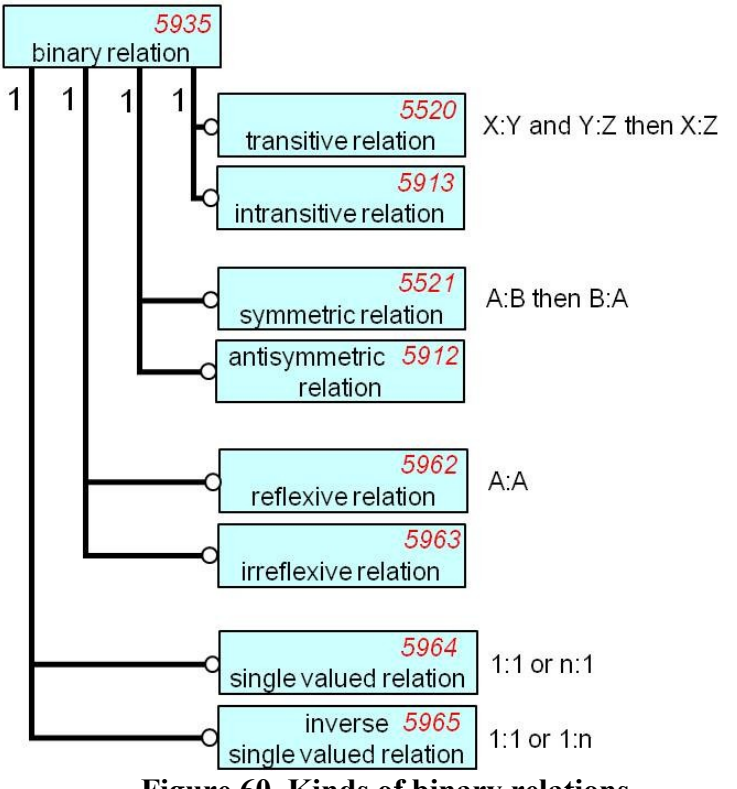


Figure 60, Kinds of binary relations

The last two kinds of relations, single valued relation and inverse single valued relation, are well known in data modeling, but should be used with care, because they are specialized versions of the more general cardinality constraints specifications as are applicable for relations between kinds of things (see chapter 10). An expression of a binary relation between kinds of things has in principle left hand as

well as right hand simultaneous cardinality constraints. An n,m right hand cardinality pair of constraints means that one individual of the kind at the left hand side may have minimally n and maximally m number of individuals of the kind at the other side at the same time for the specified kind of relation. However, those constraints may vary between possibilities, requirements and definitions about the same kind of relation. Furthermore, it should be noted that in Gellish they are defined as *simultaneous* cardinality constraints, which means that they hold for relations that have a validity period that overlap in time, but the constraints do not apply for relations that do not overlap in validity period.

7.11.1 Parent-child relations

A typical example of a transitive and irreflexive relation is an ancestor relations and its further subtypes as presented in Figure 61.

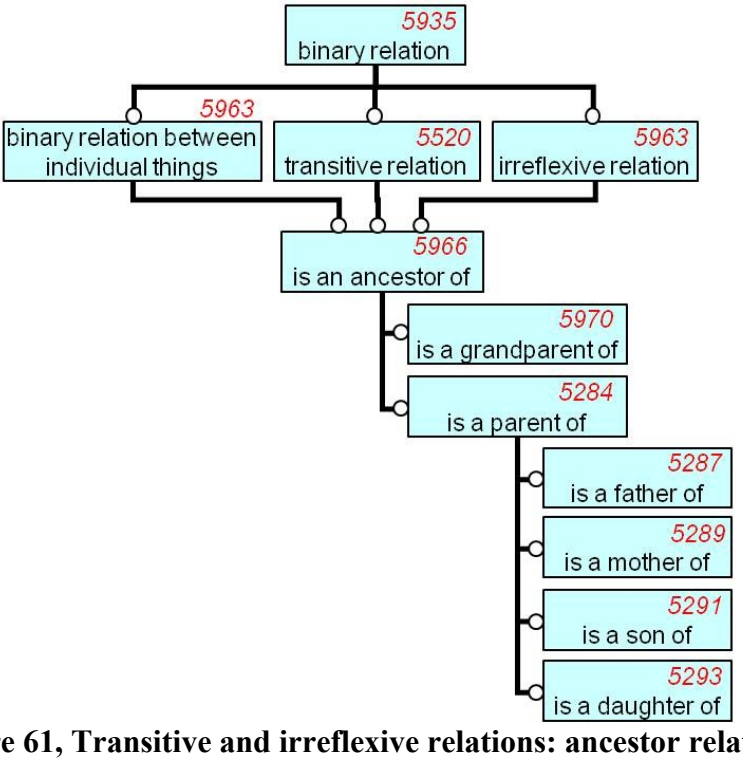


Figure 61, Transitive and irreflexive relations: ancestor relations

An ancestor relation (<is an ancestor of>, 5966) is defined as a relation chain between individual things that indicates that the person that has a role as ancestor has as descendant the person that has a role as descendant.

The ancestor relation is therefore a subtype of ‘binary relation between individual things’. However it is also a subtype of ‘transitive relation’ and a subtype ‘irreflexive relation’. This means that conclusion can be drawn from a chain of such relations.

For example, if some expressions in Formal English state the following:

A	is an ancestor of	B
B	is an ancestor of	C

Because it is included in the definition of Formal English that the ancestor relation is defined as a subtype of transitive relation, therefore from formal logic it can be deduced that

A	is an ancestor of	C

Nevertheless the latter statement is not explicit in the Formal English expressions.

Furthermore, it is included in the definition of Formal English that the ancestor relation is defined as a subtype of irreflexive relation. Therefore from formal logic it can be deduced that B is not an ancestor of A, etc. This expressed in Formal English by a denial, as follows:

denial:	B	is an ancestor of	A
denial:	C	is an ancestor of	B

As well as:

denial:	C	is an ancestor of	A

The multiple supertypes of the ancestor relation are inherited to the subtypes of that relation. Thus similar conclusions can be drawn from expressions about parents, grandparents, father, mother, etc.

7.12 Correlations

7.12.1 Correlations between individual aspects

The values of various aspects of things are often dependent on the values of other aspects of the same thing or of other things. In such cases the aspects are correlated.

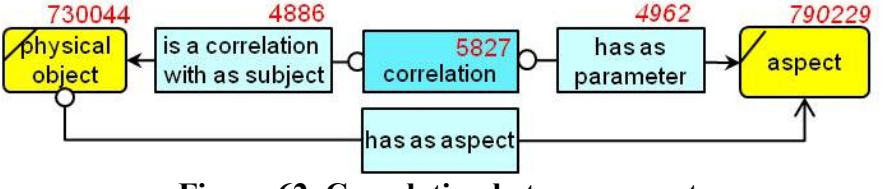


Figure 62, Correlation between aspects

Typically such an individual correlation between individual aspects can be modeled as a higher order relation between multiple aspects, whereas the aspects act as parameters in the correlation. Furthermore, such a correlation is often concerning (aspects of) one individual physical object. Then that physical object is called the subject of the correlation.

For example, an individual physical object B-1 with a mass M-1 is subject to a force F-1, which result in its acceleration A-1. The physical law that is discovered by Newton states that there is an individual correlation C-1 between that mass, force and acceleration. Such an individual correlation can be classified as being an exemplar of a known conceptual correlation, such as the law of Newton. Thus for example: F-1 = M-1 * A-1 is classified as a F = m * a correlation. The above ideas can be expressed as follows:

C-1	is a correlation with as subject	B-1
C-1	has as parameter	M-1
C-1	has as parameter	F-1
C-1	has as parameter	A-1

The classifications of these individual things specify what kinds of things they are:

C-1	is classified as a	law of Newton
B-1	is classified as a	rocket

M-1	is classified as a	mass
F-1	is classified as a	force
A-1	is classified as a	acceleration

The first four of the above expressions state that the aspects are correlated, but not how they are correlated. The classifications of each of the individual things specify what the things are, whereas the classification of the correlation as a 'law of Newton' refers to a kind of correlation which definition specifies *how* the parameters are correlated.

The above is an example of the typically case that a correlation in an individual case is a special case of a general law. The definition of a general law, such as the law of Newton, can be modeled as a conceptual correlation between kinds of aspects. The latter is discussed in par. 10.9.

7.12.2 Comparison of characteristics

Sometimes statements are made about the relative value of two different characteristics. This can be done with or without quantifying the characteristic. These are kinds of binary correlations in which aspects are compared to each other regarding their value or declared to which extent they are the same or different from each other.

The various ways in which characteristics can be compared are illustrated in Figure 63.

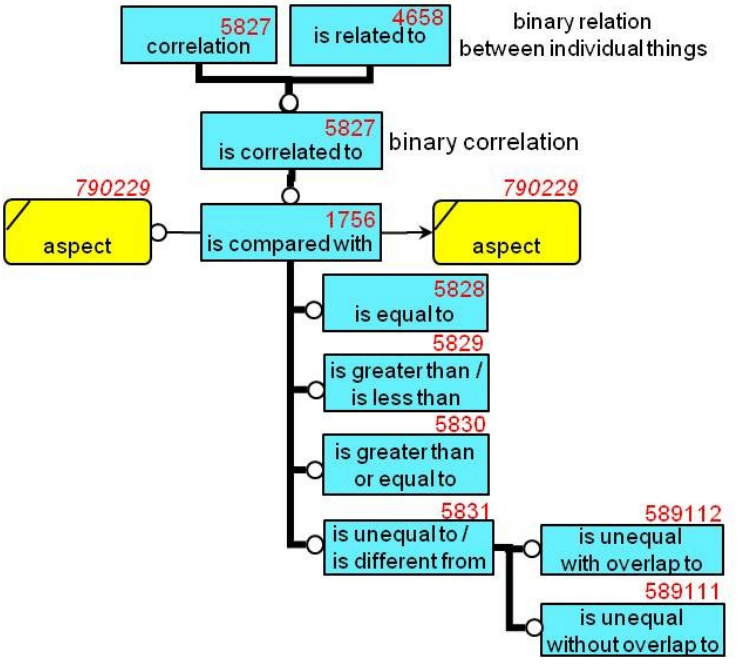


Figure 63, Comparison of characteristics

For example, it may be stated that:

risk-1	is less than	risk-2
distance-1	is greater than	distance-2
range-1	is unequal with overlap to	range-2

The kind of relation <is greater than> with its inverse <is less than> and the kind of relation <is unequal with overlap> can be meaningfully accompanied by an extent to which they are greater or less than or to which extent they overlap (see par. 7.1.1)

7.13 Positioning of objects in coordinate systems

The location of individual things specified more precisely when they are positioned in a coordinate point that is defined as part of a coordinate system (a property space), as is illustrated in Figure 64.

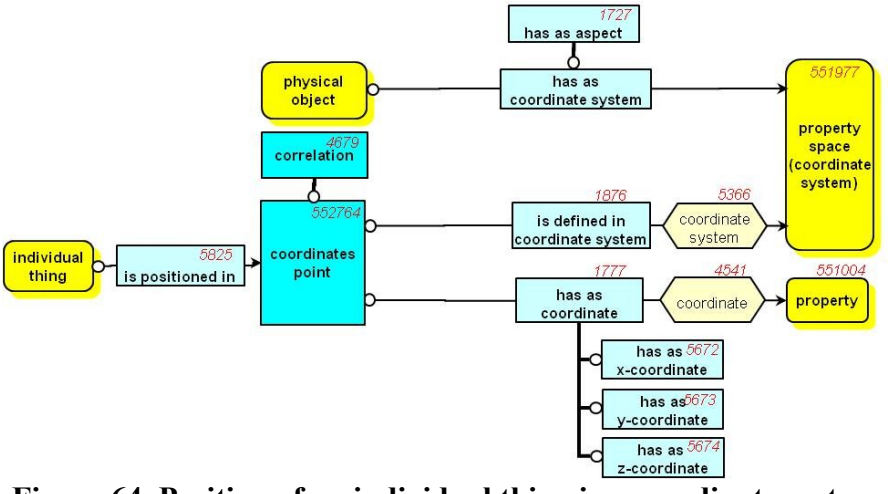


Figure 64, Position of an individual thing is a coordinate system

When a physical object is positioned in a coordinates point, it means that the origin of a coordinate system in which the shape of the positioned physical object is defined is located on a coordinates point in a coordinate system of a reference physical object. To identify the proper coordinate system it is possible to specify which physical object is the reference object for which the coordinate system is defined. Furthermore, the coordinates point has one or more properties as its coordinates, such as distance in x-direction, orientation angle, latitude, etc.

The following table provides an example of the specification of the coordinates of the Eiffel Tower.

Name of left hand object	Name of kind of relation	Name of right hand object
the earth	has as coordinate system	UPS coordinate system
The Eiffel Tower	is positioned in	Coordinate pt-1
Coordinate pt-1	is defined in coordinate system	UPS coordinate system
Coordinate pt-1	has as coordinate	Longitude-1
Coordinate pt-1	has as coordinate	Latitude-1
Coordinate pt-1	has as coordinate	Elevation-1

Table 36, Specification of coordinates

Each of the individual things in the above example needs to be classified, whereas the properties need to be quantified by a numeric value on a scale. For example, the coordinates of Greenwich, which is the reference longitude for the UPS coordinate system, can be expressed as follows:

Name of left hand object	Name of kind of relation	Name of right hand object	UoM
UPS coordinate system	is classified as a	coordinate system	
Coordinate pt-1	is classified as a	coordinate point	
Longitude-1	is classified as a	longitude	
Longitude-1	has on scale as value	0, 0, 0	deg, min, sec
Latitude-1	is classified as a	latitude	
Latitude-1	has on scale as value	51, 28, 38,	deg, min, sec
Elevation-1	is classified as a	elevation	
Elevation-1	has on scale as value	500	m

Table 37, Quantification of coordinates

A coordinates point typically is a three-dimensional relation between angles and/or distances, but in other coordinate systems it can be one or two dimensional. The number of dimensions (the rank) of the coordinate system can be specified, as well as the scale (unit of

measure) for each dimension or for all the dimensions of the coordinate system as a whole.

Note that a triple of values (numbers, separated by comma's) to denote a value of an angle in degree, minutes and seconds respectively, is regarded as one qualitative value with one UID. This notation allows for negative values. Another notation where the numbers and units of measure are combined in one string, together with a character to denote the northern or southern hemisphere is also allowed. For example, the value 51° 28' 38" N denotes the longitude of Greenwich at the northern hemisphere and 51° 28' 38" S is a point at the same angle on the southern hemisphere.

8 Relations between an individual thing and a kind

A relation between an individual thing and a kind of thing can be either a classification relation or it tells something about the individual thing by relating it to a kind, without specifying the other individual thing of that kind. The latter typically means that the individual thing is related by a relation of a particular kind to some individual thing that is classified by that kind.

For example, the statement that a particular building B-1 has an elevator, is a statement that specifies that some individual thing has a part that is of a particular kind, without identifying that individual part. This can be expressed as follows:

B-1 has a part that is classified as a elevator

This relation is in fact a short-cut relation that implies that there is some object that is a part of B-1 and that is classified as an elevator, although that object may not appear in this part of the model. However, it might well be that that individual part appears to be explicit elsewhere in the model. The verification of the consistency of the model is further discussed in par. 8.4.1.

Figure 65 presents some examples of kinds of relations in this category.

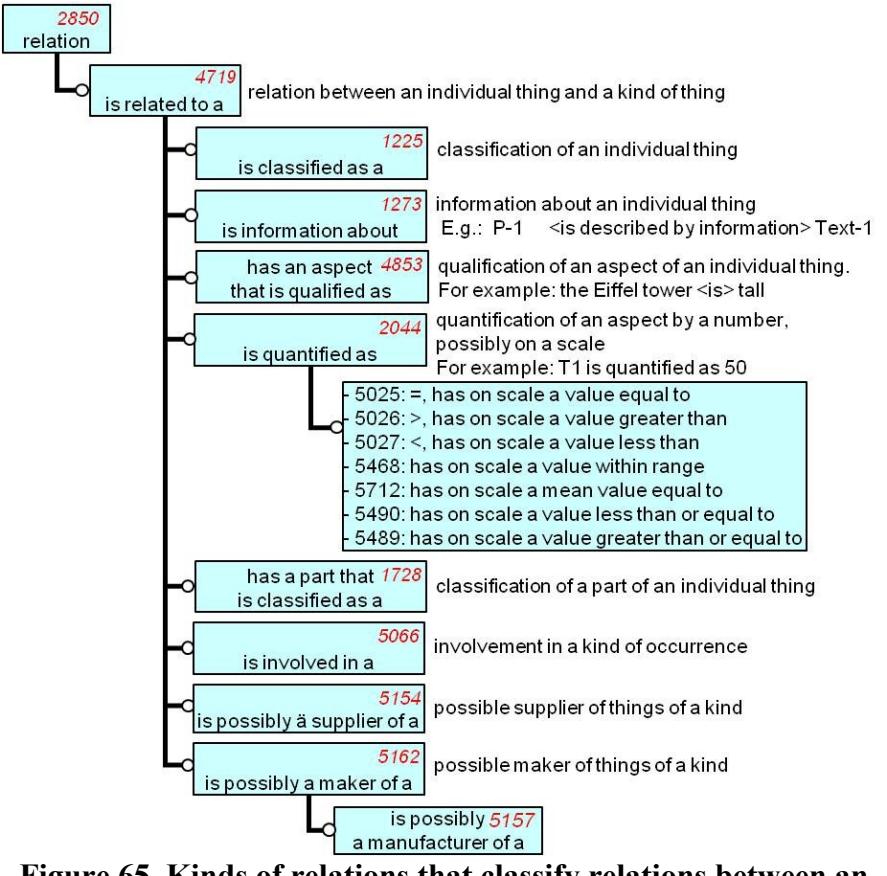


Figure 65, Kinds of relations that classify relations between an individual thing and a kind

Typically these kinds of relations are denoted by phrases that start with <is> or <has>, whereas the phrases include a role and terminate with <a>. Exceptions on this rules are kinds of relation about qualitative aspects (see next paragraph) that are denoted without a preceding <a>. This is the case because qualitative aspects, which include pieces of information (qualitative information), are defined as kinds of things.

Note that the list of kinds of relations in Figure 65 is incomplete. Each of the kinds of relations has or can have further standard subtypes and

can get additional subtypes as and when required. The following paragraphs discuss the categories and their subtypes in more detail.

8.1 Classification

Classification of individual things is a prime means to enable interpretation of what the nature of the individual things are. Therefore, for a proper interpretation it is required that each individual thing is classified. This holds for all categories of things, such as physical objects, aspects, roles, occurrences, relations, etc.

Classification is a relation between an individual thing and a kind that states that the individual thing is characterized by that kind. The interpretation is only possible when the kinds (also called concepts) are defined in a taxonomic dictionary.

Figure 66 presents a number of subtypes of the classification relation.

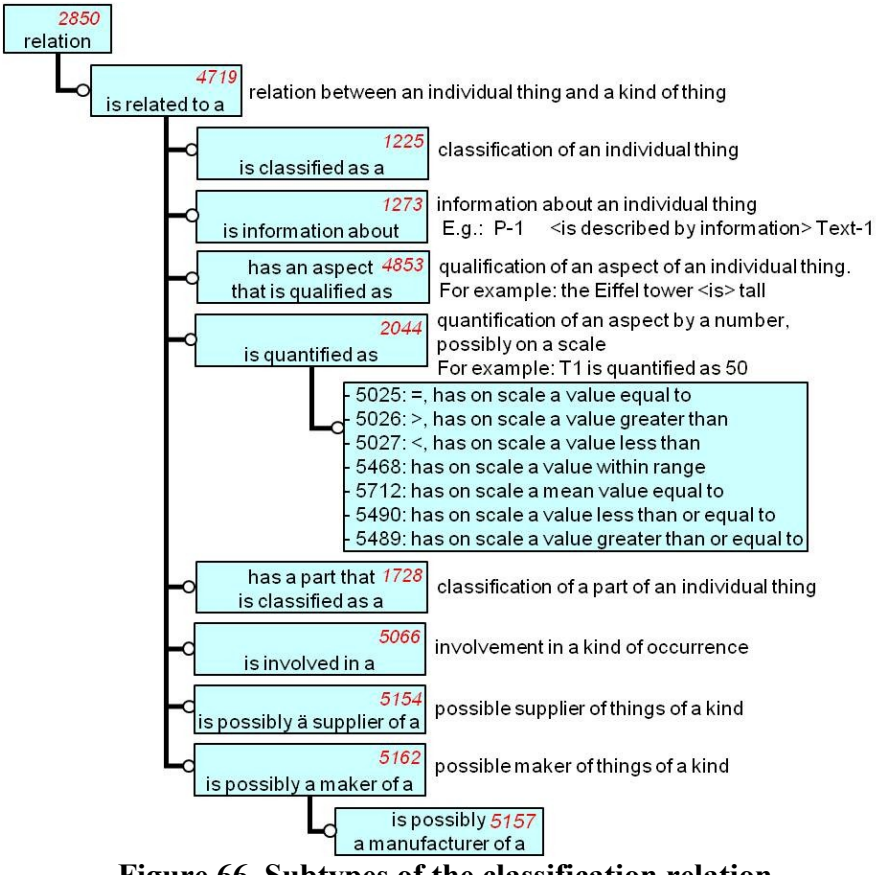


Figure 66, Subtypes of the classification relation

The subtypes are further discussed in the following paragraphs.

8.1.1 Classification by nature

The ordinary classification relation (<is classified as a>, 1225) usually classifies an individual thing by its nature or characterizing aspects. It tells what kind of thing it is. Especially solid items are typically classified by their nature, which for artifacts is determined by the characteristics of the classified items such as their shape and construction materials that make an object suitable as performer or enabler of a particular kind of activity or process (also called its function).

Note that a classification of an individual thing by a kind implies that the individual thing is implicitly also classified by all the supertypes of that kind as defined in the taxonomy. This makes that classification by a more specialized kind expresses more knowledge about the individual thing, which knowledge implies the more generic classifications.

Examples of classifications of physical objects, aspects, roles of physical objects or parties and occurrences are:

Obj-1	is classified as a	building
Obj-2	is classified as a	room
Prop-1	is classified as a	velocity
Prop-2	is classified as a	color
Role-1	is classified as a	tool
Role-2	is classified as a	supplier
Act-1	is classified as a	meeting
Proc-1	is classified as a	corrosion

The richness of the taxonomic dictionary determines the possible classifications by predefined kinds and the extent to which new kinds need to be defined in a project.

8.1.2 Classification of aspects of implied parts

In many cases aspects of components are modeled as if they are aspects of a higher level of assembly and without explicit modeling of the component. For example, assume that for a pump P-1 it is specified that it has an inlet diameter D-1. Then D-1 is often treated as a property of the pump, without modeling the inlet explicitly as a separate component. Then D-1 will not be classified just as a diameter (which would suggest that it would be a diameter of the pump), but it will be classified as an ‘inlet diameter’.

This example illustrates that in such cases the aspects are not classified by a normal kind of aspect, but by a kind of intrinsic aspect. A kind of intrinsic aspect is a role of an aspect that is by definition possessed by a particular kind of physical object. Therefore kinds of intrinsic aspects have names that include the name of the kind of physical object of which it is an aspect.

Typically the kind of physical object is a component in an assembly. Other examples of kinds of intrinsic aspects are pipe diameter and wall thickness. These are apparently a diameter of a pipe and a thickness of a wall. However, the components, such a pipe or wall, are often not identified or specified explicitly. Semantically, intrinsic aspects are roles of aspects, so that they are subtypes of 'role'. Therefore, the classification of an aspect by a kind of intrinsic aspect is in fact a classification of an aspect by role (see also par. 8.1.4). Thus, examples of classifications of aspects by a kind of intrinsic aspect are:

Prop-1	is classified as a	pipe diameter
Prop-2	is classified as a	wall thickness

Or semantically more precise:

Prop-1	is classified by role as a	pipe diameter
Prop-2	is classified by role as a	wall thickness

The definition models of kinds of intrinsic aspects relate the kinds of intrinsic aspects to the kinds of physical objects. For example, such a model includes a relation that expresses that a

pipe diameter is by definition an intrinsic aspect of a pipe.

Note that the whole phrase is the name of a kind of relation. The definition of kinds of intrinsic aspects is further discussed in chapter 9.

8.1.3 Classification by substance

Batches or streams of fluid are not only classified by their nature as a batch or stream, but typically they are also classified by the substance (stuff) that composes the fluids. For example:

Str-1	is classified as a	batch of liquid
Str-1	is classified by substance as	paint

Similarly, solid items are often classified by a material of construction (stuff) from which the things are made. This is also a classification by substance, although typically the phrase <is made of> is used for such a classification relation. For example:

P-1	is classified as a	bolt
P-1	is made of	stainless steel

The phrase <is made of> does not refer to a historic production process but merely relates the actual state of P-1 to the stuff from which P-1 is constituted.

8.1.4 Classification by role

Physical objects are often classified by the role they play or the role they are intended to play. Kinds of physical objects can be distinguished from kinds of roles, because a role is not an intrinsic aspect, but an extrinsic aspect. A role is only played as long as a physical object is functioning or installed and in use or operating. When a physical object is taken out of its operational context, for example by putting it in a warehouse, then you cannot interpret its role from its intrinsic aspects. The same holds for persons and organizations. For example student is not a kind of person, but a role of a person and customer and supplier are nor kinds of parties, but roles of parties.

Thus a taxonomic dictionary should also include kinds of roles so that physical objects can be classified according to their kind of (intended or actual) role.

The phrase <has a role as a> is therefor used as a synonym of the phrase <is classified by role as a>.

Examples of classification by role are:

John	has a role as a	student
John	has a role as a	project manager
John	has a role as a	customer
Str-1	has a role as a	input
Str-1	has a role as a	output

8.1.5 Qualification of aspects by value (qualitative aspects)

To specify what an individual aspect is, it is necessary to classify it by the nature of the aspect. For example, an individual aspect, such as A-1, can be classified as a length, diameter, temperature, color, toxicity, number of items, etc.

However, aspects are not only classified by their nature, but they can also be qualified, quantified, compared to or constrained by a qualitative aspect. Such a specification of a constraint is in fact a categorization or classification of the aspect by taking the size, magnitude or severity of the aspect as a criterion and relating it to a qualitative value of such an aspect. Therefore, the specification of a constraint is defined to be a subtype of classification.

A qualitative aspect is a kind of aspect value, which is typically included in a list of allowed values, from which a constraining value is selected.

The qualification of an aspect implies a qualification relation with a qualitative aspect, as is illustrated in Figure 67.

Figure 67, Classification and qualification of an individual aspect

A qualitative aspect is a value that may qualify an aspect. The value can be non-numeric or numeric. Qualitative values are also part of the formalized language and thus they are typically also included in the taxonomic dictionary and can be added as and when required.

If a value is non-numeric, then the value is typically denoted by a character string that is a term or name for a 'textual value'. For example, the above kinds of aspects (also called conceptual aspects) can have qualitative values, such as many, tall, hot, red and toxic.

Examples of classifications and qualifications of aspect are:

T-1	has aspect	H-1
T-1	has aspect	C-1
H-1	is classified as a	height
H-1	is qualified as	tall

C-1	is classified as a	color
C-1	is qualified as	red

A qualitative aspect is a concept that is typically denoted by a name consisting of a character string. However, a special case of qualification is a qualification by a quantitative property value (which is identified by a UID) which is in fact a quantitative value on a scale, in which case the number and the scale are not separated as two distinct things. In such cases the quantitative value is denoted by a character string that includes one or more numbers as well as a scale. For example, one may specify the following:

H-1	is qualified as	'500 mm'
T-1	is qualified as	'37 degC'

Or a surface area is specified as follows:

A-1	is qualified as	'3 x 5 inch'

Therefore it is a concept that can have a modeled definition which relates the value to the appropriate number(s) as well as to the scale (as is discussed in par. 9.5). Such qualitative values may also be included in a formalized language dictionary.

However, the above method will result in a 'combinatorial explosion' in the dictionary, because of the nearly unlimited amount of combinations of numbers and scales. Therefor, in Gellish it is preferred that a qualification by such a quantitative aspect is replaced by a quantification relation between the (individual) property and a number, with a separate specification of the scale that is used for the quantification. This is further discussed in paragraph 8.2.

Because of this possible quantitative nature of the constraining qualitative aspect, the comparison of an individual aspect can be with a point value, but also with a range. And a constraint can be that an aspect value is equal, unequal, approximately equal, higher or lower than the point value and it can be within or outside (not within) a range. For those porposes we need separate kinds of relations.

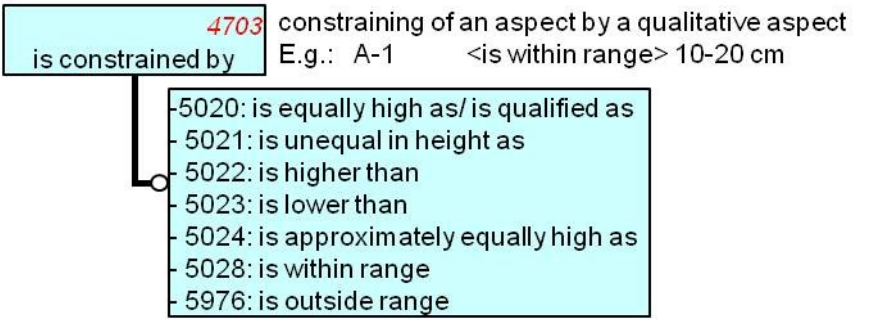


Figure 68, Comparison of aspects with qualitative aspects

Note that the range in Figure 68 has a value (with an UID) that is denoted by the character string, such as in the example '10-20 cm'.

8.1.6 Classification of collections

The classification of a collection means that the collection as a whole is classified by a kind of collections. For example:

C-1 is collectively classified as a collection of aspects

Such a classification is semantically different from a relation between a collection and a kind with the meaning that each element in the collection is of that kind.

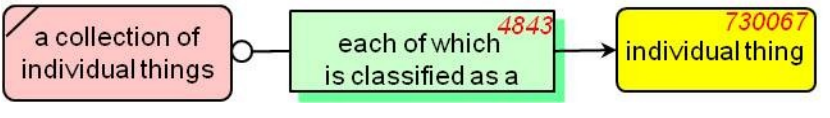


Figure 69, Collective classification

The latter relation implies a classification of each element in a collection, without mentioning or even recognizing every element in the collection. An example of a combination of the two relations is:

C-2 is collectively classified as a collection of items
C-2 each of which is classified as a M6 bolt

8.2 Quantification of properties on scales

Aspects, especially physical properties, such as diameter, temperature, etc., can not only be qualified (e.g. as high or low), but normally they are quantified by relating them to a numeric value on a scale.

If an individual aspect is a numeric quantity, then the relation between the individual aspect and the quantitative value (typically a number) is a quantification relation (2044). For example the number of bolts in our stock <is quantified as> 15.

This is illustrated in Figure 70.

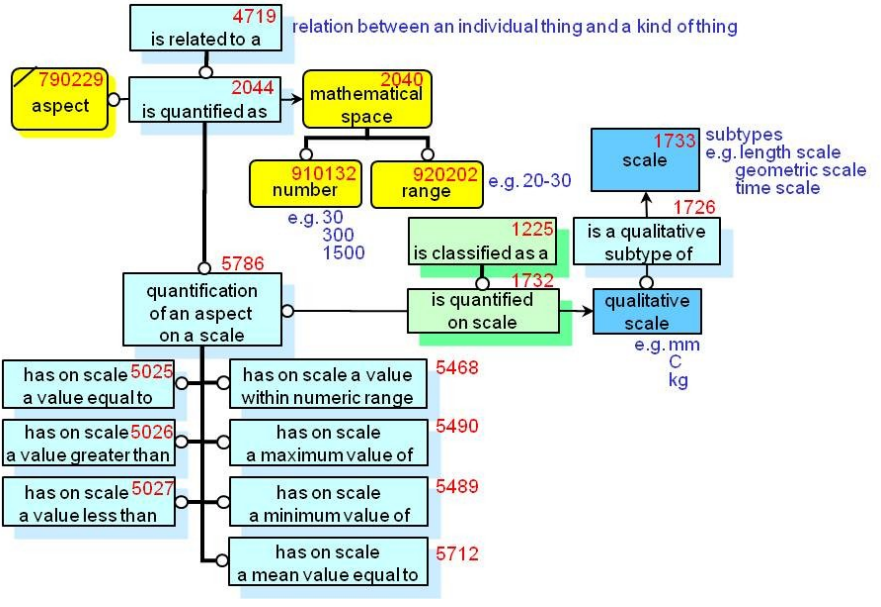


Figure 70, Quantification of aspects on scales

The quantification of a number of things is usually considered to be a quantification without using a scale. However, this can also be considered to be a quantification on the scale 'piece', because it is also possible to quantify a number of pairs or dozens, thousands, moles, etc. For example:

N-1	is quantified as	2	

This is equivalent to:

N-1	has on scale a value equal to	2	piece

Similarly:

N-2	has on scale a value equal to	3	dozen

Thus aspects are typically quantified by a number on a particular scale (5786). The use of a quantification relation in those cases implies that a specification has to be provided of the qualitative scale (unit of measure) that classifies the quantification relation. Thus the relation between the aspect and the number is not only classified as a quantification relation on a scale, but that relation is also classified as a quantification on a particular scale. In other words, the relation with the scale is a second classification of the quantification relation. A qualitative scale is also called a unit of measure, but it denotes a method to relate a property to a number. For example 'deg C' indicates a method created by Anders Celcius about how to allocate numeric values to measured temperatures.

Furthermore an aspect can be quantified as equal to, greater than or less than the numeric value. It is also possible that the value of an aspect is variable in time and/or space[9], which implies that there are also avarages over time and/or space. This means that we need relations that enable to relate to values that are averages, or better: values for average properties, such as a daily average temperature.

Examples of quantifications of aspect by a number on a scale are:

H-1	has on scale a value equal to	300	m
T-1	has on scale a value greater than	30	deg C

Note that an aspect, such as the height H-1, can be quantified as well as qualified, as is described in the previous paragraph.

As said before, the value of an aspect can also vary over time or space. For example, when a property is measured at some sampling rate by a measuring device, such as a thermometer, it may create many measured values at different moments in time. In most cases those values can be recorded as discrete values for which it is indicated that they are measured at different moments in time. This is illustrated in Table 38 in which the same property (T-1) has different values, which is recorded as different ideas about facts, each of which has a different validity period.

9 For details about the modeling of varying values over time and space is discussed in the book 'Semantic Modeling Methodology' (Ref. 2)

Name of left hand object	Name of kind of relation	Name of right hand object	Unit of measure	Date-time of start of validity	Date-time of end of validity
T-1	has on scale a value equal to	25	deg C	t1	t2
T-1	has on scale a value equal to	30	deg C	t2	t3

Table 38, Property values that vary over time

Note that conventional systems usually allow for only one value for a property, whereas they may even *define* a property as something that has only one value. However, we follow the normal practice in technology that properties can have multiple values in time (and space). Thus a property, such as T-1, does not change when its value changes.

The date-time of start of validity and end of validity determine the period during which the statement is valid, thus for example they determine the period during which a property has a value that is considered equal to the specified qualitative (or quantitative) value. When the two date-time values are equal it is assumed that the validity period lies within the period determined by that single specified value. Often such a time stamp is seen as a moment in time without a duration. However, even the specification of a second or piece of a second implies some minimal duration, such as the duration of a second.

The specification of these date-time values are in fact separate binary ideas, called contextual facts, as is further described in paragraph 7.5.3.

Quantification by property value ranges

In some cases a property is quantified on a value range. This can be specified by two quantification statements: one that specifies that the property <is quantified as greater than or equal to> a specified value and a second statement that the property <is quantified as less than or equal to> another specified value at the same time. This has as disadvantage that it is always required to search for a second

statement. It is simpler to specify that a property has a value within a range. For example by a statement such as:

T-1 has on scale a value within numeric range 20-30 degC

A numeric range is a qualitative value (a kind of aspect) that has to be defined quantitatively in the dictionary. The definition of numeric ranges is discussed in paragraph 9.5.

8.3 Textual information about individual things

A piece of textual information, called qualitative information, can be related to an individual thing or can be related to some kind of thing. The upper half of Figure 71 presents several kinds of relations that relate an individual thing to a piece of qualitative information. The lower half presents kinds of relations that can be used to relate anything to a piece of information, including also individual things.

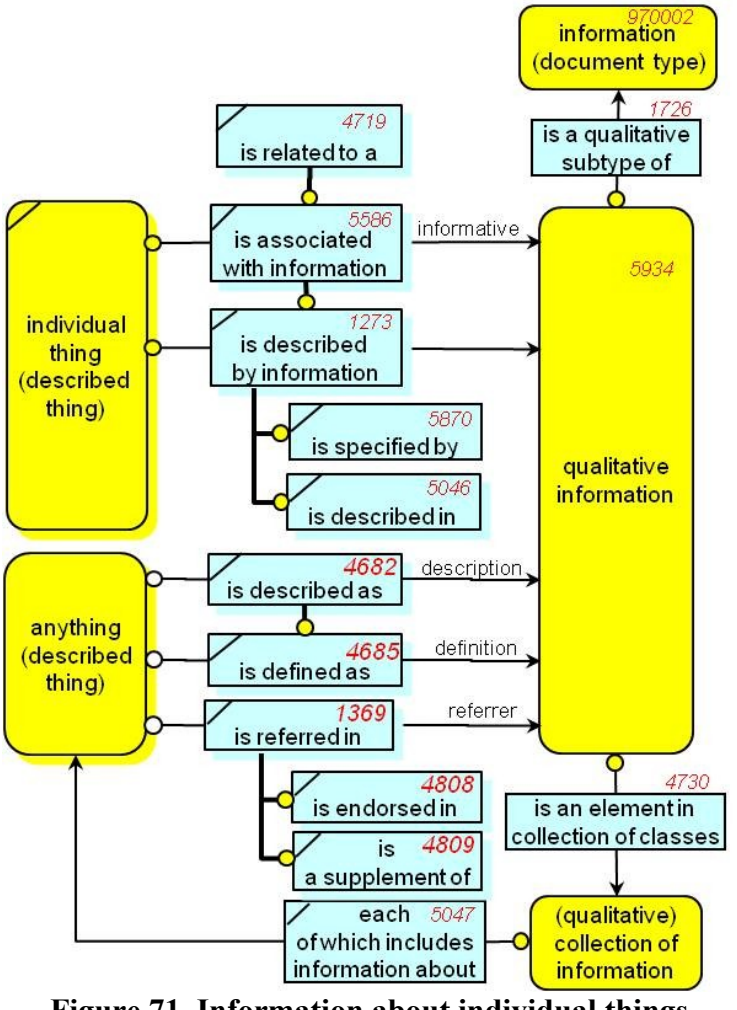


Figure 71, Information about individual things

Note that the piece of information (the qualitative information) that describes, specifies, or contains a description of the individual thing is an object that will have a UID and may have a name, whereas it is defined by a description, being the information content. This description is independent of the way in which it is presented or may be presented in multiple ways in particular formats on information carriers. For example, the text of a requirement (P-101 shall …) is a description of an object, called 'Req. 5.1', which object is the

qualitative information in which P-101 is described. This can be expressed as follows:

Left hand object	Relation type	Right hand object	Description (of left hand object)
P-101	is described in	Req. 5.1	
Req. 5.1	is a part of	Specification 101	
Req. 5.1	is a qualitative subtype of	requirement	P-101 shall …

Table 39, A piece of information about an individual thing

Note that qualitative information is abstracted from one or more physical representations in possibly multiple copies or versions. Such qualitative information classifies the information aspects in those copies and versions. Therefore, it is regarded as a qualitative kind, a qualitative subtype of the general concept 'information'. Therefore, a relation that expresses that some qualitative information is about an individual thing is a relation between that individual thing and a (qualitative) kind of thing.

8.4 Relations with implicit individual things

8.4.1 Classification of implied parts

In many models of assemblies the components are not all explicitly identified, so that the model consists of an incomplete composition hierarchy. Nevertheless, there is often a requirement to express that an assembly has one or more components of a particular kind. This means that there is a relation between the individual assembly and the kind that classifies one or more of its components. If it is known how many of the same kind of component it has, that may be specified by the right hand cardinalities. An example of such an expression is:

statement: P-1 has a part that is classified as a bearing

Note that it is also possible to explicitly state that such a component is not part of the assembly. This is specified by stating that the intention of the expression is a denial.

For example, this is specified as follows:

denial: P-1 has a part that is classified as a bearing

Such relations are short-cut relations that need a verification on their consistency with possible explicit composition specifications, because it might be that somewhere else in the model the component might be specified explicitly. Then it should be detected that these are (probably) the same components and duplication should be avoided.

The verification of the consistency of a short-cut relation and an explicit modeling of a part is enabled by the definition of the kind of relation. That definition specifies that the short-cut relation is equivalent with two relations, a composition relation and a classification relation. So, it may be that software determines that there is a part explicitly specified and that is classified conform the specification. For example as follows:

P-1 has as part B-1
B-1 is classified as a bearing

Then by inference the software can conclude that the first statement implies a part, let us call it E-1. Then it can be recorded that the short-cut relation is equivalent to the pair of detailed relations, because apparently E-1 equals B-1.

9 Hierarchical relations between kinds of things

Relations between kinds of things can be recursive or non-recursive. A non-recursive relation, also called a hierarchical relation, is a kind of relation for which by definition holds that a concept upstream in a chain of such relations may not appear also downstream in that chain. Non-recursive kinds of relations automatically result in hierarchical networks or tree-shaped networks. A hierarchical relation relates a wider concept to a narrower concept, where the wider concept is higher in the hierarchy than the narrower concept. There are various subtypes of hierarchical relations as is illustrated in Figure 72.

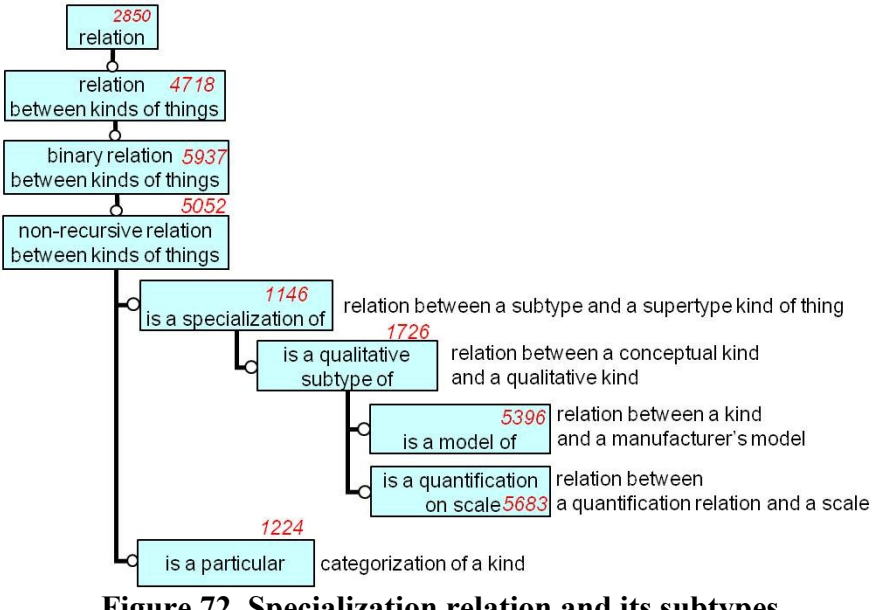


Figure 72, Specialization relation and its subtypes

Those subtypes are discussed in the following paragraphs.

9.1 Specialization relations - Taxonomies

The most important kind of hierarchical relation is the subtype-supertype relation (1146), which is also called a specialization

relation, whereas its inverse is called: a generalization relation. This kind of relation specifies that a concept is by definition a subtype of another concept. A specialization relation specifies that the subtype kind has additional constraints as criteria for membership than the supertype kind; and each member of the subtype kind is also a member of the supertype kind. This implies that a subtype kind inherits all knowledge that is valid for its supertype kind(s), because every assertion that is true for a supertype is also true for its subtypes. Formally defined:

> A <is a kind of> B if and only if for all x,
> if x <is classified as a> A, then x <is classified as a> B.

A specialization relation is *transitive* and *antisymmetric*. Transitive means that a relation of this type between two concepts implies that such a relation is also applicable between kinds of things that are indirectly related in a chain of relations of this type. This means that explicit specification of indirect relations is superfluous. Antisymmetric means: if concept A is a subtype of concept B, then B is not a subtype of A.

A specialization relation can be denoted by various phrases, such as <is a specialization of> or by one of its synonyms: <is a kind of>, <is a subtype of>, <is a subclass of> or by one of their inverse phrases, such as <is a generalization of> or <is a supertype of>.

A chain of concepts or hierarchical network of concepts that are related by relations of this kind is called a *taxonomy*.

Subtype-supertype hierarchies or taxonomies can be defined for any category of concepts. Thus there are

- taxonomies of kinds of physical objects in various domains,
- taxonomies of kinds of aspects, such as properties & qualities,
- taxonomies of kinds of roles of physical objects,
- taxonomies of kinds of roles of aspects,
- taxonomies of kinds of relations,
- taxonomies of kinds of occurrences and correlations, etc.

A consistent collection of taxonomies in various domains should form one integrated taxonomy, of which all concepts are subtypes of the generic concept ‘anything’.

9.1.1 Subtypes by distinguishing aspect values

Subtypes of a particular supertype have always an additional constraint on an aspect or on a composition for which their supertype has flexibility.

This is illustrated in Figure 73.

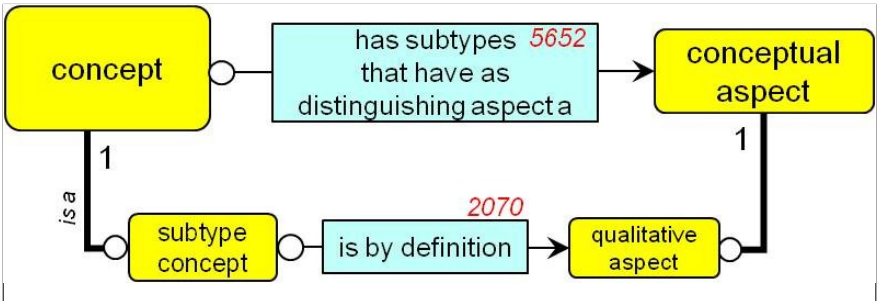


Figure 73, Qualitative aspect as discriminator

The supertype concept has flexibility on a conceptual aspect. For example the concept may have flexibility on its diameter. However the subtypes have less flexibility. For example each of them may have a defined diameter. The various subtypes thus have by definition as aspects a different qualitative aspect, such as a distinct diameter. For example, the supertype concept bolt has subtypes with various thread diameters, such as a diameter of 6 mm, 8 mm, 10 mm, etc. respectively.

9.1.2 Subtypes by distinguishing components

Figure 74 illustrates that there can be subtypes that have by definition different kinds of components as parts or they can by definition be without a component of a particular kind.

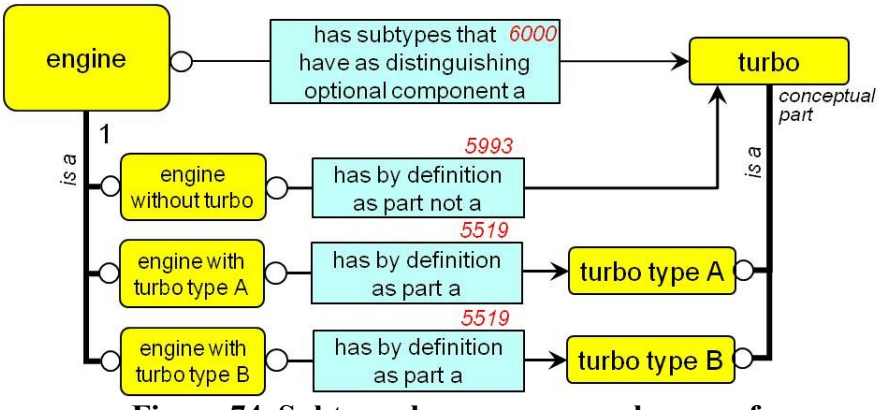


Figure 74, Subtypes by presence or absence of a kind of component

For example the supertype engine may have three subtypes: one with a turbo type A, one with a turbo type B and a third subtype without a turbo.

9.1.3 Families of subtypes

The subtypes that have different 'solutions' or values for the same kind of aspect (conceptual aspect) or have different kinds of component physical objects, together form a family of subtypes. Those different solutions are usually mutually exclusive, which implies that an individual thing can only be classified by one kind of a particular family at the same time. The criterion by which the subtypes in a family are distinguished is called the distinguishing aspect or distinguishing part and is also called its discriminator.

> Note that the family of subtypes can be determined by logic inference, so that it is not necessary to explicitly define the collection of subtypes in a family.

A concept may have multiple families of subtypes, each with its own distinguishing aspect or component. For example, the concept bolt has not only subtypes that have as distinguishing aspect their

diameter, but also subtypes by material of construction with values: stainless steel, galvanized carbon steel, copper, etc. and also subtypes with a distinguishing kind of component, being a hexagonal head, a cylindrical head, etc. Thus the supertype (bolt) has flexibility in all three aspects. The family of subtypes by material of construction are stainless steel bolt, galvanized bolt, copper bolt, etc., each of which have by definition a fixed value as its material of construction, but still have flexibility in their diameter and head type.

We can also define further subtypes that have multiple fixed values for various distinguishing aspects or parts. For example, a '6 mm, stainless steel, hexagonal head bolt' can be defined as a separate kind of bolt. Such sub-sub-types have multiple supertypes (three 'parent' kinds). The latter implies that such subtypes inherit characteristics of all their supertypes.

Individual things may be classified multiple times as being a member of multiple kinds, belonging to different families, but they may also be classified once as being a member of a sub-sub-type. For example:

Collection B1	is a collection of which each element classified as a	6 mm, stainless steel, hexagonal head bolt

9.2 Qualitative aspects

Kinds of aspects can be distinguished in conceptual aspects and qualitative aspects. Conceptual aspects are aspects that have no quality or value. Qualitative aspects are aspects of which the size or magnitude or severity is qualified or quantified by a fixed value or by a range. For example, concepts such as red, toxic, short, '5 mm' and '10-20 degC' are qualitative aspects that are qualifications of the concepts color, toxicity, length and temperature range. The latter are (generic) conceptual aspects that are unqualified.

The kind of relation that specifies that a qualitative aspect is a qualitative subtype of a conceptual aspect is called a qualification relation, which is denoted by the phrase <is a qualitative subtype of>

(1726). Note that the qualification relation is a subtype of the specialization relation.

Examples of relations between conceptual aspects and qualitative aspects are expressed as follows:

red	is a qualitative subtype of	color
toxic	is a qualitative subtype of	toxicity
5 mm	is a qualitative subtype of	distance
etc.		

9.3 Types of physical objects

Kinds of physical objects can be distinguished in generic kinds and types of physical objects. A type of physical object is a kind of which a number of its aspects have by definition a fixed value. A relation between a type of physical object and its generic supertype can be specified by using the qualification relation as well.

An example of a relation between a type of physical object (qualitative physical object) and its generic supertype is:

M6 bolt	is a qualitative subtype of	bolt

The specification that a particular type of physical object has by definition particular aspect values may be given in a textual definition that is only meant for human interpretation (as described in par. 5.8) or may be expressed in a definition model (as described in par. 10).

9.4 Manufacturer's models and standard types

Further subtypes of types of physical objects are manufacturer's models or catalogue items that are denoted by a model identifier (for a standard configuration and shape) and possibly by a standard size.

The kind of relation that can be used to specify a relation between such a manufacturer's model and a higher level concept or type can be denoted by the phrase <is a model of>. This kind of relation is a further subtype of the specialization relation (and of the qualification relation). For example:

Volvo S40	is a model of	car
Siwamat 6140	is a model of	washing machine

The manufacturer's models and standard types and sizes are typically specified in quite some detail. However, not all of their aspects need to have fixed values, and not all of their components are always fully defined, because various options may still be open. The detailed specification of a manufacturer's model by be given in a textual description (see par. 5.8) or in an explicit definition model (see par 10), or as a reference to an identified document (see par. 5.9).

9.5 Quantification of quantitative values on a scale

A special case of qualitative aspects are quantitative aspects, which are aspect values (concepts) which magnitude or size can be defined by a relation with a numeric value on a scale. Note that these are not individual aspects, but 'standard values', such as allowed values that might be defined by a property range or might be listed in pick lists. (Quantification of individual aspects is discussed in par. 8.1.5). An example of a particular length or distance is the concept '300 mm', which is a particular qualitative value that can be defined by it's quantification as being equal to the number 300 on a millimeter scale. Such a definition is then expressed as follows:

300 mm	is a qualitative subtype of	distance	
300 mm	is by definition quantified on scale as equal to	300	mm

The model that is the basis for this example is illustrated in the upper part of Figure 75.

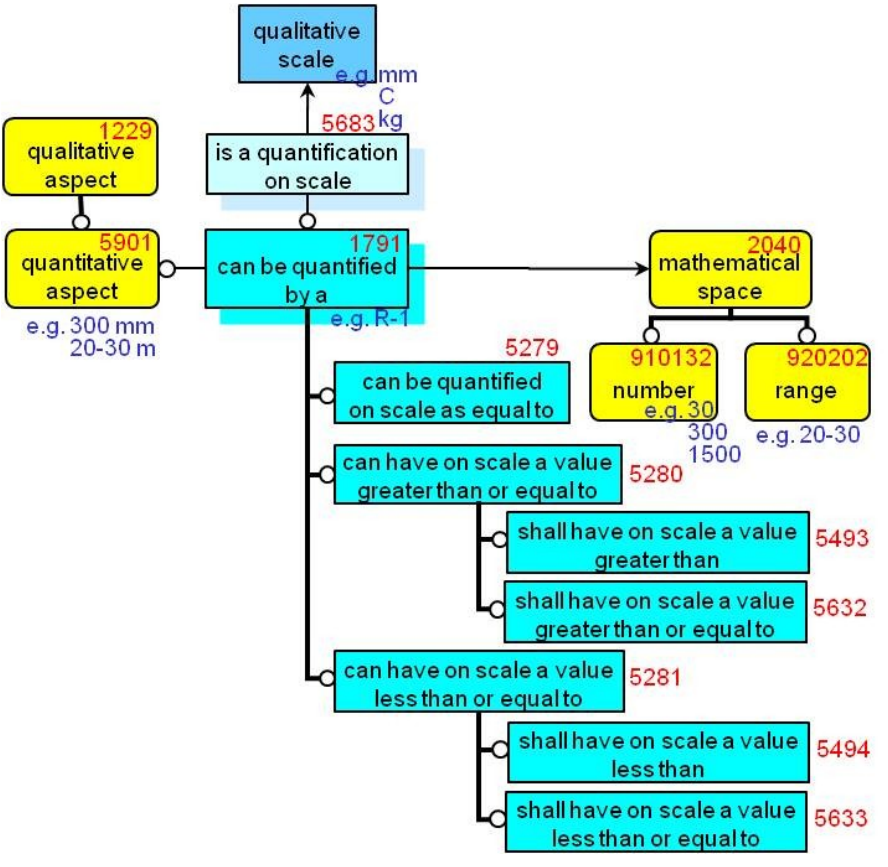


Figure 75, Quantification of quantitative aspects

In Figure 75 a relation, such as R-1, is a conceptual quantification relation (1791) between a quantitative aspect (5901) and (usually) a number (910132) (but in general a mathematical space (2040)), whereas R-1 is also qualified as a 'quantification on scale' (5683) kind of relation. In other words, the relation R-1 is qualified in two ways, first as a quantification that is using a particular scale or unit of measure. The unit of measure is made explicit by a second qualification of R-1 as a quantification on some particular scale, such as a millimeter scale.

In tabular form this is expressed as follows:

R-1 is qualified as a 'can be quantified by a' relation
R-1 is qualified as a 'mm' (scale) relation

Note 1: The scale 'mm' is a kind of relation, because it qualifies the way in which a property is related to numeric values.

Note 2: Both relations, 'can be qualified by a' and 'mm', are conceptual correlations, because they relate kinds of aspects.
(Note that qualitative aspects, such as '300 mm' and '300' as well as 'mm', are subtypes of kind of thing).

For example, the quantification relation between the length value '300 mm' and the mathematical concept '300' is a relation that is qualified by a unit of measure, being 'mm'. Such a unit of measure or scale is a particular (qualitative) method that is a mapping relation between an input (a particular length) and an output (a particular number) of the method. Therefore a scale or unit of measure is a kind of relation.

Scales are standardized and included in the Dictionary. They form their own hierarchy as illustrated in Figure 76.

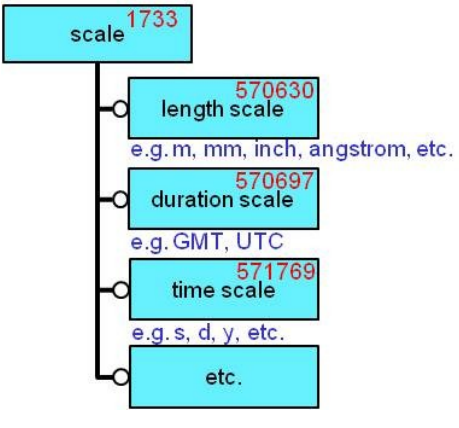


Figure 76, Scales and units of measure

It can also be predefined in a knowledge base which subtype of scale is (by definition) intended for quantification of which kind of property.

For example:

length scale is by definition a scale for a distance

The relation between a quantitative aspect and a number does not need to be an equality relation, it might also be specified by some kind of inequality relation, Therefore there are subtype kinds of relations defined that enable to express requirements that the value shall be greater than (or equal to) and/or less than (or equal to) some value. Furthermore, such expressions may not specify a possibility or a requirement, but something that is by definition the case. Therefore further subtypes are defined for the expression of definitions (which are not shown in Figure 75). The latter kinds of relations can be used for example to define the property range '2-5 mm' as follows:

Name of left hand object	Name of kind of relation	Name of right hand object	UoM
2-5 mm	is a qualitative subtype of	property range	
2-5 mm	is by definition quantified on scale as greater than or equal to	2	mm
2-5 mm	is by definition quantified on scale as less than or equal to	5	mm

A property range is not necessarily defined as being bounded by numeric values on a scale. It can also be defined as being bounded by property values (qualitative aspect). In those cases the boundaries are defined by another kind of relation, being a relation between a qualitative range and a qualitative value, which is a conceptual correlation between aspects, because both related things are kinds.

This is depicted in Figure 77.

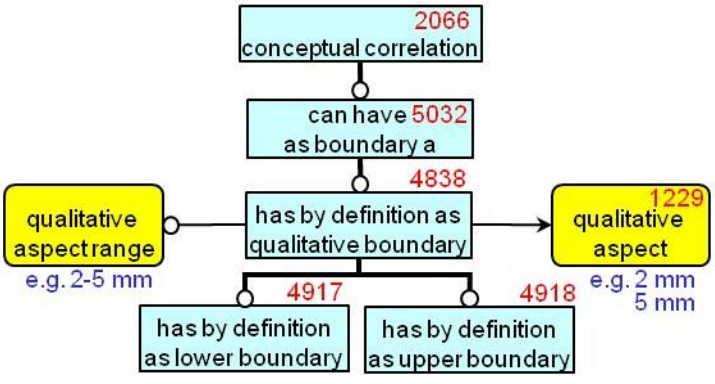


Figure 77, Definition of the boundaries of a qualitative range

The use of these kinds of relations is illustrated in the following example:

Name of left hand object	Name of kind of relation	Name of right hand object
2-5 mm	is a qualitative subtype of	property range
2-5 mm	has by definition as lower boundary	2 mm
2-5 mm	has by definition as upper boundary	5 mm

Numeric ranges differ from property ranges, because they are not quantified on a scale. However, the definition of numeric ranges is a relation between quantitative values, which is a subtype of a relation between qualitative values. Therefore, such definitions can be expressed by using the same kinds of relations as above.

For example the range '20-30' might be defined as follows:

Name of left hand object	Name of kind of relation	Name of right hand object
20-30	is a qualitative subtype of	numeric range
20-30	has by definition as lower boundary	20
20-30	has by definition as upper boundary	30

Intrinsic aspects are aspects that are by definition possessed by a particular kind of object. Therefore, usually their name includes the name of a kind of physical object. For example, 'lock position' or 'pipe diameter'. Allowed values for such kinds of intrinsic aspects are often defined by explicitly specifying discrete optional values. For example, the allowed values for the position of a lock might be specified as 'open' or 'closed'. That can be specified as follows:

Name of left hand object	Name of kind of relation	Name of right hand object
lock position	can have as option	open
lock position	can have as option	closed

The general model for such specifications is given in Figure 78.

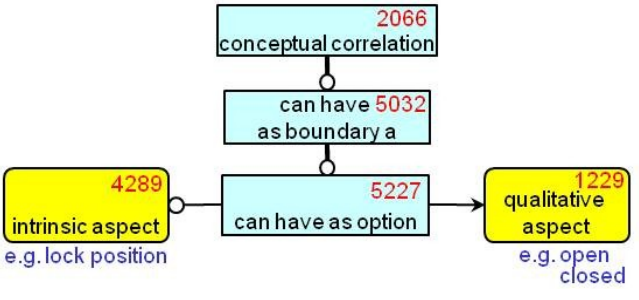


Figure 78, Specification of options for kinds of intrinsic aspects

Another way of specifying such discrete allowed values is by first creating a collection of qualitative aspects, also called an 'enumerated list of values', and the specifying that some kind of intrinsic aspect is related to that collection. For example:

lock position shall be one of the allowed lock positions

The kinds of relations that are required to define such a collection are described in chapter 11.

10 Conceptual relations between things of specified kinds

The Gellish allows expressing what is true, but also allows expressing what is untrue or is fantasy and what can not be the case in a normal world. As such the language definition does not specify constraints on what is possible or not. Nevertheless, the Gellish definition includes constraints that specify what things are. It also enables verifying the consistency of expressions. Furthermore, Gellish can be used for expressing explicitly what is known as possibilities. Such knowledge about possibilities can be expressed by using subtypes of the generic possibility relation, also called a 'conceptual relation between things of specified kinds' (4698). Such a relation expresses that relations of the specified kind between things of the specified kinds are known to be possible. Thus such a subtype relation between kinds of things expresses knowledge about possibilities for individual things of particular kinds, i.e. what can be the case. If something is a requirement, so that it shall be the case in a particular context, than it must necessarily be a possibility. Therefor, requirement relations are subtypes of possibility relations. If some relation expresses what is by definition the case for things of specified kinds, then such a relation shall and can be the case as well. Therefor, defining relations are further subtypes of requirement relations. These kinds of relations are all called 'conceptual relations' because they specify in concept (in principle) what can, shall be or is the case for individual things of the specified kinds. This means that in principle relations of such kinds can be used either to derive relations between individual things of such kinds, or they can be used to verify whether relations between individual things conform to the specified possibilities, requirements or definitions.

> It should be noted that if something is not declared as a possibility, it nevertheless may be possible. However, a party may apply the rule that individual relations may only be created after the possibility of such a relation is explicitly specified.

Figure 79 illustrates the top of this branch of the hierarchy of kinds of relations that are subtype of the generic possibility relation (4698).

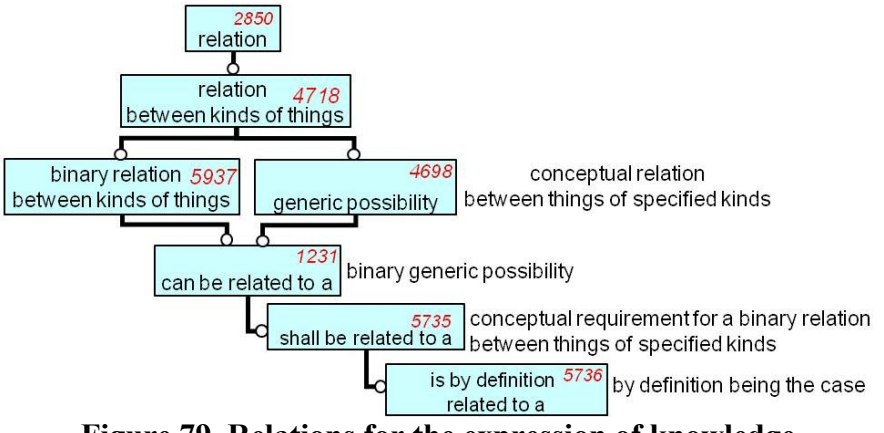


Figure 79, Relations for the expression of knowledge, requirements and definitions

Those subtypes specify what can be the case, i.e. possibilities, what shall be the case, i.e. requirements or what is required to be the case (in a particular context) and what is by definition the case, i.e. definitions.

10.1 Modeling possibilities, requirements and definitions

10.1.1 Knowledge about possibilities

Knowledge expressions typically express possibilities. In other words, they express what can be the case. Such possibilities are modeled using subtypes of a (generic) possibility relation between kinds of things. The generic possibility relation can be of any order and therefor it has as subtype a second order possibility relation, also called a binary possibility relation. Further subtypes of kind of binary possibility relation can be expressed by using subtypes of a binary generic possibility relations. Phrases that denote such subtypes typically comprise fragments such as <can be ... a> or <can have ... a>.

For example, the following kinds of relations are intended to express knowledge about possibilities:

pump	can have as part a	impeller
impeller	can have as aspect a	diameter

Chains or networks of relations of such kinds form knowledge models.

A generic possibility kind of relation defines that a relation of such a kind means that in practice such a relation is possible for at least some of the things of the related kinds. For example, at least some pumps can have an impeller. However, by default the relation does not specify that every pump can have an impeller. The relation does also not specify how many impellers a pump can have simultaneously or during its life. If any of those options is limited, then that constraint can be added by specifying cardinality constraints. Simultaneous cardinality constraints limit the number of individual things of a specified kind that are simultaneously minimally and maximally allowed to be related to one related individual thing of the other kind. Cardinalities are further discussed in par. 13 and 13.4.3.

10.1.2 Requirements

A requirements kind of relation specifies that relations are required between individual things of a kind that satisfies the requirement. This means that in practice such a relation is required by the requiring party for any of the individual things of the related kinds. Requirements shall be realized in practice as well as being incorporated in the information model that describes the realization. Realizations of possibilities and requirements are further discussed in par. 10.5.

When something is required for a particular kind of thing it implies that it is also possible for at least some of the things of such a kind (at least in the opinion of the requirer, although a requirement may be in conflict with the possibilities). Therefore, kinds of relations that express requirements are defined as subtypes of relations that express possibilities about kinds of things.

A party or standard that formulates a requirement has a role as the validity context within which the requirement holds and is recorded as a contextual fact in the appropriate Gellish expression.

Phrases that denote requirements typically comprise fragments such as <shall be ... a> or <shall have ... a>.

10.1.3 Definitions

Definitions of kinds of things are statements about what is by definition the case for those kinds of things. If for an individual thing it is specified that something is not the case, whereas for things of a particular kind it is declared that something is by definition the case, and then that means that the individual thing is not of that kind. In other words, the things that are by definition the case are *necessary conditions* for being a member of the defined kind. We distinguish between textual definitions and modeled definitions. Textual definitions that are expressed as textual information are discussed elsewhere. Here we discuss only modelled definitions, which are definitions that are expressed as definitional relations. If a definition for a kind of thing is completely modeled, then satisfying all definitional relations is *sufficient* for determining that something is a member of the specified kind. However, it should be noted that in many applications, especially in design application, individual things do not (or not yet) have aspects on the basis of which they might be *classified* by a kind, but it is the other way around: (imaginary) individual things are *declared* to be classified by a kind in order to *specify* that the definition of the kind also applies to the individual thing. Thus, in such cases the aspects of the individual things are derived from the definitions of the kinds.

Definitional relations that are necessary conditions for being member of a kind should be distinguished from things that are normally the case for well-formed things of the kind. This encyclopedic knowledge about well formed things is usually modeled as a requirement in the context of some norm for well-formed things of that kind.

Kinds of relations that express what is by definition the case are defined as subtypes of requirement relations, because what is by definition the case is also required and is also possible.

Phrases that denote definitions typically comprise fragments such as <is by definition ... a> or <has by definition ... a>.

10.2 Compositions of things of particular kinds

A conceptual composition relation between things of particular kinds (1261) expresses a possibility which is the knowledge that an individual thing of a particular kind (possibly) can have as part one or more other things of a particular kind. Such a statement is usually abbreviated by stating that

kind A <can have as part a> kind B.

Such a statement implies that the inverse is also true:

kind B <can be a part of a> kind A.

The minimum and maximum number of individual things of kind B that can simultaneously be a part of one individual thing of kind A can be specified by two simultaneous cardinality constraints. For example the constraints 0 and n, expressed as [0, n]. Normally one part of kind B can simultaneously be the part of only zero or one whole of kind A. This can be specified by the other cardinality constraints, for example as [0, 1]. Cardinality constraints in general are discussed in par. 13.4.3.

A possibility, such as specified above, can be realized by one or more individual realities that can be modeled by using the counterpart composition relation (1260). Such a relation expresses that

some individual thing <is a part of> some other individual thing (as is discussed in par. 7.1).

Such knowledge presents an option that can typically be used during design or construction of assemblies of individual things or when complex projects, activities and processes are composed.

Figure 80 illustrates that a possible realization relation (5091) between a possibility (a possible composition relation) and a reality (a real composition relation) is modeled as a relation between relations.

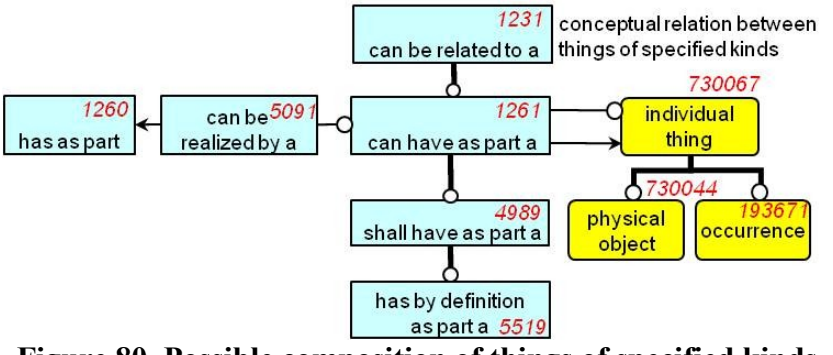


Figure 80, Possible composition of things of specified kinds

The figure also shows that the possibility is defined as a subtype of a conceptual relation between individual things (1231), which relates two different individual things (730067). This implies that the composition relation can also be applied for possible composition relations between all the subtypes of individual thing. For example it can be applied for possible compositions of physical objects of various kinds or for possible compositions of occurrences of various kinds, because they are subtypes of the concept 'individual thing'.

Furthermore, what holds for a possible relation (1261) also holds for its subtypes: a requirement for one or more composition relation (4989) and a composition relation that is by definition the case between a whole and a part individual thing of the specified kinds (5519). Usually a requirement and a definition specify that every whole of a specified kind shall have, or by definition has one or more things of a specified kind as parts. It may also specify that a part of a specified kind shall be or by definition is a part of a whole of a specified kind. The precise meaning shall be indicated by specifying the left hand and right hand simultaneous cardinality constraints.

There are various ways in which things can be composed, depending on the kind of connection between the components, if any, or

depending on the kind of component that is composed and the kind of role that is played by the component. These different ways of composition can be expressed by using different kinds of composition relation. Figure 81 presents a number of such subtypes of the conceptual composition relation.

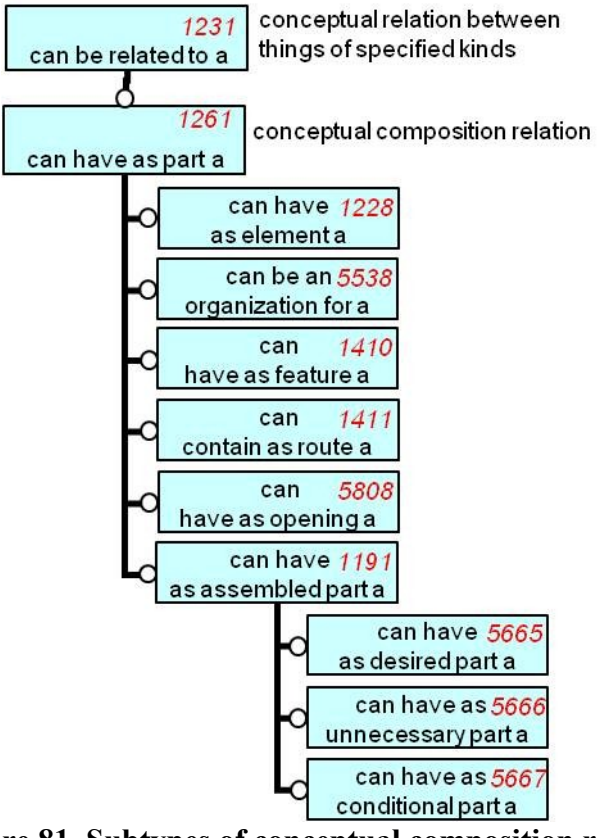


Figure 81, Subtypes of conceptual composition relation

The various subtypes of the conceptual composition relation have the following meaning:

- can have as element a

  specifies that a collection of a specified kind can have things of a specified kind as elements, whereas the elements form a collection, in which they are not coherent, although the elements may be in a sequence.

- can be an organization for a

specifies that things of a specified kind can be members of an organization of a specified kind.

- can have as feature a

specifies that a physical object of a specified kind can have a feature of a specified kind. A feature is an integral part of a whole and may have an undefined boundary with the rest of the whole. For example a rim or an integrated lifting lug that is casted in a body.

- can contain as route a

specifies that a network of a specified kind contains a route of a specified kind.

- can have as opening as

specifies that something of a specified kind can contain a hole of a specified kind.

- can have as assembled part a

specifies that a whole of a specified kind can have a part of a specified kind that is assembled in the whole. It implies that there is a connection relation between the part and one or more other parts of the same whole.
This kind of relation has the following subtypes:

- can have as desired part a

specifies that an assembled part of a specified kind is not only possible, but also desired, which expresses a preference without being an obligation. Such a preference is always only valid in a particular validity context. The specification of a validity context is discussed in par. 13,

- can have as unnecessary part a

specifies that an assembled part of a specified kind is allowed, but not required within a particular validity context.

- can have as conditional part a

specifies that an assembled part of a specified kind is only possible under a particular condition. The condition may be specified in a conditional consequence relation (an if-then relation) as is discussed in par. 10.10.

10.3 Aspects of things of particular kinds

A specification of the possibility that things of particular kinds can possess aspects of particular kinds can be expressed by using a kind of relation that specifies that an individual thing of a specified kind can possess and aspect of a specified kind. This kind of relation can be denoted by the following phrase:

<can have as aspect a>

When an individual thing possesses such an aspect in reality, that fact does not necessarily imply that an information model about the individual things also includes a possession relation for such an aspect. Therefore, the meaning of such a statement is that it is possible in reality and it is also allowed in an information model about an individual thing of such a kind in order to reflect reality or to reflect an idea (an imaginary individual thing).

Figure 82 presents this kind of relation with its supertypes and its most important subtypes.

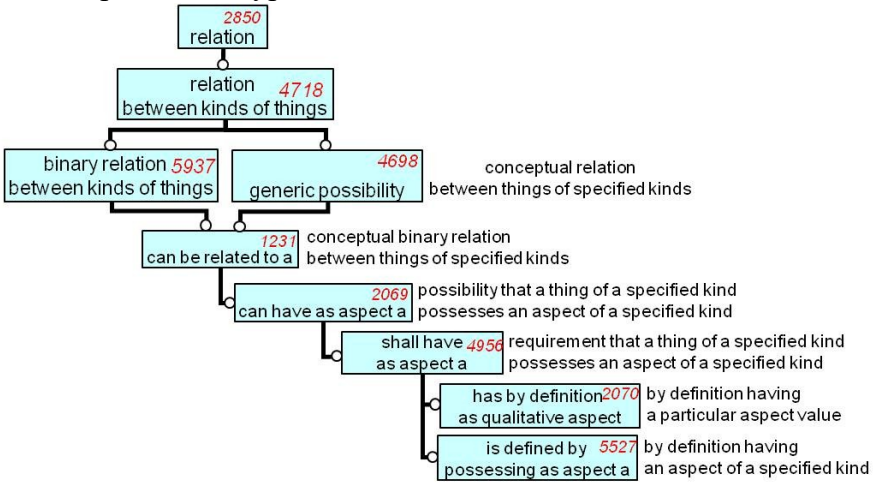


Figure 82, Possible possession of aspects by things of a kind

The main subtypes of the <can have as aspect a> kind of relation are mentioned in Figure 82. They have the following meaning:

- shall have as aspect a

  This specifies a requirement that any individual thing of a specified kind shall possess an aspect of a specified kind. This implies that a

thing of such a kind in reality shall possess such an aspect and it implies that an information model about a possessor shall include a relation with such an aspect, as well as information about the qualitative or quantitative value of the aspect. In case of a quantitative value the model shall also include a relation with a scale that is used to determine the value. This seems trivial, but for a computer it is not trivial, as it implies that several binary relations need to be specified. For example, assume in a requirements model for roads in a particular context there appears the following expression of a requirement:

road shall have as aspect a width

Then the requirement implies for any individual road it is required to specify not only that the road has an individual aspect that is classified as a width, but also that its value is specified and if the value is a quantitative value that then the scale for that quantification is given.

Thus, assume that an information model contains the expression:

B-23 is classified as a road

Then the requirement implies that three expressions are required, similar to the following example:

B-23 has as aspect W-1
W-1 is classified as a width
W-1 has on scale a value equal to 5.20 m

o The requirement holds within a particular validity context (for validity context see par. 13.4.

o This subtype has the following further subtypes:

o has by definition as qualitative aspect
[or] is by definition qualified by
[or] is by definition
This specifies that any possessor of a specified kind by definition possess an aspect that has a qualitative aspect of the specified value.

For example:
- horizontal vessel <is by definition> horizontal.

- is defined by possessing as aspect a

This specifies that a concept is defined by having a qualitative aspect (value) that is a qualitative subtype of the specified conceptual aspect. This implies that each individual thing of the specified kind has an aspect of the specified kind that is qualified by the same qualitative aspect (value).
For example:
- horizontal vessel <is defined by possessing as aspect a> orientation

Other kinds of expression can be used for specifying what the aspect values are allowed to be or by definition are.

10.3.1 Properties and qualities of things of particular kinds

Depending on the kind of aspect it is possible to use more dedicated subtypes of the <can have as aspect> relation. Some of such subtypes are presented in Figure 83.

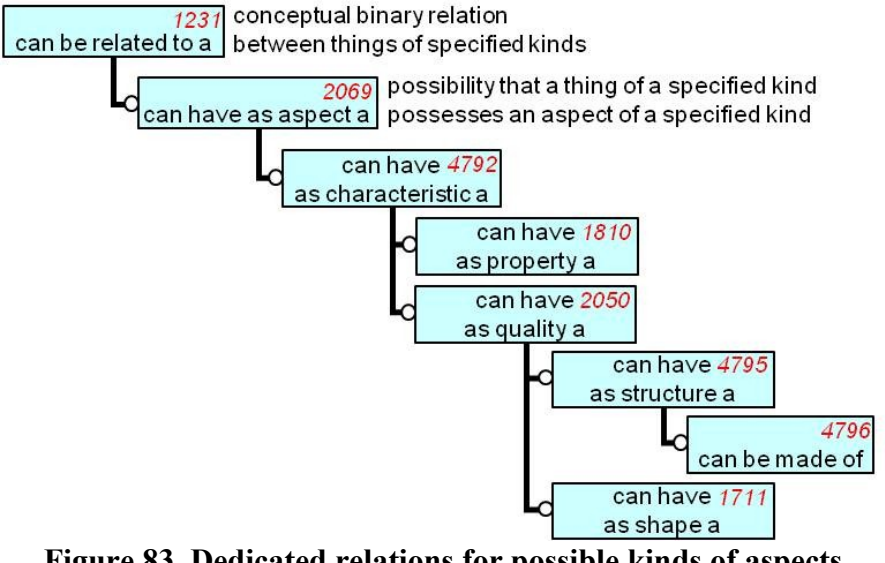


Figure 83, Dedicated relations for possible kinds of aspects

The distinction between the subtypes of characteristic, being (physical) *properties* and *qualities* and their further subtypes is discussed before in par 7.4.

As in most other cases the kinds of relations about possibilities have their corresponding kinds of relations that express requirements and things that are by definition the case.

Thus there are also kinds of relations, such as:

shall be made of
is by definition made of

The kind of shape (or qualitative shape) of physical objects of a particular kind (or of a type or manufacturer's model and size) is a special kind of quality. Shapes can be qualified by their qualitative value. For example as cylindrical shape, complex shape, etc. This can be used in expressions such as:

object type-1	shall have as shape a	cylindrical shape
cylindrical vessel	has by definition as shape a	cylindrical shape

Shapes can be specified in further detail by parametric geometric correlations (mathematical functions) or combinations of them. Such correlations are defined in coordinate systems, whereas various dimensions of the shaped physical object and its parts or features play a role as parameter values in the correlations.

10.3.2 Time aspects of states and occurrences of kinds

It may be specified that a state or an occurrence of a particular kind can or shall take place within a particular period in time. For example it can be specified that any occurrence of a particular kind can or shall occur within a particular period of time. For example, it may be specified that any payment shall be within 30 days (from the date of issue of an invoice). Then the kind of occurrence is related to a qualitative duration. This can be expressed for example as follows:

payment shall occur within a period of 30 days

Note that the qualitative duration is a qualitative aspect, which value is normally specified as a number on a scale (see also the quantification of individual aspects in par. 8.2).

It may also be specified that for any occurrence of a particular kind it is required that information is provided about the planned or actual period in time or date of occurrence (see Figure 84).

This mean that it is specified that such an occurrence shall take place within some period in time (which implies a maximum duration) such as a month in time, or that is shall occur at some date. For example a requirement for information, such as:

project	shall occur within a period of a	month in time
delivery	shall occur at a	date

The above example requirement means that it is obligatory that it is specified for any project in which month it shall take place and for any delivery what its delivery date is or will be.

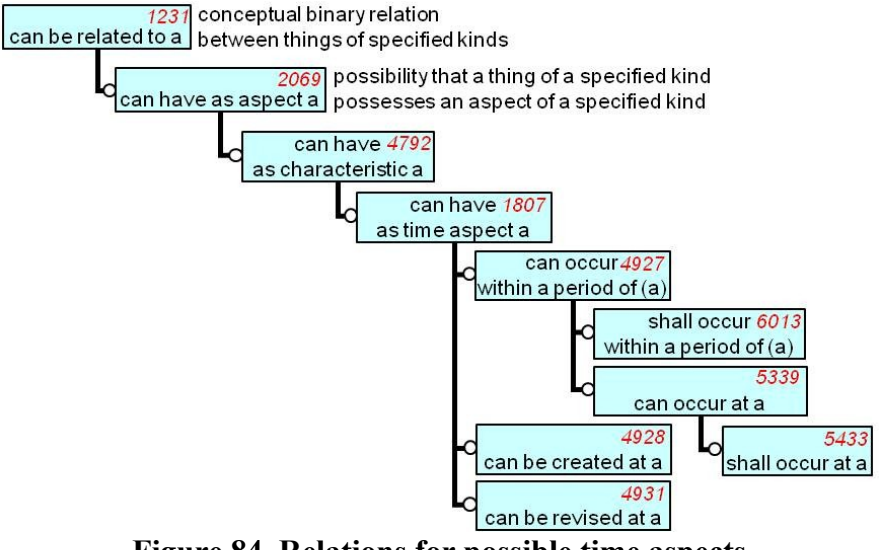


Figure 84, Relations for possible time aspects

Sometimes a planned or actual creation date or revision date for individual things is recorded, which means that the creation or revision process is completed at that date, although the process might have a longer duration than a day. It is possible to express a requirement for the recording of such a creation or revision date about any object of a kind. Such a requirement is expressed as a relation between a kind of physical object and a kind of period in time. Such a relation is in fact a short-cut relation as it implies an occurrence (a creation or revision), although that occurrence may not be modeled explicitly.

Examples of the use of such expressions of requirements are:

connection	shall be recorded to be created at a	date
pump	shall be recorded to be revised at a	date

The time aspect to which a state, occurrence or physical object is related does not need to be the concept ‘date’.

Other subtypes of time aspect are given in Figure 85.

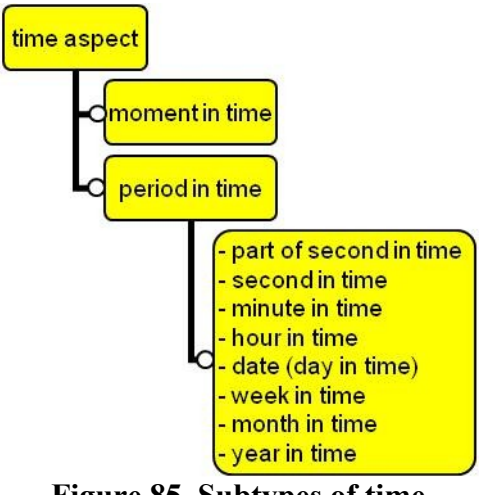


Figure 85, Subtypes of time

A moment in time is indicated as a concept that is distinct from a period in time. Nevertheless, there are strong arguments for the opinion that moment with zero duration do not exist so that a moment in time is in fact a period in time with an unspecified short duration. We therefore use a pragmatic definition that a moment in time is defined by only one time value, without a specification of a duration. On the other hand, a period in time is defined (in principle) by two time values, either by a start value and a duration, or by a start time value and a termination time value. For example, if a moment in time is denoted by a second in time, then it is assumed that the occurrence started and is possibly completed within the duration of that second.

10.3.3 Roles and role players of particular kinds

The concept 'role' is an extrinsic aspect of a role player that depends on a relation in which the role player plays such a role. Thus the concept role is a subtype of aspect. Subtypes of roles are for example usage and application. A role is typically played by something in a relation to something else, during a certain period or during the whole of the lifetime of the role player. Figure 86 presents some kinds of relations that classify relations between kinds of things and kinds of roles.

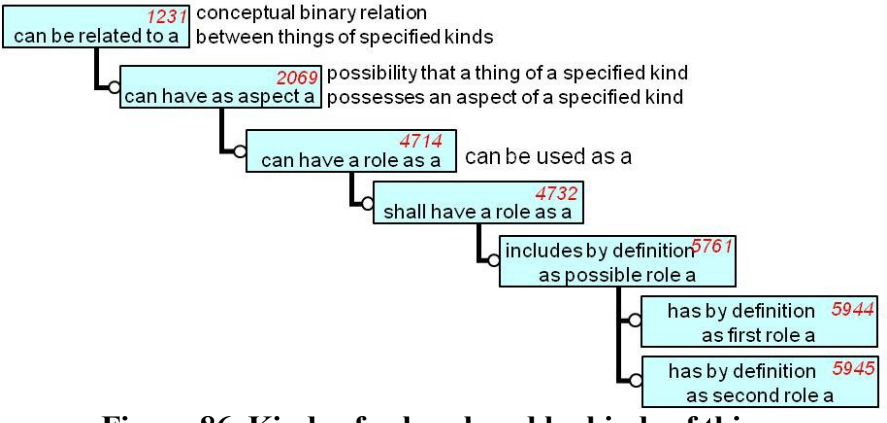


Figure 86, Kinds of roles played by kinds of things

A <can have a role as a> relation can be used to express the knowledge that any individual thing of a specified kind can play roles of a specified kind, without specifying in which kind of relation that is the case.

For example:

paint layer	can have a role as a	protector against corrosion
man	can have a role as a	father

Because usage and application are subtypes of role, there are subtypes of <can have a role as a> such as <can be used as a> and <can be applied as a>.

Separately it is possible to specify that for binary relations of a particular kind there are by definition (always) two particular kinds of roles involved. For example, a parentship relation between a

father and a child has by definition as first role a father and as second role a child. This can be specified as follows:

is a father of	has by definition as first role a	father
is a father of	has by definition as second role a	child

In other situations (in a particular context) it may be a requirement that roles of a particular kind shall be played by things of a particular kind, or the inverse, which specifies that things of a particular kind shall play a role of a particular kind. Note that a requirement relations shall be accompanied by a validity context.

All the above kinds of relations may be accompanied by cardinality constraints. For example, one man can have zero to many father roles, whereas a (biological) father role can be the role of one man only. Thus

[1,1] man	can have a role as	[0,n] father

10.3.4 Aspect of parts of things of particular kinds

Sometimes it is specified that some part of a kind of physical object has an aspect of a particular kind, without specifying which part or which kind of part is involved. To specify such knowledge or requirement requires a relations between a kind of physical object and a kind of aspect, whereas the aspect is not an aspect of the physical object itself. Such a kind of relation is presented in Figure 87.

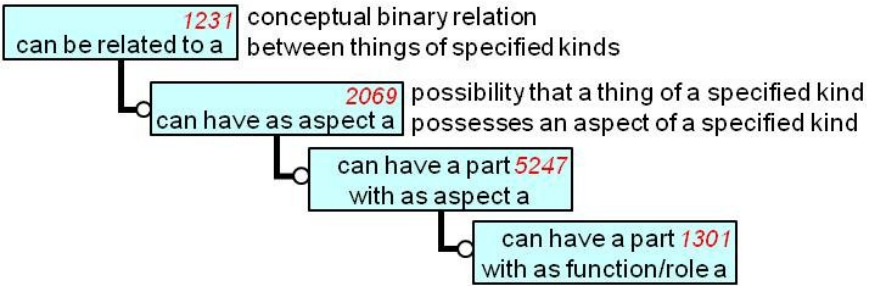


Figure 87, Kinds of aspects of a part of a kind of thing

A <can have a part with as aspect a> (5247) relation specifies that any individual physical object of a specified kind can have some part

that has an aspect of a specified kind. A relation of this kind implies that such an individual thing has an individual part, although that part may remain implicit, or may be specified explicitly in another expression. That implied part has an aspect of the specified kind.
This kind of relation has as further subtype:

- can have a part with as function a
  [or] can have a part with as role a (1301)
  This specifies that any individual physical object of a specified kind can have a part that has a role of a specified kind. Such a role may be called a function of the part. However, a function is normally an (intended) activity or process.

10.3.5 Definitions of kinds of intrinsic aspects

As discussed in par. 8.1.2, a kind of intrinsic aspect is a kind of aspect that by definition is possessed by a particular kind of things. The possessor is typically denoted by the fact that the name of the kind of intrinsic aspect includes the name of a kind of physical object that is the possessor. For example, the kind of intrinsic aspect 'pipe diameter' includes the name 'pipe' in its name. For human beings the intrinsic relation with a pipe can be interpreted from such a name. However, for computers it is necessary to specify that relation explicitly as decribed below.

Such a kind of intrinsic aspect is not an ordinary aspect, but it is a role of an aspect, because it is a role that is played by an aspect in a 'possession of aspect' relation with a possessor of a pre-defined kind. The definition of an intrinsic aspect can therefore be modeled by specifying not only a specialization relation with (a subtype of) role, but by specifying also two other relations: one relation with the kind of aspect that plays the role and the other relation with the kind of thing (physical object) that is by definition its possessor. Thus, for example:

pipe diameter	is a kind of	intrinsic aspect
pipe diameter	is by definition an intrinsic aspect of a	pipe
pipe diameter	is by definition an intrinsic	diameter

10.4 Kinds of occurrences

10.4.1 Conceptual involvements in occurrences

Kinds of things can be involved in particular kinds of roles in kinds of (higher order) relations to model the involvements e.g. in kinds of occurrences. Formal English should enable to specify the knowledge of possibilities, requirements for involvements and definitions of kinds of occurrences that imply particular kinds of things that are involved.

Conceptual involvements are presented in Figure 88.

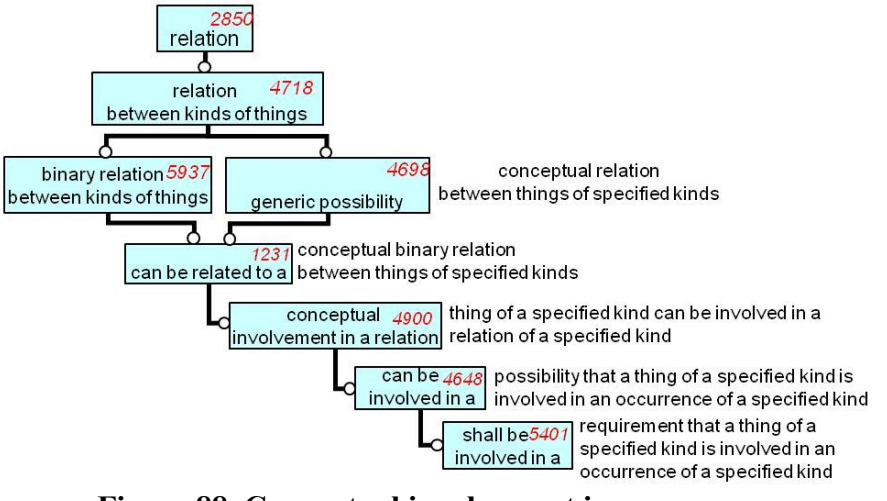


Figure 88, Conceptual involvement in occurrences

To enable to express more precisely in which kind of role some kind of thing can be or shall be involved requires the use of subtypes of the <can be involved in a> relation.

A number of examples of such subtype are given in Figure 89.

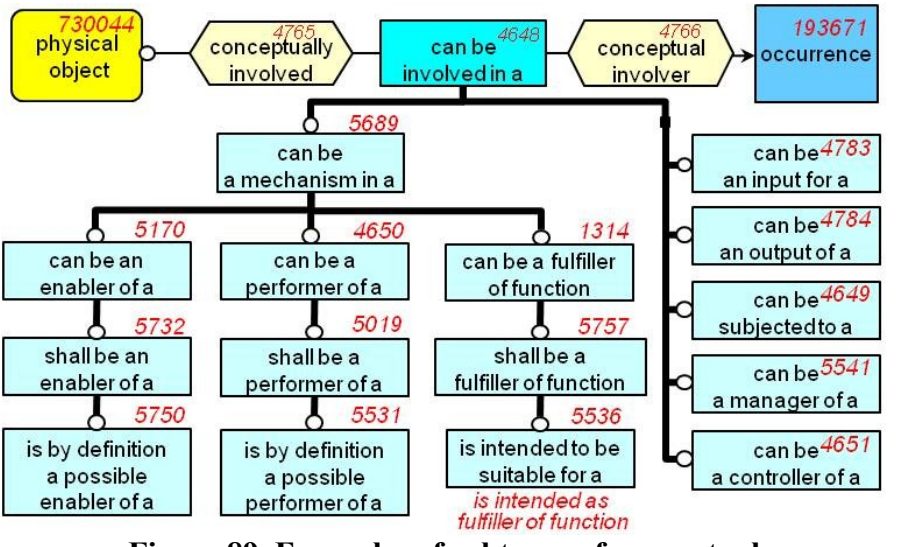


Figure 89, Examples of subtypes of conceptual involvement relations

Additional subtypes are available in the upper ontology section of the Gellish language dictionary.

10.4.2 Conceptual sequences of occurrences

A conceptual sequence of occurrences relation is defined as a relation between two concepts, the concept ‘occurrence’ in a role as conceptual temporal predecessor and the same concept ‘occurrence’ in a role as conceptual temporal successor. In other words that definition specifies the knowledge that ‘an occurrence can occur after another occurrence’. In general that statement can be expressed in Formal English as:

occurrence can occur after a occurrence

Figure 90 gives a graphical representation of that possible sequence relation.

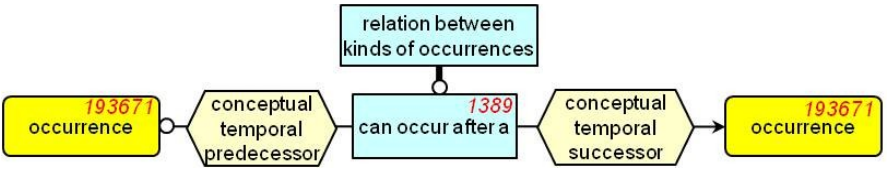


Figure 90, Example of a recursive relation

Although the predecessor and successor are the same concept, the definition of the kind of relation includes that the two roles may not be played by the same individual occurrence. In other words it specifies that an individual occurrence cannot occur before or after itself.

Recursive relations

The <can occur after a> relation is a typical example of a recursive relation. A recursive relation relates a concept (a specified kind) to itself. In general relations between kinds of things may be recursive. However, their meaning is not really recursive, because a recursive relation means that an individual thing (instance) of the specified kind can be related to *another* individual thing of the same kind, but the individual thing cannot relate to itself.

The defined kind of relation <can occur after a> enables to make similar statements about relations between two different subtypes of occurrence.

For example, the kind of relation can be used to specify:

stop of motor	can occur after a	start of motor
activity	can occur after a	activity

Note that the general definition of such kinds of relations does not put constraints on the related kinds of things, other than the specification that they must be subtypes of the related kind. This means that a stop can occur after another stop, etc. The definition does not even specify whether the related occurrences are about the same or about different things (e.g. the stop of the same or of another motor). If such constraints apply, then constraining relations shall be specified between the roles, as is illustrated for relations between individual things.

10.5 Realization of knowledge and requirements

Knowledge models and requirements models about kinds of things can be used basically in two ways: as a basis for creating information models about individual things and as a basis for verifying whether individual things satisfy the requirements that are expressed in requirements models.

The way in which knowledge models about possibilities in general can be used for creating expressions of ideas that form information models about individual things is illustrated below.

Assume that a data sheet of a centrifugal pump expresses a requirement for three impeller diameters, as given in Table 40.

Impeller diam.	Minimum		mm
	Actual		mm
	Maximum		mm

Table 40, Expression of requirements on a data sheet

The requirements in Table 40 can be transformed into modeled requirements as follows.

Name of left hand object	Name of kind of relation	Name of right hand object
centrifugal pump	shall have as part a	impeller
impeller	shall have as aspect a	minimum diameter
impeller	shall have as aspect a	actual diameter
impeller	shall have as aspect a	maximum diameter
diameter	shall be quantified on scale	mm

Table 41, Requirements for impeller diameters

The first line in Table 41 expresses that any individual thing that is classified as a centrifugal pump shall have a part that is classified as an impeller. Such a requirement is unconditional for all individual things that are classified by the specified kind. Thus in this example it applies for all centrifugal pumps. However, from a semantic modeling perspective there is a condition. That condition is: ‘there is an individual thing that is classified as a centrifugal pump’. If that

condition is satisfied, then it has as consequence that the requirement holds for the classified individual thing. Therefore we call such a requirement a quasi-unconditional requirement. This is illustrated in Figure 91.

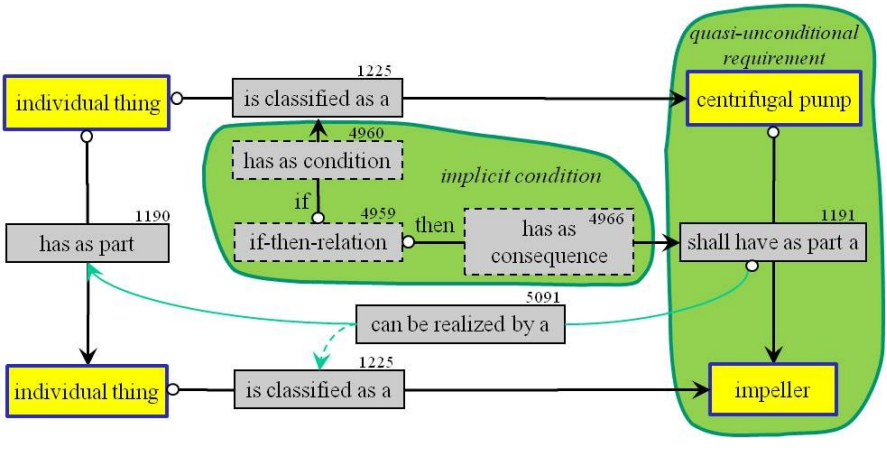


Figure 91, Realization/fulfillment of a requirement

In Figure 91 the requirement is presented at the right hand side of the figure. The condition is modeled as an 'if-then-relation' that has as component a 'has as condition' relation with the condition, which is the existence of a particular kind of relation (in the example the existence of a classification as a centrifugal pump). If that condition is satisfied then it has as a consequence that the requirement is applicable for the classified individual thing. The latter logically means that there shall be a 'has a part' relation between the classified individual thing and another individual thing and there shall be an 'is classified as a' relation between that part and the concept impeller. Thus software should be able to deduce what kind of relation should result from a particular kind of requirement relation. The upper ontology of Formal English therefore should (and does) specify what kinds of relations satisfy what kinds of requirement relations. Some examples are given in Table 42.

Name of left hand object	Name of kind of relation	Name of right hand object
shall have as part a	can be realized by a	has as part
shall have as aspect a	can be realized by a	has as aspect

Table 42, Realization of requirements

10.5.1 Design proposals

Based on requirements models as described above it is possible for software to create design proposals.

For example, the above logic holds for the satisfaction of requirements such as the requirement that an impeller shall have as aspect a diameter.

Assume that, given the above requirements model, there is an individual, called P-1 that is classified as a centrifugal pump (or as one of its subtypes), as follows:

P-1	is classified as a	centrifugal pump

Then, based on the requirements that are expressed in Table 41 it is possible that (design) software would make a proposal for a piece of an information model for the individual P-1 as given in Table 43.

P-1	has as part	I-1	
I-1	has aspect	D-1	
I-1	is classified as a	impeller	
D-1	is classified as a	minimum diameter	
D-1	has on scale a value equal to	500	mm

Table 43, Proposal derived from requirements

The proposal by software should enable a user to specify names for the new things (e.g. I-1 and D-1) and for values, such as 500, whereas the software could allocate UIDs for the new things.
Such a proposal can be presented to a user in various forms and lay-outs.

For example, it may be presented on a data sheet as follows:

centrifugal pump	P-1
impeller minimum diameter	500 mm

This presentation leaves the individual impeller and diameter nameless and thus without an identifier and it uses the layout to suggest relations of various kinds, such as the (correct) suggestion that something that is called P-1 is classified as a centrifugal pump and that some unnamed impeller is part of that pump and that the diameter is possessed by the impeller (and not by the pump).

Similarly to derivation of design proposals from requirements, software can also derive information models about individual things from knowledge models. This follows from the fact that what is required should be possible. Possibilities are prerequisites of requirements. Therefore kinds of relations that express requirements (being denoted by 'shall have…' relations) are subtypes of relations that express possibilities (denoted by 'can have…' relations). This means that the upper ontology of a formalized language should (and in formal English it actually does) specify what particular kinds of *possibilities* in general can be realized by individual ideas of related particular kinds. Such possible realizations are formally specified in the Upper Ontology section of the formal dictionary.

This is expressed for example as follows:

Name of left hand object	**Name of kind of relation**	**Name of right hand object**
can have as part a	can be realized by a	has as part
can have as aspect a	can be realized by a	has as aspect

Table 44, Realization of possibilities

By inheritance follows that the requirements relations (as expressed in Table 42) are also satisfied by the kinds of relations that are given on the right hand side of Table 44.

10.5.2 Verification of requirements

Knowledge can also be used for verifying whether a given information model about an individual thing of a specified kind satisfies requirements as expressed in a requirements model or whether it uses the knowledge about possibilities for individual things of such a kind.

The procedure to execute such a verification is the inverse process of making a proposal. For example, for an information model of an individual centrifugal pump P-2 it can be verified whether it has one or more parts that are classified as an impeller and whether those parts have specified minimum diameters, etc.

10.6 Properties and inheritance

Individual physical objects can have many aspects. Aspects of individual physical objects are defined in Formal English as separate (although dependent) individual things. Such individual aspects are related to their possessor by a possession relation. In order to define what they are, they shall be classified and qualified (or quantified) explicitly.

For kinds of physical objects it can be specified that objects of such a kind can have aspects of particular kinds. This kind of relation is defined such that it reflects the reality an object of that kind in reality (possibly but not necessarily) has such an aspect with a particular value, but that that value might not be recorded. If it is specified that objects of that kind (in a particular context) *shall* have such an aspect, then it is meant that objects of that kind not only have aspects of that kind, but also *that the values shall be recorded*.

Ideas that are expressed as relations between kinds of things (knowledge, requirements and definitions) are inherited by subtypes of the related things.

Consider for example the question “does P-1 have a diameter?”. This does mean: is there an individual aspect that is possessed by P-1, whereas the aspect is classified as a diameter *and of which its value is recorded*?

A database may for example contain relations that expresses that a pipe can have a diameter. This can be expressed as is illustrated on the first line of Table 45.

101	3	201	7
Left hand object name	**Name of kind of relation**	**Right hand object name**	**UoM**
pipe	can have as aspect a	diameter	
P-1	is classified as a	tube	
tube	is a specialization of	pipe	
P-1	has as aspect	Diameter of P-1	
Diameter of P-1	is classified as a	diameter	
Diameter of P-1	has on scale a value equal to	20	mm

Table 45, Example of an inherited idea

All the subtypes of pipe, that have a specialization relation with 'pipe', inherit this idea! (Software that implements Formal English shall ensure that this is the case). In other words all kinds of things that are subtypes in the specialization hierarchy of "pipe" inherit that they can have a diameter. On the basis of the third line in the above table (which is a statement that is typically contained in the taxonomic dictionary) a tube is a specialization (kind or subtype) of pipe. As the second line specifies that P-1 is classified as a tube, the inheritance rule implies that also object P-1 can have a diameter. These ideas do not imply a requirement for recording a numeric value according to a mass scale.

There are two options to make use of the first three lines:

1. When an individual object (such as P-1) is created and classified, then software can propose to give it a property, such as a diameter, whereas the property can be *allocated* to the item by specifying a <has as aspect> relation.
2. When individual objects are classified and they have properties, then those properties can be *verified* against the kinds of properties of the kinds of things that are defined and inherited from the specialization hierarchy of kinds of things.

If the first line in Table 45 would have used the kind of relation <shall have as aspect a>, then only the last three lines would satisfy that requirement, because the classification and the quantification of the diameter shall be as required.

10.7 Explicit modeling of roles

Roles that are required and played can be made explicit by replacing one line in an Expression table by four lines: two lines describe the roles required by the relation and the other two describe which objects play those roles.

For example, the concept of 'classification of an individual thing' (by a kind of thing) is defined in the Upper Ontology section of the Gellish Dictionary. That concept, which is represented by the phrase <is classified as a>, is defined as being a kind of relation between an individual thing and a kind of thing and by four additional defining elementary ideas that specify which kinds of roles are by definition involved in such a relation and which kinds of role players can play a role of such a kind. Thus the definition consists of five lines in an Expression table as is illustrated in Table 46.

101	1	3	15	201
Name of left hand object	**UID of idea**	**Name of kind of relation**	**UID of right hand object (UID of kind role)**	**Name of right hand object (name of kind of role)**
is classified as a	1.007.363	is a specialization of	4.719	is related to a
is classified as a	1.003.840	has by definition as first role a	3.821	classified individual thing
is classified as a	1.003.573	has by definition as second role a	3.822	classifier for individual thing
individual thing	1.001.423	can have a role as a	3.821	classified individual thing
kind	1.001.215	can have a role as a	3.822	classifier for individual thing

Table 46, Elementary ideas about roles in relations

Note 1: The right hand objects on the last four lines in Table 46 are kinds of roles. Therefore, they are defined in the Gellish Dictionary as concepts and thus they are part of the overall specialization hierarchy of concepts, with their own branch with a hierarchy (taxonomy) of kinds of roles.

Note 2: The fourth line in Table 46 defines that *an individual thing* (i.e. a member of the concept 'individual thing') can have a role as 'classified individual thing', whereas the last line defines that *a kind* (i.e. a member of the concept 'kind', which is 'individual thing' *or one of its subtypes*) can have a role as a 'classifier for an individual thing'. This is compliant with the assertion that the relation <is classified as a> is defined as being a subtype of 'relation between an individual thing and a kind of thing', as is specified on the first line.

Kinds of roles can also be made explicit by mentioning them explicitly as part of expressions. This is illustrated in Table 47.

101	1	3	15	201
Name of left hand object	**Name of left hand (kind of) role**	**Name of kind of relation**	**Name of right hand (kind of) role**	**Name of right hand object**
D-1	classified D-1	is classified as a	classifier of D-1	door
door	possessing door	can have as aspect a	height of a door	height
individual thing	conceptual player of a role	can have a role as a	conceptually played role	classified individual thing

Table 47, Explicit roles in expressions

Note that on a high level the kinds of roles that are involved in kinds of relations are explicitly defined it the upper ontology of the language definition. For example the roles 'classified' and 'classifier' are defined for <is classified as a> relations in general. This makes that it is rarely useful to explicitly mentioning the individual roles, such as in the case if the classification of D-1 on the first line of Table 47. Also the kinds of roles on the third line are already defined in the upper ontology of the definition of the concept < can have a role as a>. Constraints are often only applicable for kinds of things *when they play a particular kind of role*, thus in a particular *context*. As those constraints are not generally applicable for all things of a kinds, it is necessary to explicitly model their kinds of roles in order to be able to put constraints on them. Thus, explicitly modeling kinds of roles are important only to enable the specification of constraints on their role players in that context. This could for example be the case for expressions such as the second

row of Table 47, when there are constraints on door heights put in a particular ‘validity context’ (for the expression of validity contexts see paragraph 8.2 and 13.4.

10.8 Information requirements

Requirements to deliver particular kinds of information for products of particular kinds differs from requirements that specify the required and allowed content of a database. In this paragraph we describe how those different requirements can be expressed in Formal English

10.8.1 Product information requirements

Organizations or individual persons can express requirements for information that should be delivered by other parties, typically together with the delivery of products or services. Such hand-over requirements specify for example that for each product of a given kind it is required that particular ideas shall be documented when they are applicable for an individual product of that kind. Conventionally such requirements are expressed as requirements to deliver and fill-in some standard ‘data sheets’ or ‘spec-sheets’ for each individual product of the appropriate kinds. This caused that many organizations have developed standard forms for various kinds of products.

A modern way of specifying such requirements is to specify that such information shall be delivered in electronic form. This has resulted in the conversion of paper standard forms into electronic versions of those forms. However, those delivered electronic files could usually not be imported directly in existing databases.

Requirements that are expressed as Formal English information models that apply kinds of relations that express requirements enable computer supported verification as well as import in existing databases. Such information models are similar to knowledge models that express possibilities. The main difference is that kinds of relations of the type <can have ...> are replaced by relations of the type <shall have ...>. For example, it might be specified that for each compressor it is required to provide data for a database, such as its

name, its capacity, its impeller as one of parts, and the diameter of that part, as well as a data sheet (or a dedicated compressor data sheet). This can be specified as follows:

equipment item	shall have as name a	text string
compressor	shall be described by means of a	data sheet
compressor	shall have as aspect a	capacity
capacity	shall be quantified on scale	dm3/s
compressor	shall have as part a	impeller
impeller	shall have as aspect a	diameter

Etc.

Note that these requirements imply that for each individual thing of the specified kind such a requirement holds.

Some of the requirement might not be applicable, or might only be applicable for subtypes of the specified kinds. For example, requirement for an impeller as part is not applicable for reciprocating compressors, but only for centrifugal compressors. This means that the third requirement cannot be satisfied for the subtype ‘reciprocating compressor’. Therefore instead of the above requirement for an impeller it is better to specify:

centrifugal compressor	shall have as part a	impeller

In a conventional database it is not so simple to specify such requirements for subtypes of kinds of things if each subtype is represented by an entity type.

10.8.2 Specification of allowed values

In many cases there are constraints specified for the values that are allowed as values for required or possible aspects. Those allowed values can be either enumerated lists of values, also called pick lists, or they can be specified as a range for numeric values.

For example:

model X	shall have a color from the list of	model X colors
model X	shall have a height within range	20-30 mm

Note that these two expressions are not valid standard Formal English expressions, because the kinds of relations are not in the dictionary, because for maintaining flexibility the generic possession of aspect relation is not subtyped according to its kind of aspect.

Typically, a constraint for values of an aspect of the specified kind is applicable only in the context of a specified kind of possessor. In other words, the pick list or allowed range is not applicable for every value for an aspect of the specified kind. The constraint is only valid for aspects of the specified kind in case the aspect plays a role of being possessed by a possessor of the specified kind. In other words, the constraint is applicable for a kind of intrinsic aspect (which is a kind of role that is by definition played by an aspect towards a kind of possessor).

Therefore, to express such a constraint it is required to define an explicit kind of intrinsic aspect, which is a role that is played by the specified kind of aspect and then specify the constraint on that kind of intrinsic aspect, so that it is applicable only for aspects that play such a role. The latter specification of the constraint defines that the value for an aspect in such a role <shall be one of the> pick list collection or range of aspect values. For example,

model X shall have as aspect a color

This implies that there is a kind of intrinsic aspect, called ‘color of model X’, which can be made explicit in an Expression table as follows:

Name of left hand object	Name of kind of relation	Name of kind or right hand role	Name of right hand object
model X	shall have as aspect a	color of model X	color

Note that in the above Expression table the concept ‘color of model X’ is not explicitly defined. An explicit definition of the kind of intrinsic aspect is given by a corresponding alternative expression of the requirement that

model X shall have as intrinsic aspect a color of model X

This requires an explicit definition of the kind of intrinsic aspect. This is done by the expression of three assertions that relate it to the concepts ‘intrinsic aspect’, ‘aspect’ and to the concept that is by

definition its possessor. Such a definition is expressed for this example as follows:

color of model X	is a specialization of	intrinsic aspect
color of model X	is by definition an intrinsic	aspect
color of model X	is by definition an intrinsic aspect of a	model X

Once an intrinsic aspect is introduced it becomes possible to use it in expressions of constraints, such as:

color of model X	shall be one of the	model X colors

The definition of a (constraining) collection of enumerated qualitative aspects is discussed in chapter 11. The definition of a (constraining) range was described in paragraph 8.2.

10.8.3 Database information requirements

The specification or definition of databases conventionally includes the creation of a data model (also called schema), typically starting with a conceptual data model, which is converted in a logical data model, which is then converted in a physical data model.

Usually, such a database development process implies the specification of a specific language by defining the following components:

- The specification of the terminology. For example in the form of names of entity types, names of attribute types and allowed values for attributes.
- The specification of the structure of the expressions. For example in the form of table columns for the attributes, which imply relations of particular kinds between those attribute types, and thus imply relations between the attributes.

Note that the use of Gellish implies that such a language definition is not required any more, apart from possible necessary extensions of the Gellish dictionary.

The third component that need to be specified is:

- The specification of the information requirements and storage possibilities.

Note that a database definition does not specify *that* information should be delivered or imported. Only, if some data are entered, then it specifies which other related data can, or shall, or is allowed to be entered as well.

Sometimes detailed requirements models, such as the above requirements for compressor data, are used for definition of databases. However, this results in databases that only allow for the entry of such data. Generally databases will be designed for more flexibility. For example to enable the entry of other equipment data as well. However, that means that the above detailed requirements are usually not reflected in a database definition for equipment data.

Furthermore, the above requirements model for compressor information is independent of the structure of a database. For example a database may have an entity type 'compressor' or a more general entity type 'equipment item' and may or may not have a separate entity type for 'impeller', etc.

Requirement models in Formal English can specify the detailed information requirement only and do not need separate specifications of database requirements and specifications of storage capabilities, because Formal English expressions don't have the constraints that data structures of databases imply on the information.

10.9 Conceptual correlations and physical laws

Conceptual correlations and physical laws are higher order kinds of relations between kinds of things. They express knowledge that aspects of particular kinds are correlated. Such a correlation usually holds under the condition that such correlated aspects are possessed by the same physical object or the same collection of interacting physical objects.

This is illustrated in Figure 92.

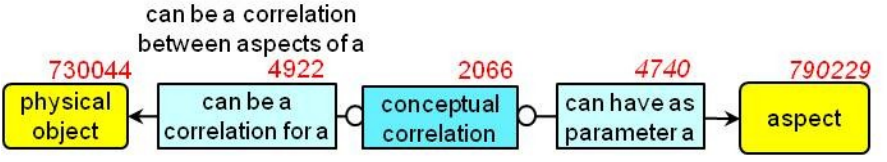


Figure 92, Correlated aspects of a physical object

For example, the physical law, as is discovered by Newton states that F = m * a. This law correlates three kinds of aspects, a force F, a mass m and an acceleration a, each of which should be possessed by or operate on the same thing of a specified kind (in this case a physical object).

The definition of a law should be implemented in an executable program with the proper kinds of aspects that act as parameters in the correlation. Such a program should be able to calculate the third parameter value when two other parameter values are known, and it should be able to verify whether the values are consistent in case all parameter values are known.

In the example of the Law of Newton the three parameters are specified in the model as follows:

Law of Newton	is a qualitative subtype of	x * y – z = 0
Law of Newton	has by definition as parameter a	force
Law of Newton	has by definition as parameter a	mass
Law of Newton	has by definition as parameter a	acceleration

Figure 93 presents a number of subtypes of conceptual correlation.

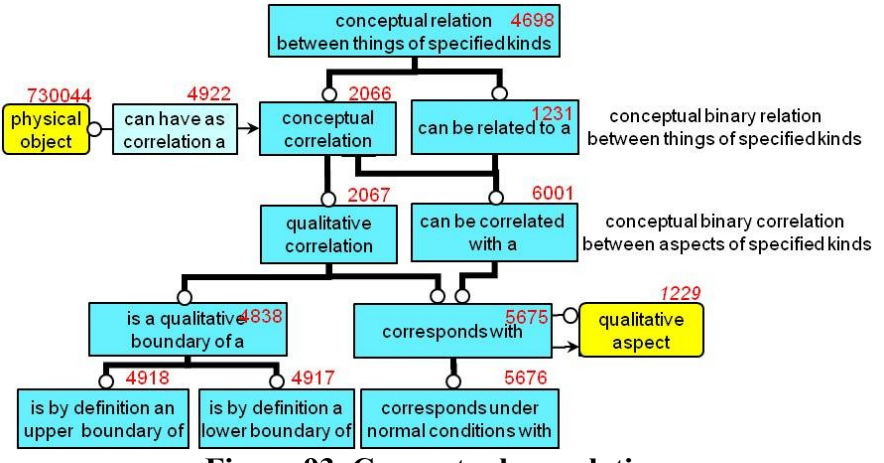


Figure 93, Conceptual correlations

The <corresponds with> relation is a binary correlation between two qualitative aspects. It can be used for example to express that a kind of substance water corresponds with a molecular weight of 18. Its subtype <corresponds under normal conditions with> specifies a correspondence under the condition that the pressure and temperature is at 'normal atmospheric conditions' (also called a normal state), which is defined as being room temperature (20 degree C) and average atmospheric pressure at sea level (1013 mbar). Thus we can state that water at normal conditions corresponds with liquid. These examples can be expressed as follows:

water	corresponds with	mw 18
water	corresponds under normal conditions with	liquid

Whereas mw 18 is defined as:

mw 18	is a qualitative subtype of	molecular weight
mw 18	is by definition quantified on scale as	18

A relation between a qualitative aspect range and its boundary value is a binary correlation. Such a boundary value is not defined as an aspect of the range, because aspects cannot have aspects, but aspects of physical objects can be correlated. For example, a boiling range

of some mixture between 80 and 100 degC can be related to its boundary values as follows:

80-100 degC	has by definition as lower boundary	80 degC
80-100 degC	has by definition as upper boundary	100 degC

10.10 Conditional consequence relations (if-then)

In paragraph 10.5 we stated that general requirements imply a condition for a consequence. Such requirements were called quasi-unconditional requirements.

There is another category conditional consequence relation: a qualitative if-then-else relation (5775). This is a higher order relation which use is illustrated in Figure 94.

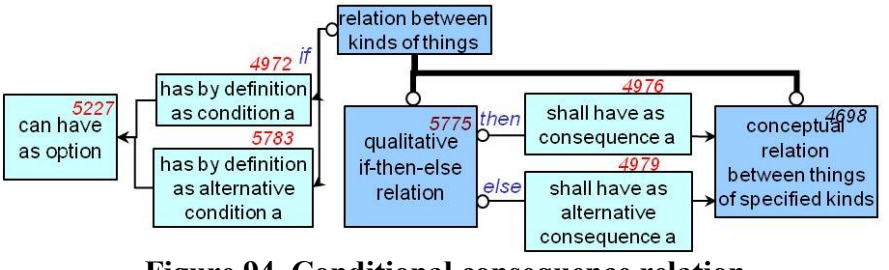


Figure 94, Conditional consequence relation

An if-then-else relation can have one or more options for conditions (that are expressed as relations). Such a condition is typically one of the options for a value of an aspect. For example, a conditional action (CA type-1) to switch a light on, under the condition that a valve is open, can be expressed as follows:

CA type-1	has by definition as condition a	option type-1
CA type-1	shall have as consequence a	action type-1

Whereas option type-1 is a condition relation of the following kind:

valve position	can have as option	open

There can be more than one of such conditions, which means that all such conditions have to be satisfied. Furthermore, there can be two groups of conditions. The second group consists of ‘alternative conditions’, which mean that either the conditions in the first group or the conditions in the second group may be satisfied.

If the condition(s) are satisfied, *then* a consequence of a particular kind (action type-1) shall be performed. An example of an action type-1 might be: ‘switch a light on’. Such an action type is a kind of occurrence, which can be specified in further detail as is described before.

If the condition(s) are not satisfied it might be the case that an alternative kind of action should be performed (an else-clause).

11 Relations with collections

Collections of things should be distinguished from kinds of things. A kind is defined independent of its number of members. A collection has by definition a particular number of elements at a particular moment in time and its number of elements may vary over time. Collections may consist of things that are collected and brought together, but may also be as-if brought together. Collections can be unordered or ordered. Examples of kinds of ordered collections are as lists and tables.

11.1 Relations between single things and collections

We can distinguish three categories of collections: collections of individual things, collections of kinds of things, and mixed collections that include individual things as well as kinds of things.

Figure 95 presents the hierarchy of relations between a single thing and one of these three categories of collections.

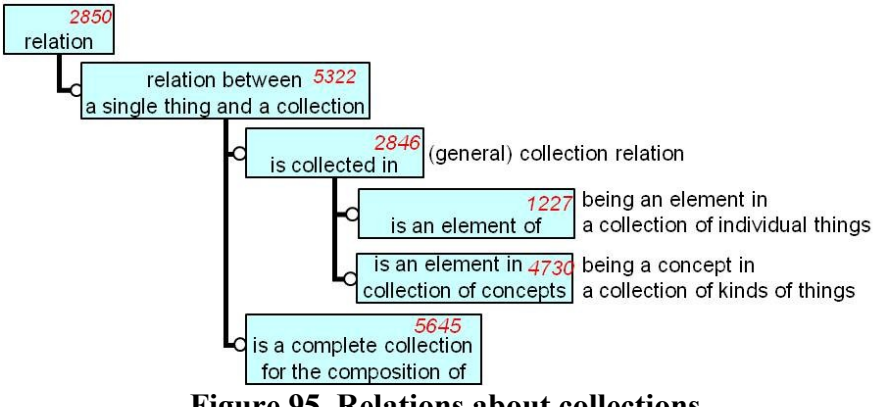


Figure 95, Relations about collections

The general collection relation (2846) in Figure 95 can be used to specify that some single thing is an element of any of the three categories of collections, its first subtypes specifies that an individual thing is an elements of (1227) a collection of individual things. The second subtype specifies that a kind of thing is an element in a collection of kinds of things (4730).

The other relation between a single thing and a collection specifies that the collection consists of all elements that are required to compose a particular individual thing.

Figure 96 illustrates how these kinds of relations can be applied.

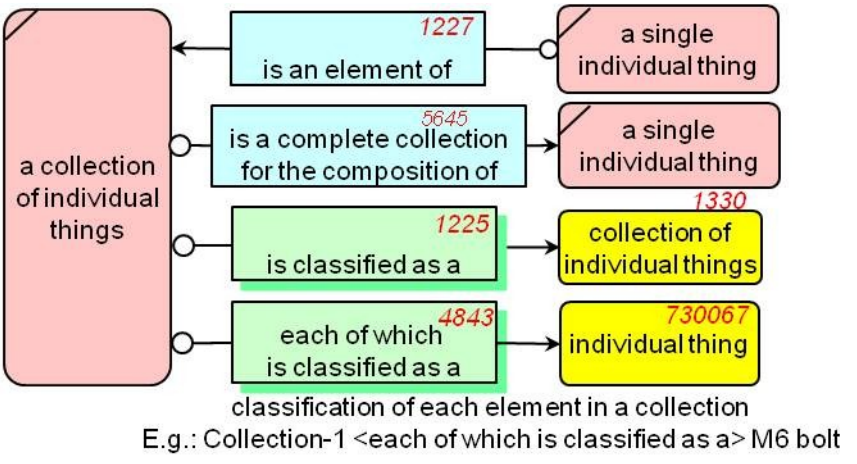


Figure 96, Individual things in collections of individual things

The left hand side of Figure 96 represents an individual collection that is composed of individual things. This is specified by the statement that it is classified as a collection of individual things. A number of elements can be collected into that collection. The statement that an individual thing is an element of a collection of individual things is expressed as in the following example:

Car-123 is an element of Company X car fleet

Instead of classifying each individual component in a collection of individual things, it is also possible to collectively classify each element in a (homogeneous) collection by one statement as is illustrated in the last line of Figure 96. This enables to work with classified elements in a collection, although the individual things that make up the collection may not be identified individually.

A collection of individual things may have aspects, such as a 'number of elements' or a (total) weight, just as single individual things can have aspects. Furthermore it can be specified that a

collection is complete and sufficient to be used to compose another individual thing.

For example, the collection of components that make up an assembly. This is illustrated on the second line in Figure 96.

Figure 95 also presents a relation that specifies that a kind of thing (a concept) is an element of a collection of kinds of things (4730). For example a company may define a collection of kinds of categories to list the kinds of things that they keep in stock. Then the idea that a kind of thing (such as spare part type 'xyz') is an element of the collection of kinds of stock items is expressed as in the following example:

xyz is an element in collection of concepts kinds of stock items

There are also other kinds of relations available that specify collective relations for each element in a collection. For example:

- A particular collection of information items consists of information items, <each of which includes information about> (5047) some object, whereas the object can be either in individual thing or a kind of thing.
- A collection of concepts consists of concepts, <each of which is a subtype of> (5095) a particular supertype concept. This specifies that the elements in the collection of concepts all are subtypes of the supertype concept.

11.2 Relations between collections

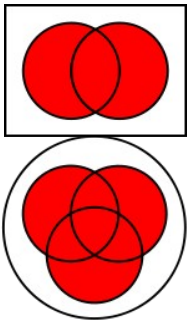
Relations between collections in general (4748) apply to all three categories of collections: collections of individual things as well as collections of kinds of things and mixed collections.

In set theory relations between collections are usually taken as operations on collections. Strictly speaking this could be interpreted as a change of state, so that the state before execution of the operation differs from the state after execution of the operation. This would mean that we can distinguish between the state of a collection before and after the operation. This might be described as a change of state process of actions that take place in time, whereas the number of elements in the collection varies over time. However, in

most cases a time that an operation takes place is not specified and is not meant to be specified. Thus usually an ‘operation’ describes a state after execution of an operation.

Furthermore, relations between collections are usually higher order relations, which involve three or more collections. Therefore, in general those relations shall be expressed in a similar way as correlations are expressed, which means that multiple binary relations are required to express one higher order relation.

Unions of collections



For example a union (operation or calculation) between collection A and collection B and possibly more collections delivers a collection C (often denoted as C = A ∪ B). In Formal English such a union operation is described by multiple binary statements, one for each collection that is united in a uniting collection. For example, assume that it holds that two collections (A and B) are united in a third one (C). This can be expressed as follows:

A	contains elements that are united in	C
B	contains elements that are united in	C

Then C contains all elements that are present in A as well as those that are present in B. It is a general rule in set theory, that if the united collections (such as A and B) contain (partly) the same elements, then the uniting collection (C) will only contain such items once (thus an element will not be duplicated, as an element can only exist once). Therefore potential duplicate relations between an element and collection C shall be eliminated. For example, assume that:

E-1	is an element of	A
E-2	is an element of	A
E-2	is an element of	B
E-3	is an element of	B

Then the above union statements imply that collection C consists of the elements E-1, E-2 and E-3. Note however that the union

statements do not imply that new statements are generated to express in an explicit way that E-1 is an element of C, etc. Thus, the union statements imply that during a search for the elements in C, the software should also search for and present the elements in A and B (while preventing to present duplicates).

It may also be that elements are explicitly stated to be an element of collection C, whereas those elements may be or may not be also elements of A or B. For example:

E-3	is an element of	C
E-4	is an element of	C
E-5	is an element of	C

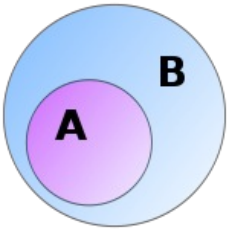
Because in this example E-3 is also an element of B directly, it will nevertheless appear only once in collection C.

Furthermore, it is possible to make an exception from being collectively declared to be united. This can be expressed by stating that an element is excluded from being collectively declared to be united in a collection. For example:

E-1	is excluded from	C

This statement overrules the consequence of the first union statements, so that E-1 is not an element of collection C although is an element of A and was initially stated to be included in C. Such an exclusion does not overrule an explicit statement about being an element of the collection.

Subsets and supersets of collections



A subset-superset relation is a binary relation between (two) collections that specifies that the elements in a collection (A) are a subset of the elements in another collection (B). This is independent of the question whether the elements in A and B are explicitly identified or declared to be elements of the collections.

For example, it may be that collection B is populated, so that its elements are explicitly declared to be an element of B. Then assume that it is stated that

A is a subset of B

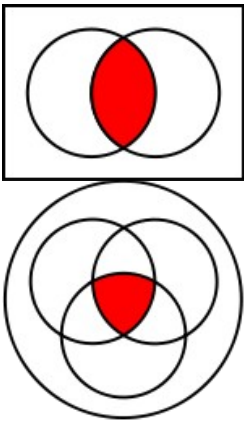
Then this statement only means that the elements of collection A are also elements of collection B, without specifying which elements belong to collection A.

The inverse statement is that:

B is a superset of A

Note that the above union statement, which states that the elements of A are united in C does not necessarily imply that C is a superset of A, as some elements may be excluded from being collectively included.

Intersections of collections



Intersection of sets as defined in set theory is a relation between a collection of sets and an intersection collection that indicates that the elements of the intersection collection consists of all the elements that are member of each of the intersected collections and no others.

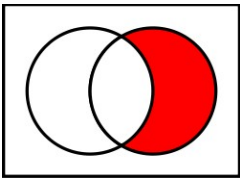
Thus to express an intersection (an intersecting collection) I of the collections A, B and C it is required to first define a collection of collections (CC). That collection has three collections as elements. This is expressed as follows:

A	is collected in	CC
B	is collected in	CC
C	is collected in	CC

Then is can be specified that

CC are sets with as intersection I

Differences of collections



A difference between sets, also called a relative complement of sets, as defined in set theory is a correlation between three collections and is usually expressed as A\B = C (or also written as A - B = C). This can be expressed as follows:

A	contains elements that are united in	C
B	contains elements that are subtracted from being united in	C

The definition of the subtraction relation (<contains elements that are subtracted from being united in>) specifies that it overrides statements that specify that elements are united in. This makes that the statements are independent of the sequence in which they appear in a database.

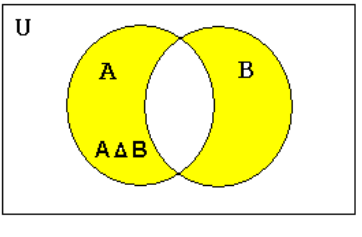
There is another kind of difference that can be expressed as a binary relation. It is a relation between a collection of collections and a resulting collection that indicates that the elements of the resulting collection consist of all the elements of the united sets except for the elements in the intersection of the sets. In other words the resulting collection consists of the elements that appear in only one of the compared collections.

For a difference of two collections A and B the difference can also be expressed as (A\B) ∪ (B\A).

But using a binary relation, starting from the same set of sets CC as above, we can express:

CC	are sets with as difference	D

In this example collection D is a collection of elements that are elements of A, B and C, but excluding the elements that are in more than one of A, B and C.



Note that for two collections A and B this relation expresses the same as a ‘symmetric difference’ between A and B as is defined in set theory. However, for more than two

collections the resulting collection differs from a symmetric difference.

12 Integration with natural languages, algorithms & documents

A database or message header specifies in which (formal) natural language the content it expressed. In addition to that, each expression (each line in an expression table) may contain a statement about the language (and the language community) of the name of the left hand object, which can deviate from the language of the whole database or message.

12.1 Natural language text

The language of a whole expression also determines the natural language that is used for the textual description or definition of a defined object (see expression component 4 and 65 in par. 13.4.3. Those descriptions can also be used to relate the concepts in a formal English expression model to natural language text. This enables that the contents of documents can be incorporated in a model, by including complete sentences or paragraphs in a model, without the need to explicitly model those sentences in detail.

This is illustrated in the following table.

UID of left hand object	Name of left hand object	UID of kind of relation	Name of kind of relation	UID of right hand object	Name of right hand object	Full description
130,069	compressor	5,298	shall be compliant with	106	API 617 par. 5.3.1	
106	API 617 par. 5.3.1	1,726	is a qualitative	970,007	requirement	The purchaser and the vendor shall mutually determine the measures that must be taken to comply with any governmental codes, regulations, ordinances, or rules that are applicable to the equipment.

Table 48, Incorporation of natural language text

Table 48 illustrates that any compressor shall be compliant with the API 617 paragraph 5.3.1 standard (according to some party or standard, which is specified in the 'validity context'). On the next line the text of that paragraph is specified in natural language English. In this way each separate requirement from such a standard can be incorporated in a formal English semantic model and related to the component to which it applies!

In addition to that it might have been specified that the paragraph is an integral part of the API 617 standard, whereas also the sequence of the paragraphs can be modeled as well as the distribution of paragraphs over chapters, etc. In that way whole documents can be modeled and later composed from the models. This simplifies maintenance of documents, especially of coherent sets of requirements documents.

12.2 Programming languages (algorithms) and formulae

In a similar way it is possible to incorporate expressions that are encoded in programming languages. This can be used to relate objects or activities to algorithms or mathematical formulae. The qualification of the algorithm or formula should provide sufficient information about the encoding system that shall be used to interpret the expression. This can be applied for example for the computer interpretable storage of formulae or algorithms that describe kinds of shapes (geometric objects).

12.3 Files with documents, drawings, etc.

A model can also incorporate a reference to an external electronic file, as is illustrated in Table 49.

UID of left hand object	Name of left hand object	UID of kind of relation	Name of kind of relation	UID of right hand object	Name of right hand object	Full description
130,069	compressor	5,298	shall be compliant with	101	API 617	
101	API 617	1,726	is a qualitative	910,137	standard specification	
101	API 617	4,996	is presented on	102	API 617.pdf	
102	API 617.pdf	1,225	is classified as a	493,748	pdf file	
102	API 617.pdf	1,227	is an element of	103	http:// www.gellish.net/ dictionary	
103	http://www. gellish.net/ dictionary	1,225	is classified as a	492,017	directory	

Table 49, Integration of document files and drawings in Formal English models

The names of things may also include file extensions, such as pdf, preceded by a dot, access paths for directory addresses and internet addresses (URL's or URI's). This enables software to recognize file formats and storage locations and to retrieve as file, launch an appropriate software application and thus display the content of the file or database in the application.

Note that a reference file can contain documents, drawings, databases, and any other binary encoded content.

13 Expression components

Expression components are the components that make up an expression conform the Gellish syntax. Correct interpretation of expressions require not only the components that express the main idea, but also those that provide the appropriate context. Therefor, the expression of each main idea should be accompanied by an appropriate number of contextual facts and it is insufficient when a semantic information model only includes expressions of core ideas (the topics with the intentions of the expressions). That additional contextual information implies that each core idea shall be related to a number of contextual information components. The following paragraphs discusses those expression components and the relations between them. Precise definitions of the expression components and kinds of relations between them are provided in the Gellish Syntax document (Ref. 4).

13.1 Expression of core ideas

A core idea is the bare idea that is intended to be expressed, without its context and irrespect of being an opinion or fact or imaginary state of affairs. The following relations are required for the expression of a core idea:

- A *binary relation* that relates two things that are involved in a main idea.
  The two related things are: a player of the first role that is required by the relation (usually[10] the left hand object) and a player of the second role that is required by the relation (usually the right hand object). The related things are represented in the relation by their respective unique identifiers (UIDs).

- A *classification* of the relation.
  This is a statement that classifies the relation by a (standard) kind of relation. This contextual fact is represented by a pair of things: the

10 In Formal English expressions it is allowed that inverse phrases for kinds of relations are used. However, implementation such as in Ideas tables may constrain the use of inverse phrases. In constrained tables the left hand object is always the player of the first role.

UID of the idea and a UID of the kind of relation that classifies the relation.

- A classification of a quantification relation by a scale (a *unit of measure*) – when applicable.
This is called a contextual fact that classifies a quantification relation by a (standard) kind of scale (also called a unit of measure). This contextual fact is represented by a pair of things: the UID of the idea and a UID of the kind of scale that classifies the relation.

- A qualification of the expression by the *intention* with which the expression is communicated.
This is a contextual fact that expresses with which intention the main idea is communicated. For example, it may express that the idea is communicated as an assertion or as a denial, a confirmation or a question. (default = assertion)

- Optionally: an extent of being the case.
This specifies qualitatively or quantitatively as a fraction or percentage on a scale to which extent a statement is the case. Especially the fraction of a whole that is occupied by a part or the fraction of a mixture that is classified by the specified kind of substance.

Any expression of a core idea in a formalized language should therefore consist of the above relations, whereas those relations relate six component. Those expression components can be represented by UIDs that are independent of any natural language. The components are presented in Table 50.

Component ID (column ID)	Description of object
1	UID of a idea
2	UID of a left hand object
60	UID of a kind of relation
15	UID of a right hand object
66	UID of a scale (unit of measure)
5	UID of an intention

Table 50, The core elements of an expression

When those six component are arranged in a syntactic structure that defines the above describes relations between them, then that structure forms an expression. Thus the expression of a core idea is an arrangement of six components, consisting of six UIDs, in a syntactic structure.

13.1.1 Ideas table core

A tabular syntactic structure is a possible implementation of an expression as the relations between the columns in the table define the relations between the components of the expression. Therefore the core of an Ideas table can represent the above expression components and their relations and thus enables the expression and interpretation of the meaning of main ideas. That core of an Ideas table is therefore defined by six columns, identified by a column ID. Each row in an Ideas table represents a combination of the six components, each represented by its own UID.

For example, Table 51 is a core of an Ideas table that illustrates the expression of idea 201.

1	2	60	15	66	5
UID of an idea	**UID of a left hand object**	**UID of a kind of relation**	**UID of a right hand object**	**UID of a UoM**	**UID of an intention**
201	101	5026	102	570423	491285

Table 51, An Ideas table with the identification and expression of one core idea

Each object that is represented in column 2 of Table 51 is called a left hand object and denotes the player of a ‘first role’ in a relation, as defined by the definition model of the specified kind of relation. By analogy column 15 denotes a right hand object, which is the player of the ‘second role’ in the relation of the specified kind.

The UIDs in Table 51 represent objects (things). The terms (names, etc.) of those objects in natural language are specified in expressions of separate contextual facts, which are called naming relations, as is specified in the paragraph 13.5. Those naming expressions are typically provided in a Naming table (see par 13.4.2). Replacing

UIDs or accompanying them by columns with terms that denote the UIDs delivers a human readable equivalent for Table 51.

Usually a Naming table will contain various synonyms terms for the objects that are denoted only by a UID in an Ideas table. Therefore, in principle it is necessary to record which term is used in which expression. This can be done by creating a separate Term Usage table that contains a specification of which terms are used for each UID that appears on a row in an Ideas table. However this issue is solved by using integrated Expression tables (see par. 15).

13.2 Expression of roles of role players

Each object that is involved in a relation plays a role of a particular kind in that relation. Thus each binary relation implies two roles played by the related objects.
Those roles often may remain implicit in expressions, because the kind of relation implies particular kinds of roles. For example, the kind of relation <is a part of> (a composition relation) implies the kinds of roles 'part' and 'whole'. The definition of those kinds of roles follows from the *definition* of the kinds of relations (as specified in the TOPini section of the Dictionary).

In some cases it is required to model those roles explicitly. This especially holds for the modeling of constraints, when those constraints are applicable only when objects of a kind play a role of a particular kind.
Roles can be made explicit by modeling a role as a separate object in either of two ways:

- By treating a role in the same way as a role player. This means that if an object plays a particular role in a relation, that fact is explicitly expressed by two relations:
  - A relation between the object and the role that is played by that object.
  - A relation between a relation and the role of the kind that is required by that relation.
- These kinds of expressions of ideas about roles can be stored in the same way as all other expressions of ideas. It only requires the

recording of all the roles as separate objects and explicit classification of those roles and defining kinds of roles in their own subtype-supertype hierarchy (taxonomy).

- By inserting a left hand and right hand kind of role in an orderly expression. This implies that for each main idea four contextual facts are defined: two that specify that the objects play roles of those kinds and two that specify that those kinds of role roles are subtypes of the kinds of roles that are by definition required by a relation of such a kind.

In a tabular form this means that an Ideas table is extended with two additional columns; one for the left hand kind of role and another for the right hand kind of role, whereas also the names of the kinds of roles need to be defined in a Naming table. The kinds of roles classify the played roles and are implied subtypes of the kinds of roles that are required by the kind of relation. This is implemented in an Ideas table as follows:

1	2	72	60	74	15	66	5
UID of a idea	UID of a left hand object	UID of a left hand kind of role	UID of a kind of relation	UID of a right hand kind of role	UID of a right hand object	UID of a scale	UID of an intention
201	101	301	5026	401	102	570423	491285

Table 52, Ideas table core extended with explicit roles

Thus the explicit modeling of the roles implies an extension of the core of a ideas table with the following columns:

Component ID (column ID)	Description of object
72	UID of a left hand kind of role
74	UID of a right hand kind of role

Table 53, Extension of the core of an Ideas table with roles

13.3 Expression of queries

The modeling of a dialogue (being a human activity) typically requires modeling the various communication activities as separate

occurrences. The questioning, answering, confirmation, etc. are modeled as activities that are classified by kinds of activities. However, with or without modeling the dialogue itself, it is also required to model a question or query as a message. The general model of a query message is discussed in par. 2.6.

The two components of an expression that express contextual facts are specified in Table 54.

Component ID (column ID)	Description of object
80	Left hand string commonality
81	Right hand string commonality

Table 54, String commonality columns in a Query table

These components imply additional relations between these components and the left hand term (character string) and right hand term respectively that should be interpreted from the syntactical structure of the expression.

A tabular implementation enables to interpret the relations from the definition of relations between the columns. Therefore, a query can be implemented as a Query table. Such a Query table is an Expression table that contains two additional columns (80 and 81) in which the commonality criteria for the left hand and the right hand term can be specified.

13.4 Expression of contexts

A proper interpretation of the meaning of an expression (or proposition) requires not only that the main idea (the topic) is expressed with the terms in the user preferred language and language community, but it also requires that the expression includes information about the context in which an expression is made. Therefore, semantic modeling not only requires expressions of the main ideas themselves, but it also requires that each expression is accompanied by additional expressions of facts about the main fact. Such additional facts are called 'contextual facts' about a main idea.

Contextual information is also required for the management of information. For example, for proper interpretation as well as for proper information management it should be recorded who has created an expression, when that was done, what the status of the expression is, since when it is outdated or replaced by another expression of an idea, in what language it is expressed, etc.

Each expression of an idea shall be accompanied by such contextual facts. Standard kinds of contextual facts are discussed below.

13.4.1 Language and language community contexts

Facts and idea's are basically natural language independent. Therefore in a semantic model that uses UIDs to represent things and ideas, also the expressions are basically language independent.

However, terms (names) of things are language and language community context dependent as was discussed in par. 5.1.
To relate language independent UIDs to natural language and language community dependent terms (names) it is necessary to specify naming relations.

Every relation between a term (including a phrase) that is used in any expression and the thing (UID) that is denoted by that term as well as the relations with the language and language community contexts form a collection of contextual binary relations or triples (also called a 'graph') about the expression.

The components of an expression that represent the language and language community contexts are given in Table 55.

Component ID (column ID)	Description of object
69	UID of a natural language
71	UID of a language community
101	Term (name, phrase, abbreviation, code, URI, number or symbol)
60	UID of kind of relation
2	UID of a thing that is denoted by the 'term' in the language, as originated in the language community

Table 55, Components for the expression of language and language community context

The relations between these components of an expression (as described in par. 5.1.2) are implemented through the syntactical structure or format of the expressions.

The definitions of the language and language community are given in the following paragraphs. The definition of the components 101, 60 and 2 were already provided in par. 13.1)

13.4.2 Naming table

In a tabular implementation the relations between the components are defined by the definition of the relations between the columns in the table. For example, each of such a collection of contextual relations can be represented in tabular form as one Naming table, provided that the relations between the columns in that table represent the kinds of relations for that collection.

Such a Naming table therefore has the following table header:

69	71	101	60	2
UID of a language	UID of a language community	Term	UID of kind of relation	UID of a named thing

Table 56, Header of a Naming table

The columns 69, 71 and 101 together form a unique key.

The columns have their own column ID's that uniquely identifies the columns, independent of a natural language. This enable that the column titles are free text descriptions that can vary per language or user preference.

69	54	71	16	101	60	2
UID of the language of the term	**Name of the language**	**UID of the language community**	**Name of the language community**	**Term (name)**	**UID/Name of kind of relation**	**UID of named thing**
910036	English	193263	engineering	pump	5117/is a name of	130206
910037	Dutch	193263	engineering	pomp	5117/is a name of	130206
910038	German	193263	engineering	Pumpe	5117/is a name of	130206
910036	English	190668	linguistics	German	5117/is a name of	910038
910038	German	190668	linguistics	Deutsch	5117/is a name of	910038
910037	Dutch	190668	linguistics	Duits	5117/is a name of	910038
910036	English	193259	ontology	assembly relation	5117/is a name of	1190
910036	English	492015	Gellish	is a part of	981/is a synonym of	1190
910036	English	492015	Gellish	has as part	1986/is an inverse of	1190
910036	English	492015	Gellish alternative	is a whole of	1986/is an inverse of	1190

Table 57, Naming table with UIDs and names of a concept in various languages

The use of a Naming table is illustrated in Table 57 on three examples:

1. The concept represented by UID 130206 is denoted in English as pump, in German as Pumpe and in Dutch as pomp. The language community where these names originate is 'engineering'. Table 57 illustrates how those various names in those three languages are allocated to the concept that is denoted by UID 130206.
2. Table 57 also illustrates an example of how names of languages differ in various languages. For example, the name of the German language, expressed in German is Deutsch and in Dutch it is Duits. Table 57 illustrates how the various names of the German language are related to the concept that is denoted by UID 910038.
3. The third example gives the names a kind of relation, its Formal English phrase as a synonym, its inverse Formal English phrase

and an alternative for the Formal English phrase.

Table 57 should be interpreted as follows:

- The table has two header rows. The numbers in the first row, 69, 54, 71, 16, 101, 60 and 2, are standardized natural language independent identifiers of the table columns. They refer to standard columns in Expression tables as is described later in this document. The texts on the second line are not-standardized names of those columns.
- The second and the fourth columns (54 and 16, in red) are added for clarification, but are semantically superfluous and are not part of a standard Naming table.
- The UID of the language in the first column (69) specifies the language in which the term in column (101) is expressed. Thus the number 910036 on the first row, which is the Gellish UID of the English language, specifies that the term 'pump' is an English term for concept 130206. Similarly, UID 910037 denotes the Dutch language and UID 910038 denotes the German language.
  Note that the fourth line specifies that the term 'English' is the English name of the language that is represented by the UID 910036.
- Column 60 denotes a UID of the kind of relation. In order to facilitate the readability of the example table the name of that kind of relation is given in addition, although that name is superfluous and does not belong to a Naming table. Note that the UID of the kind of relation could also indicate other kinds of relations, such as 'is an abbreviated name of' or 'is a code for' and some other variations. If the UID is 1986, then the 'name' consists of a phrase that denotes that in a relation the left and right terms are switched to express the same idea as when other phrases are used.
- The columns 69, 71 and 101 together form a unique key for the table.

Not only all dictionary concepts, but also each user-defined concept or individual thing (user defined object[11]) that is used in Formal English expressions of main ideas shall have a Gellish UID. Each user defined object UID shall be unique and shall be allocated conform the rules described in par. 5.1.1.

13.4.3 Prime contextual facts

Each main idea is accompanied by a number of prime and secondary contextual facts. Together that collection of facts is called the *expression context*, which is a set of kinds of contextual facts. Each of the contextual facts (which are specified below) is expressed as a binary relation that relates a pair of objects and a classification of that relation. The classification relation and the classifying kind of relation that classifies the relation may remain implicit in implementations (for example in a tabular implementation where they are defined by the definitions of the columns and the relations between the columns that make up an expression). However it depends on the kind of implementations whether the contextual relations can be interpreted from these relations and thus whether they should be made explicit in order to enable semantic interpretation. The latter is for example the case in RDF implementations.

The objects that are specified in the following table imply relations that express prime contextual facts. Definitions of these contextual facts as well as those in the next table are given in the following paragraphs.

Component ID (column ID)	Description of object
44	A pair of left hand object cardinalities.
45	A pair of right hand object cardinalities.
30	A UID of an extent to which a main idea is the case
32	A UID of a probability of the main idea
34	A UID of a location where the main idea is valid
76	A UID of the accuracy of a quantification.

11 In this document the unqualified term ‘object’ is used as synonym for the term ‘anything’.

70	A UID of a pick list for the qualification of aspects.
19	A UID of the validity context for an idea.
65	A partial definition in natural language of a concept or individual thing.
4	A full definition in natural language of a concept or individual thing.
42	A textual description of a main idea.
14	Remarks on the expression of a main idea.
8	Approval status of the expression of a main idea.

Table 58, Prime contextual facts

The definitions of these components of an expression are given in the following paragraphs.

13.4.4 Secondary contextual facts

The secondary contextual facts are facts that may contribute to the semantic interpretation of the ideas, but are mainly added for administrative reasons. They include the facts in the following table.

Component ID (column ID)	Description of object
24	Reason for latest change of status.
67	UID of the successor of the idea, in case it has the status 'replaced'.
12	UID of creator of idea.
9	Date-time of start of validity of the idea.
23	Date-time of start of availability of the expression.
22	Date-time of recording of expression. (optional)
10	Date-time of latest change of the expression.
6	UID of author of latest change of the expression.
78	UID of addressee of the expression.
13	References.
53	UID of the expression of the idea. (Line UID)
50	UID of a collection of ideas to which the idea belongs
0	A sequence in which the expressions are presented. (Presentation sequence)

Table 59, Secondary contextual facts

Uniqueness constraints are implementation constraints that intent to prevent that a database contains identical expressions in which also the contextual facts are identical. It depends on the scope of a database which expressions including context are considered to be identical. For example, in an extreme situation two identical expressions about the same idea, thus semantically having the same meaning, but expressed by different persons (originators), may be considered to be two different expressions in one context, whereas they are considered to be the same expression in another context. This means that it might be required to add the originator to the uniqueness constraint. Similarly, when a requirement is stated to be valid in multiple validity contexts, then this means that there are multiple requirements, each with its own 'idea UID'. This implies that the 'validity context UID' should be added to the second uniqueness constraints.

13.5 Naming relations for objects in expressions of ideas

In principle, every UID that is used in an expression of a main idea, or in an expression of a contextual fact, is denoted in a human readable expression by a term (name, etc.), or by more than one term in case of synonyms. The terminology is recorded in naming relations between UIDs and terms, as described in paragraph 5.1 and 13.4.1.

In Formal English Databases all the naming relations of UIDs can be recorded in a separate Naming table. However, it is also possible that they are included in an integrated Expression table (see par. 15). In an integrated Expression table the UIDs as well as the terms are included in the table itself.

Table 60 specifies all the names that imply naming relations (expressions of additional contextual facts) that are required to allocate names (terms) to the UIDs that are used to express main ideas and contextual facts.' Note that 'name' stands for a character string that can be a term, a code, a phrase, a number, a URI, etc.

Component ID (column ID)	Description of object
101	The name of a left hand object.
201	The name of a right hand object.
3	The name of a kind of relation.
7	The name of a scale (UoM).
54	The name of a language.
16	The name of a language community.
73	The name of a left hand role.
75	The name of a right hand role.
43	The name of an intention.
31	The name of an extent.
33	The name of a probability.
35	The name of a location.
12	The name of an author of latest change.
77	The name of an accuracy of quantification.
20	The name of a pick list.
68	The name of a collection of ideas.
79	The name of an addressee of the expression.

Table 60, Naming columns in an Expression table

14 Subsets of expression components & context

A formal English message or database may consist of the full set of expression component and contextual facts as defined in this document. It may also consist of a subset of them.

The definition of these subsets implicitly also define subset Expression tables.

Depending on the application, users may decide to use a flexible subset or one of the predefined standard subsets of the collection of contextual facts.

The following subsets of expressions are defined, each with its equivalent subset Expression table:

- Subset Minimum subset
- Subset Flexible subset
- Subset Nomenclature
- Subset Dictionary
- Subset Taxonomy
- Subset Product Model
- Subset Business Model (recommended)
- Query tables

These standard subsets are defined in the following paragraphs.

The subsets require the presence of all elements that are specified for the chosen subset and the elements shall be arranged in the indicated sequence, with as only exception the Flexible subset.

The default subset is the *Business Model*.

14.1 Subset: Minimum subset

A *Minimum subset* is intended to express messages in nearly natural language, while still using a Gellish formalized language dictionary and the standard Gellish formalized language kinds of relations.

A Minimum subset has limited expression capabilities and therefore only suitable for usage in simple applications in small communities. For example, the subset does not provide a mechanism for the explicit distinction between homonyms nor for the explicit distinction between languages. It is neither suitable to express intentions such as questions, denials, but is only intended to express statements. It does not provide for contextual facts, such as an approval status, source and timing information about the expressed ideas.

Users of Minimum subsets should ensure that the terms (names) of objects in the messages are unique or that the distinction between homonyms is apparent from the context in which the terms are used and that synonyms are explicitly declared to be synonyms.

A Minimum subset consists of only the core of an expression of a main idea, expressed in formalized natural language terms. Such a minimum subset consists of the following three expression components:

Component ID (column ID)	Description
3	A name of a kind of relation (= formalized language phrase)
101	A name of a left hand object
201	A name of a right hand object

Table 61, Minimum subset

Minimum subsets may be expressed (implemented) in various ways (syntactic structures or formats). For example in the form of functions, such as:

- Relation type (left hand object, right hand object)

Another implementation may be in the form of a Minimum subset Expression table, which contains only the three columns: 101, 3 and 201. An example of such an Expression table is:

101	3	201
Name of left hand object	**Name of kind of relation**	**Name of right hand object**
the Eiffel tower	is located in	Paris

Table 62, Minimum subset Expression table

Minimum subset expressions are triples of expression components that are directly compliant with the Notation 3 RDF (N3) form of the RDF standard of the World Wide Web Consortium (W3C).

> Note: A more elaborate Expression table with additional columns can also be represented as collections of triples and can also be expressed in RDF or Notation 3 RDF as is described in the last chapter.

Minimum subset+ is a in extension of Minimum subset with the idea UID, which enables the management of ideas.

14.2 Subset: Flexible subset

A *Flexible subset* is a subset that contains at least the non-optional expression components. The non-optional components are: 2, 101, 1, 60, 3, 15, 201, 8, 9 and 10 as described in Table 63.

Component ID (column ID)	Description
2	UID of left hand object
101	Name of left hand object
1	UID of main idea
60	UID of kind of relation
3	Name of kind of relation
15	UID of right hand object
201	Name of right hand object
8	Approval status
9	Date-Time of start of validity
10	Date-time of latest change
etc	Free choice of additional columns (in any sequence)

Table 63, Minimum expression components for flexible subset

Note: Expressions that consists of more than three expression components can be represented as collections of triples. For example, when they are expressed in RDF or Notation 3 RDF extended with an indicator for the collections (such as in TRIX). Such a format is described in ISO 15926-11.

The selection of additional optional columns as well as the sequence of the columns is free. The sequence of the columns in an Expression table is semantically irrelevant, because the columns shall be uniquely identified by their column identifiers and the relations between the columns are defined independent of their position in the table.

A Flexible subset may even include non-standard additional columns, which columns are then treated as comment from a formalized language perspective.

14.3 Subset: Nomenclature, Lexicon or Vocabulary

A *Nomenclature subset*, *Lexicon subset* or *Vocabulary subset* (Nomenclature for short) is intended to specify terminology. A specification of terminology implies names, synonyms, codes, abbreviations, translations, etc. that are used to denote something that is represented by a UID.

A Nomenclature subset represents a list of particular terms as 'names' of things and their unique identifier, together with the language in which the names are expressed and the language community in which the term for the thing originates.

A Nomenclature list typically includes names of concepts, but may also include names of individual things such as countries and other standard geographical objects. Organizations or projects will often maintain the nomenclature of individual things or collections of individual things. For example as represented in equipment lists, line lists, inventories, etc.

A Nomenclature subset includes contextual facts as well. For example the approval status and date-time values, sources, etc. A Nomenclature subset consists of the following expression components in the indicated sequence:

0, 69, 54, 71, 16, 2, 101, 1, 8, 67, 9, 10, 12 and 13. These expression components are given in Table 64.

Component ID (column ID)	Description
0	Presentation key
69	UID of natural language
54	Name of natural language
71	UID of language community
16	Name of language community
2	UID of left hand object
101	Name of left hand object
1	UID of main idea
8	Approval status
67	UID of succeeding idea
9	Date-Time of start of validity
10	Date-time of latest change
12	Name of author of latest change
13	UID of creator of idea

Table 64, Expression components for a vocabulary

A collection of such expression components require a syntactical structure to define the relations between the components. For example, a tabular implementation implicitly defines as contextual fact a naming relation between the UID and a term (name of thing) in the vocabulary. This relation is of the type ‘is called’ (or ‘is referenced as’). For example:

130206 is called pump.

Such a table also expresses a contextual fact that defines the language context in which the naming is done. This idea is of the type <is presented in> (English).

The Nomenclature subset also allows defining the language community (sub-culture) where a name originates (component 71 and 16). For example, the name ‘pump’ may be declared to originate in the ‘mechanical engineering’ domain.

Misspellings and a pointer to the correct spelling can also be recorded in the nomenclature table. Misspellings can be indicated by a status (column 8) 'replaced' as well as an 'identifier of successor of main idea' (column 67), which refers to the idea UID that defines the correct spelling.

Preferred terms are terms which use is preferred in a particular language community. When an organization wants to specify its own list of preferred terms it might specify them within their own language community, even specifying terms that are identical to terms that are already specified for another language community.

When a Nomenclature (or Lexicon or Vocabulary) is represented in tabular form it can be represented in a Nomenclature subset of an Expression table. Table 65 is an example of the main columns in a Nomenclature table.

54	16	2	101	1	8	67
Language	**Language community (discipline)**	**Gellish UID**	**Name of thing**	**UID of idea**	**Status**	**UID of successor of main idea**
English	mechanical engineering	130206	pump	201	accepted	
Deutsch	Maschinenbau	130206	Pumpe	202	proposed	
Nederlands	werktuigbouwkunde	130206	pompe	203	replaced	204
Nederlands	werktuigbouwkunde	130206	pomp	204	accepted	

Table 65, Nomenclature subset example

Table 65 illustrates that the same concept, represented in the formalized language by UID 130206 is denoted in English as 'pump' and in other languages by different terms, whereas the spelling 'pompe' in Dutch is a misspelling that should be replaced by 'pomp'.

14.4 Subset: Dictionary

A *Dictionary subset* is intended to provide textual definitions of things, especially of concepts, as an addition to the Nomenclature and Taxonomy subsets. This implies a relation between the thing and the text that defines the thing.

The following is an example of the core columns in a Dictionary subset of an Expression table.

54	2	101	1	4	8
Language	**UID of defined thing**	**Name of thing**	**UID of idea**	**Textual definition**	**Status**
English	130206	pump	205	is a rotating equipment item intended to increase pressure in a liquid.	accepted
Nederlands	130206	pomp	206	is een apparaat met roterende delen dat bedoeld is om de druk in een vloeistof te verhogen.	accepted

Table 66, Dictionary subset core example.

A full *Dictionary subset* consists of a Vocabulary subset (Table 64) plus two additional components: the full definition and an option for adding remarks.

Component ID (column ID)	Description
4	Full definition (natural language text)
14	Remarks

Thus a Dictionary subset comprises the following components in the indicated sequence:
0, 69, 54, 71, 16, 2, 101, 1, **4**, **14**, 8, 67, 9, 10, 12 and 13.

Note 1: It is possible to record definitions for the same concept in multiple languages.

Note 2: *Definition models* are definitions that are expressed as collections of relations between concepts. Those relations require at least a Product Model subset.

Note 3: *Verbal (spoken) or pictorial definitions* require a relation to a sound or picture (or combination of them). However the textual definition (column 4) is meant for a string in ASCII or Unicode only. Therefore, such other definitions require at least a ‘Product model’ subset, as described below.

14.5 Subset: Taxonomy

A *Taxonomy subset* is a specialization hierarchy of concepts, also called a subtyping hierarchy (sometimes erroneously called a classification hierarchy). This implies that there are subtype-supertype relations between the concepts. A subtype concept is a specialization of a supertype concept. The inverse of that relation expresses the same idea in another way, namely that a supertype concept is a generalization of a subtype concept.

Table 67 illustrates the core columns in a Taxonomy table.

54	2	101	1	15	15	8
Language	**UID of left hand object**	**Name of left hand object**	**UID of idea**	**UID of right hand object**	**Name of right hand object**	**Status**
English	130206	pump	7	130227	rotating equipment item	accepted
Nederlands	130206	pomp	7	130227	apparaat met roterende delen	ignore duplicate

Table 67, Taxonomy subset example

A specialization relation implies that the subtype concept inherits all the aspects that are intrinsic to the supertype concept.

Note that the left hand object name and the right hand object name, as well as the language, are strictly speaking superfluous, but they are added to support the readability of the expressions. If they are ignored it becomes clear that the two lines in the above example define the same idea, which is the reason why the UIDs of the ideas are identical and the status of the latter one is set at ‘duplicate’.

A *Taxonomy subset* is an extension of a Dictionary subset by including expression components for the UIDs and names of supertype concepts.

Component ID (column ID)	Description
15	UID of right hand object

201	Name of right hand object

Thus a Taxonomy subset consists of the following expression components in the indicated sequence:
0, 69, 54, 71, 16, 2, 101, 1, **15**, **201**, 14, 8, 67, 9, 10, 12 and 13.

14.6 Subset: Product Model

A *Product Model subset* is intended for use in practice of data exchange to describe individual objects (including occurrences) during their lifecycle as well as knowledge about kinds of things.

A Product Model subset consists of the following expression components in the indicated sequence:
0, 69, 54, 71, 16, **44**, 2, 101, 1, **60, 3**, **45**, 15, 201, **65, 4, 30, 31, 66, 7**, 14, 8, 67, 9, 10, 12, 13, **50** and **68**.

The expression components are presented in Table 68.

Component ID (column ID)	**Description**
0	Presentation key
69	UID of natural language
54	Name of natural language
71	UID of language community
16	Name of language community
44	Left hand object cardinalities
2	UID of left hand object
101	Name of left hand object
1	UID of main idea
60	UID of kind of relation
3	Name of kind of relation
45	Right hand object cardinalities
15	UID of right hand object
201	Name of right hand object
65	Partial definition
4	Full definition
30	UID of extent
31	Name of extent

66	UID of unit of measure
7	Name (symbol) of unit of measure (UoM)
14	Remarks
8	Approval status
67	UID of succeeding idea
9	Date-Time of start of validity
10	Date-time of latest change
12	Name of author of latest change
13	UID of creator of idea
50	UID of collection of ideas
68	Name of collection of ideas

Table 68, Expression components of a Product Model subset

Definitions of the components and implied relations are given in the Gellish Syntax document.

14.7 Subset: Business Model

A *Business Model subset* is intended for use in practice of data exchange to describe propositions. This includes business communication about both designs (imaginary objects) as well as real world objects (observed individual objects) during their lifecycle and about enquiries, answers, orders, confirmations, etc. This subset is a superset (indicated in **bold**) of the Product Model subset, so it can also be used for storage and exchange of knowledge about kinds of things.

A Business Model subset is a subset that consists of the following expression components in the indicated sequence:
0, 69, 54, 71, 16, **39**, 44, 2, 101, **72**, **73**, **5**, **43**, **19**, **18**, 1, **42**, 60, 3, **74**, **75**, 45, 15, 201, **34, 35,** 65, 4, 30, 31, **32, 33,** 66, 7, **76, 77, 34, 35, 70**, **20**, 14, 8, **24**, 67, 9, 10, **6**, 12, **78, 79,** 13, **53**, 50, 68.

The expression components in a Business Model are presented in Table 69.

Component ID (column ID)	Description
0	Presentation key
69	UID of natural language

54	Name of natural language
71	UID of language community
16	Name of language community
39	Reality
44	Left hand object cardinalities
2	UID of left hand object
101	Name of left hand object
72	UID of left hand kind of role
73	Name of left hand kind of role
4	Full definition
43	Name of intention
19	UID of validity context
18	Name of validity context
1	UID of main idea
42	Description of main idea
60	UID of kind of relation
3	Name of kind of relation
74	UID of right hand kind of role
75	Name of right hand kind of role
45	Right hand object cardinalities
15	UID of right hand object
201	Name of right hand object
34	UID of exponent
35	Name of exponent
65	Partial definition
4	Full definition
30	UID of extent
31	Name of extent
32	UID of probability
33	Name of probability
66	UID of unit of measure
7	Name (symbol) of unit of measure (UoM)
76	UID of accuracy of quantification
77	Name of accuracy of quantification
34	UID of validity location
35	Name of validity location
70	UID of pick list
20	Name of pick list
14	Remarks

8	Approval status
24	Reason
67	UID of succeeding idea
9	Date-Time of start of validity
10	Date-time of latest change
6	UID of author of latest change
12	Name of author of latest change
78	UID of addressee of expression
79	Name of addressee of expression
13	UID of creator of idea
53	UID of expression
50	UID of collection of ideas
68	Name of collection of ideas

Table 69, Expression components for a Business Model

The above-indicated sequences of expression components are defined as a handy sequence for human interpretation of a tabular content. There is no semantic meaning in that sequence, because the semantics of the relations between the components are defined explicitly.

14.8 Query subsets

A *Query subset* consists of one of the other subsets, extended with expression components for the specification of string commonality criteria.

In a tabular form a Query subset is a subset that is extended with the expression components 80 and 81.

Component ID (column ID)	Description
80	Left hand string commonality
81	Right hand string commonality

15 Implementation in the Gellish Syntax

All semantic expressions, of any 'arity', can be expressed in various syntaxes. For example in RDF triples, although such triples should be extended with a method to recognize collections of triples, which are usually called 'graphs', to represent the idea's, terms as well as contextual facts. This can be done for example by using TRIX as is specified in ISO 15926-11. A more direct powerful and efficient implementation uses the Gellish Syntax, which defines the tabular Gellish Expression format. Such a table can be used to describe any ideas as well as queries about individual things or occurrences, requirements for things or knowledge about things in general. The various standard kinds of relations that are used to classify the relations and the indication of the intentions and the contextual facts determine the categories of the expressions.

Typically a statement or question about an individual thing is modeled by a relation that is classified by a kind of relation that is denoted by a phrase that starts with "is" or "has". A requirement phrase starts with "shall" and must specify a validity context. A statement that expresses knowledge about possibilities typically uses a kind of relation that is denoted by a phrase that starts with "can have" or "can be". This is illustrated in the following table, which is a subset of Gellish expression table.

101	18	1	3	45	201
Left hand object name	**Validity context for main idea**	**UID of idea**	**Relation type name**	**Cardinalities**	**Right hand object name**
I-1		201	is a part of		P-1
impeller	handover to operations	202	shall have as aspect a		diameter
centrifugal pump		203	can have as part a	1,n	pump impeller
impeller		204	has by definition as part a	2,n	vane

Table 70, Example of Product data, a Requirement and Knowledge in one Expression table

The example in Table 70 illustrates four kinds of statements. The first one states that a particular impeller is a part of a particular pump. The second one states that information about any (model of an) impeller that is handed over to operations shall include a diameter. The third statement describes the general knowledge that any centrifugal pump can have (and at least has) one impeller. The minimum and maximum number of simultaneous instances (individual impellers for individual pumps) is indicated by the cardinalities. The last expression states that an impeller has by definition 2 or more vanes. Table 70 demonstrates that all such kinds of statements can be expressed in the same table or in tables that have the same columns and have a single common definition.

A full specification of the Gellish Syntax is presented in the separate document 'Gellish Syntax and Contextual Facts (Ref. 4).

16 References

1. Andries van Renssen, Formalized Natural Languages, Definition and Application of a Universal Information Modeling Language, Lulu 2014, (2nd edition of Gellish, a Generic Extensible Ontological Language, Delft University Press, 2005).
2. Andries van Renssen, Semantic Modeling Methodology, Lulu 2015. http://www.gellish.net/downloads
3. Andries van Renssen, Taxonomic Dictionary of Relations, Lulu 2015. http://www.gellish.net/downloads
4. Andries van Renssen, Gellish Syntax and Contextual facts. http://www.gellish.net/downloads
5. John R. Searle, Speech Acts. An assay in the Philosophy of Language, Cambridge University Press, 1969.
6. Bertrand Russell, On Denoting, Mind 14, pp. 479-493.
7. Noam Chomsky, Syntactic Structures, 12th ed., Mouton, The Hague-Paris, 1976.
8. Gellish Formalized English Taxonomic Dictionary and Ontology, http://www.gellish.net/downloads.