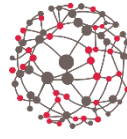




UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



**VeraTech**  
FOR HEALTH

Doctoral Program in Technologies for Health and Well-being

# Enrichment of Archetypes with Domain Knowledge to Enhance the Consistency of Electronic Health Records

Doctoral Thesis

## Author

Vicente Miguel Giménez Solano

## Supervisors

José Alberto Maldonado Segura

Montserrat Robles Viejo

Vicente Traver Salcedo

Valencia, October 2021



## ACKNOWLEDGEMENTS

---

I would like to thank the following people for helping me, not only in the development of this thesis, but in my professional career. Please, accept my gratitude.

First of all, I would like to express my sincere gratitude to Montserrat Robles and José Alberto Maldonado. They gave me the opportunity to enter in the exciting world of medical informatics seven years ago with a specialization grant that later became a pre-doctoral contract. I started with them, and they have always been by my side.

I also want to thank professor Vicente Traver, who later joined as co-director of my thesis and has helped me with good ideas to be able to finish it in a timely manner.

I want to express my gratitude to Kirstine Rosenbeck, who helped me to develop an important part of my thesis, and for giving me the opportunity to work for a few months in such a beautiful city as Aalborg.

I would like to thank professors Mar Marcos and Begoña Martínez for gave me the opportunity to work at *Universitat Jaume I* for a year, and for making me feel so comfortable working with them.

I also want to thank professor Jesualdo Fernández of *Universidad de Murcia* for allowing me to work in his research group, and for having allowed me to meet Francisco Abad, a great researcher and person.

Of course, I would like to thank Dr. Arturo Romero, CEO in Veratech for Health, and all my colleagues, for being the best people and professionals anyone would want to work with: Maryna, Cristina, Diego, David, Santi, Pablo, and ex-Veratech's Christian.

Finally, I would like to thank my family, who have always supported and trusted in me, and to whom I dedicate this thesis.

This thesis was partially funded by *Ministerio de Economía y Competitividad*, "Doctorados Industriales", grant DIN2018-009951, and by *Universitat Politècnica de València*, "Formación de Personal Investigador" (FPI-UPV).



# CONTENTS

---

Acknowledgements .....	3
Contents .....	5
List of figures .....	9
List of tables .....	13
Abbreviations and acronyms .....	15
Glossary .....	17
Abstract .....	21
Resumen.....	23
Resum.....	25
Chapter 1. Introduction .....	27
1.1    Motivation .....	27
1.2    Hypothesis, research questions and objectives.....	28
1.2.1    Hypothesis.....	28
1.2.2    Specific research questions and objectives .....	28
1.3    Contributions.....	30
1.3.1    Main contributions .....	30
1.3.2    Journal publications and conference contributions .....	33
1.3.3    Projects.....	35
1.3.4    Software .....	36
1.3.5    Grants .....	37
1.3.6    Research visits .....	37
1.3.7    Other contributions.....	38
1.4    Structure of the thesis .....	39
Chapter 2. Background .....	41
2.1    Introduction.....	41
2.2    Objectives .....	44
2.3    Consistency of EHR .....	45
2.4    Archetype-based EHR architectures .....	46
2.5    Terminology binding.....	49

2.6	SNOMED CT .....	55
2.6.1	Introduction.....	55
2.6.2	Logical model.....	57
2.6.3	Concept model .....	64
2.6.4	Expressions .....	65
2.7	SNOMED CT Expression Constraint Language .....	67
2.7.1	Simple ECs .....	71
2.7.2	Refined ECs.....	72
2.7.3	Compound ECs .....	75
2.7.4	Graphical representation of ECs.....	75
2.8	openEHR Expression Language.....	79
Chapter 3. Methods for the definition, validation, execution and visualization of SNOMED CT subsets using ECL.....		81
3.1	Introduction.....	81
3.2	Storage of the SNOMED CT database.....	82
3.3	EC simplification .....	85
3.3.1	Subsumption based .....	86
3.3.2	MRCM-based.....	88
3.3.3	Logic definition-based .....	90
3.3.4	Post-execution simplification .....	94
3.4	EC execution.....	97
3.5	Subset visualization .....	101
Chapter 4. SNQuery: ECL execution engine based on graph databases .....		107
4.1	Overview of SNQuery .....	107
4.2	Other functionalities.....	111
4.2.1	MSSSI refsets list.....	111
4.2.2	Multilingual interface .....	112
4.2.3	Mapping to ICD-10 .....	112
4.2.4	Conversion from brief to long syntax and vice versa .....	112
4.3	Evaluation.....	113
4.4	Discussion.....	118
Chapter 5. Consistency rules in archetypes.....		125

5.1	Introduction.....	125
5.2	Requirements for the consistency language in archetypes .....	127
5.2.1	Introduction.....	127
5.2.2	Constraints .....	129
5.2.3	Rules .....	129
5.2.4	Exists operator.....	131
5.2.5	For_all operator.....	131
5.2.6	Inclusion operator .....	132
5.2.7	Considerations about the use of 'EXISTS', 'FOR_ALL' and 'IN' .....	133
5.2.8	Considerations about paths.....	134
5.2.9	Considerations about nesting rules .....	134
5.2.10	Data types, operators and functions .....	135
5.3	Review of existing languages .....	136
5.3.1	Comparison between GDL and EL .....	141
5.4	Justification of choosing EL as basis.....	149
5.5	Extensions to EL.....	151
Chapter 6. The EHRules Language .....		153
6.1	Overview.....	153
6.2	Execution model.....	154
6.3	Language specification .....	154
6.3.1	General structure .....	154
6.3.2	Syntax style.....	155
6.3.3	Typing .....	155
6.3.4	Variable declaration and assignation.....	157
6.3.5	Expressions .....	159
6.3.6	Paths, contexts and conditions.....	160
6.3.7	Functions .....	161
6.3.8	Operators .....	165
6.3.9	Rules .....	166
6.3.10	Value set bindings.....	169
6.4	Uses cases .....	174

6.4.1	Guidelines for acute stroke care.....	174
6.4.2	FSIII Danish standard .....	185
6.4.3	Requisition for radiology procedures .....	189
6.5	Validation of EHR data.....	192
6.6	Discussion.....	202
Chapter 7. Conclusions and future work .....		205
7.1	Conclusions.....	205
7.2	Future work .....	207
Bibliography.....		209
Annex 1 - List of ECs examples .....		219
Annex 2 - EHRules ABNF syntax specification.....		223
Annex 3 - FSIII Condition-Interventions .....		253
Annex 4 - From EHRules to Schematron examples.....		261



## LIST OF FIGURES

---

Figure 1. A SNOMED CT subset visualized using the Neo4j browser	41
Figure 2. Archetypes status in the CKM repository	42
Figure 3. Consistency rules help solving consistency errors in EHR	43
Figure 4. Reference model of the ISO13606 standard	47
Figure 5. Example of a combination of the reference model classes of the ISO13606	48
Figure 6. Example of a patient summary archetype	49
Figure 7. The three fundamental pillars of semantic interoperability	50
Figure 8. A draw showing terminology bindings between elements of an EHR model and SNOMED CT concepts	51
Figure 9. Example of semantic binding	52
Figure 10. Example of Simple value set binding	53
Figure 11. Example of Conditional value set binding	53
Figure 12. Example of Dependency value set binding	54
Figure 13. Example of Compositional value set binding	54
Figure 14. Local extensions of SNOMED CT	55
Figure 15. Member map of SNOMED International	56
Figure 16. Overview of the SNOMED CT logical model	57
Figure 17. Examples of descriptions for the concept 22298006  Myocardial infarction	58
Figure 18. Example of IS A and attribute relationships	59
Figure 19. Example of multi-parent IS A relationships <sup>1</sup>	59
Figure 20. Logical definition of the concept 425548001  Abscess of heart	60
Figure 21. Three types of graphs	60
Figure 22. Abstract representation of some SNOMED CT top-level hierarchies	61
Figure 23. Visualization of the relationships between clinical concepts in SNOMED CT	61
Figure 24. SNOMED CT overview scheme with examples	62
Figure 25. Logical definition of the concept 2704003  Acute disease	63
Figure 26. Logical definition of the concept 64572001  Disease	63
Figure 27. Logical definition of the concept 69449002  Drug action	63
Figure 28. Examples of the domain and range specified for the attributes 363698007  Finding site  and 425391005  Using access device (attribute)	64

Figure 29. Examples of attribute relationships according to the concept model	65
Figure 30. Logic definition of the concept 174041007  Laparoscopic emergency appendectomy	66
Figure 31. SNOMED CT concept model, expressions and ECs summary	68
Figure 32. Comparison between the SNOMED CT Compositional Grammar and ECL	68
Figure 33. Abstract model of a SNOMED CT EC	70
Figure 34. SNOMED CT EC components <sup>16</sup>	71
Figure 35. Simple EC showing those concepts that are a type of diabetes mellitus	76
Figure 36. Simple EC showing those concepts that are a type of diabetes mellitus and diabetes mellitus itself	76
Figure 37. Simple EC showing those concepts that are a supertype of diabetes mellitus	76
Figure 38. Simple EC showing those concepts that are a supertype of diabetes mellitus and diabetes mellitus itself	76
Figure 39. Refined EC showing those clinical findings caused by organisms	77
Figure 40. Refined EC showing those clinical findings caused by 2 or 3 organisms	77
Figure 41. Refined EC showing those substances that causes clinical findings caused by 3 substances	78
Figure 42. Compound EC showing those lung disorders with any type of edema of trunk	78
Figure 43. Properties of the concepts “444827008  Erythema of skin (finding) ” and “39937001  Skin structure (body structure) ”, relationship “363698007  Finding site (attribute) ” between them, and relationships with ancestors (SNOMED CT July 2020 International)	85
Figure 44. Syntax tree of EC example 7	87
Figure 45. Syntax tree of EC example 2, which is the simplification of EC example 7	88
Figure 46. Syntax tree of EC example 9	89
Figure 47. Syntax tree of the simplification of EC example 9	90
Figure 48. Syntax tree of EC example 10	92
Figure 49. Logic definition of 106063007  Cardiovascular finding (finding)	92
Figure 50. Logic definitions of the concept “106063007  Cardiovascular finding (finding) ” and some of its descendants. Only “363698007  Finding site ” attribute relationships are represented	93
Figure 51. Syntax tree of the simplification of EC example 10	93

Figure 52. Syntax tree of EC example 11	95
Figure 53. Syntax tree of EC example 13, which is the simplification of EC example 11	97
Figure 54. Circle packing visualization of EC example 2 (using the SNOMED CT July 2020 International Edition). First level of visualization is displayed in (a); (b) shows the most relevant sub-hierarchy (in orange, pointed out by an arrow); (c) shows the result of zooming	105
Figure 55. SNQuery showing some disorders of lung whose associated morphology is any type of edema	108
Figure 56. Diabetes mellitus hierarchy	108
Figure 57. Diabetes mellitus hierarchy after applying the ForceAtlas2 Graph Layout Algorithm [41]	109
Figure 58. Tree-based visualization showing three levels in depth of those disorders of lung associated with edema (only intermediate concepts are shown)	109
Figure 59. Visual ECs authoring tool	110
Figure 60. Domains $D_1$ to $D_n$ over which the subset is calculated (note that $D_n$ is represented by all the SNOMED CT substrate)	116
Figure 61. Scatter plot showing the relation between domain size and execution time, and standard deviation error bars (in milliseconds) of ECs in Table 10	118
Figure 62. The PROforma task model	136
Figure 63. Selection tree of the analysed medical logic languages	150
Figure 64. IF-THEN (left), IF-THEN-ELSE (center), and nested IF-THEN-ELSE (right) rules diagrams	167
Figure 65. Original Danish radiology procedures requisition re-designed and translated into English	189
Figure 66. The definition of consistency rules in archetypes is used to validate the EHR data	192
Figure 67. The definition of consistency rules in archetypes is used to support structured data entry	193



## LIST OF TABLES

---

<i>Table 1. Structure of the thesis and relationship with the research questions, objectives, contributions, and publications</i>	40
<i>Table 2. Capabilities required by ECL</i>	70
<i>Table 3. ECL constraint operators</i>	72
<i>Table 4. ECL binary operators</i>	75
<i>Table 5. Patterns of ECs and its translation into Cypher Query Language</i>	100
<i>Table 6. Combination between EC patterns 14 and 17 translated into Cypher Query Language</i>	101
<i>Table 7. Features showed by the circle packing visualization</i>	103
<i>Table 8. Examples of ECs before and after applying the SNQuery simplification process</i>	114
<i>Table 9. Result subset size, domain size, execution time (in milliseconds), and standard deviation obtained before and after applying the simplification process of each EC</i>	115
<i>Table 10. Result subset size, domain size, execution time and standard deviation (in milliseconds) of EC11 to EC19 set of equivalent ECs. "REF" is equivalent to "363698007  Finding site (attribute)  = &lt; 127903009  Male genital organ structure (body structure) "</i>	117
<i>Table 11. Comparison between general features of Arden Syntax, GELLO, GDL and EL</i>	140
<i>Table 12. Comparison of features between GDL and EL</i>	143
<i>Table 13. EHRules primitive data types</i>	156
<i>Table 14. EHRules subset data type</i>	156
<i>Table 15. EHRules container data type</i>	157
<i>Table 16. Examples of declaration of Terminology_code variables and assignation of values</i>	158
<i>Table 17. Examples of declaration of Snomed_ec variables and assignation of values</i>	158
<i>Table 18. Built-in functions supported by EHRules</i>	164
<i>Table 19. Operators supported by EHRules</i>	166
<i>Table 20. IF-THEN, IF-THEN-ELSE and nested IF-THEN-ELSE rules syntax and examples</i>	169
<i>Table 21. Syntax and examples of simple, conditional, dependency and conditional plus dependency intensional value set bindings</i>	173
<i>Table 22. 19 thrombolysis contraindications</i>	176

<i>Table 23. Archetypes involved and contraindications by ID</i>	<i>177</i>
<i>Table 24. FSIII conditions together with its associated sets of interventions</i>	<i>186</i>
<i>Table 25. EHRrules variable declaration and assignation translated to Schematron</i>	<i>194</i>
<i>Table 26. EHRrules data types translated to Schematron</i>	<i>194</i>
<i>Table 27. EHRrules functions and operators translated to Schematron</i>	<i>195</i>
<i>Table 28. EHRrules quantifier expressions translated to Schematron</i>	<i>196</i>
<i>Table 29. EHRrules IF-THEN-ELSE rules translated to Schematron</i>	<i>196</i>
<i>Table 30. EHRrules boolean expressions translated to Schematron</i>	<i>196</i>
<i>Table 31. EHRrules simple value set binding translated to Schematron</i>	<i>197</i>
<i>Table 32. EHRrules conditional value set binding translated to Schematron</i>	<i>197</i>
<i>Table 33. EHRrules dependency value set binding translated to Schematron</i>	<i>197</i>
<i>Table 34. EHRrules conditional plus dependency value set binding translated to Schematron</i>	<i>198</i>
<i>Table 35. Translation of the 4th contraindication into Schematron rules</i>	<i>199</i>
<i>Table 36. Translation of the 5th contraindication into Schematron rules</i>	<i>200</i>
<i>Table 37. Translation of the 7th contraindication into Schematron rules</i>	<i>201</i>

## ABBREVIATIONS AND ACRONYMS

---

ABNF	Augmented Backus–Naur form
ADL	Archetype Definition Language
AML	Archetype Modelling Language
AOM	Archetype Object Model
AQL	Archetype Query Language
AM	Archetype Model
CDSS	Clinical Decision Support System
CM	Clinical Model. Detailed, reusable and domain-specific definition of a clinical concept.
CKM	Clinical Knowledge Manager
DCM	Detailed Clinical Models
EC	Expression Constraint
ECL	Expression Constraint Language
EL	openEHR Expression Language
EHR	Electronic Health Record
GDL	Guideline Definition Language
ICD	International Classification of Diseases
IHTSDO	International Health Terminology Standards Development Organization
IM	Information Model, a conceptual model of the information needed to support a business function or process
ISO	International Organization for Standardization
MRCM	Machine Readable Concept Model
OCL	Object Constraint Language
OWL	Web Ontology Language
RM	Reference Model
SNS	Spanish health system ( <i>Sistema Nacional de Salud</i> )
SNOMED CT	Systematized Nomenclature of Medicine – Clinical Terms
XML	Extensible Mark-up Language
XSD	XML Schema Definition





## GLOSSARY

---

**Archetype:** representation of clinical concepts in a formal way. They link to clinical terminology. They are a hierarchical combination of reference model structures constrained to fit the definition of the modelled concept.

**Clinical Information Model:** standard data structure for EHR storage and communication, such as openEHR, ISO 13606, or HL7 CDA.

**Consistency:** a dimension of quality. Health data are considered to be consistent when there is agreement or compatibility between the data elements.

**Consistency rule:** expression that can be evaluated to a truth value depending on whether it satisfies an offset of conditions. They are associated with the quality of the EHR data.

**Content (or value set) binding:** it is a type of terminological content binding where the associated set depends on a condition.

**Content (or value set) binding:** type of terminological binding where an element of the archetype is associated with a set of terminological concepts to define its possible content. That is, the element of the archetype can only contain as a value, one concept from that set. Otherwise, the archetype will be semantically inconsistent.

**Detailed clinical information model:** data structure built from a standardized reference model whose main purpose is the communication of EHR data. For example, archetypes, templates, or forms.

**Electronic Health Record (EHR):** set of electronic documents where the medical history of a patient throughout his life is recorded. Accordingly with the standard CEN/ISO 13940, a health record is a data repository regarding the health and healthcare of a subject of care. EHR systems currently in use merely records the resulting information from the interaction

among a patient and healthcare professionals. The health record should carry out several functionalities: to act as a reminder and help in the clinical management of the patient; to facilitate the communication among the different components of the clinical team and the continuity of care; to represent the care provided, enabling the retrospective analyses of clinical practice.

**Extensional subset:** subset defined by a list of concepts. For example, type I diabetes, type II diabetes, etc.

**ICD:** the International Classification of Diseases (ICD) is a classification of diseases, symptoms, abnormal findings, etc. It is the most widely used disease coding system in EHR systems in the Spanish *Sistema Nacional de Salud* (SNS).

**Intensional (or comprehension) subset:** subset defined by an expression. For example, 'all types of diabetes'.

**Interoperability levels:** *Level 0:* no interoperability at all. *Level 1:* technical and syntactical interoperability (no semantic interoperability). Systems in this level are capable to communicate data among them, but not its meaning. *Level 2:* Partial semantic interoperability. Systems in this level are capable to understand the meaning of some of the communicated data. *Level 2a:* unidirectional semantic interoperability. *Level 2b:* bidirectional semantic interoperability of meaningful fragments. *Level 3:* full semantic interoperability, sharable context, seamless co-operability between systems.

**Medical ontology:** formal description of the concepts and relationships in the biomedical domain, where the meanings and hierarchical relationships between the terms and concepts of such domain are specified.

**Organizational interoperability:** based on business rules. It deals with the possible cooperation between different organizations under the same context and workflows.

**Post-coordinated concept:** unlike pre-coordinated, post-coordinated concepts are concepts that do not exist in terminology. They are created according to the particular requirements following the rules defined in the conceptual model so that they are semantically correct.

**Pre-coordinated concept:** concept defined by its attribute and hierarchical relationships, that is, by its logical definition. Pre-coordinated concepts are part of the terminology and do not need to be created.

**Semantic binding:** one of the two types of terminological binding in which an element of the archetype is associated with a concept of the terminology to define the meaning of the element unequivocally.

**Semantic interoperability:** ability of two or more computer systems to understand the meaning of the information that is transmitted between them, automatically and without human intervention.

**SNOMED CT:** Systematized Nomenclature of Medicine - Clinical Terms is the most comprehensive, multilingual and codified clinical terminology developed in the world. It is structured as a directed and acyclic graph. It is therefore a medical ontology.

**SNOMED CT Concept Model:** set of rules that govern how SNOMED CT concepts can be related through their attributes. Both post-coordinated expressions and expression constraints must be defined following their rules to be semantically correct. It is made up of domains, attributes, and ranges.

**SNOMED CT expressions:** there are two types of expressions, pre-coordinated, made up of a single identifier, and post-coordinated, made up of more than one identifier. At the syntactic level, post-coordinated expressions are built based on the compositional grammar of SNOMED CT and, at the semantic level, following the conceptual model.

**SNOMED CT Expression Constraints:** computable rules (in other words, evaluable / executable expressions) whose purpose is to define a subset of clinical concepts. To do this, it uses constraint operators to navigate through SNOMED CT hierarchies, refinements to filter by attributes, and logical operators, among others.

**SNOMED CT Expression Constraint Language:** the language used to define SNOMED CT expression constraints.

**SNOMED CT Machine Readable Concept Model (MRCM):** a version of the SNOMED CT Concept Model. It is expressed using the SNOMED CT Expression Constraint Language.

**Subset (of clinical concepts):** set of clinical concepts that generally belong to the same subdomain. For example, pulmonary diseases associated with edema, types of diabetes mellitus, clinical findings, radiological procedures on the heart, etc.

**Syntactic interoperability:** ability of two or more systems to transfer data and documents. It ensures that the structure of the documents is correct.

**Technical interoperability:** ability of two or more systems to communicate at the physical level (cabling, plugs, protocols, etc.). It deals with the transfer of bytes.

**Terminology binding:** association between an element of the clinical information model, such as an archetype, and a concept or set of concepts in a clinical terminology, such as SNOMED CT. Its main function is to define the meaning and content of the information model unequivocally in order to achieve a high degree of semantic interoperability.

**Terminological model or medical terminology:** set of terms that represent the concepts in the specific field of medicine.

## ABSTRACT

---

Consistency of EHR data, as a dimension of quality, is considered an essential requirement to the improvement of healthcare delivery, clinical decision-making processes, and the promotion of clinical research. In this context, cooperation between information and domain models has been considered essential in the literature, but it has not been adequately addressed by the scientific community to date.

The main contribution of this thesis is the development of methods and tools for the inclusion of terminology binding expressions in consistency rules. Specific contributions are:

- Definition of a method to execute ECs over a SNOMED CT graph-oriented database.
- Definition of methods to simplify ECs before and after its execution and semantic validation according to the SNOMED CT Machine Readable Concept Model (MRCM).
- Definition of a method to visualize, dynamically explore, understand and validate SNOMED CT subsets.
- Development of SNQuery, an execution platform that executes, simplifies and validates ECs, and visualizes the resulting subsets.
- Definition of EHRules, an expression language based on the openEHR Expression Language for the specification of consistency expressions in archetypes, including value set bindings, in order to enrich archetypes with domain knowledge.
- Definition of a method to execute EHRules expressions in order to validate the consistency of EHR data by executing such rules over patient data instances.

Our objective is that these contributions help to enhance the quality of EHR, as they provide methods and tools for the validation and enhancement of the EHR data consistency. We also intend, by defining value set bindings between information models and clinical terminologies, to raise the level of semantic interoperability, for which the definition of terminological bindings is crucial.



## RESUMEN

---

La consistencia de los datos de la HCE, como dimensión de la calidad, se considera un requisito esencial para la mejora de la prestación de la asistencia sanitaria, los procesos de toma de decisiones clínicas y la promoción de la investigación clínica. En este contexto, la cooperación entre la información y los modelos de dominio se considera esencial en la literatura, pero la comunidad científica no la ha abordado adecuadamente hasta la fecha.

La contribución principal de esta tesis es el desarrollo de métodos y herramientas para la inclusión de expresiones de enlaces terminológicos en reglas de consistencia. Las contribuciones específicas son:

- Definición de un método para ejecutar *ECs* sobre una base de datos de SNOMED CT orientada a grafos.
- Definición de métodos para simplificar *ECs* antes y después de su ejecución, y su validación semántica conforme al *Machine Readable Concept Model* de SNOMED CT (MRCM).
- Definición de un método para visualizar, explorar dinámicamente, comprender y validar subconjuntos de SNOMED CT.
- Desarrollo de SNQuery, una plataforma que ejecuta, simplifica y valida *ECs* y visualiza los subconjuntos resultantes.
- Definición de EHRules, un lenguaje de expresiones basado en el *openEHR Expression Language* para la especificación de reglas de consistencia en arquetipos, incluido el enlace terminológico de contenido, con el fin de enriquecer los arquetipos con conocimiento del dominio.
- Definición de un método para ejecutar las expresiones de EHRules con el fin de validar la consistencia de los datos de la HCE mediante la ejecución de dichas expresiones sobre instancias de datos de pacientes.

Nuestro objetivo es que estas contribuciones ayuden a mejorar la calidad de la HCE, ya que proporcionan métodos y herramientas para la validación y mejora de la consistencia de los datos de la HCE. Pretendemos, además, mediante la definición de enlaces de contenido entre modelos de información y terminologías clínicas, elevar el nivel de interoperabilidad semántica, para lo cual la definición de enlaces terminológicos es crucial.





## RESUM

---

La consistència de les dades de la HCE, com a dimensió de la qualitat, es considera un requisit essencial per a la millora de la prestació de l'assistència sanitària, els processos de presa de decisions clíniques i la promoció de la investigació clínica. En aquest context, la cooperació entre la informació i els models de domini es considera essencial en la literatura, però la comunitat científica no l'ha abordada adequadament fins hui.

La contribució principal d'aquesta tesi és el desenvolupament de mètodes i ferramentes per a la inclusió d'expressions d'enllaços terminològics en regles de consistència. Les contribucions específiques són:

- Definició d'un mètode per a executar *ECs* sobre una base de dades de SNOMED CT orientada a grafs.
- Definició de mètodes per a simplificar *ECs* abans i després de la seua execució, i la seua validació semàntica conforme al *Machine Readable Concept Model* de SNOMED CT (MRCM).
- Definició d'un mètode per a visualitzar, explorar dinàmicament, comprendre i validar subconjunts de SNOMED CT.
- Desenvolupament de SNQuery, una plataforma que executa, simplifica i valida *ECs* i visualitza els subconjunts resultants.
- Definició de EHRules, un llenguatge d'expressions basat en l'openEHR *Expression Language* per a l'especificació de regles de consistència en arquetips, inclòs l'enllaç terminològic de contingut, amb la finalitat d'enriquir els arquetips amb coneixement del domini.
- Definició d'un mètode per a executar les expressions de EHRules amb la finalitat de validar la consistència de les dades de la HCE mitjançant l'execució d'aquestes expressions sobre instàncies de dades de pacients.

El nostre objectiu és que aquestes contribucions ajuden a millorar la qualitat de la HCE, ja que proporcionen mètodes i ferramentes per a la validació i millora de la consistència de les dades de la HCE. Pretenem, a més, mitjançant la definició d'enllaços de contingut entre models d'informació i terminologies clíniques, elevar el nivell d'interoperabilitat semàntica, per a la qual cosa la definició d'enllaços terminològics és crucial.



## CHAPTER 1. INTRODUCTION

---

### 1.1 MOTIVATION

The quality of the biomedical data contained in the Electronic Health Record (EHR) is a matter of special importance today, both for the improvement and reduction of errors in healthcare activity, as well as for either human or automatic decision-making processes (Clinical Decision Support Systems - CDSS). There are several conceptual frameworks for assessing the quality of biomedical data [1–3] and in all of them the dimension of consistency appears as basic.

The data are integrated into a specific medical domain, and it is necessary to assure the consistency of these data in accordance with the rest of the data in the domain. Despite the importance of the consistency of data and its relationship with semantic interoperability, it is a field that needs to be explored in depth. Concepts such as semantic binding and value set binding of information models, which are essential when developing semantically interoperable EHR systems, are key points and need to be strongly addressed from the point of view of research and development of standard EHR modelling tools.

In the particular case of clinical archetypes, the definition of semantic consistency rules has not been given the attention it deserves. Although it is true that the Archetype Definition Language (ADL) has a section of rules and a basic language for the definition of simple invariants, in general it is a section that, in practical terms, is yet to be defined and therefore is not currently implemented. In this sense, it is required to develop a language for expressing the clinical knowledge contained in the EHR by means of rules, including terminological bindings. In this context, SNOMED CT, which is considered the most comprehensive, multilingual and codified clinical healthcare terminology in the world [4], plays a key role, as well as the SNOMED CT Expression Constraint Language (ECL) [5] for the definition of intensional subsets of medical concepts.

As a consequence, the development of methods and tools for the specification of consistency rules involving archetypes and terminologies may represent a significant leap

in the total evaluation capacity of health data, as well as a competitive advantage over classic data profiling tools.

## 1.2 HYPOTHESIS, RESEARCH QUESTIONS AND OBJECTIVES

### 1.2.1 Hypothesis

The main hypothesis of this thesis is that the combination and joint exploitation of the most advanced techniques for information and domain models will facilitate the specification of the domain knowledge necessary to specify, improve, and measure the consistency of clinical data. Cooperation between information and domain models has been considered essential in the literature, but it has not been adequately addressed by the scientific community to date. Achieving consistency, together with common used data quality dimensions, such as completeness, correctness, plausibility, and currency, can contribute to the improvement of healthcare delivery, clinical decision-making processes, and clinical research. Unreliable data (e.g., incorrect or incomplete data, or missing data entries) will lead to biased analysis and wrong efforts to enhance the quality of clinical data, and also wrong conclusions in research studies. Quality assessment of clinical data is primordial to identify and alleviate problems in data quality for proper use and reuse [6].

### 1.2.2 Specific research questions and objectives

In order to confirm the hypothesis, the following research questions are identified:

R1	Can SNOMED CT Expression Constraints (ECs) be efficiently executed over a SNOMED CT graph-oriented database for the definition of subsets?
R2	Can ECs be simplified before and after its execution, and semantically validated?
R3	Can SNOMED CT subsets be visualized, explored, understood and validated?
R4	Can methods presented in R1, R2 and R3 be efficiently implemented into an execution engine?

R5	Can an expression language for the specification of consistency rules in archetypes, including value set bindings that make use of ECs, be defined?
R6	Can the expression language presented in R5 be applied to enrich the rules section of archetypes with domain knowledge?
R7	Can expressions defined with the language presented in R5 be executed over EHR data, including the execution of ECs in the engine presented in R4, in order to validate the consistency of the EHR data?

The following research objectives will answer satisfactorily the previous research questions:

O1	To define a method for the execution of ECs over a graph-oriented database.
O2	To define methods for pre- and post-execution simplification and semantic validation of ECs.
O3	To provide a method for the visual exploration of SNOMED CT subsets.
O4	To develop an EC execution platform that makes use of the methods presented in O1, O2 and O3.
O5	To define an expression language for the specification of consistency rules in archetypes.
O6	To enrich the rules section of the archetypes with the language of O5.
O7	To validate the consistency of EHR by executing the expressions inside of the rules section of archetypes, including the execution of ECs by using the engine of O4.

## 1.3 CONTRIBUTIONS

### 1.3.1 Main contributions

Main contributions included in this thesis are shown below. Contributions are associated with objectives. Additionally, they are associated with journal or conference publications where they have been published (see section 1.3.2).

C1	Definition of a method to execute ECs over a SNOMED CT graph-oriented database.
	Contributes to research objective O1.
	Published in P1, P3, P5, P6, P7
	For the representation of SNOMED CT, we have used a graph database model where the schema and instances are represented as graphs and the data manipulation is expressed by graph-oriented operations. Specifically, we have used Neo4j, a graph-oriented database software. Neo4j is a highly scalable native property graph database, purpose-built to leverage data and data relationships, which comes with a powerful query language called Cypher. The structurally similar data model facilitates the execution of complex queries that go beyond subsumption queries. We have implemented a mechanism for translating ECs into Cypher clauses that are executed over the SNOMED CT database in order to obtain the result subset.

C2	Definition of methods to simplify ECs before and after its execution and semantic validation according to the SNOMED CT Machine Readable Concept Model (MRCM).
	Contributes to research objective O2.
	Published in P1, P4

	<p>The main idea of the EC simplification is to reduce the complexity of the expression and, in turn, the execution time. It can be achieved by removing redundant concepts, superfluous refinements or by narrowing down the focus concept without affecting the completeness of the computed subset. Performing such simplification has several advantages. First, it diminishes the number of comparisons to be carried out during query execution, therefore it may have impact on query answering time. Second, the cut-down query is more amenable to human reading. Third, it may help to validate the terminology itself. We have defined three different methods to simplify ECs before they are executed (pre-execution): subsumption-based, MRCM-based, and logic definition-based; and one method based on the mining of the result subset (post-execution). The methods based on the MRCM and the logic definition check whether the EC satisfies the rules defined in the SNOMED CT Concept Model and the logical definition of the involved concepts.</p>
--	--

C3	Definition of a method to visualize, dynamically explore, understand and validate SNOMED CT subsets.
	Contributes to research objective O3.
	Published in P1, P3, P4, P5, P6, P7
	<p>Understanding and validating a result subset require to provide users with information about how the concepts that make up the subset are related in terms of hierarchies and which of these hierarchies are more relevant in terms of recall and precision. The visual representation of this information may facilitate the validation of the subset at a glance, for instance by detecting irrelevant concepts or hierarchies. A key requirement is to provide the possibility to dynamically explore sub-hierarchies by zooming into them to see their details. It is important to note that the presentation of all this information to the user requires a compact visual representation in order to see as much information as possible at a glance. To collect and present</p>

	<p>this information, a graphical visualization based on the circle packing is proposed that fits particularly well with our requirements.</p>
--	---

C4	<p>Development of SNQuery, an execution platform that executes, simplifies and validates ECs, and visualizes the resulting subsets.</p>
	<p>Contributes to research objective O4.</p>
	<p>Published in P1, P3, P4, P5, P6, P7</p>
	<p>We have developed SNQuery, an EC execution platform that makes use of the methods that we have defined for execution, simplification, semantic validation and visual representation. Simplification methods are applied by SNQuery iteratively, i.e., the methods are executed sequentially in a loop until the EC cannot be further simplified. This process yields both the semantic validation and the simplification of the EC. To execute the EC, SNQuery performs a translation process between the EC and the Cypher Query Language, which yields a single or multiple target Cypher queries that are executed over the Neo4j graph database to produce the intended result subset. Additionally, an analysis of SNQuery in terms of execution times has been performed.</p>

C5	<p>Definition of EHRules, an expression language based on the openEHR Expression Language for the specification of consistency expressions in archetypes, including value set bindings, in order to enrich archetypes with domain knowledge.</p>
	<p>Contributes to research objective O5 and O6.</p>
	<p>Published in P2</p>
	<p>After a survey on existing languages for expressing clinical knowledge formally, we have defined EHRules, a language that extends the openEHR EL with some elements that are particularly needed for the specification of</p>



	<p>nested rules and value set bindings, including the ability to support SNOMED CT ECL. EHRules is provided as a way of authoring expressions and rules in textual form. This approach is the same as with any programming language, where the usual form for learning and programming is the abstract language form, while the computational form is an abstract syntax tree (AST). The EHRules language provides a syntax that may be used to specify archetype rules. Our objective is to provide a mechanism to define scenario-dependant constraints in clinical information models, specifically archetypes.</p>
--	--

C6	<p>Definition of a method to execute EHRules expressions in order to validate the consistency of EHR data by executing such rules over patient data instances.</p>
	<p>Contributes to research objective O7.</p>
	<p>Published in P2</p>
	<p>The main objective of the definition of consistency rules in the rules section of archetypes is to improve and validate the EHR data consistency, as a dimension of quality. For this purpose, we have developed an automatic translation process that converts EHRules expressions into a set of Schematron rules that are executed against EHR data instances. Specifically, the process creates a Schematron file that is capable of being opened in any tool that includes a Schematron rules execution engine to validate XML files and check whether these rules are met or not by the XML data, i.e., to set whether EHR instances are valid or not from a consistence point of view.</p>

### 1.3.2 Journal publications and conference contributions

The work presented in this thesis spreads through over seven years of research and development. The following journal publications and conference contributions describe the results achieved and reflect the work done in the context of specific research projects

that required the use of SNOMED CT for the definition of terminological bindings with clinical information models, and to enrich the definition of archetypes with specific domain knowledge in order to achieve high levels of EHR data quality and, in turn, high levels of semantic interoperability between EHR systems.

Journal publications	
P1*	<p>Vicente Miguel Giménez Solano; José Alberto Maldonado Segura; Diego Boscá Tomás; Santiago José Salas García; Montserrat Robles Viejo. <i>Definition and validation of SNOMED CT subsets using the expression constraint language</i>. Journal of Biomedical Informatics. 117, pp. 1 - 15. 2021. ISSN 1532-0464</p> <p><b><i>*This publication directly contributes to the results of this thesis. It describes the methods and tools included in Chapters 3 and 4.</i></b></p>
P2	<p>José Alberto Maldonado Segura; Mar Marcos; Jesualdo Tomás Fernández-Breis; Vicente Miguel Giménez Solano; María Del Carmen Legaz-García; Begoña Martínez-Salvador. CLIN-IK-LINKS: A platform for the design and execution of clinical data transformation and reasoning workflows. Computer Methods and Programs in Biomedicine. 197. 2020.</p>
Conference contributions	
P3	<p>Vicente Miguel Giménez Solano; José Alberto Maldonado Segura; Diego Boscá Tomás; David Moner Cano. <i>Terminology binding for the measurement of consistency of health data</i>. SNOMED CT Expo. Online, 2021.</p>
P4	<p>Vicente Miguel Giménez Solano; José Alberto Maldonado Segura; Diego Boscá Tomás; Santiago José Salas García. <i>SNQuery: un motor para ejecutar consultas sobre SNOMED CT</i>. XXIII Congreso Nacional de Informática de la Salud. Madrid, 2020.</p>
P5	<p>Vicente Miguel Giménez Solano; José Alberto Maldonado Segura; Diego Boscá Tomás; David Moner Cano. <i>Validation and simplification of expression constraints</i>. SNOMED CT Expo. Online, 2020.</p>

P6	Vicente Miguel Giménez Solano; José Alberto Maldonado Segura; Santiago José Salas García; Diego Boscá Tomás; Montserrat Robles Viejo. <i>Implementation of an execution engine for SNOMED CT Expression Constraint Language</i> . Medical Informatics Europe Conference. Munich, 2016.
P7	Vicente Miguel Giménez Solano; José Alberto Maldonado Segura; Montserrat Robles Viejo. <i>Definición de subconjuntos en SNOMED CT</i> . XXXIII Congreso Anual de la Sociedad Española de Ingeniería Biomédica. Madrid, 2015.
P8	Vicente Miguel Giménez Solano; José Alberto Maldonado Segura; Montserrat Robles Viejo. <i>Definición de Subconjuntos en SNOMED CT</i> . XIV Congreso Nacional de Documentación Médica. Granada, 2015.

### 1.3.3 Projects

The author of the thesis has participated in three R&D projects in the field of semantic interoperability of health information. These projects served to learn about the problems and needs for creating semantically interoperable EHR systems. Specifically, the author has focused its work in the enrichment of archetypes using consistency rules and value set bindings with SNOMED CT subsets. Projects are listed below:


Project	<i>Semantic Enrichment of Archetypes Applied to the Validation and Measurement of Health Data Quality</i>
Entity	<i>Ministerio de Ciencia, Innovación y Universidades</i>


Project	<i>Clinical Knowledge and Information Models to Link the Electronic Health Record with Clinical Decision Support Systems I</i>
Entity	<i>Universitat Politècnica de València</i>

Project	<i>Elaboration of Archetypes of Clinical Documents of the Minimum Set of Data of Clinical Reports and Electronic Prescription of The Spanish Sistema Nacional de Salud (SNS), and Implementation of Digital Services for Provision of Models for the Electronic Health Record and Electronic Prescription</i>
Entity	<i>Universitat Politècnica de València</i>

### 1.3.4 Software

The author of this thesis has been involved in the design and development of two web applications required for the adequate development of the thesis. They are listed next.

	<p>SNQuery is a web platform that includes an execution engine for the ECL SNOMED CT language. It allows to edit, parse, simplify, semantically validate and execute ECs in order to specify valid and efficient SNOMED CT subsets. It includes a visual tool for exploring and analyzing the result subset. Additionally, for training purposes, SNQuery incorporates a visual authoring tool that is particularly useful for users without a deep knowledge of ECL. The tool provides a series of predefined templates that can be combined to create simple and complex ECs.</p>
--	---



SNLoader is a web application that includes a module to import and transform RF2 SNOMED CT files into a Neo4j graph-oriented database. SNLoader has the ability to create both international editions of SNOMED CT and local extensions, such as the Spanish SNS clinical and drugs extensions, including reference sets (refsets). Additionally, SNLoader contains a validator module of extensions able to search for consistency errors in RF2 extensions. The results of both validation of extensions and creation of databases are presented in two separate reports.

### 1.3.5 Grants

The author was beneficiary of grant DIN2018-009951 (*Doctorados Industriales*) of the *Ministerio de Economía y Competitividad* of Spain between 2020 and 2021. The objective of this grant is to support the development of doctoral thesis inside private companies. Thus, the research results are grounded in real needs of the market and provide a direct benefit to the company activity. In this case, the thesis was partially developed in VeraTech for Health, a spin-off company of *Universitat Politècnica de València*.

Additionally, the author was beneficiary of a pre-doctoral contract for the training of research staff (*Formación de Personal Investigador - FPI*) of *Universitat Politècnica de València* between 2015 and 2018. The objective of this grant is to enable the research training of recent graduates by allowing their exclusive dedication to research training activities through the completion of a doctoral thesis. In this case, the thesis was partially developed in *Instituto Universitario de las Tecnologías de la Información y las Comunicaciones - ITACA* of *Universitat Politècnica de València*.

### 1.3.6 Research visits

The author of the thesis did a research stay in the Department of Health Science and Technology at the University of Aalborg (Denmark), from April 1<sup>st</sup> 2017 to June 30<sup>th</sup> 2017. During the stay, a literature review of existing languages to express medical logic was

performed, including PROforma, GLIF3, Asbru, SAGE, EON, Arden Syntax, GELLO, Guideline Definition Language (GDL) and openEHR Expression Language (EL). Two real use cases from the host group were analysed: the Danish standard FSIII for the specification of inter-municipal guidelines for documenting social- and healthcare observations and interventions in home nursing, rehabilitation, exercising and home care; and a requisition for radiology procedures from the North Denmark Region clinical information system. After the survey on existing languages, a comprehensive comparison between the Guideline Definition Language (GDL) and the openEHR Expression Language (EL) was carried out. The rationale of comparing these two languages was that both are rule-based with inbuilt mechanisms for accessing archetype instances and they can accommodate terminology bindings. Requirements for an expression language that was proposed for the definition of consistency rules in archetypes were defined, including conditional value set binding rules. This language later became the EHRules language, which is part of this thesis.

### 1.3.7 Other contributions

During the development of the thesis, the author has been involved in the preparation of material and teaching of both theoretical and practical classes in the subject Information Systems and Telemedicine II of the Degree in Biomedical Engineering of the *Universitat Politècnica de València* for two years. Practical classes included basic learning and handling of SNQuery.

Additionally, the author has participated as teacher focused in SNOMED CT and UMLS terminologies in the Master in Medical Documentation at *Universitat de València*. Practical classes included basic learning and handling of SNQuery.

SNQuery is included in the list of implementations of the ECL language of SNOMED International<sup>1</sup>.

Since 2018, the author is member of the SNOMED International Languages Project Group (SLPG), where he participates actively in the development and improvement of the ECL language.

---

<sup>1</sup> <https://confluence.ihtsdotools.org/display/slp/snomed+ct+expression+constraint+language>

## 1.4 STRUCTURE OF THE THESIS

The thesis is structured as follows. Chapter 1 introduces the motivation and the main hypothesis, together with the research questions, research objectives, and contributions of the thesis. Chapter 2 presents an introduction to the thesis including an explanation of its objectives. It also introduces the fundamentals of consistency as a dimension of quality of EHR, and the basic elements of semantic interoperability, namely archetype-based EHR architectures, terminology binding, and SNOMED CT, including the ECL language. The chapter ends with an introduction to the openEHR EL language. In Chapter 3 we present a series of methods that we have defined to execute, simplify and semantically validate ECs over a graph-oriented database, and a method to visualize and explore the result subsets. Chapter 4 describes SNQuery, an execution platform that the author has developed in collaboration with his research colleagues, which incorporates the methods presented in Chapter 3, together with a validation of the platform in terms of execution times. The chapter ends with a discussion that involves the content of both Chapter 3 and Chapter 4, which are closely related. Chapter 5 identifies the requirements for the definition of consistency rules in archetypes, including value set binding rules, for which it is necessary to provide an expression language allowing such abilities. The chapter ends with a literature review of existing languages for specifying medical knowledge and logics, and the justification of choosing EL to be the basis of our expression language, along with a list of extensions to be made on EL to support our specific requirements. Chapter 6 describes the EHRules expression language for the definition of consistency rules in archetypes, together with three real uses cases to validate the language. The chapter ends with an explanation of the rule execution process for the validation of EHR data instances from a consistency point of view, and a discussion. Finally, Chapter 7 presents the conclusions of the thesis and introduces future works.

At the end of the thesis four annexes are added, namely: Annex 1: List of ECs examples; Annex 2: EHRules ABNF syntax specification; Annex 3: FSIII Condition-Interventions; and Annex 4: From EHRules to Schematron examples

Table 1 shows the overall structure of the thesis, where each chapter is associated with its research questions, objectives, contributions, and publications.

Chapter	Title	Research questions	Objectives	Contributions	Publications
1	Introduction				
2	Background				
3	Methods for the definition, validation, execution and visualization of SNOMED CT subsets using the Expression Constraint Language	R1, R2, R3	O1, O2, O3	C1, C2, C3	P1, P4, P5, P6, P7, P8
4	SNQuery: ECL execution engine based on graph databases	R4	O4	C4	P1, P4, P5, P6, P7, P8
5	Consistency rules in archetypes	R5, R6	O5, O6	C5	P3
6	The EHRules Language	R7	O7	C6	P3
7	Conclusions and future work				

Table 1. Structure of the thesis and relationship with the research questions, objectives, contributions, and publications



## CHAPTER 2. BACKGROUND

### 2.1 INTRODUCTION

An EHR represents the set of information about a person's health, as well as the care processes and health procedures received over the years. Ideally, this information should cover from birth to death.

However, the enormous variability of health information is a problem when building a unified EHR from existing data in different health centres or information systems. For example, SNOMED CT<sup>2</sup>, the world-wide reference terminology in medicine, defines more than 300,000 different concepts and more than 1,500,000 relationships between these concepts (see Figure 1).

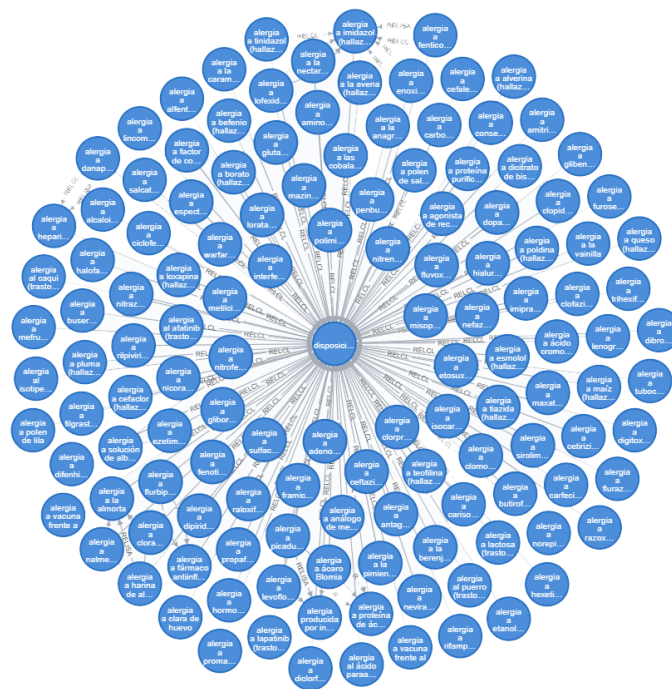


Figure 1. A SNOMED CT subset visualized using the Neo4j browser

<sup>2</sup> [www.snomed.org](http://www.snomed.org)

In the case of clinical information models, such as types of clinical documents, the public repository of openEHR archetypes (i.e., the Clinical Knowledge Manager - CKM<sup>3</sup>) contains almost 800 different definitions either in published, draft or pre-draft states (see Figure 2).

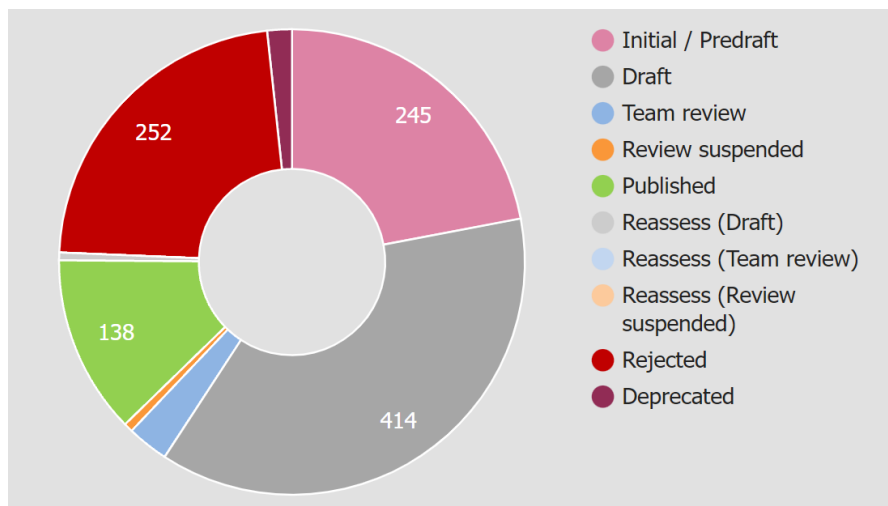


Figure 2. Archetypes status in the CKM repository

Faced with the problem of accessing, communicating and aggregating health information that is distributed and represented in different formats, it is necessary to create definitions of the contents of the EHR through formal mechanisms such as detailed clinical models (such as archetypes) and applying a clear and well specified methodology. It is also necessary to have tools that facilitate the management and use of these definitions.

On the other hand, clinical information systems are essential to facilitate patient care and follow-up through access to up-to-date information about them at different stages of care. However, the use of such data for the extraction of knowledge or decision-making, or to offer reliable and shareable healthcare reports, requires having the data of adequate quality. The lack of quality can hinder both the information and knowledge generation processes and lead to biased results in health processes such as research or the monitoring of indicators. From a data quality point of view, the complexity and variability of medical data makes it necessary to verify its quality from various points of view, which are generally

<sup>3</sup> <https://ckm.openehr.org/ckm/>

known as data quality dimensions. One of these dimensions is consistency, that is, data should satisfy a set of constraints that ensure its integrity and coherence. These constraints can be of multiple types, such as formats, ranges, allowed values, references to external terminologies, compliance with data structures, domain rules (for example, an EHR should not contain obstetric data if the patient is male), presence of mandatory values or consistency of calculated values (see Figure 3).

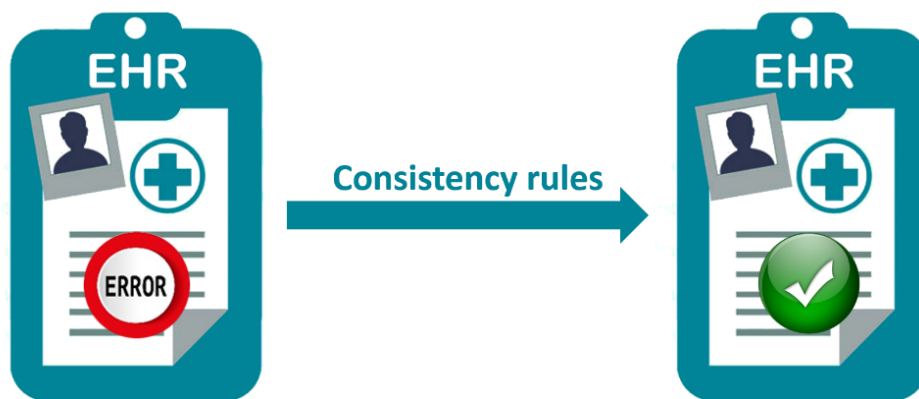


Figure 3. Consistency rules help solving consistency errors in EHR

The development of the consistency dimension requires the specification of complex domain rules. We have detected that these rules must include medical terminologies, conditional value set bindings, intensional specification of subsets and the exploitation of its concept model if it exists, as is the case of SNOMED CT.

The consistency dimension is completely dependent on the use scenario, and therefore it is necessary to explicitly define the conditions that the data must satisfy in order to be considered of quality. The approach that we follow in this thesis is to use standards for the communication and representation of health data [7], including terminologies [8,9], for the specification of these conditions. This approach brings several advantages. On the one hand, it facilitates the interoperability of project developments with EHR systems. On the other hand, due to its generality it allows the representation of any type of data structure, facilitating the reuse of structures and quality criteria between different scenarios. To this end, there are standard languages for the definition of complex clinical concepts such as

ADL. ADL natively supports in both version 1.4 (ISO standard) and version 1.5 various types of constraints completely related to this quality dimension. ADL incorporates constructors for the basic specification of constraints on the domain of attributes (consistency) and enforceability (completeness and contextualization). Therefore, it is a suitable base for this work.

## 2.2 OBJECTIVES

The execution of this thesis is intended to address the problem of having quality medical data with a high level of consistency, which will help to alleviate the problem of accessibility, communication and aggregation of the health data. In this regard, the main purpose of this thesis is the development of methods and tools for the specification of domain rules. These rules include SNOMED CT Expression Constraints (ECs) [5] for the intensional specification of subsets, and value set bindings to enhance the consistency of health data. Performing such terminology bindings will also help achieving semantic interoperability between EHR systems.

The objectives are:

1. To define and implement a method for the execution of ECs for the definition of subsets of medical concepts over a graph-oriented SNOMED CT database.
2. To define and implement methods for the simplification and semantic validation of ECs pre- and post-execution.
3. To provide a method for the visual representation of SNOMED CT subsets in order to explore, understand and validate its content.
4. To develop an EC execution platform which makes use of the methods presented in objectives 1, 2 and 3 to define and provide validated SNOMED CT subsets.
5. To define an expression language for the specification of consistency rules, including simple, conditional and dependency value set bindings in clinical models, such as archetypes, to define the meaning and content of the elements and data structures contained in the EHR.
6. Based on the expression language, to enrich the rules section of the archetypes with multi-attribute consistency rules with support to clinical terminologies for the explicit and computable definition of constraints on the structure and domain of

EHR extracts. Such consistency rules should make use of the SNOMED CT ECL for the definition and binding to SNOMED CT subsets.

7. To validate EHR data instances from a consistency point of view by executing consistency rules over patient data contained in the EHR.

## 2.3 CONSISTENCY OF EHR

In the context of EHRs, one of the most broadly adopted conceptualizations of data quality is based on the 'fitness for use' concept, i.e., data are of enough quality when they serve the needs of a given user pursuing specific goals. Five of the most important dimensions of data quality are completeness, correctness, concordance (i.e., consistency), plausibility, and currency [1]. Such dimensions answer the following questions:

- *Completeness*: Is a truth about a patient present in the EHR?
- *Correctness*: Is an element that is present in the EHR true?
- *Concordance*: Is there agreement between elements in the EHR, or between the EHR and another data source?
- *Plausibility*: Does an element in the EHR makes sense in light of other knowledge about what that element is measuring?
- *Currency*: Is an element in the EHR a relevant representation of the patient state at a given point in time?"

Additionally, Weiskopf and Weng [1] identify up to seven broad categories of methods for the assessment of data quality: comparison with gold standards, data element agreement, data source agreement, distribution comparison, validity checks, log review, and element presence. These assessment methods are defined as follows:

- *Gold standard*: A dataset drawn from another source or multiple sources, with or without information from the EHR, is used as a gold standard.
- *Data element agreement*: Two or more elements within an EHR are compared to see if they report the same or compatible information.
- *Element presence*: A determination is made as to whether or not desired or expected data elements are present.

- *Data source agreement*: Data from the EHR are compared with data from another source to determine if they are in agreement.
- *Distribution comparison*: Distributions or summary statistics of aggregated data from the EHR are compared with the expected distributions for the clinical concepts of interest.
- *Validity check*: Data in the EHR are assessed using various techniques that determine if values 'make sense'.
- *Log review*: Information on the actual data entry practices (e.g., dates, times, edits) is examined."

According to the literature, there exist four terms to describe concordance: agreement, consistency, reliability and variation [1] (note that, in this thesis, we use such terms as synonyms). Health data are consistent when there is agreement or compatibility between the data elements [1]. This means that the values stored by a set of different data items make sense when considered together, especially diagnoses and associated information such as medications or procedures. The measurement of consistency is generally based on elements contained by the EHR, although some researchers also include information from other sources. These other sources included billing information, paper records, patient-reported data, and physician-reported data. In both cases, it is mandatory to have a detailed specification of the conditions that data must satisfy to be considered as consistent, which includes the validation of the terminological concepts that are used. Another approach was to compare distributions of data within the EHR with distributions of the same information from similar medical practices or with national rates [1].

## 2.4 ARCHETYPE-BASED EHR ARCHITECTURES

The complete and reliable representation of the clinical information that is part of the EHR of a patient is a complex problem to solve that has been a focus of attention of Spanish and European R & D & I policies for more than a decade [1]. The problem includes both the formal definition of the contents of the EHR and its semantic interoperability [7].

Traditional development models require a deep analysis of the requirements for the correct operation of the systems. This means that systems must be continuously modified

due to changing requirements and transformed and adapted to new needs. Furthermore, it is difficult to represent all the different concepts or clinical business objects that the system works with.

The dual model of development [7] proposes a clear separation between the information and the business concepts used by the system. Information is structured data that is stored in the system and that does not change over time. The set of business concepts with which professionals work represents the implicit knowledge of the domain that influences in how the information system works. For example, in the healthcare case, the concepts include the different types of clinical reports, tests, laboratory results, etc. In the dual model, these concepts are represented through archetypes that can be modified in their definition without affecting the operation of the rest of the system or the data already stored. In addition, archetypes provide information with a semantic layer that describes its exact meaning.

A reference model (RM) represents the global characteristics of the components of the EHR, defines how they are aggregated, and the information of context required to comply with the ethical, legal and provenance requirements of health information. This model defines the set of classes that make up the basic and generic components of the EHR. Due to this genericity, a RM is considered a stable model over time, which will require little or no change to adapt to new information recording needs (see Figure 4 and Figure 5).

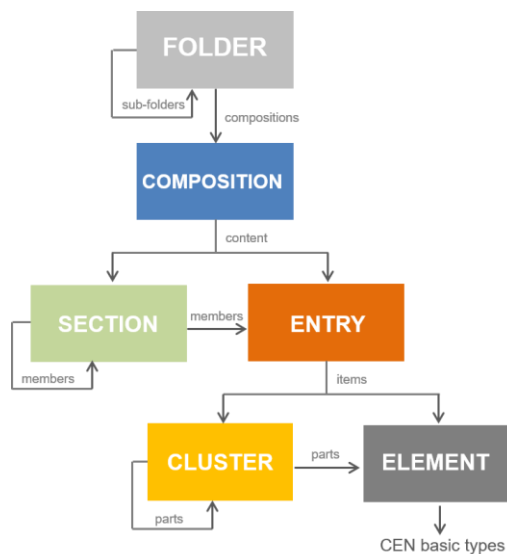


Figure 4. Reference model of the ISO13606 standard

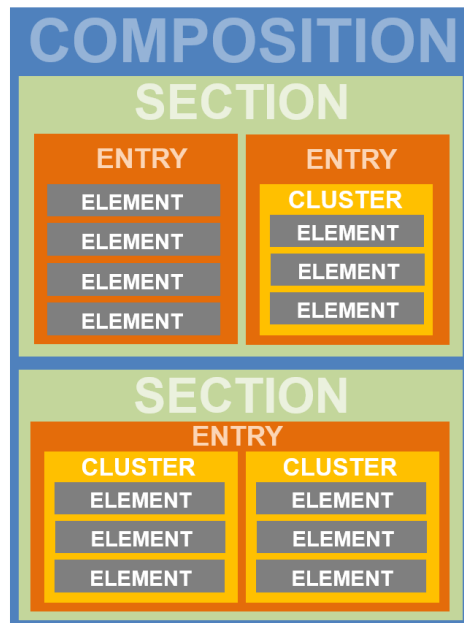


Figure 5. Example of a combination of the reference model classes of the ISO13606

Once this generic RM has been defined, the way to define specific information structures, i.e., the definition of the so-called clinical concepts, remains pending. This task is satisfied through detailed clinical models in the form of archetypes [10]. An archetype is the formal definition of a clinical concept through the combination and constraint of the classes defined in the RM for specific clinical domains. These definitions will guide systems in how to create data instances (RM instances) that meet all the characteristics and constraints defined in the archetype. From the point of view of information structures, an archetype provides a default structure for EHR information based on elements of the RM, as well as constraints on values, types, and cardinalities, among others, of those elements that give rise to valid data instances. From a semantic or knowledge point of view, an archetype provides a high-level semantic description of clinical concepts that can be automatically processed by health information systems and regardless of computer systems facing changes in the knowledge domain.

The approach based on archetypes is currently receiving the attention of multiple organizations, both public and private, since it is a future option for the management of health information and documentation [11,12] and its reuse in clinical research [13,14]. As



an example, the Spanish *Ministerio de Sanidad, Servicios Sociales e Igualdad* has developed within its national EHR project (*Historia Clínica Digital del Sistema Nacional de Salud*) a set of ISO 13606 archetypes that define the contents of the Minimum Reporting Data Set Clinics that must be shared by the different Autonomous Communities (see Figure 6).

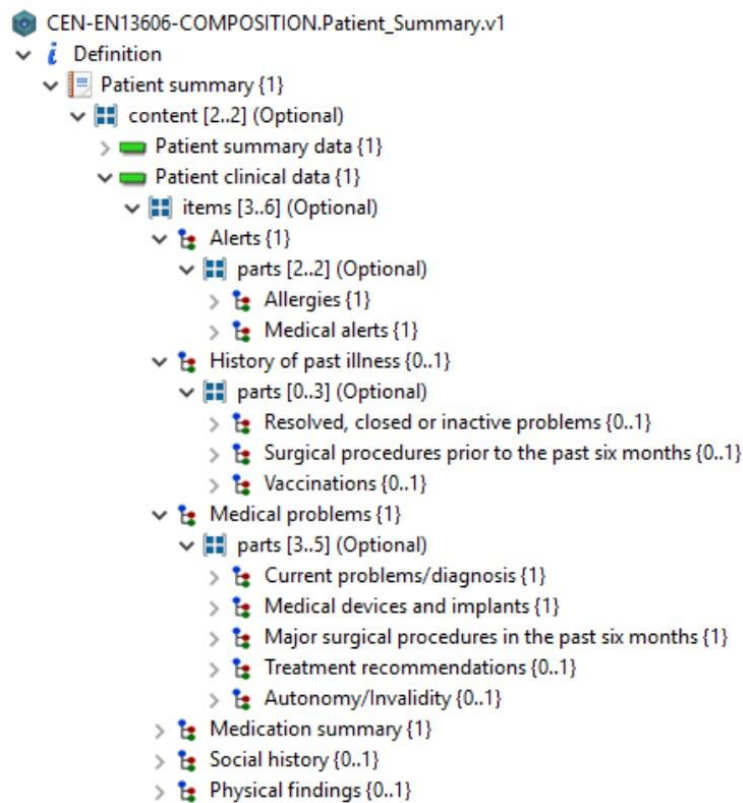


Figure 6. Example of a patient summary archetype

## 2.5 TERMINOLOGY BINDING

Decision support and information retrieval, which are two important objectives of EHR systems, require the information to be computable in a meaningful way. To interpret the meaning of information correctly depends on:

- The way information is structured
- The way clinical concepts are represented
- The way the terminology is used within the structure

Therefore, to make EHR meaningful, it is required a consistent approach to the interface between structural and terminological representations of information<sup>4</sup>.

In general terms, terminology binding between clinical information models and terminologies is a required step in order to achieve a high level of semantic interoperability between EHR systems, and therefore a high level of quality of EHR (see Figure 7).

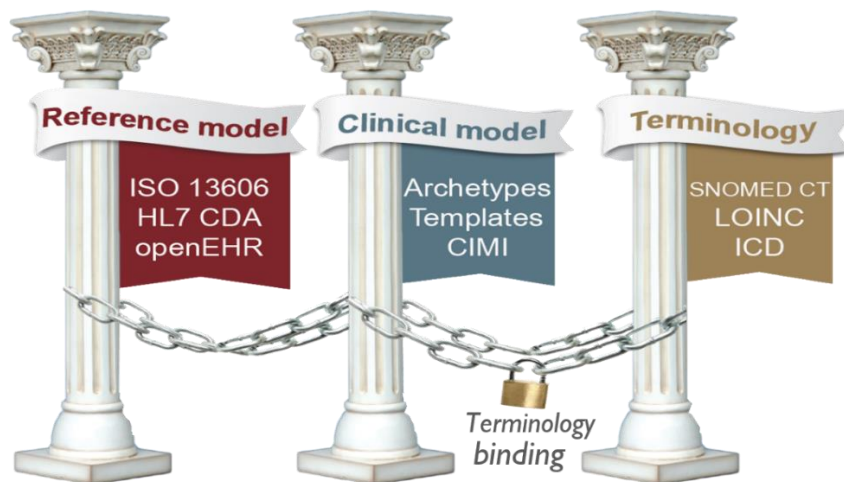


Figure 7. The three fundamental pillars of semantic interoperability

Historically, designers of information models assumed that codes from any terminology could be used within their information models. At the same time, the developers of medical code systems and terminologies had the assumption that their products could be used in any model. Both parts made assumptions about the nature of each other's work. This usually leads to overlaps and problems between the structured models and representation of meaning, which in turn increases the possibility of having ambiguities and different ways of specifying the same clinical meaning. As a result, the consistent retrieval of information based on semantics is hindered.

Currently it is widely accepted that information models and terminologies are interdependent. There is a need to design and build on the strengths of both models and terminologies, and to define guidelines in order to avoid ambiguities and set manners of representing meaningful information. It should be noted that these guidelines should define the way in which terminologies, such as SNOMED CT, are bound to EHR. These

binding is known as terminology binding (see Figure 8), and it provides the key to the consistent use of SNOMED CT in EHR systems<sup>4</sup>.

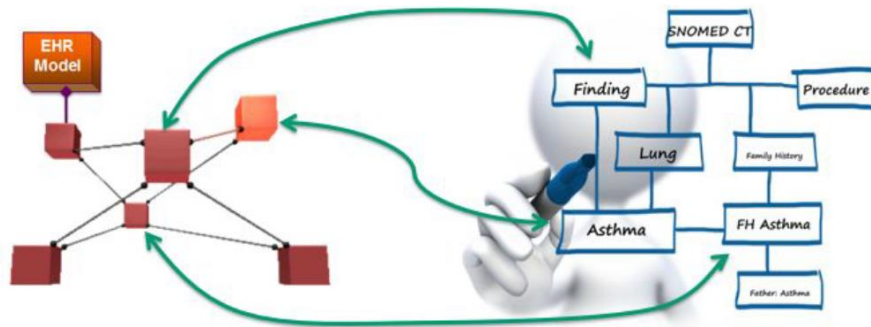


Figure 8. A draw showing terminology bindings between elements of an EHR model and SNOMED CT concepts<sup>4</sup>

Such guidelines should be defined to establish the way in which information models should be bounded to SNOMED CT. To support the creation of such guidelines, a literature review has recently been performed in [15] about terminology binding processes regarding SNOMED CT to find both recommendations and knowledge gaps. After analyzing 55 papers published from 2004 to 2020, the authors have given recommendations for practitioners and researchers. They have concluded that a better knowledge of SNOMED CT and a better cooperation between informaticians and domain experts could solve the problems on how information models and SNOMED CT should be combined using terminology bindings to share EHR data unambiguously and, in turn, improving semantic interoperability. Specifically, the authors have identified three topics about the recommendations found in the papers.

- *Ensure domain knowledge and informatics competence*: more than a half of the papers analysed that include recommendations emphasize the need of knowledge of both medical domain and informatics inside the project [16–19].

<sup>4</sup> Dr Linda Bird (SNOMED International) - Introduction to Terminology Binding

- *Follow a process including validation:* the papers included in the review involving descriptions of processes start with domain analysis, making process flow diagrams or sorting of the input material [17]. Additionally, either a validation step or a continuous validation process is recommended [19]. In one paper the authors have found recommendations regarding hierarchies and granularity when choosing a concept [20].
- *Plan for maintenance:* plans and systems for the maintenance of subsets and maps are required [21], which can be an important expense [22].

Finally, the authors give a set of recommendations for both practice and research based on the findings reported, including to sort input data, to select relevant concepts and to validate them, to involve domain experts and invest time in educating them in informatics and SNOMED CT, to use general concepts for unspecified terms, to enable free text entries when subsets are incomplete, to report incorrect terms to National Release Centre (NRC) or SNOMED International, and to develop local extensions when required, among many others.

There exist two types of terminology binding. First, semantic (or model meaning) binding, which binds an element of the information model with a concept or expression in the terminology in order to define the meaning of such element (see Figure 9).

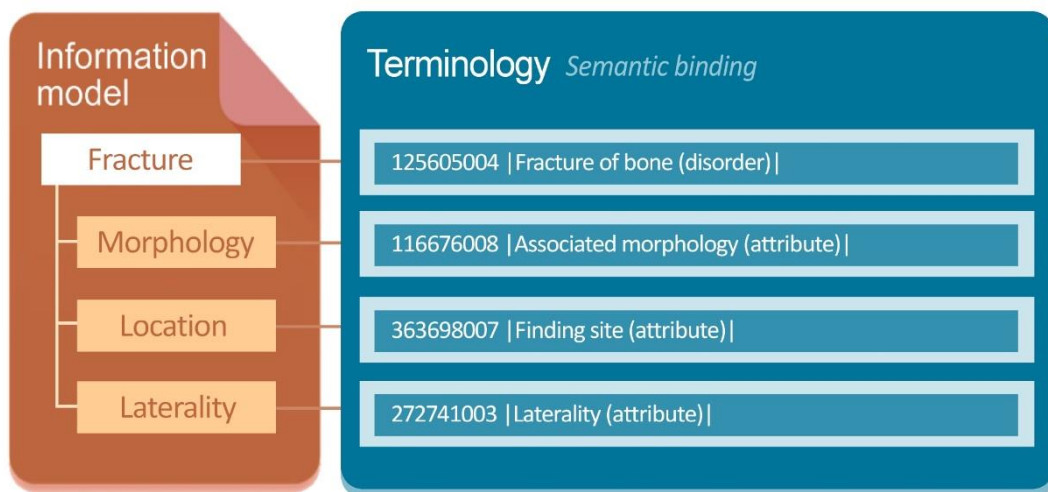


Figure 9. Example of semantic binding

And second, value set (or content) binding, which associates an element of the information model with a subset of concepts of the terminology. In other words, a value set binding records the subset of possible values which can populate a coded data element or coded attribute in the information model. At the same time, there exist four types of value set binding: simple, where a given coded data element is bound to a single value set (see Figure 10); conditional, where a condition is used to determine which value set is used (see Figure 11); dependency, where the value of one data element depends on the value of another (see Figure 12); and compositional, where the value of a data element is composed from the values of more than one data element (see Figure 13).

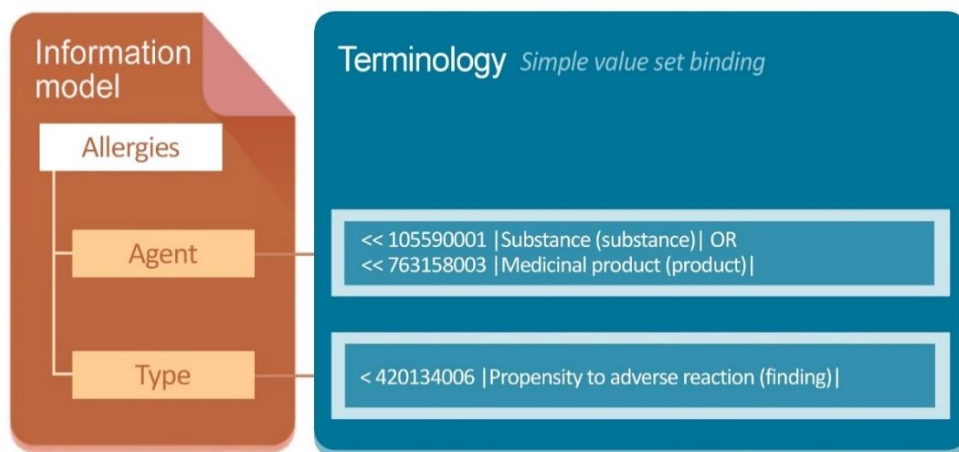


Figure 10. Example of Simple value set binding

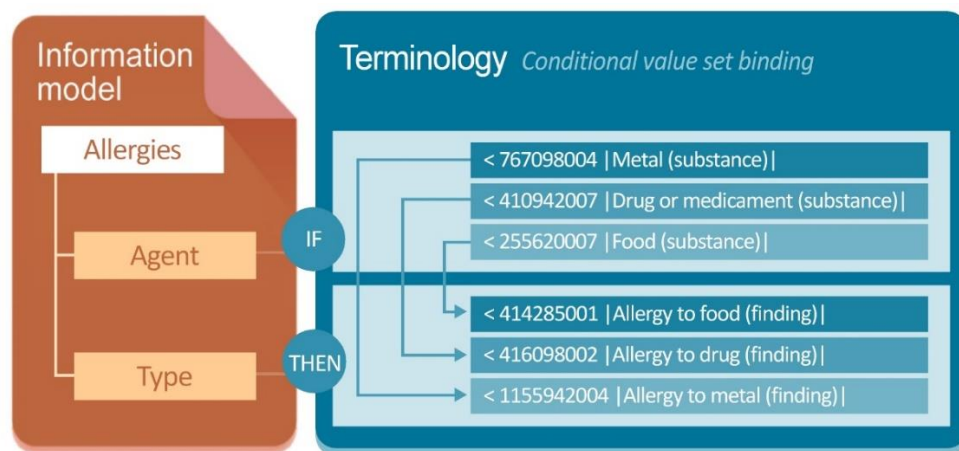


Figure 11. Example of Conditional value set binding

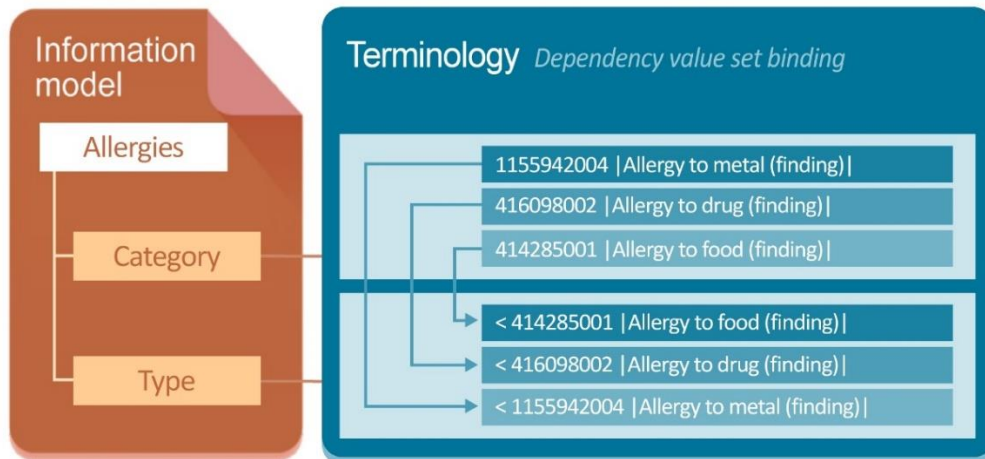


Figure 12. Example of Dependency value set binding

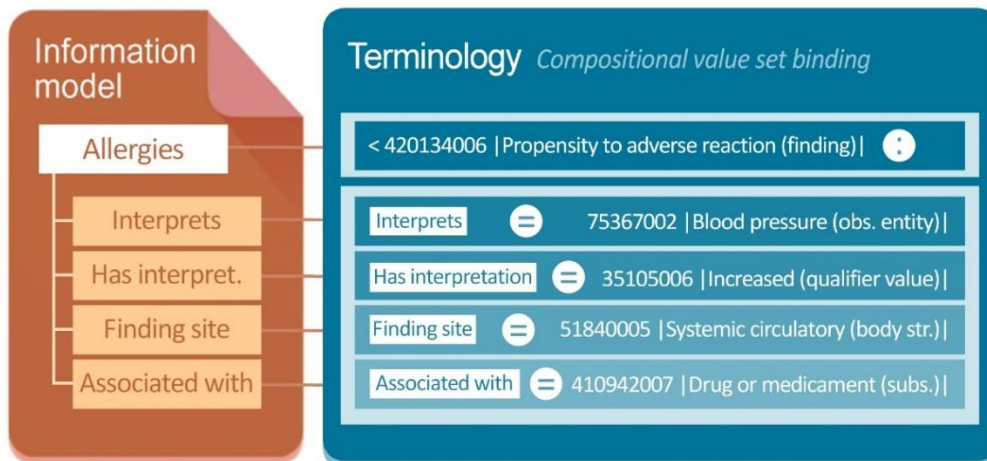


Figure 13. Example of Compositional value set binding

Additionally, terminology bindings include metadata, such as scope or type. Metadata allows understanding the binding rightly. The information model artefact, such as an archetype, to which the terminology is bound, could be either a whole information model, a single data group or data element, a datatype attribute, or a data value. The terminology artefact bound to the model could be a code, a set of codes either extensionally or intensionally defined by means of an EC, an expression, or a set of expressions.

## 2.6 SNOMED CT

### 2.6.1 Introduction

SNOMED CT is the acronym of Systematized Nomenclature of Medicine - Clinical Terms. SNOMED International is the organization that maintains and distributes SNOMED CT. It is considered the most complete terminology. In 2016 it was judged “the best available core reference terminology for cross border, national, and regional eHealth deployments in Europe” [23]. It is scalable, that is, it offers the possibility of adding new concepts using some mechanisms (i.e., pre- and post-coordinated expressions). It is translated from English into several languages: Spanish, Danish, Dutch, Swedish, etc. It contains local extensions that contain new concepts created specifically for each extension according to particular requirements (see Figure 14). SNOMED CT releases include mappings to other terminological standards, such as the International Classification of Diseases (ICD).

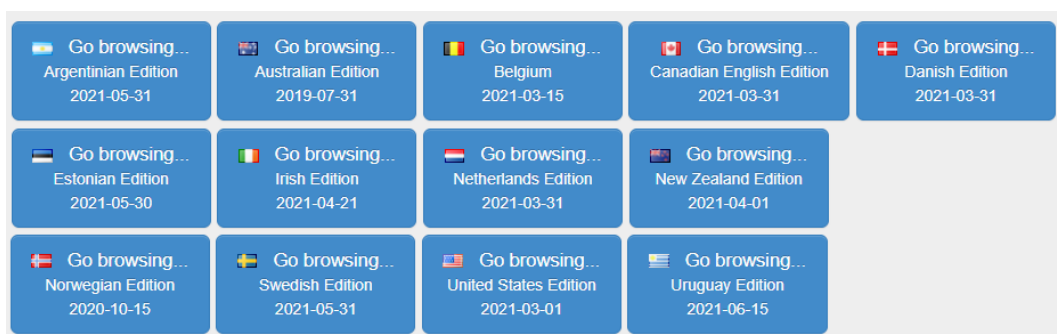


Figure 14. Local extensions of SNOMED CT <sup>5</sup>

The founding nine charter Members were Australia, Canada, Denmark, Lithuania, Sweden, the Netherlands, New Zealand, the United Kingdom and the United States. Trading as SNOMED International, the organization has grown to 40 Members (see Figure 15) and has issued Affiliate Licenses to more than 5,000 individuals and organizations.

<sup>5</sup> <https://browser.ihtsdotools.org/>

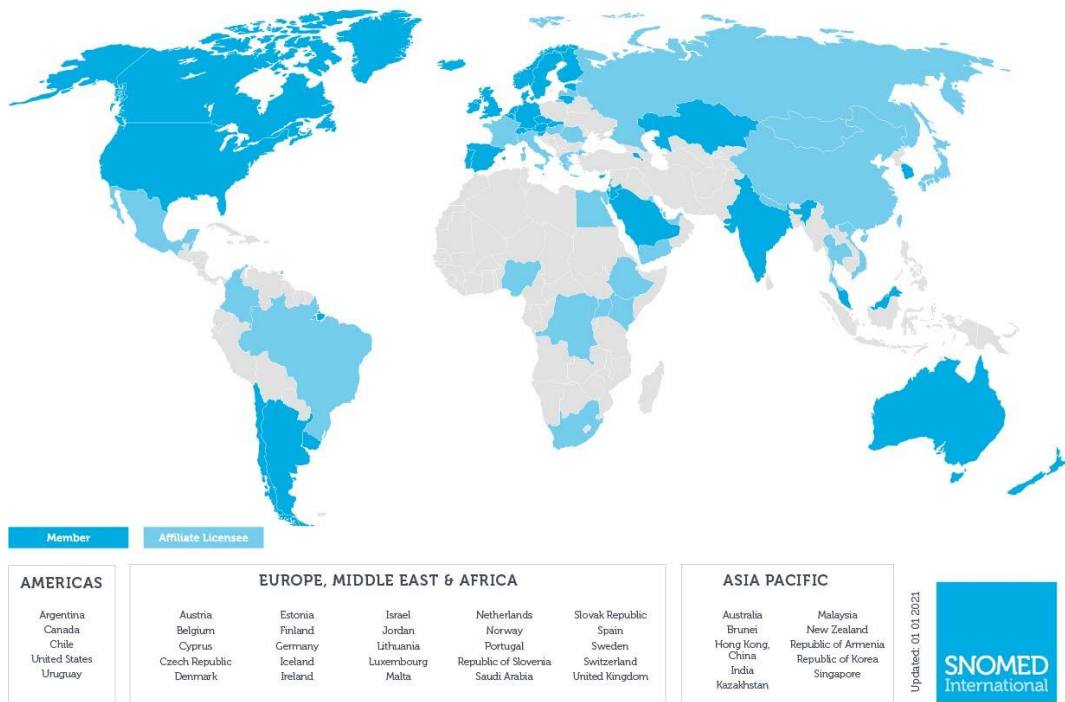


Figure 15. Member map of SNOMED International <sup>6</sup>

Using SNOMED CT requires an Affiliate License. The obtaining process depends on where SNOMED CT is to be used. If it is a member country, such as Spain<sup>7</sup> then it is possible to download the files from SNOMED CT directly. In the case of Spain, they can be downloaded after obtaining a license from *Ministerio de Sanidad, Consumo y Bienestar Social* <sup>8</sup>. It is important to note that SNOMED International releases two annual versions of SNOMED CT, one in January and one in July. In addition, the Spanish translation is released three months later (April and October) and the Spanish extension is released five months later (June and December).

SNOMED CT is distributed to its affiliates through a series of downloadable files. The RF2 (Release Format 2) is the format currently used to represent the SNOMED CT content. These files contain a set of records whose fields (columns) are separated by tabs. In

<sup>6</sup> <https://www.snomed.org/our-customers/members>

<sup>7</sup> <https://www.snomed.org/our-customers/member/spain>

<sup>8</sup> <https://snomed-ct.sanidad.gob.es/snomed-ct/solicitudLicencia.do>



In addition, the UTF-8 encoding system is used. There are three individual files with specific fields for each of the SNOMED CT components: concepts, descriptions, and relationships. All components of SNOMED CT have unique and permanent identifiers that are reflected in their corresponding files. Likewise, there are also individual files with specific fields for each refset. The RF2 format is also used for SNOMED CT extensions.

The three distributions released by SNOMED CT are:

- *Full Release*: contains all the versions of all the components to date in the form of a history, so that the status of SNOMED CT can be seen at a specific time.
- *Delta Release*: contains only the changes (creations and deactivations) compared to the previous version.
- *Snapshot Release*: contains the latest version of all components.

## 2.6.2 Logical model

The SNOMED CT logical model defines the components and the structure that represent its content, specifically: *concepts*, *descriptions* and *relationships* (see Figure 16).

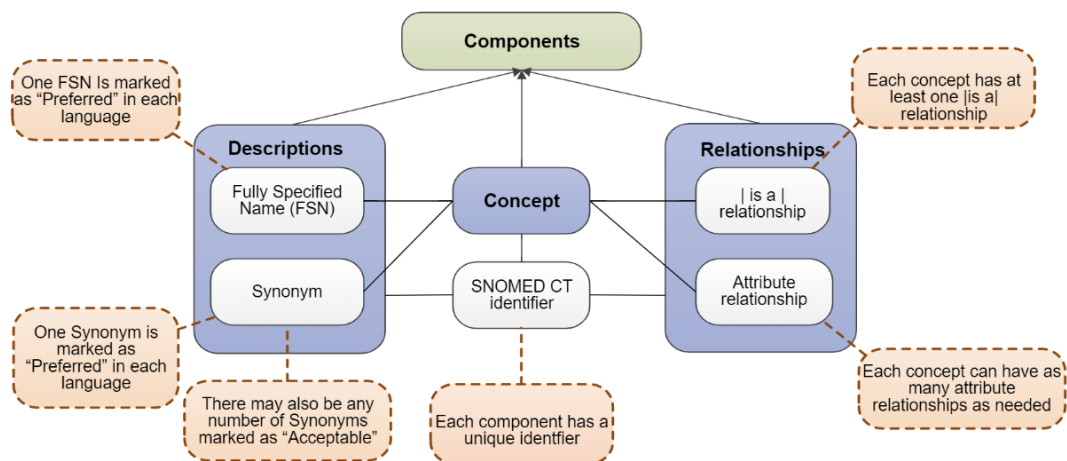


Figure 16. Overview of the SNOMED CT logical model<sup>9</sup>

<sup>9</sup> <https://confluence.ihtsdotools.org/display/DOCSTART/5.+SNOMED+CT+Logical+Model>

A SNOMED CT concept is a numerical identifier that represents a unique clinical meaning. For example, 277170006, 217082002, 31978002, 43706004 ...

A description is a text that is assigned to a concept to be human readable. For example, 277170006 |Edema of ear canal|, 217082002 |Accidental fall|, 31978002 |Fracture of tibia| or 43706004 |Ascorbic acid|. Each concept is assigned to a complete description that contains a semantic tag at the end of the description enclosed in parentheses (i.e., Fully Specified Name - FSN) and several synonyms (one preferred and the rest acceptable). Likewise, the descriptions themselves also have an associated identifier (see Figure 17).

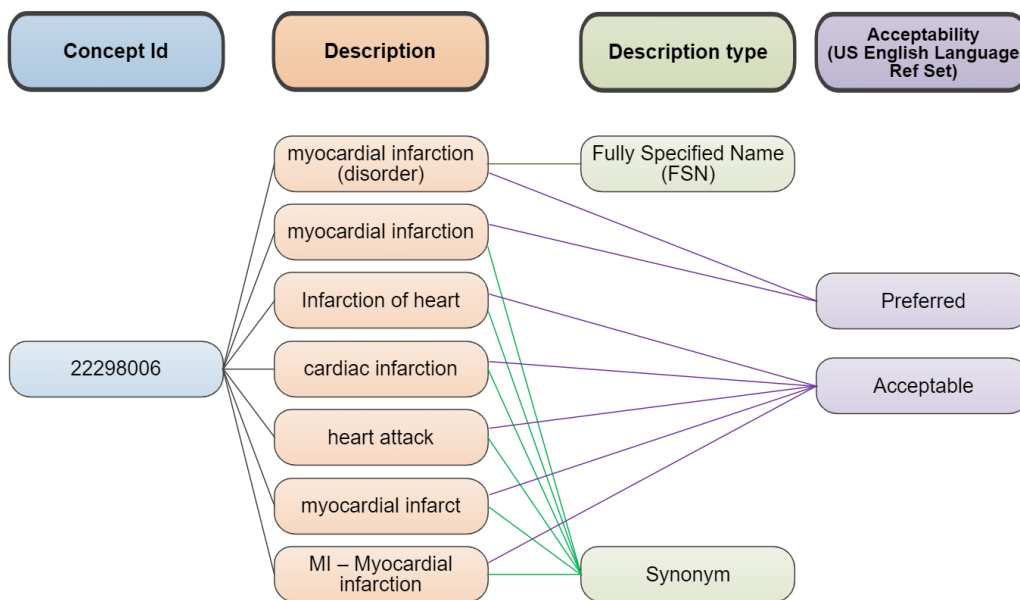


Figure 17. Examples of descriptions for the concept 22298006 |Myocardial infarction|<sup>10</sup>

A relationship is an association between two concepts. There are two types of relationships: *subtype* (i.e., *IS A*) and *attribute relationships*. *IS A* relationships define hierarchies, i.e., specializations. In SNOMED CT a concept can be a specialization of more than one concept (i.e., poly-hierarchy/multi-parent). We say parent concepts and children concepts. The deeper a concept is in a hierarchy, the greater its level of granularity or

<sup>10</sup> <https://confluence.ihtsdotools.org/display/DOCSTART/5.+SNOMED+CT+Logical+Model>

detail. And the higher it is in the hierarchy, the more general or grouping concept it will be. Attribute relationships are used to define the meaning of concepts by linking concepts from different hierarchies. There are about 120 types of attributes. For example, laterality, severity, has active ingredient, priority, associated morphology, causative agent, finding site, associated with, etc. (see Figure 18 and Figure 19).

Relationships is what makes SNOMED CT so powerful as it can be considered as an ontology. Concepts are not mere lists (as is the case in classifications like ICD-9), but are connected to each other.

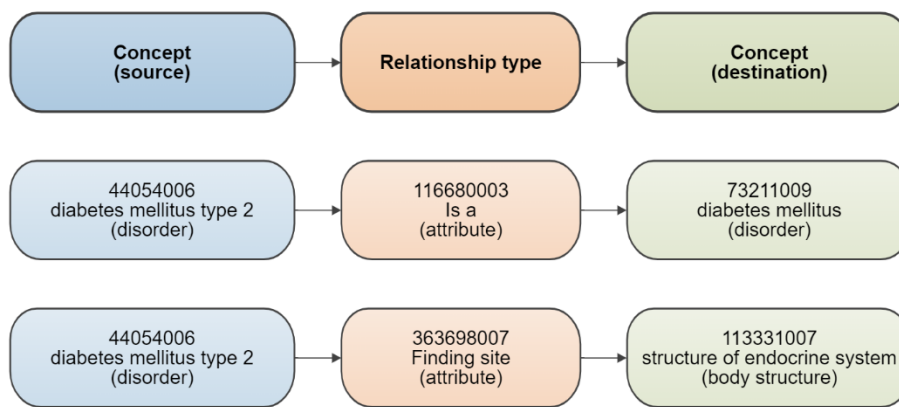


Figure 18. Example of IS A and attribute relationships <sup>11</sup>

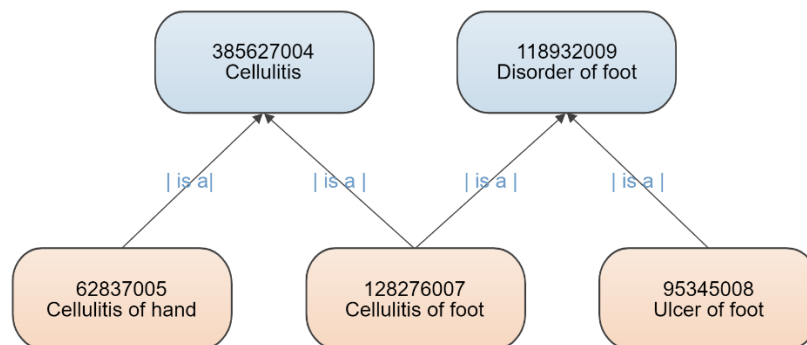


Figure 19. Example of multi-parent IS A relationships <sup>11</sup>

<sup>11</sup> <https://confluence.ihtsdotools.org/display/DOCSTART/5.+SNOMED+CT+Logical+Model>

In SNOMED CT concepts are associated with a *logical definition* comprising by subtype relationships and their attribute relationships. See Figure 20 for an example.

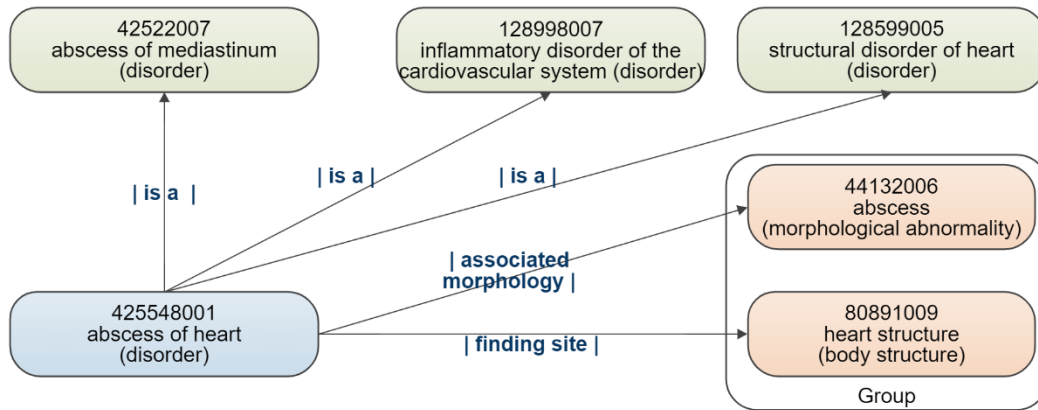


Figure 20. Logical definition of the concept 425548001 |Abscess of heart|<sup>12</sup>

SNOMED CT is, by definition, a *directed, acyclic graph*, i.e., the set of concepts are connected to one another by lines (edges) in which each connection has a specified direction such that no route that follows the direction of the connections enters a loop (see Figure 21).

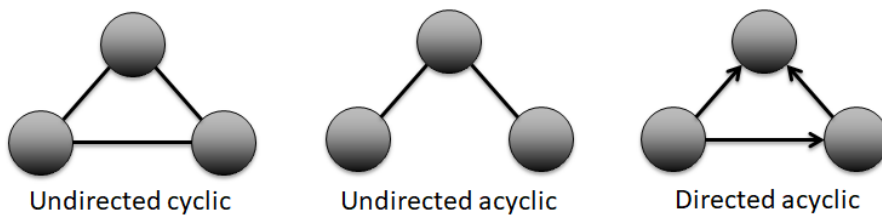


Figure 21. Three types of graphs

SNOMED CT contains more than 350,000 active concepts (more than 480,000 including also inactive concepts), and more than 1,100,000 active relationships, including both IS A and attribute relationships. Concepts are arranged into 19 *top-level hierarchies*, such as:

<sup>12</sup> <https://confluence.ihtsdotools.org/display/DOCSTART/5.+SNOMED+CT+Logical+Model>

procedures, clinical findings, organisms, substances and body structures, among others (see Figure 22). Figure 23 shows a visualization of the SNOMED CT content. This is how SNOMED CT looks like, where black points are the concepts and grey lines are the IS A relationships. Top-level hierarchies correspond to the darkest areas. Figure 24 shows an overview of the components and associated information of SNOMED CT.

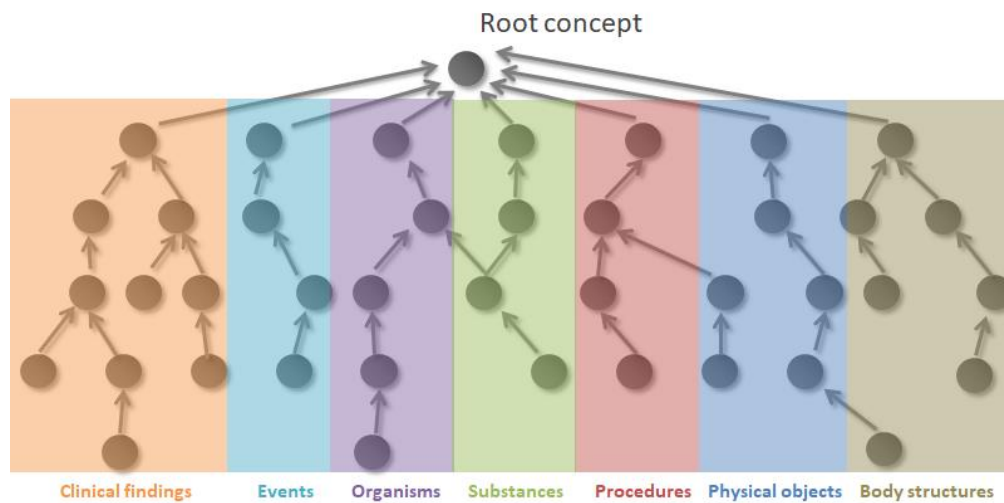


Figure 22. Abstract representation of some SNOMED CT top-level hierarchies

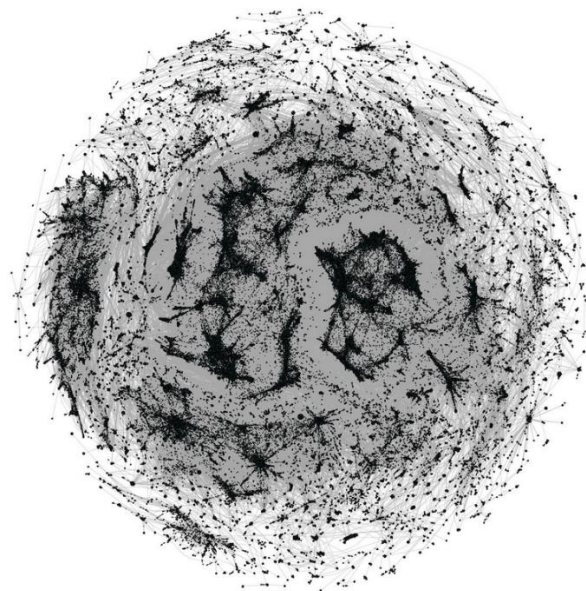


Figure 23. Visualization of the relationships between clinical concepts in SNOMED CT <sup>13</sup>

<sup>13</sup>[https://www.reddit.com/r/dataisbeautiful/comments/1zas7r/visualization\\_of\\_the\\_relationships\\_between/](https://www.reddit.com/r/dataisbeautiful/comments/1zas7r/visualization_of_the_relationships_between/)

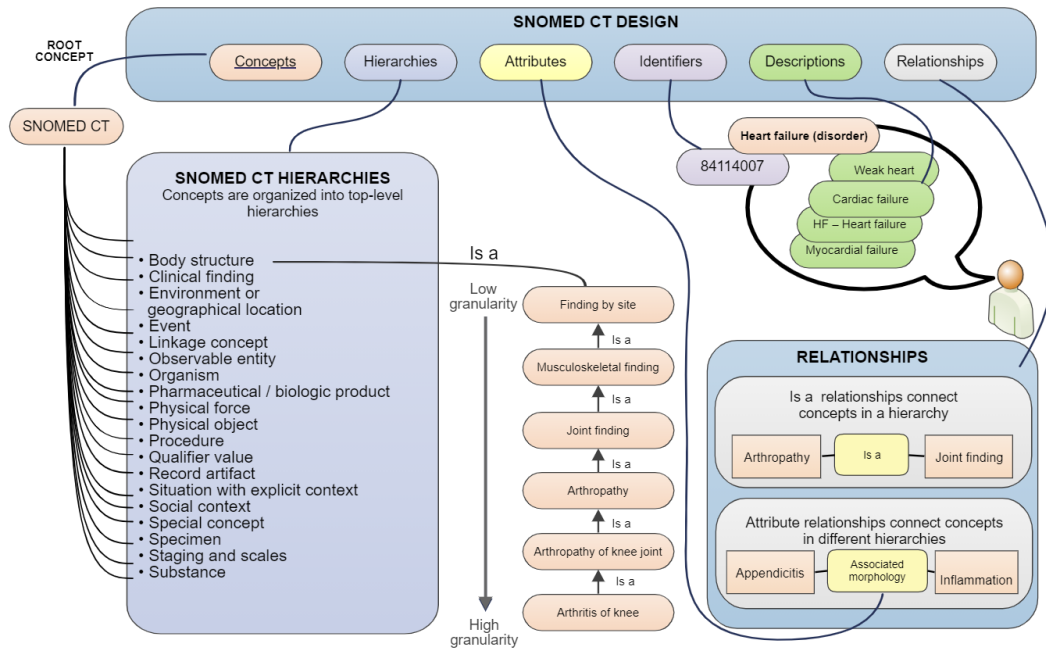


Figure 24. SNOMED CT overview scheme with examples<sup>14</sup>

Considering its state of definition (i.e., its defining characteristics), there are two types of concepts in SNOMED CT: *primitive* concepts and *fully defined* concepts.

A concept is fully defined when its defining characteristics are sufficient to distinguish its meaning from other similar concepts. For example, Figure 25 shows the logical definition of the fully defined concept 2704003 |Acute disease|<sup>15</sup>.

<sup>14</sup> <https://confluence.ihtsdotools.org/display/DOCSTART/4.+SNOMED+CT+Basics>

<sup>15</sup> For more information on SNOMED CT diagrammatic notation: <https://confluence.ihtsdotools.org/display/DOCDIAG>

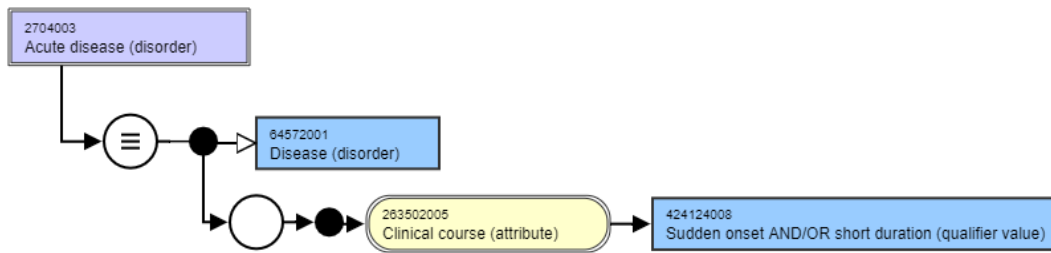


Figure 25. Logical definition of the concept 2704003 |Acute disease|

Any SNOMED CT concept that includes these relationships in its logical definition is a descendant (i.e., subtype) of 2704003 |Acute disease|.

In contrast to fully defined concepts are primitive concepts, whose defining characteristics are not sufficient to distinguish their meaning from other concepts. For example, the concepts 64572001 |Disease| and 69449002 |Drug action| are primitive concepts, as shown in Figure 26 and Figure 27, respectively.

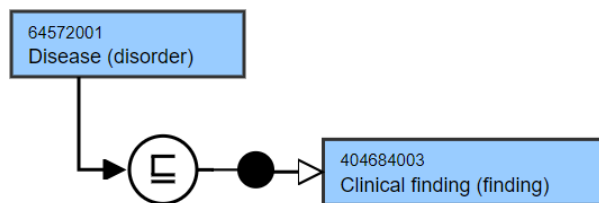


Figure 26. Logical definition of the concept 64572001 |Disease|

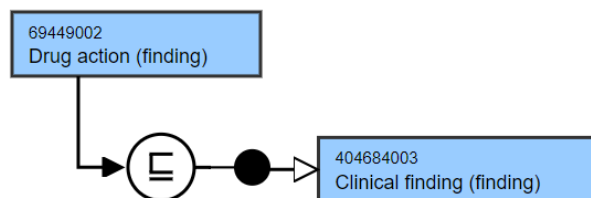


Figure 27. Logical definition of the concept 69449002 |Drug action|

### 2.6.3 Concept model

The *SNOMED CT Concept Model* is a set of rules that specify how two concepts of two different hierarchies can be related through attribute relationships. In other words, the concept model establishes how concepts are related in the terminology. The concept model defines the *domain* (i.e., the source concepts of the relationship), the *attribute*, and the *range* (i.e., the destination concept/s of the relationship). The domain is the hierarchy or hierarchies where a certain attribute can be applied. On the other hand, the range is the concept, hierarchy or hierarchies allowed as values for a certain attribute. The concept model is the mechanism that specifies the semantics of SNOMED CT. Figure 28 and Figure 29 show some examples.

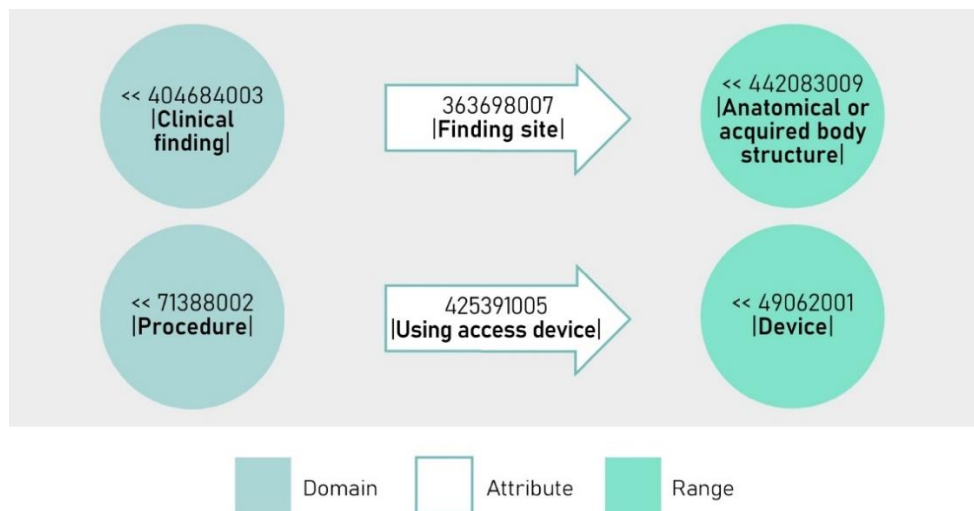


Figure 28. Examples of the domain and range specified for the attributes 363698007 |Finding site| and 425391005 |Using access device (attribute)|



DOMAIN	ATTRIBUTE	RANGE
Clinical finding <i>Aortitis</i>	Associated morphology →	Morphologic abnormality <i>Inflammatory</i>
Clinical finding <i>Hookworm infection</i>	Associated with →	Organism <i>Uncinaria</i>
Clinical finding <i>Skin graft hematoma</i>	Associated with →	Procedure <i>Transplantation</i>
Clinical finding <i>Famotidine overdose</i>	Causative agent →	Substance <i>Famotidine</i>
Clinical finding <i>Solar comedone</i>	Causative agent →	Physical force <i>Sun</i>

Figure 29. Examples of attribute relationships according to the concept model

The *Machine Readable Concept Model* (MRCM) represents the rules of the concept model in a way that can be read by a computer and applied to assure that the definitions of the concepts and expressions comply with these rules. The MRCM can be used for a variety of purposes, including the creation and validation of SNOMED CT concepts, expressions, queries and ECs, natural language processing (NLP), and terminology binding to information models to support queries and interoperability. The MRCM has recently been developed by SNOMED International using ECL, which defines the subsets that can be used as the domain and range of a given attribute relationship.

#### 2.6.4 Expressions

A SNOMEDCT *expression* is a set of concepts that, syntactically combined, represent a certain clinical meaning at the required level of detail. There are two types of SNOMED CT expressions: on the one hand, pre-coordinated expressions, made up of a single identifier (i.e., a concept) and, on the other hand, post-coordinated expressions, made up of more than one identifier. Both type of expressions are defined syntactically by means of the SNOMED CT Compositional Grammar [24].

For example, the concept 174041007 |Laparoscopic emergency appendectomy| is a pre-coordinated expression, whose logical definition is presented in Figure 30.

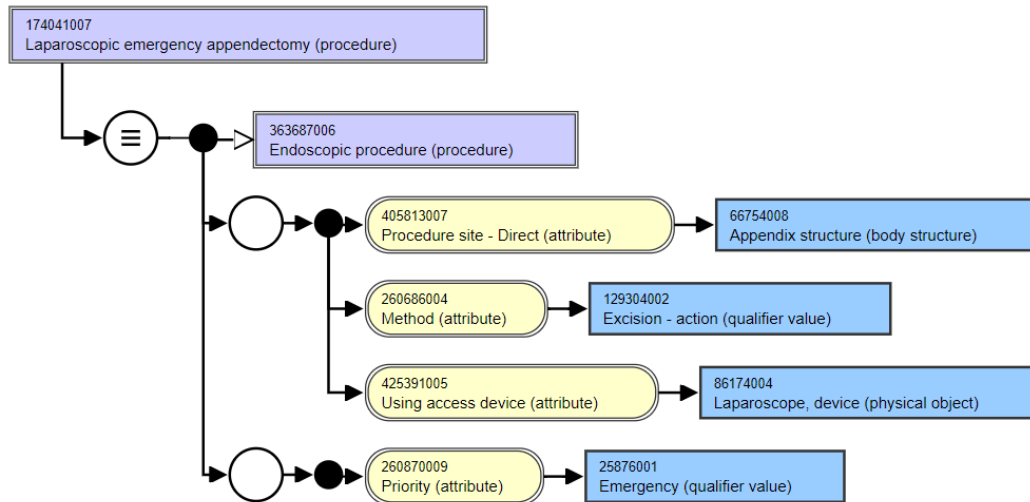


Figure 30. Logic definition of the concept 174041007 |Laparoscopic emergency appendectomy|

The pre-coordinated concept 174041007 |Laparoscopic emergency appendectomy| has the desired level of detail. But what if this concept did not exist in SNOMED CT? We could build it thanks to the post-coordination mechanism. To post-coordinate it is necessary to consider the rules of the SNOMED CT concept model so that the expression makes sense (i.e., it is semantically valid). The following would be the equivalent post-coordinated expression:

80146002 |Excision of appendix (procedure)|:  
 260870009 |Priority (attribute)| = 25876001 |Emergency (qualifier value)|,  
 425391005 |Using access device (attribute)| = 86174004 |Laparoscope, device|

It should be noted that post-coordinated expressions are defined by using the SNOMED CT Compositional Grammar. The previous post-coordinated expression defines an appendectomy whose priority is emergency and whose access device is a laparoscope.

The following more complex example also defines a post-coordinated expression. Specifically, it is a skin burn with an associated morphology of third-degree burn caused by hot water located in the index finger of the left hand:

284196006  Burn of skin :
116676008  Associated morphology  = 80247002  Third degree burn injury ,
246075003  Causative agent  = 47448006  Hot water ,
363698007  Finding site  = 83738005  Index finger ,
272741003  Laterality  = 7771000  Left

## 2.7 SNOMED CT EXPRESSION CONSTRAINT LANGUAGE

ECL is a development of SNOMED International that enables the intensional definition of sets of clinical meanings by defining ECs.

ECs are built by adding constraint and set operators to expressions to define subsets of concepts instead of just one clinical meaning (like in post-coordination) according to the SNOMED CT Concept Model. These subsets are useful in value binding between information models, such as archetypes, and SNOMED CT, in order to obtain high levels of semantic interoperability (see Figure 31).

ECL is a formal syntax for representing SNOMED CT ECs. ECs are computable rules for defining bounded subsets of clinical meanings represented by both pre-coordinated or post-coordinated expressions. ECs can be used to constrain the valid values of a data item in an EHR. They also can be used as the intensional definition of a set of concept references, as a computer processable query that identifies a set of expressions, or as a constraint that restricts the domain and range of an attribute defined in the SNOMED CT Concept Model.

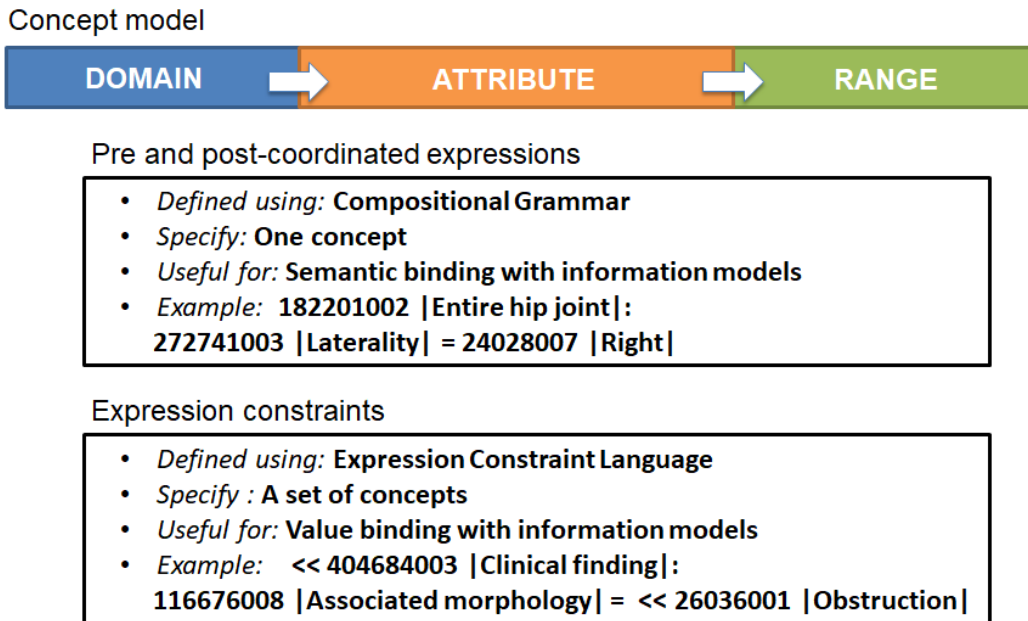


Figure 31. SNOMED CT concept model, expressions and ECs summary

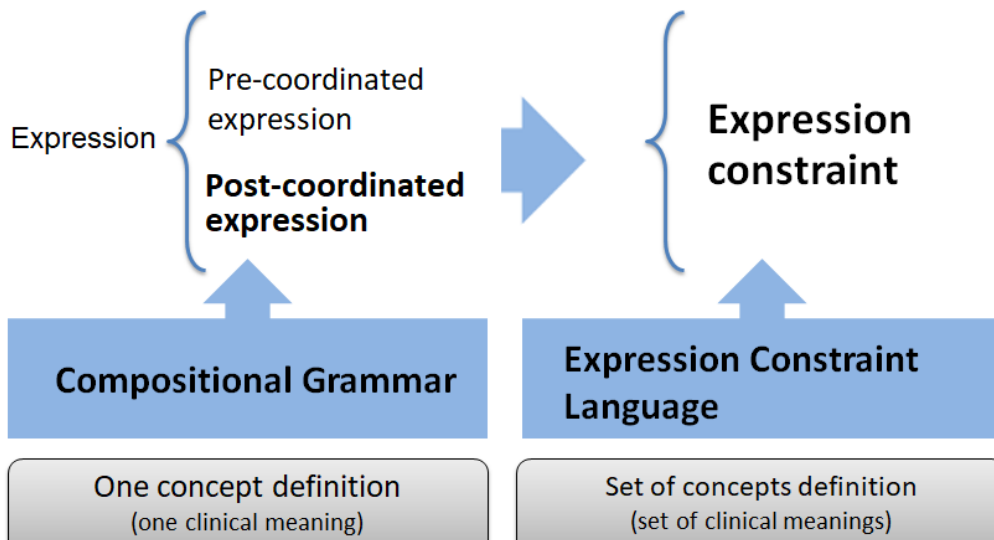


Figure 32. Comparison between the SNOMED CT Compositional Grammar and ECL

The SNOMED CT ECL must support the capabilities listed in Table 2.

Capability	Description
Concept reference	Ability to reference pre-coordinated SNOMED CT concepts using their identifier and an optional natural language description.
Concept hierarchies	Ability to select a set of concepts, such as the descendants of a concept, the descendants and the concept itself, the ascendants, and the ascendants and the concept itself.
Parents and children	Ability to select a set of concepts, such as parents and children.
Conjunction	Ability to connect two ECs, attribute groups or attribute sets by using the logical AND operator.
Disjunction	Ability to connect two ECs, attribute groups or attribute sets by using the logical OR operator.
Refinement	Ability to refine (i.e., filter) the meaning of an EC through one or more attribute values.
Reverse	Ability to constrain the source concepts of a set of relationships and refer to the target concepts of such relationships.
Dotted attribute	Ability to refer the value (or set of values) of an attribute that is included in the definition of a set of concepts.
Attribute group	Ability to group a collection of attributes that work together as part of a refinement.
Attribute	Ability to specify an attribute name-attribute value pair that refines the meaning of the expressions resulting from the constraint.
Nesting	Ability to use an EC to represent the valid set of focus concepts or attribute values.
Attribute values comparator	Ability to compare the attribute value of the resulting expressions with the attribute value in the EC using comparison operators (i.e., =, <>, !=).

Member of	Ability to select a set of concepts that are referenced by the members of a reference set.
Exclusion	Ability to filter a set of expressions from the result by either removing expressions whose focus concept is on a specific set, or removing expressions whose attribute value is equal to a certain value.
Any	Ability to reference any concept of the substrate, without depending on the availability of a root concept.

Table 2. Capabilities required by ECL

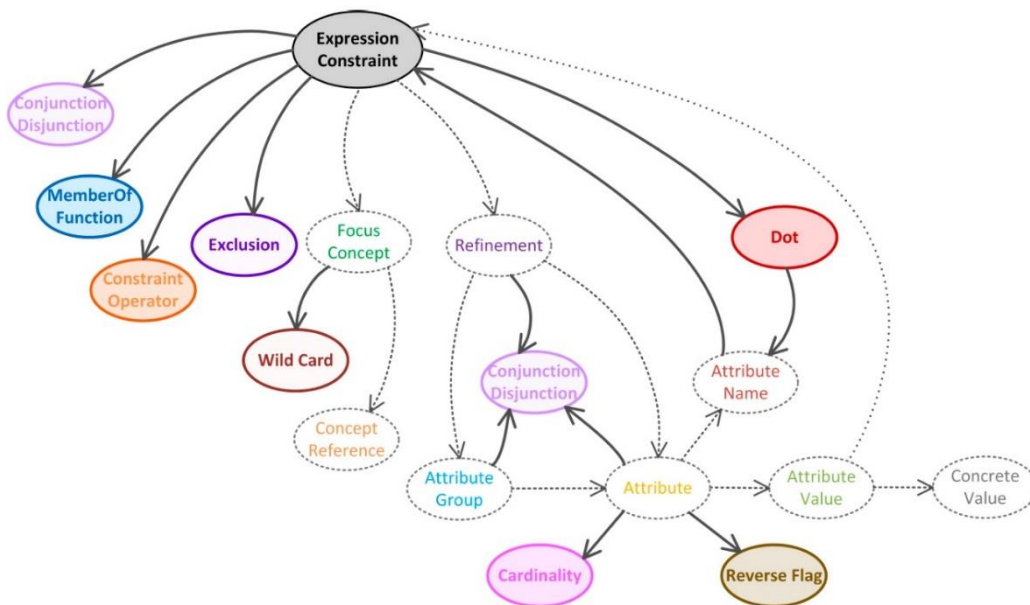


Figure 33. Abstract model of a SNOMED CT EC <sup>16</sup>

<sup>16</sup> <https://confluence.ihtsdotools.org/display/DOECL/4.+Logical+Model>

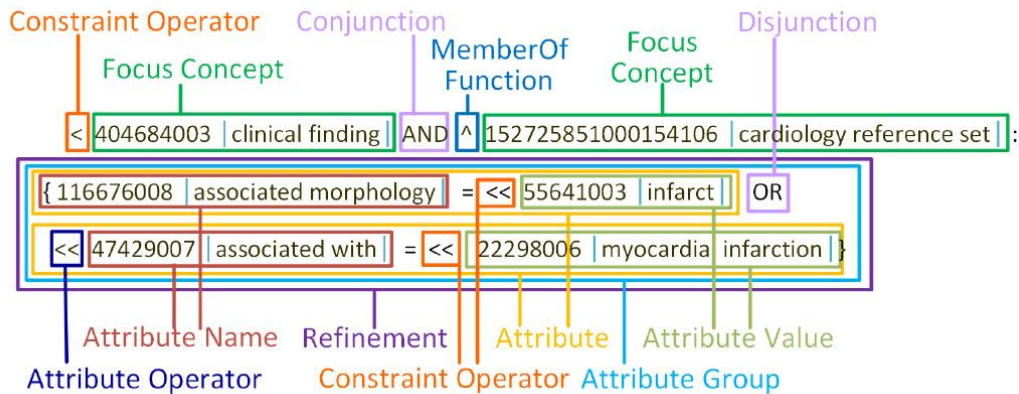


Figure 34. SNOMED CT EC components <sup>16</sup>

ECL presents two logically equivalent syntaxes: brief syntax and long syntax. The brief syntax, which is considered the normative syntax and whose use is recommended in interoperable communications, uses a set of symbols with the objective of being as compact as possible. The long syntax, whose aim is to increase the human readability of the ECL, replaces the set of symbols of the brief syntax by English-based texts [5]. There are three types of ECs: simple, refined and compound. They are explained below using both brief and long syntaxes.

### 2.7.1 Simple ECs

To define a simple EC, a constraint operator is applied to a focus concept, which is composed by a numerical identifier and, optionally, a description enclosed by a pair of “|” characters. Constraint operators traverse the hierarchical relationships (i.e., “116680003 |Is a (attribute)|” relationship) to return the set of concepts that are connected (either directly or transitively) to the focus concept. There are different operators traversing the hierarchy in diverse ways. Constraint operators are presented in Table 3.

Brief syntax	Long syntax	Description
<!	childOf	Children
<<!	childOrSelfOf	Concept itself and children
<	descendantOf	Descendants
<<	descendantOrSelfOf	Concept itself and descendants
>!	parentOf	Parents
>>!	parentOrSelfOf	Concept itself and parents
>	ancestorOf	Ascendants
>>	ancestorOrSelfOf	Concept itself and ascendants
^	memberOf	Members of a reference set

Table 3. ECL constraint operators

Note that the term “descendants” can be understood as “types” or “subtypes”, and “ascendants” or “ancestors” as “supertypes”. Furthermore, it is also possible to reference all the concepts in the given substrate with the wildcard symbol (i.e., “\*”) or with “any”. As an example, EC example 1 defines the subset of all types of disorders of blood vessel, including the concept “27550009 |Disorder of blood vessel (disorder)|” itself.

EC example 1

<< 27550009  Disorder of blood vessel (disorder)
--

### 2.7.2 Refined ECs

Refinements allow filtering matching concepts by adding conditions in form of attribute relationships. Only those concepts whose defining attributes satisfy the given refinements are selected. As with the SNOMED CT Compositional Grammar (SCG) [24], refinements are placed after a “:” symbol. For example, EC example 2 defines the subset of disorders of body system located in the blood vessel structure.



*EC example 2*

```
< 362965005 |Disorder of body system (disorder)|:
  363698007 |Finding site (attribute)| = << 59820001 |Blood vessel structure (body structure)|
```

Multiple attributes can be defined in an EC. They can be combined using logical operators (i.e., ‘AND’ -or comma- and ‘OR’), grouped, or both. As in SCG, EC uses curly braces to define an attribute group, i.e., a set of attributes that should be grouped. For instance, EC example 3 represents the subset of clinical findings whose associated morphologies are a stenosis located in the pulmonary valve structure and a congenital septal defect located in the interatrial septum structure.

*EC example 3*

```
< 404684003 |Clinical finding (finding)|:
  {363698007 |Finding site (attribute)| = << 39057004 |Pulmonary valve|,
  116676008 |Associated morphology (attribute)| = << 415582006 |Stenosis|},
  {363698007 |Finding site (attribute)| = << 58095006 |Interatrial septum structure|,
  116676008 |Associated morphology| = << 396351009 |Congenital septal defect|}
```

Attribute refinements may be preceded by a cardinality (i.e., “[min..max]” or “[min to max]”) that represents a constraint on the minimum and maximum number of times that the given attribute may appear in a matching clinical meaning (default “[1..\*]”). For example, EC example 4 is satisfied only by those allergy conditions that are caused by two or more substances.

*EC example 4*

```
< 473011001 |Allergic condition (disorder)|:
  [2..*] 246075003 |Causative agent (attribute)| = < 105590001 |Substance
  (substance)|
```

It is important to mention that only non-redundant attribute relationships are considered in the cardinality count. Two attribute relationships are redundant if they meet any of the next two cases:

- 1) For the same attribute, attribute values have a subsumption relationship. For example, in the following attribute relationships: “246075003 |Causative agent (attribute)| = 373265006 |Analgesic (substance)|” and “246075003 |Causative agent (attribute)| = 387494007 |Codeine (substance)|”, the first one is redundant since both use a “246075003 |Causative agent (attribute)|” attribute and “387494007 |Codeine (substance)|” is a subtype of “373265006 |Analgesic (substance)|”.
  
- 2) Both attributes and attribute values have a subsumption relationship. For example, in “47429007 |Associated with (attribute)| = 373265006 |Analgesic (substance)|” and “246075003 |Causative agent (attribute)| = 387494007 |Codeine (substance)|”, the first attribute relationship is redundant since the “246075003 |Causative agent (attribute)|” attribute is a subtype of the “47429007 |Associated with (attribute)|” attribute and “387494007 |Codeine (substance)|” is a subtype of “373265006 |Analgesic (substance)|”.

Note that a relationship that is part of a relationship group is only regarded as redundant if the relationship group as a whole subsumes another relationship group [25].

Sometimes, it may be necessary to select the destination concept of a relationship and constrain the source concept to a given value. To achieve this, it is possible to add a reverse flag (i.e., “R” or “reverseOf”) that makes the matching relationships to be traversed in the reverse of the normal direction (reversed attributes). For example, EC example 5 represents the subset of foods that cause any type of allergic condition.

*EC example 5*

< 255620007  Food (substance) : R 246075003  Causative agent (attribute)  = < 473011001  Allergic condition (disorder)
---

### 2.7.3 Compound ECs

Compound ECs can be defined by using binary operators between simple, refined or nested compound ECs. ECL binary operators are presented in Table 4.

Brief and long syntax	Description
AND (or comma)	Conjunction
OR	Disjunction
MINUS	Exclusion

Table 4. ECL binary operators

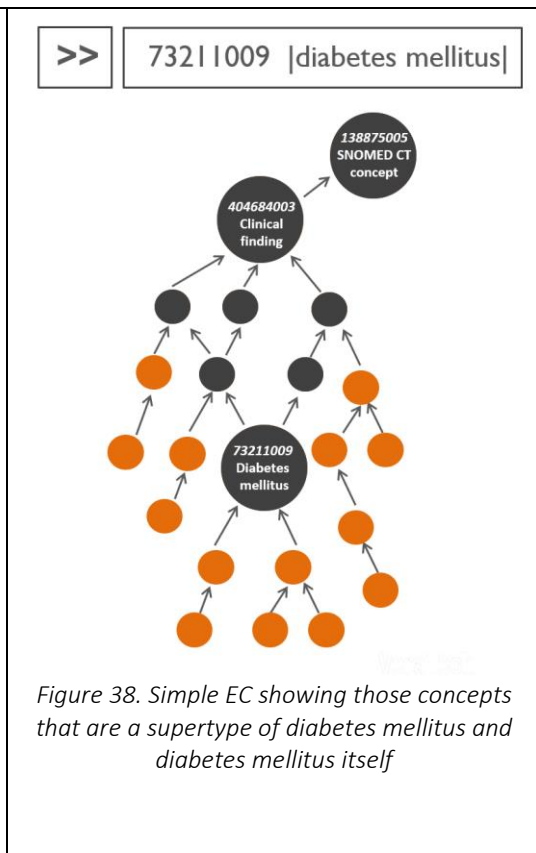
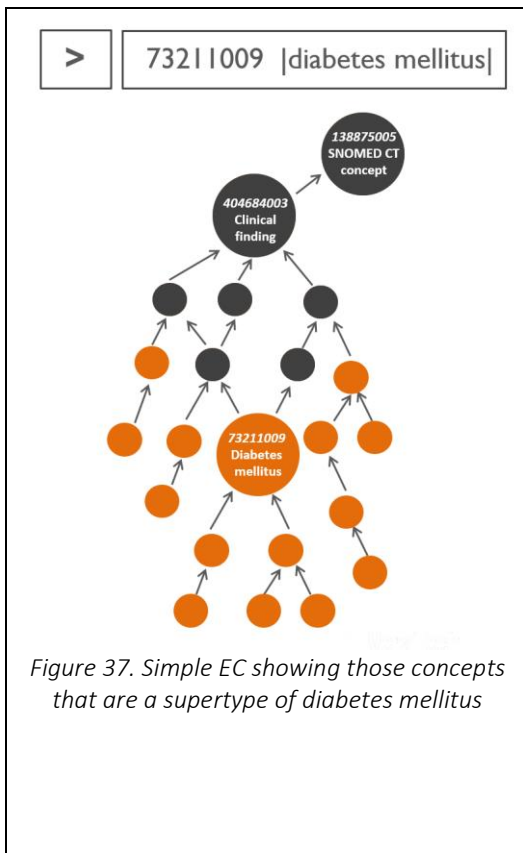
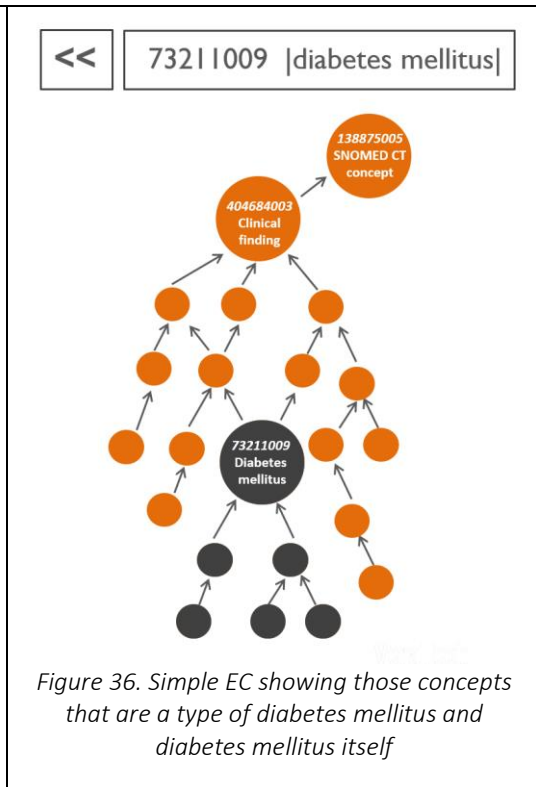
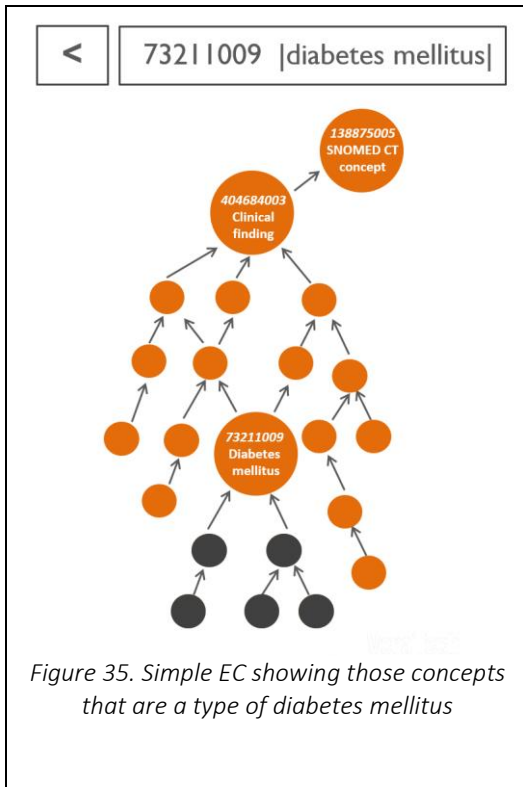
For instance, EC example 6 defines the subset that contains the tetanus, diphtheria and pertussis vaccines.

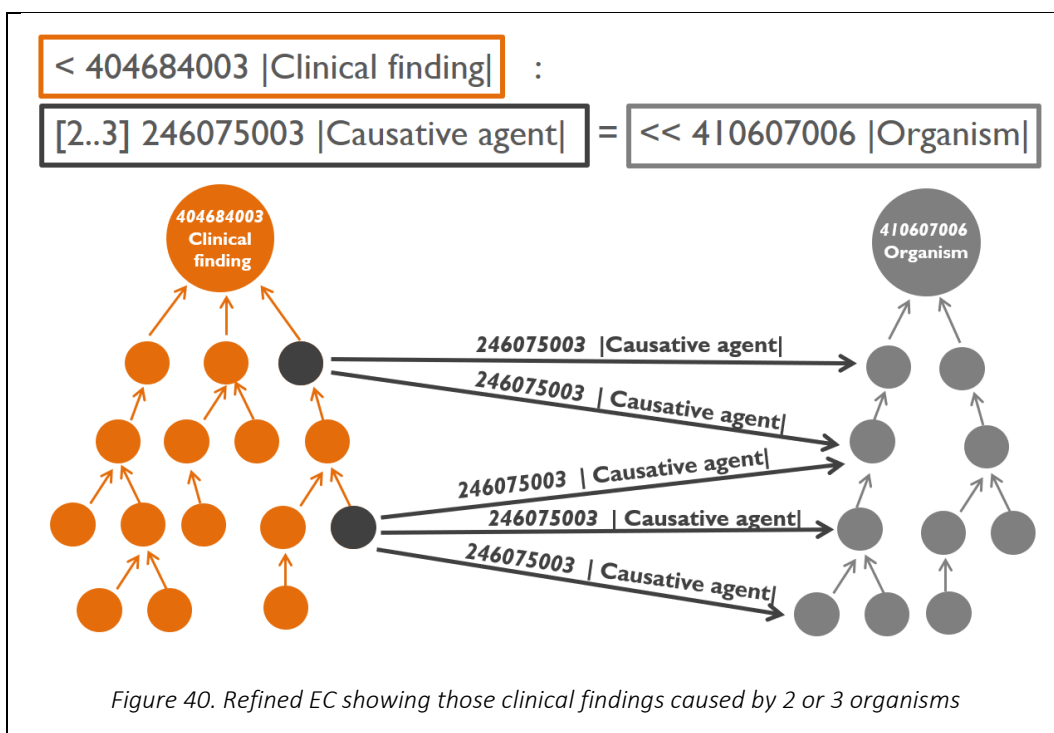
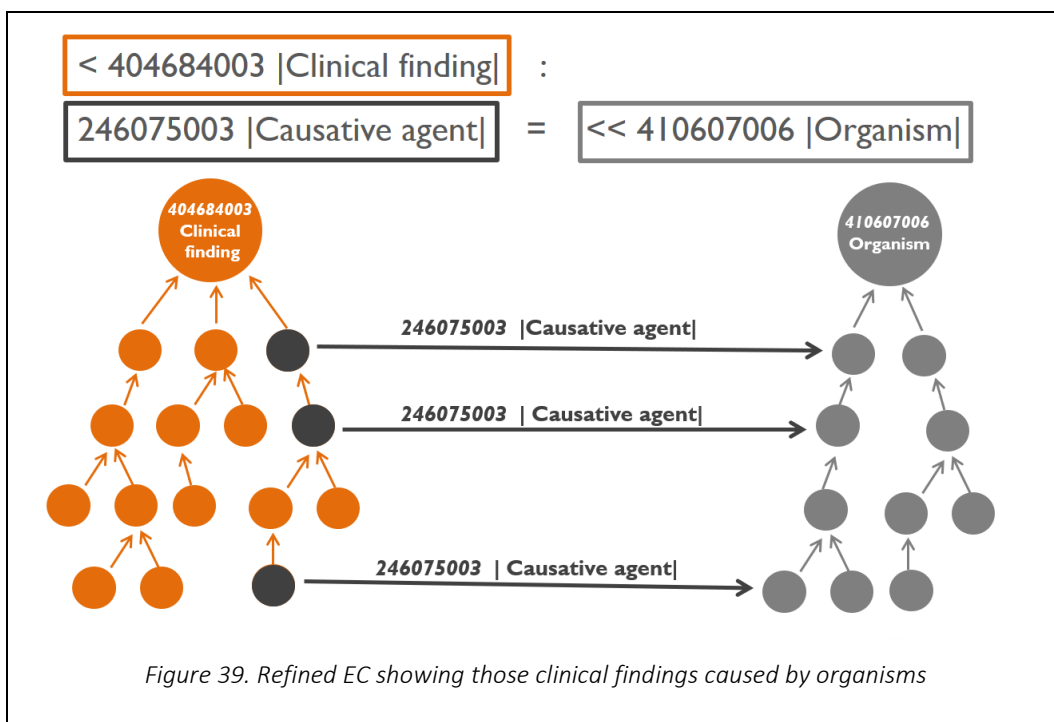
*EC example 6*

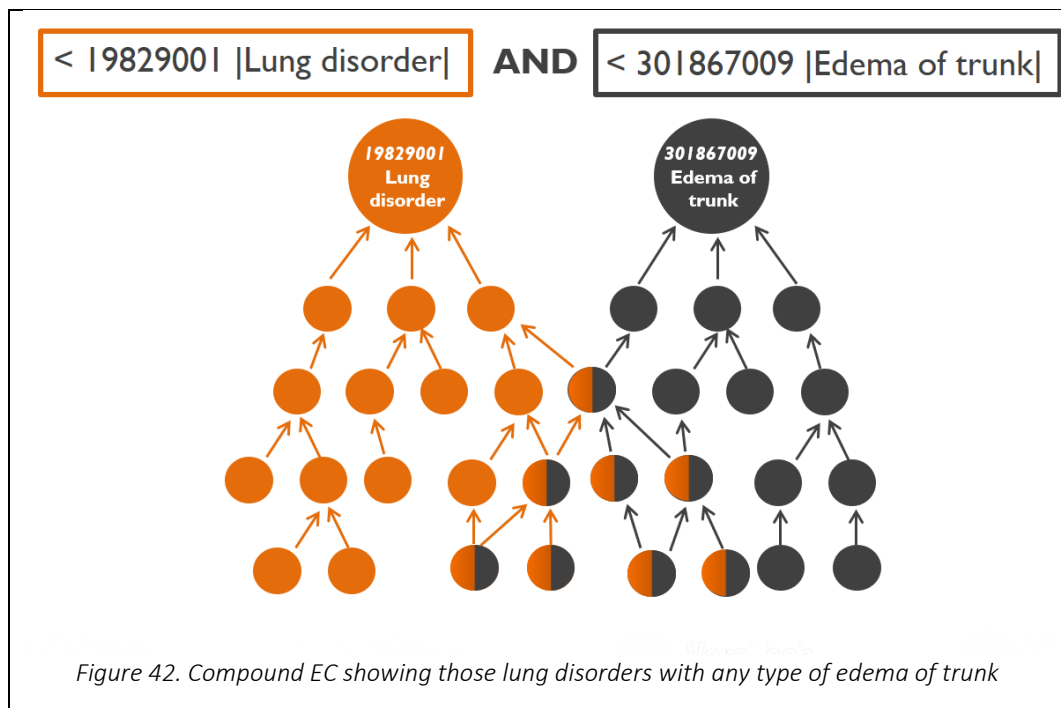
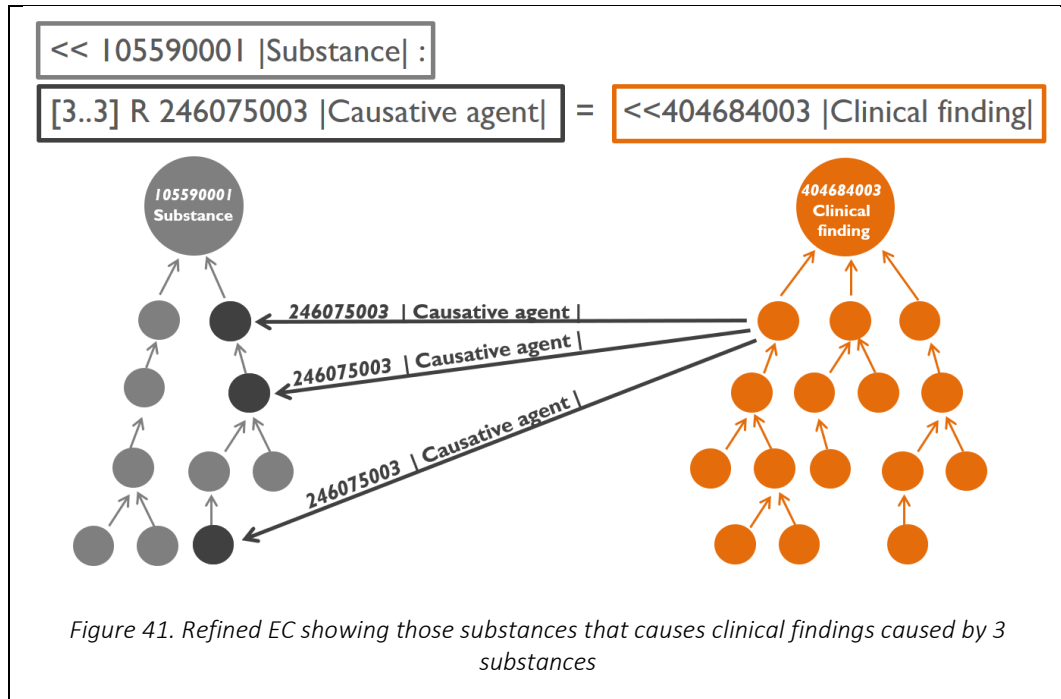
871742003 |Tetanus vaccine (medicinal product)| OR 871729003 |Diphtheria vaccine (medicinal product)| OR 871758000 |Pertussis vaccine (medicinal product)|

### 2.7.4 Graphical representation of ECs

We show some additional examples of ECs and its corresponding graphical representation in Figure 35.







## 2.8 OPENEHR EXPRESSION LANGUAGE

Semantics of openEHR computable expressions are defined by the Expression Object Model (EOM). These computable expressions can be used in healthcare and life sciences in those scenarios where it is required to specify rules and expressions on data. The model is designed as an extensible core formalism and therefore it can be used in other formalisms, such as archetypes.

EL is an abstract syntax counterpart to the EOM, and may be considered a default syntax, although it is possible to define other syntaxes. The first use of EL is to specify and explain the semantics of the EOM, and the second one is to author expressions and rules by means of text.

EL formalism requires extensions for use, such as operators, functions, leaf types, and other features that are needed in specific contexts. The semantic requirements are for expressions, including arithmetic, boolean, and relational operators, functions, quantifier operators, operator precedence, parentheses, constant values, and certain kinds of variables. However, there is no support in the core specification for procedural semantics or most of the other complexities of full-blown programming languages.

One use of the EL language is for writing boolean expressions inside a computational context. It is expected to include a multi-section self-standing *EL text* with strong similarities with archetypes.

The primitive data types of EL include the most common ones, such as Boolean, Integer, Real, and so on. It also includes container types, such as List, Set, and Hash. It also includes quantification operators, which can make use of container types and an Interval type, which is the same as in the openEHR Foundation Types.

An *EL text* is a series of statements. They can be either declarations, assignments or assertions. Variables read the values from the data context. EL allows either bounding and local variables to be used without restrictions. However, if a bound variable cannot be evaluated from the data context, it should be thrown an exception indicating that such variable cannot be evaluated.

The terminal entities of EL include constants, variables, literals, function calls and raw paths, among others. On the other hand, non-terminal entities include operators and functions. EL allows using common date and time functions, as well as aggregate functions. It supports three groups of operators: arithmetic, which include subtraction, addition, multiplication and the most common; comparison operators, such as equal, distinct, and greater than; and boolean operators.



## CHAPTER 3. METHODS FOR THE DEFINITION, VALIDATION, EXECUTION AND VISUALIZATION OF SNOMED CT SUBSETS USING ECL

---

### 3.1 INTRODUCTION

SNOMED CT is the most comprehensive, multilingual and codified clinical healthcare terminology in the world [4]. Each SNOMED CT concept has a formal logic definition represented by a set of defining relationships to other concepts. These relationships are governed by the SNOMED CT Concept Model, which is a set of rules that determine the ways in which concepts are permitted to be modelled [4]. The MRCM represents these rules in a form that can be read by a computer [26].

SNOMED CT ECL is a declarative language developed by SNOMED International for the definition of SNOMED CT ECs. ECs are executable expressions that define bounded sets of clinical meanings. In ECL, these sets are defined by stating constraints over the logic definition of concepts. The execution of the constraints on some substrate (typically a SNOMED CT edition) yields the intended subset, i.e., the set of clinical meanings that satisfies the constraints.

Intensional definition of subsets in general, and ECs in particular, are important in several use cases. They are used in terminology binding between clinical information models and terminologies for defining the set of valid values of codified data [27]. They are also useful to define intensional refsets, such as language preferences or map refsets. They provide a means to query SNOMED CT content, including the content of a SNOMED CT edition or SNOMED CT expressions stored within an EHR. Additionally, ECs are used in the definition of the MRCM.

The evaluation of ECL rules requires an execution engine able to receive an EC as input, parse it, generate an execution plan, execute the plan against a given SNOMED CT substrate and return the matching concepts or expressions to the client. Additionally, although not mandatory, it is recommended to check ECs for conformance against the

MRCM prior to execution [5] and to provide methods for representing, understanding and validating the resulting subsets.

Potential users of such engine include SNOMED CT designers and developers of information models, data entry interfaces, storage systems, decision support systems, retrieval and analysis systems, communication standards and terminology services. An EC engine is also useful for SNOMED CT terminology developers, such as concept model designers, content authors, map developers, subset and constraint developers, and release process managers [5].

One of the key aspects when developing an EC execution engine is how to represent SNOMED CT in a database in a way that allows the execution of the complete set of EC clauses efficiently in terms of execution time. Due to the intrinsic nature of SNOMED CT, a directed and acyclic graph, storing its content in a graph database seems to be a logical choice [28,29]. A contribution of this chapter is the analysis and use of a graph database [30] to store and manage the SNOMED CT content, and to simplify and execute ECs. For this purpose, we use Neo4j, a highly scalable native graph database, purpose-built to leverage data and relationships, which is available in an open source community edition [31]. Specifically, we address three important challenges: representation of SNOMED CT content, simplification and execution of ECs, and rendering of the subsets in order to understand and validate them. All these methods have been implemented in a publicly accessible EC execution platform called SNQuery<sup>17</sup> [32].

## **3.2 STORAGE OF THE SNOMED CT DATABASE**

Since SNOMED CT is a directed and acyclic graph, the use of a graph database seems to be a logical choice for the representation of the SNOMED CT content. In this regard, graph databases bring several potential advantages over traditional database systems, such as relational [33]. Graph databases emphasize connectedness of data that fits the polyhierarchical and ontological nature of SNOMED CT. They provide a great flexibility to add new nodes and relationships to the graph without affecting existing queries [30].

---

<sup>17</sup> <https://snquery.veratech.es>

Finally, the structurally similar data model facilitates the execution of complex queries that go beyond subsumption queries (i.e., the descendants of a given concept) as required by ECL.

For the representation of SNOMED CT, we have used a graph database model where the schema and instances are represented as graphs and the data manipulation is expressed by graph-oriented operations. Concretely, we have used the Property Graph Data Model [34,35]. Currently, it is the most common graph data model in industry and is gaining prevalence in academy [36]. A property graph is a directed labelled multigraph where nodes and edges could be associated with any number of attributes (also called properties) in the form of key-value pairs. From a data modelling point of view, nodes represent entities (for instance, SNOMED CT concepts), edges represent the relationships between the entities, and properties represent specific characteristics (i.e., attributes) of an entity or relationship.

We have used Neo4j, a graph-oriented database software, for the persistence of SNOMED CT. Neo4j is a highly scalable native property graph database, purpose-built to leverage data and data relationships, which is available in an open source community edition. It comes with a powerful query language called Cypher [36,37].

SNOMED CT is stored as follows. SNOMED CT concepts are represented by graph nodes labelled with a “Concept” label. In Neo4j, labels are used to group nodes into sets where all nodes that have a certain label belong to the same set, in our case the set of concepts. For each concept, we have stored its identifier and Fully Specified Name (FSN) as properties, which are required for query execution and presentation of results. SNOMED CT relationships between concepts are represented by graph edges. Note that relationship types in SNOMED CT are defined as concepts (for instance, the “finding site” attribute relationship is represented by the concept “363698007 |Finding site (attribute)|”, and the “is a” hierarchy relationship by the concept “116680003 |Is a (attribute)|”). We store four types of relationships in the graph:

- 'IS A' to model parent-child concepts association.
- 'RELCL' as the transitive closure of the 'IS A' relationship, i.e., each concept is directly associated with all its ancestors. It has been calculated and stored for efficiency purposes since it speeds up query execution time.
- Attribute relationships by means of the prefix 'REL' followed by the concept identifier. For example, the "clinical course" attribute, which corresponds to the concept "263502005 |Clinical course (attribute)|", is stored in the graph as the relationship type 'REL263502005'<sup>18</sup>. Edges representing attribute relationships have a "relGroup" property containing the group number.
- 'MEMBER OF' relationships to define refsets in the graph. For instance, the concept "840539006 |Disease caused by Severe acute respiratory syndrome coronavirus 2 (disorder)|" has a 'MEMBER OF' relationship to the concept "64401000122105 |SARS-CoV-2 (foundation metadata concept)|", which defines the SARS-CoV-2 refset in the Spanish SNS SNOMED CT extension.

In addition, we have calculated and stored for each concept its minimum depth in the graph (property "minDepth") and the total number of descendants (property "descsNum"), since they are necessary to build the circle packing visualization presented in section 3.5. Figure 43 shows an example of the representation of the SNOMED CT components using the previous graph database modelling.

We have implemented a loading module to populate the graph database from the SNOMED CT Snapshot Release Format 2 (RF2) files (i.e., concepts, relationships and descriptions) into the Neo4j database.

---

<sup>18</sup> Another option we considered, which is discussed in section 4.4, was to define a single 'REL' relationship type and make use of a property to store the concept identifier of the SNOMED CT attribute.

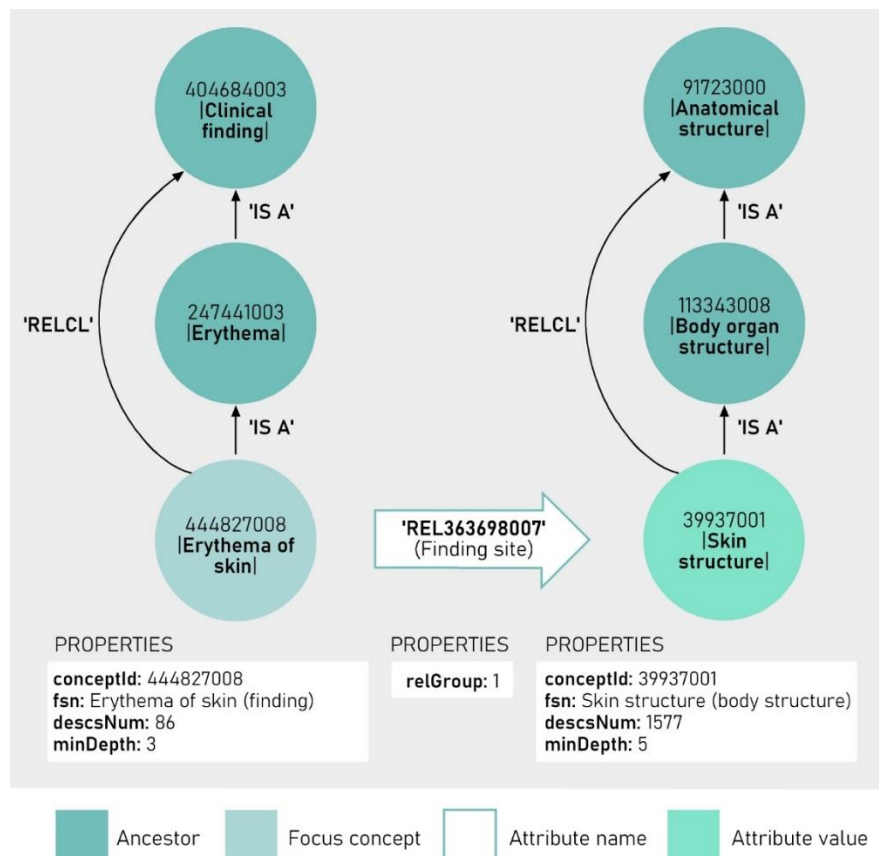


Figure 43. Properties of the concepts “444827008 |Erythema of skin (finding)|” and “39937001 |Skin structure (body structure)|”, relationship “363698007 |Finding site (attribute)|” between them, and relationships with ancestors (SNOMED CT July 2020 International)

### 3.3 EC SIMPLIFICATION

The main idea of the EC simplification is to reduce the complexity of the expression and, in turn, the execution time. It can be achieved by removing redundant concepts, superfluous refinements or by narrowing down the focus concept. It is important to note that the simplification must not affect the completeness of the computed subset of concepts. Note that the result of the simplification depends on the SNOMED CT edition, since editions potentially contain changes from one to another regarding concepts and relationships. Performing such simplification has several advantages. First, it diminishes the number of comparisons to be carried out during query execution. Therefore, it may have impact on

query answering time. Second, the cut-down query is more amenable to human reading. Third, it may help to validate the terminology itself (as discussed in section 4.4).

We have defined three different methods to simplify ECs before they are executed (pre-execution): subsumption-based, MRCM-based, and logic definition-based; and one method based on the mining of the result subset (post-execution). The methods based on the MRCM and the logic definition check whether the EC satisfies the rules defined in the SNOMED CT Concept Model and the logical definition of the involved concepts, respectively. If the EC does not conform to them, it will be regarded as semantically invalid since the EC cannot be satisfied by any concept. It should be noted that the simplification methods should be applied iteratively until the EC cannot be further simplified.

### 3.3.1 Subsumption based

In this section, we provide a method for the elimination of redundant concepts or relationships in ECs to improve the efficiency of EC engines. A concept or a relationship in a given EC is redundant if it can be removed from the EC without affecting the result, i.e., the subset defined. We detect redundant concepts or relationships by analyzing subsumption relationships between concepts. When there is subsumption between the concepts involved in a domain, range or attribute relationships, it may be possible to simplify the EC by applying the following rules (note that they are also valid for the "<" constraint operator):

Let "A", "B" be concepts. Let "r", "s" be attributes.

- 1)  $\ll A = (\ll A \text{ OR } \ll B)$  if "A" is the same as or subsumes "B"
- 2)  $\ll B = (\ll A \text{ AND } \ll B)$  if "A" is the same as or subsumes "B"
- 3)  $(r \ll A) = (r \ll A \text{ OR } s \ll B)$  if "r" is the same as or subsumes "s", "A" is the same as or subsumes "B", and both relationships are within the same group
- 4)  $(s \ll B) = (r \ll A \text{ AND } s \ll B)$  if "r" is the same as or subsumes "s", "A" is the same as or subsumes "B", and both relationships are within the same group

As an example, EC example 7 defines the subset that contains those disorders of body system and disorders of hematopoietic structures that are in any blood vessel structure and in any soft tissues (see Figure 44).

## EC example 7

```

(< 362965005 |Disorder of body system (disorder)| OR
 < 414027002 |Disorder of hematopoietic structure (disorder)|):
 363698007 |Finding site (attribute)| =
 << 59820001 |Blood vessel structure (body structure)|,
 363698007 |Finding site (attribute)| = << 87784001 |Soft tissues (body structure)|

```

We observe that the domain (i.e., “< 362965005 |Disorder of body system (disorder)| OR <414027002 |Disorder of hematopoietic structure (disorder)|”) is a disjunction expression in which the concept “414027002 |Disorder of hematopoietic structure (disorder)|” is subsumed by “< 362965005 |Disorder of body system (disorder)|”. Therefore, “414027002 |Disorder of hematopoietic structure (disorder)|” is redundant. Additionally, the relationship “363698007 |Finding site (attribute)| = << 87784001 |Soft tissues (body structure)|” is redundant because “59820001 |Blood vessel structure (body structure)|” is subsumed by “87784001 |Soft tissues (body structure)|”. Therefore, EC example 7 can be rewritten as EC example 2 (see Figure 45).

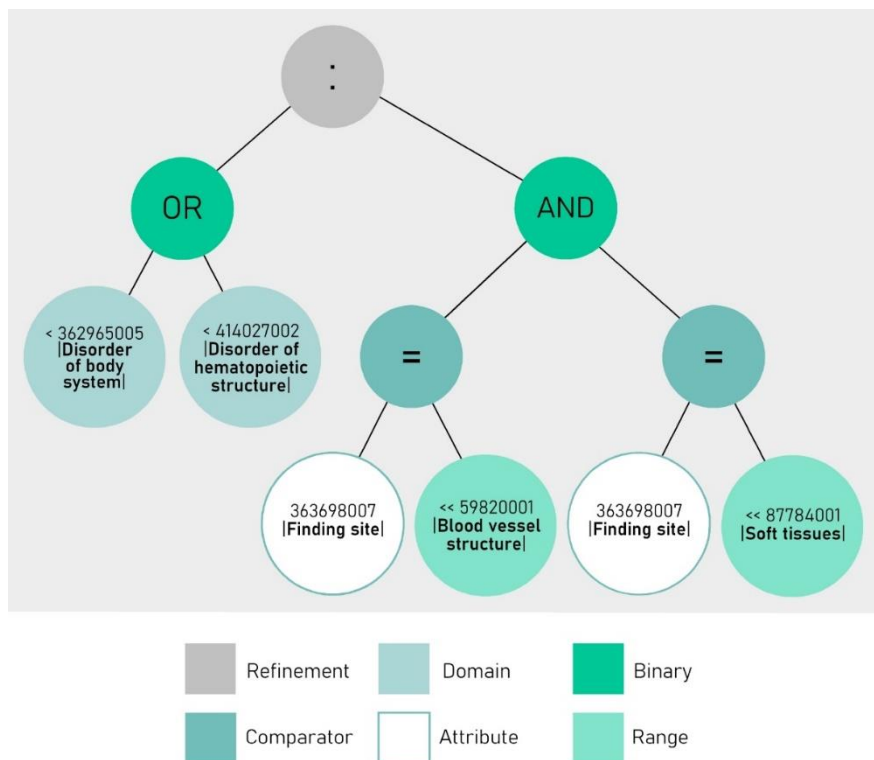


Figure 44. Syntax tree of EC example 7

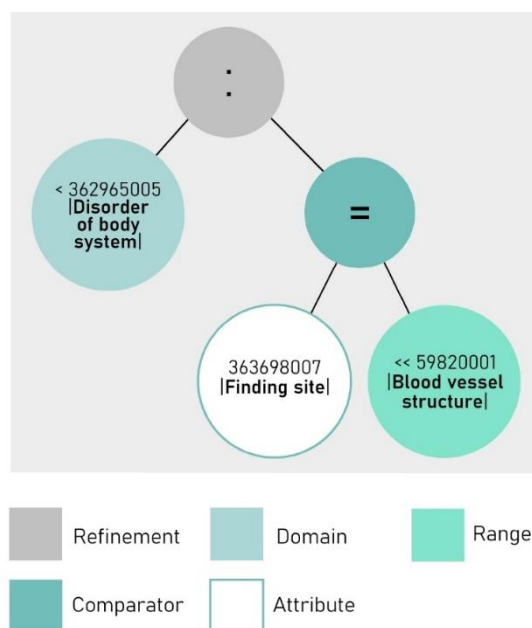


Figure 45. Syntax tree of EC example 2, which is the simplification of EC example 7

### 3.3.2 MRCM-based

The main goal of the MRCM-based simplification method is to optimize the execution time of the EC by removing attribute relationships that are not compliant with the MRCM. Note that conditions that contain non-compliant attribute relationships are never satisfied. Therefore, it is necessary to compare the triplets domain-attribute-range of the EC against the rules defined in the MRCM, including whether they are correctly grouped. Our approach is to evaluate the triplets from a logical point of view, i.e., a triplet that is equivalent to a MRCM rule is evaluated to True, and a triplet that does not conform to the MRCM is evaluated to False (it is never satisfied). Logical combinations between True and False triplets are evaluated according to the propositional logic conjunction and disjunction truth tables, leading in the empty set as a simplification when the final result is False. For instance, the execution of EC example 8 does not return any concept since it does not conform to the MRCM.



## EC example 8

```
<< 404684003 |Clinical finding (finding)| :
39133001 |With severity (attribute)| = << 272141005 |Severities (qualifier value)|
```

Note that the attribute “39133001 |With severity (attribute)|” is not an active concept in the SNOMED CT substrate, and thus, the triplet is evaluated to False. In such case, the simplification results in the empty set, and the EC can be logically represented with a refinement that is never met, i.e., “<< 404684003 |Clinical finding (finding)| : False”. In EC example 9, the first triplet is evaluated to False, whereas the second conforms to the MRCM, therefore it must remain (see Figure 46).

## EC example 9

```
<< 404684003 |Clinical finding (finding)| :
39133001 |With severity (attribute)| = << 272141005 |Severities (qualifier value)| OR
246075003 |Causative agent (attribute)| = << 410942007 |Drug or medicament
(substance)|
```

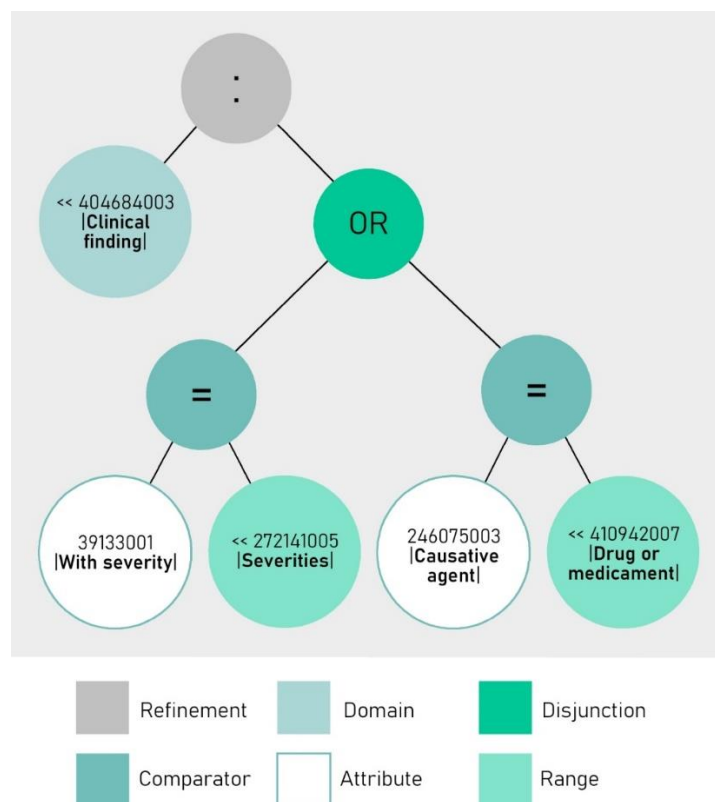


Figure 46. Syntax tree of EC example 9

As a result, the refinement is simplified to “False OR 246075003 |Causative agent (attribute)| = << 410942007 |Drug or medicament (substance)|”, which is equivalent to “246075003 |Causative agent (attribute)| = << 410942007 |Drug or medicament (substance)|” (see Figure 47).

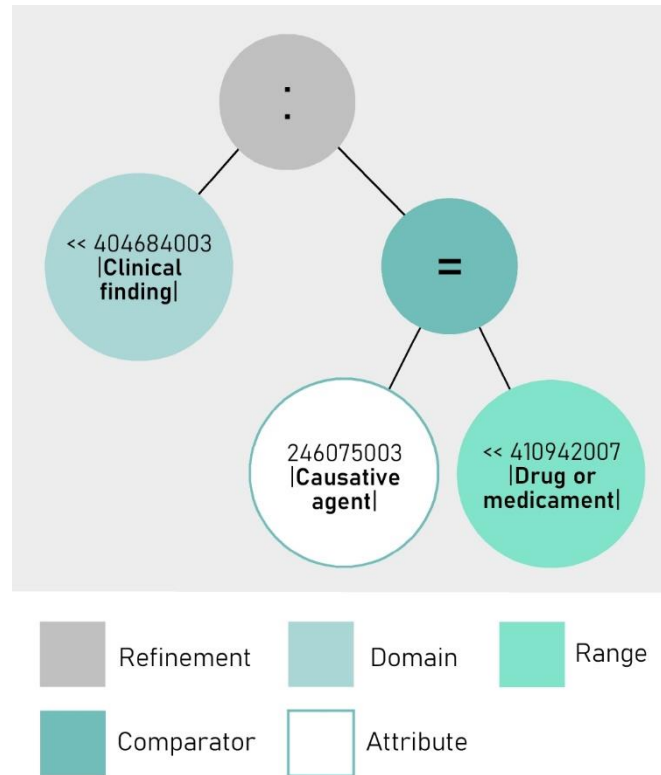


Figure 47. Syntax tree of the simplification of EC example 9

### 3.3.3 Logic definition-based

We leverage the logic definition of the focus concept of the EC to provide a method for the deletion of superfluous attribute relationships to improve EC execution time. Considering the mechanism of inheritance of SNOMED CT, we know that the descendants of the focus concept have, as minimum, the same attributes and attribute values -or descendants- as the focus concept. Therefore, we can detect superfluous attribute relationships by analyzing subsumption between concepts. Specifically, it may be possible to simplify the EC by applying the following rule (note that it is also valid for the “<” constraint operator):

Let “A”, “B”, “C” be concepts. Let “r”, “s” be attributes. Let “s=C” be part of the logic definition of “A”.

- 1)  $\ll A = (\ll A: r = \ll B)$  if “r” is the same as or subsumes “s” and “B” is the same as or subsumes “C”

Let us consider EC example 10 as an example (see Figure 48). The focus concept (i.e., “106063007 |Cardiovascular finding (finding)|”) has as finding site a structure of the cardiovascular system (see Figure 49). Therefore, its descendants must have as finding site a descendant of “113257007 |Structure of cardiovascular system (body structure)|” or the concept itself. Figure 50 shows some descendants of “106063007 |Cardiovascular finding|” and its logic definitions (note that only “363698007 |Finding site|” attribute relationships are shown while dotted |Is a| relationships indicate that there exist concepts in that path but they are not represented in order to simplify the figure). Consequently, in EC example 10, the attribute relationship is superfluous since it subsumes the attribute relationship defined in the logic definition of the focus concept, i.e., “113257007 |Structure of cardiovascular system (body structure)|” is a subtype of “91723000 |Anatomical structure (body structure)|”. Therefore, the refinement can be removed (see Figure 51). In other words, given that we are looking for cardiovascular findings located in any anatomical structure, and all cardiovascular findings are in the cardiovascular system, it is superfluous to define this condition in the EC.

#### *EC example 10*

< 106063007 |Cardiovascular finding (finding)|:  
 363698007 |Finding site (attribute)| = < 91723000 |Anatomical structure (body structure)|

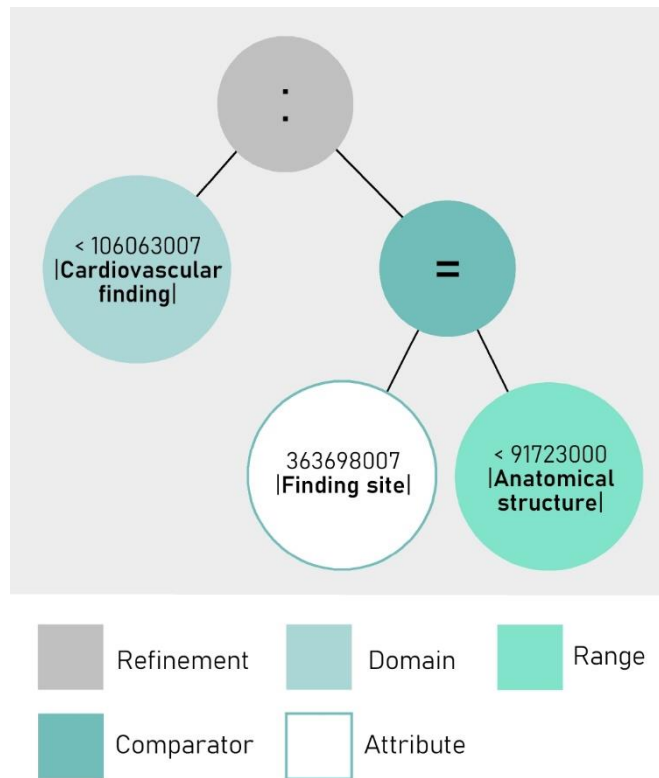


Figure 48. Syntax tree of EC example 10

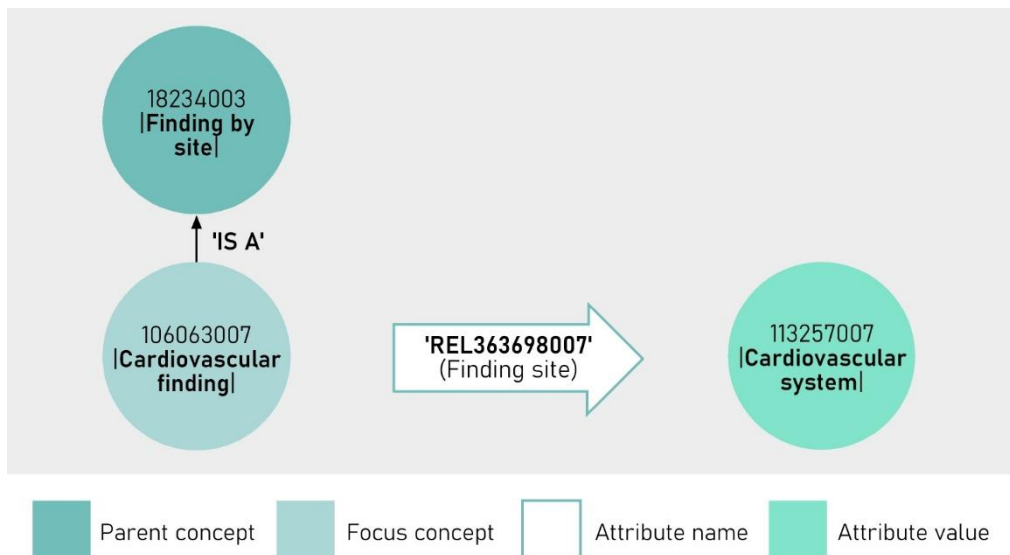


Figure 49. Logic definition of 106063007 |Cardiovascular finding (finding)|

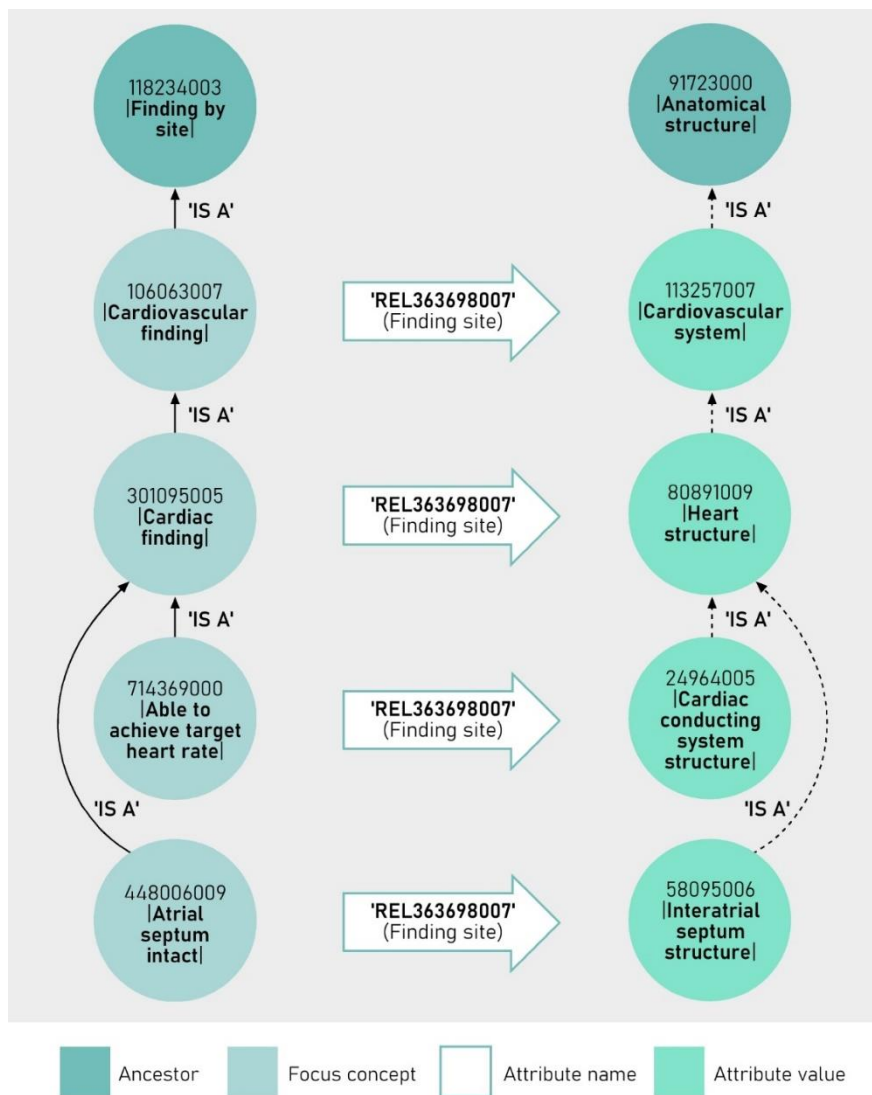


Figure 50. Logic definitions of the concept “106063007 |Cardiovascular finding (finding)|” and some of its descendants. Only “363698007 |Finding site|” attribute relationships are represented

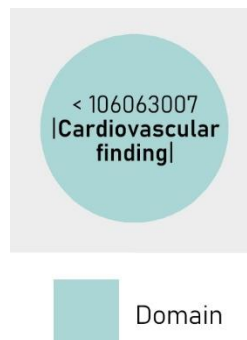


Figure 51. Syntax tree of the simplification of EC example 10

### 3.3.4 Post-execution simplification

In addition to pre-execution simplification, it is possible to further simplify the EC by analyzing its result subset. The general idea is to try to find a descendant of the focus concept whose set of descendants includes the whole result subset. If such concept exists, it can be used as new focus concept without altering the result. The substitution of the focus concept by a more specialized one may have a significant impact on the execution time as the number of comparisons performed in the execution of the EC is reduced. Furthermore, the refinements can be removed if the set of descendants of the new focus concept matches with the result subset.

To illustrate our method, we use recall and precision. In our context, recall stands for the fraction of concepts in the result subset that are included in a sub-hierarchy, while precision is the fraction of concepts in a sub-hierarchy that belong to the result subset. We define them as:

$$recall(E, C) = \frac{|{\text{answer}(E)} \cap \{\ll C\}|}{|{\text{answer}(E)}|}$$

$$precision(E, C) = \frac{|{\text{answer}(E)} \cap \{\ll C\}|}{|\{\ll C\}|}$$

where  $E$  is an EC,  $C$  is each descendant of the focus concept of  $E$ ,  $\text{answer}(E)$  is the result subset of  $E$ , and  $\{\ll C\}$  ( $\{< C\}$  if  $C$  does not belong to the result subset) is the set of descendants of concept  $C$ . If both recall and precision are equal to 1, then we can assure that  $\{\ll C\}$  is equal to the result subset. Therefore, the original EC can be rewritten simply as  $\ll C$ . Otherwise, if only recall is equal to 1 (and, therefore, precision is less than 1), we can conclude that  $\{\ll C\}$  includes the result subset, but also contains non-relevant concepts, therefore we must keep the refinements. Finally, if only precision is equal to 1 (and, therefore, recall is less than 1), we can assure that  $\{\ll C\}$  is included in the result subset. In this case, it is not possible to simplify the EC since the result subset contains more concepts than those included in  $\{\ll C\}$ .

Taking EC example 2 as an example, the application of the previous method yields as result the much simpler EC example 1. In this case, both recall and precision are equal to 1. In other words,  $\{answer(E)\}$  and  $\{\ll C\}$  define the same subset of concepts.

It may be the case that multiple concepts have a recall equal to 1 and a precision less than 1. In those cases, we select the concept with the maximum precision (i.e., the concept with the highest F1 score). As an example, EC example 11 defines the subset of those clinical findings that are located in the lower respiratory tract structure and whose associated morphology is any subtype of malignant neoplasm (see Figure 52).

#### EC example 11

< 404684003 |Clinical finding (finding)| :  
 363698007 |Finding site (attribute)| = << 82094008 |Lower respiratory tract structure (body structure)| ,  
 116676008 |Associated morphology (attribute)| = << 367651003 |Malignant neoplasm (morphologic abnormality)|

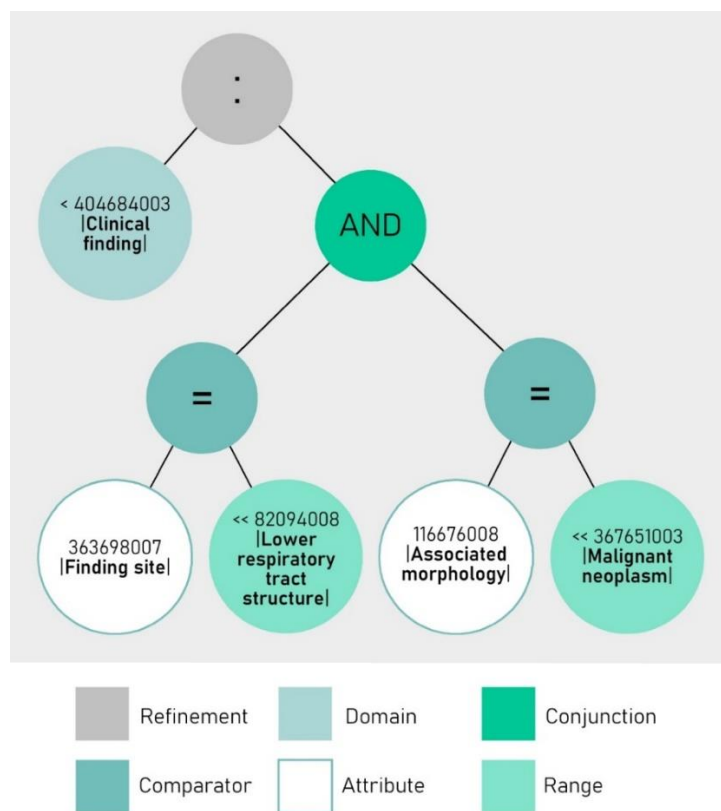


Figure 52. Syntax tree of EC example 11

If we apply the post-execution simplification, we obtain EC example 12.

*EC example 12*

```
< 301226008 |Lower respiratory tract finding (finding)|:  
363698007 |Finding site (attribute)| = << 82094008 |Lower respiratory tract structure  
(body structure)|,  
116676008 |Associated morphology (attribute)| = << 367651003 |Malignant  
neoplasm (morphologic abnormality)|
```

Note that the refinement is retained and the original focus concept “404684003 |Clinical finding (finding)|” (115537 descendants) is replaced by the much more specialized concept “301226008 |Lower respiratory tract finding (finding)|” (1753 descendants).

The substitution of the focus concept allows further simplification of EC example 11 by applying the pre-execution simplifications. The logic definition of the focus concept of EC example 12 states that it has as finding site the lower respiratory tract structure. This means that all descendants of the focus concept have as finding site a descendant of “82094008 |Lower respiratory tract structure (body structure)|” or itself. Therefore, according to the logic definition-based simplification, it is possible to remove this attribute relationship since it is redundant. As a result, we obtain EC example 13 (see Figure 53).

*EC example 13*

```
< 301226008 |Lower respiratory tract finding (finding)|:  
116676008 |Associated morphology (attribute)| = << 367651003 |Malignant  
neoplasm (morphologic abnormality)|
```



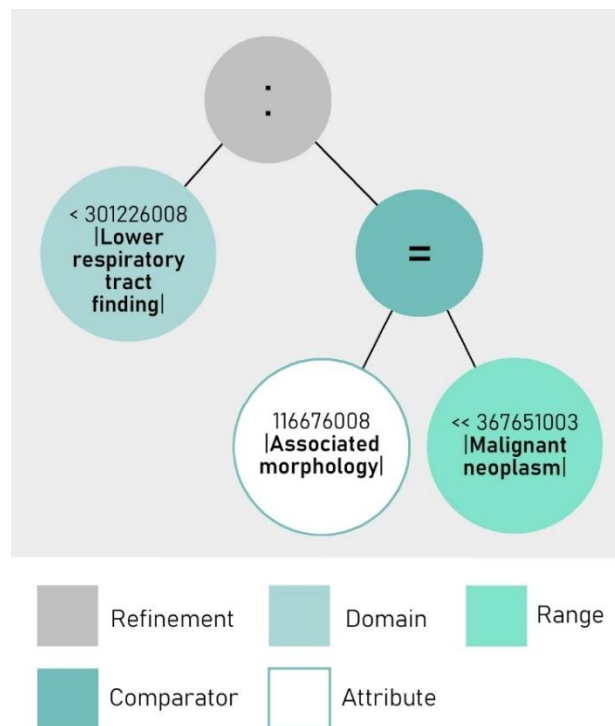


Figure 53. Syntax tree of EC example 13, which is the simplification of EC example 11

### 3.4 EC EXECUTION

Since the SNOMED CT graph database is stored as a property graph, the execution of ECs requires translating them from ECL to a query language for such databases, in our case to the Cypher Query Language. Therefore, we have implemented a mechanism for translating ECs into Cypher clauses that are executed over the SNOMED CT database in order to obtain the result subset.

Cypher is a declarative query language that provides capabilities for both querying and modifying data in property graphs. It allows users to store and retrieve data from a Neo4j graph database. Cypher has been designed intentionally similar to SQL to help users in the transition between the two languages.

Pattern matching is the pivotal concept in Cypher queries. Its syntax provides a visual form as “ASCII-Art” to express patterns, such as “(c1)-[r]->(c2)”. Nodes are enclosed in parentheses and can have zero to many labels. Labels are written after the node separated

with a “.” symbol (e.g., “(node: Concept)”). Relationships are enclosed in square brackets and are represented using an arrow between two nodes (“-->” or “<--”) (e.g., “- [:REL] ->”). Properties are represented using curly braces within the parentheses of a node or the square brackets of a relationship (e.g., “(node: Concept {conceptId: '816994007'})”, “[:REL {relId: '116676008'}]”).

To search for patterns, Cypher provides the “MATCH” clause. There are as minimum three components on every Cypher “MATCH” clause when looking for simple paths in a graph: the source node, the destination node and an edge between them. In terms of our serialization of SNOMED CT, simple paths are represented by a source concept, a destination concept, and a relationship between them. For instance, the following “MATCH” clause matches all concepts that are descendants of the concept “85828009 |Autoimmune disease (disorder)|”: “MATCH (a: Concept {conceptId: '85828009'}) <-[[:RELCL]] - (r: Concept)” (note that ‘RELCL’ is the transitive closure of the ‘IS A’ relationship). This produces several bindings for the variable “r”, one for each descendant concept.

Cypher allows traversing relationships of a variable length by using “- [:REL\*min..max] ->”, where “min” and “max” are optional and default to 1 and many, respectively. Note that min=0 allows including concepts that do not participate in any “REL” relationship. For instance, in “(c: Concept) - [:RELCL\*0..1] -> (d: Concept {conceptid: '59820001'})” the variable “c” is bound not only to the descendants of the concept “59820001”, but also to the concept “59820001” itself.

The “WITH” clause is used to pipe the results from one query part to be used as starting point or criteria in the next query part. This clause allows the manipulation of the output before passing it on to the following query part, including aggregation functions.

The “RETURN” clause computes and projects expressions, in our case: nodes, relationships, properties or patterns.

The expressive power of Cypher makes it possible to translate an EC into a single or multiple target queries whose execution produces the subset defined by the source EC. Cypher query takes as input a property graph and returns a table. In our case, the return table is composed of two columns: concept identifier and FSN. The strategy when querying the SNOMED CT graph using the Cypher language is to look for paths that match the constraints

imposed by the given EC for both nodes and relationships. Let us take as an example EC example 2. The focus concept is “362965005 |Disorder of body system (disorder)|”, therefore the result concepts must be descendants of it. The set of descendants can be calculated using the following MATCH clause: “MATCH (a: Concept {conceptId: '362965005'}) <-[:RELCL] - (r: Concept)”, where “r” is bound to each descendant concept of “362965005 |Disorder of body system (disorder)|”. The set of valid concepts, i.e., the values of variable “r”, must also satisfy the refinement condition, which can be expressed by the following “MATCH” clause: “MATCH (r: Concept) -[:REL363698007] -> (c: Concept) -[:RELCL\*0..1] -> (d: Concept {conceptId: '59820001'})”, where “REL363698007” is the “363698007 |Finding site (attribute)|” attribute and “59820001” is the identifier of the concept “59820001 |Blood vessel structure (body structure)|”. As a result, the Cypher query equivalent to EC example 2 is:

```
MATCH (a: Concept {conceptId: '362965005'}) <-[:RELCL] - (r: Concept)
-[:REL363698007] -> (c: Concept) -[:RELCL*0..1] ->
(d: Concept {conceptId: '59820001'})
RETURN DISTINCT r.conceptId AS CONCEPTID, r.fsn AS FSN
```

Table 5 shows the translation into Cypher of some basic EC patterns.

	Simple EC pattern	Cypher translation pattern
1	<A	MATCH R -[:RELCL] -> A RETURN R
2	<<A	MATCH R -[:RELCL*0..1] -> A RETURN R
3	>A	MATCH R <-[:RELCL] - A RETURN R
4	>>A	MATCH R <-[:RELCL*0..1] - A RETURN R
5	<!A	MATCH R -[:ISA] -> A RETURN R
6	^A	MATCH R -[:MEMBEROF] -> A RETURN R
	Compound EC pattern	
7	<A AND <B	MATCH A <-[:RELCL] - R -[:RELCL] -> B RETURN R

8	<A OR <B	MATCH R - [:RELCL] -> C WHERE C.id = A.id OR C.id = B.id RETURN R
9	<A MINUS <B	MATCH R - [:RELCL] -> A WHERE NOT R - [:RELCL] -> B RETURN R
	Refined EC pattern	
10	<A : B = <C	MATCH A <- [:RELCL] - R - [:RELB] -> D - [:RELCL] -> C RETURN DISTINCT R
11	<A : {B = <C}	MATCH A <- [:RELCL] - R - [r:RELB] -> D - [:RELCL] -> C WHERE r.relGroup <> '0' RETURN DISTINCT R
12	<A : rev B = <C	MATCH A <- [:RELCL] - R <- [:RELB] - D - [:RELCL] -> C RETURN DISTINCT R
13	<A : B != <C	MATCH A <- [:RELCL] - R - [:RELB] -> D WHERE NOT D - [:RELCL] -> C RETURN DISTINCT R
14	<A : [a..b] B = <C	MATCH A <- [:RELCL] - R - [r:RELB] -> D - [:RELCL] -> C WITH R, COUNT(r) AS N WHERE N >= a AND N <= b RETURN R
15	<A : B = (<C OR <D)	MATCH A <- [:RELCL] - R - [:RELB] -> F - [:RELCL] -> E WHERE E.id = C.id OR E.id = D.id RETURN DISTINCT R
16	<A : B = (<C AND <D)	MATCH C <- [:RELCL] - E - [:RELCL] -> D MATCH A <- [:RELCL] - R - [:RELB] -> E RETURN DISTINCT R
17	<A : B = (<C MINUS <D)	MATCH E - [:RELCL] -> C WHERE NOT E - [:RELCL] -> D MATCH A <- [:RELCL] - R - [:RELB] -> E RETURN DISTINCT R
18	<A : B = (<C : D = <E)	MATCH C <- [:RELCL] - F - [:RELD] -> G - [:RELCL] -> E MATCH A <- [:RELCL] - R - [:RELB] -> F RETURN DISTINCT R
19	(<A : B = <C) : D = <E	MATCH A <- [:RELCL] - R - [:RELB] -> F - [:RELCL] -> C MATCH R - [:RELD] -> G - [:RELCL] -> E RETURN DISTINCT R

Table 5. Patterns of ECs and its translation into Cypher Query Language

It should be noted that more complex EC patterns could be translated by combining basic ones. As an example, a complex EC containing attribute cardinality and a compound range can be translated into Cypher by combining patterns 14 and 17 of Table 5 as shown in Table 6.

	Refined EC pattern	Cypher translation pattern
20	<A : [a..b] B = (<C MINUS <D)	<pre> MATCH E - [:RELCL] -&gt; C WHERE NOT E - [:RELCL] -&gt; D MATCH A &lt;- [:RELCL] - R - [r:RELB] -&gt; E WITH R, COUNT(r) AS N WHERE N &gt;= a AND N &lt;= b RETURN R </pre>

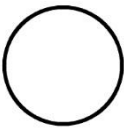

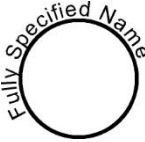
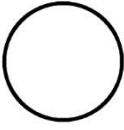

Table 6. Combination between EC patterns 14 and 17 translated into Cypher Query Language

### 3.5 SUBSET VISUALIZATION

Understanding and validating a result subset require to provide users with information about how the concepts that make up the subset are related in terms of hierarchies and which of these hierarchies are more relevant in terms of recall and precision. The visual representation of this information may facilitate the validation of the subset at a glance, for instance by detecting irrelevant concepts or hierarchies. Requirements for such visualization also include showing the FSN of concepts, whether the represented concept is included in the result subset, and the heterogeneity of the concepts contained in a sub-hierarchy. In addition, a key requirement is to provide the possibility to dynamically explore sub-hierarchies by zooming into them to see their details. It is important to note that the presentation of all this information to the user requires a compact visual representation in order to see as much information as possible at a glance. To collect and present this information, a graphical visualization based on the circle packing [38] is proposed that fits particularly well with our requirements. Circle packing is a geometrical concept that defines a collection of circles that are connected without overlapping in a way that each circle touches another. It has been implemented using the D3 Javascript library [39], which allows to explore subsets dynamically, the customization of colors, sizes and border styles to represent particular meanings, as well as to show textual information and numerical data

inside or around circles. The data shown by the circle packing is first calculated by SNQuery and stored in a JSON file, which can be integrated with other visualization methods to represent the same or part of the data in a different way, such as a tree.

We have applied the circle packing to represent the concepts of the result subset arranged in sub-hierarchies. Specifically, each circle represents a root concept of a sub-hierarchy and comes with additional information such as recall and precision. These root concepts are either members of the result subset or they have at least one descendant in it. Only non-leaf concepts are shown. Each circle is ordered in levels, which correspond to the depth of the sub-hierarchy that it represents. These levels can be dynamically explored by zooming into and out the circles. Additionally, and in order to locate outlier concepts, it is possible to calculate the descendant concepts that are out of the result subset and the result concepts that are out of the sub-hierarchy represented by the circle. The first level displays the focus concept of the EC. Table 7 shows the features provided by the circle packing visualization.

Feature		Description
Sub-hierarchy		A sub-hierarchy is represented by a circle.
Sub-circles		The set of child sub-hierarchies are represented by sub-circles.
FSN		The Fully Specified Name of the top concept of the sub-hierarchy is shown around the circle.
Border style		The border of the circle is drawn in solid line if the top concept of the sub-hierarchy is included in the result subset.
		The border of the circle is drawn in dashed line if the top concept of the sub-hierarchy is not included in the result subset.

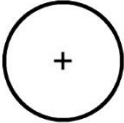



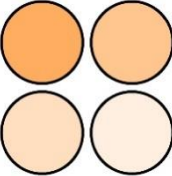
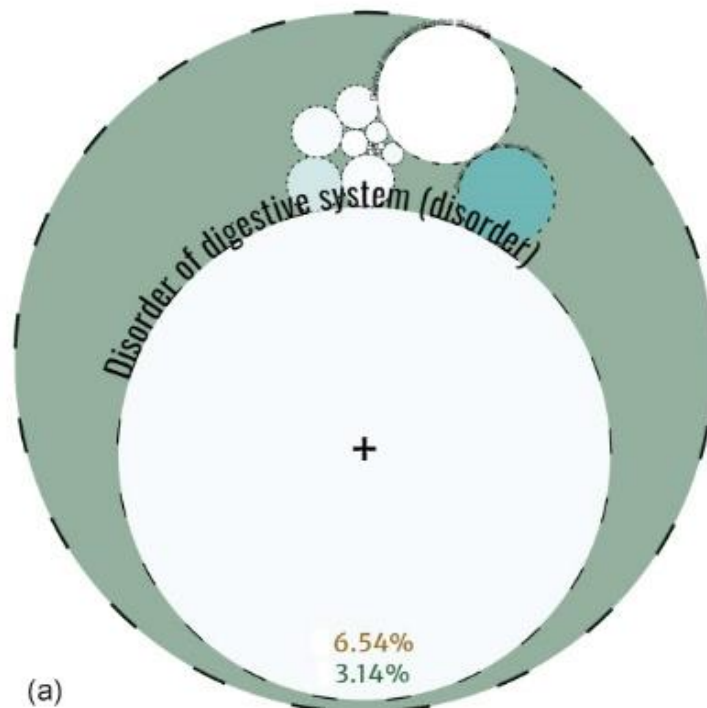
“+” symbol		Circles marked with a “+” symbol can be zoomed in to show their child sub-hierarchies.
Size		Size of circles is proportional to its polyhierarchical degree measured as the total number of ancestors of the concepts contained. Polyhierarchical degree is intended to determine the heterogeneity of the concepts contained in a sub-hierarchy. The higher the degree, the more heterogeneous is the sub-hierarchy.
Recall		For each circle it is shown its recall, i.e., the percentage of the subset that is matched by the descendants.
Precision		For each circle it is shown its precision, i.e., the percentage of descendants that are included in the subset.
Color		Green color is used to show the relevance in terms of F1 score. The more intense the green, the higher is F1 score and therefore the more relevant is the sub-hierarchy represented by the circle.
		Orange color is used to show the four most relevant sub-hierarchies in terms of F1 score. The more intense the orange, the more relevant is the sub-hierarchy. They are displayed on demand without needing to zoom into the circles.

Table 7. Features showed by the circle packing visualization

As an example, Figure 54 shows the sequence followed to find and display the most relevant sub-hierarchy of EC example 2. Figure 54a shows the first level of the circle packing of EC example 2. The focus concept of the EC (i.e., “362965005 |Disorder of body system (disorder)|”) is at the root and therefore it encompasses all circles. It is also displayed the “53619000 |Disorder of digestive system (disorder)|” sub-hierarchy. It covers most of the image, which is indicative that it is a heterogeneous sub-hierarchy. It appears with a very light shade of green (i.e., low relevance) since it has a low F1 score (recall=6.54%, precision=3.14%). The rest of the image is covered by smaller circles (i.e., more specific

sub-hierarchies). One of them is colored with the more intense green (i.e., “49601007 |Disorder of cardiovascular system (disorder)|”), meaning that it is the most relevant sub-hierarchy at this level. It also contains the most relevant sub-hierarchy of EC example 2 (in orange), as shown in Figure 54b (i.e., “27550009 |Disorder of blood vessel (disorder)|”). Figure 54c shows the result of zooming into the “49601007 |Disorder of cardiovascular system (disorder)|” sub-hierarchy. At this level, several sub-hierarchies with different relevance are shown, including “27550009 |Disorder of blood vessel (disorder)|” (in orange), which is, as aforementioned, the most relevant sub-hierarchy of EC example 2 (recall=100%, precision=100%). Note that since “27550009 |Disorder of blood vessel (disorder)|” has both recall and precision equal to 100%, the post-execution simplification of EC example 2 is “<<27550009 |Disorder of blood vessel (disorder)|”.





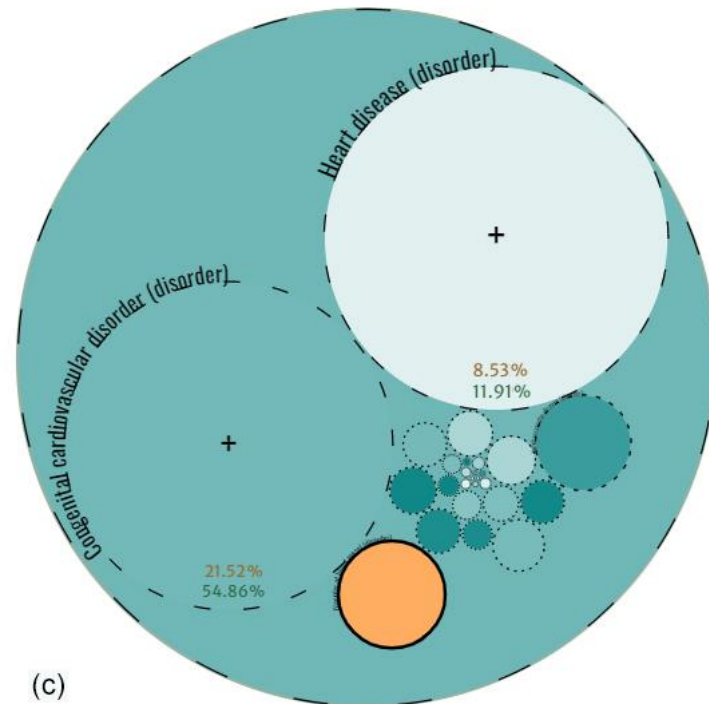
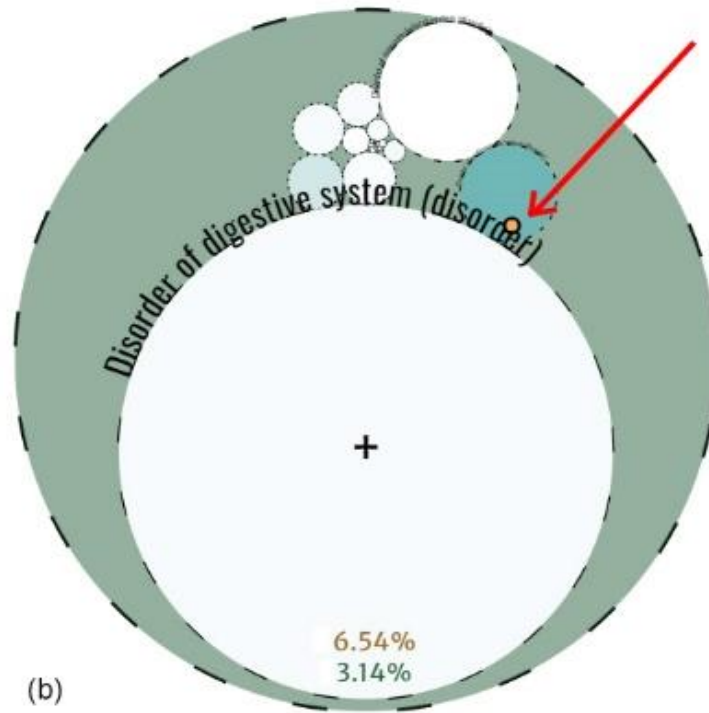


Figure 54. Circle packing visualization of EC example 2 (using the SNOMED CT July 2020 International Edition). First level of visualization is displayed in (a); (b) shows the most relevant sub-hierarchy (in orange, pointed out by an arrow); (c) shows the result of zooming

In the following chapter we explain how the methods presented in this chapter are applied and put into practice in a tool that has the ability to validate, simplify and execute ECs. The subset visualization method presented here has also been incorporated into the tool. Additionally, an evaluation of the tool in terms of execution times has been performed and presented. At the end of Chapter 4 we present a discussion about the contents of both Chapter 3 and Chapter 4, since they are closely linked.

## CHAPTER 4. SNQUERY: ECL EXECUTION ENGINE BASED ON GRAPH DATABASES

---

### 4.1 OVERVIEW OF SNQUERY

SNQuery is an EC execution platform that makes use of the methods defined, implemented and presented in Chapter 3. The process to create, parse, simplify, semantically validate and execute an EC, and visualize the result subset is as follows. After an EC is entered, SNQuery parses it and, if the syntax is correct, it applies the simplification methods (note that applying the simplification methods is optional and it depends on the user to apply them or not). The simplification methods are applied by SNQuery iteratively, i.e., the methods are executed sequentially in a loop until the EC cannot be further simplified. This process yields both the semantic validation and the simplification of the EC. To execute the EC, SNQuery performs the translation process between the EC and the Cypher Query Language described in section 3.4, which yields a single or multiple target Cypher queries that are executed over the Neo4j graph database to produce the intended result subset.

SNQuery presents the result subset in sortable tabular form by default, as shown in Figure 55. In addition to the numerical identifier and the FSN, each concept is linked to the SNOMED International SNOMED CT Browser [40] to explore its details. SNQuery allows the exportation of subsets as txt, xls and csv file formats.

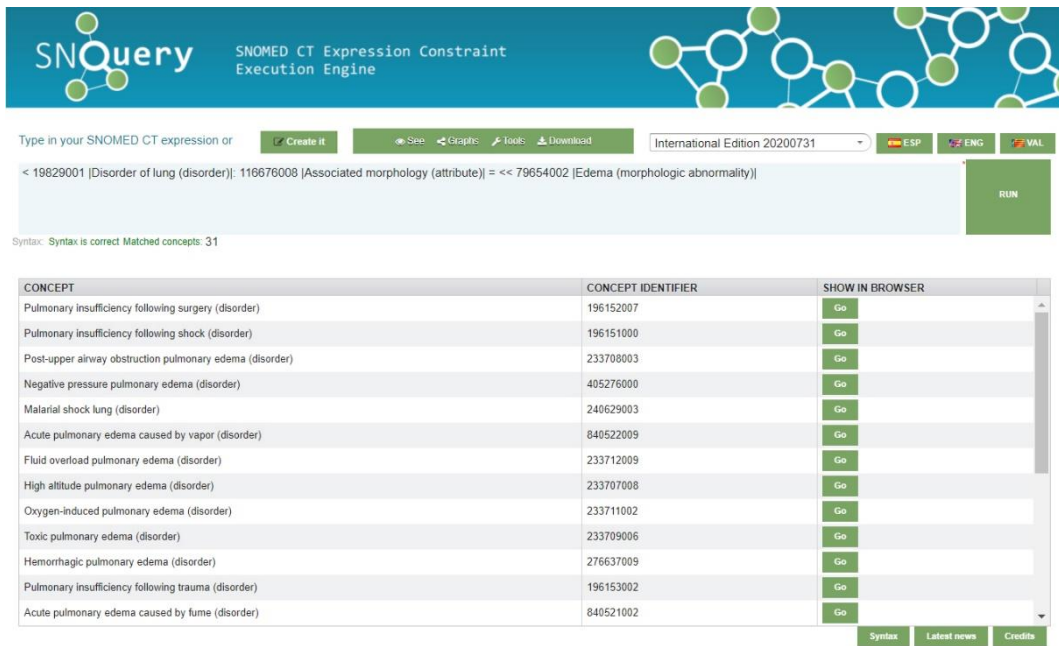


Figure 55. SNQuery showing some disorders of lung whose associated morphology is any type of edema

In addition to the circle packing visualization method described in section 3.5, SNQuery also incorporates a hierarchy-based and a tree-based visual representations that can be used to visualize the subsets. As examples, we show Figure 56, Figure 57, and Figure 58.

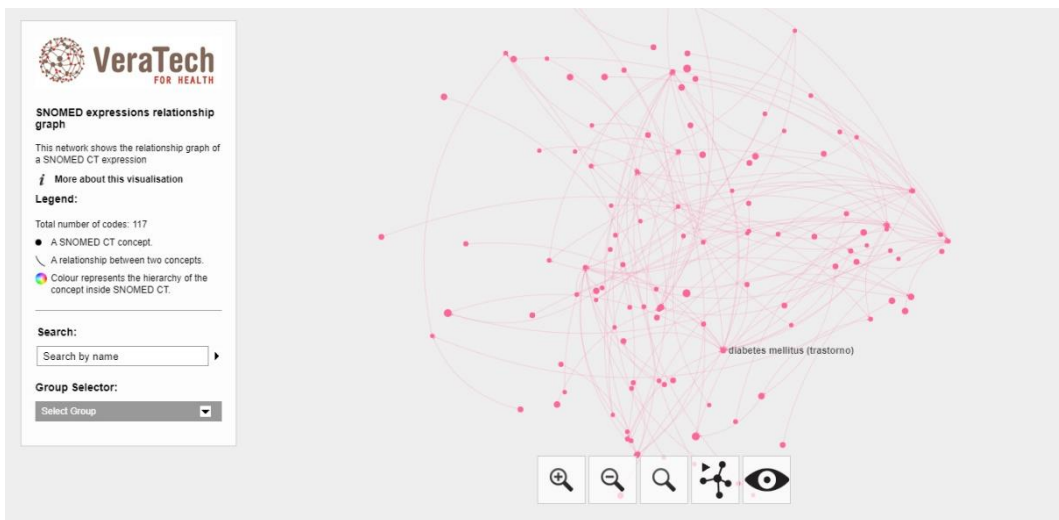


Figure 56. Diabetes mellitus hierarchy

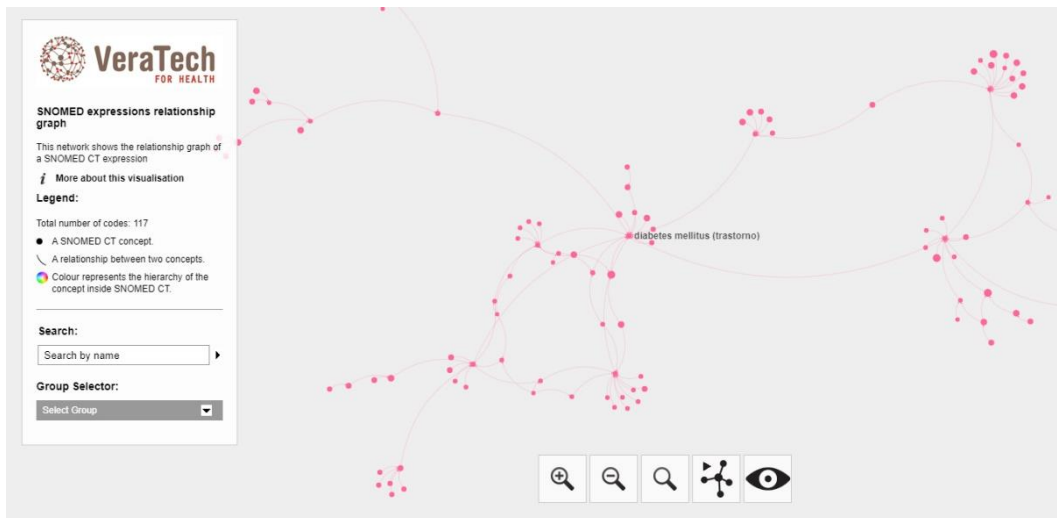


Figure 57. Diabetes mellitus hierarchy after applying the ForceAtlas2 Graph Layout Algorithm [41]

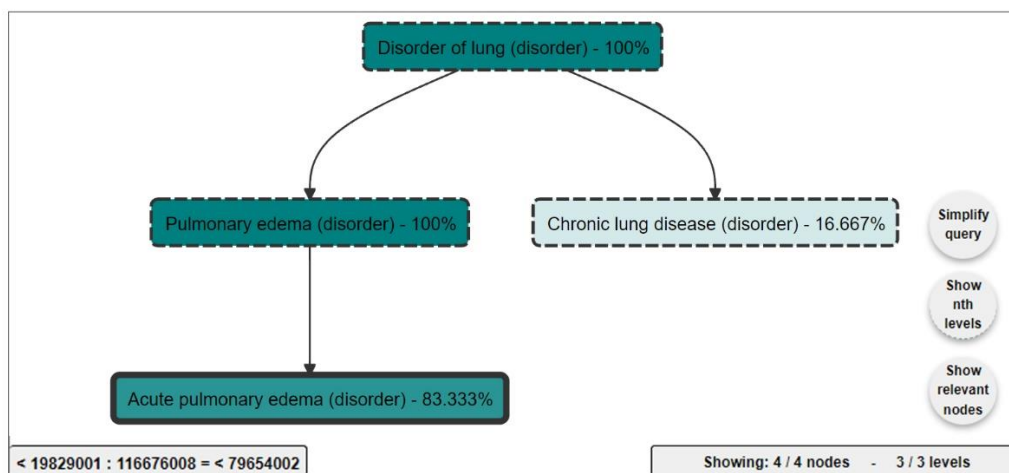


Figure 58. Tree-based visualization showing three levels in depth of those disorders of lung associated with edema (only intermediate concepts are shown)

In order to define ECs in SNQuery, users must be familiar with the basics of the ECL syntax [5] (see section 2.7) and, in turn, with the logical model of SNOMED CT [4]. As with any computer-interpretable language, it requires a learning phase. The time spent in this process will depend on the background and specific skills of the user. SNQuery includes several mechanisms to assist users in creating ECs, such as:

- Syntactic validation of the EC
- Historic of the last executions
- Validation of the EC according to the concept model
- ECs examples section aimed at being the basis for building ECs instead of defining them from scratch

Additionally, for training purposes, SNQuery incorporates a visual authoring tool that is particularly useful for users without a deep knowledge of ECL. The tool provides a series of predefined templates that can be combined to create simple and complex ECs (see Figure 59). It also integrates different ways to search for concepts in SNOMED CT. Moreover, the concept search is restricted to those concepts that are valid at a given point of the EC according to the concept model (e.g., given a domain, it presents a list with the allowed attributes; given an attribute, it allows to search for concepts only in the permitted domain and range).

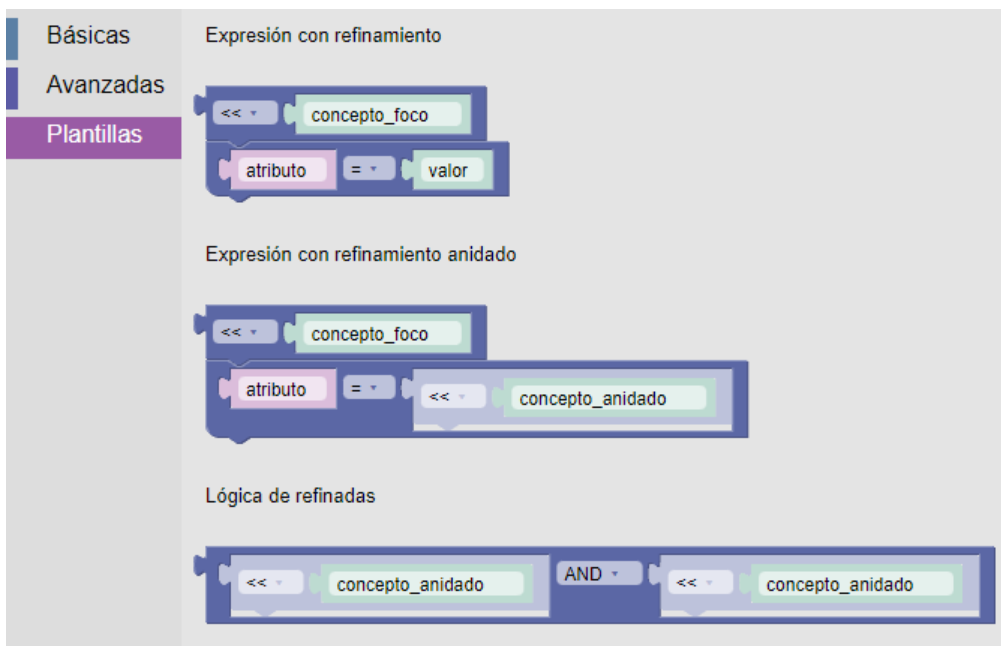


Figure 59. Visual ECs authoring tool

SNQuery allows the selection of a SNOMED CT database to be the substrate over which ECs are executed. Each database corresponds to an edition/release that have been previously generated using our loading module, which makes use of the SNOMED CT Snapshot RF2 files. The loading module is also able to generate any local extension and to incorporate it as a substrate in SNQuery, such as the SNOMED CT UK Clinical and Drug Extensions. Additionally, since the content of SNOMED CT evolves from one release to another, SNQuery incorporates a way to check the differences on the results obtained when executing an EC over different substrates. The purpose of this comparator module is to detect new, missing, and inactive concepts over time, to make semantic interoperability easier.

SNQuery also allows checking whether a concept is included in the subset defined by a given EC, by using a self-created inclusion operator (“IN”). As an example, the execution of EC example 14 yields the concept “31874001 |True (qualifier value)|” since “707480001 |Chronic hemolytic anemia (disorder)|” is included in the subset of diseases that interpret a red blood cell count procedure.

*EC example 14*

```
707480001 |Chronic hemolytic anemia (disorder)| IN < 64572001 |Disease
(disorder)| : 363714003 |Interprets (attribute)| = 14089001 |Red blood cell count
(procedure)|
```

## 4.2 OTHER FUNCTIONALITIES

### 4.2.1 MSSSI refsets list

The MSSSI, which stands for *Ministerio de Sanidad, Servicios Sociales e Igualdad español*, launches a new version of its clinical concept extension twice a year, which includes approximately 80 refsets. MSSSI refsets are also updated into SNQuery, including *allergies, alerts, blood pressure automonitorization, chronic obstructive pulmonary disease (COPD) phenotypes, problem status indicators, surgical records, severity indicators, diabetes complications, types of documents for personal identification, specialties, sexes, individualized vaccines and anesthesia types*, among others. Note that it is not only possible to retrieve the members of a refset using the “^” ECL operator but to refine them by using

the ':' operator , as in the following EC, which define the members of the *MSSSI allergies refset* that are caused by a subtype of vegetable:

```
^ 900000051000122100 |Patient allergies refset (foundation metadata concept)|:  
246075003 |Causative agent (attribute)| = < 22836000 |Vegetable (substance)|
```

#### 4.2.2 Multilingual interface

The platform supports three languages at this moment: English, Spanish and Valencian. It should be noted that the language selector only affects the interface of the platform, not the resulting subsets retrieved after executing an EC, which depend on the selected SNOMED CT edition.

#### 4.2.3 Mapping to ICD-10

Each of the concepts of a subset retrieved after executing its defining EC in the platform, can be mapped to its corresponding ICD-10 code (or codes). For this purpose, SNQuery makes use of the official mapping files delivered by SNOMED International twice a year and included in the international and Spanish editions.

#### 4.2.4 Conversion from brief to long syntax and vice versa

As explained earlier in this thesis, ECL offers two logically equivalent syntaxes. On the one hand, brief syntax is the normative syntax and it is aimed to be as compact as possible. On the other hand, long syntax uses English as alternative to the symbols that are defined in the brief syntax. The platform provides a converter from brief to long syntax and vice versa. It should be noted that the concepts and attributes of the ECs are completed with its FSN when using this conversion process.



### 4.3 EVALUATION

An analysis of SNQuery in terms of execution times has been performed. Table 8 shows a collection of 10 ECs before and after applying the simplification methods. Label “S” stands for “Simplification”. We also show a description for each EC.

	Expression Constraint	Description
EC <sub>1</sub>	< 473011001  Allergic condition (disorder)	Allergic conditions
S	Not applicable	-
EC <sub>2</sub>	< 473011001  Allergic condition (disorder)  : 246075003  Causative agent (attribute)  = < 255620007  Food (substance)	Food allergies
S	No simplification	-
EC <sub>3</sub>	< 473011001  Allergic condition (disorder)  : 246075003  Causative agent (attribute)  != < 255620007  Food (substance)	Allergies caused by something different to food
S	No simplification	-
EC <sub>4</sub>	< 255620007  Food (substance)  : R 246075003  Causative agent (attribute)  = < 473011001  Allergic condition (disorder)	Foods that cause allergies
S	No simplification	-
EC <sub>5</sub>	< 75478009  Poisoning (disorder)  : 246075003  Causative agent (attribute)  = (< 105899005  Animal agent (substance)  AND < 35331000  Toxic substance (substance) )	Poisonings caused by animal agents that are toxic substances
S	<< 44400004  Toxic effect of venom (disorder)	Toxic effects of venoms
EC <sub>6</sub>	< 404684003  Clinical finding (finding)  : 116676008  Associated morphology (attribute)  = << 79654002  Edema (morphologic abnormality)	Clinical findings associated with any type of edema
S	<< 267038008  Edema (finding)	Edemas
EC <sub>7</sub>	< 781474001  Allergic disorder (disorder)  : 370135005  Pathological process (attribute)  = << 472964009  Allergic process (qualifier value) , 363698007  Finding site (attribute)  = << 442083009  Anatomical or acquired body structure , 116676008  Associated morphology (attribute)  = << 49755003  Morphologically abnormal structure	Allergic disorders with any type of allergic process as a pathological process, located in any anatomical or acquired body

		structure and associated with any morphologically abnormal structure
S	< 781474001  Allergic disorder (disorder)	Allergic disorders
EC <sub>8</sub>	< 414029004  Disorder of immune function (disorder)  : 370135005  Pathological process (attribute)  = 263680009  Autoimmune process (qualifier value)	Disorders of immune function with an autoimmune process as pathological process
S	<< 85828009  Autoimmune disease (disorder)	Autoimmune diseases
EC <sub>9</sub>	< 106063007  Cardiovascular finding (finding)  : 363698007  Finding site (attribute)  = << 113257007  Cardiovascular system (body structure)	Cardiovascular findings that are located in the cardiovascular system
S	< 106063007  Cardiovascular finding (finding)	Cardiovascular findings
EC <sub>10</sub>	< 404684003  Clinical finding (finding)  : 363698007  Finding site (attribute)  = << 82094008  Lower respiratory tract structure  , 116676008  Associated morphology (attribute)  = << 367651003  Malignant neoplasm	Clinical findings located in the lower respiratory tract and associated to a malignant neoplasm
S	< 301226008  Lower respiratory tract finding (finding)  : 116676008  Associated morphology (attribute)  = << 367651003  Malignant neoplasm	Lower respiratory tract findings associated to a malignant neoplasm

Table 8. Examples of ECs before and after applying the SNQuery simplification process

Table 9 shows the execution times and standard deviations (in milliseconds) of the ECs presented in Table 8. Label “R\_SIZE” is the cardinality of the result subset (i.e., number of concepts retrieved after executing the EC). “D\_SIZE”, “D\_SIZE\_S”, “T (SD)”, and “TS (SD)” are the domain size and execution time (average and standard deviation) obtained before and after applying the simplification methods, respectively. Each execution time has been calculated as the average execution time of 100 runs over the SNOMED CT July 2020

International Edition on a Windows 10 Pro x64, Intel Core i7-6700HQ CPU 2.60 GHz 16GB RAM. It should be noted that SNQuery uses an eager evaluation approach (i.e., it is retrieved the full result subset).

	EC <sub>1</sub>	EC <sub>2</sub>	EC <sub>3</sub>	EC <sub>4</sub>	EC <sub>5</sub>	EC <sub>6</sub>	EC <sub>7</sub>	EC <sub>8</sub>	EC <sub>9</sub>	EC <sub>10</sub>
R_SIZE	1710	89	1439	73	67	584	273	455	8101	195
D_SIZE	1710	1710	1710	1710	3620	115537	273	1713	8101	115537
D_SIZE_S	-	-	-	-	67	584	273	455	8101	1753
T (SD)	328 (15)	561 (21)	2509 (45)	554 (32)	673 (18)	985 (26)	1571 (50)	793 (21)	1581 (39)	1592 (49)
TS (SD)	-	-	-	-	211 (13)	318 (10)	244 (15)	321 (21)	531 (15)	1011 (30)

*Table 9. Result subset size, domain size, execution time (in milliseconds), and standard deviation obtained before and after applying the simplification process of each EC*

The data presented in Table 9 suggest that execution time does not depend on the size of the result subset neither on the size of the domain of the EC. It is also observed that the more drastic the simplification, the greater the reduction of execution time, being EC<sub>7</sub> the EC with the highest percentage decrease (84.47%), EC<sub>10</sub> with the lowest (36.49%), and being 63.88% the percentage decrease average.

One of the potential advantages when leading with graph-oriented databases is the great flexibility to add new nodes and relationships to the graph without affecting existing queries [30]. In this regard, we have evaluated if an increase in the substrate will increase the EC execution time and degrade the performance.

The experiment consists of increasing the number of concepts in the domain of a given refined EC (i.e., the number of nodes to be traversed to calculate the subset is increased) but keeping the result subset constant (see Figure 60).

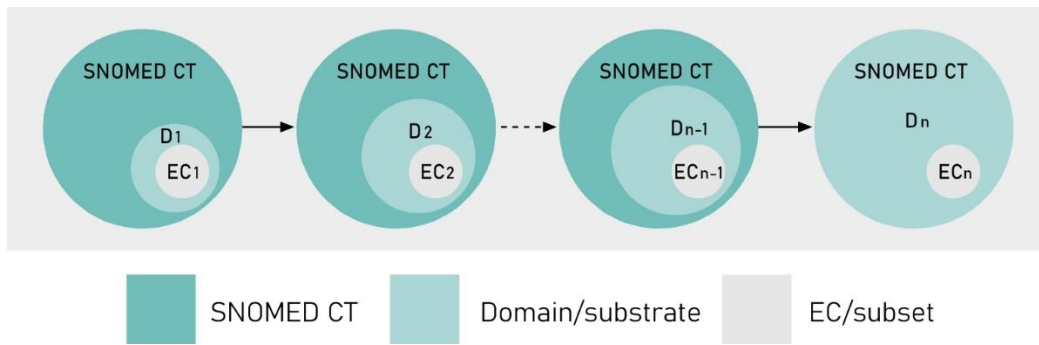


Figure 60. Domains  $D_1$  to  $D_n$  over which the subset is calculated (note that  $D_n$  is represented by all the SNOMED CT substrate)

For this purpose, we leverage our post-execution simplification method, since it allows narrowing down the focus concept of a refined EC (see section 3.3.4). For example, EC example 15 can be simplified to EC example 16 and therefore all the concepts in a hierarchy path from “249230006 |Male genitalia finding (finding)|” to “138875005 |SNOMED CT Concept|” can be used as focus concept without altering the result.

*EC example 15*

< 138875005 |SNOMED CT Concept| : 363698007 |Finding site (attribute)| =  
 < 127903009 |Male genital organ structure (body structure)|

*EC example 16*

< 249230006 |Male genitalia finding (finding)| : 363698007 |Finding site (attribute)| =  
 < 127903009 |Male genital organ structure (body structure)|

Table 10 shows a set of equivalent ECs ( $EC_{11}$  to  $EC_{19}$ ) in terms of returned concepts, where  $EC_{11}$  is the EC obtained after applying our post-execution simplification method in the original EC ( $EC_{19}$ );  $EC_{12}$  to  $EC_{18}$  are equivalent intermediate ECs that have as focus concept a concept in a hierarchy path from “249230006 |Male genitalia finding (finding)|” to “138875005 |SNOMED CT Concept|”. “R\_SIZE” is the size of the result subset, “D\_SIZE” is the size of the domain, and “T (SD)” is the average execution time of 100 runs and the standard deviation (in milliseconds).

	Expression Constraint	R_SIZE	D_SIZE	T (SD)
EC <sub>11</sub>	< 249230006  Male genitalia finding (finding) : REF	1072	1215	717 (25)
EC <sub>12</sub>	< 300479008  Genital finding (finding) : REF	1072	4296	739 (28)
EC <sub>13</sub>	< 118238000  Urogenital finding (finding) : REF	1072	6767	761 (27)
EC <sub>14</sub>	< 822987005  Finding of abdominopelvic segment of trunk (finding) : REF	1072	15335	810 (30)
EC <sub>15</sub>	< 302292003  Finding of trunk structure (finding) : REF	1072	23969	837 (32)
EC <sub>16</sub>	< 301857004  Finding of body region (finding) : REF	1072	63109	917 (41)
EC <sub>17</sub>	< 118234003  Finding by site (finding) : REF	1072	74251	937 (32)
EC <sub>18</sub>	< 404684003  Clinical finding (finding) : REF	1072	115537	1008 (47)
EC <sub>19</sub>	< 138875005  SNOMED CT Concept : REF	1072	354383	1228 (45)

Table 10. Result subset size, domain size, execution time and standard deviation (in milliseconds) of EC<sub>11</sub> to EC<sub>19</sub> set of equivalent ECs. "REF" is equivalent to "363698007 |Finding site (attribute)| = < 127903009 |Male genital organ structure (body structure)|"

We see that the substrate grows by 29067.32% (from 1215 to 354383 nodes), while execution time only grows by 71.27% (from 717 to 1228 milliseconds). Additionally, the same experiment was performed by adding a cardinality constraint to the EC (i.e., "[2..\*] 363698007 |Finding site (attribute)|"), which defines those findings located in two or more male genital organ structures. We obtained similar results, since execution time increased by 92.12% (from 647 to 1243 milliseconds). Results of Table 10 are presented in Figure 61.

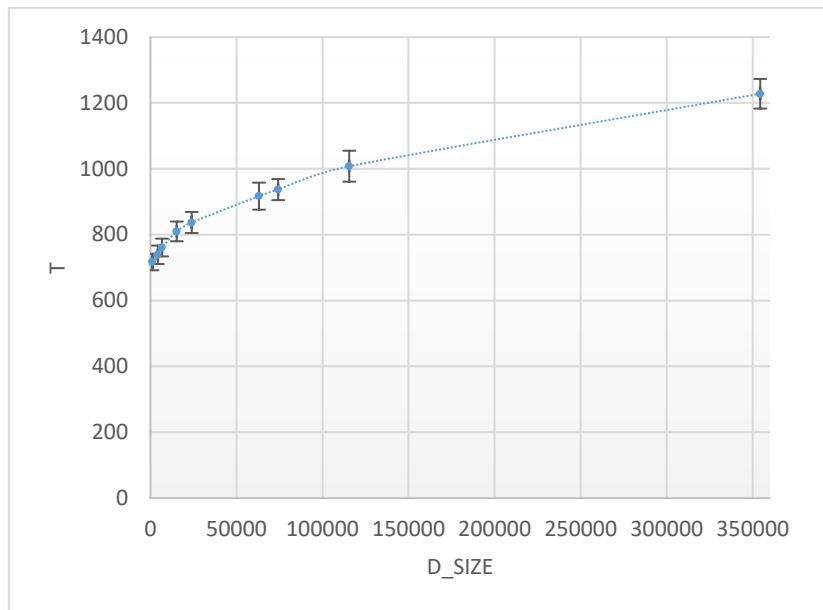


Figure 61. Scatter plot showing the relation between domain size and execution time, and standard deviation error bars (in milliseconds) of ECs in Table 10

In view of these results, we conclude that execution time of ECs does not increase at the same rate as the substrate size does (i.e., execution time increase is not directly proportional to database size increase), but it has a negligible growth compared to that of the substrate. This experiment shows the suitability of representing an ontological model, such as SNOMED CT, in a graph database, since the database size, the result subset size or the domain size does not affect performance, as suggested by Table 9 and Table 10.

#### 4.4 DISCUSSION

In health institutions generally coexist many different information systems, so information is stored in separate islands. Moreover, there is a high degree of fragmentation between institutions regarding EHR data. When patient information is needed, querying only a part of the information is a potential risk for the patient. There are, thus, great benefits in the design and construction of semantically interoperable information systems able to share, aggregate, analyse and use external information automatically and in a meaningful way [27].

The need to achieve high levels of semantic interoperability in the health domain is regarded as a crucial issue. However, it is not an easy task. For this purpose, it is an essential requirement to bind information models to terminological models (i.e., clinical terminologies such as SNOMED CT, LOINC or ICD, among others). This bind is known as terminology binding, and it is important since the link between information model and terminology is significant in achieving specific business or clinical objectives: data capture, retrieval and querying, information model library management and, as aforementioned, semantic interoperability.

There exist two types of terminology binding. First, semantic binding (also known as model meaning binding), gives unequivocal meaning to the information structures contained in the information model by means of a link between an element of the model and a pre- or post-coordinated term of the terminology. Semantic binding between archetypes and pre-coordinated terms can be carried out manually by health terminology experts, which is a hard task due to the large size of terminologies; and semi-automatically, by means of string comparison-based and semantic techniques that obtain a subset of candidate terms [42–45]. Second, content binding (also known as value set binding) constrains the set of possible coded values or clinical meanings of a data element within an information model. Content binding requires a mechanism to specify sets of terms intensionally (i.e., by means of an expression whose execution returns the set of terms). It is also possible to define the sets extensionally (i.e., by enumeration) but it has some disadvantages, such as the difficulty in the maintenance or the uphill task of managing large sets of terms [46]. One way to define such intensional sets is using a declarative language to interrogate the substrate of the terminology. The syntax of the language must allow the specification of the terms to be selected and included in the set and the way these terms must be related. Moreover, the operators of the language must be aligned with the logical model of the terminology. In this sense, ECL is a development of SNOMED International that enables the intensional definition of sets of clinical meanings through ECs.

Furthermore, data instances need to be validated against the information model. In this sense, if the information model contains constraint bindings to SNOMED CT subsets defined through ECs, it is also possible to validate the contents of data instances with the given subsets.

Regarding the representation and storage of SNOMED CT in a database using its RF2 files, we have explored several options. First, we focused on a relational database and SQL as a query language. However, this option was dismissed because of the complexity in the ECL to SQL translation process and the lack of efficiency when querying the data [33]. The natural way to store the concepts and relationships of SNOMED CT was established to be a graph, given its acyclic directed graph structure. Again, there was a range of possibilities, such as creating an XML database and using the XQL language (XML Query Language) for queries, or creating a database of RDF (Resource Description Framework) graph and using the SPARQL (Protocol and RDF Query Language) as a query language. We found some drawbacks in the literature in the use of Semantic Web technologies when dealing with large ontologies such as SNOMED CT. The main one is related to inference performance over the EL+ subset used in SNOMED CT. Previous studies [47,48] showed big differences on query execution times between reasoners even for simple queries, such as computing the descendants of a concept. Moreover, in some cases, execution times were significantly high. In addition, one of the studies [48] showed a correlation between the execution times of the queries and the number of returned subclasses. Additionally, when translating the SNOMED CT Query Language into SPARQL 1.1 under the RDF entailment regime [47], the authors found that the significantly larger number of pre-computed files required to generate the inferred model was a drawback. Considering the previous shortcomings and the performance of graph databases, it was determined that Neo4j and Cypher could be used for the execution of ECs [49]. The expressiveness of Cypher makes it possible to translate an EC into a set of Cypher queries whose execution on a property graph produces the subset defined by the source expression. It should be noted that there exist multiple graph databases, such as Amazon Neptune, ArangoDB, Dgraph, or JanusGraph. The choice of Neo4j was based on the availability of an open source edition and on the expressive power of the Cypher query language. Regarding the Neo4j database, it was possible to model SNOMED CT in different ways. Potentially, the model affects the execution time of the queries. We tested two different approaches that differ in the way SNOMED CT attribute relationships are stored. In the first one, attribute relationships were modelled using one type of Neo4j relationship (i.e., “[:REL]”) with a property that stored the identifier of the attribute relationship (e.g., “[:REL {releid: '363698007'}]”). In the second one, each attribute relationship was modelled using a different type of Neo4j relationship (e.g.,



“[:REL363698007]”). All ECs executed in this paper use the latter approach. After a comparative analysis of EC execution for both approaches, we concluded that there were no significant differences, except in one particular type of refined EC. Concretely, refinements that include attribute cardinalities and “not equal” comparison operator. In this case, the second approach was four times faster on average than the first one. As mentioned, the modelling of SNOMED CT properties and relationships in the database, as well as the simplification methods described in this paper, potentially have an impact on the execution times of the ECs.

Both simplification methods and visual representation of subsets presented in this chapter are useful for validating the consistency of extensional refsets and the terminology itself. Extensional refsets can have several potential problems, such as the existence of homograph words that cause the choosing of wrong FSN and, hence, wrong concept identifiers. The validation process is also used to search for outlier concepts (i.e., concepts included in the refset/terminology without any relation with the global meaning of the refset/sub-hierarchy). For instance, the visualization of the “900000101000122100 |Scales and reference clinical assessment systems for primary care reference set (foundation metadata concept)|” refset of the Spanish SNS, shows that 60 out of 61 concepts are subtypes of the “254291000 |Staging and scales (staging scale)|” hierarchy. The remaining concept “225392000 |Pressure ulcer risk assessment (procedure)|” is included in the “71388002 |Procedure (procedure)|” hierarchy. Therefore, it is a potential error of classification. A useful application of our EC post-execution simplification method is to generate simple ECs equivalent to extensional refsets. For example, the application of the simplification process to the “900000191000122105 |Severity indicator reference set (foundation metadata concept)|” SNS refset yields as a result the EC “< 272141005 |Severities (qualifier value)|”.

Moreover, the “900000021000122107 |Diabetes diagnoses reference set (foundation metadata concept)|” SNS refset contains almost the 80% of the concepts in “< 73211009 |Diabetes mellitus (disorder)|” SNS sub-hierarchy. Hence, the remaining 20% might be proposed to be added to the refset. Additionally, 14% of the concepts in the refset are not included in the sub-hierarchy, so that they could be inspected to detect potential errors.

Regarding the validation of the terminology itself, taking EC example 2 as an example, it would be expected that all the member concepts are descendant of “27550009 |Disorder of blood vessel (disorder)|”. The simplification of EC example 2 yields as result EC example 1. Therefore, we can asseverate that all disorders of body system located in a blood vessel structure are descendants of “27550009 |Disorder of blood vessel (disorder)|”. Furthermore, all the members of the “<< 27550009 |Disorder of blood vessel (disorder)|” sub-hierarchy contain at least a location that is a blood vessel structure.

It should be noted that the expressive power of ECL allows defining in an intensional way complex subsets and extensional value sets used in clinical practice. For instance, the NLM Value Set Authority Center [50] provides an extensive set of refsets, such as the ‘Head and Neck Anatomic Locations’ value set, which could be expressed by the EC:

<pre>&lt;&lt; 69536005  Head structure (body structure)  OR &lt;&lt; 45048000  Neck structure (body structure) </pre>
---

or the ‘Rheumatoid Arthritis’ value set, which could be expressed as:

<pre>&lt;&lt; 69896004  Rheumatoid arthritis (disorder) </pre>
--

Currently, there exist several efforts focused on the management of large collections of value sets to support effective and interoperable health information exchange[50–55]. In this regard, it should be noted that the proposed methods for the validation and visualization of SNOMED CT refsets could be applied to the representation and validation of value sets if their content is mapped to SNOMED CT concepts.

It is important to mention that there exist more implementations of the ECL [40,56–58]. At this point, these engines are useful to execute ECs. However, they do not provide other capabilities, such as to validate ECs from a semantic point of view, to include strategies and methods to simplify the ECs or to visually represent the subsets in order to understand the data and to validate them. Our effort has focused not only on the execution of ECs but on all the mentioned abilities.

SNQuery is not directly extensible to other terminologies and ontologies, since ECL is intended to express SNOMED CT concept model constraints. However, it is possible to indirectly query other terminologies by using mappings from/to SNOMED CT, such as the official map from SNOMED CT to ICD-10 delivered by SNOMED International, which is incorporated in SNQuery.

In the following chapter we present the requirements for an expression language able to specify consistency rules in archetypes. We also do a literature review in order to look for the language for expressing clinical logic that best suits our particular scenario.



## CHAPTER 5. CONSISTENCY RULES IN ARCHETYPES

---

### 5.1 INTRODUCTION

The healthcare sector is increasingly producing and consuming larger amounts of data and information. Exchanging information in a meaningful way is a critical issue, although this objective is still far from being sufficiently addressed. The predictable change towards personalized/precision medicine will lead to a drastic increase of the size and complexity of EHR systems, which will, in turn, affect the integration of clinical data. Therefore, reaching a high level of semantic interoperability is one of the most important challenges in achieving meaningful use of EHR.

In this sense, it is necessary to ensure semantic consistency of EHR data as a basic and primary requirement to improve healthcare, as well as for secondary uses of EHR data, such as clinical research and academic. Since consistency is mainly dependent on the usage scenario, it is an essential requirement to explicitly and formally define the constraints that the data must satisfy to be considered consistent, including the agreement or compatibility among the data elements [1]. The measure of consistency is generally based on elements contained by the EHR, but some researchers also include information from other sources, such as billing information, paper records, patient-reported data, and physician-reported data [1]. From a clinical point of view, one of the approaches for assessing consistency is to study the concordance between the elements of EHR; for example, diagnostics and associated information, such as treatments or procedures (e.g. a stroke can be associated with a set of radiology procedures, including any type of computed tomography of head, a magnetic resonance imaging of head and a doppler ultrasonography of carotid arteries). Therefore, it is mandatory to have a detailed formal specification of knowledge of the clinical domain, such as consistency rules.

It is important to emphasize that there exist three fundamental pillars on which semantic interoperability sits: a reference model plus detailed clinical information models (i.e., two level information model) and clinical terminology [59]. Moreover, it is essential to bind clinical information models with terminology by means of semantic and value set terminological binding as an interface between them. Furthermore, SNOMED CT ECL can

be used to define SNOMED CT subsets of clinical concepts which are useful in value set binding. Hence, one primary requirement of information model consistency rules is the ability of handling with terminological subsets in order to define both value set binding and conditional value set binding.

The specification of consistency rules, including simple, conditional and dependency intensional value set bindings to enrich the definition of clinical information models have not been deeply explored. New proposals are required in order to formally specify domain knowledge. These rules will provide some advantages in data entry, validation, processing and interpretation of EHR data. Some EHR information modelling approaches can accommodate such rules. For instance, the rules section of ADL2 archetypes –previously known as invariant section in ADL 1.4.

In order to define consistency rules, including value set bindings, it is required a formal language. In this chapter, we analyse the requirements for its definition based on a literature review of existing medical logic languages, including PROforma, Asbru, GLIF3, SAGE, EON, Arden Syntax, GELLO, GDL and openEHR EL. After the review, we study the possibility of using GDL and EL as a first approximation for expressing consistency rules, including simple, conditional and dependency intensional value set binding rules. The rationale was that both languages are rule-based with inbuilt mechanisms for accessing archetypes instances and they can accommodate terminology bindings. The limitations encountered are considered as a point of departure for creating a new sub-language able to be embedded in both, GDL and EL, to meet all the requirements. This language is proposed for the definition of consistency rules, including simple, conditional and dependency intensional value set bindings. Furthermore, as a part of this work, it is necessary to study the potential locations for these rules and the ways they can be represented: in the information model, as a new artefact of the terminological model, as independent rules or as a combination of some of them.

## 5.2 REQUIREMENTS FOR THE CONSISTENCY LANGUAGE IN ARCHETYPES

The specification of semantic consistency rules to enrich the definition of information models such as archetypes has not been deeply explored to date. The incorporation of these consistency rules will provide some advantages to validate, process and interpret EHRs. Likewise, these semantic consistency rules will ease the validation and measurement of the consistency of data contained on EHRs by means of consistency metrics.

In this chapter, our objective is to propose a language to specify semantic consistency rules to improve the consistency of EHRs. This language will also allow the definition of aggregation functions and conditional value set bindings between information models and clinical terminologies, such as SNOMED CT.

To define this language, some languages with similar purposes have been studied. Specifically, some of the data types and operators of GELLO and openEHR Expression Language have been identified to be useful for this language. However, we have adopted and adapted our own variation of these operators and data types in some cases.

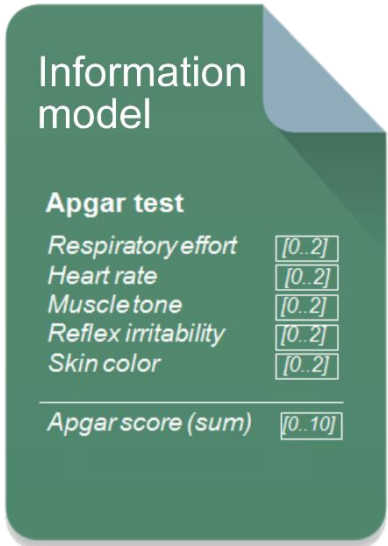
### 5.2.1 Introduction

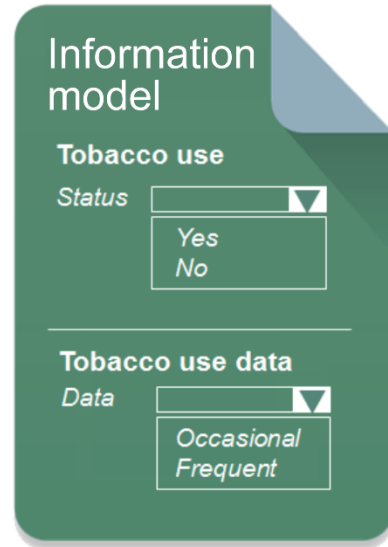
The language has two types of statements:

- *Constraints*: relational and logical expressions that must be satisfied by the data instance of the information model.
- *Rules*: constraints that must be satisfied by the data instance based on a condition, which is also a constraint.

Note that a constraint can be understood as a particular case of a rule where its condition is fixed to True. Both constraints and rules are evaluated to True or False.

Examples:

 <p><b>Information model</b></p> <p><b>Apgar test</b></p> <p>Respiratory effort [0..2]  Heart rate [0..2]  Muscle tone [0..2]  Reflex irritability [0..2]  Skin color [0..2]</p> <hr/> <p>Apgar score (sum) [0..10]</p>	<ul style="list-style-type: none"> <li>▪ <b>Information model:</b> <i>Apgar test</i></li> <li>▪ <b>Type of statement:</b> constraint</li> <li>▪ <b>Description:</b> ‘Apgar score (sum)’ value must be the sum of all the measurements obtained after performing the test to a patient. To specify this constraint, it is required to equate the resulting sum of all the measurements to ‘Apgar score (sum)’ value. If the constraint is satisfied then it is evaluated to True. In other case, it is evaluated to False.</li> </ul>
--	--

 <p><b>Information model</b></p> <p><b>Tobacco use</b></p> <p>Status <input type="text" value="▼"/>  Yes  No</p> <hr/> <p><b>Tobacco use data</b></p> <p>Data <input type="text" value="▼"/>  Occasional  Frequent</p>	<ul style="list-style-type: none"> <li>▪ <b>Information model:</b> <i>Tobacco use</i></li> <li>▪ <b>Type of statement:</b> rule</li> <li>▪ <b>Description:</b> if ‘Status’ value is ‘Yes’ (this comparison is defined by a constraint) then ‘Data’ value must contain some of the available options (this is also defined by a constraint). If both constraints are evaluated to True, then the rule is evaluated to True.</li> </ul> <p><i>(To see all other True/False combinations, please read the RULES section).</i></p>
--	--

In addition to the use of numerical and string values, the language can read values from data instances. These values are referenced by means of *paths*. So that, paths are used to compound the expressions of the language.

Furthermore, *variable declaration* and *assignment* are supported to provide clarity to expressions. On the contrary, since expressions contain paths, they can become long and unclear to the naked eye.



There are two types of variables. First, *primitive* data types variables (e.g. integer, float, char, boolean, etc.) and second, data types defined on a *reference model* (e.g. basic types, codes and texts, magnitudes, time values, etc.).

### 5.2.2 Constraints

There are two types of constraints:

- *Relational*: expressions that use relational operators (i.e., =, !=, <, >, <=, >=).
- *Logical*: expressions that combines relational expressions using logical operators (i.e., AND, OR, XOR, NOT).

There are also *arithmetic* expressions. These type of expressions make use of arithmetic operators (i.e., +, -, \*, /, ^) and they return a numerical value rather than being evaluated to True or False. So that, they can't be considered constraints but they are part of relational expressions and these in turn are part of logical expressions.

Examples:

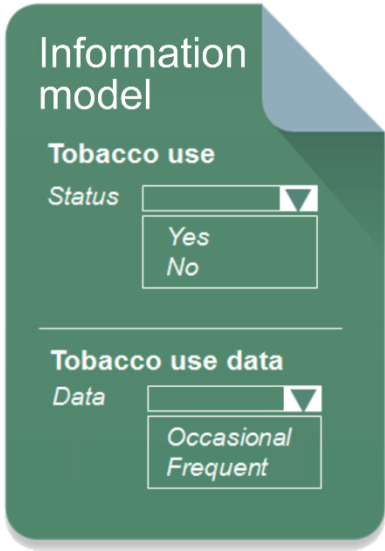
Expression type	Definition	Evaluation
Arithmetic	$3+5-8*9$	-64
Relational	$3+5-8*9 < 1000$	True
Logical	$(3+5-8*9 < 1000) \text{ AND } (10 \leq 5)$	False

Note: a precedence order is established to operators (it will be defined later). This order can be altered using parentheses.

### 5.2.3 Rules

In general, a rule is a structure of the form IF-THEN [-ELSE]. Some of the constraints we need to specify must be evaluated based on a condition (e.g. conditional value set binding). Both, the antecedent of the rule (IF) and the consequents (THEN and optionally ELSE) are constraints.

Example:

 <p><b>Information model</b></p> <p><b>Tobacco use</b></p> <p>Status <input type="text" value="Yes"/> Yes No</p> <hr/> <p><b>Tobacco use data</b></p> <p>Data <input type="text" value="Occasional"/> Occasional Frequent</p>	<ul style="list-style-type: none"> <li>▪ <b>Information model:</b> <i>Tobacco use</i></li> <li>▪ <b>Type of statement:</b> rule</li> <li>▪ <b>Description:</b> if 'Status' value is 'Yes' (this comparison is defined by a constraint) then 'Data' value must contain some of the available options (this is also defined by a constraint).</li> <li>▪ <b>Antecedent constraint of the rule (IF):</b> 'Status' value is equal to 'Yes'.</li> <li>▪ <b>Consequent constraint of the rule (THEN):</b> 'Data' value is equal to 'Occasional' or 'Frequent'.</li> <li>▪ <b>Consequent constraint of the rule (ELSE):</b> 'Data' is empty.</li> </ul>
--	--

In our case, the structure of a rule is of the form: *IF constraint1 THEN constraint2 [ELSE constraint3]*, where *constraint1*, *constraint2* and *constraint3* are evaluated to True or False based on the values read from the data instance of the information model. Depending on the combination of these evaluations, the rule is evaluated to True or False as it is shown in the following truth table for the *Tobacco use* example:

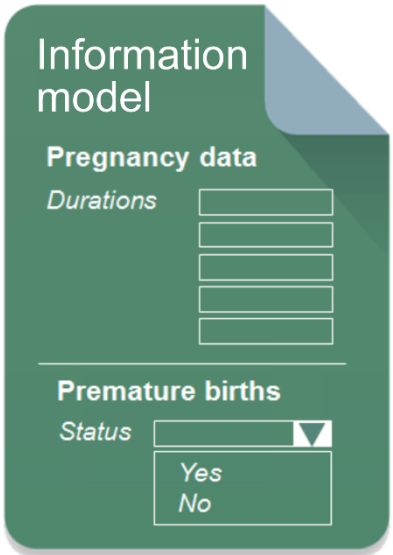
Evaluation of the antecedent (IF)	Evaluation of the consequent (THEN)	Evaluation of the consequent (ELSE)	Evaluation of the rule
V ('Status' is 'Yes')	V ('Data' is 'Occasional or Frequent')	-	V
V ('Status' is 'Yes')	F ('Data' is empty)	-	F
F ('Status' is 'No')	-	V ('Data' is empty)	V
F ('Status' is 'No')	-	F ('Data' is 'Occasional or Frequent')	F

### 5.2.4 Exists operator

The 'EXISTS' operator (logic existential quantifier) has two distinct usages.

- Usage 1: to check the existence of a path on the data instance of the information model.
- Usage 2: it is applied on lists to check that at least one element satisfies a specific condition/constraint.

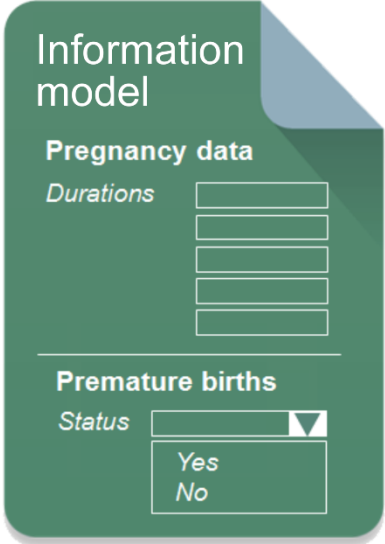
Usage 2 example:

 <p><b>Information model</b></p> <p><b>Pregnancy data</b></p> <p><i>Durations</i></p> <p><input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/></p> <hr/> <p><b>Premature births</b></p> <p><i>Status</i></p> <p><input type="text" value="Yes"/> <input type="text" value="No"/></p>	<ul style="list-style-type: none"> <li>▪ <b>Information model:</b> <i>Premature births</i></li> <li>▪ <b>Type of statement:</b> rule</li> <li>▪ <b>Description:</b> if at least one element from the 'Durations' list is less than 37 weeks (represented using a constraint with 'EXISTS'), then 'Status' must contain a 'Yes' value. In other case, 'Status' must be 'No'.</li> <li>▪ <b>Antecedent constraint of the rule (IF):</b> there exist at least one element in 'Durations' whose value is less than 37.</li> <li>▪ <b>Consequent constraint of the rule (THEN):</b> 'Status' is equal to 'Yes'.</li> <li>▪ <b>Consequent constraint of the rule (ELSE):</b> 'Status' is equal to 'No'.</li> </ul>
---	--

### 5.2.5 For\_all operator

The 'FOR\_ALL' operator (logic universal quantifier) is used on lists to check that all the elements satisfy a specific condition/constraint.

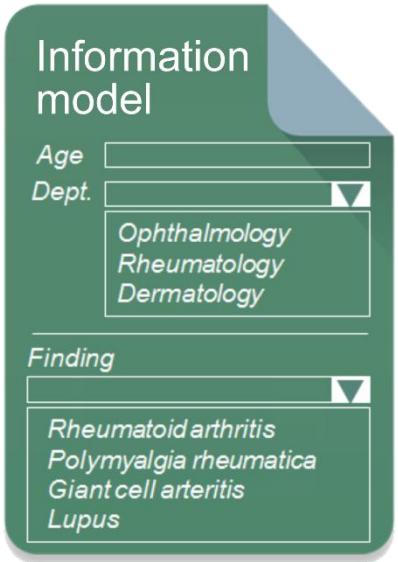
For example:

 <p><b>Information model</b></p> <p><b>Pregnancy data</b></p> <p><i>Durations</i> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/></p> <hr/> <p><b>Premature births</b></p> <p><i>Status</i> <input type="text"/> Yes No</p>	<ul style="list-style-type: none"><li>▪ <b>Information model:</b> <i>Premature births</i></li><li>▪ <b>Type of statement:</b> rule</li><li>▪ <b>Description:</b> if all the elements from the 'Durations' list are greater or equal than 37 weeks (represented using a constraint with 'FOR_ALL'), then 'Status' must contain a 'No' value. In other case, 'Status' must be 'Yes'.</li><li>▪ <b>Antecedent constraint of the rule (IF):</b> all the elements from the 'Durations' list are greater or equal than 37.</li><li>▪ <b>Consequent constraint of the rule (THEN):</b> 'Status' is equal to 'No'.</li><li>▪ <b>Consequent constraint of the rule (ELSE):</b> 'Status' is equal to 'Yes'.</li></ul>
---	---

### 5.2.6 Inclusion operator

The 'IN' operator is used to check whether a concept of certain clinical terminology (e.g. SNOMED CT) is included into a set of concepts of that terminology. This set is defined intensionally by means of an EC in the case of SNOMED CT. The 'IN' operator is especially useful for creating constraints and rules for (conditional) value set binding validation.

Example:

 <p>The image shows a green information model form with a white border and a folded top-right corner. It has three sections: 'Age' with a text input field; 'Dept.' with a dropdown menu showing 'Ophthalmology', 'Rheumatology', and 'Dermatology'; and 'Finding' with a dropdown menu showing 'Rheumatoid arthritis', 'Polymyalgia rheumatica', 'Giant cell arteritis', and 'Lupus'.</p>	<ul style="list-style-type: none"> <li>▪ <b>Information model:</b> <i>Findings by age and department</i></li> <li>▪ <b>Type of statement:</b> rule</li> <li>▪ <b>Description:</b> if 'Age' value is greater than 75 and 'Dept.' is 'Rheumatology' (represented using a constraint), then 'Finding' must contain a concept included in the rheumatic diseases in geriatrics set (represented using a constraint with 'IN').</li> <li>▪ <b>Antecedent constraint of the rule (IF):</b> 'Age' value is greater than 75 and 'Dept.' is 'Rheumatology'.</li> <li>▪ <b>Consequent constraint of the rule (THEN):</b> 'Finding' is a concept which is included in the rheumatic diseases in geriatrics set.</li> </ul>
--	---

The structure of this rule is of the form: *IF constraint1 THEN constraint2*, where *constraint2* uses the 'IN' operator as follows: 'Finding' IN 'Rheumatic diseases in geriatrics set'.

### 5.2.7 Considerations about the use of 'EXISTS', 'FOR\_ALL' and 'IN'

Internally, these three operators make use of logical operators.

- 'EXISTS' is a succession of 'OR' (*element1 satisfies condition OR element2 satisfies condition OR... elementN satisfies condition*).
- 'FOR\_ALL' is a succession of 'AND' (*element1 satisfies condition AND element2 satisfies condition AND... elementN satisfies condition*).

In the case of 'IN' it is performed an intersection between the concept and the set. For this purpose, it is used the 'AND' operator. In the case of SNOMED CT the evaluation is executed externally using SNQuery Execution Engine which implements some optimizations for the AND operator.

Therefore, apart from 'IN' (which is executed externally), the functionality of 'EXISTS' and 'FOR\_ALL' can be achieved using constraints (logical expressions) of the language itself. So that, it is possible to dispense with them. However, it can become much more comfortable to use them since it can be used lists with a lot of elements. One proposal about their use in the language is to treat them as functions. For example:

- IN (*concept, set*)
- EXISTS (*path*)
- EXISTS (*list, condition*)
- FOR\_ALL (*list, condition*)

### 5.2.8 Considerations about paths

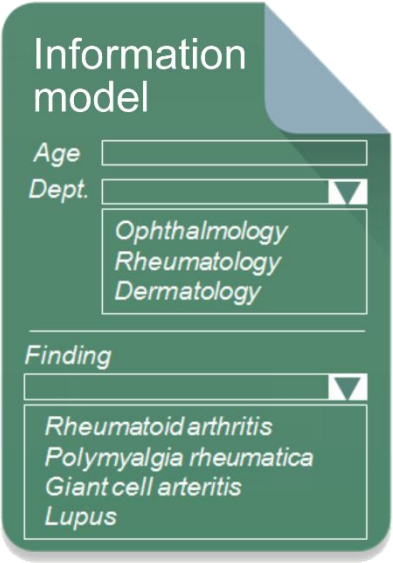
Regarding the use of paths, it is mandatory to allow setting contexts in order to define groups of expressions that are related. A context should be defined by means of a path. Additionally, the definition of contexts should be carried out based on conditions over them, such as selecting several child elements or setting conditions of element values. Finally, it is important to allow raw paths to be used directly as variables to generate expressions directly using paths rather than introducing variables.

### 5.2.9 Considerations about nesting rules

A rule is a mechanism that forces the accomplishment of a certain constraint according to a condition (remember that the condition is also a constraint). In general, there may be several conditions. The constraint forced to be satisfied depends on which condition is evaluated to True. This mechanism can be achieved by concatenating rules (i.e., IF-THEN, IF THEN, IF-THEN...). However, it is more efficient to nest them (i.e., IF-THEN-ELSE (IF-THEN-ELSE (IF-THEN-ELSE)...)). In this way, the evaluation process is carried out until one of the conditions is evaluated to True (in the case of concatenated rules, it is evaluated the condition of each and every rule, although it is known a priori that, at most, only one of the conditions will be evaluated to True).

Nested rules are useful for conditional value set binding as it is shown in the example below.

Example:

 <p>The image shows a green document icon titled 'Information model'. It contains three input fields: 'Age' with a text box, 'Dept.' with a dropdown menu showing 'Ophthalmology', 'Rheumatology', and 'Dermatology', and 'Finding' with a dropdown menu showing 'Rheumatoid arthritis', 'Polymyalgia rheumatica', 'Giant cell arteritis', and 'Lupus'.</p>	<ul style="list-style-type: none"> <li>▪ <b>Information model:</b> <i>Findings by age and department</i></li> <li>▪ <b>Type of statement:</b> rule</li> <li>▪ <b>Description:</b> if 'Age' value is greater than 75 and 'Dept.' is 'Rheumatology' (represented using a constraint), then 'Finding' must contain a concept included in the rheumatic diseases in geriatrics set (represented using a constraint with 'IN'). But there are other possible combinations. For example: 'Age' is greater than 75 and 'Dept.' is equal to 'Dermatology'. In this case, 'Finding' must contain a concept included in the dermatological diseases in geriatrics set. In general, according to the values of 'Age' and 'Dept.', 'Finding' should be included in one set of diseases or another.</li> </ul>
---	---

The structure of this rule is as follows:

```

IF constraint1 THEN constraintA ELSE
  (IF constraint2 THEN constraintB ELSE
    (IF constraint3 THEN constraintC ELSE
      (and so on...))
    )
  )

```

### 5.2.10 Data types, operators and functions

Regarding data types, the language must allow using the common primitive data types in most of programming languages, such as Integer, Real, Boolean, String or lists of any primitive type. Additionally, it should add two new data types: *Terminology\_code*, to be bound to terminological codes; and *Snomed\_ec*, to be bound to SNOMED CT ECs.

The language also should have the ability to call usual basic functions, such as math functions, text functions, date and time functions and aggregate functions. Additionally, it should include functions to check whether a path exists or is empty.

Regarding operators, there should be supported arithmetic, relational and boolean operators, as well as logical quantifiers. Additionally, an inclusion operator should be supported in order to evaluate whether a terminological code is included or not in a subset of codes defined by means of a SNOMED CT EC.

### 5.3 REVIEW OF EXISTING LANGUAGES

There is a myriad of languages intended to formally represent clinical knowledge, such as that present in clinical guidelines (CGs) [60,61] and in clinical trial eligibility criteria [62]. We identify several task-oriented knowledge formalisms that have been developed over the years, including but not limited to PROforma[63,64], Asbru [65,66], GLIF3 [67–69], SAGE [70,71] and EON [72,73]. The purpose of these languages is to capture the content and structure of CGs and eligibility criteria by means of Task-Network Models (TNMs) in a computer-processable way, leading to the definition of Computer Interpretable Guidelines (CIGs) and computable knowledge representations for eligibility criteria. TNMs hierarchically decompose CGs algorithms into networks of component tasks (i.e., plans, actions, decisions and enquiries, among others, depending on the particularities of the language) that unfold over time [74] (see Figure 62). They represent CGs formally allowing a CIG execution engine to execute the represented knowledge over patient-specific data. Task-oriented models control the flow over the task-network for modelling the guidelines (who does what, when, how and where).

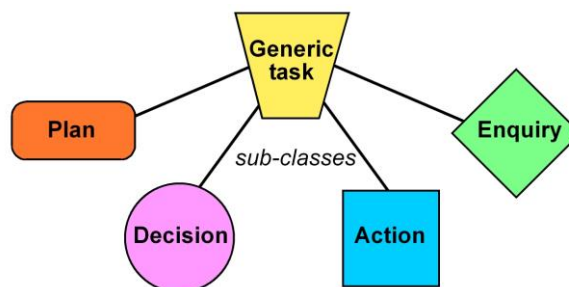


Figure 62. The PROforma task model



On the other hand, such clinical knowledge is capable of being formally modelled by means of rule-based technologies [75] instead of using TNM formalisms. A rule formalism is a logic-based language where each rule can be interpreted as an “if-then” statement. Rules can be interpreted by an inference engine and are easy to share and implement in different clinical environments. In [75] it was established two main groups of rule formalisms that can potentially model CGs knowledge: first, Rule Based Systems (RBSs), which comprise both medical oriented rules and general-purpose production rules; and second, Semantic Web languages with rule extensions.

A well-known medical oriented rule formalism is the Arden Syntax [76–81], which is used to define rules as Medical Logic Modules (MLMs) that are executed in CDSS. Arden Syntax is a widely recognized HL7 standard for representing clinical and scientific knowledge, and can deal with terminologies such as SNOMED CT [82]. An early version of GLIF, which was called Guideline Expression Language (GEL), used it as its expression language [68]. When developing an MLM using the Arden Syntax, the patient data model is local to the institution, so it is difficult to share MLMs and knowledge bases between organizations (i.e., the “curly braces problem”). HL7 standard information models, such as the Virtual Medical Record (vMR) and FHIR, have been proposed to serve as standard data models for the Arden Syntax.

Our survey includes other languages, such as the GELLO Expression Language [83]. GELLO is an object-oriented constraint language based on the Object Constraint Language (OCL) [84]. It is intended to be a standard expression language for Clinical Decision Support (CDS) and it can be used with any object-oriented data model. Decision rules, eligibility criteria and patient state specifications can be represented with GELLO expressions. GLIF3, which is an object-oriented model, uses GELLO as the expression language for specifying the logic of the task-network flow.

Additionally, we analysed the openEHR Guideline Definition Language (GDL), which is a rule-based language that expresses CDS logic as production rules. It is agnostic to languages and reference terminologies. Its rules can be combined together as building blocks to support single decision making and more complex, chained decision-making processes.

GDL rules can be used to drive at-point-of-care CDSS as well as retrospective populational analytics [85]. openEHR has recently launched the Guideline Definition Language v2 (GDL2), which introduces some changes respect to the original release of GDL, such as data binding agnostic to EHR data models.

Besides, a recent proposal by openEHR that is still under development, the Expression Language (EL), can be used to define rules in archetypes and expressions within decision logic models (DLMs) designed for use with openEHR Task Planning (TP), among others. It is based on a limited first-order predicate logic language with extensions for numeric sub-expressions and it is suitable to be included in other expression- and rule-based formalisms, such as GDL [86].

Due to our need to represent clinical knowledge as expressions and rules as the core part of the present chapter, we have selected, as first filtering, the presented rule-based languages as the most suitable for our purposes, instead of TNM formalisms, which suit better for modelling plans, decisions and actions of CGs.

*Table 11* shows a comparative and additional information about Arden Syntax, GELLO, GDL and EL in terms of type of model, patient data standards and use of terminologies, including the ability to deal with semantic binding and simple, conditional and dependency extensional and intensional value set binding.

	Arden Syntax	GELLO	GDL	EL
Model	Rule-based	Expression-based <i>Definition of rules is also supported.</i>	Rule-based	Expression-based <i>Support to “implies” logical operator, equivalent to “if-then”.</i>
Patient data standards	No <i>Not designed to lead with a particular standard patient data model. Patient data access is local to the institution (i.e., the curly braces problem).</i>	Yes <i>Can fully provide expression support for any properly defined view of the HL7 RIM (such as vMR), regardless of any particular specification of an object-oriented data model.</i>	Yes <i>Agnostic to EHR data models, such as openEHR and ISO13606 archetypes, and HL7 FHIR resources.</i>	Yes <i>openEHR and ISO13606 archetypes.</i>
Semantic binding	Yes <i>Although it does not use any standard clinical terminology, it is possible to encode query data elements in a standard way, using a terminology, such as SNOMED CT.</i>	Yes <i>Can query SNOMED CT and other reference terminologies.</i>	Yes <i>Support to standard terminologies (section “Ontology”, subsection “term_bindings”).</i>	Yes <i>By means of the “Terminology_code” primitive data type for terminology code references.</i>
Simple value set binding (extensional subsets)	Yes <i>By means of storing a list of terminology codes.</i>	Yes <i>By means of storing a list of terminology codes.</i>	Yes <i>It is possible to bind a locally defined term to multiple concepts/refsets defined by external reference terminologies.</i>	Yes <i>By means of a container type of terminology codes (e.g., List, Set...).</i>

Conditional value set binding (extensional subsets)	Yes <i>By means of using if-then rules with a list of terminology codes.</i>	Yes <i>By means of using if-then rules with a list of terminology codes.</i>	Yes <i>By means of using when-then rules.</i>	Yes <i>By means of using the "implies" logical operator.</i>
Dependency value set binding (extensional subsets)	No	No	No	No
Simple value set binding (intensional subsets)	No	No	No	No
Conditional value set binding (intensional subsets)	No	No	No	No
Dependency value set binding (intensional subsets)	No	No	No	No

Table 11. Comparison between general features of Arden Syntax, GELLO, GDL and EL

After the comparison analysis of the general features of the expression languages presented in *Table 11* (i.e., the Arden Syntax, GELLO, GDL and EL), it is the time to evaluate its suitability to be the basis for the archetype constraints and rules language.

First, we divide them into two groups regarding the ability to natively reference archetype elements through paths. The first group is arranged by the Arden Syntax and the GELLO expression language. None of them are conceived to work with paths; concretely, the Arden Syntax is not designed to lead with a particular standard patient data model but patient data access is local to the institution (i.e., the curly braces problem). On the other hand, GELLO can fully provide expression support for any properly defined view of the HL7 RIM (such as vMR), regardless of any particular specification of an object-oriented data model. The second group is composed by GDL and EL. Both languages are conceived to work with archetypes and hence to reference its elements via paths. Considering that the Arden Syntax and GELLO would add unnecessary complexity to the expression language, and that none of them is conceived to reference archetype elements through paths, it makes sense to focus on the second group, i.e., GDL and EL. The Arden Syntax and GELLO do not offer advantages when working with medical terminologies in general and with intensional subsets and value set bindings in particular. Moreover, GELLO incorporates most of Object Constraint Language (OCL) [84] functionality with the exception of some unneeded capabilities which have been removed, such as invariants -also for constraints. These features are not required because GELLO is designed for writing queries, and not for constraining models [83].

We have carried out a more detailed comparison between GDL and EL to find out which of them best suits the purpose we want to specify consistency rules in archetypes.

### 5.3.1 Comparison between GDL and EL

Since GDL and EL are the most suitable languages for the definition of consistency rules in archetypes, we have performed a comparison between them in terms of the necessary features for defining consistency rules. The result is summarized in *Table 12*. We also show some features that are not required at this point of the thesis but it might be studied and proposed for the language in the future, such as the ability to deal with several archetypes and to assign a priority to the constraints and rules. It should be noted that neither GDL

nor EL can define intensional subsets of terminology concepts and hence the ability to evaluate the membership of a given concept in such subsets.

	GDL <sup>19</sup>	EL <sup>20</sup>
<b>Variables and data types</b>		
Use of variables (type + name) <i>Required</i>	No <i>It uses 'gtXXXX' as name, and type is implicit. It is possible to associate it to a text in the term_definitions section.</i>	Yes <i>Variables are declared with a formal type.</i>
Generic List type (including Terminology_code type) <i>Required</i>	No	Yes
<b>Data access</b>		
Access to data using paths <i>Required</i>	Yes <i>By means of the archetype_bindings section.</i>	Yes <i>Bound variables include an assignment to a path within an assumed data context.</i>
Access several archetypes <i>Not required</i>	Yes <i>By means of the archetype_bindings section.</i>	No <i>Expressions are defined in the rules section inside an archetype.</i>
<b>Rules</b>		
IF-THEN[-ELSE] rules <i>Required</i>	Yes* <i>It uses WHEN-THEN, which is equivalent to IF-THEN. *WHEN-THEN-ELSE is not supported.</i>	Yes* <i>It uses IMPLIES, which is equivalent to IF-THEN. *IMPLIES-ELSE is not supported.</i>
Name for rules <i>Required</i>	No <i>It uses 'gtXXXX' as name. It is possible to associate it to a text in the term_definitions section.</i>	Yes
Criticality of rules (error/warning) <i>Required</i>	No	No

<sup>19</sup> GDL current specification document (May 2019)  
<https://specifications.openehr.org/releases/CDS/latest/GDL.html>

<sup>20</sup> EL current specification document (August 2021)  
<https://specifications.openehr.org/releases/BASE/Release-1.0.4/expression.html>

Priority for rules <i>Not required</i>	Yes <i>To ensure execution order of the rules.</i>	No
Logical quantifiers		
Universal quantifier (FOR_ALL) <i>Required</i>	Yes	Yes
Existential quantifier (THERE_EXISTS) <i>Required</i>	No	Yes
Natural language descriptions		
Description of elements <i>Required</i>	Yes <i>In the term_definitions section.</i>	Yes <i>By using comments.</i>
Terminologies		
Use of terminologies <i>Required</i>	Yes <i>In the term_binding subsection.</i>	Yes <i>By using the Terminology_code data type.</i>
ECL embedded <i>Required</i>	No	No
Inclusion operator (for value set binding) <i>Required</i>	No	No

Table 12. Comparison of features between GDL and EL

One of the key points when analysing the suitability of GDL and EL to choose one of them to be the basis of the consistency rules language in archetypes, is the ability to deal with medical standard terminologies in general, and with intensional subsets and value set bindings in particular. In both cases, it is possible to reference concepts from standard terminologies, such as SNOMED CT or ICD10. In the case of GDL, it provides the `term_binding` subsection (inside of the ontology section) that allows binding an element from the guide to a standard concept. For example:

```

term_definitions = <
  ["SNOMEDCT"] = (TERM_BINDING) <
    bindings = <
      ["gt0105"] = (BINDING) <
        codes = <[SNOMEDCT::389086002|Hypoxia|],...>
        uri = <"">
      >
    >
  >
>

```

On the other hand, EL allows creating variables of Terminology\_code type. For example:

```
$hypoxia_concept: Terminology_code := [snomed_ct::389086002|Hypoxia|]
```

Moreover, in GDL it is possible to bind a locally defined term to multiple concepts/refsets defined by external reference terminologies. This ability allows defining extensional lists of concepts and hence simple extensional value set bindings. For example:

```

term_definitions = <
  ["SNOMEDCT"] = (TERM_BINDING) <
    bindings = <
      ["gt0106"] = (BINDING) <
        codes = <[SNOMEDCT::51440002|Right and left|],
                 [SNOMEDCT::24028007|Right|],
                 [SNOMEDCT::7771000|Left|],...>
        uri = <"">
      >
    >
  >
>

```

EL also allows creating extensional lists of terminology concepts and therefore simple extensional value set bindings. In this case, EL makes us of container variables, such as lists of terminology codes. For example:



```

$laterality: List<Terminology_code>
              := [snomed_ct::51440002|Right and left|,
                  snomed_ct::24028007|Right|,
                  snomed_ct::7771000|Left|]

```

On the other hand, neither GDL nor EL allow specifying intensional subsets. This means that it is not possible to natively create any type of intensional value set binding, such as simple value set binding, conditional value set binding or dependency value set binding. Furthermore, neither GDL nor EL allow checking the membership of a given code into an extensionally or intensionally defined subset (an inclusion operator is not specified).

Hence, in terms of support to terminologies, from our perspective and the requirements for the semantic consistency constraints and rules language in archetypes, at this point both GDL and EL offer the same possibilities. We need to further analyse the rest of the components that we presented in Table 12.

Regarding the declaration of variables of a set of data types, GDL is not a typed language. This means that it uses 'gtXXXX' (i.e., guideline term XXXX) as name, and type is implicit. It is possible to associate it to a text in the term\_definitions section. For instance:

```

term_definitions = <
  ["en"] = (TERM_DEFINITION) <
    terms = <
      ["gt0003"] = (TERM) <
        text = <"Diagnosis">
      >
      ["gt0014"] = (TERM) <
        text = <"Hypertension">
      >
      ["gt0102"] = (TERM) <
        text = <"Diabetes">
      >
      ["gt0105"] = (TERM) <
        text = <"Atrial fibrillation">
      >
    >
  >
>

```

On the other hand, EL is strong typed, which means that it allows the declaration (and initialization) of typed variables (Integer, Real, String, Boolean, Terminology\_code, etc.),

including lists (List<Integer>, List<Real>, List<Boolean>, List<Terminology\_code>, etc.). For example:

```
$systolic: Integer := 118
$diastolic: Integer := 72
$normal_pressure: Boolean := $systolic < 120 AND $diastolic < 80
$normal_pressure_code: Terminology_code :=
[snomed_ct::2004005|Normal blood pressure (finding)]
$normal_pressure_text: String := 'Normal blood pressure'
```

In this case, EL suits better our purpose on creating consistency rules on archetypes since the declaration and use of typed variables is important when specifying constrains and rules over EHR data.

About referencing elements in the archetype by using paths (i.e., EHR data access), GDL allows it by means of the archetype\_bindings section. For example:

```
definition = (GUIDE_DEFINITION) <
  archetype_bindings = <
    [1] = (ARCHETYPE_BINDING) <
      archetype_id = <"openEHR-EHR-EVALUATION.problem-
diagnosis.v1">
      domain = <"EHR">
      elements = <
        ["gt0107"] = (ELEMENT_BINDING) <
          path = <"/data[at0001]/items[at0002.1]">
        >
      >
    >
  >
>
```

Note that it is necessary to set the name of the archetype (i.e., archetype\_id = <"openEHR-EHR-EVALUATION.problem-diagnosis.v1">) whose elements are being referenced since GDL allows referencing elements from multiple archetypes in order to create rules by combining data from several pieces of EHR (note that at this point of our thesis, referencing

elements from multiple archetype is not required). In contrast to GDL, EL only allows referencing elements from the host archetype of the constraints and rules ('Rules' section). For example (we assume that the host archetype is openEHR-EHR-EVALUATION.problem-diagnosis.v1):

```
$problem_diagnosis: String :=/data[at0001]/items[at0002.1]
```

Therefore, both GDL and EL allows accessing EHR data by means of paths to the archetype elements.

Regarding the specification of IF-THEN rules, both GDL and EL languages support them. In the case of GDL, it uses the syntax WHEN-THEN (inside of the rules section), which is equivalent to IF-THEN. Nesting of WHEN-THEN (i.e., WHEN-THEN-ELSE WHEN-THEN) is not supported. GDL does not allow the specification of a name for a rule directly. It uses 'gtXXXX' as name. It is possible to associate this name to a text in the term\_definitions section. GDL allows setting a priority of each rule to ensure execution order of the rules (note that this point is not required for the consistency rules language in archetypes). For example:

```
rules = <
  ["gt0018"] = (RULE) <
    when = <"$gt0108!=null",...>
    then = <"$gt0014=1|local::at0031|Present|",...>
    priority = (11)
  >
  ["gt0019"] = (RULE) <
    when = <"$gt0109!=null",...>
    then = <"$gt0010=1|local::at0034|Present|",...>
    priority = (9)
  >
  ["gt0026"] = (RULE) <
    then = <"$gt0016.magnitude=( ( ( ( (
(gt0009.value+$gt0010.value)+$gt0011.value)+$gt0015.value)+$gt0012.va
lue)+$gt0013.value)+$gt0014.value)",...>
    priority = (1)
  >
```

On the other hand, EL uses the syntax IMPLIES, which is equivalent to IF-THEN. Nesting of IMPLIES rules by means of IMPLIES-ELSE IMPLIES is not supported. In contrast to GDL, EL allows the specification of a name for a rule directly. For example:

```
Smoker_details_recorded: $is_smoker implies exists $smoking_details
```

Neither GDL nor EL supports the definition of a criticality of rules (i.e., error and warning). This means that it is not possible to separate rules into two groups: those that will throw an error if they are not satisfied by the EHR data, and those that will result in a warning if they are not satisfied by the EHR data. This functionality should be supported by the semantic constraint and rules language in archetypes since we need to analyse and improve the consistency of EHR. Some of the EHR data will be considered as critical data and rules involving them should be satisfied, while other type of data will be considered as secondary data and rules involving them will not result in an inconsistent EHR if they are not satisfied.

Regarding logical quantifiers, GDL supports the universal quantifier FOR\_ALL, while EL allows using both FOR\_ALL universal quantifier and THERE\_EXISTS existential quantifier. For example:

```
there_exists v : container_var | <Boolean expression mentioning v>  
for_all v : container_var | <Boolean expression mentioning v>
```

Finally, both GDL and EL allow describing the elements of the specified constraints and rules by means of using natural language. GDL allows it in the term\_definitions section (inside of the ontology section) using the description field. For example:

```
ontology = (GUIDE_ONTOLOGY) <  
  term_definitions = <  
    ["en"] = (TERM_DEFINITION) <  
      terms = <["gt0003"] = (TERM) <  
        text = <"Generic name">
```

```

        description = <"The generic name of the drug which
is    an alternative name to the name of medication">
    >
    ["gt0004"] = (TERM) <
        text = <"Date (time) of first administration">
        description = <"The date and time (if required) the
medication is/was first administered">
    >
    ["gt0005"] = (TERM) <
        text = <"Date (time) of last administration">
        description = <"The date and time (if required) the
medication is to be administered for the last time">
    >

```

On the other hand, EL allows describing elements of constraints and rules by using comments. For example:

```

$systolic: Integer := 118    -- systolic blood pressure measured
$diastolic: Integer := 72   -- diastolic blood pressure measured

```

## 5.4 JUSTIFICATION OF CHOOSING EL AS BASIS

Both GDL and EL approximate the purpose of the consistency rules language in archetypes, each of them with its own particularities. None of them fit 100% our purposes, so it becomes mandatory, after taking one of them as a basis to develop the language, to extend it with new requirements. Considering the features presented in Table 12 and analysed above, we have decided to choose EL as a basis for the development of the language for representing semantic constraint and rules in archetypes. Basically, our decision is motivated as follows:

- The purpose of the GDL language is to formally represent clinical guidelines, while one of the purposes of EL is to specify invariants as constraints in the rules section of archetypes. In this sense, although both languages model clinical knowledge formally, EL fits much better our purpose.

- EL allows the declaration of variables of a series of data types, such as Integer, Real, Boolean, String, Data, Duration, Terminology\_code, etc. EL also allows the definition of container types, such as list, e.g., List<Integer>, List<String>, List<Terminology\_code>, etc.
- EL allows naming the rules natively.
- EL supports both FOR\_ALL logical universal quantifier and THERE\_EXISTS logical existential quantifier.
- EL does not need to explicitly specify the name of the archetype whose elements are being referenced via paths to define the constraints and rules since they are defined in the rules section of such archetype.

Figure 63 shows the way that the presented languages for the representation of clinical knowledge have been selected to choose one of them to be the basis for the semantic constraint and rules language in archetypes. The scheme starts with two groups. First, Task Network Models (TNMs) and second, Expression Languages. This is followed by a screening based on the particularities of each language and finally EL is chosen as the language that best fits to serve as a basis.

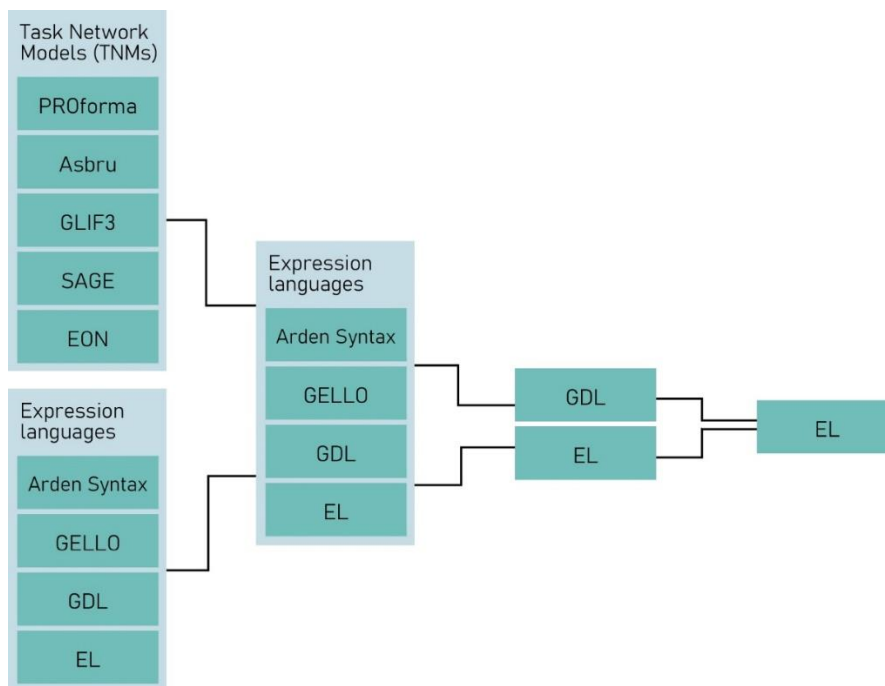


Figure 63. Selection tree of the analysed medical logic languages

## 5.5 EXTENSIONS TO EL

As explained before, we need to extend EL with extra constructs in order to cover the full fledge of consistency rules and simple, conditional and dependency intensional value set bindings. The next extensions need to be included into the EL language:

- Although EL supports IMPLIES rules (i.e., IF-THEN rules), the language should allow nesting rules via the ELSE clause. Hence, we need to extend EL to support IF-THEN-ELSE nested rules.
- EL does not support the specification of a criticality of rules. Therefore, we need to extend EL to support it.
- Definition of contexts to group related expressions is allowed by using paths and conditions on them.
- The key point of the semantic constraint and rules language in archetypes is the ability to deal with simple, conditional and dependency intensional value set binding. For this purpose, we need to meet four requirements:
  - o Ability to reference terminological concepts.
  - o Ability to represent the clinical knowledge by means of IF-THEN-ELSE rules, which is a required extension to EL as mentioned above.
  - o Ability to specify subsets of clinical concepts intensionally. For this purpose, one option is to embed the SNOMED CT ECL into EL so that it is possible to define intensional subsets of SNOMED CT concepts inside of EL. Therefore, we need to extend EL by embedding the ECL language and by creating the `Snomed_ec` data type (i.e., SNOMED CT Expression Constraint data type) to allow the definition of variables able to store ECs.
  - o Finally, ability to check the membership of a given clinical concept in an intensional subset defined by means of an EC. For this purpose, we need to extend EL with an inclusion operator.

In the next chapter we specify a concrete syntax for the expression language that covers the requirements that we have presented in this chapter along with a translation into Schematron. Additionally, three real uses cases are presented in order to validate the language.





## CHAPTER 6. THE EHRULES LANGUAGE

---

### 6.1 OVERVIEW

# EHRules

## An archetype rules language

The EHRules language, which is specified here, takes the openEHR Expression Language (EL) [86] as its core part, and add some extensions that are needed in our specific scenario (section 5.5). EHRules is provided as a way of authoring expressions and rules in textual form. This approach is the same as with any programming language, where the usual form for learning and programming is the abstract language form, while the computational form is an abstract syntax tree (AST) or similar. The core part of EHRules include the following key features taken from EL:

- variable declarations, assignments and expressions
- strong typing
- standard logical operators including universal and existential quantifier, as well as user-defined operators
- standard arithmetic and relational comparison operators, enabling the use of numeric
- parentheses for overriding operator precedence
- functions, including built-ins like `current_date`, standard functions such as `max()` defined on primitive types

As aforementioned, since EHRules takes as basis the EL language, part of this EHRules specification is directly taken from or based in the EL specification.

EHRules is based on a limited first-order predicate logic language with the addition of arithmetic and relational operators to enable the use of numeric elements. Expressions may contain variable references, value references and functions. Some elements of

EHRules are similar to the Object Constraint Language (OCL). Due to the need to accommodate the use of EHRules in XML instances, especially by using paths, it is also inspired in the XML Path Language (XPath). However, the syntax is designed to be comprehensible to developers familiar with modern object-oriented and functional languages such as Java, C# or Python.

## 6.2 EXECUTION MODEL

The execution model of EHRules consists on evaluating the expressions, which are in form of text, against a data context. The data context is what determines whether an expression is evaluated to true or false. The data context typically consists on XML data instances conforming to EHR standards. Specifically, such data context corresponds to openEHR and ISO13606 archetype XML data instances. The text may contain symbols representing internal variables and bound variables. Bound variables map to entities in the data context by using paths.

## 6.3 LANGUAGE SPECIFICATION

### 6.3.1 General structure

In the current version of EHRules, expressions may be declarations and assignments, assertions (arithmetic, relational and boolean), and rules. In our context, an assertion can be understood as a rule whose antecedent is set to true. Expressions contain symbols that represent typed variables, texts, numbers, operators and functions. Variables are either bound or local, and are defined putting \$ before names, e.g., \$heart\_rate. A bound variable is mapped to an element in the data context by using its path. This is achieved by statements that assign a path in the data context to a symbolic variable. In this sense, assignment means associating a path that references a field in the data context with a named and typed variable.

### 6.3.2 Syntax style

In the syntax style 'snake\_case', each space is replaced by an underscore character (i.e., '\_') and the first letter of each word is written in lowercase. There exists also the 'CamelCase' style. A study found that readers can recognize 'snake\_case' values more quickly than camel case [87]. Either may be used in real applications. The syntax style used in the EL specification is the 'snake\_case', in common with other openEHR specifications. On the other hand, the syntax style used in this specification and examples written in EHRules is also the 'snake\_case' one.

### 6.3.3 Typing

As EL, EHRules is fully typed. The type set of EHRules is a subset of that used in other openEHR components, and therefore supports some of the basic types described in the openEHR Foundation Types Specification plus an extension type for SNOMED CT ECs:

- Primitive types: Boolean, Integer, Real, String, Date, Time, Date\_time, Duration, and Terminology\_code
- Subset type: Snomed\_ec, i.e., SNOMED CT Expression Constraint
- Container type: List<T>, i.e., a list of any primitive type T, or the type Snomed\_ec

In Table 13, Table 14, and Table 15 we present the data types, operators and functions that are supported by the EHRules language, as well as an associated description and some examples.

#### 6.3.3.1 Primitive data types

Data type	Description
Integer	Integer value
Examples	2, 3, -5, 48, -100
Real	Real value
Examples	-5.34, 0.489, 23.343, -500.98
String	String value
Examples	'Hello world!', 'This is a String'

Boolean	Boolean value
Examples	true, false
Date	Date value
Examples	[2006-11-22], [1980-10-30]
Time	Time value
Examples	[08:30:00], [22:55:59]
Date_time	Date and time value
Examples	[2006-11-22 08:30:00], [1980-10-30 18:57:01]
Duration	Duration value
Examples	[23Y], [10M 10m], [10Y 10M 10D 10h 10m 10s]
Terminology_code	Terminology code
Examples	[snomed_ct::703118005 Feminine gender (finding)]

Table 13. EHRules primitive data types

### 6.3.3.2 Subset data type

Data type	Description
Snomed_ec	SNOMED CT Expression Constraint
Examples	[snomed_ct_ec::<<95660002  Thunderclap headache ]
	[snomed_ct_ec::<<417746004  Traumatic injury  OR <<105612003  Injury of internal organ ]
	[snomed_ct_ec::< 404684003  Clinical finding (finding)  : 363698007  Finding site (attribute)  = << 39057004  Pulmonary valve ]

Table 14. EHRules subset data type

### 6.3.3.3 Container type

Data type	Description	
List<T>	List of any primitive type T, or the type Snomed_ec	
Examples	List<Integer>	{5, -23, 44, 2}, {-500, 1024}
	List<Real>	{3.25}, {99.89, 722.0, -125.115}
	List<String>	{'Hello', 'World'}, {'This', 'is', 'a', 'string'}
	List<Terminology_code>	{[snomed_ct::66264000 Todd's paresis[]], [snomed_ct::76571007 Septic shock[]], [snomed_ct:: 439127006 Thrombosis[]]}

Table 15. EHRules container data type

### 6.3.4 Variable declaration and assignation

In the current version of EHRules it is not allowed only variable declaration neither only variable assignation, but both declaration and assignation should be defined in the same sentence. Variables are declared with a formal type. Bound variables include an assignment to a path within an assumed data context (i.e., an archetype instance). Local variable declarations include an assignment to a value-returning expression.

An assignment to a variable is expressed using the “:=” operator. An assignment is made in a declaration in the same way as in many programming languages. The right hand side of an assignment is any value-returning expression. It is important to note that variables are referenced within assignments and expressions using the same syntax, i.e., \$var\_name. Some examples of variable declaration and assignation of Terminology\_code and Snomed\_ec data types are illustrated in Table 16 and Table 17.

Terminology_code
Assigns a path
\$diagnosis: Terminology_code := /data[at0001]/items[at0002.1]/value;
Assigns a literal
\$gender: Terminology_code := [snomed_ct::703118005 Feminine gender (finding)];
Assigns a list of literals, expressions and paths
\$list_term_code: List<Terminology_code> := {[snomed_ct::703118005 Feminine gender (finding)], /data[at0001]/items[at0002.1]/value, \$diagnosis};

Table 16. Examples of declaration of Terminology\_code variables and assignation of values

Snomed_ec
Assigns a literal
\$hemorrhage: Snomed_ec := [snomed_ct_ec::<<50960005 Hemorrhage ];
Assigns an expression
\$diagnosis: Snomed_ec := \$hemorrhage;
Assigns a list of literals, expressions and paths
\$list_snomed_ec: List<Snomed_ec> := {[snomed_ct_ec::<<50960005 Hemorrhage ], \$diagnosis, \$hemorrhage, value/value}

Table 17. Examples of declaration of Snomed\_ec variables and assignation of values

Variables that are bound to entities in the archetype data instance works differently from local variables, since their availability depends on the existence of the entities referenced by its path. There is no guarantee that the referenced value is available. In the case that bound variables do not exist in the data context, they cannot be evaluated and therefore constraints and rules making use of such variables will throw an error in execution time. The approach used for EHRules, as in EL, is to allow bound variables to be used freely, as

for local variables, but in order to impose more control, EHRules provides two path functions that can be used within an expression to ensure that one or more variables can be populated before proceeding with logic that depends on them. The two functions (i.e., `existsPath()` and `emptyPath()`) are presented later in this specification.

### 6.3.5 Expressions

As in EL, Expressions constitute the main part of the EHRules language. They consist of a familiar typed, operator-based syntax common to many programming languages and logics. Formally, an expression is one of the following:

- Terminal entities
  - o Literal
  - o Variable
  - o Function
  - o Path
- Non-terminal entities
  - o Operator

The use of literals, variables, constants and function calls is the same as in common languages. All expressions end with a semicolon (“;”) and it is possible to assign a descriptive name to each of them. Inline comments are allowed by preceding the text with double slash (i.e., `//text`) and also multi-line comments by enclosing the text between slash and asterisk (i.e., `/* text */`) As an example:

```
// expression containing a variable and function call
['Expression 1']
currentDate() - $date_of_birth;

/*
  expression containing:
  - Two variables
  - One function call
*/
```

```
[‘Expression 2’]
```

```
$date_time_of_initial_onset < (currentDateTime() - $threshold_time);
```

Note that the EHRules syntax is presented formally in ABNF (Augmented Backus–Naur form) in Annex 2, along with a description of each token and production rule.

### 6.3.6 Paths, contexts and conditions

Regarding the use of paths, in EHRules it is mandatory to set contexts in order to define groups of expressions that are related. A context is defined by means of a path. For example:

```
/* note that height, weight and temperature are not a variables but child elements of the path */
```

```
context: ENTRY/items[at0014]/exploration;
```

```
height > 120;
```

```
weight < 100;
```

```
temperature < 37;
```

```
/* or equivalently */
```

```
context: /;
```

```
ENTRY/items[at0014]/exploration/height > 120;
```

```
ENTRY/items[at0014]/exploration/weight < 100;
```

```
ENTRY/items[at0014]/exploration/temperature < 37;
```

```
/* or equivalently */
```

```
context: /ENTRY;
```

```
items[at0014]/exploration/height > 120;
```

```
items[at0014]/exploration/weight < 100;
```

```
items[at0014]/exploration/temperature < 37;
```



The definition of contexts can be carried out based on conditions on them. The allowed syntax and expressions for such conditions is defined in the ABNF EHRules specification. As an example:

<pre>context: ENTRY/items[at0011]/exploration; condition: count(children()) &gt; 3; // Condition on number of child elements</pre>
<pre>context: ENTRY/items[at0011]/exploration; count(temperature/children()) &gt;= 2; // Condition on number of child elements</pre>
<pre>context: ENTRY/items[at0011]/exploration; condition: height &gt; 185; // Condition on a child element value</pre>
<pre>context: ENTRY/items[at0011]/exploration/weight; condition: . &lt; 100; // Condition on the element referenced by the path  /* which is equivalent to */ context: ENTRY/items[at0011]/exploration; condition: weight &lt; 100; // Condition on a child element value</pre>

Finally, it is important to note that, as in the openEHR EL language, in EHRules, raw paths may be used directly as variables. This is primarily to allow UI expression building tools that work based on the path map of a data context (e.g., an openEHR archetype) to generate expressions directly using paths rather than introducing variables.

### 6.3.7 Functions

Functions are considered terminal entities in the EHRules language, and they are of a built-in type. For example:

<pre>\$date_of_birth: Date := /items[at0002]/items[at0003]/value; \$current_date: Date := currentDate(); \$is_correct_age: Boolean := \$current_date &gt; \$date_of_birth;</pre>
--

Or equivalently in two lines:

```
$date_of_birth: Date := /items[at0002]/items[at0003]/value;
$is_correct_age: Boolean := currentDate() > $date_of_birth;
```

Or just in one line:

```
$is_correct_age: Boolean := currentDate() > /items[at0002]/items[at0003]/value;
```

The EHRules built-in functions are presented in Table 18 along with examples.

Function	Description		
Path functions			
existsPath	Returns True if a path exists, otherwise it returns False		
Example	existsPath(/items[at0002]/items[at0003]/value);		
Result	true (we assume that the path exists)		
emptyPath	Returns True if a path is empty, otherwise it returns False		
Example	emptyPath(/items[at0002]/items[at0003]/value);		
Result	false (we assume that the path is not empty)		
Real function			
round	Rounds a real number to a specified number of positions		
Examples	round(233.4567, 3);	round(1999.88, 1);	
Results	233.457	1999.9	
String functions			
length	Returns the number of characters in a string		
Examples	length('Hello');	length(value/value);	length(\$var_str);
Results	5	5 (we assume 'Hello')	5 (we assume 'Hello')
trim	Returns a string with leading and trailing whitespace removed		

Examples	trim(' Hello ');	trim(value/value);	trim(\$var_str);
Results	'Hello'	'Hello' (we assume it)	'Hello' (we assume it)
concat (+)	Returns the concatenation of the specified strings		
Example	'Hello ' + 'world' + '!' + \$var_str + value/value;		
Result	'Hello world! Hello Hello' (we assume the last two 'Hello')		
toUpperCase	Converts a string to upper case		
Examples	toUpperCase('Hello');	toUpperCase(\$var_str);	
Results	HELLO	HELLO (we assume 'Hello')	
toLowerCase	Converts a string to lower case		
Examples	toLowerCase('Hello');	toLowerCase(value/value);	
Results	hello	hello (we assume 'Hello')	
Date and time functions			
currentDate	Returns the current date		
How to use it	currentDate();		
Result	[2021-09-10] (we assume it)		
currentTime	Returns the current time		
How to use it	currentTime();		
Result	[21:36:00] (we assume it)		
currentDateTime	Returns the current date and time		
How to use it	currentDateTime();		
Result	[2021-09-10 21:36:00] (we assume it)		
Aggregate functions*			
*We assume that both \$var_list and list/values references the list {-5, -1, 5, 20, 80, 100, 2348}			
count	Returns the number of values in a specified variable containing a list or path that references a list. It is expected to support literal lists in future versions of EHRules.		

Examples	<code>count(\$var_list);</code>	<code>count(list/values);</code>
Results	7	7
avg	Returns the sum of the values in a specified variable containing a list or path that references a list divided by the number of values. It is expected to support literal lists in future versions of EHRules.	
Examples	<code>avg(\$var_list);</code>	<code>avg(list/values);</code>
Results	363.85	363.85
min	Returns the lowest value in a specified variable containing a list or path that references a list. It is expected to support literal lists in future versions of EHRules.	
Examples	<code>min(\$var_list);</code>	<code>min(list/values);</code>
Results	-5	-5
max	Returns the highest value in a specified variable containing a list or path that references a list. It is expected to support literal lists in future versions of EHRules.	
Examples	<code>max(\$var_list);</code>	<code>max(list/values);</code>
Results	2348	2348
sum	Returns the sum of the values of a specified variable containing a list or path that references a list. It is expected to support literal lists in future versions of EHRules.	
Examples	<code>sum(\$var_list);</code>	<code>sum(list/values);</code>
Results	2547	2547

Table 18. Built-in functions supported by EHRules

### 6.3.8 Operators

Expressions can include assignment, arithmetic, relational and logical operators, plus the existential and universal quantifiers. The full operator set is shown below in Table 19, along with their symbolic representation, a textual description, and some examples.

Operator	Description		
Assignment operator			
:=	Assigns a value to a specified variable name in a declaration		
Examples	\$var: Integer := -1024;	\$var: List<Real> := {1.4, -2.0, 98.15};	
Arithmetic operators Numeric result – descending precedence order The use of parentheses to change precedence is allowed			
^	Exponentiation		
*	Multiplication		
/	Division		
%	Modulo division		
+	Addition		
-	Subtraction		
Example	5 / (-10.5 + \$var) + ((value/value % length('Hello')) ^ avg(\$list))		
Relational operators Boolean result – Equal precedence			
=	Equality relation between numerical		
!= or <>	Inequality relation between numerical		
<	Less than relation between numerical		
<=	Less than or equal relation between numerical		
>	Greater than relation between numerical		
>=	Greater than or equal relation between numerical		
Examples	3 + 2 < 55;	\$num_var < value/value;	23 - 5 >= \$num_var;

Logical operators Boolean result – descending precedence order The use of parentheses to change precedence is allowed	
NOT	Logical negation
AND	Logical conjunction
OR	Logical disjunction
Example	NOT (2 < 5 AND \$num_var < value/value) OR 235 >= \$num_var;
Logical quantifiers Boolean result	
FOR_ALL	Universal quantifier
Example	FOR_ALL \$i IN ENTRY/items/value/value : \$i>=37 AND \$i<=41;
THERE_EXISTS	Existential quantifier
Example	THERE_EXISTS \$i IN ENTRY/items/value/value : \$i<37 OR \$i>41;
Inclusion operator Boolean result	
IN	Inclusion of a terminological code in a subset of codes
Example	\$diagnosis IN [snomed_ct_ec::<<56265001  Heart disease ];

Table 19. Operators supported by EHRules

### 6.3.9 Rules

An IF-THEN-ELSE rule is an expression that evaluates a condition and depending on the resulting truth value (i.e., true or false), the result is one of two possible expressions. Only the expression associated with the THEN clause is mandatory. EHRules supports both IF-THEN and IF-THEN-ELSE rules, and also nested IF-THEN-ELSE rules (see Figure 64).

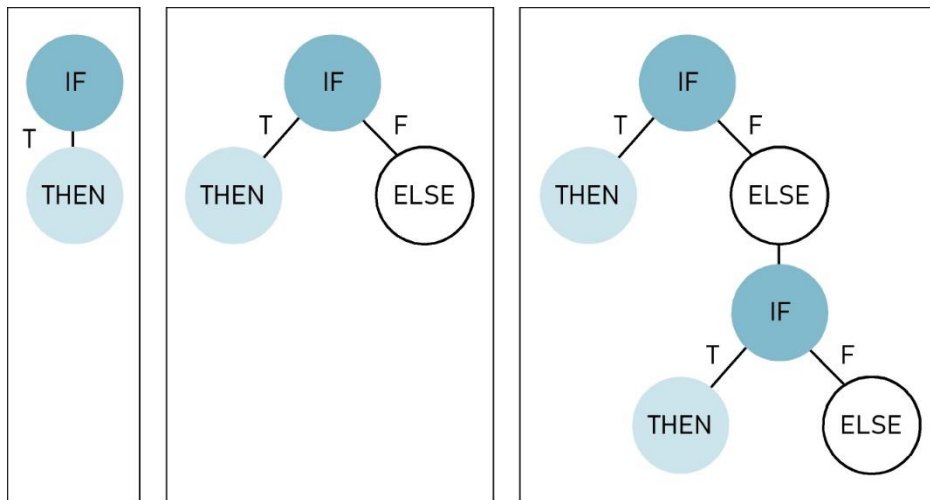


Figure 64. IF-THEN (left), IF-THEN-ELSE (center), and nested IF-THEN-ELSE (right) rules diagrams

In Table 20 we show the EHRules syntax of the presented rules and some examples.

IF-THEN		
Syntax	<ul style="list-style-type: none"> <li>- IF <i>boolean_expression1</i> THEN <i>boolean_expression2</i>;</li> <li>- IF <i>boolean_expression1</i> THEN {<i>boolean_expression2</i>; <i>boolean_expression3</i>; <i>boolean_expressionN</i>;};</li> </ul>	
Examples	IF \$temperature > 37 THEN \$fever = true;	IF temperature/value > 37 THEN \$fever = true;
	IF temperature/value IN \$temperatures THEN \$fever = true;	IF \$temperature > 37 THEN {\$fever = true; \$medication = true; \$discharge = false;};
IF-THEN-ELSE		
Syntax	<ul style="list-style-type: none"> <li>- IF <i>boolean_expression1</i> THEN <i>boolean_expression2</i>; ELSE <i>boolean_expression3</i>;</li> <li>- IF <i>boolean_expression1</i> THEN {<i>boolean_expression2</i>; <i>boolean_expression3</i>; <i>boolean_expression4</i>;} ELSE {<i>boolean_expression5</i>; <i>boolean_expression6</i>; <i>boolean_expressionN</i>;};</li> </ul>	
Examples	IF \$temperature > 37	IF temperature/value > 37

	THEN \$fever = true; ELSE \$fever = false;	THEN \$fever = true; ELSE \$fever = false;
	IF temperature/value IN \$temperatures THEN \$fever = true; ELSE \$fever = false;	IF \$temperature > 37 THEN {\$fever = true; \$medication = true; \$discharge = false;} ELSE {\$fever = false; \$medication = false; \$discharge = true;} }
IF-THEN-ELSE IF-THEN-ELSE		
Syntax	<ul style="list-style-type: none"> <li>- IF <i>boolean_expression1</i> THEN <i>boolean_expression2</i>; ELSE IF <i>boolean_expression3</i>; THEN <i>boolean_expression4</i>; ELSE <i>boolean_expression5</i>;</li> </ul>	
	<ul style="list-style-type: none"> <li>- IF <i>boolean_expression1</i> THEN {<i>boolean_expression2</i>; <i>boolean_expression3</i>; <i>boolean_expression4</i>;} ELSE IF <i>boolean_expression5</i>; THEN {<i>boolean_expression6</i>; <i>boolean_expression7</i>; <i>boolean_expression8</i>;} ELSE {<i>boolean_expression9</i>; <i>boolean_expression10</i>; <i>boolean_expressionN</i>;} }</li> </ul>	
Examples	IF \$temperature > 39 THEN \$high_fever = true; ELSE IF \$temperature > 37 THEN \$fever = true; ELSE \$fever = false;	IF temperature/value > 39 THEN \$high_fever = true; ELSE IF temperature/value > 37 THEN \$fever = true; ELSE \$fever = false;
	IF temperature/value IN \$high_temperatures THEN \$high_fever = true; ELSE IF temperature/value IN \$temperatures THEN \$fever = true; ELSE \$fever = false;	IF \$temperature > 39 THEN {\$high_fever = true; \$fever = true; \$double_medication = true; \$medication = true; \$discharge = false;} ELSE IF \$temperature > 37 THEN {\$high_fever = false; \$fever = true; \$double_medication = false; \$medication = true; \$discharge = false;} }



		<pre>ELSE {\$high_fever = false; \$fever = false; \$double_medication = false; \$medication = false; \$discharge = true;}</pre>
--	--	---

Table 20. IF-THEN, IF-THEN-ELSE and nested IF-THEN-ELSE rules syntax and examples

### 6.3.10 Value set bindings

A key point of the EHRules language is the possibility to specify value set bindings between archetypes and SNOMED CT subsets. As stated in section 2.5, while semantic binding (also known as model meaning binding) is used to define the meaning of an information model artefact using a concept or expression from the terminology, the purpose of value set binding is to record the set of possible values which can populate a given coded data element or attribute in the information model. There exist four types of value set binding:

- Simple: the data element is associated with a single extensional or intensional value set.
- Conditional: the data element is associated with a single extensional or intensional value set depending on a condition.
- Dependency: the data element is associated with a single extensional or intensional value set depending on the value of another data element.
- Compositional: the data element is associated with a single extensional or intensional value set composed by another data elements.

The current version of EHRules is intended to allow the definition of simple, conditional and dependency value set bindings. Additionally, EHRules supports a fourth type of value set binding, which is a combination between conditional and dependency value set binding. As explained in section 5.5, to meet these requirements EHRules supports the following abilities:

- Reference to terminological concepts by means of the Terminology\_code data type, which is already a requirement of the EL language.

- Representation of the clinical knowledge by means of IF-THEN-ELSE rules, which is a required extension to EL.
- Specification of subsets of clinical concepts intensionally. For this purpose, EHRules allows embedding the SNOMED CT ECL so that it is possible to define intensional subsets of SNOMED CT concepts by means of the Snomed\_ec data type (i.e., SNOMED CT Expression Constraint data type).
- Membership of a given clinical concept in an intensional subset defined by means of an EC. For this purpose, EHRules supports an inclusion operator (i.e., IN).

Table 21 shows some examples of simple, conditional, dependency and conditional plus dependency intensional value set bindings.

Simple intensional value set binding	
Syntax	<ul style="list-style-type: none"> <li>- <i>concept</i> IN <i>subset</i>;</li> </ul> <p>where <i>concept</i> can be a literal, a variable containing a concept, or a path referencing a concept; and <i>subset</i> can be a literal, a variable containing a subset, or a path referencing a subset</p>
Examples	[snomed_ct::417532002  Allergy to fish (finding)] IN [snomed_ct_ec::<< 420134006  Propensity to adverse reaction (finding)];
	[snomed_ct::417532002  Allergy to fish (finding)] IN \$allergies_subset;
	\$allergy_to_fish IN \$allergies_subset;
	ENTRY/items[at0007]/value/concept IN \$allergies_subset;
Conditional intensional value set binding	
Syntax	<ul style="list-style-type: none"> <li>- IF <i>boolean_expression1</i> THEN <i>concept</i> IN <i>subset1</i>;</li> <li>ELSE IF <i>boolean_expression2</i> THEN <i>concept</i> IN <i>subset2</i>;</li> <li>ELSE <i>concept</i> IN <i>subset3</i>;</li> </ul> <p>where only the first IF-THEN is mandatory</p>

Examples	IF \$gender = [snomed_ct::248153007   Male (finding)] THEN \$procedure IN \$male_procedures;
	IF \$gender = [snomed_ct::248153007   Male (finding)] THEN \$procedure IN \$male_procedures; ELSE IF \$gender = [snomed_ct::248152002   Female (finding)] THEN \$procedure IN \$female_procedures; ELSE \$procedure IN \$all_procedures;
	IF ENTRY/items[at0003]/value/age > 75 AND ENTRY/items[at0004]/value/department = [snomed_ct:: 309941000   Rheumatology department] THEN ENTRY/items[at0004]/value/diagnosis IN \$rheumatic_diseases_in_geriatrics;
	IF \$procedure_priority = [snomed_ct::25876001   Emergency (qualifier value)] AND \$procedure_method IN snomed_ct_ec::<< 129284003   Surgical action (qualifier value)] THEN \$procedure IN [snomed_ct_ec::<< 73994005   Emergency operation (procedure)];
	IF \$gender = [snomed_ct::248152002   Female (finding)] AND \$procedure_method = [snomed_ct::312250003   Magnetic resonance imaging - action (qualifier value)] AND \$procedure_site_direct IN [snomed_ct_ec::<<43174007   Gonadal structure (body structure)] THEN \$procedure IN [snomed_ct_ec::<<241628003   Magnetic resonance imaging of ovary (procedure)];
Dependency intensional value set binding	
Syntax	- <i>concept</i> IN <i>subset</i> ; where <i>concept</i> can be represented by a literal, a variable containing a concept, or a path referencing a concept; and

	<p><i>subset</i> can be represented by either a literal that references an element of the archetype (i.e., a concept or a post-coordinated expression) via its path or a variable containing its path, or a variable containing such literal</p>
Examples	<pre>\$surgical_procedure: Terminology_code := ENTRY/items[at0009]/value/concept; [snomed_ct::80146002 Excision of appendix (procedure)] IN [snomed_ct_ec::&lt;[[\$surgical_procedure]]];</pre>
	<pre>[snomed_ct::80146002 Excision of appendix (procedure)] IN [snomed_ct_ec::&lt;[ENTRY/items[at0009]/value/concept]];</pre>
	<pre>\$surgical_procedure_in_cardiovascular_system: Terminology_code := ENTRY/items[at0010]/value/concept; [snomed_ct::425785006 Repair of tetralogy of Fallot with absent pulmonary valve (procedure)] IN [snomed_ct_ec::&lt;[[\$surgical_procedure_in_cardiovascular_system]]];</pre>
	<pre>[snomed_ct::425785006 Repair of tetralogy of Fallot with absent pulmonary valve (procedure)] IN [snomed_ct_ec::&lt;[ENTRY/items[at0010]/value/concept]];</pre>
Conditional plus dependency intensional value set binding	
Syntax	<ul style="list-style-type: none"> <li>- IF <i>boolean_expression1</i> THEN <i>concept</i> IN <i>subset1</i>; <ul style="list-style-type: none"> <li>o ELSE IF <i>boolean_expression2</i> THEN <i>concept</i> IN <i>subset2</i>;</li> <li>o ELSE <i>concept</i> IN <i>subset3</i>;</li> </ul> </li> <li>- where only the first IF-THEN is mandatory and <i>concept</i> IN <i>subset</i> expressions use the same syntax as that presented in dependency intensional value set binding</li> </ul>
Examples	<pre>\$surgical_procedure: Terminology_code := ENTRY/items[at0009]/value/concept;</pre>

<pre>IF NOT emptyPath(ENTRY/items[at0009]/value/concept) THEN [snomed_ct::80146002 Excision of appendix (procedure)] IN [snomed_ct_ec::&lt;[[\$surgical_procedure]]];</pre>
<pre>IF NOT emptyPath(ENTRY/items[at0009]/value/concept) THEN [snomed_ct::80146002 Excision of appendix (procedure)] IN [snomed_ct_ec::&lt;[[ENTRY/items[at0009]/value/concept]]];</pre>
<pre>\$surgical_procedure_in_cardiovascular_system: Terminology_code := ENTRY/items[at0010]/value/concept; IF NOT emptyPath(ENTRY/items[at0010]/value/concept) THEN [snomed_ct::425785006 Repair of tetralogy of Fallot with absent pulmonary valve (procedure)] IN [snomed_ct_ec::&lt;[[\$surgical_procedure_in_cardiovascular_system]]];</pre>
<pre>IF NOT emptyPath(ENTRY/items[at0010]/value/concept) THEN [snomed_ct::425785006 Repair of tetralogy of Fallot with absent pulmonary valve (procedure)] IN [snomed_ct_ec::&lt;[[ENTRY/items[at0010]/value/concept]]];</pre>

Table 21. Syntax and examples of simple, conditional, dependency and conditional plus dependency intensional value set bindings

ECL allows not only the specification of intensional subsets but also simulated extensional subsets by means of concatenating disjunctions of concepts (i.e., concept1 OR concept2 OR... conceptN). Therefore, EHRules allows both specification of intensional and extensional value set bindings. Below is an example of conditional extensional value set binding:

<pre>\$laterality: Terminology_code := ENTRY/items[at0023]/value/laterality; IF NOT emptyPath(\$laterality) THEN \$laterality IN [snomed_ct_ec::24028007 Right (qualifier value)  OR 7771000 Left (qualifier value)  OR 51440002 Right and left (qualifier value)];</pre>
---

Additionally, ECL allows referencing the elements of extensional refsets, such as the Spanish SNS Refset for SARS-CoV-2, or the SNOMED CT General Dentistry Diagnostic Refset. Therefore, EHRules also allows the specification of extensional value set bindings by referencing external refsets. For example (note that the '^' ECL operator references the members of a refset):

```
$procedure IN [snomed_ct_ec::^900050181000122100|SNS SARS-CoV-2 Refset|];
```

## 6.4 USES CASES

EHRules has been tested in three real use cases: a formal definition of practice guidelines for acute stroke care [88], the FSIII Danish standard for the specification of guidelines for documenting healthcare observations and interventions in home care [89,90], and a requisition for radiology procedures from the North Denmark Region clinical information system. The objective of this test is to evaluate to what extent EHRules is able to represent formally by means of expressions, including rules and value set bindings, the medical knowledge contained in those uses cases.

### 6.4.1 Guidelines for acute stroke care

In [88], Nadim Anani, Rong Chen et. al. extracted rules from the European clinical practice guidelines as well as from treatment contraindications for acute stroke care and represented them using GDL. They successfully represented clinical rules about 14 out of 19 contraindications for thrombolysis and other aspects of acute stroke care with 80 GDL rules. The rules were based on 14 reused international openEHR archetypes (one of which was modified) and 2 newly created archetypes. The archetypes were based on the openEHR reference information model (openEHR RM) and were of the CARE\_ENTRY type, i.e., OBSERVATIONS, EVALUATIONS, INSTRUCTIONS and ACTIONS. They concluded that shareable guideline knowledge for use in automated retrospective checking of guideline compliance may be achievable using GDL.

Our purpose is to represent the contraindications for thrombolysis by using the EHRules language. Table 22 shows the 19 contraindications extracted from [88] expressed in natural language. It also shows whether they have been represented in either GDL and EHRules.

ID	Thrombolysis contraindications	GDL	EHRules
1	Stroke onset more than 4.5 hours ago	Yes	Yes
2	Symptom presentation suggesting another aetiology than that of stroke and/or the patient recovered within 30 minutes	No	No
3	Unclear stroke symptoms	No	No
4	National Institutes of Health Stroke Scale (NIHSS) score higher than 25	Yes	Yes
5	CT scan shows haemorrhage	Yes	Yes
6	CT scan shows major stroke that covers more than 30% of the middle cerebral artery	No	No
7	Blood glucose is lower than 3 mmol/litre or higher than 22 mmol/litre	Yes	Yes
8	Blood pressure is higher than 185/110 mmHg despite two attempts of intravenous beta-blocking bolus treatment (approximately 20 mg of Labetalol per bolus)	No	No
9	History of cerebral haemorrhage or intracranial bleeding	Yes	Yes
10	Patient describes an explosive headache (that resembles a subarachnoid haemorrhage)	Yes	Yes
11	Ongoing or recent severe haemorrhage (extracranial or intracranial)	No	No

12	Likely postictal paresis	Yes	Yes
13	Suspected septic shock	Yes	Yes
14	Bleeding disorder or anticoagulation treatment	Yes	Yes
15	One of the following: infectious endocarditis, pericarditis, ventricular thrombosis, atrial septal aneurysm, severe heart failure, pancreatitis, severe liver damage	Yes	Yes
16	One of the following in the last week: lumbar puncture, central venous catheter	Yes	Yes
17	One of the following in the last month: operation/biopsy from parenchymatous organs, trauma with internal injuries, duodenal ulcer, bleeding from the urinary tract	Yes	Yes
18	One of the following in the last three months: stroke, head trauma, operation in the central nervous system, definite gastrointestinal bleeding	Yes	Yes
19	Pregnancy, childbirth in the last month, breastfeeding (relative contraindications)	Yes	Yes

Table 22. 19 thrombolysis contraindications

In view of the results presented in Table 22, there is a complete matching between what contraindications can be represented using either GDL and EHRules. As stated in [88], the five rules that have not been represented neither in GDL nor in EHRules (i.e., 2, 3, 6, 8, and 11) constitute a challenge due to some unspecific expressions such as ‘unclear’ and ‘major’, as well as missing temporal aspects such as the time interval between the ‘two attempts of intravenous beta-blocking bolus treatment’ or in ‘recent severe haemorrhage’. A higher level of detail in those rules should allow their representation in both GDL and EHRules.



It is important to note that the GDL language allows referencing more than one archetype in each rule. In contrast to EHRules, whose expressions and rules are intended to be incorporated into the 'Rules' section of one archetype to improve the consistency of such archetype, the GDL rules are defined into a separated file and therefore rules can access several archetypes. However, although three of the contraindications presented in Table 22 need to access two different archetypes (14, 17 and 18), we have been able to separate each of them into two expressions (IDa and IDb) since our purpose at this point is to evaluate the expressiveness of EHRules using real use cases containing clinical knowledge. The archetypes used for the representations and the contraindications involved are showed in Table 23.

ID	openEHR archetype
1, 9, 10, 12, 13, 14a, 15, 17b, 18b, 19	openEHR-EHR-EVALUATION.problem-diagnosis.v1
16, 17a, 18a	openEHR-EHR-ACTION.procedure.v1
4	openEHR-EHR-OBSERVATION.nihss.v1
5	openEHR-EHR-ITEM_TREE.imaging.v1
7	openEHR-EHR-OBSERVATION.lab_test-blood_glucose.v1
14b	openEHR-EHR-INSTRUCTION.medication.v1

*Table 23. Archetypes involved and contraindications by ID*

Below it is shown in tabular form the 14 out of 19 contraindications that we have been able to represent using EHRules, associated with the archetypes accessed by using paths in order to read patient data. It is assumed that the context has been defined as the root path in the archetype in all cases (i.e., context: /;)

	openEHR-EHR-EVALUATION.problem-diagnosis.v1
1	<pre> /*Stroke onset more than 4.5 hours ago*/  \$date_time_of_initial_onset: Date_time := /data[at0001]/items[at0003]/value; \$current_date_time: Date_time := currentDateTime(); \$threshold_time: Duration := [4h 30m];  ['Expression 1'] \$date_time_of_initial_onset &lt; (\$current_date_time - \$threshold_time); </pre>

	openEHR-EHR-OBSERVATION.nihss.v1
4	<pre> /*National Institutes of Health Stroke Scale (NIHSS) score higher than 25*/  \$NIHSS_score: Real := /data[at0001]/events[at0002]/data[at0003]/items[at0085];  ['Expression 4'] \$NIHSS_score &gt; 25; </pre>

	openEHR-EHR-ITEM_TREE.imaging.v1
5	<pre> /*CT scan shows haemorrhage*/  \$finding: Terminology_code := /items[at0002]/items[at0003]/value; \$hemorrhage: Snomed_ec := [snomed_ct_ec::&lt;&lt;50960005 Hemorrhage];  ['Expression 5'] \$finding IN \$hemorrhage; </pre>

	openEHR-EHR-OBSERVATION.lab_test-blood_glucose.v1
7	<pre>/*Blood glucose is lower than 3 mmol/litre or higher than 22 mmol/litre*/ \$blood_glucose: Real := data[at0001]/events[at0002]/data[at0003]/items[at0078.2]/value;  ['Expression 7'] \$blood_glucose &lt; 3.0 OR \$blood_glucose &gt; 22.0;</pre>

	openEHR-EHR-EVALUATION.problem-diagnosis.v1
9	<pre>/*History of cerebral haemorrhage or intracranial bleeding*/ \$diagnosis: Terminology_code := /data[at0001]/items[at0002.1]/value; \$cerebral_intracranial_haemorrhage: Snomed_ec := [snomed_ct_ec::&lt;&lt;274100004 Cerebral hemorrhage  OR &lt;&lt;1386000 Intracranial hemorrhage ];  ['Expression 9'] \$diagnosis IN \$cerebral_intracranial_haemorrhage;</pre>

	openEHR-EHR-EVALUATION.problem-diagnosis.v1
10	<pre>/*Patient describes an explosive headache (that resembles a subarachnoid haemorrhage)*/ \$diagnosis: Terminology_code := /data[at0001]/items[at0002.1]/value; \$thunderclap_headache: Snomed_ec := [snomed_ct_ec::&lt;&lt;95660002  Thunderclap headache ];  ['Expression 10'] \$diagnosis IN \$thunderclap_headache;</pre>

	openEHR-EHR-EVALUATION.problem-diagnosis.v1
12	<pre> /*Likely postictal paresis*/  \$diagnosis: Terminology_code := /data[at0001]/items[at0002.1]/value; \$confidence: String := /data[at0001]/items[at0.55]/value; \$postictal_paresis: Snomed_ec := [snomed_ct_ec::&lt;&lt;66264000 Todd's paresis ];  ['Expression 12'] \$diagnosis IN \$postictal_paresis AND \$confidence = 'Suspicion'; </pre>

	openEHR-EHR-EVALUATION.problem-diagnosis.v1
13	<pre> /*Suspected septic shock*/  \$diagnosis: Terminology_code := /data[at0001]/items[at0002.1]/value; \$confidence: String := /data[at0001]/items[at0.55]/value; \$septic_shock: Snomed_ec := [snomed_ct_ec::&lt;&lt;76571007 Septic shock ];  ['Expression 13'] \$diagnosis IN \$septic_shock AND \$confidence = 'Suspicion'; </pre>

14	openEHR-EHR-EVALUATION.problem-diagnosis.v1
	<pre>/*Bleeding disorder or anticoagulation treatment*/  \$diagnosis: Terminology_code := /data[at0001]/items[at0002.1]/value; \$bleeding_tendency: Snomed_ec := [snomed_ct_ec::&lt;&lt;64779008 Blood coagulation disorder ];  ['Expression 14a'] \$diagnosis IN \$bleeding_tendency;</pre>
	openEHR-EHR-INSTRUCTION.medication.v1
	<pre>\$medication: Terminology_code := activities[at0001]/description[openEHR- EHR-ITEM_TREE.medication.v1]/items[at0001]/value; \$anticoagulants: Snomed_ec := [snomed_ct_ec::&lt;&lt;48603004  Product containing warfarin  OR &lt;&lt;714788005  Product containing dabigatran  OR &lt;&lt;442539005 Product containing rivaroxaban ];  ['Expression 14b'] \$medication IN \$anticoagulants;</pre>

	openEHR-EHR-EVALUATION.problem-diagnosis.v1
15	<pre> /*One of the following: infectious endocarditis, pericarditis, ventricular thrombosis, atrial septal aneurysm, severe heart failure, pancreatitis, severe liver damage*/ \$diagnosis: Terminology_code := /data[at0001]/items[at0002.1]/value; \$contraindicative_diagnoses: Snomed_ec := [snomed_ct_ec::&lt;&lt;56675007 Acute heart failure  OR &lt;&lt;95440004 Atrial septal aneurysm  OR &lt;&lt;233850007 Infective endocarditis  OR &lt;&lt;439127006 Thrombosis  OR &lt;&lt;75694006 Pancreatitis  OR &lt;&lt;3238004  Pericarditis  OR &lt;&lt;59927004  Hepatic failure ]; ['Expression 15'] \$diagnosis IN \$contraindicative_diagnoses; </pre>

	openEHR-EHR-ACTION.procedure.v1
16	<pre> /*One of the following in the last week: lumbar puncture, central venous catheter*/ \$procedure: Terminology_code := /description[at0001]/items[at0002]/value; \$date_time: Date_time := /time/value; \$lumbar_puncture_or_central_venous_catheter: Snomed_ec := [snomed_ct_ec::&lt;&lt;277762005 Lumbar puncture  OR &lt;&lt;233527006 Central venous cannula insertion ]; \$current_date_time: Date_time := currentDate(); \$threshold_time: Duration := [168h]; ['Expression 16'] \$procedure IN \$lumbar_puncture_or_central_venous_catheter AND \$date_time &gt;= (\$current_date_time - \$threshold_time); </pre>

17	openEHR-EHR-ACTION.procedure.v1
	<pre> /*One of the following in the last month: operation/biopsy from parenchymatous organs, trauma with internal injuries, duodenal ulcer, bleeding from the urinary tract*/  \$procedure: Terminology_code := /description[at0001]/items[at0002]/value; \$date_time: Date_time := /time/value; \$operation_biopsy_parenchymatous_organisms: Snomed_ec := [snomed_ct_ec::(&lt;&lt;86273004  Biopsy  OR &lt;&lt;387713003  Surgical procedure ): * = &lt;&lt;116005007  Entire parenchymatous viscus ]; \$current_date_time: Date_time := currentDateTime(); \$threshold_time: Duration := [720h];  ['Expression 17a'] \$procedure IN \$operation_biopsy_parenchymatous_organisms AND \$date_time &gt;= (\$current_date_time - \$threshold_time); </pre>
	openEHR-EHR-EVALUATION.problem-diagnosis.v1
	<pre> \$diagnosis: Terminology_code := /data[at0001]/items[at0002.1]/value; \$date_time: Date_time := /time/value; \$diagnosis_subset: Snomed_ec := [snomed_ct_ec::(&lt;&lt;417746004  Traumatic injury  OR &lt;&lt;105612003  Injury of internal organ  OR &lt;&lt;51868009  Ulcer of duodenum  OR &lt;&lt;249273002  Finding of urinary tract proper )]; \$current_date_time: Date_time := currentDateTime(); \$threshold_time: Duration := [720h];  ['Expression 17b'] \$diagnosis IN \$diagnosis_subset AND \$date_time &gt;= (\$current_date_time - \$threshold_time); </pre>

18	openEHR-EHR-ACTION.procedure.v1
	<pre> /*One of the following in the last three months: stroke, head trauma, operation in the central nervous system, definite gastrointestinal bleeding*/  \$procedure: Terminology_code := /description[at0001]/items[at0002]/value; \$date_time: Date_time := /time/value; \$operation_cns: Snomed_ec := [snomed_ct_ec::&lt;&lt;387713003  Surgical procedure : * = &lt;&lt;21483005  Structure of central nervous system ]; \$current_date_time: Date_time := currentDateTime(); \$threshold_time: Duration := [2160h]; ['Expression 18a'] \$procedure IN \$operation_cns AND \$date_time &gt;= (\$current_date_time - \$threshold_time); </pre>
	openEHR-EHR-EVALUATION.problem-diagnosis.v1
<pre> \$diagnosis: Terminology_code := /data[at0001]/items[at0002.1]/value; \$date_time: Date_time := /time/value; \$diagnosis_subset: Snomed_ec := [snomed_ct_ec::&lt;&lt;249273002  Finding of urinary tract proper  OR &lt;&lt;127295002  Traumatic brain injury  OR &lt;&lt;230690007  Cerebrovascular accident ]; \$current_date_time: Date_time := currentDateTime(); \$threshold_time: Duration := [2160h];  ['Expression 18b'] \$diagnosis IN \$diagnosis_subset AND \$date_time &gt;= (\$current_date_time - \$threshold_time); </pre>	



19	<p style="background-color: #e1eef6; margin: 0; padding: 2px;">openEHR-EHR-EVALUATION.problem-diagnosis.v1</p> <pre style="margin: 0; padding: 2px;"> /*Pregnancy, childbirth in the last month, breastfeeding (relative contraindications)*/ \$diagnosis: Terminology_code := /data[at0001]/items[at0002.1]/value; \$last_occurrence: Date := /data[at0001]/items[at0018]/items[at0020]/value; \$breastfeeding_pregnancy: Snomed_ec := [snomed_ct_ec::&lt;&lt;169741004  Breast fed  OR &lt;&lt;77386006  Pregnant ]; \$child_birth: Snomed_ec := [snomed_ct_ec::&lt;&lt;169836001 Birth of child ]; \$current_date_time: Date_time := currentDate(); \$threshold_time: Duration := [720h]; ['Expression 19'] (\$diagnosis IN \$breastfeeding_pregnancy) OR (\$diagnosis IN \$child_birth AND \$last_occurrence &gt;= (\$current_date_time - \$threshold_time)); </pre>
----	--

#### 6.4.2 FSIII Danish standard

The FSIII (Fælles Sprog III) Danish standard for the specification of guidelines for documenting healthcare observations and interventions in home care<sup>21</sup> is a common municipal method and standard for the documentation of the municipal task solution in the area of health and elderly. FSIII must contribute to better coherence and more data reuse in the municipalities IT-based care records. This is done through the implementation of uniform concepts, classifications and adapted workflows.

FSIII focuses on the interdisciplinary citizens journal, where documented information is collected or reused and shared between the various professional groups and municipal functions. The method includes both authority and supplier, and all involved professional groups and functions are responsible for updating, reusing and maintaining information that relates to the citizens course<sup>22</sup>.

<sup>21</sup> <http://www.fs3.nu/>

<sup>22</sup> <https://www.kl.dk/kommunale-opgaver/sundhed/digitalisering-paa-sundhedsomraadet/faelles-sprog-iii/>

One of the main aspects of FSIII is to associate conditions, such as ‘Chronic pain’, ‘Problems with personal care’, ‘Sleeping problems’ or ‘Hearing problems’ with a set of interventions for each condition (i.e., C-I associations). Both conditions and interventions are explicitly associated with SNOMED CT concepts by using RF2 refsets. Since the FSIII standard associates a series of conditions with a series of subsets, it fits perfectly as a use case for defining conditional value set bindings. Table 24 shows some examples of the FSIII conditions together with its associated sets of interventions (translated from the original Danish file). The whole set of C-I associations can be found in Annex 2.

ID	Condition	Interventions
1	Problems with personal care	<ul style="list-style-type: none"> <li>- Collaboration with networks</li> <li>- Rehabilitation</li> <li>- Guidance</li> <li>- Relocation and mobilization</li> <li>- Training</li> <li>- Support for ADL<sup>23</sup> activity</li> </ul>
2	Chronic pain	<ul style="list-style-type: none"> <li>- Nonpharmacological pain relief</li> <li>- Pain assessment</li> <li>- Training</li> <li>- Collaboration with networks</li> <li>- Medicine administration</li> <li>- Guidance</li> <li>- Intravenous medical treatment</li> <li>- Drug dispensing</li> </ul>
3	Sleeping problems	<ul style="list-style-type: none"> <li>- Medicine administration</li> <li>- Collaboration with networks</li> <li>- Guidance</li> <li>- Drug dispensing</li> <li>- Assessment of sleep pattern</li> </ul>

Table 24. FSIII conditions together with its associated sets of interventions

---

<sup>23</sup> Activities of Daily Living

Following with our purpose to analyse the expressivity of EHRules, we have selected three C-I associations (note that it is not our aim to exhaustively represent all C-I associations but to take a representative example) and have represented them into one conditional value set binding nested rule. Also note that we are not using archetypes in this case, so access to patient data is simulated by means of dummy paths. Names of C-I presented in Table 24 does not necessarily fit with its associated concepts descriptions in SNOMED CT. Specifically, the conditions that we have chosen as representative examples are: 'Finding related to ability to perform personal care activity', 'Chronic pain', and 'Cognitive function finding'. Below we show the EHRules expressions that represent such C-I associations.

```

/*Examples of FSIII condition-interventions (C-I) associations rules*/
context: /;
/*Patient condition and intervention*/
$condition: Terminology_code := path/to/condition; //dummy
$intervention: Terminology_code := path/to/intervention; //dummy
/*Subsets of conditions*/
$care_activity_conditions: Snomed_ec := [snomed_ct_ec::<<365178001 |Finding
related to ability to perform personal care activity|];
$chronic_pain_conditions: Snomed_ec := [snomed_ct_ec::<<82423001 |Chronic
pain|];
$cognitive_function_conditions: Snomed_ec := [snomed_ct_ec::<<373930000
|Cognitive function finding|];
/*Subsets of interventions*/
$care_activity_interventions: Snomed_ec := [snomed_ct_ec::<<304560004
|Assisting with activity of daily living| OR <<52052004 |Rehabilitation therapy| OR
<<225430005 |Procedures relating to mobility| OR <<385990006 |Support system
interventions| OR <<409073007 |Education|];
$chronic_pain_interventions: Snomed_ec := [snomed_ct_ec::<<182970005 |Pain
relief| OR <<225399009 |Pain assessment| OR <<409073007 |Education| OR
<<385990006 |Support system interventions| OR <<18629005 |Administration of

```

```
drug or medicament| OR <<431215000 |Administration of substance via
intravenous route| OR <<385796006 |Medication prefill preparation|];
$cognitive_function_interventions: Snomed_ec := [snomed_ct_ec::<<385796006
|Medication prefill preparation| OR <<304560004 |Assisting with activity of daily
living| OR <<133921002 |Emotional support| OR <<52052004 |Rehabilitation
therapy| OR <<409073007 |Education| OR <<225220004 |Communication
interventions| OR <<18629005 |Administration of drug or medicament|];
/*Conditional value set binding nested rule*/
['FSIII C-I rules']
IF $condition IN $care_activity_conditions THEN $intervention IN
$care_activity_interventions;
ELSE IF $condition IN $chronic_pain_conditions THEN $intervention IN
$chronic_pain_interventions;
ELSE IF $condition IN $cognitive_function_conditions THEN $intervention IN
$cognitive_function_interventions;
```

Basically, the conditional value set binding nested rule states that if the patient has a registered condition related with care activity, then the registered associated intervention should be included in the subset of interventions in care activity. Otherwise, if the patient has a registered condition related with chronic pain, then the registered associated intervention should be included in the subset of interventions for chronic pain. Finally, if the patient has a registered condition related with cognitive functions, then the registered associated intervention should be included in the subset of interventions for cognitive functions. Both subsets of conditions and interventions are specified by means of SNOMED CT ECs and stored in `Snomed_ec` data type variables. The evaluation of the membership of the concepts into the subsets is performed by using the inclusion ('IN') EHRules operator.

### 6.4.3 Requisition for radiology procedures

We also have represented some of the medical knowledge contained in a requisition for radiology procedures from the North Denmark Region clinical information system (see Figure 65) by using EHRules. As in the use case of the FSIII Danish standard, we are not using archetypes to represent patient data, so we are simulating access to patient data by using dummy paths.

ORDERING DEPARTMENT	
Doctor responsible for requisition (code):	<input type="text"/>
Ordering department:	<input type="text"/> Choose from the list...
Contact person regarding this requisition:	<input type="text"/>
- Phone number:	<input type="text"/>
- Radiopager:	<input type="text"/>

PATIENT	
CPR number:	<input type="text"/>
In- or outpatient:	<input type="radio"/> Outpatient <input type="radio"/> Inpatient
Allergic reaction:	<input type="text"/> <input type="checkbox"/> No applicable
Lab values:	<input type="text"/> <input type="checkbox"/> No applicable
Diabetes:	<input type="text"/> <input type="checkbox"/> No applicable
Problem incl. primary diagnosis:	
<input type="text"/>	

WANTED PROCEDURE	
Priority	<input type="radio"/> Acute <input type="radio"/> Planned <input type="radio"/> Other
Wanted procedure:	<input type="text"/> <input type="button" value="Confirm"/>
	<input type="text"/> <input type="button" value="Delete"/>
Reason for requisition:	<input type="text"/>
- Side	<input type="radio"/> Right <input type="radio"/> Left <input type="radio"/> Bilateral <input type="radio"/> None
Transportation:	<input type="radio"/> No transport <input type="radio"/> Walking <input type="radio"/> Sitting <input type="radio"/> Lying down

Figure 65. Original Danish radiology procedures requisition re-designed and translated into English

In this requisition, there exist a number of dependencies. For example:

- If the ordering department is an 'ear, nose and throat oncology department', then the procedure that you are looking for is probably something related with finding tumours in ear nose throat, e.g., 'CT scan of head', and the diagnosis is probably some 'cancer'.
- If the ordering department is a 'lung department', then probably it is required some 'scan of the lungs', and the diagnosis might be 'COPD'.

Considering this kind of dependencies, we have formally represented some examples of rules by using EHRules. As in the FSIII standard use case, it is not the objective to be exhaustive with the medical knowledge and dependencies that potentially can be contained by the requisition, but only to specify some representative examples. Specifically, we want to check whether the selected ordering department is included in the subset of SNOMED CT departments; we also want to check whether the established diagnoses are directly related or corresponds with the selected ordering department; and finally we want to assure that the chosen procedures are related with the diagnoses. Below we show the EHRules expressions that represent some examples of dependencies.

```
/* Examples of dependencies */

context: /;

/*Checks that the department exists in SNOMED CT*/
$department_name: Terminology_code := path/to/department_name; //dummy
$departments: Snomed_ec := [snomed_ct_ec::<<309912009 |Medical
department]];

/* Simple value set binding */
['Rule 1']
$department_name IN $departments;
```

```

/*Checks that the diagnosis corresponds to the department*/
$department_name: Terminology_code := path/to/department_name; //dummy
$diagnosis: Terminology_code := path/to/diagnosis;

/*Conditional value set binding nested rule*/
['Rule 2']
IF $department_name = [snomed_ct::309915006 |Cardiology department|]
THEN $diagnosis IN [snomed_ct_ec::<<56265001 |Heart disease|];
ELSE IF $department_name = [snomed_ct::309937004 |Neurology department|]
THEN $diagnosis IN [snomed_ct_ec::<<118940003 |Neurological disorder|];

/*Checks that the procedure corresponds to the diagnosis*/
$diagnosis: Terminology_code := path/to/diagnosis; //dummy
$procedure: Terminology_code := path/to/procedure; //dummy

/*Conditional value set binding nested rule*/
['Rule 3']
IF $diagnosis IN [snomed_ct_ec::<<56265001 |Heart disease|]
THEN $procedure IN [snomed_ct_ec::<<363679005 |Imaging|: * = <<80891009
|Heart structure|];
ELSE IF $diagnosis IN [snomed_ct_ec::<<118940003 |Neurological disorder|]
THEN $procedure IN [snomed_ct_ec::<<363679005 |Imaging|: * = <<69536005
|Head structure|];

```

## 6.5 VALIDATION OF EHR DATA

The EHRules language provides a syntax that may be used to specify archetype rules. As mentioned above, our objective is to provide a mechanism to define scenario-dependant constraints in clinical information models, specifically archetypes, including value set bindings. For this purpose, EHRules provides some extensions to support value set bindings, which include the ability to support SNOMED CT ECL.

Our language proposal is intended to be used in clinical information models in two main scenarios. The first one is to define constraints, including value set bindings, in order to validate the consistency of data instances of a given clinical information model (see Figure 66). The second one is to support structured data entry in such a way that the consistency of new data instances is assured (see Figure 67).

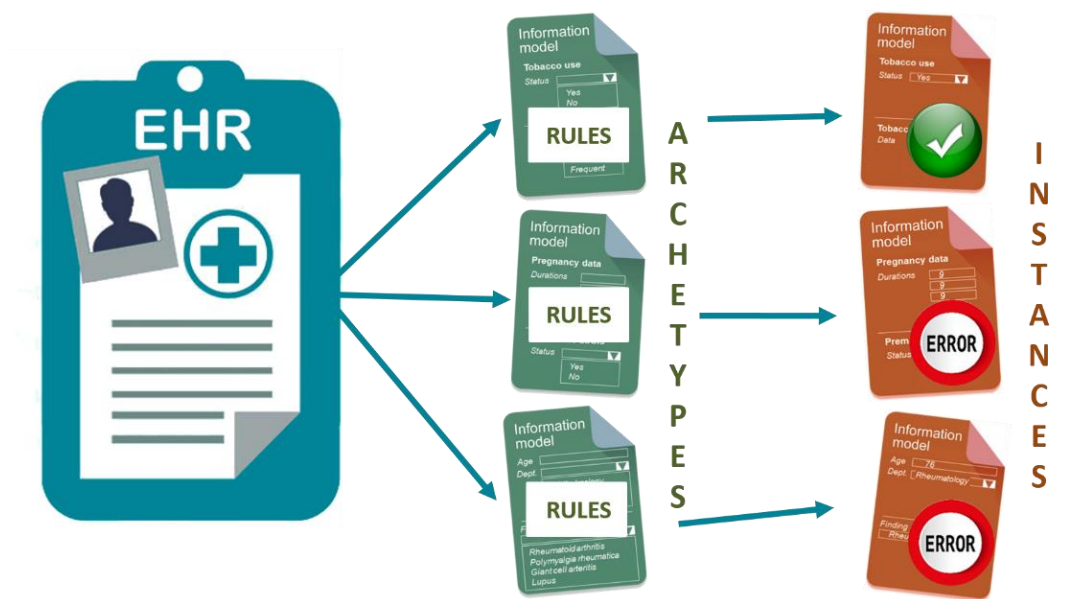


Figure 66. The definition of consistency rules in archetypes is used to validate the EHR data



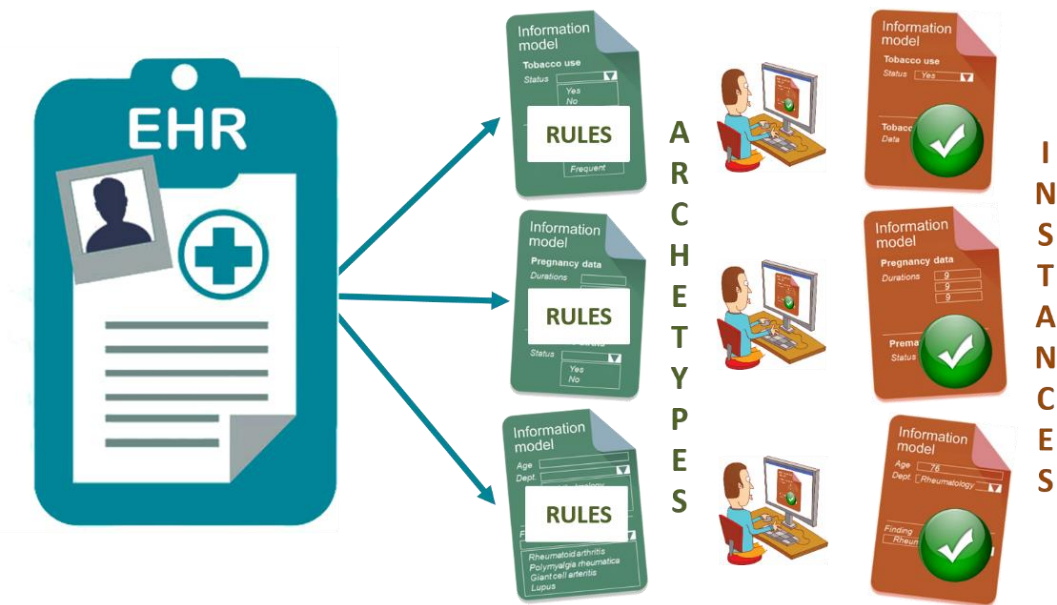


Figure 67. The definition of consistency rules in archetypes is used to support structured data entry

For the first use of such constraints, we have developed an automatic translation process that converts EHRules expressions into a schema language for XML data. For this purpose, we use Schematron as target schema language. Schematron is a language for the validation of XML documents. It allows the definition of assertion about the presence or absence of patterns in XML trees. If the assertion fails, a diagnostic message supplied by the author of the schema can be displayed. Schematron differs from most other XML schema languages (such as XML Schema or Relax NG) in that it is based on rules that use path expressions. One advantages of a rule-based approach is that the translation of archetype rules is easier in comparison with structure-based XML schema languages. The resulting Schematron programs are executed against data instances in order to determine whether instances are consistently valid or not. For the second use of such expressions, it is expected to develop an execution engine that executes EHRules expressions in order to provide users with the set of valid SNOMED CT concepts at any point of the data instances creation. Both approaches make use of SNQuery execution engine [32] to evaluate SNOMED CT ECs.

The translation process is as follows. First, the user introduces expressions, including value set bindings, in our Java translator application. The application, after validating the correctness of the syntax, translates the EHRules expressions into a set of Schematron

rules. Finally, such Schematron rules are stored into a .sch file (i.e., a Schematron file). Such Schematron file is then capable of being opened in any tool that includes a Schematron rules execution engine to validate XML files and check whether these rules are satisfied or not by the XML data. In our case, we use Oxygen XML editor [91] as testing tool.

Since the purpose of this section is to show how the translation process works, we below show a set of translation patterns. Finally, we leverage the use cases presented in section 6.4 and show how the translation process works in some representative examples of expressions and rules (a comprehensive list of examples can be found in Annex 4).

Tables 25-34 show a set of translation patterns from EHRules expressions to Schematron rules.

Variable declaration and assignation	
EHRules	Schematron
\$varName: varType := varValue;	<let name="varName" value="xs:varType(varValue)"/>

Table 25. EHRules variable declaration and assignation translated to Schematron

Data types	
EHRules	Schematron
Integer	xs:integer
Real	xs:decimal
String	xs:string
Boolean	xs:boolean
Date	xs:date
Time	xs:time
Date_time	xs:dateTime
Duration	xs:duration
Terminology_code	xs:string
Snomed_ec	xs:string

Table 26. EHRules data types translated to Schematron

Functions		Operators	
EHRules	Schematron	Arithmetic	
Path functions			
existsPath	exists	+	+
emptyPath	empty	-	-
Round function		*	*
round	round	/	div
String functions		%	mod
length	string-length	^	math:pow
trim	normalize-space	Relational	
concat (+)	concat	=	=
toUpperCase	upper-case	!= or <>	!=
toLowerCase	lower-case	<	&lt;
Date and time functions		<=	&lt;=
currentDate	current-date	>	>
currentTime	current-time	>=	>=
currentDateTime	current-dateTime	Logical	
Aggregate functions		NOT	not
count	count	AND	and
avg	avg	OR	or
min	min	FOR_ALL	every
max	max	THERE_EXISTS	some
sum	sum	Other	
		IN	in
		:	satisfies
		IF	if
		THEN	then
		ELSE	else

Table 27. EHRrules functions and operators translated to Schematron

Quantifier expressions	
EHRules	Schematron
for_all \$it IN <i>list</i> : <i>boolean_expression</i> ;	every \$it in <i>list</i> satisfies <i>boolean_expression</i>
there_exists \$it IN <i>list</i> : <i>boolean_expression</i> ;	some \$it in <i>list</i> satisfies <i>boolean_expression</i>

Table 28. EHRules quantifier expressions translated to Schematron

Rules	
EHRules	Schematron
IF <i>boolean_expression1</i> THEN { <i>boolean_expression2</i> ; <i>boolean_expression3</i> ; <i>boolean_expressionN</i> ;} ELSE { <i>boolean_expression4</i> ; <i>boolean_expression5</i> ; <i>boolean_expressionM</i> ;} ;	if ( <i>boolean_expression1</i> ) then ( <i>boolean_expression2</i> ) and ( <i>boolean_expression3</i> ) and ( <i>boolean_expressionN</i> ) else ( <i>boolean_expression4</i> ) and ( <i>boolean_expression5</i> ) and ( <i>boolean_expressionM</i> ) ;

Table 29. EHRules IF-THEN-ELSE rules translated to Schematron

Boolean expressions	
EHRules	Schematron
<i>boolean_expression</i> ; (launches an error if fails)	<assert test=" <i>boolean_expression</i> " flag="error" fpi="expr"> Failed name expression</assert>
<i>boolean_expression</i> ; (launches a warning if fails)	<assert test=" <i>boolean_expression</i> " flag="warning" fpi="expr"> Failed name expression</assert>

Table 30. EHRules boolean expressions translated to Schematron

Simple value set binding	
EHRules	Schematron
<pre>snomed_concept IN snomed_ec;</pre>	<pre>&lt;assert test="document(iri-to-uri(concat( 'snquery_ws?code=', 'snomed_concept', '&amp;url=http://snomed.info/ecl/', 'snomed_ec')) //fhir:valueBoolean/@value='true'" flag="error" fpi="expr"&gt; Failed name expression &lt;/assert&gt;</pre>

Table 31. EHRrules simple value set binding translated to Schematron

Conditional value set binding	
EHRules	Schematron
<pre>IF boolean_expression THEN snomed_concept IN snomed_ec;</pre>	<pre>&lt;assert test="(if (boolean_expression) then document(iri-to-uri(concat( 'snquery_ws?code=', 'snomed_concept', '&amp;url=http://snomed.info/ecl/', 'snomed_ec')) //fhir:valueBoolean/@value='true' else true())" flag="error" fpi="expr"&gt; Failed name expression &lt;/assert&gt;</pre>

Table 32. EHRrules conditional value set binding translated to Schematron

Dependency value set binding	
EHRules	Schematron
<pre>snomed_concept IN ECL_operator \$var_concept;</pre>	<pre>&lt;assert test="document(iri-to-uri(concat( 'snquery_ws?code=', 'snomed_concept', '&amp;url=http://snomed.info/ecl/', 'ECL_operator', \$var_concept))//fhir:valueBoolean/@value='true'" flag="error" fpi="expr"&gt; Failed name expression&lt;/assert&gt;</pre>

Table 33. EHRrules dependency value set binding translated to Schematron

Conditional plus dependency value set binding	
EHRules	Schematron
IF <i>boolean_expression</i> THEN <i>snomed_concept</i> IN <i>ECL_operator</i> \$ <i>var_concept</i> ;	<pre> &lt;assert test="(if (<i>boolean_expression</i>) then document(iri-to-uri(concat( '<i>snquery_ws?code=</i>', '<i>snomed_concept</i>', '<i>&amp;url=http://snomed.info/ecl/</i>', '<i>ECL_operator</i>', '<i>\$var_concept</i>')))//fhir:valueBoolean/@value='true'" flag="error" fpi="expr"&gt; Failed name expression&lt;/assert&gt;                     </pre>

Table 34. EHRules conditional plus dependency value set binding translated to Schematron

In Table 35, Table 36, and Table 37 we present the results after translating rules 4, 5 and 7 of the guidelines for acute stroke care use case [88].

ID	Thrombolysis contraindication	
4	Description	National Institutes of Health Stroke Scale (NIHSS) score higher than 25
	Archetype	openEHR-EHR-OBSERVATION.nihss.v1
	EHRules	<pre>\$NIHSS_score: Real := /data[at0001]/events[at0002]/data[at0003]/items[at0085]; ['Expression 4'] \$NIHSS_score &gt; 25;</pre>

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron" queryBinding='xslt2'>
  <ns prefix="xsi" uri="http://www.w3.org/2001/XMLSchema-instance"/>
  <ns prefix="math" uri="http://www.w3.org/2005/xpath-functions/math"/>
  <ns prefix="fhir" uri="http://hl7.org/fhir"/>
  <title>EL lang</title>
  <phase id="definition">
  </phase>
  <pattern id="rules">
    <rule context="/">
      <let name="NIHSS_score"
value="data[@archetype_node_id='at0001']/events/data[@archetype_node_id='at0003']/items[@archetype_node_id='at0085']"/>
      <assert test="($NIHSS_score castable as xs:decimal) and
(xs:decimal($NIHSS_score) > 25)" fpi="expr" role="error"> Failed 'Rule 2'</assert>
    </rule>
  </pattern>
</schema>
```

Table 35. Translation of the 4th contraindication into Schematron rules

ID	Thrombolysis contraindication	
5	Description	CT scan shows haemorrhage
	Archetype	openEHR-EHR-ITEM_TREE.imaging.v1
	EHRules	<pre>\$finding: Terminology_code := /items[at0002]/items[at0003]/value; \$hemorrhage: Snomed_ec := [snomed_ct_ec::&lt;&lt;50960005 Hemorrhage]]; ['Expression 5'] \$finding IN \$hemorrhage;</pre>

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron" queryBinding='xslt2'>
  <ns prefix="xsi" uri="http://www.w3.org/2001/XMLSchema-instance"/>
  <ns prefix="math" uri="http://www.w3.org/2005/xpath-functions/math"/>
  <ns prefix="fhir" uri="http://hl7.org/fhir"/>
  <title>EL lang</title>
  <phase id="definition">
  </phase>
  <pattern id="rules">
    <rule context="/">
      <let name="finding"
value="items[@archetype_node_id='at0002']/items[@archetype_node_id='at0003']/
value"/>
      <let name="hemorrhage"
value="&lt;&lt;50960005|Hemorrhage|"/>
      <assert test="($finding castable as xs:string and
$hemorrhage castable as xs:string) and document(iri-to-
uri(concat('https://snquery.veratech.es/VTTermService/ws/ValueSet/$validate-
code?code=', xs:string($finding), '&url=http://snomed.info/ecl/',
xs:string($hemorrhage))))//fhir:valueBoolean/@value='true' fpi="expr"
role="error"> Failed 'Expression 5'</assert>
    </rule>
  </pattern>
</schema>
```

Table 36. Translation of the 5th contraindication into Schematron rules



ID	Thrombolysis contraindication	
7	Description	Blood glucose is lower than 3 mmol/litre or higher than 22 mmol/litre
	Archetype	openEHR-EHR-OBSERVATION.lab_test-blood_glucose.v1
	EHRules	<pre>\$blood_glucose: Real := data[at0001]/events[at0002]/data[at0003]/ items[at0078.2]/value;  ['Expression 7'] \$blood_glucose &lt; 3.0 OR \$blood_glucose &gt; 22.0;</pre>

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron" queryBinding='xslt2'>
  <ns prefix="xsi" uri="http://www.w3.org/2001/XMLSchema-instance"/>
  <ns prefix="math" uri="http://www.w3.org/2005/xpath-functions/math"/>
  <ns prefix="fhir" uri="http://hl7.org/fhir"/>
  <title>EL lang</title>
  <phase id="definition">
  </phase>
  <pattern id="rules">
    <rule context="/">
      <let name="blood_glucose"
value="data[@archetype_node_id='at0001']/events/data[@archetype_node_id='at0003']/items[@archetype_node_id='at0078.2']/value"/>
      <assert test="($blood_glucose castable as xs:decimal) and
((xs:decimal($blood_glucose) < 3.0) or (xs:decimal($blood_glucose) > 22.0))"
fpi="expr" role="error"> Failed 'Expression 7'</assert>
    </rule>
  </pattern>
</schema>
```

Table 37. Translation of the 7th contraindication into Schematron rules

## 6.6 DISCUSSION

The openEHR EL, which is still under development, provides a syntax that may be used to specify archetype rules. As mentioned above, our objective is to provide a mechanism to define scenario-dependant constraints in clinical information models, including value set bindings. For this purpose, we have developed EHRules, an EL-based language with some extensions to support value set bindings, which include the ability to support SNOMED CT ECL.

We have divided the constraints into two types: critical (i.e., role 'error') and secondary (i.e., role 'warning'). A critical constraint needs to be satisfied by a data instance to be considered valid from a consistency point of view. A secondary constraint results in a warning if it is not satisfied, and the data instance is still considered valid even if the constraint is not met. Therefore, an instance is considered valid or not valid according with the accomplishment of both type of constraints. This implementation is intended to be incorporated into LinkEHR Interoperability Platform [92] in order to increase its ability to define advanced semantic constraints in archetypes.

One of the key points when approaching the definition of semantic rules in clinical information models to improve the consistency of EHR data is whether such rules should be specified inside of an archetype (i.e., inside of the ADL 'Rules' section) or outside, i.e., in a separate artefact. In this sense, both approaches should be studied. In this thesis, our objective is to improve the consistency and therefore the quality of patient data instances that are associated with a specific archetype. Taking this premise into account, we decided to leverage EL since its development was started thinking about the underused ADL 'Rules' section. In this context, both EL and EHRules allow the definition of expressions without referencing the containing archetype. This reference is done implicitly since all expressions are included in one particular archetype. The second approach, i.e., to specify consistency rules in a separate artefact, allows defining expressions by referencing multiple archetypes, such as in the GDL language, where the possibility of accessing different patient data instances is primordial in order to define formal representations of clinical practice guidelines. Therefore, the objective of EL/EHRules is not exactly the same as GDL's. The EHRules language must be sufficiently expressive to meet the requirements of the use

cases for which it has been designed. Nevertheless, functionality without a corresponding use case, such as allowing access to multiple archetypes, has not been included, as this increases the complexity of implementation unnecessarily.

A second relevant matter of discussion is about simple value set binding. It is legitimate to define those bindings into the 'Terminology' section of the ADL, where bindings to terminologies are defined, instead of into the 'Rules' section. In the 'Terminology' section, bindings can be defined either as an 'internal' value set consisting of at-codes, or as a value set defined in an external terminology and referenced by a binding. The main advantage while defining simple value set bindings in the 'Terminology' section is that it is possible to set bindings to any terminology rather than only to SNOMED CT. In the current version of EHRules, rules are defined in the 'Rules' section, and it is only possible to specify value set bindings to SNOMED CT. The EHRules approach allows the definition of both intensional and extensional value set bindings by leveraging the SNOMED CT ECL, which is a clear advantage. In contrast, value sets defined in the 'Terminology' section are extensional. In general, from our point of view it makes sense to define simple value set bindings inside of the 'Rules' section since they are somehow a type of IF-THEN rules where the rule antecedent is set to true. For example, the following simple value set binding, which states that the patient diagnosis should be included in the subset of calculus findings of SNOMED CT:

```
$diagnosis IN [snomed_ct_ec::<< 313413008 |Calculus finding (finding)|];
```

is equivalent to the following conditional value set binding rule:

```
IF true THEN
$diagnosis IN [snomed_ct_ec::<< 313413008 |Calculus finding (finding)|];
```



## CHAPTER 7. CONCLUSIONS AND FUTURE WORK

---

### 7.1 CONCLUSIONS

The development of the methods presented in Chapter 3 and its inclusion in the SNQuery platform facilitates the intensional definition of subsets. It also provides mechanisms for simplification and semantic validation of ECs, and visualization of resulting subsets. These mechanisms are intended to help understand the subsets and validate them. The availability of these subsets is useful to bind content between clinical information models, such as archetypes, and SNOMED CT, which is a necessary step to achieve a high level of EHR semantic interoperability [59].

However, the usefulness of ECs does not end here. We have shown in this thesis a way to enrich archetypes with domain knowledge in the form of advanced data consistency rules by means of EHRules expressions. These rules may be useful, for example, to check whether the value of an archetype node is in accordance with the value of another node, or to specify the value of a node as the result of applying an operation on a set of nodes. Furthermore, we have shown that it is also possible to specify rules involving the result subset of an EC in order to define several types of value set bindings, i.e., simple, conditional, dependency, and conditional plus dependency extensional and intensional.

In this sense, this thesis combines a series of objectives that respond to each of the particular needs that support the final purpose of enhancing the consistency of EHR.

#### **O1. To define a method for the execution of ECs over a graph-oriented database.**

We have leveraged the benefits of graph databases in general, and Neo4j in particular, which includes a powerful query language called Cypher, to persist the content of SNOMED CT. We have defined a translation process between ECL and Cypher for the evaluation of ECs over the SNOMED CT graph, which yields the intended subset of medical concepts.

#### **O2. To define methods for pre- and post-execution simplification of ECs, and semantic validation.**

We have defined three different methods to simplify ECs before they are executed (pre-execution): subsumption-based, MRCM-based, and logic definition-based; and one

method based on the mining of the result subset (post-execution). The methods based on the MRCM and the logic definition check whether the EC satisfies the rules defined in the SNOMED CT Concept Model and the logical definition of the involved concepts, respectively.

**O3. To provide a method for the visual exploration of SNOMED CT subsets.**

A graphical visualization based on the circle packing is proposed for understanding and validating subsets. This method provides information about how the concepts that make up the subset are related in terms of hierarchies and which of these hierarchies are more relevant in terms of recall and precision. The visual representation of this information may facilitate the validation of the subset at a glance, for instance by detecting irrelevant concepts or hierarchies.

**O4. To develop an EC execution platform that makes use of the methods presented in O1, O2 and O3.**

We have developed SNQuery, a web platform able to create, parse, simplify, semantically validate and execute ECs, and visualize the result subsets. In addition to the circle packing visualization method, SNQuery also incorporates a hierarchy-based and a tree-based visual representations that can be used to visualize the subsets. Additionally, it also incorporates extra functionalities, such as historic of the last executions, ECs examples section aimed at being the basis for building ECs instead of defining them from scratch, a list of MSSSI refsets, multilingual interface, mapping to ICD-10, and conversion from ECL brief to long syntax and vice versa.

**O5. To define an expression language for the specification of consistency rules in archetypes.**

We have extended the openEHR EL language with some requirements to support value set bindings between archetypes and SNOMED CT. This extensions include: ability to define IF-THEN-ELSE nested rules, specification of criticality in rules, definition of contexts to group related expressions by using paths and conditions over them, ability to specify subsets of clinical concepts intensionally in ECL, a new variable type for ECs called `Snomed_ec`, and ability to check the membership of a given concept in an intensional subset by supporting an inclusion operator. We have called this new language EHRules.

**O6. To enrich the rules section of the archetypes with the language of O5.**

The purpose of the EHRules language is to define expressions to be included in the ‘Rules’ section of archetypes to enhance the consistency of EHR, and therefore its quality. In this context, terminology is a key point, since it is a required step to bind information models with domain models to achieve high levels of semantic interoperability. Accordingly, EHRules allows the definition of simple, conditional, dependency, and conditional plus dependency extensional and intensional value set bindings between archetypes and SNOMED CT, for which definition of subsets by means of ECs is supported.

**O7. To validate the consistency of EHR by executing the expressions inside of the rules section of archetypes, including the execution of ECs by using the engine of O4.**

We have developed an automatic translation process that converts EHRules expressions into a schema language for XML data. Specifically, we use Schematron as target schema language. The resulting Schematron programs are executed against data instances in order to determine whether instances are consistently valid or not.

**7.2 FUTURE WORK**

In this thesis we have presented a mechanism to improve the consistency of EHR data through the definition of expressions inside of archetypes that represent clinical knowledge. For the validation of EHR data instances we translate EHRules constraints into a Schematron script whose execution yields as a result the set of constraints that are not satisfied. One natural continuation of this approach is the ability to rank EHR data instances from a consistency point of view, i.e., given a set of instances of the same archetype, we are able to order them from highest to lowest consistency or vice versa. To perform this, it is required to extend the EHRules language with rule weights and to define a series of consistency metrics.

Furthermore, it is important to remark some points that should be considered in the context of specifying semantic constraints with EL/EHRules. One relevant issue is related with the specialisation of archetypes. Concretely, whether EHRules expressions should be inherited and whether the expression can be extended with extra conditions, i.e., substituted by a more specific expression.

Another relevant issue is about supporting structured data entry in such a way that the consistency of new data instances is assured. From this perspective, it is required an execution engine that executes EHRules constraints in order to provide users with the set of valid SNOMED CT concepts at any point of the data instances creation.

Although addressing the mentioned issues is considered outside the scope of this thesis, it is expected to tackle them as part of the future journey of our work. Additionally, we aim to integrate SNQuery into LinkEHR. We would like to extend LinkEHR with the advanced terminology binding presented in this thesis by supporting the EHRules language, allowing both semantic and value set binding between archetypes and SNOMED CT to improve the consistency of EHR and therefore its quality



## BIBLIOGRAPHY

---

- [1] N.G. Weiskopf, C. Weng, Methods and dimensions of electronic health record data quality assessment: Enabling reuse for clinical research, *Journal of the American Medical Informatics Association*. 20 (2013). <https://doi.org/10.1136/amiajnl-2011-000681>.
- [2] R.Y. Wang, Beyond accuracy: What data quality means to data consumers, *Journal of Management Information Systems*. 12 (1996). <https://doi.org/10.1080/07421222.1996.11518099>.
- [3] K. Lee, N. Weiskopf, J. Pathak, A Framework for Data Quality Assessment in Clinical Research Datasets, *AMIA Annual Symposium Proceedings. AMIA Symposium*. 2017 (2017).
- [4] SNOMED CT Starter Guide, SNOMED International, 2017. <http://snomed.org/sg>.
- [5] SNOMED CT Expression Constraint Language Specification and Guide, SNOMED International, 2020. <http://snomed.org/ecl>.
- [6] H. Aerts, D. Kalra, C. Sáez, J.M. Ramírez-Anguita, M.A. Mayer, J.M. Garcia-Gomez, M. Durà-Hernández, G. Thienpont, P. Coorevits, Quality of hospital electronic health record (EHR) data based on the international consortium for health outcomes measurement (ICHOM) in heart failure: Pilot data quality assessment study, *JMIR Medical Informatics*. 9 (2021). <https://doi.org/10.2196/27842>.
- [7] D. Kalra, Electronic health record standards. *Yearbook of Medical Informatics*. (2006). <https://doi.org/10.1055/s-0038-1638463>.
- [8] C.G. Chute, Clinical classification and terminology: Some history and current observations, *Journal of the American Medical Informatics Association*. 7 (2000). <https://doi.org/10.1136/jamia.2000.0070298>.
- [9] K.R. Gøeg, M. Hummeluhr, An empirical approach to enhancing terminology binding - An HL7 FHIR SNOMED CT example, in: *Studies in Health Technology and Informatics*, 2018. <https://doi.org/10.3233/978-1-61499-852-5-206>.
- [10] T. Beale, Archetypes: Constraint-based Domain Models for Future- proof Information Systems, *OOPSLA 2002 Workshop on Behavioural Semantics*. (2001).
- [11] D.K. Sharma, H.R. Solbrig, C. Tao, C. Weng, C.G. Chute, G. Jiang, Building a semantic web-based metadata repository for facilitating detailed clinical modeling

- in cancer genome studies, *Journal of Biomedical Semantics*. 8 (2017).  
<https://doi.org/10.1186/s13326-017-0130-4>.
- [12] M.D.C. Legaz-García, C. Martínez-Costa, M. Menárguez-Tortosa, J.T. Fernández-Breis, A semantic web based framework for the interoperability and exploitation of clinical models and EHR data, *Knowledge-Based Systems*. 105 (2016).  
<https://doi.org/10.1016/j.knosys.2016.05.016>.
- [13] D.K. Sharma, H.R. Solbrig, E. Prud'hommeaux, J. Pathak, G. Jiang, Standardized Representation of Clinical Study Data Dictionaries with CIMI Archetypes, *AMIA Annual Symposium Proceedings. AMIA Symposium*. 2016 (2016).
- [14] M.B. Späth, J. Grimson, Applying the archetype approach to the database of a biobank information management system, *International Journal of Medical Informatics*. 80 (2011). <https://doi.org/10.1016/j.ijmedinf.2010.11.002>.
- [15] A.L.L.R.A.K.D. Rossander, A State-of-the Art Review of SNOMED CT Terminology Binding and Recommendations for Practice and Research, *Methods of Information in Medicine*. (2021).
- [16] C.H. Ivory, Mapping perinatal nursing process measurement concepts to standard terminologies, *CIN - Computers Informatics Nursing*. 34 (2016).  
<https://doi.org/10.1097/CIN.0000000000000243>.
- [17] K. Bernstein, M. Bruun-Rasmussen, S. Vingtoft, A method for specification of structured clinical content in electronic health records, in: *Studies in Health Technology and Informatics*, 2006.
- [18] D.B. Hier, S.U. Brint, A Neuro-ontology for the neurological examination, *BMC Medical Informatics and Decision Making*. 20 (2020).  
<https://doi.org/10.1186/s12911-020-1066-7>.
- [19] G. Wade, S.T. Rosenbloom, Experiences mapping a legacy interface terminology to SNOMED CT, in: *BMC Medical Informatics and Decision Making*, 2008.  
<https://doi.org/10.1186/1472-6947-8-S1-S3>.
- [20] a Randorff Højen, K. Rosenbeck Gøeg, Snomed CT implementation. Mapping guidelines facilitating reuse of data. *Methods of Information in Medicine*. 51 (2012).
- [21] F. Bakhshi-Raiez, L. Ahmadian, R. Cornet, E. de Jonge, N.F. de Keizer, Construction of an interface terminology on SNOMED CT: Generic approach and its application in intensive care, *Methods of Information in Medicine*. 49 (2010).  
<https://doi.org/10.3414/ME09-01-0057>.
- [22] S. Maulden, P. Greim, O. Bouhaddou, P. Warnekar, L. Megas, F. Parrish, M.J. Lincoln, Using SNOMED CT as a mediation terminology: Mapping issues, lessons

- learned, and next steps toward achieving semantic interoperability, in: CEUR Workshop Proceedings, 2008.
- [23] D.S.S.K.D. et al. Kalra, ASSESS CT Recommendations— Assessing SNOMED CT for Large Scale eHealth Deployments in the EU, 2016.
  - [24] SNOMED CT Compositional Grammar Specification and Guide, SNOMED International, 2020. <http://snomed.org/scg>.
  - [25] SNOMED CT Technical Implementation Guide, SNOMED International, 2015. <http://snomed.org/tig>.
  - [26] SNOMED CT Machine Readable Concept Model Specification, SNOMED International, 2017. <http://snomed.org/mrcm>.
  - [27] Semantic interoperability for better health and safer healthcare: deployment and research roadmap for Europe, 2009. <https://doi.org/10.2759/38514>.
  - [28] A. Lysenko, I.A. Roznovať, M. Saqi, A. Mazein, C.J. Rawlings, C. Auffray, Representing and querying disease networks using graph databases, *BioData Mining*. 9 (2016). <https://doi.org/10.1186/s13040-016-0102-8>.
  - [29] W.S. Campbell, J. Pedersen, J.C. McClay, P. Rao, D. Bastola, J.R. Campbell, An alternative database approach for management of SNOMED CT and improved patient data queries, *Journal of Biomedical Informatics*. 57 (2015). <https://doi.org/10.1016/j.jbi.2015.08.016>.
  - [30] I. Robinson, J. Webber, E. Eifrem, Graph Databases. New opportunities for connected data, 2015. <https://doi.org/10.1016/b978-0-12-407192-6.00003-0>.
  - [31] The Neo4j Getting Started, Neo4j, 2020. <https://neo4j.com/docs/getting-started/current/>.
  - [32] V.M. Giménez-Solano, J.A. Maldonado, D. Boscá, S. Salas-García, M. Robles, Definition and validation of SNOMED CT subsets using the expression constraint language, *Journal of Biomedical Informatics*. 117 (2021). <https://doi.org/10.1016/j.jbi.2021.103747>.
  - [33] C. Vicknair, X. Nan, M. Macias, Y. Chen, Z. Zhao, D. Wilkins, A comparison of a graph database and a relational database: A data provenance perspective, in: *Proceedings of the Annual Southeast Conference*, 2010. <https://doi.org/10.1145/1900008.1900067>.
  - [34] R. Angles, The Property Graph Model, *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management (CEUR Workshop Proceedings)*. (2018).

- [35] M.A. Rodriguez, P. Neubauer, Constructions from dots and lines, *Bulletin of the American Society for Information Science and Technology*. 36 (2010). <https://doi.org/10.1002/bult.2010.1720360610>.
- [36] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, A. Taylor, Cypher: An evolving query language for property graphs, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data, Association for Computing Machinery, 2018*: pp. 1433–1445. <https://doi.org/10.1145/3183713.3190657>.
- [37] The Neo4j Cypher Manual, Neo4j, 2020. <https://neo4j.com/docs/cypher-manual/current/>.
- [38] C.R. Collins, K. Stephenson, A circle packing algorithm, *Computational Geometry: Theory and Applications*. 25 (2003). [https://doi.org/10.1016/S0925-7721\(02\)00099-8](https://doi.org/10.1016/S0925-7721(02)00099-8).
- [39] GitHub - dagrejs/dagre-d3: A D3-based renderer for Dagre. <https://github.com/dagrejs/dagre-d3>.
- [40] SNOMED International, SNOMED CT Browser. <https://browser.ihtsdotools.org/>.
- [41] M. Jacomy, T. Venturini, S. Heymann, M. Bastian, ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software, *PLoS ONE*. 9 (2014). <https://doi.org/10.1371/journal.pone.0098679>.
- [42] I. Berges, J. Bermudez, A. Illarramendi, Binding SNOMED CT Terms to Archetype Elements, *Methods of Information in Medicine*. 54 (2015). <https://doi.org/10.3414/me13-02-0022>.
- [43] P. Sciences, R. Qamar, A. Rector, Semantic Mapping of Clinical Model Data To Biomedical Terminologies To Facilitate, *Healthcare Computing 2007 Conference*. (2007).
- [44] M. Meizoso García, J.L. Iglesias Allones, D. Martínez Hernández, M.J. Taboada Iglesias, Semantic similarity-based alignment between clinical archetypes and SNOMED CT: An application to observations, *International Journal of Medical Informatics*. 81 (2012). <https://doi.org/10.1016/j.ijmedinf.2012.02.007>.
- [45] S. Yu, D. Berry, J. Bisbal, An investigation of semantic links to Archetypes in an external clinical terminology through the construction of terminological “Shadows,” in: *Proceedings of the IADIS International Conference E-Health 2010, EH, Part of the IADIS Multi Conference on Computer Science and Information Systems 2010, MCCSIS 2010, 2010*.
- [46] L. Chu, V. Kannan, M.A. Basit, D.J. Schaefflein, A.R. Ortuzar, J.F. Glorioso, J.R. Buchanan, D.L. Willett, SNOMED CT Concept Hierarchies for Computable Clinical

Phenotypes From Electronic Health Record Data: Comparison of Intensional Versus Extensional Value Sets, *JMIR Medical Informatics*. 7 (2019) e11487. <https://doi.org/10.2196/11487>.

- [47] M.A. Casteleiro, D. Tsarkov, B. Parsia, U. Sattler, Using semantic web technologies to underpin the SNOMED CT query language, in: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017. [https://doi.org/10.1007/978-3-319-71078-5\\_20](https://doi.org/10.1007/978-3-319-71078-5_20).
- [48] K. Dentler, R. Cornet, A. ten Teije, N. de Keizer, Comparison of reasoners for large ontologies in the OWL 2 EL profile, *Semantic Web*. 2 (2011). <https://doi.org/10.3233/SW-2011-0034>.
- [49] F. Holzschuher, R. Peinl, Querying a graph database - Language selection and performance considerations, *Journal of Computer and System Sciences*. 82 (2016). <https://doi.org/10.1016/j.jcss.2015.06.006>.
- [50] O. Bodenreider, D. Nguyen, P. Chiang, P. Chuang, M. Madden, R. Winnenburg, R. McClure, S. Emrick, I. D'Souza, The NLM value set authority center, in: *Studies in Health Technology and Informatics*, 2013. <https://doi.org/10.3233/978-1-61499-289-9-1224>.
- [51] K.J. Peterson, G. Jiang, S.M. Brue, F. Shen, H. Liu, Mining Hierarchies and Similarity Clusters from Value Set Repositories, *AMIA Annual Symposium Proceedings. AMIA Symposium. 2017* (2017).
- [52] D.L. Willett, V. Kannan, L. Chu, J.R. Buchanan, F.T. Velasco, J.D. Clark, J.S. Fish, A.R. Ortuzar, J.E. Youngblood, D.G. Bhat, M.A. Basit, SNOMED CT Concept Hierarchies for Sharing Definitions of Clinical Conditions Using Electronic Health Record Data, *Applied Clinical Informatics*. 9 (2018) 667–682. <https://doi.org/10.1055/s-0038-1668090>.
- [53] K.W. Fung, J. Xu, S. Gold, The Use of Inter-terminology Maps for the Creation and Maintenance of Value Sets, *AMIA Annual Symposium Proceedings. AMIA Symposium. 2019* (2019).
- [54] D.A. Springate, E. Kontopantelis, D.M. Ashcroft, I. Olier, R. Parisi, E. Chamapiwa, D. Reeves, ClinicalCodes: An online clinical codes repository to improve the validity and reproducibility of research using electronic medical records, *PLoS ONE*. 9 (2014). <https://doi.org/10.1371/journal.pone.0099825>.
- [55] R. Williams, B. Brown, E. Kontopantelis, T. van Staa, N. Peek, Term sets: A transparent and reproducible representation of clinical code sets, *PLoS ONE*. 14 (2019). <https://doi.org/10.1371/journal.pone.0212291>.
- [56] Ontoserver Query. <https://ontoserver.csiro.au/shrimp/ecl.html>.

- [57] Slang. <http://slang.snomedic.com:8080/yats/>.
- [58] GitHub - slaverman/SnoLyze: SNOMED CT Expression Constraint Language Execution Engine in R. <https://github.com/slaverman/SnoLyze>.
- [59] A. Muñoz Carrero, A. Romero Gutiérrez, G. Marco Cuenca, A. Abad Acebedo, J. Cáceres Tello, R. Sánchez de Madariaga, P. Serrano Balazote, D. Moner Cano, J. Maldonado Segura, Manual práctico de interoperabilidad semántica para entornos sanitarios basada en arquetipos, Unidad de Investigación en Telemedicina y e-Salud. Instituto de Salud Carlos III. Ministerio de Economía y Competitividad. Madrid, 2013.
- [60] S.W. Tu, M.A. Musen, A flexible approach to guideline modeling. Proceedings / AMIA Annual Symposium. AMIA Symposium. (1999).
- [61] M. Peleg, Computer-interpretable clinical guidelines: A methodological review, Journal of Biomedical Informatics. 46 (2013). <https://doi.org/10.1016/j.jbi.2013.06.009>.
- [62] C. Weng, S.W. Tu, I. Sim, R. Richesson, Formal representation of eligibility criteria: A literature review, Journal of Biomedical Informatics. 43 (2010). <https://doi.org/10.1016/j.jbi.2009.12.004>.
- [63] J. Fox, N. Johns, A. Rahmzadeh, Disseminating medical knowledge: The PROforma approach, Artificial Intelligence in Medicine. 14 (1998). [https://doi.org/10.1016/S0933-3657\(98\)00021-9](https://doi.org/10.1016/S0933-3657(98)00021-9).
- [64] J. Fox, N. Johns, A. Rahmzadeh, R. Thomson, PROforma: A method and language for specifying clinical guidelines and protocols, in: Studies in Health Technology and Informatics, 1996. <https://doi.org/10.3233/978-1-60750-878-6-516>.
- [65] A. Seyfang, S. Miksch, M. Marcos, Combining diagnosis and treatment using ASBRU, in: International Journal of Medical Informatics, 2002. [https://doi.org/10.1016/S1386-5056\(02\)00064-3](https://doi.org/10.1016/S1386-5056(02)00064-3).
- [66] S. Miksch, Y. Shahar, P. Johnson, Asbru: a task-specific, intention-based, and time-oriented language for representing skeletal plans, 7th Workshop on Knowledge Engineering: Methods & Languages. (1997).
- [67] M. Peleg, A.A. Boxwala, O. Ogunyemi, Q. Zeng, S. Tu, R. Lacson, E. Bernstam, N. Ash, P. Mork, L. Ohno-Machado, E.H. Shortliffe, R.A. Greenes, GLIF3: the evolution of a guideline representation format. Proceedings / AMIA Annual Symposium. AMIA Symposium. (2000).
- [68] M. Peleg, A.A. Boxwala, E. Bernstam, S. Tu, R.A. Greenes, E.H. Shortliffe, Sharable representation of clinical guidelines in GLIF: Relationship to the Arden Syntax,

- Journal of Biomedical Informatics. 34 (2001).  
<https://doi.org/10.1006/jbin.2001.1016>.
- [69] A.A. Boxwala, M. Peleg, S. Tu, O. Ogunyemi, Q.T. Zeng, D. Wang, V.L. Patel, R.A. Greenes, E.H. Shortliffe, GLIF3: A representation format for sharable computer-interpretable clinical practice guidelines, *Journal of Biomedical Informatics*. 37 (2004). <https://doi.org/10.1016/j.jbi.2004.04.002>.
- [70] S.W. Tu, J. Campbell, M.A. Musen, The SAGE guideline modeling: Motivation and methodology, in: *Studies in Health Technology and Informatics*, 2004.  
<https://doi.org/10.3233/978-1-60750-944-8-167>.
- [71] S.W. Tu, J.R. Campbell, J. Glasgow, M.A. Nyman, R. McClure, J. McClay, C. Parker, K.M. Hrabak, D. Berg, T. Weida, J.G. Mansfield, M.A. Musen, R.M. Abarbanel, The SAGE Guideline Model: Achievements and Overview, *Journal of the American Medical Informatics Association*. 14 (2007). <https://doi.org/10.1197/jamia.M2399>.
- [72] S.W. Tu, M.A. Musen, Modeling data and knowledge in the EON guideline architecture, in: *Studies in Health Technology and Informatics*, 2001.  
<https://doi.org/10.3233/978-1-60750-928-8-280>.
- [73] M.A. Musen, S.W. Tu, A.K. Das, Y. Shahar, EON: A Component-Based Approach to Automation of Protocol-Directed Therapy, *Emerging Infectious Diseases*. 3 (1996).  
<https://doi.org/10.1136/jamia.1996.97084511>.
- [74] D.R. Sutton, P. Taylor, K. Earle, Evaluation of PROforma as a language for implementing medical guidelines in a practical context, *BMC Medical Informatics and Decision Making*. 6 (2006). <https://doi.org/10.1186/1472-6947-6-20>.
- [75] N. Iglesias, J.M. Juarez, M. Campos, Comprehensive analysis of rule formalisms to represent clinical guidelines: Selection criteria and case study on antibiotic clinical guidelines, *Artificial Intelligence in Medicine*. 103 (2020).  
<https://doi.org/10.1016/j.artmed.2019.101741>.
- [76] G. Hripcsak, P. Ludemann, T.A. Pryor, O.B. Wigertz, P.D. Clayton, Rationale for the Arden syntax, *Computers and Biomedical Research*. 27 (1994).  
<https://doi.org/10.1006/cbmr.1994.1023>.
- [77] G. Hripcsak, O.B. Wigertz, P.D. Clayton, Origins of the Arden Syntax, *Artificial Intelligence in Medicine*. 92 (2018).  
<https://doi.org/10.1016/j.artmed.2015.05.006>.
- [78] Arden Syntax v2.10 (Health Level Seven Arden Syntax for Medical Logic Systems, Version 2.10).  
[https://www.hl7.org/implement/standards/product\\_brief.cfm?product\\_id=372](https://www.hl7.org/implement/standards/product_brief.cfm?product_id=372).

- [79] S. Kraus, M. Rosenbauer, L. Schröder, T. Bürkle, K.P. Adlassnig, D. Toddenroth, A detailed analysis of the Arden Syntax expression grammar, *Journal of Biomedical Informatics*. 83 (2018). <https://doi.org/10.1016/j.jbi.2018.05.008>.
- [80] M. Samwald, K. Fehre, J. de Bruin, K.P. Adlassnig, The Arden Syntax standard for clinical decision support: Experiences and directions, *Journal of Biomedical Informatics*. 45 (2012). <https://doi.org/10.1016/j.jbi.2012.02.001>.
- [81] R.A. Jenders, K.P. Adlassnig, K. Fehre, P. Haug, Evolution of the Arden Syntax: Key Technical Issues from the Standards Development Organization Perspective, *Artificial Intelligence in Medicine*. 92 (2018). <https://doi.org/10.1016/j.artmed.2016.08.001>.
- [82] R.A. Jenders, Evaluation of SNOMED CT as a reference terminology for standardized data queries in the Arden syntax, in: *Studies in Health Technology and Informatics*, 2017. <https://doi.org/10.3233/978-1-61499-830-3-1326>.
- [83] HL7 Version 3 Standard: GELLO, A Common Expression Language, Release 2. [https://www.hl7.org/implement/standards/product\\_brief.cfm?product\\_id=5](https://www.hl7.org/implement/standards/product_brief.cfm?product_id=5).
- [84] OMG, Object Constraint Language Specification v2.2, Management. 03 (2010).
- [85] openEHR Guideline Definition Language (GDL). <https://specifications.openehr.org/releases/CDS/latest/GDL2.html>.
- [86] openEHR Expression Language (EL). [https://specifications.openehr.org/releases/LANG/latest/expression\\_language.html](https://specifications.openehr.org/releases/LANG/latest/expression_language.html).
- [87] B. Sharif, J.I. Maletic, An eye tracking study on camelcase and under-score identifier styles, in: *IEEE International Conference on Program Comprehension*, 2010. <https://doi.org/10.1109/ICPC.2010.41>.
- [88] N. Anani, R. Chen, T. Prazeres Moreira, S. Koch, Retrospective checking of compliance with practice guidelines for acute stroke care: A novel experiment using openEHR's Guideline Definition Language, *BMC Medical Informatics and Decision Making*. 14 (2014). <https://doi.org/10.1186/1472-6947-14-39>.
- [89] A.R. Højen, K.R. Gøeg, P.B. Elberg, Re-use of SNOMED CT subset in development of the Danish national standard for home care nursing problems, in: *Studies in Health Technology and Informatics*, 2015. <https://doi.org/10.3233/978-1-61499-512-8-140>.
- [90] K.R. Gøeg, P.B. Elberg, A.R. Højen, U.L. Eskildsen, SNOMED CT as Reference Terminology in the Danish National Home Care Documentation Standard, *Studies in Health Technology and Informatics*. 235 (2017). <https://doi.org/10.3233/978-1-61499-753-5-461>.



- [91] Y.X. Li, K.T. Chau, Z.H. Wei, Q. bin Kang, A comparative study on two XML editors (oxygen and ultraedit), in: ACM International Conference Proceeding Series, 2019. <https://doi.org/10.1145/3309074.3309126>.
  
- [92] J.A. Maldonado, D. Moner, D. Boscá, J.T. Fernández-Breis, C. Angulo, M. Robles, LinkEHR-Ed: A multi-reference model archetype editor based on formal semantics, *International Journal of Medical Informatics*. 78 (2009). <https://doi.org/10.1016/j.ijmedinf.2009.03.006>.



## ANNEX 1 - LIST OF ECS EXAMPLES

---

### *EC example 1*

<< 27550009 | Disorder of blood vessel (disorder)|

### *EC example 2*

< 362965005 | Disorder of body system (disorder)|:  
363698007 | Finding site (attribute)| = << 59820001 | Blood vessel structure (body structure)|

### *EC example 3*

< 404684003 | Clinical finding (finding)|:  
{363698007 | Finding site (attribute)| = << 39057004 | Pulmonary valve|,  
116676008 | Associated morphology (attribute)| = << 415582006 | Stenosis|},  
{363698007 | Finding site (attribute)| = << 58095006 | Interatrial septum structure|,  
116676008 | Associated morphology| = << 396351009 | Congenital septal defect|}

### *EC example 4*

< 473011001 | Allergic condition (disorder)|:  
[2..\*] 246075003 | Causative agent (attribute)| = < 105590001 | Substance (substance)|

### *EC example 5*

< 255620007 | Food (substance)|:  
R 246075003 | Causative agent (attribute)| = < 473011001 | Allergic condition (disorder)|

### *EC example 6*

871742003 | Tetanus vaccine (medicinal product)| OR 871729003 | Diphtheria vaccine (medicinal product)| OR 871758000 | Pertussis vaccine (medicinal product)|

EC example 7

(< 362965005 |Disorder of body system (disorder)| OR  
< 414027002 |Disorder of hematopoietic structure (disorder)|):  
363698007 |Finding site (attribute)| =  
<< 59820001 |Blood vessel structure (body structure)|,  
363698007 |Finding site (attribute)| = << 87784001 |Soft tissues (body structure)|

EC example 8

<< 404684003 |Clinical finding (finding)|:  
39133001 |With severity (attribute)| = << 272141005 |Severities (qualifier value)|

EC example 9

<< 404684003 |Clinical finding (finding)|:  
39133001 |With severity (attribute)| = << 272141005 |Severities (qualifier value)| OR  
246075003 |Causative agent (attribute)| = << 410942007 |Drug or medicament  
(substance)|

EC example 10

< 106063007 |Cardiovascular finding (finding)|:  
363698007 |Finding site (attribute)| = < 91723000 |Anatomical structure (body  
structure)|

EC example 11

< 404684003 |Clinical finding (finding)|:  
363698007 |Finding site (attribute)| = << 82094008 |Lower respiratory tract structure  
(body structure)|,  
116676008 |Associated morphology (attribute)| = << 367651003 |Malignant  
neoplasm (morphologic abnormality)|

EC example 12

< 301226008 |Lower respiratory tract finding (finding)|:  
363698007 |Finding site (attribute)| = << 82094008 |Lower respiratory tract structure  
(body structure)|,  
116676008 |Associated morphology (attribute)| = << 367651003 |Malignant  
neoplasm (morphologic abnormality)|

*EC example 13*

< 301226008 |Lower respiratory tract finding (finding)|:  
116676008 |Associated morphology (attribute)| = << 367651003 |Malignant  
neoplasm (morphologic abnormality)|

*EC example 14*

707480001 |Chronic hemolytic anemia (disorder)| IN < 64572001 |Disease  
(disorder)|: 363714003 |Interprets (attribute)| = 14089001 |Red blood cell count  
(procedure)|

*EC example 15*

< 138875005 |SNOMED CT Concept|: 363698007 |Finding site (attribute)| =  
< 127903009 |Male genital organ structure (body structure)|

*EC example 16*

< 249230006 |Male genitalia finding (finding)|: 363698007 |Finding site (attribute)| =  
< 127903009 |Male genital organ structure (body structure)|



## ANNEX 2 - EHRULES ABNF SYNTAX SPECIFICATION

---

**expression** = context colon ( path / divide ) end [ condition colon  
contextBooleanExpression end ] 1\*( ( [ exprname ] [ role ] booleanExpression end ) / ( variableDeclarationAssignment end ) / ( listVariableDeclarationAssignment end ) / ( [ exprname ] [ role ] ruleExpression ) )

**ruleExpression** = if ruleAntecedent then ruleConsequent [ else ( ruleConsequent / ruleExpression ) ]

**ruleAntecedent** = booleanExpression

**ruleConsequent** = ( booleanExpression end ) / ( ocur 1\*( booleanExpression end ) ccur )

**quantificationExpression** = ( existentialQuantifier / universalQuantifier ) variable inclusion ( variable / path ) colon booleanExpression

**variableDeclarationAssignment** = variable colon ( ( duration assig ( durationExpression / null ) ) / ( dateTime assig ( dateTimeTypeExpression / null ) ) / ( date assig ( dateTypeExpression / null ) ) / ( time assig ( timeTypeExpression / null ) ) / ( integer assig ( arithmeticExpression / null ) ) / ( real assig ( arithmeticExpression / null ) ) / ( boolean assig ( booleanExpression / null ) ) / ( string assig ( stringExpression / null ) ) / ( terminologyCode assig ( snomedConceptExpresion / null ) ) / ( snomedEc assig ( snomedEcExpression / null ) ) )

**listVariableDeclarationAssignment** = variable colon list less ( ( ( integer / real ) greater assig ( null / variable / ( ocur ( arithmeticExpression / null ) \*( comma ( arithmeticExpression / null ) ) ccur ) ) ) / ( boolean greater assig ( null / variable / ( ocur ( booleanExpression / null ) \*( comma ( booleanExpression / null ) ) ccur ) ) ) / ( string greater assig ( null / variable / ( ocur ( stringExpression / null ) \*( comma ( stringExpression / null ) ) ccur ) ) ) / ( terminologyCode greater assig ( null / variable / ( ocur ( snomedConceptExpresion / null ) \*( comma ( snomedConceptExpresion / null ) ) ccur ) ) ) / ( snomedEc greater assig ( null / variable / ( ocur ( snomedEcExpression / null ) \*( comma ( snomedEcExpression / null ) ) ccur ) ) ) / ( date greater assig ( null /

*variable* / ( *ocur* ( *dateTypeExpression* / *null* ) \*( *comma* ( *dateTypeExpression* / *null* ) )  
*ccur* ) ) / ( *time greater assig* ( *null* / *variable* / ( *ocur* ( *timeTypeExpression* / *null* ) \*(  
*comma* ( *timeTypeExpression* / *null* ) ) *ccur* ) ) ) / ( *dateTime greater assig* ( *null* /  
*variable* / ( *ocur* ( *dateTimeTypeExpression* / *null* ) \*( *comma* (  
*dateTimeTypeExpression* / *null* ) ) *ccur* ) ) ) / ( *duration greater assig* ( *null* / *variable* / (  
*ocur* ( *durationExpression* / *null* ) \*( *comma* ( *durationExpression* / *null* ) ) *ccur* ) ) ) )  
***booleanExpression*** = *unaryBoolean* \**subBooleanExpression*  
***subBooleanExpression*** = ( *conjunction* / *disjunction* ) *unaryBoolean*  
***unaryBoolean*** = ( *not elementBoolean* ) / *elementBoolean*  
***elementBoolean*** = *existsPathFunction* / *emptyPathFunction* / *relationalExpression* /  
*INExpression* / ( *opar booleanExpression cpar* ) / *quantificationExpression* / *bool* / *path*  
/ *variable*  
***relationalExpression*** = ( *arithmeticExpression* ( *equal* / *notequal* / *less* / *lessequal* /  
*greater* / *greaterequal* ) *arithmeticExpression* ) / ( *stringExpression* ( *equal* / *notequal* )  
*stringExpression* ) / ( *snomedConceptExpresion* ( *equal* / *notequal* )  
*snomedConceptExpresion* ) / *relationalExpressionBoolean* / ( *dateTypeExpression* (  
*equal* / *notequal* / *less* / *lessequal* / *greater* / *greaterequal* ) *dateTypeExpression* ) / (  
*timeTypeExpression* ( *equal* / *notequal* / *less* / *lessequal* / *greater* / *greaterequal* )  
*timeTypeExpression* ) / ( *dateTimeTypeExpression* ( *equal* / *notequal* / *less* / *lessequal* /  
*greater* / *greaterequal* ) *dateTimeTypeExpression* ) / ( *durationExpression* ( *equal* /  
*notequal* ) *durationExpression* ) / ( *opar relationalExpression cpar* )  
***relationalExpressionBoolean*** = ( ( [ *not* ] *INExpression* ) / *variable* / *path* / *bool* /  
*existsPathFunction* / *emptyPathFunction* ) / ( *opar* ( ( [ *not* ] *INExpression* ) / *variable* /  
*path* / *bool* / *existsPathFunction* / *emptyPathFunction* ) *cpar* ( *equal* / *notequal* ) ( ( [ *not* ]  
*INExpression* ) / *variable* / *path* / *bool* / *existsPathFunction* / *emptyPathFunction* )  
) / ( *opar* ( ( [ *not* ] *INExpression* ) / *variable* / *path* / *bool* / *existsPathFunction* /  
*emptyPathFunction* ) *cpar* )  
***arithmeticExpression*** = *term* \**subArithmeticExpression*  
***subArithmeticExpression*** = ( *plus* / *minus* ) *term*  
***term*** = *expTerm* \**subTerm*  
***subTerm*** = ( *multiply* / *divide* / *modulus* ) *expTerm*



**expTerm** = unary \*expSubTerm  
**expSubTerm** = exponent unary  
**unary** = ( ( minus / plus ) ( element / roundFunction / variable / path / lengthFunction / countFunction / avgFunction / minFunction / maxFunction / sumFunction ) ) / ( element / roundFunction / variable / path / lengthFunction / countFunction / avgFunction / minFunction / maxFunction / sumFunction ) )  
**element** = constant / ( opar arithmeticExpression cpar )  
**roundFunction** = round opar arithmeticExpression comma constant cpar  
**countFunction** = count opar ( variable / path ) cpar  
**avgFunction** = avg opar ( variable / path ) cpar  
**minFunction** = min opar ( variable / path ) cpar  
**maxFunction** = max opar ( variable / path ) cpar  
**sumFunction** = sum opar ( variable / path ) cpar  
**lengthFunction** = length opar ( text / variable / path ) cpar  
**currentDateFunction** = currentdate opar cpar  
**currentTimeFunction** = currenttime opar cpar  
**currentDateTimeFunction** = currentdatetime opar cpar  
**existsPathFunction** = existspath opar path cpar  
**emptyPathFunction** = emptypath opar path cpar  
**INExpression** = snomedConceptExpresion inclusion snomedEcExpression  
**dateTypeExpression** = variable / path / currentDateFunction / ( obra constant minus constant minus constant cbra ) / ( opar dateTypeExpression cpar )  
**timeTypeExpression** = variable / path / currentTimeFunction / ( obra constant colon constant colon constant cbra ) / ( opar timeTypeExpression cpar )  
**dateTimeTypeExpression** = variable / path / currentDateTimeFunction / ( obra constant minus constant minus constant constant colon constant colon constant cbra ) / ( opar dateTimeTypeExpression cpar )  
**durationExpression** = variable / path / ( obra [ minus ] years months days hours minutes seconds cbra ) / ( opar durationExpression cpar )  
**stringExpression** = concatExpression / text / variable / path / toUpperCaseFunction / toLowerCaseFunction / trimFunction / ( opar stringExpression cpar )

```
concatExpression = ( text / variable / path ) 1*subConcatExpression
subConcatExpression = plus ( text / variable / path )
toUpperCaseFunction = touppercase opar ( text / variable / path ) cpar
toLowerCaseFunction = tolowercase opar ( text / variable / path ) cpar
trimFunction = trim opar ( text / variable / path ) cpar
snomedConceptExpression = ( obra snomedheader constant [ desc ] cbra ) / variable /
path
snomedEcExpression = ( obra echeader 1*( constant / nonDigit ) cbra ) / variable /
path
contextBooleanExpression = contextUnaryBoolean *contextSubBooleanExpression
contextSubBooleanExpression = ( conjunction / disjunction ) contextUnaryBoolean
contextUnaryBoolean = ( not contextElementBoolean ) / contextElementBoolean
contextElementBoolean = contextRelationalExpression / ( opar
contextBooleanExpression cpar )
contextRelationalExpression = ( countFunction / avgFunction / minFunction /
maxFunction / sumFunction / path / decimal ) ( equal / notequal / less / lessequal /
greater / greaterequal ) ( text / constant )
comment = ( "/" *( digit / nonDigit ) ) / ( "/" *( digit / nonDigit ) "*" )
years = constant "Y"
months = constant "M"
days = constant "D"
hours = constant "h"
minutes = constant "m"
seconds = constant "s"
boolean = "Boolean"
integer = "Integer"
real = "Real"
date = "Date"
time = "Time"
dateTime = "Date_time"
duration = "Duration"
```

```

string = "String"
terminologyCode = "Terminology_code"
snomedEc = "Snomed_ec"
not = ( "not" / "NOT" )
conjunction = ( "and" / "AND" )
disjunction = ( "or" / "OR" )
equal = "="
notequal = "!=" / "<>"
less = "<"
lessequal = "<="
greater = ">"
greaterequal = ">="
plus = "+"
minus = "-"
multiply = "*"
divide = "/"
exponent = "^"
modulus = "%"
inclusion = ( "in" / "IN" )
existentialQuantifier = ( "there_exists" / "THERE_EXISTS" )
universalQuantifier = ( "for_all" / "FOR_ALL" )
error = quot "error" quot
warning = quot "warning" quot
role = obra ( error / warning ) cbra
exprname = obra text cbra
constant = 1*digit [ decimal 1*digit ]
desc = pipe text pipe
nonDigit = %x41-5A / %x61-7A / "_ "
digit = %x30-39
end = ";"
opar = "("

```

```
cpar = ")"  
dollar = "$"  
obra = "["  
cbra = "]"  
ocur = "{"  
ccur = "}"  
quot = ""  
colon = ":"  
assig = ":@"  
decimal = "."  
comma = ","  
pipe = "|"  
list = "List"  
existspath = "existsPath"  
emptypath = "emptyPath"  
bool = "true" / "false"  
if = ( "if" / "IF" )  
then = ( "then" / "THEN" )  
else = ( "else" / "ELSE" )  
null = ( "null" / "NULL" )  
text = quot *( constant / nonDigit ) quot  
snomedheader = "snomed_ct::  
echeader = "snomed_ct_ec::  
round = "round"  
trim = "trim"  
count = "count"  
avg = "avg"  
min = "min"  
max = "max"  
sum = "sum"  
length = "length"
```

```

currentdate = "currentDate"
currenttime = "currentTime"
currentdatetime = "currentDateTime"
toupper = "toUpperCase"
tolower = "toLowerCase"
variable = dollar 1*nonDigit *( constant / nonDigit )
context = "context"
condition = "condition"
children = "children" opar cpar
path = 1*( [ divide ] ( ( 1*( constant / nonDigit ) ) / ( decimal decimal ) / children ) [
obra 1*( constant / nonDigit / minus / decimal ) cbra ] ) [ divide ]

```

### Informative comments on the EHRules ABNF syntax

```

expression = context colon ( path / divide ) end [ condition colon
contextBooleanExpression end ] 1*( ( [ exprname ] [ role ] booleanExpression end ) / (
variableDeclarationAssignment end ) / ( listVariableDeclarationAssignment end ) / ( [
exprname ] [ role ] ruleExpression ) )

```

Every expression or set of expressions is preceded by a context definition, i.e., a path referencing a section in the archetype, and optionally by a condition for that path. An expression is either a boolean expression, a variable declaration and assignment, a list variable declaration and assignment, or a rule expression. Both boolean expression and rule expression are optionally preceded by a name and role, i.e., error or warning. If no role is defined, error is assumed. Each expression is ended by a semicolon (;).

```

ruleExpression = if ruleAntecedent then ruleConsequent [ else ( ruleConsequent /
ruleExpression ) ]

```

A rule expression evaluates a condition (i.e., a rule antecedent) and depending on the resulting truth value (i.e., true or false), the result is one of two possible expressions

(i.e., rule consequents). Any rule is composed by at least one IF-THEN statement. The ELSE clause is optional and can evaluate a rule consequent or nest a rule expression, leading to nesting IF-THEN-ELSE statements.

***ruleAntecedent*** = *booleanExpression*

A rule antecedent is a boolean expression that is evaluated to true or false.

***ruleConsequent*** = ( *booleanExpression end* ) / ( *ocur 1\*( booleanExpression end ) ccur* )

A rule consequent is either a boolean expression or a set of boolean expressions enclosed in curly braces and separated by semicolons (;). A rule consequent is evaluated to true if the boolean expression or set of boolean expressions are evaluated to true.

***quantificationExpression*** = ( *existentialQuantifier / universalQuantifier* ) *variable inclusion ( variable / path ) colon booleanExpression*

A quantification expression is composed by either the standard operator from predicate logic THERE\_EXISTS (i.e., the existential quantifier) or by the FOR\_ALL standard logic operator (i.e., the universal quantifier). The first one evaluates whether there exists at least one element in a list (stored in a variable or referenced in the archetype by its path – *(variable / path)*) and represented by an iterator variable (*variable inclusion*) that satisfies the condition specified by a boolean expression that includes such iterator variable and that is preceded by a colon (:), which is usually read as “such that”. The second one (i.e., the FOR\_ALL universal quantifier) evaluates whether for all the elements in the list, the boolean expression is evaluated to true, i.e., the condition represented is satisfied for all the elements in the list. A quantification expression is evaluated to true or false.

***variableDeclarationAssignment*** = *variable colon ( ( duration assign ( durationExpression / null ) ) / ( dateTime assign ( dateTimeTypeExpression / null ) ) / ( date assign ( dateTypeExpression / null ) ) / ( time assign ( timeTypeExpression / null ) ) / ( integer*

*assign ( arithmeticExpression / null ) ) / ( real assign ( arithmeticExpression / null ) ) / ( boolean assign ( booleanExpression / null ) ) / ( string assign ( stringExpression / null ) ) / ( terminologyCode assign ( snomedConceptExpresion / null ) ) / ( snomedEc assign ( snomedEcExpression / null ) ) )*

A variable declaration and assignation is a statement that associates a data type with a variable name (i.e., declaration) and that assigns an expression compatible with such data type (i.e., assignation). The valid associations between data types and expressions are the following:

Data type	Expression
<i>integer</i>	<i>arithmeticExpression</i>
<i>real</i>	<i>arithmeticExpression</i>
<i>boolean</i>	<i>booleanExpression</i>
<i>string</i>	<i>stringExpression</i>
<i>date</i>	<i>dateTypeExpression</i>
<i>time</i>	<i>timeTypeExpression</i>
<i>dateTime</i>	<i>dateTimeTypeExpression</i>
<i>duration</i>	<i>durationExpression</i>
<i>terminologyCode</i>	<i>snomedConceptExpression</i>
<i>snomedEc</i>	<i>snomedEcExpression</i>

***listVariableDeclarationAssignation*** = *variable colon list less ( ( ( integer / real ) greater assign ( null / variable / ( ocur ( arithmeticExpression / null ) \*( comma ( arithmeticExpression / null ) ccur ) ) ) / ( boolean greater assign ( null / variable / ( ocur ( booleanExpression / null ) \*( comma ( booleanExpression / null ) ccur ) ) ) / ( string greater assign ( null / variable / ( ocur ( stringExpression / null ) \*( comma ( stringExpression / null ) ccur ) ) ) / ( terminologyCode greater assign ( null / variable / ( ocur ( snomedConceptExpression / null ) \*( comma ( snomedConceptExpresion / null )*

```

ccur )) / ( snomedEc greater assig ( null / variable / ( ocur ( snomedEcExpression /
null ) *( comma ( snomedEcExpression / null ) ) ccur )) ) / ( date greater assig ( null /
variable / ( ocur ( dateTypeExpression / null ) *( comma ( dateTypeExpression / null ) )
ccur )) ) / ( time greater assig ( null / variable / ( ocur ( timeTypeExpression / null ) *(
comma ( timeTypeExpression / null ) ) ccur )) ) / ( dateTime greater assig ( null /
variable / ( ocur ( dateTimeTypeExpression / null ) *( comma (
dateTimeTypeExpression / null ) ) ccur )) ) / ( duration greater assig ( null / variable / (
ocur ( durationExpression / null ) *( comma ( durationExpression / null ) ) ccur )) ) )

```

A list variable declaration and assignment is a statement that associates a data type with a list variable name (i.e., list declaration) and that assigns a set of expressions compatible with such data type (i.e., list assignment). The valid associations between data types and expressions are the same as in the *variableDeclarationAssignment* rule presented above. When more than one expression is assigned to a list variable, the expressions are enclosed in curly braces and separated by commas. The data type of the list is enclosed between the 'less than' and the 'greater than' symbols.

***booleanExpression*** = unaryBoolean \*subBooleanExpression

A boolean expression is composed by a unary boolean expression and optionally by one or more sub boolean expressions. A boolean expression is evaluated to true or false.

***subBooleanExpression*** = ( conjunction / disjunction ) unaryBoolean

A sub boolean expression combines a conjunction (i.e., AND) or disjunction (i.e., OR) operator with a unary boolean expression.

***unaryBoolean*** = ( not elementBoolean ) / elementBoolean

A unary boolean is either a boolean element or a boolean element preceded by a not operator (i.e., NOT). A unary boolean is evaluated to true or false.



**elementBoolean** = *existsPathFunction / emptyPathFunction / relationalExpression / INExpression / ( opar booleanExpression cpar ) / quantificationExpression / bool / path / variable*

A boolean element is either an exists path function, an empty path function, a relational expression, an inclusion expression, a boolean expression enclosed in parentheses, a quantification expression, a boolean value (i.e., true or false), a path, or a variable. A boolean element is evaluated to true or false.

**relationalExpression** = *( arithmeticExpression ( equal / notequal / less / lessequal / greater / greaterequal ) arithmeticExpression ) / ( stringExpression ( equal / notequal ) stringExpression ) / ( snomedConceptExpresion ( equal / notequal ) snomedConceptExpresion ) / relationalExpressionBoolean / ( dateTypeExpression ( equal / notequal / less / lessequal / greater / greaterequal ) dateTypeExpression ) / ( timeTypeExpression ( equal / notequal / less / lessequal / greater / greaterequal ) timeTypeExpression ) / ( dateTimeTypeExpression ( equal / notequal / less / lessequal / greater / greaterequal ) dateTimeTypeExpression ) / ( durationExpression ( equal / notequal ) durationExpression ) / ( opar relationalExpression cpar )*

A relational expression is either a comparison of two expressions of the same type by using a set of relational operators, a relational boolean expression, or a relational expression enclosed in parentheses. A relational expression is evaluated to true or false. The following comparisons are allowed by the syntax.

Left expression	Operators	Right expression
<i>arithmeticExpression</i>	<i>=, !=, &lt;, &lt;=, &gt;, &gt;=</i>	<i>arithmeticExpression</i>
<i>stringExpression</i>	<i>=, !=</i>	<i>stringExpression</i>
<i>snomedConceptExpresion</i>	<i>=, !=</i>	<i>snomedConceptExpresion</i>
<i>dateTypeExpression</i>	<i>=, !=, &lt;, &lt;=, &gt;, &gt;=</i>	<i>dateTypeExpression</i>
<i>timeTypeExpression</i>	<i>=, !=, &lt;, &lt;=, &gt;, &gt;=</i>	<i>timeTypeExpression</i>

<i>dateTimeTypeExpression</i>	<i>=, !=, &lt;, &lt;=, &gt;, &gt;=</i>	<i>dateTimeTypeExpression</i>
<i>durationExpression</i>	<i>=, !=</i>	<i>durationExpression</i>
<p><b><i>relationalExpressionBoolean</i></b> = ( ( [ not ] <i>INExpression</i> ) / <i>variable</i> / <i>path</i> / <i>bool</i> / <i>existsPathFunction</i> / <i>emptyPathFunction</i> ) / ( <i>opar</i> ( ( [ not ] <i>INExpression</i> ) / <i>variable</i> / <i>path</i> / <i>bool</i> / <i>existsPathFunction</i> / <i>emptyPathFunction</i> ) <i>cpar</i> ( <i>equal</i> / <i>notequal</i> ) ( ( [ not ] <i>INExpression</i> ) / <i>variable</i> / <i>path</i> / <i>bool</i> / <i>existsPathFunction</i> / <i>emptyPathFunction</i> ) ) / ( <i>opar</i> ( ( [ not ] <i>INExpression</i> ) / <i>variable</i> / <i>path</i> / <i>bool</i> / <i>existsPathFunction</i> / <i>emptyPathFunction</i> ) <i>cpar</i> )</p>		
<p>A relational boolean expression is a comparison between two boolean valued expressions by using equal and not equal relational operators (i.e., =, !=). Both left and right sides of the expression can be preceded by a not operator and are either an inclusion expression, a variable, a path, a boolean value (i.e., true or false), an exists path function, or an empty path function. A relational boolean expression is evaluated to true or false.</p>		
<p><b><i>arithmeticExpression</i></b> = <i>term</i> *<i>subArithmeticExpression</i></p>		
<p>An arithmetic expression contains a term optionally followed by one or more sub arithmetic expressions. An arithmetic expression is evaluated to a number.</p>		
<p><b><i>subArithmeticExpression</i></b> = ( <i>plus</i> / <i>minus</i> ) <i>term</i></p>		
<p>A sub arithmetic expression consists of either a plus or minus operator followed by a term.</p>		
<p><b><i>term</i></b> = <i>expTerm</i> *<i>subTerm</i></p>		
<p>A term is either an exponential term or an exponential term followed by one or more sub terms.</p>		
<p><b><i>expTerm</i></b> = <i>unary</i> *<i>expSubTerm</i></p>		

An exponential term is either a unary or a unary followed by one or more exponential sub terms.

***subTerm*** = ( *multiply / divide / modulus* ) *expTerm*

A sub term combines a multiply, a divide, or a modulus operator with an exponential term.

***expSubTerm*** = *exponent unary*

An exponential sub term represents an exponent operator with a unary.

***unary*** = ( ( *minus / plus* ) ( *element / roundFunction / variable / path / lengthFunction / countFunction / avgFunction / minFunction / maxFunction / sumFunction* ) ) / ( *element / roundFunction / variable / path / lengthFunction / countFunction / avgFunction / minFunction / maxFunction / sumFunction* ) )

A unary is either an element, a variable, a path, or any of the following functions: roundFunction, lengthFunction, countFunction, avgFunction, minFunction, maxFunction, or sumFunction, optionally preceded by either a minus or plus operator.

***element*** = *constant / ( opar arithmeticExpression cpar )*

An element is either a constant (i.e., either an integer or a real number) or an arithmetic expression enclosed in parentheses.

***roundFunction*** = *round opar arithmeticExpression comma constant cpar*

The round function consists of the literal “round” followed by two parameters separated by a comma and enclosed in parentheses, i.e., an arithmetic expression and a constant. The round function rounds a real number to a specified number of positions.

***countFunction*** = *count opar ( variable / path ) cpar*

The count function consists of the literal “count” followed by a parameter represented by a variable or path that reference a list and enclosed in parentheses. The count function returns the number of values in a specified list.

***avgFunction*** = *avg opar ( variable / path ) cpar*

The average function consists of the literal “avg” followed by a parameter represented by a variable or path that reference a list and enclosed in parentheses. The avg function returns the average of the values in a specified list, that is, the sum of the values divided by the number of values.

***minFunction*** = *min opar ( variable / path ) cpar*

The minimum function consists of the literal “min” followed by a parameter represented by a variable or path that reference a list and enclosed in parentheses. The minimum function returns the lowest value in a specified list.

***maxFunction*** = *max opar ( variable / path ) cpar*

The maximum function consists of the literal “max” followed by a parameter represented by a variable or path that reference a list and enclosed in parentheses. The maximum function returns the highest value in a specified list.

***sumFunction*** = *sum opar ( variable / path ) cpar*

The sum function consists of the literal “sum” followed by a parameter represented by a variable or path that reference a list and enclosed in parentheses. The sum function returns the sum of the values of a specified list.

***lengthFunction*** = *length opar ( text / variable / path ) cpar*

The length function consists of the literal “length” followed by a parameter represented by a text, variable or path that reference a string and enclosed in parentheses. The length function returns the number of characters in a string.

***currentDateFunction*** = *currentdate opar cpar*

The current date function consists of the literal “currentDate” followed by open and closed parentheses. The current date function returns the current date.

***currentTimeFunction*** = *currenttime opar cpar*

The current time function consists of the literal “currentTime” followed by open and closed parentheses. The current time function returns the current time.

***currentDateTimeFunction*** = *currentdatetime opar cpar*

The current date and time function consists of the literal “currentDateTime” followed by open and closed parentheses. The current date and time function returns the current date and time.

***existsPathFunction*** = *existspath opar path cpar*

The exists path function consists of the literal “existsPath” followed by a parameter represented by a path and enclosed in parentheses. The exists path function returns true if the specified path exists, otherwise it returns false.

***emptyPathFunction*** = *emptypath opar path cpar*

The empty path function consists of the literal “emptyPath” followed by a parameter represented by a path and enclosed in parentheses. The empty path function returns true if the specified path is empty, otherwise it returns false.

***INExpression*** = *snomedConceptExpresion inclusion snomedEcExpression*

An inclusion expression consists of a snomed concept expression followed by the inclusion operator (i.e., IN) and a snomed expression constraint expression. An inclusion expression returns true if the concept represented by the snomed concept expression is included in the subset represented by the snomed expression constraint expression, otherwise it returns false. It is used in all types of value set binding.

***dateTypeExpression*** = *variable / path / currentDateFunction / ( obra constant minus constant minus constant cbra ) / ( opar dateTypeExpression cpar )*

A date type expression is either a literal date, a variable, a path, the current date function or a date type expression enclosed in parentheses. A date type expression represents a date.

***timeTypeExpression*** = *variable / path / currentTimeFunction / ( obra constant colon constant colon constant cbra ) / ( opar timeTypeExpression cpar )*

A time type expression is either a literal time, a variable, a path, the current time function or a time type expression enclosed in parentheses. A time type expression represents a time.

***dateTimeTypeExpression*** = *variable / path / currentDateTimeFunction / ( obra constant minus constant minus constant constant colon constant colon constant cbra ) / ( opar dateTimeTypeExpression cpar )*

A date time type expression is either a literal date and time, a variable, a path, the current date and time function or a date time type expression enclosed in parentheses. A date time type expression represents a date and time.

***durationExpression*** = *variable / path / ( obra [ minus ] years months days hours minutes seconds cbra ) / ( opar durationExpression cpar )*

A duration expression is either a literal positive or negative duration, a variable, a path, or a duration expression enclosed in parentheses. A duration expression represents a positive or negative duration.

***stringExpression*** = *concatExpression* / *text* / *variable* / *path* / *toUpperCaseFunction* / *toLowerCaseFunction* / *trimFunction* / ( *opar stringExpression cpar* )

A string expression is either a concat expression, a text, a variable, a path, the upper case or lower case functions, the trim function, or a string expression enclosed in parentheses. A string expression represents a text.

***concatExpression*** = ( *text* / *variable* / *path* ) 1\**subConcatExpression*

A concat expression represents either a text or a variable or path representing a text, followed by one or more sub concat expressions.

***subConcatExpression*** = *plus* ( *text* / *variable* / *path* )

A sub concat expression contains the plus operator (i.e., +) followed by either a text or a variable or path representing a text.

***toUpperCaseFunction*** = *touppercase* *opar* ( *text* / *variable* / *path* ) *cpar*

The to upper case function consists of the literal "toUpperCase" followed by a parameter represented by a text, variable or path that reference a string and enclosed in parentheses. The to upper case function converts a string to upper case.

***toLowerCaseFunction*** = *tolowercase* *opar* ( *text* / *variable* / *path* ) *cpar*

The to lower case function consists of the literal "toLowerCase" followed by a parameter represented by a text, variable or path that reference a string and enclosed in parentheses. The to lower case function converts a string to lower case.

***trimFunction*** = *trim* *opar* ( *text* / *variable* / *path* ) *cpar*

The trim function consists of the literal “trim” followed by a parameter represented by a text, variable or path that reference a string and enclosed in parentheses. The trim function returns a specified string with leading and trailing whitespace removed.

***snomedConceptExpression*** = ( obra snomedheader constant [ desc ] cbra ) / variable / path

A snomed concept expression is either a literal, i.e., the snomed header followed by a number with an optional description and enclosed in brackets, a variable or a path representing a snomed concept expression. Note that the snomed concept represented by the snomed concept expression is later parsed using the ECL syntax supported in SNQuery.

***snomedEcExpression*** = ( obra echeader 1\*( constant / nonDigit ) cbra ) / variable / path

A snomed expression constraint expression is either a literal, i.e., the expression constraint header followed by any text and enclosed in brackets, a variable or a path representing a snomed expression constraint expression. Note that the expression constraint represented by the snomed expression constraint expression is later parsed using the ECL syntax supported in SNQuery.

***contextBooleanExpression*** = contextUnaryBoolean \*contextSubBooleanExpression

A context boolean expression is a context unary boolean optionally followed by one or more context sub boolean expressions. A context boolean expression is used to define the condition that a given context of a expression or set of expressions should meet, and it is optional.

***contextSubBooleanExpression*** = ( conjunction / disjunction ) contextUnaryBoolean

A context sub boolean expression is either a conjunction or disjunction, followed by a context unary boolean.



***contextUnaryBoolean*** = ( *not contextElementBoolean* ) / *contextElementBoolean*

A context unary boolean is a context element boolean optionally preceded by a not operator.

***contextElementBoolean*** = *contextRelationalExpression* / ( *opar contextBooleanExpression cpar* )

A context element boolean is either a context relational expression or a context element boolean enclosed in parentheses.

***contextRelationalExpression*** = ( *countFunction* / *avgFunction* / *minFunction* / *maxFunction* / *sumFunction* / *path* / *decimal* ) ( *equal* / *notequal* / *less* / *lessequal* / *greater* / *greaterequal* ) ( *text* / *constant* )

A context relational expression is a comparison between either a count, average, minimum, maximum or sum functions, a path or a decimal, and a text or constant, by using the usual set of relational operators (i.e., =, !=, <, <=, > and >=).

***comment*** = ( *"/" \*( digit / nonDigit )* ) / ( *"/\*" \*( digit / nonDigit ) "\*" /* )

A comment provides additional human-readable details about the expression. Comments begin with a forward slash directly followed by a star (i.e., /\*) and end with a star directly followed by a forward slash (i.e., \*/). Inline comments are preceded with a double slash (i.e., //).

***years*** = *constant "Y"*

A number followed by the literal "Y".

***months*** = *constant "M"*

A number followed by the literal "M".

<b><i>days</i></b> = constant "D"
A number followed by the literal "D".
<b><i>hours</i></b> = constant "h"
A number followed by the literal "h".
<b><i>minutes</i></b> = constant "m"
A number followed by the literal "m".
<b><i>seconds</i></b> = constant "s"
A number followed by the literal "s".
<b><i>boolean</i></b> = "Boolean"
The boolean data type, which is represented by the literal "Boolean".
<b><i>integer</i></b> = "Integer"
The integer data type, which is represented by the literal "Integer".
<b><i>real</i></b> = "Real"
The real data type, which is represented by the literal "Real".
<b><i>date</i></b> = "Date"
The date data type, which is represented by the literal "Date".
<b><i>time</i></b> = "Time"
The time data type, which is represented by the literal "Time".

<b><i>dateTime</i></b> = "Date_time"
The date and time data type, which is represented by the literal "Date_time".
<b><i>duration</i></b> = "Duration"
The duration data type, which is represented by the literal "Duration".
<b><i>string</i></b> = "String"
The string data type, which is represented by the literal "String".
<b><i>terminologyCode</i></b> = "Terminology_code"
The terminology code data type, which is represented by the literal "Terminology_code". In the current version of EHRules, only SNOMED CT concepts are supported.
<b><i>snomedEc</i></b> = "Snomed_ec"
The snomed expression constraint data type, which is represented by the literal "Snomed_ec". The snomed expression constraint data type is used in all variants of value set binding.
<b><i>not</i></b> = ( "not" / "NOT" )
The logical negation operator (case-insensitive).
<b><i>conjunction</i></b> = ( "and" / "AND" )
The logical conjunction operator (case-insensitive).
<b><i>disjunction</i></b> = ( "or" / "OR" )
The logical disjunction operator (case-insensitive).

<b><i>equal</i> = "="</b>
The relational equality operator.
<b><i>notequal</i> = "!=" / "&lt;&gt;"</b>
The relational inequality operator.
<b><i>less</i> = "&lt;"</b>
The relational 'less than' operator.
<b><i>lessequal</i> = "&lt;="</b>
The relational 'less than or equal' operator.
<b><i>greater</i> = "&gt;"</b>
The relational 'greater than' operator.
<b><i>greaterequal</i> = "&gt;="</b>
The relational 'greater than or equal' operator.
<b><i>plus</i> = "+"</b>
The addition arithmetic operator and also the concatenation string operator.
<b><i>minus</i> = "-"</b>
The subtraction arithmetic operator.
<b><i>multiply</i> = "*"</b>
The multiplication arithmetic operator.

<b><i>divide</i></b> = "/"
The division arithmetic operator.
<b><i>exponent</i></b> = "^"
The exponentiation arithmetic operator.
<b><i>modulus</i></b> = "%"
The modulus arithmetic operator.
<b><i>inclusion</i></b> = ( "in" / "IN" )
The inclusion operator to test whether a terminology concept is included in a specified subset of concepts. It is also the inclusion operator used in quantification expressions to associate an iterator variable with the list to be iterated. It is case-insensitive in both uses.
<b><i>existentialQuantifier</i></b> = ( "there_exists" / "THERE_EXISTS" )
The existential quantifier operator (case-insensitive).
<b><i>universalQuantifier</i></b> = ( "for_all" / "FOR_ALL" )
The universal quantifier operator (case-insensitive).
<b><i>error</i></b> = quot "error" quot
The literal string 'error' to be used as a role for an expression or set of expressions inside a context.
<b><i>warning</i></b> = quot "warning" quot

<p>The literal string 'warning' to be used as a role for an expression or set of expressions inside a context.</p>
<p><b>role</b> = obra ( error / warning ) cbra</p>
<p>The role to be assigned to an expression or set of expressions inside a context. If no role is specified, role 'error' is assumed. The role is specified by using either the literal 'error' or 'warning' enclosed in brackets.</p>
<p><b>exprname</b> = obra text cbra</p>
<p>The name for either one expression or one rule. Its specification is optional and it consists of a text enclosed in brackets.</p>
<p><b>constant</b> = 1*digit [ decimal 1*digit ]</p>
<p>A constant consists of one or more digits optionally followed by a decimal symbol and one or more digits.</p>
<p><b>desc</b> = pipe text pipe</p>
<p>A description to be used as term of a snomed concept. It consists of a text enclosed by a pair of pipe characters.</p>
<p><b>nonDigit</b> = %x41-5A / %x61-7A / "_ "</p>
<p>Characters from 'A' to 'Z' either in upper and lower case, and the underscore symbol.</p>
<p><b>digit</b> = %x30-39</p>
<p>Any digit 0 through 9.</p>
<p><b>end</b> = ";"</p>

Any expression in EHRules, including declaration and assignation of variables should en with a semicolon.
<b><i>opar = "("</i></b>
The open parenthesis.
<b><i>cpar = ")"</i></b>
The close parenthesis.
<b><i>dollar = "\$"</i></b>
The dollar symbol. Every variable name in EHRules begins with it.
<b><i>obra = "["</i></b>
The open bracket.
<b><i>cbra = "]"</i></b>
The close bracket.
<b><i>ocur = "{"</i></b>
The open curly brace.
<b><i>ccur = "}"</i></b>
The close curly brace.
<b><i>quot = ""</i></b>
The quote symbol. Every string in EHRules is enclosed by a pair of quotes.

<b><i>colon</i> = ":"</b>
The colon symbol. Every variable name is followed by a colon and a data type. It is also used to specify the boolean expression in a quantifier expression, where it is usually read as 'such as'.
<b><i>assign</i> = ":="</b>
The assignation operator. It used to assign a value to a variable of any data type.
<b><i>decimal</i> = "."</b>
The decimal point.
<b><i>comma</i> = ","</b>
The comma symbol. It used to separate either the elements in a list or the parameters in a function.
<b><i>pipe</i> = " "</b>
The pipe symbol, which is used to enclose the description of a snomed concept expression.
<b><i>list</i> = "List"</b>
The word 'List', which is used to define list variables.
<b><i>existspath</i> = "existsPath"</b>
The literal 'existsPath' used as name for the exists path function.
<b><i>emptypath</i> = "emptyPath"</b>
The literal 'emptyPath' used as name for the empty path function.



<b><i>bool</i></b> = "true" / "false"
A boolean value (i.e., true or false). Only allowed in lower case.
<b><i>if</i></b> = ( "if" / "IF" )
The 'if' clause that precedes the rule antecedent (case-insensitive).
<b><i>then</i></b> = ( "then" / "THEN" )
The 'then' clause that precedes the rule consequent when the antecedent is evaluated to true (case-insensitive).
<b><i>else</i></b> = ( "else" / "ELSE" )
The 'else' clause that precedes either the rule consequent or a nested rule expression when the antecedent is evaluated to false (case-insensitive).
<b><i>null</i></b> = ( "null" / "NULL" )
The 'null' value to be assigned as a value of any type of variable, including list variables.
<b><i>text</i></b> = quot *( constant / nonDigit ) quot
A text to be used as a string. The text is enclosed by a pair of quotes.
<b><i>snomedheader</i></b> = "snomed_ct::"
The snomed header used in snomed concept expressions.
<b><i>echeader</i></b> = "snomed_ct_ec::"
The expression constraint header used in snomed expression constraint expressions.
<b><i>round</i></b> = "round"

The word 'round' used in the round function.
<b><i>trim = "trim"</i></b>
The word 'trim' used in the trim function.
<b><i>count = "count"</i></b>
The word 'count' used in the count function.
<b><i>avg = "avg"</i></b>
The word 'avg' used in the average function.
<b><i>min = "min"</i></b>
The word 'min' used in the minimum function.
<b><i>max = "max"</i></b>
The word 'max' used in the maximum function.
<b><i>sum = "sum"</i></b>
The word 'sum' used in the sum function.
<b><i>length = "length"</i></b>
The word 'length' used in the length function.
<b><i>currentdate = "currentDate"</i></b>
The literal 'currentDate' used in the current date function.
<b><i>currenttime = "currentTime"</i></b>

The literal 'currentTime' used in the current time function.
<b><i>currentdatetime</i></b> = "currentDateTime"
The literal 'currentDateTime' used in the current date and time function.
<b><i>touppercase</i></b> = "toUpperCase"
The literal 'toUpperCase' used in the to upper case function.
<b><i>tolowercase</i></b> = "toLowerCase"
The literal 'toLowerCase' used in the to lower case function.
<b><i>variable</i></b> = dollar 1*nonDigit *( constant / nonDigit )
A variable name, which consists of a dollar symbol followed by one or more letters, and one or more digits or letters.
<b><i>context</i></b> = "context"
The word 'context', which is used to specify the context of either a expression or a set of expressions. Specifying a context is mandatory.
<b><i>condition</i></b> = "condition"
The word 'condition', which is used to set a condition to a context. Specifying a condition to a context is optional.
<b><i>children</i></b> = "children" opar cpar
The children function, which consists of the word 'children' followed by an open and closed parentheses. The use of this function is specific to context conditions and it returns the number of child elements of a given path.

**path** = 1\*( [ divide ] ( ( 1\*( constant / nonDigit ) ) / ( decimal decimal ) / children ) [ obra 1\*( constant / nonDigit / minus / decimal ) cbra ] ) [ divide ]

A path is a sequence of slashes symbols (i.e., /) followed by either numbers, letters or archetype terms enclosed in brackets (e.g., /items[at0016]). Two points in a row references the parent element (e.g., ../weight). To set the root of an archetype, the slash symbol is used as a whole path (i.e., /). Note that, although the path includes the children() function, it only makes sense to use it in context conditions.

## ANNEX 3 - FSIII CONDITION-INTERVENTIONS

---

ID	Condition	Interventions
1	Problems with personal care	<ul style="list-style-type: none"> <li>- Collaboration with networks</li> <li>- Rehabilitation</li> <li>- Guidance</li> <li>- Relocation and mobilization</li> <li>- Training</li> <li>- Support for ADL<sup>24</sup> activity</li> </ul>
2	Chronic pain	<ul style="list-style-type: none"> <li>- Nonpharmacological pain relief</li> <li>- Pain assessment</li> <li>- Training</li> <li>- Collaboration with networks</li> <li>- Medicine administration</li> <li>- Guidance</li> <li>- Intravenous medical treatment</li> <li>- Drug dispensing</li> </ul>
3	Sleeping problems	<ul style="list-style-type: none"> <li>- Medicine administration</li> <li>- Collaboration with networks</li> <li>- Guidance</li> <li>- Drug dispensing</li> <li>- Assessment of sleep pattern</li> </ul>
4	Problems with venous ulcer	<ul style="list-style-type: none"> <li>- Wound treatment</li> <li>- Training</li> <li>- Guidance</li> <li>- Medicine administration</li> <li>- Compression treatment</li> </ul>
5	Hearing problems	<ul style="list-style-type: none"> <li>- Collaboration with networks</li> <li>- Training</li> <li>- Guidance</li> <li>- Care when using personal aids</li> <li>- Special form of communication</li> </ul>

---

<sup>24</sup> Activities of Daily Living

6	Mental problems	<ul style="list-style-type: none"> <li>- Drug dispensing</li> <li>- Training</li> <li>- Medicine administration</li> <li>- Mental support</li> <li>- Guidance</li> <li>- Psychiatric care</li> </ul>
7	Acute pain	<ul style="list-style-type: none"> <li>- Pain assessment</li> <li>- Intravenous medical treatment</li> <li>- Nonpharmacological pain relief</li> <li>- Drug dispensing</li> <li>- Medicine administration</li> </ul>
8	Circadian rhythm problems	<ul style="list-style-type: none"> <li>- Collaboration with networks</li> <li>- Medicine administration</li> <li>- Assessment of sleep pattern</li> <li>- Guidance</li> </ul>
9	Problems with insight into treatment purposes	<ul style="list-style-type: none"> <li>- Training</li> <li>- Guidance</li> </ul>
10	Problems with diabetic ulcers	<ul style="list-style-type: none"> <li>- Medicine administration</li> <li>- Guidance</li> <li>- Training</li> <li>- Wound treatment</li> <li>- Compression treatment</li> </ul>
11	Problems with trauma wounds	<ul style="list-style-type: none"> <li>- Guidance</li> <li>- Training</li> <li>- Wound treatment</li> <li>- Medicine administration</li> </ul>
12	Problems with abuse	<ul style="list-style-type: none"> <li>- Guidance</li> <li>- Collaboration with networks</li> <li>- Medicine administration</li> <li>- Mental support</li> <li>- Drug dispensing</li> </ul>
13	Problems with food intake	<ul style="list-style-type: none"> <li>- Parenteral nutrition</li> <li>- Training</li> <li>- Surveys and measurement of values</li> <li>- Collaboration with networks</li> <li>- Tube feeding</li> </ul>

		<ul style="list-style-type: none"> <li>- Nutrition screening</li> <li>- Guidance</li> <li>- Medicine administration</li> <li>- Nutrition efforts</li> </ul>
14	Problems with underweight	<ul style="list-style-type: none"> <li>- Fluid per os</li> <li>- Guidance</li> <li>- Parenteral nutrition</li> <li>- Collaboration with networks</li> <li>- Tube feeding</li> <li>- Nutrition screening</li> <li>- Nutrition efforts</li> <li>- Training</li> </ul>
15	Problems with urination	<ul style="list-style-type: none"> <li>- Guidance</li> <li>- Catheter placement and care</li> <li>- Drug dispensing</li> <li>- Surveys and measurement of values</li> <li>- Medicine administration</li> </ul>
16	Problems with surgical wounds	<ul style="list-style-type: none"> <li>- Drainage care</li> <li>- Wound treatment</li> <li>- Guidance</li> <li>- Medicine administration</li> <li>- Training</li> </ul>
17	Problems with the sense of touch	<ul style="list-style-type: none"> <li>- Guidance</li> <li>- Relocation and mobilization</li> <li>- Drug dispensing</li> <li>- Training</li> </ul>
18	Problems with stool incontinence	<ul style="list-style-type: none"> <li>- Training</li> <li>- Incontinence treatment</li> <li>- Rehabilitation</li> <li>- Drug dispensing</li> <li>- Guidance</li> <li>- Medicine administration</li> <li>- Surveys and measurement of values</li> </ul>
19	Inappropriate weight change	<ul style="list-style-type: none"> <li>- Nutrition efforts</li> <li>- Guidance</li> <li>- Nutrition screening</li> <li>- Parenteral nutrition</li> </ul>

		<ul style="list-style-type: none"> <li>- Tube feeding</li> <li>- Training</li> </ul>
20	Problems with arterial ulcer	<ul style="list-style-type: none"> <li>- Wound treatment</li> <li>- Guidance</li> <li>- Training</li> <li>- Medicine administration</li> </ul>
21	Problems with communication	<ul style="list-style-type: none"> <li>- Guidance</li> <li>- Special form of communication</li> <li>- Collaboration with networks</li> </ul>
22	Problems with mobility and movement	<ul style="list-style-type: none"> <li>- Support for ADL activity</li> <li>- Training</li> <li>- Guidance</li> <li>- Collaboration with networks</li> <li>- Relocation and mobilization</li> <li>- Drug dispensing</li> <li>- Medicine administration</li> <li>- Treatment with orthopedic aids</li> <li>- Rehabilitation</li> </ul>
23	Periodic pain	<ul style="list-style-type: none"> <li>- Nonpharmacological pain relief</li> <li>- Intravenous medical treatment</li> <li>- Medicine administration</li> <li>- Drug dispensing</li> <li>- Pain assessment</li> </ul>
24	Problems with socializing	<ul style="list-style-type: none"> <li>- Collaboration with networks</li> <li>- Psychiatric care</li> <li>- Guidance</li> <li>- Mental support</li> </ul>
25	Circulatory problems	<ul style="list-style-type: none"> <li>- Medicine administration</li> <li>- Surveys and measurement of values</li> <li>- Training</li> <li>- Circulation treatment</li> <li>- Guidance</li> <li>- Compression treatment</li> </ul>
26	Problems with urinary incontinence	<ul style="list-style-type: none"> <li>- Incontinence treatment</li> <li>- Catheter placement and care</li> <li>- Medicine administration</li> </ul>



		<ul style="list-style-type: none"> <li>- Guidance</li> <li>- Rehabilitation</li> <li>- Drug dispensing</li> </ul>
27	Problems with disease insight	<ul style="list-style-type: none"> <li>- Guidance</li> <li>- Collaboration with networks</li> <li>- Training</li> </ul>
28	Problems with cancerous lesions	<ul style="list-style-type: none"> <li>- Guidance</li> <li>- Wound treatment</li> <li>- Medicine administration</li> <li>- Collaboration with networks</li> <li>- Training</li> <li>- Drainage care</li> </ul>
29	Memory problems	<ul style="list-style-type: none"> <li>- Medicine administration</li> <li>- Support for ADL activity</li> <li>- Psychiatric care</li> <li>- Mental support</li> <li>- Collaboration with networks</li> <li>- Drug dispensing</li> <li>- Rehabilitation</li> <li>- Guidance</li> </ul>
30	Problems with the sense of smell	<ul style="list-style-type: none"> <li>- Guidance</li> <li>- Support for ADL activity</li> <li>- Rehabilitation</li> </ul>
31	Problems with fluid intake	<ul style="list-style-type: none"> <li>- Intravenous fluid therapy</li> <li>- Nutrition screening</li> <li>- Training</li> <li>- Guidance</li> <li>- Nutrition efforts</li> <li>- Collaboration with networks</li> <li>- Tube feeding</li> </ul>
32	Stomach and intestinal problems	<ul style="list-style-type: none"> <li>- Treatment and care of gastrointestinal problem</li> <li>- Drug dispensing</li> <li>- Medicine administration</li> <li>- Guidance</li> <li>- Surveys and measurement of values</li> <li>- Ostomy care</li> </ul>

33	Problems with mixed wounds	<ul style="list-style-type: none"> <li>- Compression treatment</li> <li>- Wound treatment</li> <li>- Medicine administration</li> <li>- Training</li> <li>- Guidance</li> </ul>
34	Cognitive problems	<ul style="list-style-type: none"> <li>- Drug dispensing</li> <li>- Support for ADL activity</li> <li>- Mental support</li> <li>- Rehabilitation</li> <li>- Training</li> <li>- Guidance</li> <li>- Special form of communication</li> <li>- Medicine administration</li> </ul>
35	Problems with fluid from drains	<ul style="list-style-type: none"> <li>- Drainage care</li> </ul>
36	Problems with pressure ulcers	<ul style="list-style-type: none"> <li>- Medicine administration</li> <li>- Guidance</li> <li>- Treatment and care of skin problem</li> <li>- Relocation and mobilization</li> <li>- Training</li> <li>- Wound treatment</li> <li>- Collaboration with networks</li> </ul>
37	Problems with sexuality	<ul style="list-style-type: none"> <li>- Collaboration with networks</li> <li>- Guidance</li> <li>- Mental support</li> <li>- Medicine administration</li> <li>- Training</li> </ul>
38	Problems with the sense of sight	<ul style="list-style-type: none"> <li>- Training</li> <li>- Collaboration with networks</li> <li>- Guidance</li> <li>- Drug dispensing</li> <li>- Medicine administration</li> <li>- Rehabilitation</li> <li>- Support for ADL activity</li> <li>- Care when using personal aids</li> </ul>
39	Problems with the sense of taste	<ul style="list-style-type: none"> <li>- Guidance</li> </ul>

40	Respiratory problems	<ul style="list-style-type: none"> <li>- Secret suction</li> <li>- Respirator treatment</li> <li>- Surveys and measurement of values</li> <li>- Medicine administration</li> <li>- Collaboration with networks</li> <li>- Respiratory therapy</li> <li>- Mental support</li> <li>- Guidance</li> <li>- Tracheostomy care</li> <li>- Oxygen treatment</li> <li>- Training</li> </ul>
41	Problems with obesity	<ul style="list-style-type: none"> <li>- Nutrition efforts</li> <li>- Collaboration with networks</li> <li>- Training</li> <li>- Nutrition screening</li> <li>- Guidance</li> </ul>
42	Emotional problems	<ul style="list-style-type: none"> <li>- Guidance</li> <li>- Psychiatric care</li> <li>- Collaboration with networks</li> <li>- Mental support</li> <li>- Medicine administration</li> </ul>
43	Other skin and mucous membrane problems	<ul style="list-style-type: none"> <li>- Medicine administration</li> <li>- Collaboration with networks</li> <li>- Treatment and care of skin problem</li> <li>- Guidance</li> <li>- Training</li> </ul>
44	Problems with daily activities	<ul style="list-style-type: none"> <li>- Relocation and mobilization</li> <li>- Guidance</li> <li>- Training</li> <li>- Support for ADL activity</li> <li>- Collaboration with networks</li> <li>- Rehabilitation</li> </ul>



## ANNEX 4 - FROM EHRULES TO SCHEMATRON EXAMPLES

<i>Definition of the context for the expressions</i>	
EHRules	context: /;
SCH	<rule context="/">
<i>Declaration and assignment of Integer and Real variables</i>	
EHRules	\$varA: Integer := 6; \$varD: Integer := \$varB+1-1-1; \$total: Integer := path/to/value + \$varB - \$varB; \$varF: Integer := length('Hello world'); \$varH: Real := round(5.1234, 3) + 1 - 1;
SCH	<let name="varA" value="6"/> <let name="varD" value="(((xs:integer(\$varB) + 1) - 1) - 1)"/> <let name="total" value="((path/to/value + xs:integer(\$varB)) - xs:integer(\$varB))"/> <let name="varF" value="string-length('Hello world')"/> <let name="varH" value="((round(5.1234, 3) + 1) - 1)"/>
<i>Relational expressions involving Integer and Real</i>	
EHRules	\$varA = 6.0; \$total = \$varA+\$varB-\$varB; \$varA = (11 % 3)+4; \$varA >= path/to/value;
SCH	<assert test="(\$varA castable as xs:integer) and (xs:integer(\$varA) = 6.0)" fpi="expr" role="error"> Failed '(\$varA = 6.0)'/</assert>  <assert test="(\$total castable as xs:integer and \$varA castable as xs:integer and \$varB castable as xs:integer) and (xs:integer(\$total) = ((xs:integer(\$varA)

	<pre>+ xs:integer(\$varB)) - xs:integer(\$varB)))" fpi="expr" role="error"&gt; Failed '(\$total = ((\$varA + \$varB) - \$varB))'&lt;/assert&gt;  &lt;assert test="(\$varA castable as xs:integer) and (xs:integer(\$varA) = ((11 mod 3) + 4))" fpi="expr" role="error"&gt; Failed '(\$varA = ((11 % 3) + 4))'&lt;/assert&gt;  &lt;assert test="(\$varA castable as xs:integer) and (xs:integer(\$varA) &gt;= path/to/value)" fpi="expr" role="error"&gt; Failed '(\$varA &gt;= path/to/value)'&lt;/assert&gt;</pre>
<i>Declaration and assignment of Date variables</i>	
EHRules	<pre>\$varDA: Date := [2006-11-22]; \$varDB: Date := path/to/date; \$varDC: Date := currentDate();</pre>
SCH	<pre>&lt;let name="varDA" value="xs:date('2006-11-22')"/&gt; &lt;let name="varDB" value="path/to/date"/&gt; &lt;let name="varDC" value="current-date()"/&gt;</pre>
<i>Relational expressions involving Dates</i>	
EHRules	<pre>\$varDB = [2006-11-23]; path/to/date = [2006-11-23]; \$varDC &gt; \$varDA; currentDate() &gt; \$varDA; currentDate() &gt; \$varDA and not path/to/date &gt; currentDate();</pre>
SCH	<pre>&lt;assert test="(\$varDB castable as xs:date) and (xs:date(\$varDB) = xs:date('2006-11-23'))" fpi="expr" role="error"&gt; Failed '(\$varDB = [2006-11- 23])'&lt;/assert&gt;  &lt;assert test="(path/to/date = xs:date('2006-11-23'))" fpi="expr" role="error"&gt; Failed '(path/to/date = [2006-11-23])'&lt;/assert&gt;  &lt;assert test="(\$varDC castable as xs:date and \$varDA castable as xs:date) and (xs:date(\$varDC) &gt; xs:date(\$varDA))" fpi="expr" role="error"&gt; Failed '(\$varDC</pre>

	<pre>&gt; \$varDA)'&lt;/assert&gt;  &lt;assert test="(\$varDA castable as xs:date) and (current-date() &gt; xs:date(\$varDA))" fpi="expr" role="error"&gt; Failed '(currentDate() &gt; \$varDA)'&lt;/assert&gt;  &lt;assert test="(\$varDA castable as xs:date) and ((current-date() &gt; xs:date(\$varDA)) and not ((path/to/date &gt; current-date())))" fpi="expr" role="error"&gt; Failed '((currentDate() &gt; \$varDA) AND NOT (path/to/date &gt; currentDate()))'&lt;/assert&gt;</pre>
<i>Declaration and assignation of Time variables</i>	
EHRules	<pre>\$varTA: Time := [08:30:00]; \$varTB: Time := path/to/time; \$varTC: Time := currentTime();</pre>
SCH	<pre>&lt;let name="varTA" value="xs:time('08:30:00')"/&gt; &lt;let name="varTB" value="path/to/time"/&gt; &lt;let name="varTC" value="current-time()"/&gt;</pre>
<i>Relational expressions involving Times</i>	
EHRules	<pre>\$varTB=[08:30:00]; path/to/time = [08:30:00]; \$varTC != \$varTA; currentTime() != \$varTA; currentTime() != \$varTA and not path/to/time = currentTime();</pre>
SCH	<pre>&lt;assert test="(\$varTB castable as xs:time) and (xs:time(\$varTB) = xs:time('08:30:00'))" fpi="expr" role="error"&gt; Failed '(\$varTB = [08:30:00])'&lt;/assert&gt;  &lt;assert test="(path/to/time = xs:time('08:30:00'))" fpi="expr" role="error"&gt; Failed '(path/to/time = [08:30:00])'&lt;/assert&gt;  &lt;assert test="(\$varTC castable as xs:time and \$varTA castable as xs:time) and (xs:time(\$varTC) != xs:time(\$varTA))" fpi="expr" role="error"&gt; Failed '(\$varTC</pre>

	<pre>!= \$varTA)'&lt;/assert&gt;  &lt;assert test="(\$varTA castable as xs:time) and (current-time() != xs:time(\$varTA))" fpi="expr" role="error"&gt; Failed '(currentTime() != \$varTA)'&lt;/assert&gt;  &lt;assert test="(\$varTA castable as xs:time) and ((current-time() != xs:time(\$varTA)) and not ((path/to/time = current-time())))" fpi="expr" role="error"&gt; Failed '((currentTime() != \$varTA) AND NOT (path/to/time = currentTime()))'&lt;/assert&gt;</pre>
<i>Declaration and assignation of DateTime variables</i>	
EHRules	<pre>\$varDTA: Date_time := ([2006-11-22 08:30:00]); \$varDTB: Date_time := path/to/date_time; \$varDTC: Date_time := currentDateTime();</pre>
SCH	<pre>&lt;let name="varDTA" value="xs:dateTime('2006-11-22T08:30:00')"/&gt; &lt;let name="varDTB" value="path/to/date_time"/&gt; &lt;let name="varDTC" value="current-dateTime()"/&gt;</pre>
<i>Relational expressions involving DateTimes</i>	
EHRules	<pre>\$varDTB=[2006-11-22 18:57:01]; path/to/date_time &gt; [2006-11-22 08:30:00]; \$varDTC != \$varDTA; currentDateTime() &gt; \$varDTA; currentDateTime() &gt; \$varDTA and not path/to/date_time &gt; currentDateTime();</pre>
SCH	<pre>&lt;assert test="(\$varDTB castable as xs:dateTime) and (xs:dateTime(\$varDTB) = xs:dateTime('2006-11-22T18:57:01'))" fpi="expr" role="error"&gt; Failed '(\$varDTB = [2006-11-22 18:57:01])'&lt;/assert&gt;  &lt;assert test="(path/to/date_time &gt; xs:dateTime('2006-11-22T08:30:00'))" fpi="expr" role="error"&gt; Failed '(path/to/date_time &gt; [2006-11-22</pre>



	<pre>08:30:00)]&lt;/assert&gt;  &lt;assert test="(\$varDTC castable as xs:dateTime and \$varDTA castable as xs:dateTime) and (xs:dateTime(\$varDTC) != xs:dateTime(\$varDTA))" fpi="expr" role="error"&gt; Failed '(\$varDTC != \$varDTA)'&lt;/assert&gt;  &lt;assert test="(\$varDTA castable as xs:dateTime) and (current-dateTime() &gt; xs:dateTime(\$varDTA))" fpi="expr" role="error"&gt; Failed '(currentDateTime() &gt; \$varDTA)'&lt;/assert&gt;  &lt;assert test="(\$varDTA castable as xs:dateTime) and ((current-dateTime() &gt; xs:dateTime(\$varDTA)) and not ((path/to/date_time &gt; current-dateTime())))" fpi="expr" role="error"&gt; Failed '((currentDateTime() &gt; \$varDTA) AND NOT (path/to/date_time &gt; currentDateTime()))'&lt;/assert&gt;</pre>
<i>Declaration and assignment of Duration variables</i>	
EHRules	<pre>\$varDuF: Duration := [10Y 10M 10D 10h 10m 10s]; \$varDuG: Duration := [600s]; \$varDuH: Duration := [1Y]; \$varDul: Duration := [12M];</pre>
SCH	<pre>&lt;let name="varDuF" value="xs:duration('P10Y10M10DT10H10M10S')"/&gt; &lt;let name="varDuG" value="xs:duration('PT600S')"/&gt; &lt;let name="varDuH" value="xs:duration('P1Y')"/&gt; &lt;let name="varDul" value="xs:duration('P12M')"/&gt;</pre>
<i>Relational expressions involving Durations</i>	
EHRules	<pre>\$varDuH = \$varDul; [1Y 0s] = [12M]; [-0h 10m 0s] = [-600s];</pre>
SCH	<pre>&lt;assert test="(\$varDuH castable as xs:duration and \$varDul castable as xs:duration) and (xs:duration(\$varDuH) = xs:duration(\$varDul))" fpi="expr" role="error"&gt; Failed '(\$varDuH = \$varDul)'&lt;/assert&gt;</pre>

	<pre>&lt;assert test="(xs:duration('P1YTOS') = xs:duration('P12M'))" fpi="expr" role="error"&gt; Failed '([1Y 0s] = [12M])'&lt;/assert&gt;  &lt;assert test="(xs:duration('-PTOH10MOS') = xs:duration('-PT600S'))" fpi="expr" role="error"&gt; Failed '([-0h 10m 0s] = [-600s])'&lt;/assert&gt;</pre>
<i>Declaration and assignment of String variables</i>	
EHRules	<pre>\$varA: String := ' Hello '; \$varB: String := trim(\$varA); \$varD: String := \$varA + \$varB; \$total: String := path/to/string; \$varF: String := toUpperCase(\$varD);</pre>
SCH	<pre>&lt;let name="varA" value=" Hello, " /&gt; &lt;let name="varB" value="normalize-space(xs:string(\$varA))" /&gt; &lt;let name="varD" value="concat(xs:string(\$varA), xs:string(\$varB))" /&gt; &lt;let name="total" value="path/to/string" /&gt; &lt;let name="varF" value="upper-case(xs:string(\$varD))" /&gt;</pre>
<i>Relational expressions involving Strings</i>	
EHRules	<pre>\$varA = ' Hello, '; \$varB = 'Hello, '; \$varD = ' Hello, Hello, '; \$total = 'Total'; \$varF = ' HELLO, HELLO, ';</pre>
SCH	<pre>&lt;assert test="(\$varA castable as xs:string) and (xs:string(\$varA) = ' Hello, ')" fpi="expr" role="error"&gt; Failed '(\$varA = ' Hello, ')&lt;/assert&gt;  &lt;assert test="(\$varB castable as xs:string) and (xs:string(\$varB) = 'Hello, ')" fpi="expr" role="error"&gt; Failed '(\$varB = 'Hello, ')&lt;/assert&gt;  &lt;assert test="(\$varD castable as xs:string) and (xs:string(\$varD) = ' Hello, Hello, ')" fpi="expr" role="error"&gt; Failed '(\$varD = ' Hello, Hello, ')&lt;/assert&gt;</pre>

	<pre>&lt;assert test="(\$total castable as xs:string) and (xs:string(\$total) = 'Total')" fpi="expr" role="error"&gt; Failed '(\$total = 'Total')'&lt;/assert&gt;  &lt;assert test="(\$varF castable as xs:string) and (xs:string(\$varF) = ' HELLO, HELLO,')" fpi="expr" role="error"&gt; Failed '(\$varF = ' HELLO, HELLO,')'&lt;/assert&gt;</pre>
<i>Declaration and assignation of Boolean variables</i>	
EHRules	<pre>\$varA: Boolean := true; \$varC: Boolean := path/to/boolean; \$varE: Boolean := \$varA and \$varB; \$varK: Boolean := there_exists \$i IN path/to/booleans : \$i&gt;5; \$varN: Boolean := for_all \$i IN path/to/booleans : \$i&gt;=-1 and \$i&lt;=6;</pre>
SCH	<pre>&lt;let name="varA" value="true()"/&gt; &lt;let name="varC" value="path/to/boolean"/&gt; &lt;let name="varE" value="(xs:boolean(\$varA) and xs:boolean(\$varB))"/&gt; &lt;let name="varK" value="(some \$i in path/to/booleans satisfies ((\$i) &gt; 5))"/&gt; &lt;let name="varN" value="(every \$i in path/to/booleans satisfies (((\$i) &gt;= (0 - 1)) and ((\$i) &lt;= 6)))/&gt;</pre>
<i>Relational expressions involving Booleans</i>	
EHRules	<pre>\$varA = true; \$varC = false; \$varE = false; \$varK = true; \$varN = true;</pre>
SCH	<pre>&lt;assert test="(\$varA castable as xs:boolean) and (xs:boolean(\$varA) = true())" fpi="expr" role="error"&gt; Failed '(\$varA = true)'&lt;/assert&gt;  &lt;assert test="(\$varC castable as xs:boolean) and (xs:boolean(\$varC) = false())" fpi="expr" role="error"&gt; Failed '(\$varC = false)'&lt;/assert&gt;</pre>

	<pre>&lt;assert test="(\$varE castable as xs:boolean) and (xs:boolean(\$varE) = false())"   fpi="expr" role="error"&gt; Failed '(\$varE = false)'&lt;/assert&gt;  &lt;assert test="(\$varK castable as xs:boolean) and (xs:boolean(\$varK) = true())"   fpi="expr" role="error"&gt; Failed '(\$varK = true)'&lt;/assert&gt;  &lt;assert test="(\$varN castable as xs:boolean) and (xs:boolean(\$varN) =   true())" fpi="expr" role="error"&gt; Failed '(\$varN = true)'&lt;/assert&gt;</pre>
<i>Declaration and assignation of List variables</i>	
EHRules	<pre>\$varQ: List&lt;Integer&gt;:={10,20,\$varO,\$varP+10,50}; \$varR: List&lt;String&gt;:={'red', 'green', 'blue'}; \$varS: List&lt;Boolean&gt;:={true, true, false};</pre>
SCH	<pre>&lt;let name="varQ" value="(((10 , 20) , xs:integer(\$varO)) , (xs:integer(\$varP + 10)) , 50)"/&gt; &lt;let name="varR" value="('red' , 'green') , 'blue'"/&gt; &lt;let name="varS" value="((true() , true()) , false())"/&gt;</pre>
Simple value set binding	
EHRules	<pre>\$varIN_01: Boolean := [snomed_ct::168539009] IN [snomed_ct_ec::&lt; 138875005  snomed root ];</pre>
SCH	<pre>&lt;let name="varIN_01" value="document(iri-to- uri(concat('https://snquery.veratech.es/VTTermService/ws/ValueSet/\$valida te-code?code=', '168539009', '&amp;url=http://snomed.info/ecl/', '&amp;lt; 138875005  snomed root '))//fhir:valueBoolean/@value='true'"/&gt;</pre>
Conditional value set binding	
EHRules	<pre>\$varEc: Snomed_ec := [snomed_ct_ec::&lt; 138875005  snomed root ]; \$varABU: Integer := 3; IF \$varABU=3 THEN path/to/concept IN \$varEc;</pre>
SCH	<pre>&lt;let name="varEc" value="&amp;lt; 138875005  snomed root "/&gt; &lt;let name="varABU" value="3"/&gt;</pre>

	<pre>&lt;assert test="(\$varABU castable as xs:integer and \$varEc castable as xs:string) and (if ((xs:integer(\$varABU) = 3)) then document(iri-to- uri(concat('https://snquery.veratech.es/VTTermService/ws/ValueSet/\$valida te-code?code=', path/to/concept, '&amp;url=http://snomed.info/ecl/', xs:string(\$varEc))))//fhir:valueBoolean/@value='true' else true()" fpi="expr" role="error"&gt; Failed 'IF (\$varABU = 3) THEN path/to/concept IN \$varEc;'&lt;/assert&gt;</pre>
Dependency value set binding	
EHRules	<pre>\$varConcept: Terminology_code := path/to/concept; [snomed_ct::80146002] IN [snomed_ct_ec::&lt; [[ \$varConcept ]]]; [snomed_ct::80146002] IN [snomed_ct_ec::&lt; [[ path/to/concept ]]]; </pre>
SCH	<pre>&lt;let name="varConcept" value="path/to/concept"/&gt;  &lt;assert test="document(iri-to- uri(concat('https://snquery.veratech.es/VTTermService/ws/ValueSet/\$valida te-code?code=', '80146002', '&amp;url=http://snomed.info/ecl/', '&amp;lt;' , \$varConcept)))/fhir:valueBoolean/@value='true'" fpi="expr" role="error"&gt; Failed '[snomed_ct::80146002] IN [snomed_ct_ec::&amp;lt; [[ \$varConcept ]]]'&lt;/assert&gt;  &lt;assert test="document(iri-to- uri(concat('https://snquery.veratech.es/VTTermService/ws/ValueSet/\$valida te-code?code=', '80146002', '&amp;url=http://snomed.info/ecl/', '&amp;lt;' , path/to/concept)))/fhir:valueBoolean/@value='true'" fpi="expr" role="error"&gt; Failed '[snomed_ct::80146002] IN [snomed_ct_ec::&amp;lt; [[ path/to/concept ]]]'&lt;/assert&gt;</pre>
Conditional + dependency value set binding	
EHRules	<pre>\$varPostcoordination: Terminology_code := path/to/postcoordinated; \$varConcept: Terminology_code := path/to/concept; \$varABU: Integer := 3;</pre>

	<pre>IF \$varABU=2 THEN [snomed_ct::80146002] IN [snomed_ct_ec::&lt;[[ \$varPostcoordination]]]; ELSE [snomed_ct::80146002] IN [snomed_ct_ec::&lt; [[ \$varConcept]]];</pre>
SCH	<pre>&lt;let name="varConcept" value="path/to/concept"/&gt; &lt;let name="varPostcoordination" value="path/to/postcoordinated"/&gt; &lt;let name="varABU" value="3"/&gt;  &lt;assert test="(\$varABU castable as xs:integer) and (if ((xs:integer(\$varABU) = 2)) then document(iri-to- uri(concat('https://snquery.veratech.es/VTTermService/ws/ValueSet/\$valida te-code?code=', '80146002', '&amp;url=http://snomed.info/ecl/', '&amp;lt;' , \$varPostcoordination))))//fhir:valueBoolean/@value='true' else document(iri- to- uri(concat('https://snquery.veratech.es/VTTermService/ws/ValueSet/\$valida te-code?code=', '80146002', '&amp;url=http://snomed.info/ecl/', '&amp;lt;' , \$varConcept))))//fhir:valueBoolean/@value='true')" fpi="expr" role="error"&gt; Failed 'IF (\$varABU = 2) THEN [snomed_ct::80146002] IN [snomed_ct_ec::&amp;lt; [[ \$varPostcoordination]]]; ELSE [snomed_ct::80146002] IN [snomed_ct_ec::&amp;lt; [[ \$varConcept]]]'&lt;/assert&gt;</pre>
<i>Definition of contexts with conditions</i>	
EHRules	<pre>context: this/is/a/dummy/path; condition: count(children()) &gt; 3;  context: this/is/a/dummy/path; condition: height &gt; 185;  context: this/is/a/dummy/path; condition: . = 80;</pre>
SCH	<pre>&lt;rule context="this/is/a/dummy/path[(count(child:*) &gt; 3)]"&gt; &lt;rule context="this/is/a/dummy/path[(height &gt; 185)]"&gt; &lt;rule context="this/is/a/dummy/path[(. = 80)]"&gt;</pre>