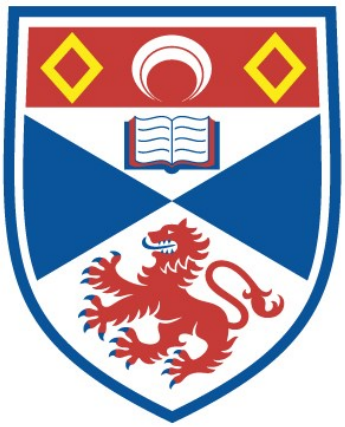
A clinical decision support system for detecting and mitigating potentially inappropriate medications

Guilherme Alfredo Redeker

A thesis submitted for the degree of PhD
at the
University of St Andrews



2024

ABSTRACT

Background: Medication errors are a leading cause of preventable harm to patients. In older adults, the impact of ageing on the therapeutic effectiveness and safety of drugs is a significant concern, especially for those over 65. Consequently, certain medications called *Potentially Inappropriate Medications* (PIMs) can be dangerous in the elderly and should be avoided. Tackling PIMs by health professionals and patients can be time-consuming and error-prone, as the criteria underlying the definition of PIMs are complex and subject to frequent updates. Moreover, the criteria are not available in a representation that health systems can interpret and reason with directly.

Objectives: This thesis aims to demonstrate the feasibility of using an ontology/rule-based approach in a clinical knowledge base to identify potentially inappropriate medication(PIM). In addition, how constraint solvers can be used effectively to suggest alternative medications and administration schedules to solve or minimise PIM undesirable side effects.

Methodology: To address these objectives, we propose a novel integrated approach using formal rules to represent the PIMs criteria and inference engines to perform the reasoning presented in the context of a Clinical Decision Support System (CDSS). The approach aims to detect, solve, or minimise undesirable side-effects of PIMs through an ontology (knowledge base) and inference engines incorporating multiple reasoning approaches.

Contributions: The main contribution lies in the framework to formalise PIMs, including the steps required to define guideline requisites to create inference rules to detect and propose alternative drugs to inappropriate medications. No formalisation of the selected guideline (Beers Criteria) can be found in the literature, and hence, this thesis provides a novel ontology for it. Moreover, our process of minimising undesirable side effects offers a novel approach that enhances and optimises the drug rescheduling process, providing a more accurate way to minimise the effect of drug interactions in clinical practice.

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Juliana Bowles, whose expertise, encouragement, and unwavering commitment have been instrumental in shaping my research trajectory. Your guidance has been transformative, and I am truly grateful for the opportunities to learn under your supervision.

I would also like to thank Martina for her support and invaluable insights, which have been instrumental in shaping the pharmacist aspects of this work. Many thanks also to Dr Joan Espasa Arxer and Dr Stephen Brown for their patience and guidance during the viva. Their invaluable feedback for this thesis is very much appreciated.

I am deeply grateful to my friends, especially Jordina, Diana, Marco, Thais and Ricardo, who have enriched my academic experience with insightful discussions and collaborative efforts and for providing me with moments of respite and encouragement, reminding me of the importance of balance and laughter amidst the academic challenges.

Furthermore, I would like to express my gratitude to my family for their belief in me and continuous support. Your incentive has fueled my determination to pursue this academic endeavour.

A special thank you goes to my wife, Alice, and our children, João Victor and Lucas. Your unwavering support has made the challenges of this journey more manageable, and I am deeply grateful for your presence by my side. João Victor and Lucas, you have brought joy and inspiration into my life every day. Finally, I want also to express my gratitude to my cats, Charlie and Mia, for their companionship and support, without which I would not be where I am today.

Declaration

Candidate's declaration

I, Guilherme Alfredo Redeker, do hereby certify that this thesis, submitted for the degree of PhD, which is approximately 41,000 words in length, has been written by me, and that it is the record of work carried out by me, or principally by myself in collaboration with others as acknowledged, and that it has not been submitted in any previous application for any degree. I confirm that any appendices included in my thesis contain only material permitted by the 'Assessment of Postgraduate Research Students' policy.

I was admitted as a research student at the University of St Andrews in May 2019.

I received funding from an organisation or institution and have acknowledged the funder(s) in the full text of my thesis.

Date 18 August 2023 Signature of candidate

Supervisor's declaration

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of PhD in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree. I confirm that any appendices included in the thesis contain only material permitted by the 'Assessment of Postgraduate Research Students' policy.

Date 18 August 2023 Signature of supervisor

Permission for Publication

In submitting this thesis to the University of St Andrews we understand that we are giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. We also understand, unless exempt by an award of an embargo as requested below, that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that this thesis will be electronically accessible for personal or research use and that the library has the right to migrate this thesis into new electronic forms as required to ensure continued access to the thesis.

I, Guilherme Alfredo Redeker, confirm that my thesis does not contain any third-party material that requires copyright clearance.

The following is an agreed request by candidate and supervisor regarding the publication of this thesis:

Printed copy

No embargo on print copy.

Electronic copy

No embargo on electronic copy.

Date 18 August 2023 Signature of candidate

Date 18 August 2023 Signature of supervisor

Underpinning Research Data or Digital Outputs

Candidate's declaration

I, Guilherme Alfredo Redeker, hereby certify that no requirements to deposit original research data or digital outputs apply to this thesis and that, where appropriate, secondary data used have been referenced in the full text of my thesis.

Date 18 August 2023 Signature of candidate

Data Management

Collection and Transfer

The data was extracted in an ANONYMISED form by the Brazilian hospital and made available to the researcher in a directory informed by the hospital. Therefore, no private identifiable patient data is shown. Once available by the hospital, the data was downloaded directly to the University network (Central file space).

Storage, Backup and Access

The extracted data is stored at the University Research Information System Pure. Computers used to collate data have limited access measures via usernames and passwords. The data access is granted to the researcher and supervisor. In addition, authorised representatives from the University of St. Andrews, host institution and the regulatory authorities to permit project-related monitoring, audits and inspections, where this is possible within the policies and regulations of the data source, are also granted.The study complies with all applicable medical confidentiality and data protection principles and laws (the Data Protection Act 1998 and GDPR) concerning collecting, storing, processing, and disclosing personal data.

Sharing and Publication

The data is documented in this PhD thesis, reports to the university, conference presentations and academic publications always in an ANONYMISED form, which means that no private identifiable patient data is shown.

Retention and Destruction

The data will be available indefinitely

FUNDING

The research reported in this thesis has been supported by the University of St Andrews (School of Computer Science) and partly supported by the SCCH competence center INTEGRATE (FFG grant no. 892418).

PUBLICATIONS

Some of the work described in this dissertation has been published or is currently under review for publication. The following list gives an overview of the papers that have been have been published or are currently in print.

Guilherme Redeker and Juliana Bowles. An Ontology-based Approach for Detecting and Classifying Inappropriate Prescribing. To appear in *the 7th International Joint Conference on Rules and Reasoning (RuleML+RR), Oslo, Norway, September 2023.* InPrint

Guilherme Redeker and Juliana Bowles. Tackling polypharmacy: A multi-source decision support system. In *Digital Personalized Health and Medicine*, pages 688–692. IOS Press, 2020

CONTENTS

List of Figures

List of Tables

LISTINGS

Acronyms

ABox Assertion component

AGS American Geriatric Society

B.PIM Beers Criteria Potentially Inappropriate Medications

CDSS Clinical Decision Support System

CMAX Maximum medication concentrations in serum

DDDS Potentially Inappropriate Medication Due to Drug-Disease or Drug-Syndrome Interactions That May Exacerbate the Disease or Syndrome

DDI Potentially Clinically Important Drug-Drug Interactions

EMR Eletronic Medical Record

FOL First order logic

PIMs Potentially Inappropriate Medications

SMT Satisfiability Modulo Theories

SPARQL Simple Protocol and RDF Query Language

SWRL Semantic Web Rule Language

TBox Terminology component

TMAX Time to peak drug concentration in serum

UWC Drugs To Be Used With Caution

VLKF Medications That Should Be Avoided or Have Their Dosage Reduced With Varying Levels of Kidney Function

Chapter One

Introduction

This thesis investigates how a variety of computer science-based techniques can be extended and adapted in the context of medical systems in order to tackle real clinical needs in novel and efficient ways. We explore different aspects of knowledge representation, including the use of ontologies and reasoning with constraint solvers, and how they can best be integrated in order to provide a valuable setting for medical systems.

The novel contribution this thesis makes is to demonstrate how integrated inference engines combining formal knowledge representation and reasoning can be used to obtain a holistic approach — a comprehensive methodology. This holistic approach takes into account all pertinent elements and their interrelations, enhancing the resolution of practical and complex problems within the healthcare domain.

1.1 Motivation

There are many medical problems that deserve our further attention and require a better understanding of what can be done and how computer science can help. Efforts to develop effective solutions that can support healthcare professionals in their decision-making processes across various healthcare fields are currently on the rise [162, 138, 119, 62, 127, 66, 48]. These fields typically include forming a diagnosis, detecting potential drug interactions, suggesting alternate medications, and selecting appropriate therapies and treatments. One of such problem concerns the use, and often misuse, of medications and their complex consequences.

Specifically, supporting the decision process associated to prescribing the most appropriate drugs and administering them safely (e.g., at correct times, in the right way and dosage) is still a challenge that lacks satisfactory solutions in practice often relying on staff experience and competence. There are studies [104, 167, 24, 158, 107, 29] that make partial contributions for particular aspects of the problem, for example, checking the interaction between drugs or diseases. However, an integrated solution, which supports several kinds of interactions, guides healthcare professionals in their choice of alternative drugs when an interaction is detected or when it is detected but cannot be avoided (usually when drugs cannot be removed as their therapeutic benefit outweighs their risks) and offers ways to minimise interaction effects, does not exist at present.

According to the WHO[160], medication errors incur a global cost of approximately US$42 billion annually. In England, more than 237 million medication errors are made annually, the avoidable consequences of which cost the NHS upwards of £98 million and more than 1700 lives yearly, according to national estimates[17]. In Brazil, it is estimated that approximately 5% to 6% of hospitalisations are related to medication use, particularly affecting the elderly, according to the Brazilian government[5].

Let us consider situations where computer-based systems could have a real impact if used to support medical decision-making. It is estimated that 1% of hospitalisations in the general population and 2-5% of hospitalisations in the elderly are caused by drug interaction problems [86]. A system that could identify and solve these interactions could reduce these estimated percentages and help reduce the burden faced by secondary healthcare providers. Furthermore, during hospitalisation, patients remain exposed to medication problems which may in fact be worse when their medical history is not known or considered. For example, prescribing drugs that interact with other medications or administering drugs incorrectly and at the wrong time are common types of problems [134, 33]. Furthermore, to be able to consider all the elements and consequences medications can have on patients at all times is not an easy task, and it is not surprising that some of these may be missed by physicians [90]. Every year new drugs are discovered; for example, between 2019 and 2021, the Food and Drug Administration (FDA)(2022) approved 151 new drugs. Hence, healthcare professionals may be unaware of the impact of prescribing new drugs, including possible conflicts with current guidelines, drug interactions and adverse reactions.

A particular challenge that could be addressed by computer-based systems is the identification of different types of drugs which constitute so-called *Potentially Inappropriate Medications* (PIMs). PIMs occur due to variations in the absorption, distribution, metabolism, excretion and physiological effects of the drug [93]. PIMs correspond to prescriptions that should be avoided for older adults in most situations and for all under certain conditions where the risks outweigh the benefits [117]. The prescription of PIMs is, however, common in the general older population and aggravated due to their prevalence of comorbidities. In Brazil, which is the low and middle-income (LMI) country used throughout this thesis for comparison, the prescription of PIMs ranges from 20,3% to 54,6% [42].

The abundance of information in a digital age hints at new potential ways to explore how computer science can perhaps help improve and tackle this situation. While machine learning has gained attention, it is often perceived as a black box by doctors [54], leaving them unable to track, and hence trust, the origin of the information they rely on. In contrast, formal methods offer distinct advantages as they cater to all necessary requirements, ranging from knowledge representation to traceability.

Computer-based systems should be used to promote medication safety by facilitating evidence-informed medication use, reducing the incidence of harmful medication errors, and improving the efficiency of healthcare systems in practice [74]. Clinical Decision Support System (CDSS) seems to outperform conventional tools as an information resource, offering easy access and intuitive handling in the daily routine, particularly benefiting less-experienced medical practitioners [104]. In the Brazilian hospital context, a CDSS for managing prescriptions could change considerably current practice, reducing elderly patients' exposure to PIMs and hospitalisations due to adverse drug reactions. In addition, solutions could be used to facilitate and reduce the load on physicians, and more generally healthcare providers, helping them by taking into account factors such as drug interactions, dose, age, gender, ethnicity, etc, that may influence treatment.

Drawing from my 15+ years of professional experience within the Brazilian hospital environment, it has become evident that prescription errors in Brazil are often related to poor professional practice, incomplete or inexisting guidelines, and a lack of automated tools that can be used to aid medical doctors in their decisions. In addition, pharmacists (for example, in hospitals) also often fail to double-check whether medication prescriptions issued by doctors are safe. The

lack of good and helpful computer-based systems across Brazilian healthcare is a problem that is addressed in the context of this thesis.

The overall problem provides a motivation to explore in which way different techniques from formal methods and computer science can help, and how both foundation development and design of better solutions can shape the improvement of healthcare provision worldwide. This thesis contributes to research in this area and investigates the integration of various approaches in novel ways to achieve this goal.

1.2 Research problem context

The prevalence of Potentially Inappropriate Medications PIMs in prescriptions, globally and specifically in the Brazilian context, highlights the ongoing need for improvements in the prescription process. This requirement is attributed to several factors, including the absence of a formal knowledge base, the lack of dedicated reasoning engines for addressing PIMs, the diverse parameters associated with each PIM criterion, and frequent guideline updates. Moreover, not only the identification but also the development of solutions to rectify or minimise PIM-related issues remains a challenge insufficiently addressed by automated systems. Consequently, the prescription screening for managing PIMs by healthcare professionals can be a complex, time-consuming, and error-prone task, potentially making it unfeasible in daily prescribing practice without dedicated system support.

The development of an approach that handles a variety of different types of interaction constraints, for instance, constraints between age x drug x drug, age x drug x disease and age x drug x gender, that constitute the PIMs rules, might be a solution to embrace the above-mentioned problems. Furthermore, this approach should facilitate multiple reasoning by inference engines with a formal knowledge base in a semantic, readable way by humans and computers alike, which might be an approach to embrace the above-mentioned problems.

A computer-based system such as a clinical decision support system framework could be used to detect interactions, supporting health professionals in taking better decisions. Nevertheless, detecting interactions is just one step of the decision process. After the interaction is detected, the professional already has a decision to make: either prescribe the drug with interaction or seek an alternative drug

that would avoid this conflict. Consequently, a framework could support health professionals in finding other suitable alternative drugs in a knowledge base that provide an equal or similar therapeutic benefit. Though seeking an alternative does not sound too complex, finding a drug that matches all the other prescribed drugs, with patient characteristics and clinical conditions, may involve a series of complex constraints, which can be very demanding to be accomplished by humans without the aid of suitable tools.

Sometimes, there are situations where it is not feasible to find alternative drugs that meet the interaction constraints, and then a different solution needs to be proposed. One common practice is to adjust the drug schedule to avoid administering interacting drugs together [121, 53, 38]. Rescheduling medications can, however, be a time-consuming task, especially when dealing with multiple prescriptions and other variables like hospital routines and standard administration times. It can be challenging to schedule prescriptions that meet all the constraints, and it is often unfeasible to find an optimal, safe solution without the support of an automated system. Indeed, this is also a task that could be automated.

Considering the aspects mentioned above, there are many challenges in handling a CDSS framework towards solving complex drug-related issues such as (1) how to build a shareable knowledge base with inappropriate drugs and alternative drugs, (2) how to define inference rules that consider the patient and prescription profile to detect inappropriate drugs and suggest alternative drugs if these are available, (3) how to minimise the interaction efficiently by rescheduling drugs when possible and, (4) how to validate and evaluate the framework. Therefore, to embrace all the aspects of dealing with drug issues (and find a framework that could be similarly applied to tackle other complex medical issues), we formulated the following research question:

> *What formal semantics and structures are required to develop a CDSS framework that can handle potentially inappropriate drug prescribing effectively through the use of a reliable knowledge base and which supports various reasoning purposes?*

To gain a deeper understanding of the research question, we can break down the research question into following sub-questions:

- How can we formalise PIMs in a clinical knowledge base for reasoning?

- Which reasoning methods best support clinical decision-making for PIMs?
- Can the proposed approach be used in practice to identify and solve PIMs in real scenarios?

These research sub-questions will guide us in defining a framework to formalise the knowledge captured from PIMs clinical guidelines and perform reasoning over it. Therefore, we must define a formalisation approach that suits a reasoning methods that will detect and tackle PIMs. Additionally, we want to check whether this approach can be applicable to a realistic scenario, integrated with EMR data, where it can be validated and compared with other approaches.

1.3 Objectives

To address the research question, this thesis aims to demonstrate the capability of using an ontology/rule-based approach in a clinical knowledge base to identify potentially inappropriate medication(PIM). Moreover, with an SMT solver, suggest alternative medications and administration schedules to solve or minimise PIM undesirable side effects. We investigate the foundations of a formal framework to support different reasoning tasks over drug interaction constraints. Therefore, to build the framework, we decomposed the following objectives to structure a roadmap to achieving the broader objectives outlined by the research questions:

- Define a framework workflow to tackle drug interaction problems;
- Create a knowledge base which adequately formalises PIMs constraints and relevant data to the solution of drug interactions;
- Develop reasoning mechanisms to provide knowledge for supporting clinical decision making when dealing with prescriptions for the elderly;
- Validate the completeness of the knowledge base;
- Developing a proof-of-concept tool to evaluate the framework with a real hospital dataset.

In Figure 1.1, we sketch the proposed approach to outline the overall strategy and methodology, at a very high level, that incorporates the research questions and

the objectives to support health professionals in their decision-making process concerning patient prescriptions. The approach is composed of three main parts: the input data provided by health professionals and EMR, the inference engine developed (with an ontology reasoner and constraint solver) and the knowledge base built in an ontology.

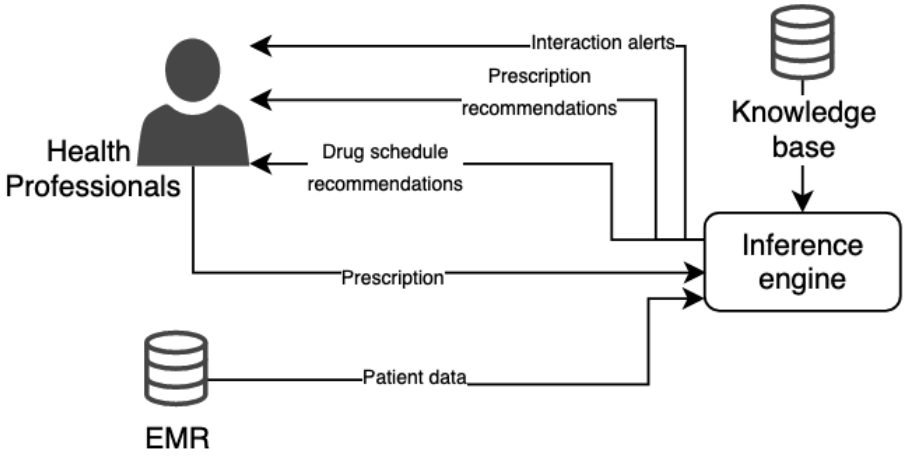


Figure 1.1: Approach overview

The process starts when the health professional prescribes drugs for a patient, and the prescription data is sent to the inference engine. Patient data, which include drugs that the patient has been taking, diagnosed conditions, age and gender, is also sent to the inference engine. The inference engine initially gathers the input data and matches it with the knowledge base rules. Thereupon, it infers and checks whether there are interactions or not. If interactions are found, the inference engine tries to solve them first by seeking alternative drugs that fit the constraints. In case no alternatives can be found, the engine tries to reschedule the drugs to minimise the interaction peaks. As a result, the inference engine returns to the user alerts with the interactions found, recommendations of alternative drugs to resolve identified interactions, and/or a revised medication schedule to minimise the effects of drug interactions.

The above vision combines multiple approaches to solving or minimising interaction problems by combining the inference engine and knowledge base, which will be explained in detail in the following chapters. Furthermore, to analyse the model correctness, we will evaluate our approach in a Brazilian hospital EMR database context. Additionally, we will measure the number of interactions, alternative drugs, and drug rescheduling and benchmark our findings against the Brazilian

hospital CDSS, rooted based on the Beers Criteria guideline.

1.4 Outline

In the next chapters, this thesis will go into detail on how we investigate and address the objectives that we want to achieve. This thesis is organised as follows:

Chapter 2 establishes the context of the research and the related work that sets the foundations and motivations of our work to solve drug interaction problems. In this chapter, we first introduce basic concepts of subjects related to this research to facilitate the understanding of our contribution. Then we detail the related work to discuss and compare the techniques used to solve drug interaction problems.

Chapter 3 starts by presenting the main concepts and structure of the CDSS. We describe the framework workflow and explain the tasks that belong to it and their outputs. Part of this chapter was published in Redeker and Bowles (2020).

Chapter 4 defines the concepts, requirements, features and properties of the ontology for the CDSS. It explains how the ontology was built, how drug interaction rules were defined and how the discovered knowledge can be extracted from the ontology to an external environment.

Chapter 5 describes the implementation of the formal model to define drug alternatives. This chapter describes the implementation rules defined over the ontology to provide drug alternatives for the prescribed drugs, the integration of the ontology results with the SMT solver and finally, the SMT solver based approach used to check if there are alternative drugs that fit all the prescription requirements.

Chapter 6 defines how the drugs were rescheduled when the interaction between drugs could not be avoided. This chapter presents the formalisation of the SMT-based scheduling rules required to maximise the distance between the administration of drugs with interaction to minimise their effects.

Chapter 7 details the development and integration of the components that form the CDSS framework. We demonstrate how the requirements and formalised rules were implemented in the knowledge base and in the inference engines. Moreover,

it describes the validation of the adopted approach by executing test cases in order to consist the correctness and completeness of the framework.

Chapter 8 describes the evaluation of the framework. First, it explains the details of the hospital dataset used and how it can be integrated with the developed framework. Then, it describes the experiments done to evaluate the framework, and compares the developed approach with other existing tools available. It gives a detailed account of the obtained results and how it compares with current solutions used by the Brazilian hospital and beyond.

Chapter 9 summarises our findings overall and gives suggestions for further research directions and future work.

Chapter Two
Context

2.1 Background

2.1.1 Clinical Decision Support Systems

Clinical knowledge is updated and expanded continuously, quickly leading to a challenging and very complex environment for health professionals to gather information and make inferences concerning patient data and their care. Inevitably, this leads to situations where clinicians may overlook some vital information or take longer to make a decision [21, 50]. A clinical decision support system (CDSS) can effortlessly tackle complex tasks and provide accurate information using patient data generating patient-specific advice, enhancing the decisio-making process when needed [151]. CDSS are tools that (should) support health professionals' decision-making process during clinical tasks for individual patients considering their current clinical condition [13, 151]. It has been shown that the incorporation of CDSS can substantially prevent medical errors leading to an improvement over patient safety and the quality of care [13, 139]. For a CDSS to be designed to improve the quality of decision-making, it needs to gather accurate and correct clinical knowledge (e.g. guidelines, ontologies and clinical knowledge bases), patient information and other evidence-based digital health information [148].

2.1.1.1 CDSS functions and interventions

There are different directions that CDSS can tackle. For example, they can be incorporated into medical practice to detect drug-drug, drug-food and drug-

allergies interactions, suggesting drug alternatives, drug doses, selecting appropriate therapy, supporting diagnoses and comparing drug and clinical parameters such as laboratory values [74, 50, 148].

The categorisation of a CDSS is usually based on the approach that the system was built for, which consists of the system function and system interventions. The system function has two methods: to assess whether the decision that was taken is true or not, which may be used to define a patient diagnosis or detect interactions; and to support the health professional in what to do, for example, when a diagnosis is defined, which drugs or guidelines are recommended, or in case of interaction is found, what are the options to avoid or minimise it. Most of CDSS provide a hybrid solution considering the two methods, assess the patient data and support the decision [156]. As for the interventions, they are classified as active or passive. An active intervention supports the user without the need to ask for it. This kind of intervention works in real-time, and the system automatically assesses the data supporting the user [81]. However, this kind of intervention sometimes overwhelms the user due to the excessive number of alerts or recommendations, driving the user to overlook them. On the other hand, a passive intervention demands more user effort, as the system will check the data only when the user considers that the system intervention can be valuable. This intervention prevents an excessive number of alerts or recommendations. However, relevant interventions may be left out.

2.1.1.2 Knowledge and Non-knowledge-based CDSS

A CDSS can be seen an intelligent system that provides accurate knowledge to support health professionals [151]. The intelligence embedded in the system can be categorised according to the source of knowledge that the system infers from patient data, and can be knowledge-based or non-knowledge-based [148].

Knowledge-based

A *knowledge-based* CDSS, supports health professionals through the knowledge obtained from many databases (e.g. Drugbank [159]), literature-based, practice-based, or patient-directed evidence. The knowledge is usually extracted and validated from these sources by professionals, from which logic rules following the assumptions IF-THEN can be created [148]. The general concept of knowledge-based CDSS designed by Berner and Lande (2016) consists of three components: a mechanism to communicate with the user(input/output), a knowledge base, and

an inference engine as shown in Figure 2.1.

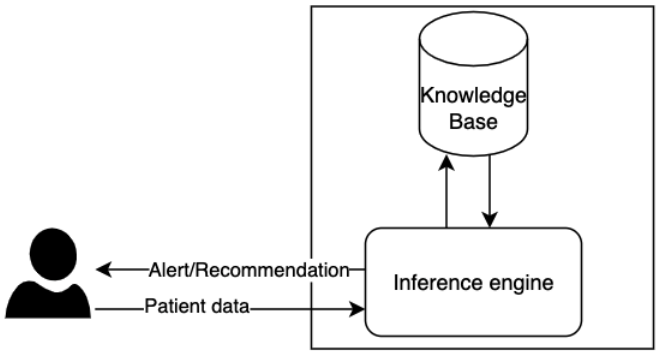


Figure 2.1: Simple user interaction with a knowledge-based CDSS

The communication mechanism between users and CDSS happens by sending and/or receiving data. The process of sending data happens either by entering patient data directly or by passing data indirectly through a system with EMR, while receiving data happens by displaying the result from the inference engine in a front end (e.g., alerts or recommendations).

A CDSS needs a medical knowledge that aligns with the inference engine. The inference engine is the key component responsible for reasoning through the contents of the knowledge base. Without this alignment, the CDSS cannot be effective in providing accurate clinical decision support [9, 75].

The inference engine's function is to check whether there are alerts or recommendations from the knowledge base and send them back to the user. This process begins by receiving the patient data, selecting rules from the knowledge base that matches the data, to finally infer these rules and the data.

The knowledge base compiles and stores rules defined from the knowledge sources, usually in the form of IF-THEN rules [13]. An example of an IF-THEN rule to prevent drug interaction prescriptions is: IF a new order is placed for a drug and this drug interacts with another drug that the patient is already taking, THEN the system displays an alert, informing the interactions, and might also recommend another drug with the same therapeutic benefit but without interactions.

Non-knowledge-based

A non-knowledge-based CDSS is a non–rule-based approach [165]. The creation

of knowledge is not based on human perception but done through artificial intelligence, machine learning, or statistical pattern recognition [148]. In knowledge-based approaches, the user must create rules into the system to build knowledge. Thus, the system cannot learn from the patient data or previous decisions. In contrast, a non-knowledge-based approach comes from the system's ability to learn from a data source without demanding an external domain knowledge such as medical expertise [120].

Similar to the knowledge-based CDSS, the non-knowledge-based CDSS has four components: a mechanism to communicate with the user(input/output), an algorithm, and an AI inference engine, as shown in Figure 2.2. The mechanism to communicate with the user has the same functions as the knowledge-based, which is the interaction between the user and the system, by entering patient data and getting results from the AI inference engine. The AI inference engine infers the patient data through the algorithm, generating an output (alert or recommendation) for the user [148].

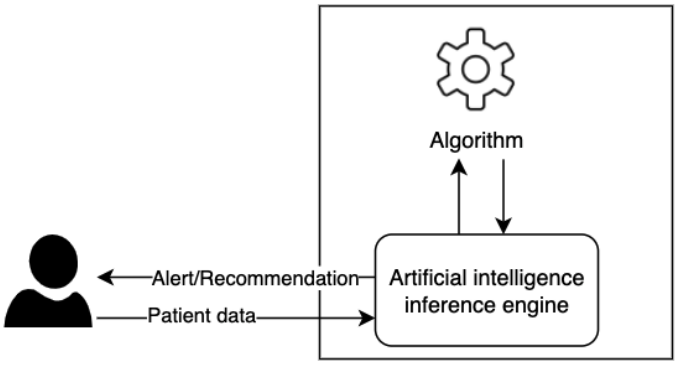


Figure 2.2: CDSS Nonknowledge base

Knowledge-based approaches are built in rule-based logic, which easily allows traceability, a key aspect considered in this research and highlighted in the motivation. The detection and resolution of the PIMs provided by our approach must be traced and explained until the information source. In Non-knowledge-based approaches, traceability can be very complex and imprecise, leading us to avoid using it.

2.1.2 Knowledge representation and reasoning

Knowledge is defined by Chen, (2010) "as information (which can be expressed in the form of propositions) from the environment". According to Bolisani

and Bratianu, (2018), "Knowledge can be obtained only from rational reasoning grounded in axioms, like in mathematics, and it should be distinguished from opinion, which is a product of our senses". Hence, knowledge can be defined as the reasoning result of a logical structure. Reasoning is one of the primary forms of simulated thinking process and inferring new conclusions from existing assumptions[30].

Knowledge representation is how knowledge propositions are formally structured by symbols[30]. The knowledge formalism provides a means to represent it in an ambiguity-free and systematic way. [123]. In computer science, there are several approaches for knowledge representation, such as rule-based systems[163], semantic nets [136], and epistemic modal logic[77]. In cases of shortages of formal means for representing knowledge, this can result in misinterpretation, and, consequently, in poor decision-making [78].

Knowledge representation and reasoning can be defined as manipulating symbols and encoding propositions to produce representations of new propositions. The goal is to encode a particular kind of knowledge by a formal language, and infer new knowledge from it with a reasoner [30, 20].

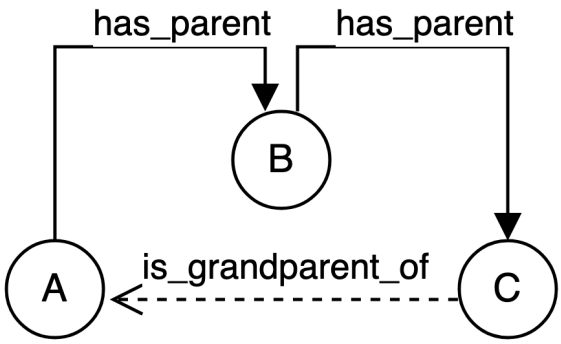


Figure 2.3: Knowledge representation

For example, in Figure 2.3, we formalise a very basic family genealogy of three elements, A, B and C, linking them with the property has_parent, in which A has parent B (say $P(A,B)$), and B has parent C (say $P(B,C)$). We also create the object property is_grandparent_of (say Q) as a transitive property, which means that if individual x is related to individual y by P, and y is related to individual z by P, then x will be related to z by Q, which is formulated in general by: $\forall_{x,y,z} P(x,y) \wedge P(y,z) \Rightarrow Q(z,x)$.

In the formula, the elements x, y and z are individual persons, P is the property has_parent, and Q is the property is_grandparent_of, thus, if x has_parent y and y has_parent z, it implies that z is_grandparent_of x. The assertion of C is_grandparent_of A was not defined previously. This straightforward example demonstrated how the reasoner is able to infer and discover new assumptions based on previously formalised ones.

2.1.2.1 Ontology

Compared to popular knowledge databases, an ontology can show information in a logically structured and user friendly way. In the context of our problem domain, popular drug interaction knowledge databases do not have all the needed information required, are not available in a structured format and are not able to provide interoperability with other ontologies [103]. An ontology is useful in this context because it is a straightforward formalism that enables inserting and editing medical knowledge from multiple sources [97] and can thus solve semantic interoperability issues [103].

Ontologies establish a formal definition of concepts used to express either generic knowledge or knowledge from a particular domain or practice [67, 147]. An ontology is defined with entities, attributes and relationships that present an interoperable format understandable by both humans and machines. It can therefore be used to develop knowledge-based systems. [30]. Schulz and Jansen (2013) affirm that "Ontological representation of entity types presupposes the existence of an objective reality about which truths can be discovered by scientific methods". As we will see later on, we too assume that the knowledge we capture and represent is evidence-based and gathered from medical research, clinical studies, and so on.

An ontology comprises four main components: *concepts* (classes), *instances* (individuals), *relations* and *axioms*. Concepts represent a set or class of entities within a domain and are expressed using formal definitions that capture the requirements for class membership. Concepts can be organised in superclass/subclass hierarchies, known as a *taxonomy*. For example, Paracetamol and Drug are two classes. Since Paracetamol is a drug, it also belongs to the class Drug. Thereby, it is implied that all instances of Paracetamol are Drugs.

Concepts can be defined in two categories:

- Primitive: only has the necessary conditions and is insufficient to fully define the concept [79, 147, 72]. In other words, a primitive concept does not have unique relationships sufficient to distinguish it from their parents and siblings, and therefore do not have any specified equivalent classes. It is defined by the subsumption operator $\subseteq$.
- Defined: is defined by the equivalence operator $\equiv$ and is specified with the full set of necessary and sufficient conditions. In other words, defined concepts have both necessary and sufficient relationships to distinguish them from their parents and siblings [79, 147, 72].

Instances (individual): represent objects in the domain of interest, they can be referred to as instances of concepts. An ontology should not contain any instances, because it is supposed to be a conceptualisation of the domain. The combination of an ontology with associated instances is what is known as a knowledge base. However, deciding whether something is a concept of an instance is difficult, and often depends on the application.

Relations: express relationships between two concepts or objects(e.g. classes or individuals). For example, a class patient could be represented as a relationship between the classes person and hospital. There are two common types of relation: taxonomic and non-taxonomic [149, 80]:

- Taxonomic relations represent direct is-a or sub-classOf relationships between classes, forming the ontology's hierarchy of entities. For example, Paracetamol is sub-classOf Drug.
- Non-taxonomic relations are links that enrich the ontology semantically, but do not alter its structure. For instance, side effect might not be a non-taxonomic relation. For instance, Paracetamol hasSideEffect Mouth ulcers.

Axioms: enforce restrictions on classes or instances. These formal definitions are typically expressed using logic-based languages like first-order logic. They are categorised based on the role they act. For example, a disjointness axiom between classes states that instances of one class cannot be instances of another class (or equivalently, the classes do not share instances). A domain axiom assigns a domain class to a property (relation), meaning that any instance linked to that

property must be from the domain class. Similarly, a range axiom assigns a range class, such as integer or string, to the values of a property [149, 80].

2.1.2.2 Semantic Web Rule Language

Many health decision process systems, such as a CDSS, are often modelled by a declarative approach, for example as a rule-based system. Nevertheless, interoperability among current rule-based systems is restricted. The Semantic Web Rule Language (SWRL) has emerged as a first-step solution to improve rule-based systems interoperability [87].

Similar to other rule languages, SWRL rules are written as antecedent-consequent pairs. The meaning of the rules can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold [114]. Furthermore, SWRL provides inferential reasoning capabilities that can deduce new knowledge from an existing ontology [168].

A simple example [161] of how an SWRL rule can assert that the combination of the *hasParent* and *hasBrother* properties implies the *hasUncle* property could be defined as follows (in SWRL notation):

$$hasParent(?x1, ?x2) \wedge hasBrother(?x2, ?x3) \Rightarrow hasUncle(?x1, ?x3)$$

The rule can be read as follows: if $x1$ has parent $x2$ and $x2$ has brother $x3$ then $x1$ has an uncle $x3$. The conditions before the implication symbol form the *antecedent* of the rule, whereas what comes after the implication is known as the *consequent*.

2.1.3 SAT/SMT Solvers

SAT (Boolean Satisfiability) and SMT (Satisfiability Modulo Theories) solvers are sometimes seen as efficient automatic theorem provers for classical logic [91, 130]. A SAT solver can be used for solving problems encoded in propositional logic (e.g., the rules of a game such as Sudoku) by finding a variable assignment (known as a solution) such that formulae representing the problem are satisfied [106]. An SMT solver extends SAT by supporting higher-level theories such as arithmetic, arrays, bit-vectors, sets, and more. In other words, SAT and SMT have similar purposes, but SAT deals solely with Boolean variables, whereas SMT can capture more expressive theories and combine their reasoning capabilities.

These solvers have a wide range of applications, such as, planning [18, 52] and scheduling [133, 131] and formal verification [96]. In health the domain SAT/SMT have been applied for example, for tackling conflicts in medical guidelines [23], finding the best solutions for treatment [24], avoiding medication conflicts [83].

Some of the most commonly used SMT solvers include Z3 [40], CVC5 [10] and Yices [49]. The difference between these solvers lies mainly in built-in first-order theories that they support, how forthright it is to define the constraint and performance in solving problems.

Satisfiability checking aims at automated solutions for determining the satisfiability of existentially quantified logical formulas [3]. Such formulas are Boolean combinations of theory constraints, where the form of the theory constraints depends on with which theory is instantiated. Most of the SAT/SMT solvers are based on the DPLL algorithm based on backtracking to decide the satisfiability of propositional logic formulas. The process of implementing SMT models starts with the problem formulation in which the problem is expressed in mathematical logic, such as first-order logic, indicating all necessary conditions and restrictions [59]. Based on this description, the solver translates the restrictions into mathematical formulas, considering the specific theories related to the problem. Thereafter, the solver translates the restrictions into mathematical formulas, considering the specific theories related to the problem.

Next, the solver applies satisfaction techniques to determine whether values are assigned to variables that satisfy all conditions simultaneously. If a solution satisfies all the constraints, the solver presents a model with the values. However, if no value assignment satisfies the constraints and consequently a solution cannot be found, the solver provides evidence explaining why the constraints cannot be satisfied simultaneously[41].

2.1.3.1 SMT-based optimization

In addition to determining satisfiability, in some SMT solvers, optimisation objectives such as maximisation and minimisation can be defined. In the case of minimisation, it seeks the lowest value; for maximisation, it aims for the largest. For example, to optimise a schedule, the objective is to maximise schedule occupancy. On the other hand, a minimisation objective could be applied to minimise the production costs, operational expenses, or financial risks. When a maximisation and minimisation objective is defined, the SMT solver continues

its exploration until it identifies a solution that satisfies all constraints while optimising the objective function. The solver gradually adjusts the values of the variables, using heuristic search techniques or specific optimisation algorithms aiming to refine the search towards the most promising region of the solution space. The optimisation process is iterative, with the solver continually updating variable values to improve the approximation of the optimal solution. In addition to improving the objective function, the ideal solution also satisfies all the constraints of the problem [15, 88, 133].

2.1.4 Drug Interaction

Drug interactions can be defined as the act of one drug modifying the effect(s) or action(s) of another drug, food, alcohol or herbal products [92, 109]. Additionally, drug interaction can also be caused by the effect that a drug has on a patient condition, for example, disease, clinical condition or nutritional status [92]. Interactions are not always seen as an unwanted reaction, since they sometimes represent a therapeutic benefit. Nevertheless, in some cases, they potentially have life-threatening consequences [68]. Moreover, drug interactions may result in adverse side effects or reduce or enhance the action of medications. For example, leading to toxic reactions or impacting the efficacy of the drugs administered [26].

2.1.4.1 Interaction types

Interactions can broadly be classified as pharmacokinetic and pharmacodynamic [68]. Pharmacokinetics can be defined as what the body does with drug, such as absorption, distribution, metabolism or excretion of the drug, resulting in a modification of the drug concentration in the body [92, 43].

The pharmacokinetics process starts with absorption of drug molecules crossing biological membranes from the administration site into the plasma. An example of an interaction effect in the absorption process is the binding of drugs that may change the gastric pH delay absorption of certain drugs in the stomach.

After being absorbed in the distribution process, the drug is delivered from the plasma to body tissues and organs. An interaction can affect this process, resulting in a change in drug concentration at the site of action. The metabolism is the process of converting drug molecules into active metabolites compounds throughout the body, also called biotransformation, which is mainly carried out

by the liver. An interaction may increase the concentration of some drugs by decreasing their metabolism.

The excretion process is responsible for the clearance of the drug from the body, usually eliminated in the urine. An interaction in this process may happen when drugs compete for the same excretion activity, resulting in the decrease of the elimination of the drugs and increasing the toxicity in serum concentrations [45, 109, 92, 61, 16].

Pharmacodynamics is defined as what a drug does to the body. It can be defined as the altered pharmacological effect of a drug by a combination with another drug through the relationship between the drug concentration and its receptors, mechanism of action, and therapeutic effect [108, 152]. The interaction happens when a drug has an antagonistic, additive, synergistic or indirect pharmacological effect on another [71]. Moreover, it is more challenging to detect pharmacodynamic interactions in comparison to pharmacokinetics, and it can result in significant losses to human health. For example, combining drugs with opposing effects can result in loss of drug effect or intensify drug activity with exaggerated or unusual effects. [45, 144, 61]

2.1.5 Drugs potentially risky for elderly people

The elderly represent the major consumers of drugs, mainly due to chronic diseases that require the prescription of multiple drugs, known as polypharmacy. Polypharmacy is characterised as the concomitant use of multiple medications [31, 99], and is commonly linked to the treatment of multiple chronic diseases, also known as multimorbidity, which occurs with a high prevalence in the elderly [154]. The use of multiple medications must be safe and effective [110], and this should be under scrutiny when the risks of medication combinations outweigh the benefits, resulting in adverse drug reactions. An adverse drug reaction is defined as any undesirable medical occurrence caused by a pharmaceutical product. However, it is not necessarily related to treatment[124], as it may occur for different reasons, such as wrong drug, dosage or route [35].

Age-related physiology changes renal elimination and liver metabolism, consequently affecting pharmacokinetics and pharmacodynamics. Moreover, factors such as physiologic changes, decline in functional abilities, reduced homoeostatic mechanisms, disease associated with ageing and frailty, and polypharmacy are

risks associated with developing clinically significant drug-drug interactions. Hence, older adults are more susceptible to drug interactions than younger ones. [92, 68, 45, 109, 28]

Concerning hospital admissions, a study [32] revealed that 43.3% of patients were taking at least one potentially inappropriate medication (PIM) and during hospitalisation this number increased. Between 22,2% to 61.9% of the patients were prescribed PIMs throughout the hospitalisation, where on average each patient took 14,4 different medications [64]. A high rate of patients was observed with three or more comorbidities and with polypharmacy or high-level polypharmacy.

A strong correlation between polypharmacy and multiple chronic diseases with PIMs has been established in [89, 27, 141, 166, 100]. To treat multiple chronic diseases, physicians might prescribe multiple drugs [11] and patients are hence more likely exposed to problems related to over-prescribing and inappropriate use of concomitant drugs [94], such as adverse drug reactions and drug interactions [112]. For patients who need more than three medications, the risk of therapeutic problems is more than 50% [90], and is related to adverse drug events [150]. Consequences of such events include an increased risk of hospitalisations, effects on the quality of life [143], and ultimately a higher risk of morbidity and mortality [150]. The detection of polypharmacy could reduce adverse outcomes [11] and improve the patient's quality of life and life expectancy [143].

2.1.5.1 Beers Criteria

The Beers Criteria [1][2] is a list of drugs that should be avoided in most circumstances by patients aged 65 or above. It was first established by Dr Mark Beers and colleagues in 1991 for long-term care facility residents [12]. Since then, it was updated by him until 2011, when the American Geriatric Society (AGS) took over the responsibility for updating and maintaining it. Since 2012, it has been periodically updated at 3-yearly intervals [140].

The list is divided into five categories:

1. Potentially Inappropriate Medication Use in Older Adults (Table 2);

2. Potentially Inappropriate Medication Use in Older Adults Due to Drug-Disease or Drug-Syndrome Interactions That May Exacerbate the Disease or Syndrome

[1] This thesis used the 2019 version of the Beers Criteria [117].
[2] An updated version of the Beers Criteria has been recently released in 2023. [118].

(Table 3);

3. Potentially Inappropriate Medications: Drugs To Be Used With Caution in Older Adults (Table 4);

4. Potentially Clinically Important Drug-Drug Interactions That Should Be Avoided in Older Adults(Table 5);

5. Medications That Should Be Avoided or Have Their Dosage Reduced With Varying Levels of Kidney Function in Older Adults (Table 6).

For the definition of criteria for each category, drugs are listed according to a specific hierarchy. The first category of drugs are classified according to Organ System, Therapeutic Category and Drug; the second in Disease or Syndrome; the third only in Drugs; the fourth in Object Drug and Class; and the fifth in Class Medication and Medication. Moreover, each criteria has a Rationale, Recommendation, Quality of Evidence and Strength of Recommendation, which is defined based on the reviews conducted by the AGS [117].

The Rationale explains the reasons for the criteria, for example, uncertain effectiveness, safer alternative drugs or an adverse reaction such as physical dependence or heart failure. The Recommendation column describes what should be done on each criteria, which is basically avoid, reduce the dose or avoid in a specific circumstance (for example, "Avoid for treatment of nocturia or nocturnal polyuria"). The Quality of Evidence is classified as High-quality, Moderate-quality and Low-quality rates which are defined by the AGS merging the American College of Physicians based approach and the Grading of Recommendations Assessment, Development and Evaluation based approach. Finally, the Strength of Recommendation is classified either as Strong where the harms, adverse events, and risks clearly outweigh benefits or Weak where the harms, adverse events, and risks may not outweigh benefits. These are defined based on the quality of evidence, the frequency and severity of potential adverse events [117].

It should be remarked that the drug list is a recommendation of drugs, and it does not mean that the drugs should never be used [46]. The list is developed to support clinical decisions as a clinical guideline with a clinical recommendation [65]. Thus, the prescription of listed drugs should always be analysed whether the risks may outweigh the benefits for each patient [46].

As noticed, the Beers Criteria has a category named Potentially Inappropriate

Medication Use in Older Adults (PIM), however PIM is also the main class of all drugs that are considered inappropriate, including the Beers Criteria ones. Consequently, it can be confusing/ambiguous to identify to which PIM the drugs are classified. Therefore, in the text, when referring to the category PIM of the Beers Criteria, we will use the acronym Beers Criteria Potentially Inappropriate Medications (B.PIM).

2.2 Related Work

As mentioned earlier in the thesis, medication errors are a leading cause of preventable harm to patients [157]. According to the World Health Organization (WHO)[113], these errors are originated from systemic issues and human factors such as fatigue, suboptimal environmental conditions, or staff shortages, impacting various stages of the medication process, including prescribing, transcribing, dispensing, administration, and monitoring. The consequences of such errors can range from severe harm to disability and, in extreme cases, death.

The WHO launched the third Global Patient Safety Challenge to tackle unsafe medication practices and medication errors [113]. It emerged from the recognition that medication errors and associated harm remains a problem worldwide. Furthermore, strategies for enhancing medication safety and reducing medication errors commonly make use of CDSS [74, 148]. Therefore, medication management is a promising target for patient safety measures.

Prescription of PIMs is a worldwide problem, associated with increased adverse drug reactions, mortality, and healthcare costs [126]. Monteiro et al. (2019) affirms that there are various approaches to preventing PIMs. For example, the use of a CDSS may decrease the number of possibly incorrect prescriptions. Computerised interventions have been proposed as a viable technique for improving prescriptions [36], which can also positively influence and change physicians' prescribing practice to prevent the prescription of inappropriate drugs [126]. However, according to Sönnichsen et al. (2016), there are limited pieces of evidence that current approaches improve clinically relevant endpoints. Therefore, more effective strategies are needed.

This section gets across selected research done in the knowledge representation and reasoning area/field for CDSS to detect and solve drug-related problems. First, we analyse some approaches that partially tackle drug and disease related

problems, then, we present a bibliometric analysis about CDSS to tackle PIMs. We then go into detail about the main studies found in the bibliometric analysis related to CDSS for tackling PIMs. Additionally, we analyse ontologies' knowledge base, drug recommendations, conflict recommendation management, and drug scheduling approaches developed to handle PIMs.

2.2.1 Drug and disease-related problems approaches

Several approaches aim to enhance healthcare clinical decision support systems (CDSS), each offering distinctive insights and technical methods. For example, [162] compares patient data with guideline recommendations and displays matching treatments. Moreover, [24],[107] and [158] addressed adverse interactions that occur when a patient with comorbid diseases is managed according to concurrently applied clinical practice guidelines, defining the best treatment path. In order to formalise guidelines, [66] introduces a framework for translating logical segments of Knowledge Artifacts into Satisfiability Modulo Theory (SMT) models, where the segments are automatically translated and verified using the Z3 SMT solver. Apart from [162], all the other approaches were implemented with formal methods such as SMT [24, 107, 66] and constraint logic programming[158].

A web-based Clinical CDSS was developed by [119], delivering access to essential clinical documents for pediatric surgery trainees. In the same way, [138] proposes the integration of Semantic Web technologies into clinical decision support systems to improve the transparency, explainability, and specificity of clinical decision support systems by leveraging semantic technologies and ontologies. The focus of [119] was improving trainees' access to clinical resources and their confidence in patient management through web analytics and surveyed trainees. In contrast, [138] intended to provide healthcare practitioners with more personalized and evidence-based clinical knowledge.

A predictive model to detect drug-target interactions from diverse biological data to detect biological process interactions was proposed by [29]. Moreover, [62] proposed a predictive model to identify the optimal set of biomarkers for sepsis, while [127] proposed a genetic-type algorithm for anaemia diagnosis, management and classification.

The proposed studies address some of the drugs and disease-related problems. If these approaches could be merged, they would provide a powerful tool to

support health professionals. However, some gaps have to be addressed when they are analysed separately. For example, [162], besides checking the compatibility of guidelines, could check the interactions that may happen. In contrast, [24], [158] and [158] check the interaction between guidelines and provide alternatives; however, these approaches do not have a knowledge base that can be modified or incremented by professionals and share with other CDSS. Additionally, they do not take into account drugs previously prescribed.

The approach proposed by [119] does not consider patient parameters to provide precise information. In contrast, [138] aimed to provide personalized and evidence-based clinical knowledge in an ontology knowledge base that allows it to be shared with CDSS. However, neither approach solves the interactions by providing alternatives or minimising them. Likewise, the approaches [29], [62] and [127], which aim to predict drug-drug interaction sepsis and anaemia, do not support health professionals in how to tackle the interactions.

2.2.2 PIMs CDSS bibliometric analysis

A bibliometric analysis was conducted to define the Potentially Inappropriate Medications (PIMs) CDSS related works. The literature search was performed on Scopus, which is the largest abstract and citation database of peer-reviewed literature. A query was performed on the database based on keywords terms related to CDSS for PIMs as follows:

((KEY ({clinical decision support system}) OR KEY ({CDSS})
OR KEY ({decision support system}) OR KEY ({CDS})
OR KEY ({clinical decision system}) OR KEY ({Computer-Assisted}))
AND
(KEY ({Potentially inappropriate medication}) OR KEY ({PIM})
OR KEY ({Inappropriate Prescribing}) OR KEY ({polypharmacy})
OR KEY ({Drug related problem})))

The query result gave 393 documents which matched the above parameters. Figure 2.4 shows the document numbers per year of publication, which indicates an increase in publications over the years, especially since 2016 when the number of publications reached more than 40 for the first time. This trend indicates that the topic has relevance in the scientific field and that gaps still needs to be addressed. The bibliometric analysis for *Potentially Inappropriate Medications (PIMs)* CDSS

highlighted 11 studies that target similar goals to this research.

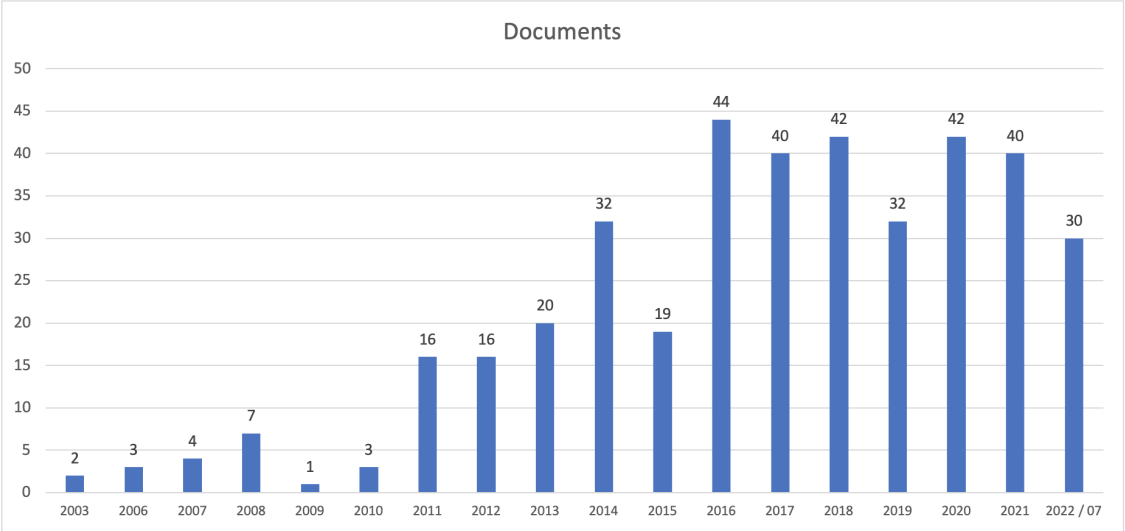


Figure 2.4: Publication per year

The selected documents from the query were refined in order to select the ones that had as a target a CDSS to tackle PIMs. Therefore, an analysis was performed to select the relevant documents according to the following criteria:

- exclusion of irrelevant documents based on the title;
- exclusion of irrelevant documents based on the abstract;
- exclusion of irrelevant documents based on the full text.

As a result of the analysis, 11 documents which fit the criteria were selected (cf. column Document in Table 2.1), which are explained in the next section.

2.2.3 PIMs CDSS qualitative analysis

The bibliometric analysis for PIMs CDSS highlighted 11 studies that target similar goals to this research. Therefore, Table 2.1 was developed to list key elements of these studies. The analysed elements were defined according to the goals of this research to investigate gaps that still need to be solved or improved.

Document	Guideline	Knowledge representation type	Shareable knowledge base	Alternative drug recommendation	Conflict management	Drug scheduling recommendation	Pharmacokinetics parameters	Input	Output
Frutos et al. (2022)	BEERS [117]	Proprietary CDSS	No	No	No	No	No	Drug Patient conditions	Alerts
Mouazer et al. (2022)	STOPP&START [111]	Ontology	Yes	No	No	No	No	Disease Lab values Drugs Patient conditions	Alerts
Rogero-Blanco et al. (2020)	BEERS [116] STOPP&START [111]	Proprietary CDSS	No	No	No	No	No	Drug Patient conditions	Textual report
McDonald et al. (2019)	BEERS [116] STOPP&START [111] Choosing Wisely Scientific literature on deprescribing	Proprietary CDSS	No	Deprescribing recommendations	No	No	No	Drugs Comorbidities Measure of frailty	Alerts Textual report
García-Caballero et al. (2018)	STOPP [111]	Excel spreadsheet	No	No	No	No	No	Drug	Alerts
Verdoorn et al. (2018)	BEERS [116] STOPP&START [111]	Proprietary CDSS	No	No	No	No	No	Drug Patient conditions	Alerts
Johansson-Pajala et al. (2018)	BEERS [115] STOPP&START [111]	Proprietary CDSS	No	No	No	No	No	Drug Symptom assessments	Alerts Quality report
Niehoff et al. (2016)	BEERS [115] STOPP [111] Inappropriate renal dosing [111]	Specialised software	No	Deprescribing recommendations Dose adjustment	No	No	No	Drugs Age Chronic conditions	Alerts

Document	Guideline	Knowledge representation type	Shareable knowledge base	Alternative drug recommendation	Conflict management	Drug scheduling recommendation	Pharmacokinetics parameters	Input	Output
Alagiakrishnan et al. (2016)	BEERS [115] Cockcroft–Gault formula	Rule based	No	No	No	No	No	Drug	Alerts
Sönnichsen et al. (2016)	NICE-EBMG EU(7) PIMs list[129] SFINX RENBASE	Rule based	No	Depres-cribing recommen-dations	No	No	No	Diagnoses Drugs Symptoms Biometric measurements Lab values	Alerts Textual report
Ghibelli et al. (2013)	BEERS [55] ACB scale [22]	Proprietary CDSS	No	No	No	No	No	Drugs	Textual report

Table 2.1: Technical description and features of the CDSSs developed for PIMs

Afterwards, we will conduct a comparative analysis of the approaches employed in these studies with those employed in this research. We conducted the qualitative analysis in three main groups: (1) how PIMs were detected; (2) suggestion of alternative drugs; and (3) if another approach such as drug scheduling recommendation was adopted.

2.2.3.1 Clinical decision support systems do detect PIM

The *first column* corresponds to the study's reference, sorted according to the publication date. The studies show that the research of developing computational mechanisms to tackle Potentially Inappropriate Medications (PIMs) is still an issue that has been demanding efforts to be solved. The studies were usually applied to patients aged 65 years or older. However, in some studies (e.g., [7, 76, 146]) only patients over 75 years were considered. The general aim of those studies consisted in addressing Potentially Inappropriate Medications (PIMs). These studies used a variety of approaches to address Potentially Inappropriate Medications (PIMs) in a different context. Some of them were applied to real scenarios and databases, such as for community pharmacies [155], primary care [132, 7], general practitioners in the ambulatory setting [56], hospitals [95, 146, 58] and nursing homes [76, 57].

In the *second column*, we list the guidelines used for each study. Here, [56] and [101] focused on one guideline, while all the others used several simultaneously. The most used PIMs guidelines in the papers were the BEERS Criteria [117, 116, 115] and the STOPP & START [111] approach. However, some studies did not consider the whole guidelines. For example, [56] used only three drug classes (Benzodiazepines, Proton Pump Inhibitors, and Non-steroidal anti-inflammatories) of the Beers Criteria guideline, [132] omitted the A1 STOPP criterion, [155] incorporated just some selected clinical rules, and [146] defined a minimum number of prescribed drugs as inclusion criteria which might exclude patients affected by only one PIMs drug. In addition to guidelines, some studies also used specific formulas to detect, for example, inappropriate renal dosing [105] such as Cockcroft–Gault [7] or particular systems such as RENBASE [146].

In the *third and fourth column* (Knowledge representation type | Shareable knowledge base), we checked how the knowledge was represented and if the knowledge base used by these studies was shareable and which architecture was used. The knowledge base is composed of the rules extracted from the guidelines. For example, the rules extracted from the Beers Criteria can be translated into a logical

form and stored in a knowledge base. There were studies [56, 132, 95, 155, 7, 58] where the rules had been developed in proprietary CDSS and others [57, 105, 146] where the rules were programmed in specialised software. In those studies, the knowledge base was not shareable, therefore, not allowing it to be reused or integrated with other sources of knowledge. There was, however, one approach [101] where the knowledge base was shareable. In [101], a framework to formalise rules in PIMs was proposed and an ontology was developed to formalise the rules. The framework is based on three main clinical elements: prescriptions, disease and observations. For example, rules from the guideline STOPP & START [111] were formalised.

In the *Input* column, we analysed which kind of data the approaches handled to detect the interactions. Three studies [57, 7, 58] only assessed the drug input. Notwithstanding, most studies considered other kinds of relevant data such as the patient condition, diseases and laboratory exams, which are vital to cover all the guideline requirements.

In the *Output* column, we evaluated the outcomes provided by each CDSS. The most common output was alerts to notify health professionals about the detection of drug-related problems. In addition, [132, 95, 146, 58] provided textual reports that consist of a descriptive report of the findings.

2.2.3.2 Drug recommendations

The prescription of potentially inappropriate medications may increase the risk of harm. Therefore, AGS Health in Aging Foundation [73] and Hanlon et al. (2015) [63] proposed a list of evidence-based alternative medication treatments to avoid problems that might be caused by inappropriate drugs, along with some non-pharmacological approaches when appropriate. These lists of alternative drugs could be incorporated into a CDSS in order to suggest safe alternatives when an inappropriate drug is prescribed.

The *Alternative drug recommendation* column analysed if the proposed CDSSs suggested alternative drugs in case of drug problems, to understand how the rules were designed and in which situations the CDSS provided the alternatives. In this regard, no approach recommended alternative drugs. Nevertheless, three approaches [95, 105, 146] provide deprescribing recommendations and [105] additionally provided dose adjustment advice. These studies did not suggest alternative drugs for drugs classified as PIMs. Furthermore, when

providing alternatives or dealing with multiple knowledge bases or guidelines, it was checked if the CDSS could detect and solve inconsistencies. For example, inconsistencies between drug recommendations, such as when one guideline recommends a drug while another recommends avoiding the same drug. None of the listed studies considered this feature.

2.2.3.3 Drug scheduling

Drug interactions can be co-related with the time that drugs are administered, so-called Time-dependent Drug–Drug Interaction. The coadministration of drugs that have interaction can decrease absorption or affect the metabolism of one or both of them. Consequently, some strategies exist to avoid or minimise the side effects of this interaction by defining appropriate administration times and staggered dosing.

Consider administering drugs at distinct times may minimise the interaction [4]. The changes in pharmacokinetic parameters such as absorption, distribution, metabolism, and elimination can be eliminated when properly scheduling the time to peak drug concentration in serum TMAX between drugs that interact[164]. Nagai et al. (2022) [102] affirms that "to take advantage of its benefits in clinical practice requires full comprehension of this strategy on the part of both prescriber and patient".

Some studies have proposed a rescheduling approach to tackle drug interactions, such as [153] and [69]. However, they do not automate the process of minimising the interaction by considering the TMAX. According to Van der Sijs et al. (2009) [153], the approach was considered inefficient and error-prone in suggesting interaction interventions.

Therefore, we checked if the studies listed in the bibliometric analysis proposed an alternative approach in case of drug interaction without alternative drugs, such as the drug rescheduling recommendation. Moreover, we analysed which parameters were considered in this approach, such as Pharmacokinetics. However, none of the listed studies considered an alternative approach to tackle PIMs.

2.2.3.4 Summary of the analysis

The qualitative analysis we did on relevant existing approaches provides us with insights on what is currently available for tackling PIMs as well as current

limitations. Regarding the guidelines, the majority of the studies used at least one of the widely used guidelines (Beers Criteria and/or STOPP&START), which highlights that these guidelines are universally considered standards for listing PIMs. However, the lack of an approach that provides a shareable knowledge base indicates that sharing knowledge between CDSS is still a problem that needs to be addressed, as only [60] considered this to some extent. System interoperability is crucial in healthcare contexts because many other systems and tools are already in place in this domain. Not addressing this from the beginning will limit the applicability and adaptability of any new developed solution in practice.

Several studies used the same PIMs guideline, and thereby, a shareable knowledge base could improve the quality and facilitate the comparison among approaches. This highlights the importance of organisations (such as AGS) distributing their guidelines in a machine-readable/standardised form that would facilitate the use and comparison among approaches. Detecting a drug-related problem is the first step to help health professionals avoid prescribing PIMs. The deprescribing recommendations and dose adjustment that some studies proposed are strategies that already help health professionals to minimise drug-related problems.

Providing alternative drug recommendations, which can be assessed by a conflict management tool to avoid drug-related problems, is a further step towards supporting them in taking better decisions. Moreover, conflict management tools can provide essential support for health professionals to choose a drug without drug-related problems when alternative drugs are suggested or when multiple guidelines are used simultaneously. In several approaches, more than one guideline was used. Hence, a conflict management tool could detect nonconformity between rules. Nonetheless, these studies proposed neither alternative drug recommendations nor conflict management tools. In the case of interacting drugs that do not have suitable alternatives and need to be prescribed, we can attempt to minimise the interaction between them by rescheduling and considering pharmacokinetics parameters. This seems to be a novel approach to deal with PIMs as no other study and corresponding tool currently exists that does this.

Finally, the relevance of the approaches listed and discussed (cf. Table 2.1) is clear in the context of our work, and we will revisit them when we compare our proposed framework with theirs in Section 8.3.

2.3 Summary

In this chapter, we introduced the subjects of our study which are related to tackling PIMs. We described the concepts of a CDSS, knowledge representation and reasoning, and SAT/SMT solvers at a high-level. These are the main approaches that our framework will adopt, and we go into further details when using them throughout the thesis. Moreover, we introduce general concepts about drug interactions and, more specifically, inappropriate medications to motivate the requirements we should consider when building our own version of a CDSS framework for tackling PIMs.

We described the state-of-the-art for tackling PIMs, and presented related work that serves as the foundation for our research. As far as we are aware, no research proposes a framework to comprehensively address PIMs. For this reason, we will adapt several approaches that have been discussed in this chapter when building our framework to formally check and solve inappropriate medications.

CHAPTER THREE

CDSS FRAMEWORK WORKFLOW

In this chapter, we provide a comprehensive description of the workflow and underlying concepts of the proposed Clinical Decision Support System (CDSS) framework to detect and tackle PIMs. Here, our purpose is to give a broad overview of the CDSS, the underlying workflow, the key computer science-based techniques that form the CDSS and how they are integrated. The main contribution of the presented CDSS lies in the inference engines incorporating multiple reasoning approaches, which are presented to perform specific tasks to tackle PIMs. By reflecting on the architectural choices, we discuss how the same approach can be applied to tackle other medical issues.

3.1 CDSS framework

The main focus of our research concerns knowledge representation and reasoning of PIM constraints. To this end, we develop a CDSS framework which integrates multiple reasoning approaches with an underlying formal and shareable knowledge base. The framework scheme shown in Figure 3.1 is based on a so-called knowledge CDSS[13], a known approach used to combine user input/output, a knowledge base, and an inference engine.

The process starts when the user (e.g., health professional) enters data into the CDSS. The data consists of patient parameters such as age and gender, clinical

conditions (including lab results, diseases and syndromes) and prescription. The data is integrated into the CDSS by the inference engine which consists of three components. The first is the Beers Criteria Reasoner, which aims to detect and classify PIMs. The second is the Drug Alternative Solver, which aims to solve PIMs by finding alternative drugs. Finally, the Rescheduling Solver aims to minimise the interaction between drugs and determine when drugs that interact should be prescribed. These components interact with a knowledge base formalising the Beers Criteria, alternative drugs and the time to peak drug concentration in serum, also known as TMAX.

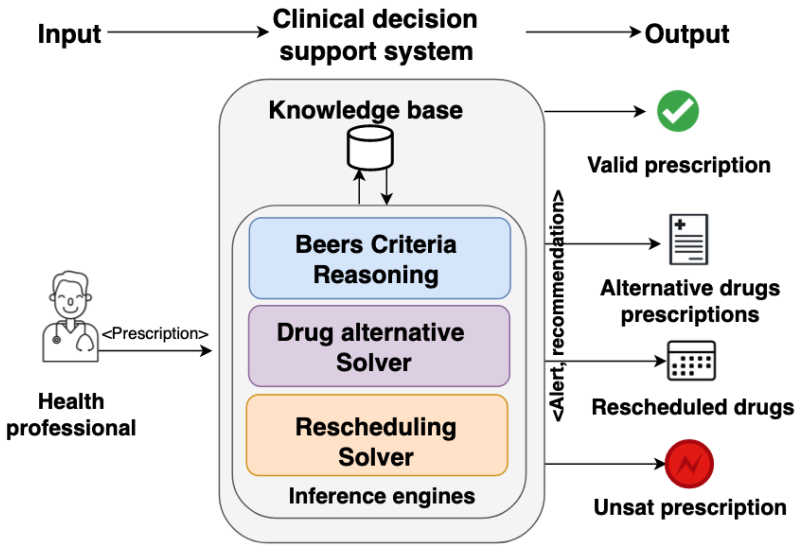


Figure 3.1: Overview of our CDSS

The CDSS provides different alerts/recommendations (outputs) for a given patient prescription (input). When no interaction is found in the prescription by the Beers Criteria Reasoner, then the CDSS returns a valid prescription message. However, when one or more PIMs are found, the Drug Alternative Solver searches for alternative drug solutions that attempt to solve the detected PIMs, returning the prescription with the found alternative drugs. If there are no alternative drugs to resolve the original interaction(s), then the Rescheduling Solver searches for ways in which rescheduling the drugs can maximise the distance between TMAX to minimise the interaction. Finally, when solvers do not find a solution, then the CDSS returns an Unsat prescription, which means that the CDSS was not able to solve the PIMs or find a better alternative.

The framework scheme demonstrates at an abstract level the main features of the CDSS and how it contributes to supporting health professionals in tackling PIMs

and taking decisions for improving the drug prescription. In the following sections, we will briefly describe each component of the framework scheme, starting with the knowledge base, which interacts with all the inference engines and is the main source of knowledge of the CDSS. Later chapters go into further details for all the elements of our CDSS.

3.1.1 Knowledge base - the Beers Criteria ontology

In order to support medical decision-making, it is essential to have a knowledge source for constructing a CDSS. Such a source can typically be acquired from a variety of resources, including drug databases (e.g. Drugbank [159]), literature, clinical practice, or patient-centred evidence. To ensure readability for both humans and computers and provide reliable decision-making information, information can be organised and formalised into an ontology [148].

Screening and early detection of PIMs improve quality-based medical care. Several guidelines and national evidence-based screening tools are available, including the Beers Criteria from the US and STOPP and EU(7)- PIM from Europe [70]. The Beers Criteria has become one of the most widely used and reliable tools by clinicians, educators, researchers, healthcare administrators, and regulators to identify PIM use in elderly patients [141, 117]. It aims to improve geriatric care by reducing exposure to PIMs [6]. Given the global utilization of the Beers Criteria and its specific adoption by the hospital database that we will conduct our evaluation, we have opted for this guideline for this research.

At a high level, our knowledge base comprises the Beers Criteria ontology, which gathers and formalises information on the Beers Criteria from journals and translates it into a formal representation (e.g., taxonomy, logical axioms and inference rules). The Beers Criteria consist of a list of PIMs that older adults in specific circumstances should avoid and can be seen as a source of knowledge for defining interactions and drug alternatives targeting specifically the elderly.

The Beers Criteria is not available in a formal representation that reasoners can interpret. To obtain a formal representation, we need a taxonomy which is best captured as a hierarchy with distinct groups of medications according to different criteria, such as drug-disease or drug-syndrome, drug-drug interactions, or drugs to be used with caution. In addition, we need logical axioms and inference rules to define the classification and groups associated with a drug. No formalisation

of Beers Criteria can be found in the literature, and this thesis hence proposes a novel ontology for it.

In order to suggest alternative drugs to those that are classified as PIM, we follow the suggestions from [73, 63] and formalise these as ontology inference rules and use them to capture the constraints that each drug has to satisfy to be considered an alternative. The alternative drugs were added to the same ontology of the Beers Criteria as they share a similar taxonomy and parameters. If the alternative drugs would have a different taxonomy and parameters or the number of guidelines were expanded, then building a specific taxonomy for alternative drugs would be appropriate. Finally, a few additional parameters are added for each drug to support the rescheduling process. With all the required knowledge suitably formalised, the ontology makes it straightforward to detect PIMs, find alternative drugs if applicable and provide drug parameters for minimising interactions by rescheduling drugs if possible.

3.1.1.1 PIMs formalisation

The Beers Criteria are commonly captured in a table with entries in natural language (this will be described in detail in Chapter 4, see Figure 4.4 for an example). The first step in the formalisation of the Beers Criteria list consists of converting the information from this table into an ontology. This process happens by identifying the elements that the ontology has to consider and the taxonomy to build the ontology hierarchy. To illustrate the process, Figure 3.2 shows a hypothetical formal representation of DrugA that is classified as B.PIM based on the Beers Criteria.

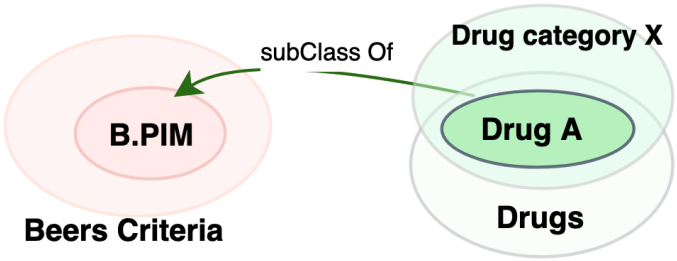


Figure 3.2: A B.PIM drug example

A taxonomy consists of a hierarchy of classes and subclasses. Here, there are two main classes: Beers Criteria and Drugs. Drug A belongs to the Drug category X, which in turn belongs to the main class Drugs. The arrow subClass Of indicates

that Drug A belongs to class B.PIM, which means that patient parameters satisfy an inference rule, as follows:

$$\forall_{d,p,pr} DrugA(d) \wedge Patient(p) \wedge Prescription(pr) :$$
$$hasPrescription(p,pr) \wedge hasDrug(pr,d) \wedge hasAge(p,v) \wedge v \geq 65 \Rightarrow B.PIM(d)$$

The above inference rule can be interpreted as follows: for any drug d of the class drug A, and an arbitrary patient p and prescription pr such that pr contains d, and p's age is greater than or equal to 65, then the drug d is a B.PIM which belongs to the class Beers Criteria.

The formalisation of all the PIMs is detailed in Chapter 4.1.7, which includes the definition of classes, properties and formalised rules and the taxonomy of the underlying ontology.

3.1.1.2 Alternative drugs

The alternative drugs consist of drugs that can be switched with drugs classified as PIMs in order to minimise the risks for the patient. To formalise the alternative drugs, rules were defined according to specific criteria for each drug. For the definition of alternative drugs, a taxonomy was defined considering the drug hierarchy. For example, Figure 3.3 illustrates the main class Alternative drugs, which is composed of the drug class Alt category X and the two alternative drug classes Alternative 1 and Alternative 2.

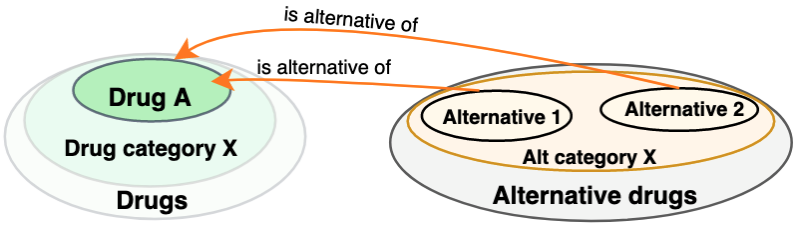


Figure 3.3: Alternative drug example

These two drugs are considered alternatives for Drug A, linked by the element is alternative of. Therefore the following rule was defined to state this condition.

$$\forall_{p,d,a,pr} Patient(p) \wedge Drug(d) \wedge DrugA(d) \wedge Prescription(pr) \exists_{altD} Alt\_category\_X(altD) :$$
$$hasPrescription(p, pr) \wedge hasDrug(pr, d) \wedge hasAge(p, v) \wedge v \geq 65 \Rightarrow isAlternativeoF(altD, d)$$

The rule can be read as follows: for arbitrary patient p, drug d and prescriptions pr, if the following statements are satisfied:

- d is a Drug of the class Drug A
- patient p has the prescription pr
- prescription pr contains drug d
- patient p has age value v greater than or equal to 65, and
- there exists an alternative category X $altD$ for the drug d.

Then $altD$ is an alternative for drug d.

3.1.1.3 Drug parameter formalisation for rescheduling

The rescheduling process uses some drug parameters such as the TMAX to define the constraints. Therefore, these parameters are included in the Beers Criteria ontology for each drug by assigning elements and values. Figure 3.4 illustrates an example where the element hasTmax is used to associate Drug A with the integer value 60, indicating that it takes 60 minutes for the drug to reach its maximum concentration in the blood following administration to a patient.

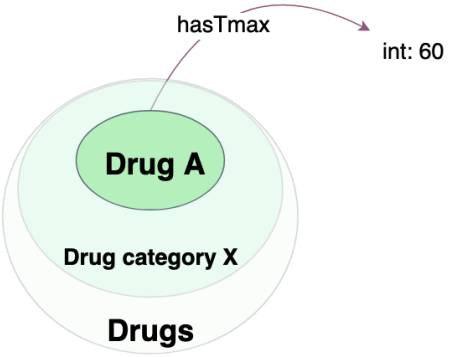


Figure 3.4: Adding Cmax value to class drug

The TMAX is used as a parameter by the rescheduling solver to define, if possible, a safe distance between drugs that have an undesired interaction. Hence, the rescheduling solver aims to maximise the distance between the TMAX values and consequently minimise the interaction.

The main goal of the solver is to minimise the interactions between drugs that have an undesired side effect. Hence, the rescheduling solver aim is to maximise the distance between the TMAX values and consequently minimise the interaction. Moreover, other parameters such as fixed administration time, hospital routines and patient preferences are also considered by the solver.

3.1.2 Inference engines

As mentioned earlier, the proposed CDSS inference engine is composed of three core elements: the Beers Criteria reasoner, the Drug alternative Solver and the Rescheduling Solver (cf. Figure 3.1). Within the engine, the patient data (input) is integrated with the knowledge available in the ontology to execute the reasoning required to detect existing PIMs, to find alternative drugs (if available), and to suggest alternative prescriptions (without PIMs or with a minimal level of interaction between PIMs respectively).

Each element performs a specific task. The Beers Criteria reasoner aims to detect PIMs, the Drug alternative Solver aims to find alternative drugs that can be switched with drugs classified as PIMs and finally, the Rescheduling Solver aims to minimise the interaction between drugs. The order of execution of the framework's sequence can be modified according to specific circumstances, similar to how the process can be halted at any point in any of the inference engines if a solution is discovered or if a solution is deemed impossible. In other words, the framework's execution sequence is flexible and adaptable to changing conditions.

3.1.2.1 Beers Criteria reasoner

The Beers Criteria reasoner aims to identify PIMs. After patient records and prescriptions are integrated into the ontology, the reasoner infers the presence of PIMs in the given patient's prescription. Once the ontology reasoning is complete, the resulting output is further analysed to verify whether any PIMs were found or not. If no PIMs were detected, the process concludes by sending a message informing that it is a valid prescription. However, if there is an interaction between

any of the prescribed drugs, the drug alternative solver is triggered to explore the possibility of finding substitute drugs for the detected PIMs.

3.1.2.2 Drug alternative solver

The purpose of the Drug Alternative Solver is to address drug interactions by identifying alternatives that can replace the prescribed drugs. This means that if there is a drug classified as PIMs, then the solver tries to find an alternative drug that does not have interaction with another prescribed drug. SMT solvers are considered very efficient in solving constraint problems such as drug interactions. They aim to automate and check the satisfiability of the defined constraints that have to be satisfied simultaneously. All the drug constraints can be easily expressed in the SMT model. For example, in suggesting alternative drugs, the solver checks if drugs that are considered alternatives do not have interaction with other prescribed drugs. Therefore, the solver aims to find a prescription without interaction among drugs, taking into account the alternative drugs.

Patient data, prescription, interactions and alternative drugs are inserted into the SMT model [40]. The solver considers a result satisfiable if there are no drugs classified as PIMs or drug interactions. In case the solver can not find a satisfiable result be satisfied, the conflict drugs (due to interaction or PIMs)are identified and extracted. These unsatisfiable drugs are then forwarded to the rescheduling inference module for further analysis and processing.

3.1.2.3 Rescheduling Solver

SMT solvers are considered very efficient in solving optimisation problems. In addition to determining satisfiability, optimisation objectives such as maximisation and minimisation can be defined. Hence, the solver aims to find a solution that satisfies all the problem's constraints and improves the objective function. The objective of the rescheduling inference engine is to adjust the time at which interacting drugs are administered, taking into account the maximum concentration of the drug (CMAX) in order to reduce the likelihood (and to some extent intensity) of interactions. Besides the CMAX parameter, additional constraints are also taken into account, for example, rules were defined based on hospital routines to determine the drug schedule constraints, such as meal times. The solver aims to adjust the administration times to increase the distance between drug CMAX points, thereby minimising interaction likelihood.

3.2 Clinical Decision Support System (CDSS) workflow

The CDSS workflow consists of a sequence of activities integrating the framework elements as illustrated in Figure 3.5. There are two main swimlanes, one to indicate the Input/Output and another to capture the clinical decision support system. The Input/Output swimlane corresponds to the activities carried out outside the CDSS, such as sending requests and receiving results. The CDSS swimlane contains the knowledge base swimlane (where the Beers Criteria ontology is defined) and the inference engines swimlane, which makes use of the Beers Criteria reasoner and the other solvers.

The process begins when the Input/Output swimlane sends a request to the knowledge base, which contains information about the patient and their prescription. These data are then integrated with the ontology and processed using the ontology reasoner in the Beers Criteria reasoner swimlane, which detects drugs categorised as PIMs and determines the type of interaction. Following this, the Beers Criteria reasoner verifies if any interactions were identified. If no interactions are found, a message is sent to the Input/Output to indicate no interactions, and the process ends. However, if interactions are found, a message is sent to the Input/Output with the drug interaction list and type. Moreover, the knowledge base is queried for potential alternative drugs. The query results are sent to the drug alternative solver swimlane, where it is checked if any alternatives are available. The prescription is sent to the rescheduling solver swimlane if there are no alternative drugs. In contrast, if alternative drugs are available, a message with the alternative drugs is sent to the Input/Output, and the prescription,

interactions, and alternative drugs are inserted into the SMT solver. The alternative inference is then executed to assess if the alternative drugs meet all the constraints, such as interaction with other prescribed drugs, in order to address all the PIMs issues. The solver then checks whether a prescription without PIMs can be identified. If prescriptions are found, they are sent to the Input/Output swimlane. However, if a prescription cannot be found, it is forwarded to the rescheduling solver.

The rescheduling swimlane receives a prescription and selects drugs that can be rescheduled. Only drugs that interact with other drugs are considered, as drugs which have interaction with other parameters such as age, gender, or disease

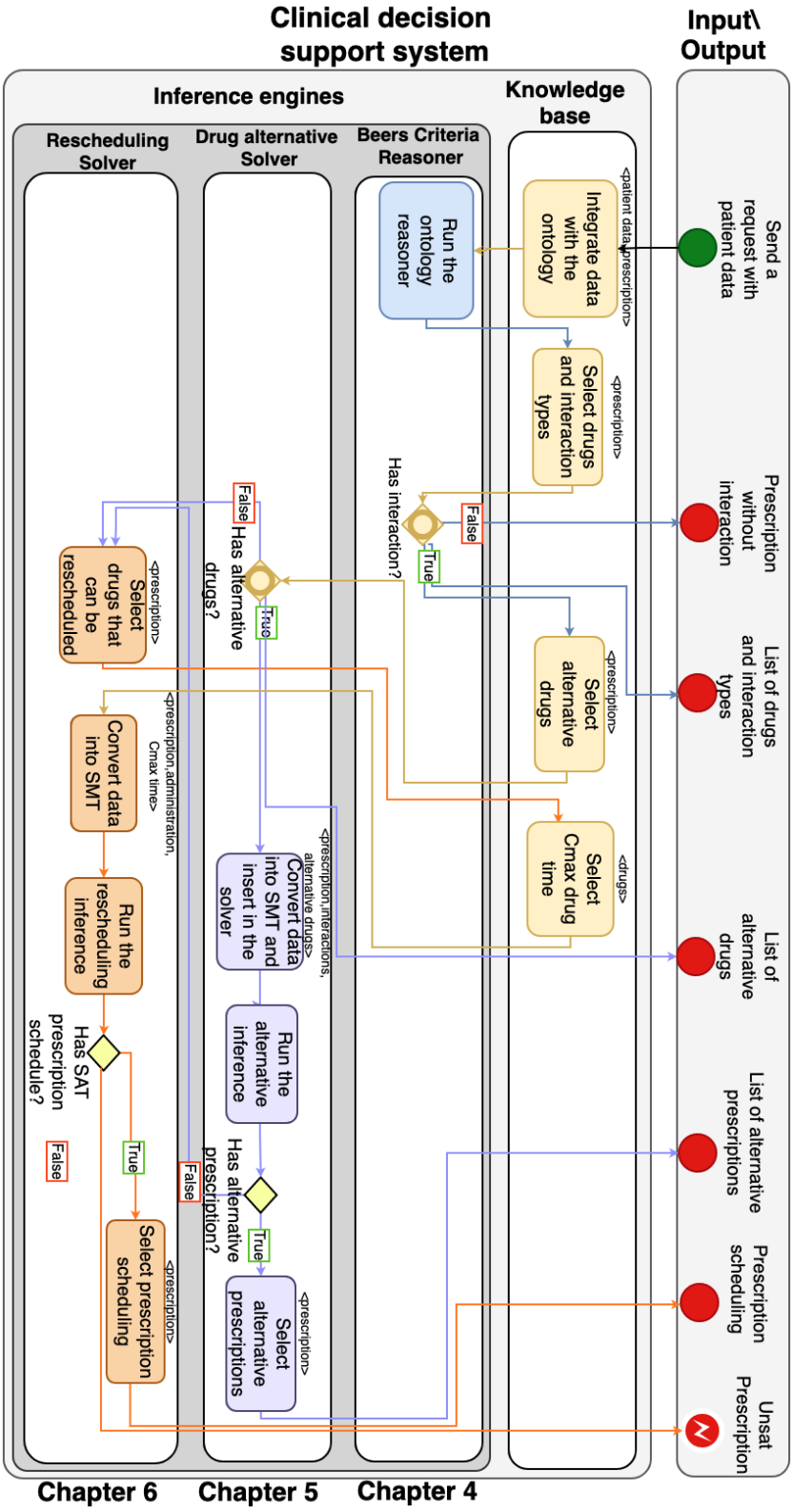


Figure 3.5: CDSS workflow

do not benefit from rescheduling. Once the drugs are selected, the swimlane retrieves the CMAX value from the knowledge base. The drugs, administration time, CMAX value, and additional constraints, such as fixed administration time or meal times, are then converted into an SMT model. Next, the rescheduling inference is executed to maximise the distance between drugs' TMAX. If a valid rescheduling model is found, it is selected from the inference result and sent to the Input/Output swimlane. However, if a rescheduling model cannot be found, a message indicating that the prescription is Unsat is sent to the Input/Output swimlane. To clarify the CDSS outputs, Table 3.1 lists them together with their meaning.

Output	**Description**
Prescription without interaction	No constraints were found, the prescription is satisfiable
Drug interaction list	List of drugs and their interactions
Drug alternative list	List of drugs and their alternatives
Alternative prescriptions	Prescriptions listing alternative drugs
Drugs scheduling	Drugs and administration times
Prescription Unsat [1]	When the inference engine was not able to solve drug interaction constraints

Table 3.1: Outputs from the CDSS framework

The outputs generated by the CDSS are examples of messages that can be transmitted to external sources. In addition to the messages that have been predefined, the CDSS has the capability to extract supplementary information from its knowledge base as needed. The strength of recommendation, the quality of evidence and the side effects of the interactions are examples of information that could be extracted. This information can be customised when selected in the knowledge base for the specific needs, avoiding alert fatigue.

3.3 Tackling other medical issues

The proposed CDSS was designed to tackle PIMs problems. However, this same CDSS could be adapted to tackle other medical issues. To do this we

would, nonetheless, have to consider other ontologies as these would form the basis to encode knowledge and reasoning for different problems. For example, if we consider an ontology with information on diagnose rules the inference engines could be used to support patient diagnosis. Additionally, (clinical or nursing) guidelines could also be formalised in an ontology to support the treatment definition and/or searching for inconsistencies between guidelines and/or medications used.

The CDSS can be used either with all the elements that compose it or as a stand-alone solution comprising of individual elements of interest. Moreover, for employing the proposed CDSS for other medical issues, some changes might be necessary to fit the requirements of the new medical context. Hence, we propose a framework that can be adapted and expanded to tackle different medical issues.

3.4 Summary

This chapter shows the architecture of the proposed CDSS which combines multiple approaches for solving or minimising drug interaction problems and consists of an ontology (knowledge base) and a set of inference engines used as required. Additionally, we detail the interaction between the CDSS and external input and outputs, and outline how it can be applied to tackle other medical issues.

The knowledge base underlying our CDSS will be presented incrementally and each inference engine will be further detailed in accordance to Figure 3.5. Details of the Beers Criteria ontology and its rules can be found in Chapter 4. Chapter 5 describes how the alternative drug rules were added to the ontology, and the constraints that each alternative has to satisfy to be chosen as a valid alternative for the original patient prescription. Chapter 6 focuses on medication rescheduling making use of drug parameters (such as TMAX) and the SMT solver. The development and evaluation of the framework is covered in later chapters.

CHAPTER FOUR

ONTOLOGY FOR DRUG INTERACTIONS

Before we can develop a Clinical Decision Support System (CDSS) specifically for managing medications for the elderly, we need to equip such a CDSS with an adequate knowledge base over PIMs. This chapter presents and defines an ontology to capture Beers Criteria content. Concretely, we focus and expand on the knowledge acquisition, requirements elicitation and architecture used to build the ontology. Moreover, we define the semantic rules for PIMs, aiming to detect them in patient prescriptions.

4.1 Beers Criteria Ontology

The main source of knowledge of the CDSS is a new ontology which captures the Beers Criteria. This ontology includes all information on interactions, recommendations, side effects and alternative drugs related to PIMs, which will be required before making recommendations concerning prescriptions for the elderly. To facilitate personalised recommendations, the CDSS combines not only the ontology rules but also patient-specific information. This means that the ontology also assists healthcare providers in suggesting additional steps when PIMs are identified.

The Beers Criteria is composed of a list of *Potentially Inappropriate Medication Use in Older Adults*. The Beers Criteria Ontology aims to embrace all those listed drugs

and create rules to detect drug interactions in patient prescriptions by reasoning over the ontology rules and patient data. The construction of the ontology went through a series of steps. The first step concerned knowledge acquisition, in which the Beers Criteria were analysed to identify how the PIMs rules were composed and structured, which variables were used and which additional information could be extracted besides the rules. In the second step, the result of the knowledge acquisition was scrutinised to determine the requirements to build the ontology, for example, how the PIMs should be defined. Finally, all the necessary elements were put together to build the ontology and define the PIMs rules based on the requirements.

In Section 2.1.2.1, we provided an overview explanation of the concepts, definitions and components that make up the ontology. Ontologies usually consist of concepts (classes) organised hierarchically in accordance to their relationships [25]. Taxonomies can be part of an ontology as they categorize terms hierarchically based on generalization relationships. However, different from a taxonomy that only allows a parent-child relationship, an ontology enables the formal and structured representation of concepts as well as reasoning and inference capabilities.

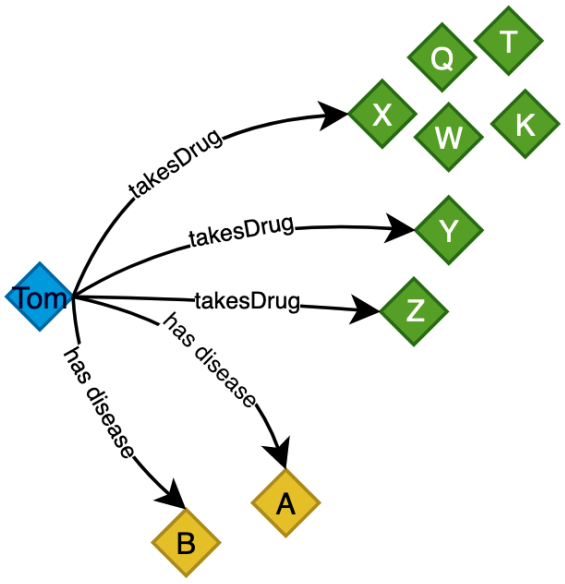


Figure 4.1: Ontology's individuals .

In order to make it easier to comprehend the ontology-building process and the logic used to define its components, we will explain some fundamental concepts relating to individuals, classes, and properties in the upcoming example. Figure 4.1

illustrates different elements of the ontology shown as coloured lozenges. In this example, the individual patient Tom (blue) is represented as taking drugs X, Y, and Z (green), as well as having diseases A and B (yellow). The arrows depicted in the figure symbolise the properties of these individuals, which are binary relationships between elements (e.g., an individual and the drugs he or she takes, an individual and the diseases he or she has, etc). In Figure 4.1, we use different colours to highlight the difference between the kinds of individuals in the ontology, and clustered them according to their domain. In an ontology, these clusters are known as classes, where each class can contain a set of individuals.

Following on from the previous example, Figure 4.2 shows the three different classes of individuals: Patient (blue), Disease (yellow) and Drugs (green).

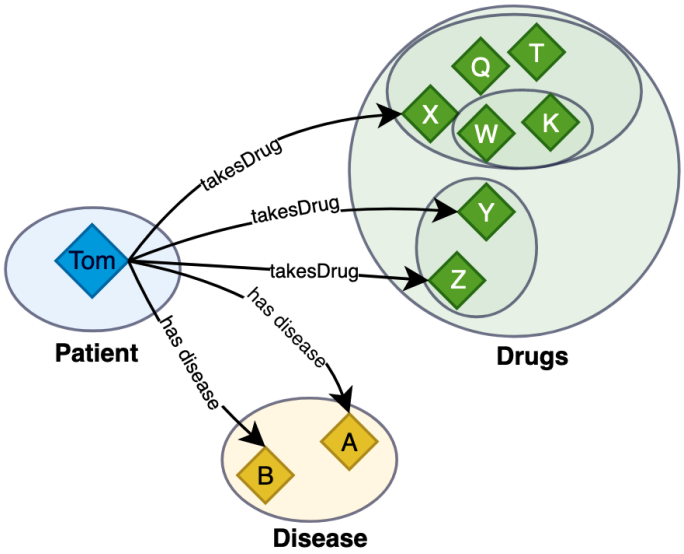


Figure 4.2: Individuals clustered in classes.

Classes can be arranged in a hierarchical model with superclasses and subclasses, also known as a taxonomy. For example, in the class Drugs, there are three subclasses, representing three sets {Y, Z}, {X, Q, T, W, K} and {W, K}. The classes {Y, Z} and {X, Q, T, W, K} are subclasses of the class Drugs, and {W, K} is a subclass of class {X, Q, T, W, K}. Therefore an individual can belong to several classes, as W and K belong to three. These drug classes can represent, for example, drug categories and therefore have to be named, as shown in Figure 4.3 with the drug classes A, B and C. For instance, let us consider that class A represents the individuals {X, Q, T, W, K}, class C {W, K} and class B {Y, Z} .

The definition of the individuals, classes, hierarchy and relations between elements

is a crucial process to build the Beers Criteria ontology knowledge base for the CDSS to detect incorrect prescribing for elderly patients. Based on the ontology elements, for example, inference rules can be defined to determine the detection of PIMs in patient prescriptions.

Figure 4.3 expands the previous example with some basic ontology elements of the Beers Criteria. In addition to the patient's age, there are other reasons a drug may be classified as PIMs, such as, due to disease, clinical conditions or interaction between drugs. In the example shown, a Patient has a Disease and/or has done an Exam. Moreover, the Patient also has a Prescription which has the drugs classes A,C and B. Notice that C is a subclass of A. A and C have an interaction with B and C with a Disease. Moreover, A,C and B have interaction with Patient Age and B also has interaction with a clinical condition detected in a Exam.

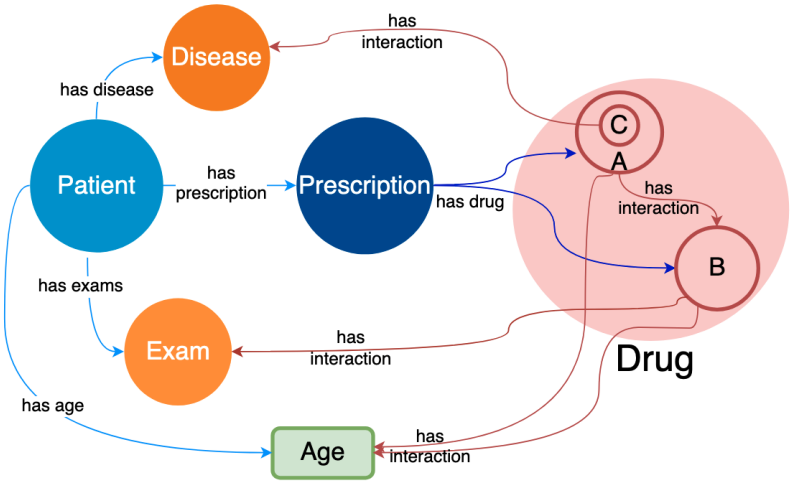


Figure 4.3: Ontology classes and relations.

In Example 4.3, we state that there are interactions involving drugs, age, exam and disease. The ontology asserts if an interaction exists by the definition of inference rules. For example, *DrugA* has interaction with *DrugB*, hence, an inference rule between these two drugs can be defined in the logical expression 4.1.

$$\begin{aligned}\forall_{p,pr,da,db}\ &\text{Patient}(p) \wedge \text{Prescription}(pr) \wedge \text{hasPrescription}(p, pr) \\ &\wedge \text{DrugA}(da) \wedge \text{DrugB}(db) \wedge \text{hasDrug}(pr, da) \\ &\wedge \text{hasDrug}(pr, db) \\ &\Rightarrow \text{hasInteractionWith}(da, db) \end{aligned} \tag{4.1}$$

This logical expression states that for any patient *p*, prescription *pr*, DrugA *da*, and DrugB *db*, if the patient has the prescription, and both instances of DrugA and DrugB are included in the prescription, then ***da*** has an interaction with ***db***. In other words, the expression implies that if a patient has a prescription that includes both an instance of DrugA and an instance of DrugB, then there will be an interaction between the two drugs. It holds true for all possible scenarios where the conditions in the expression are met.

This section provided a brief idea about the ontology aims and the logic of defining the ontology elements. To build the Beers Criteria ontology, several elements still have to be defined to create the necessary rules to detect PIMs. In the next section, we explain how the ontology elements were extracted from the Beers Criteria list.

4.1.1 Knowledge Acquisition

The knowledge acquisition to build this ontology was performed on the version from 2019 of the Beers Criteria [117] list of potentially inappropriate medication use in older adults. Each PIM in the Beers Criteria list was scrutinised to identify how it could be formalised and the elements that must be incorporated into the ontology. For example, classes, object properties and data properties are explained in detail in the following subsection.

The Beers Criteria are divided into five categories, as mentioned earlier in Section 2.1.5.1. Figure 4.4 shows how PIMs are described in the Beers Criteria for the category of drugs potentially inappropriate for older adults, which consists of an interaction between the age (age > 65) and a drug. Recollect that, as previously explained, we are using the acronym B.PIM for drugs listed in the Beers Criteria PIM category. In the example of Figure 4.4, Column 1 corresponds to the list of B.PIM, which can be categorized according to Organ System, Therapeutic Category or Drug(s) in a taxonomic hierarchy, which in this example has three levels. The first level corresponds to the therapeutic drug category Anticholinergics(highlighted in yellow). The second level corresponds to the drug class First-generation antihistamines (highlighted in green). Finally, the third level corresponds to the list of drugs that compose the drug class, such as Brompheniramine and Carbinoxamine. Moreover, this column also details if a drug is potentially inappropriate only in a specific circumstance. For example, Diphenhydramine is potentially inappropriate only when the administration route

[1] is oral(highlighted in pink).

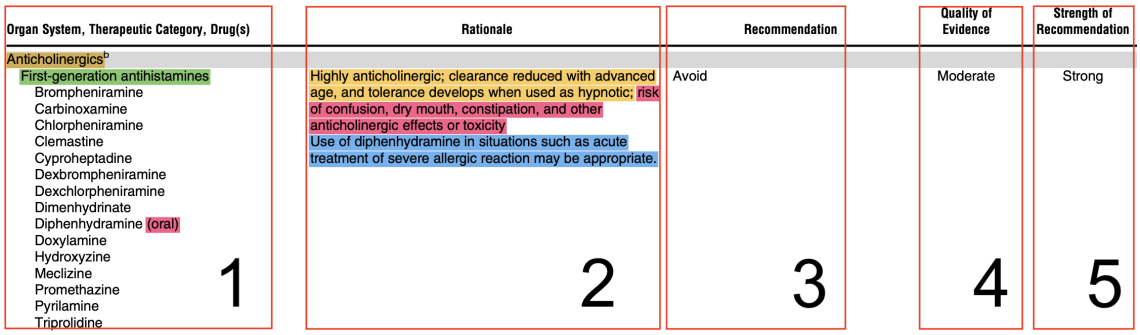


Figure 4.4: The Beers Criteria Table.

Column 2 (Rationale) provides information about why the interaction happens, what side effects the interaction may cause for the patient and particular drug situations. To facilitate the visualisation, we highlight the reason in yellow, the side effects in pink and the individual circumstances in blue. For example, in Figure 4.4, the interaction happens due to highly anticholinergic; clearance reduced with advanced age, and tolerance develops when used as hypnotic. Additional information is given for the clinician to consider, such as side effects, including the risk of confusion, dry mouth, constipation, and other anticholinergic effects or toxicity. Finally, it provides individual circumstance information for using diphenhydramine in situations such as acute treatment of severe allergic reactions may be appropriate. Column 3 provides the Recommendation for the B.PIM, which is avoid in this case, for all listed drugs. Next, Column 4 provides the Quality of evidence of the B.PIM, which can be High, Moderate and Low and finally, Column 5 provides the Strength of Recommendation, which can be either Strong or Weak.

This section clarified how knowledge is obtained from the Beers Criteria. Based on the information gathered from the knowledge acquisition, we will detail the requirements that were defined to consider all the PIMs rules in the next section.

4.1.2 Construction of ontology's requirements

The knowledge acquisition elucidated that the Beers Criteria comprises miscellaneous interaction rules, depending on the interaction variables (e.g. drug dose, age, disease, lab exam results and gender). Therefore, the following competency

[1]The administration route is categorized based on the location where the drug is applied into the body, such as oral or intravenous [82].

requirements were pinpointed to ensure that the proposed ontology considers all the interaction requirements. These requirements were defined after analysing the rules and details of the Beers Criteria interactions.

An ontology for Beers Criteria needs to consider the following requirements:

1. Formalise interactions between:
    1.1 Age-drug
    1.2 Age-drug-drug
    1.3 Age-drug-drug_length_therapy
    1.4 Age-drug-dose
    1.5 Age-drug-gender
    1.6 Age-drug-disease
    1.7 Age-drug-administration route
    1.8 Age-drug-clinical conditions
2. Formalise interaction rules among multiple drugs
3. Provide a list of known side-effects of an interaction belonging to one of the categories listed above.
4. Define multi-language labels

The requirements detail interactions between many variables that can classify drugs as PIMs. A parameter that is always considered is the patient's age, as the Beers Criteria focuses on patients over 65. Although there are interactions for different year ranges, for example, some drugs are considered PIMs only for patients older than 75. Moreover, interactions among drugs usually happen between two drugs. However, for the Beers Criteria, there are cases where drugs are considered PIMs when an interaction happens among three or more drugs from the same therapeutic drug groups. Hence, the interaction rules should consider not only drugs, but therapeutic drug groups and drug classes.

The *daily dose amount* can be a factor for a drug to be classified as potentially inappropriate; therefore, the ontology has to consider interactions between age,

drug and the daily dose amount. Additionally, the *drug length of therapy* is also a parameter for PIMs. This means that some drugs are inappropriate when taken for more than a certain period, for example, some drugs are considered PIMs when prescribed for more than eight weeks. Furthermore, age and drugs may interact with gender, disease, and administration route. Finally, clinical conditions related to kidney function (e.g., the creatinine clearance levels) can affect drug reactions. Hence, laboratory exams have to be considered in the ontology.

The rationale column from the Beers Criteria table provides additional information, such as side effects. Therefore, this information has to be incorporated into the ontology, as it may support clinicians in making decisions. Finally, the last requirement is about multi-language labels. It is a requirement not directly related to the Beers Criteria list. Even though the Beers Criteria are published in English, multi-language labelling allows the ontology to be used in several languages and hence, to be incorporated in many different countries. For example, the EMR used to perform the evaluation is in Portuguese; hence, for the ontology to recognise the Portuguese labels (e.g., drug names), a multi-language label is necessary.

4.1.3 Requirement elicitation

The definition of the ontology's requirements provides the necessary specifications the ontology must hold. Moreover, the requirements provide insights about elements the ontology will have to capture in order to define the PIMs rules. However, the definition of the PIMs in the Beers Criteria table are not defined in formal logic. Therefore, to establish requirement elements and relations for building the ontology, the atomic data semantic triple was employed, consisting of a subject, predicate and object. Semantic triples can be used in natural language to extract and represent information from unstructured text. Figure 4.5 shows an example of a *patient takes a medication* in a semantic triple. To formalise this example, the **subject** was defined as Patient, the **predicate** as takes and **object** as Medication.

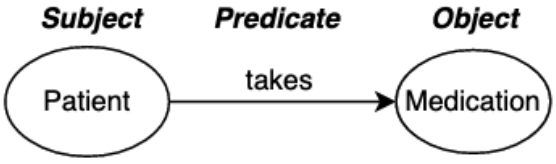


Figure 4.5: Basic semantic triple model.

Figure 4.6 illustrates several semantic triples in which a Patient **takes** Drug A and **suffers** from Parkinson's. Drug A **belongs to** the Object Drug and **has the side effect** Confusion, which **belongs to** the Object Side Effect. Moreover, Parkinson's **has as a treatment indication** Drug A and **belongs to** the object Disease.

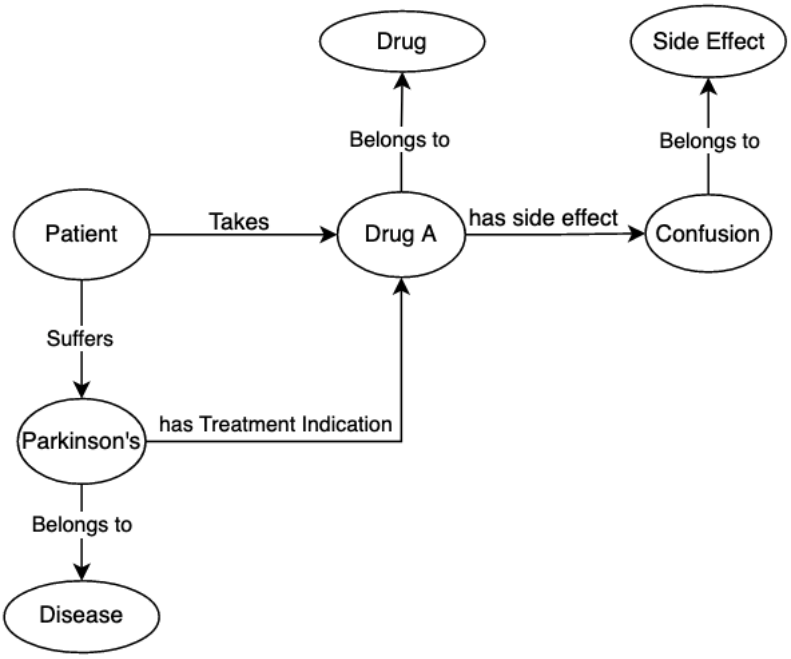


Figure 4.6: Semantic triple model multiple relations.

Through the utilisation of semantic triples it was possible to identify a range of distinct classes and properties. Furthermore, semantic triples facilitated the identification of the taxonomy classes. These elements are the foundation for the development of the Beers Criteria ontology.

The following sections provide an in-depth explanation of the components that have been identified for the development of the Beers Criteria ontology. Throughout these sections, each identified element will be discussed, and its meaning will be explained in detail in order to provide a clear understanding of the ontology and how it is composed.

4.1.4 Ontology elements

The development of the Beers Criteria ontology involved establishing a collection of classes and subclasses, as well as object properties and data properties. The forthcoming sections will detail each ontology component, outlining its specific

features and characteristics to thoroughly understand its construction and role in the ontology.

4.1.4.1 Classes

The main building blocks of an ontology are classes. Classes are groups or collections of objects that assemble common characteristics, organised in the form of a hierarchy tree. A set of classes and subclasses build the foundation for the ontology taxonomy based on Beers Criteria elements. In the list below, we can find the description of the main ontology classes.

- **Beers Criteria**: corresponds to the main PIMs class. This class is subdivided into five Beers Criteria groups and further into several subclasses where each PIMs rule is defined.
- **Quality of Evidence**: establishes the level of the PIMs evidence, which can be High, Low or Moderate.
- **Strength of Recommendation**: establishes the level of the PIMs recommendation, which can be Weak or Strong.
- **Side effect**: represents the possible adverse effects of a PIMs.
- **Drug Category**: defined the drug hierarchy according to the Beers Criteria which is consistent with for example the AHFS Pharmacologic Therapeutic Classification[2].
- **Drug**: establishes the main class of all drugs that constitute the ontology.
- **Disease**: represents the diseases that are related to the Beers Criteria.
- **Administration Route**: represents all kinds of routes to administer drugs, such as oral, injection or nasal.
- **Exam**: represents the exams that are related to the Beers Criteria.
- **Gender**: defines the Beers Criteria genders.
- **Patient**: represent the patients instances.

[2]The AHFS organises drugs based on their pharmacological and therapeutic properties.

- **Release Drug**: represents how the drug can be released, whether immediate or Short-acting.

As previously mentioned, classes are organised in a hierarchical taxonomy. For example, Figure 4.7 shows a sample[3] of how it represents the taxonomy for drugs belonging to the class Central nervous system active drugs. This class has two subclasses Anti epileptic and Benszodiazepines with their respective drug classes. Moreover, the drug classes cloBAZam and clonezaPAM are in an intersection zone, which means that they belong to both subclasses (Anti epileptic and Benszodiazepines).

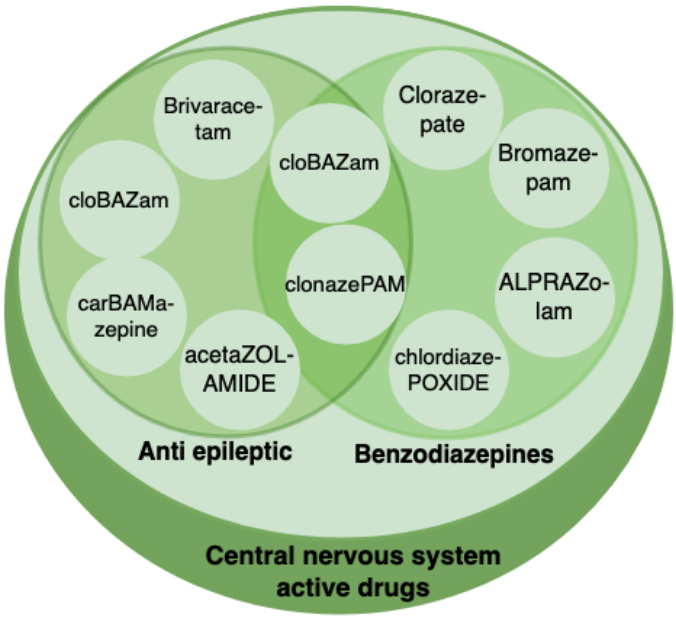


Figure 4.7: Visualisation of the drug categories hierarchy tree for central nervous system active drugs.

Additionally to a class hierarchy, it is possible to define some parameters that belong to each class. For example, Figure 4.8 defines the Quality of evidence and the Strength of recommendation of the Interaction class which is a subclass of B.PIM which in turn is a subclass of Beers Criteria. In this example, the Interaction class is linked to the Quality of evidence subclass Low by the object property has quality of evidence and to the Strength of recommendation subclass Strong by the object property has strength of recommendation.

[3]P.S.: In the Beers Criteria, it is important to note that there are additional drugs that belong to the Central nervous system active drugs category that are not depicted in the figure.

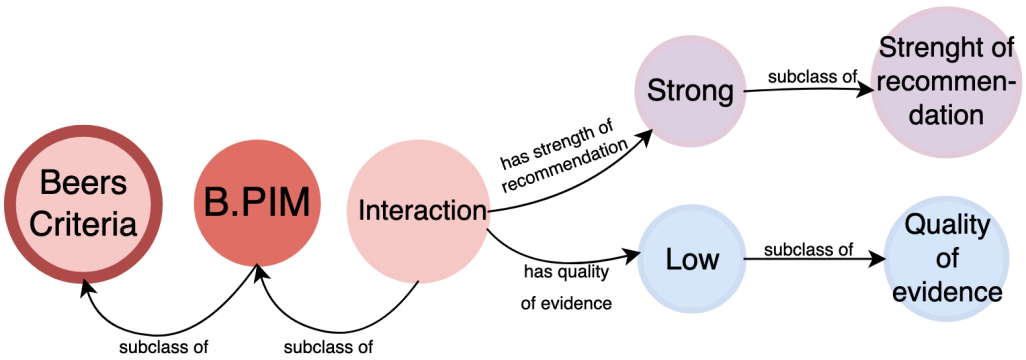


Figure 4.8: Class parameters.

Another relevant feature that has to be defined is the *Disjoint Classes* rule. The ontology allows an individual to belong to several classes or a class to be a subclass of more than one superclass. In an ontology, by default anything that is not formally prohibited is considered possible. However, in some situations, an individual or class cannot belong to another class or superclass. For example, an individual of an administration route Nasal cannot simultaneously be an individual of the administration route Injection. Therefore, a disjoint class rule has to be imposed in these cases, not allowing an individual to belong to multiple classes. Figure 4.9 demonstrates the administration route class and the subclasses Nasal and Injection that have a property Disjoint with between them. For an arbitrary individual x, this property can be expressed in logical terms declared in the Formula (4.2).

$$(\neg Nasal(x) \wedge Injection(x)) \vee (Nasal(x) \wedge \neg Injection(x)) \tag{4.2}$$

According to the statement, for any value of x, x is not $Nasal$, and x is an $Injection$ or vice versa. Therefore, the ontology would be inconsistent if an individual x would belong to both classes (Nasal and Injection). The same inconsistency would arise if there would be a class InjectionNasal simultaneously a subclass of Nasal and Injection.

4.1.4.2 Data properties

Data properties are the relation between an instance and literal datatype values, such as integer, Boolean, varchar or date. For example, to define a drug dose, the

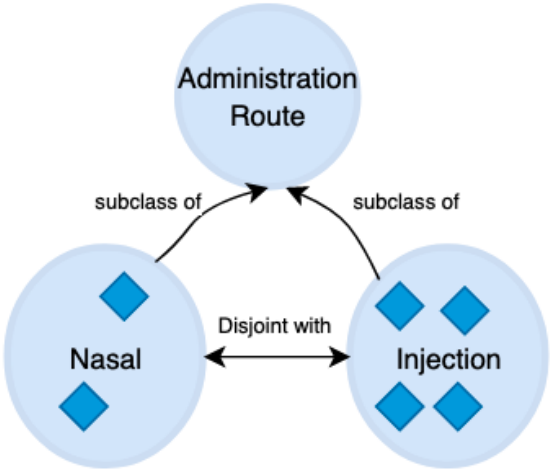


Figure 4.9: Disjoint Classes.

data property hasDailydoseValue is used in a drug instance, where a float number is informed on the property. The items below explain each data property defined in the ontology based on the requirements to build the Beers Criteria rules.

- **hasDailydoseValue**: is used to record the total drug dose per day. Its domain is Drug and range is float;
- **hasDate**: specifies the exam and prescription date. Its domain is Prescription and Exam, and range is date;
- **hasExamValue**: is used to record an exam result. Its domain is Exam, and range is float;
- **hasLenghtDrugTherapy**: identifies how long the patient takes a specific drug during hospitalisation.Its domain is Drug, and range is int;
- **hasPatientAgeValue**: is used to record the patient's age. Its domain is Patient, and range is int;
- **hasDrugType**: tells the drug type, which can be composed or single. A composed drug has more than one active ingredient.Its domain is Drug, and range is char;
- **isCriticalPatient**: informs the patient criticality level.Its domain is Patient, and range is char;
- **isFirstLineDrug**: reports if a drug is regarded as the first line for the treatment. Its domain is Drug, and range is char.

For each data property, it is possible to define the domain and the range. For example, for the data property *hasDailydoseValue*, the domain is the class Drug, with means that only drug classes are allowed to be the subject of this data property. Moreover, the range defines the object of the data property, which means the data type that can be linked, for example, float, string or date.

4.1.4.3 Object properties

Object properties allow for a relationship between two individuals. In order to establish this relationship, it is necessary to have a subject, a predicate, and an object, as illustrated in Figure 4.5. For example, if we want to establish a relationship between a subject prescription and an object drug, we would use the predicate hasDrug. The following list provides a detailed explanation of each object property defined in the ontology.

- **hasDisease**: is used for to store the patient diseases;
- **hasDrug**: is used to associate the prescription with the prescribed drugs;
- **isDrugOf**: is the inverse property of hasDrug. It is used to link prescribed drugs with a prescription;
- **hasExam**: informs the patient exams;
- **hasGender**: is used to record the patient gender;
- **hasPrescription**: is used to associate the patient to prescriptions;
- **hasQualityofEvidence**: informs the quality of evidence for a particular interaction;
- **hasRoute**: is used to record the administration route of a specific drug;
- **hasStrengthofRecommendation**: informs the strength of recommendation for a specific interaction;
- **hasTreatmentIndication**: is used to store the treatment indication of a particular drug;
- **hasInteractionWith**: identifies the drug interactions of a specific drug;
- **toRelease**: is used to record the drug administration release schema.

Similar to the data property, it is possible to define the domain and range of an object property. However, the domain and the range are usually classes. For example, Figure 4.10 shows that the object property hasDisease has the class Patient as the domain and the class Disease as the codomain/range.

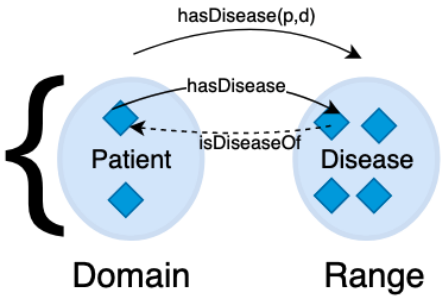


Figure 4.10: Object property parameters.

The object property domain can be defined as declare in Formula 4.3.

$$\forall x \exists y : hasDisease(x, y) \rightarrow Patient(x) \tag{4.3}$$

The statement can be read as: "For all x, if there exists a y such that x has disease y, then x is a *Patient*" asserting that if there is at least one individual y that is associated with an individual x through the relationship *hasDisease*, then x must be a patient.

The object property range can be defined as declare in Formula 4.4.

$$\forall x \forall y : hasDisease(x, y) \rightarrow Disease(y) \tag{4.4}$$

The statement means that: "For all x and y, if x has disease y, then y is a *Disease*", asserting that if an individual x is in a relationship *hasDisease* with an individual y, then y is a *Disease*.

Another parameter that can be defined for an object property is the Inverse Of, which means it operates in the opposite direction. For instance, in Figure 4.10, we have the objects hasDisease in the arrow from the class Patient to the class Disease and its Inverse of property in the dotted arrow isDiseaseOf. This means that when the object hasDisease is instantiated, the object isDiseaseOf will also be instantiated for the same classes in the opposite direction.

4.1.4.4 Annotation property

Ontology annotation properties are used to provide additional information about entities within an ontology. They are used to annotate or describe the entities and can be used to write a comment or version. It is also possible to create a new type of annotation. For the Beers Criteria ontology, the annotation properties Comment describes the reason why the drug is classified as PIMs. Moreover, the Recommendation details what is suggested in case of PIMs.

The annotation label defines the class names. A class can have more than one label, for example, the drug Acetaminophen is also known as Paracetamol, hence, this drug is defined under two labels. Additionally, it is also possible to determine the label language as displayed in Figure 4.11, which means it can define the drug name labels in English and Portuguese, for instance, as the database where the evaluation of the ontology will be performed is in Portuguese.

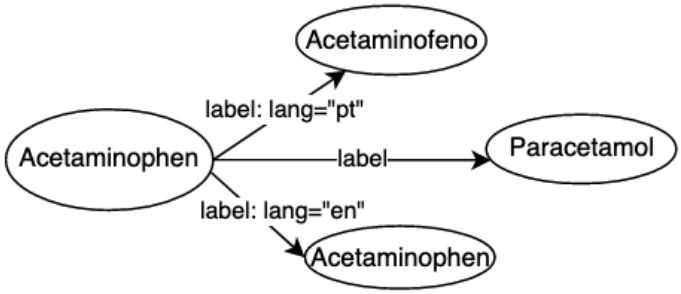


Figure 4.11: Drug label.

4.1.5 Ontology conceptual model

The outcome of the performed analysis to define the ontology elements is shown in Figure 4.12. It represents a conceptual model that reflects the main classes and relationships between elements required to build the ontology based on the Beers Criteria. The model comprises ellipses that correspond to classes, rectangles that denote data properties, and arrows that represent object properties.

To facilitate the understanding of the ontology element's representation, these were grouped into three main categories represented by blue, green, and red ellipses. The elements associated with the patient have been placed within the blue ellipse, while those connected to drugs and prescriptions are located in the green ellipse. Finally, the red ellipse contains elements linked to the Beers Criteria class, allowing for a clear and concise representation of the information.

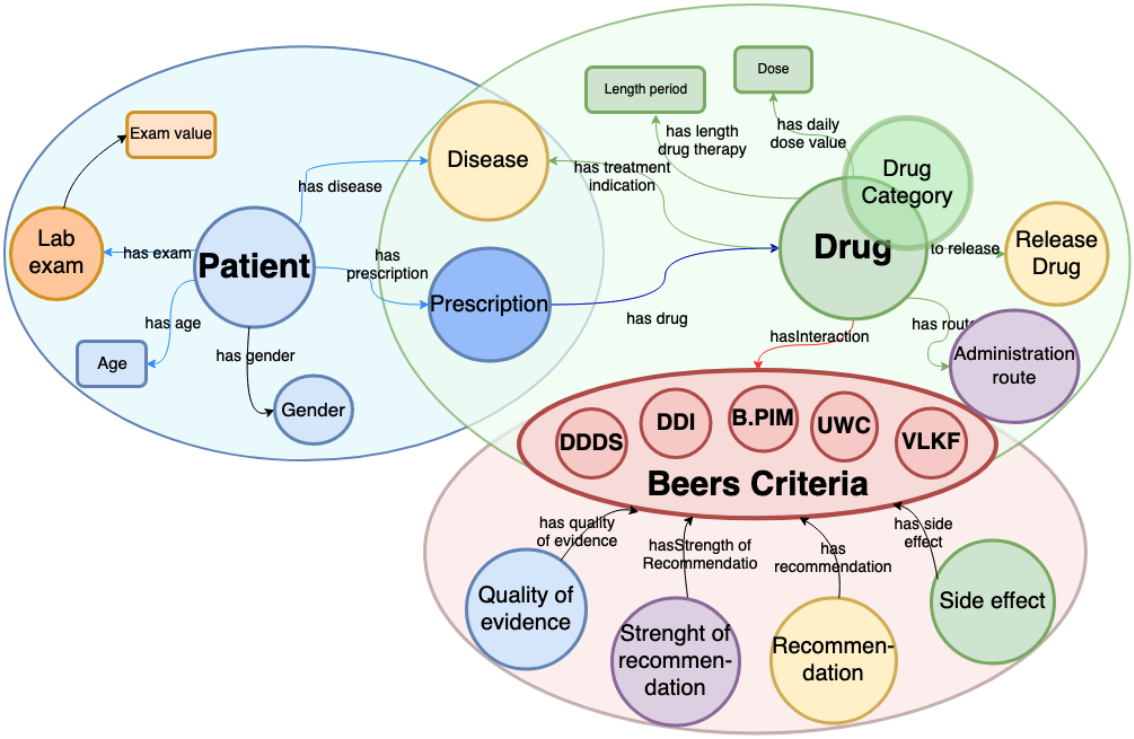


Figure 4.12: Beers conceptual model.

The Patient class is linked with class Gender, and the data property Age through the object properties has gender and has age. Moreover, the classes Disease, Lab exam and Prescription are also linked with the Patient by the object's properties has disease, has exam and has prescription. For the class Lab exam, the values of the exam are defined in the data property Exam value linked by the object property has exam value.

The Prescription class gathers the drugs prescribed for the patient. Hence, this class is linked to the class Drug with the object property has drug. Furthermore, for each prescribed drug, the Dose and Length therapy are defined through the data properties has daily dose value and has length drug therapy. Additionally, the Administration route and Release drug are defined through the object properties to release and has route.

A Drug class belongs to at least one Drug category class, linked by the object property subclass. If a prescribed drug has an interaction with another drug, for example, it will be part of one or more Beers Criteria subclass. In the Beers Criteria class, all the PIMs categories are defined in its subclasses. For each PIMs, the Quality of evidence, Strength of recommendation and Recommendation are

defined by their respective object properties and equally the Side effect when available.

The definition of the ontology elements allows constructing inference rules. These rules aim to assess patient prescription and data in the ontology to assert by the ontology reasoner if there are drugs classified as PIMs. The following sections will detail the definition of those rules and how ontology reasoning is performed.

4.1.6 Ontology Reasoning

A reasoner in an ontology can be used in many contexts: to perform inference rules on ontology assertions to derive additional knowledge or conclusions, as well as to verify the consistency of the ontology analysing if there are no contradictory facts, such as classes that were defined as disjoint sharing an individual or subclass.

An ontology can be analysed from two perspectives. From the Terminology component (TBox) perspective, which constitutes the formal domain of interest by defining classes and properties; and from the Assertion component (ABox) perspective [39]. In TBox, the reasoning happens at the class level, whereas in ABox the reasoning is done directly over individuals [85]. In the following example, we define what constitutes a *Patient* and a *PolypharmacyPrescription*. We define the statements for **Tbox** and assert the individuals in **Abox**.

TBox

$$Patient(x) \Rightarrow Person(x) \wedge \exists_y(hasPrescription(x,y) \wedge Prescription(y))$$
$$PolypharmacyPrescription(y) \Rightarrow Prescription(y) \wedge$$
$$\forall_{i,j=1}^{5},\ i \neq j\ \wedge Drug(z_i) \wedge Drug(z_j) \wedge hasDrug(y,z_i) \wedge hasDrug(y,z_j)$$

Individual definitions

$$Person(John) \wedge Prescription(prescription1) \wedge$$
$$hasPrescription(John, prescription1) \wedge hasDrug(prescription1, Drug1) \wedge$$
$$hasDrug(prescription1, Drug2) \wedge hasDrug(prescription1, Drug3) \wedge$$
$$hasDrug(prescription1, Drug4) \wedge hasDrug(prescription1, Drug5)$$

ABox

$$John \Rightarrow Patient$$
$$prescription1 \Rightarrow PolypharmacyPrescription$$

The *Tbox* assertion states that an individual x is a *Patient* only if x is a *Person* and there exists a *Prescription* y such that x has the prescription y. Additionally, an individual x is a *PolypharmacyPrescription* only if x is a *Prescription* and there exist at least 5 individual drugs y_i, with $1 \leq i \leq 5$ all distinct (i.e., for any $i \neq j$, $y_i \neq y_j$) such that x has drug y_i. These assertions can be used to represent and reason about the relationships between *Patient*, *Prescription*, and *Drug*. From the example above, it is possible to denote in the ABox that *John* is a *Patient* as he is an individual *Person* and has *Prescription*1. His prescription, *Prescription*1, is a *PolypharmacyPrescription* as it contains at least 5 different prescribed drugs.

4.1.7 Beers Criteria Rules

The ontology inference rules aim to detect PIMs and classify them according to categories from the Beers Criteria. There are five main categories classes:*Potentially Inappropriate Medication Due to Drug-Disease or Drug-Syndrome Interactions That May Exacerbate the Disease or Syndrome (DDDS)*, *Potentially Clinically Important Drug-Drug Interactions (DDI)*, *Beers Criteria Potentially Inappropriate Medications (B.PIM)*, *Drugs To Be Used With Caution (UWC)*, and *Medications That Should Be Avoided or Have Their Dosage Reduced With Varying Levels of Kidney Function (VLKF)*. When a drug is classified as PIMs, it can belong to one or more of these categories. In the following sections, we detail each category subclass. All the Beers Criteria rules are defined in the ontology with Semantic Web Rule Language (SWRL), which will be further explained in Chapter 7.

4.1.7.1 Potentially Inappropriate Medication Due to Drug-Disease or Drug-Syndrome Interactions That May Exacerbate the Disease or Syndrome (DDDS)

The DDDS[4] subclass is composed of four main groups of disease classes, Cardiovascular, Central nervous system, Gastrointestinal and Kidney/Urinary tract as shown in Figure 4.13. We also define subclasses for each particular disease or syndrome(e.g. DDDS_Heart failure and DDDS_Syncope). Within each disease

[4]Refers to Table 3 of the Beers Criteria from the 2019 version.

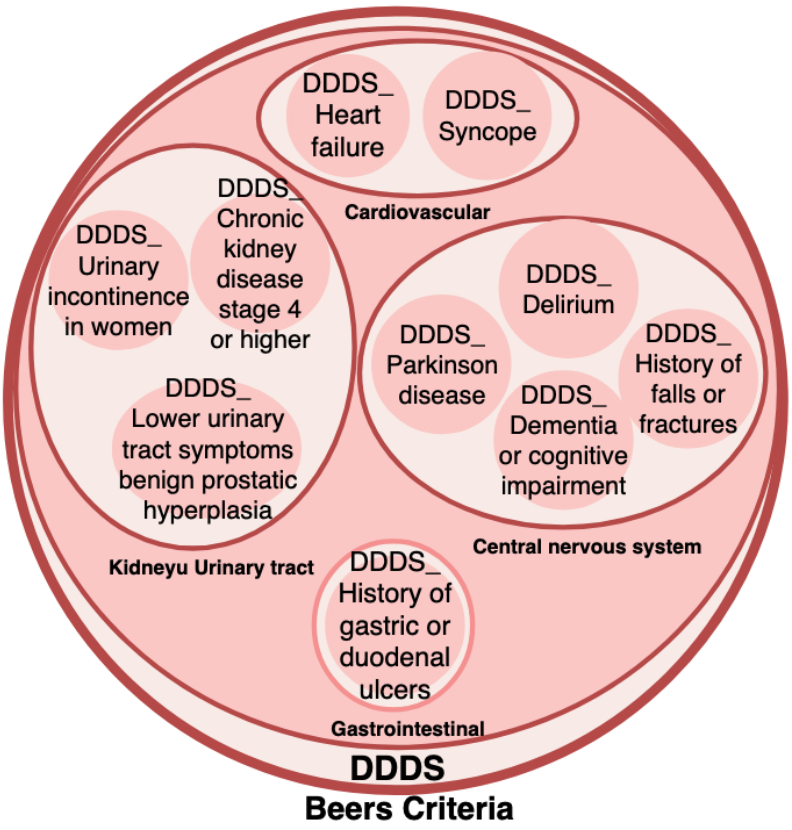


Figure 4.13: Drug-Disease or Drug-Syndrome Interactions classes hierarchy.

or syndrome, additional classes were created for a drug or a drug category class that are PIMs, as illustrated in Figure 4.14. For example, Cilostazol is considered a potentially inappropriate medication for heart failure. Therefore, the DDDS_Cilostazol class was created for class DDDS_Heart_failure.

For each drug or drug category class, a rule was defined to classify a drug as DDDS. The Rule 4.5 is an example of how an inference rule was defined for the drug Cilostazol [5] . Let p be an arbitrary patient, pr be a prescription, d be a drug, and a be an integer.

$$\begin{aligned}(Patient(p) \wedge Prescription(pr) \wedge Cilostazol(d) \wedge Heart\_failure(ti) \wedge \\ a \geq 65 \wedge hasPrescription(p,pr) \wedge hasPatientAgeValue(p,a) \wedge \\ hasTreatmentIndication(p,ti)) \Rightarrow DDDS\_Cilostazol(d)\end{aligned} \tag{4.5}$$

[5]Refers to Table 3 -> Cardiovascular ->Heart failure - of the Beers Criteria from the 2019 version.

The statement says: if patient p has a prescription pr for Cilostazol d, has a diagnosis of heart failure with treatment indication ti, and is aged 65 or older ($a \geq 65$), then Cilostazol d is classified as DDDS_Cilostazol, which is a subclass of DDDS_Heart_failure.

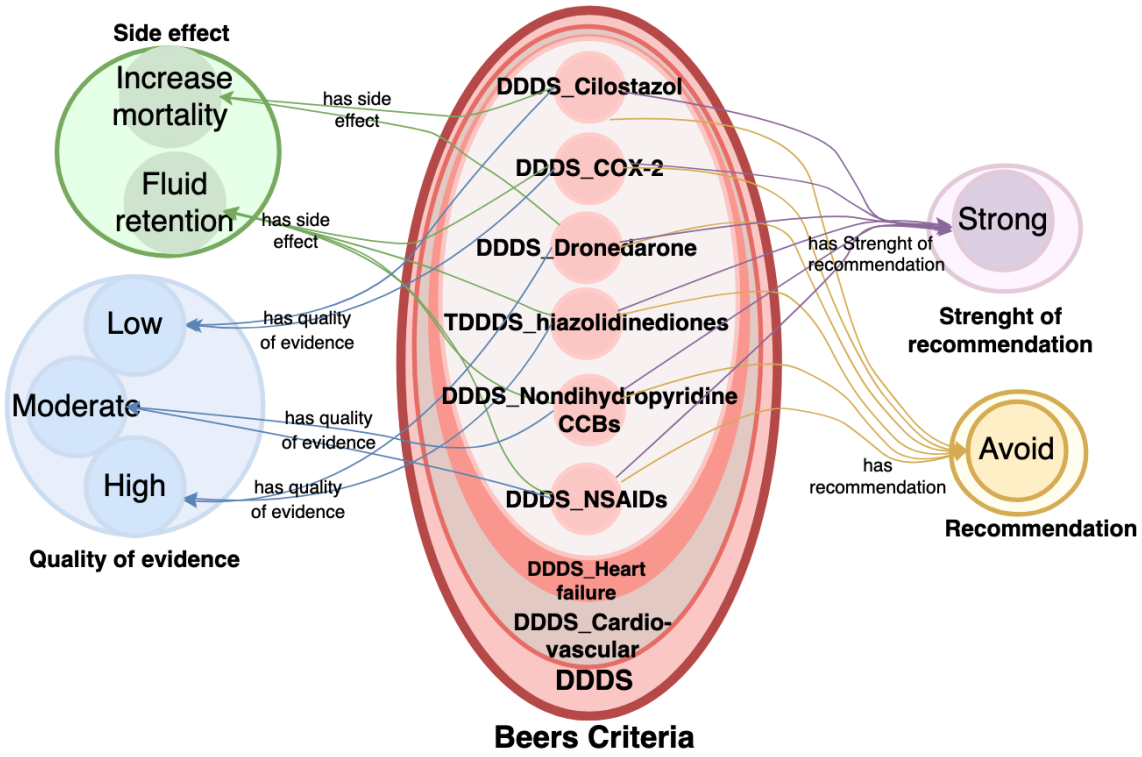


Figure 4.14: Drug-Disease or Drug-Syndrome Interactions Cardiovascular Rule.

In Figure 4.14, drugs and drugs category subclasses of the class DDDS_Heart failure are displayed. The figure also shows additional classes and subclasses related to each DDDS. For example, the class DDDS_Cilostazol is linked to the side effect Increase Mortality by the object property has side effect. Additionally, this drug has quality of evidence Low, has strength of recommendation Strong and has recommendation Avoid. These parameters were formalised as defined in Rule 4.6.

$$\begin{aligned}
\forall_i, DDDS_Cilostazol(i) \rightarrow DDDS_Heart_failure(i) \wedge \\
\exists_{QoE}, hasQualityofEvidence(i, QoE) \wedge Low(Qoe) \wedge \\
\exists_{Se}, hasSideEffect(i, Se) \wedge Increase_mortality(Se) \wedge \\
\exists_{SoR}, hasStrengthofRecommendation(i, SoR) \wedge Strong(SoR) \\
\exists_{SoR}, hasRecommendation(i, r) \wedge Avoid(r)
\end{aligned} \tag{4.6}$$

The rule can be read as: "For all interaction *i*, if *i* is *DDDS_Cilostazol*, then *i* is also *DDDS_Heart_failure*. Moreover *i* has quality of evidence *QoE Low*, side effect *Se Increase_mortality*, strength of recommendation *SoR Strong* and recommendation *r* to *Avoid*.

4.1.7.2 Potentially Clinically Important Drug-Drug Interactions (DDI)

The DDI[6] class is formed of twenty-one subclasses as illustrated in Figure 4.15. The subclasses were named according to the drug or drug category that compose the interaction. For example, the class CNS_Active_Drugs x CNS_Active_Drugs means an interaction between drugs from the same drug category Central Nervous System Active Drugs. In comparison, Warfarin x Amiodarone[7] means an interaction between these two drugs. For this interaction, the inference Rule 4.7 was defined, where we assume that *p* is an arbitrary patient, *pr* is a prescription, d_1 and d_2 are drugs, and *a* is an integer.

$$\begin{aligned}
Patient(p) \wedge Prescription(pr) \wedge Warfarin(d_1) \wedge Amiodarone(d_2) \wedge a \geq 65 \\
\wedge hasPrescription(p, pr) \wedge hasDrug(pr, d_1) \wedge hasDrug(pr, d_2) \\
\wedge hasPatientAgeValue(p, a) \\
\Rightarrow WarfarinXAmiodarone(d_1) \wedge WarfarinXAmiodarone(d_2) \wedge \\
hasInteractionWith(d_1, d_2) \wedge hasInteractionWith(d_2, d_1)
\end{aligned} \tag{4.7}$$

The statement asserts that if patient *p* aged 65 or older has a prescription *pr* and the prescription includes both d_1 and d_2, where d_1 is of type Warfarin and d_2 is of

[6]Refers to Table 5 of the Beers Criteria from the 2019 version.
[7]Refers to Table 5 -> Warfarin x Amiodarone - of the Beers Criteria from the 2019 version.

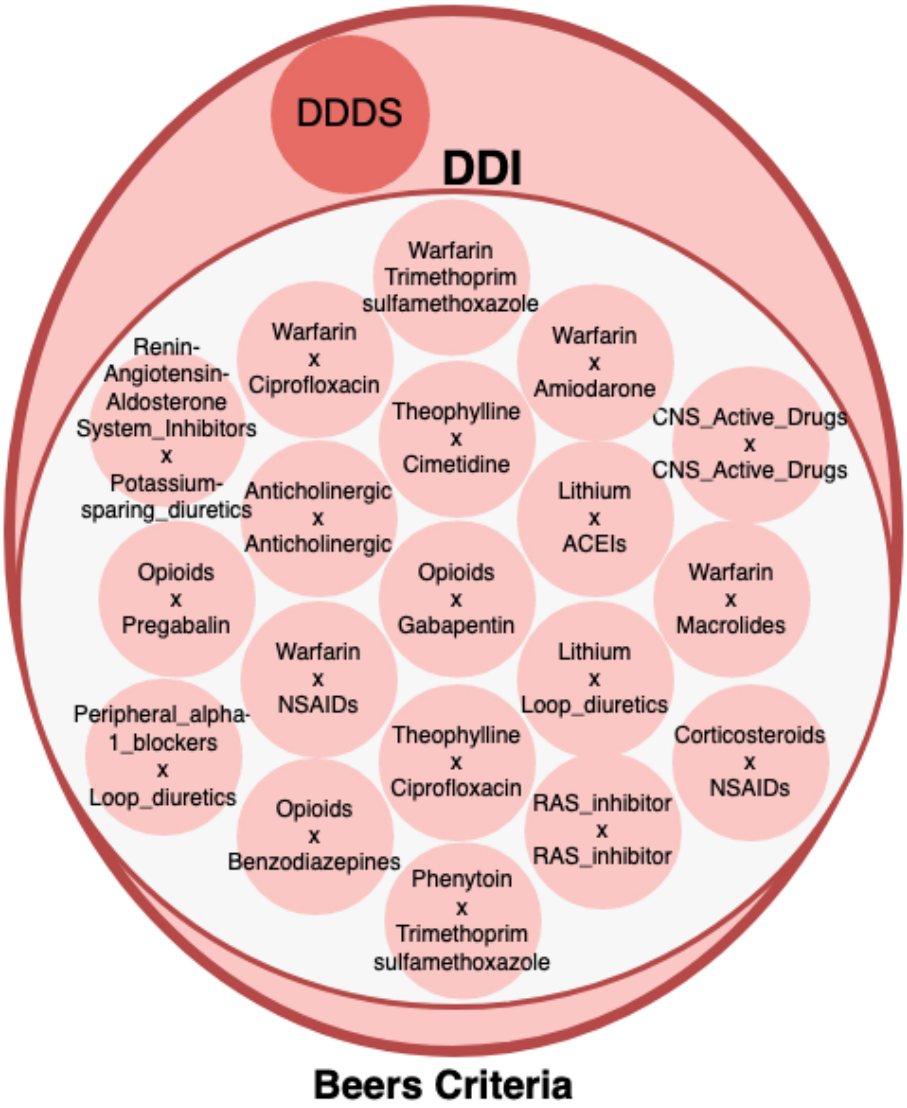


Figure 4.15: Drug-Drug Interactions classes hierarchy.

type Amiodarone, then d_1 and d_2 belong to the class WarfarinXAmiodarone, and there is a bidirectional interaction between these two drugs asserted by the object property hasInteractionWith.

4.1.7.3 Beers Criteria Potentially Inappropriate Medications (B.PIM) and Drugs To Be Used With Caution (UWC)

The B.PIM[8] class is composed of 45 subclasses, and UWC[9] class of 17 subclasses. For all these classes, inference rules were defined. For example, for the category

[8]Refers to Table 2 of the Beers Criteria from the 2019 version.
[9]Refers to Table 4 of the Beers Criteria from the 2019 version.

drug, Antithrombotics[10] the Rule 4.8 was defined.

$$\begin{aligned} &Patient(p) \wedge Prescription(pr) \wedge Dipyridamole(d) \wedge Oral(r) \wedge a \geq 65 \\ &\wedge Short\_acting(sa) \wedge hasPrescription(p, pr) \wedge hasDrug(pr, d) \\ &\wedge hasPatientAgeValue(p, a) \wedge hasRoute(d, r) \wedge toRelease(d, sa) \\ &\Rightarrow PIM\_Antithrombotics(d) \end{aligned} \tag{4.8}$$

The statement says that for a patient p aged over 65 with a prescription pr with the medication Dipyridamole d to be administered orally and short acting, the medication d is potentially inappropriate and classified as PIM_Antithrombotics.

For the classes that belong to the UWC class, rules were also created for each drug. For example, for drug Rivaroxaban[11] was defined the Rule 4.9.

$$\begin{aligned} &Patient(p) \wedge Prescription(pr) \wedge Rivaroxaban(d) \\ &\wedge Venous\_thromboembolism(ti) \wedge a \geq 75 \wedge hasPatientAgeValue(p, a) \\ &\wedge hasTreatmentIndication(p, ti) \Rightarrow UWC\_Rivaroxaban(d) \end{aligned} \tag{4.9}$$

The formula states that if a patient p is prescribed Rivaroxaban for the treatment of venous thromboembolism and the patient is 75 years or older, then Rivaroxaban is classed as UWC_Rivaroxaban(d). As for all drugs classified as inappropriate, for this UWC, the properties were also defined according to Rule 4.10.

$$\begin{aligned} &\forall_i UWC\_Rivaroxaban(i) \rightarrow UWC(i) \wedge \exists QoE \wedge \\ &\exists QoE, hasQualityofEvidence(i, QoE) \wedge Moderate(Qoe) \wedge \\ &\exists Se, hasSideEffect(i, Se) \wedge Gastrointestinal\_bleeding(Se) \wedge \\ &\exists SoR, hasStrengthofRecommendation(i, SoR) \wedge Strong(SoR) \end{aligned} \tag{4.10}$$

[10] Refers to Table 2 -> Antithrombotics -> Dipyridamole of the Beers Criteria from the 2019 version.

[11] Refers to Table 4 -> Dabigatran/Rivaroxaban of the Beers Criteria from the 2019 version.

Interactions that are classified as UWC_Rivaroxaban are also classified as UWC and have $Moderate(Qoe)$ quality of evidence, $Gastrointestinal\_bleeding(Se)$ as a side effect, and $Strong(SoR)$ strength of recommendation.

4.1.7.4 Medications That Should Be Avoided or Have Their Dosage Reduced With Varying Levels of Kidney Function (VLKF)

The VLKF[12] class is composed of five main subclasses: *VLKF Anti-infective*, *VLKF Cardiovascular or hemostasis*, *VLKF Central nervous system and analgesics*, *VLKF Gastrointestinal* and *VLKF Hyperuricemia*. Subclasses and rules were created for each drug or drug category classified as VLKF, totalling twenty-nine subclasses and rules. An example of these rules is defined below. A particularity of *Medications That Should Be Avoided or Have Their Dosage Reduced With Varying Levels of Kidney Function (VLKF)* is that all rules have the *Creatinine_Clearance exam results* due to the expected problems associated to kidney function. Rule 4.11 was defined for VLKF_Fondaparinux[13] . This rule can be read as: if patient p aged over 65 has a prescription prcontaining Fondaparinux d, and has a Creatinine_Clearance e exam value v less than 30, then d is classified as VLKF_Fondaparinux.

$$\begin{aligned} &Patient(p) \wedge Prescription(pr) \wedge Creatinine\_Clearance(e) \\ \wedge &Fondaparinux(d) \wedge hasPrescription(p, pr) \wedge hasExam(p, e) \\ &\wedge hasDrug(pr, d) \wedge hasExamValue(e, v) \wedge v \\ &< 30 \wedge hasPatientAgeValue(p, a) \wedge a \geq 65 \\ &\Rightarrow VLKF\_Fondaparinux(d) \end{aligned} \tag{4.11}$$

The result of our research of the Beers Criteria as documented has led to the main Beers Criteria classes as illustrated in Figure 4.16. After defining the five classes and their rules to detect and classify PIMs, the ontology can already be integrated with patient data. In the next section, we provide examples of how patient data can be integrated into the ontology and analyse the resulting outputs.

More details of the ontology can be seen in Appendix B.

[12]Refers to Table 6 of the Beers Criteria from the 2019 version.

[13]Refers to Table 6 -> Fondaparinux - of the Beers Criteria from the 2019 version.

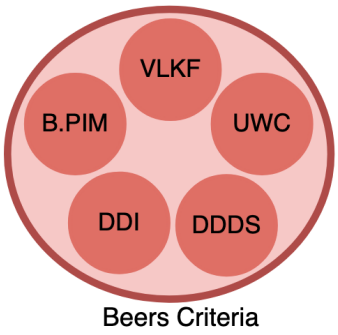


Figure 4.16: Beers Criteria classes

4.1.8 Applying the Beers Criteria ontology

After building the ontology and defining the rules to detect PIMs, it is possible to demonstrate how the ontology would perform over patient data. Therefore, we will simulate a fictional scenario to illustrate how some classes' relationships happen and how the PIMs are classified by the PIMs rules. This simulation is explained in two steps. First, we introduce a scenario with a patient with a disease, prescription, drugs and information on their administration route. Then, we present a scenario with drugs classified as PIMs asserted by the ontology rules.

In Figure 4.17, several individuals were defined, represented as lozenges inside the classes, which are shown as ellipses. The relations between individuals and between classes are given by arrows. An ellipse that is inside another ellipse represents the class's taxonomy of class and subclass.

Inside the class Patient, represented with a light blue ellipse, we find the individual Tom. This individual is linked to the individual M(Male) from the class Gender by the object property hasGender and with the individual P1 from the class Prescription by object property hasPrescription. Moreover, Tom is linked by the object property hasDisease to the individual P1_History_of_falls, and he is 75, given by the link set by the data property hasPatienAgeValue.

The individual prescription P1 is linked with four individual drugs (shown as green lozenges) by the object property hasDrug which all belong to the class Drug. The individual P1_Metoclopramide is a Metoclopramide drug type which belongs to the Prokinectic Agents drug category class. The individualP1_Triazolam belongs to the Benzodiazepines drug category class while P1_Codeine and P1_Morphine both belong to drug category class Opiate Agonists. All of these three drugs

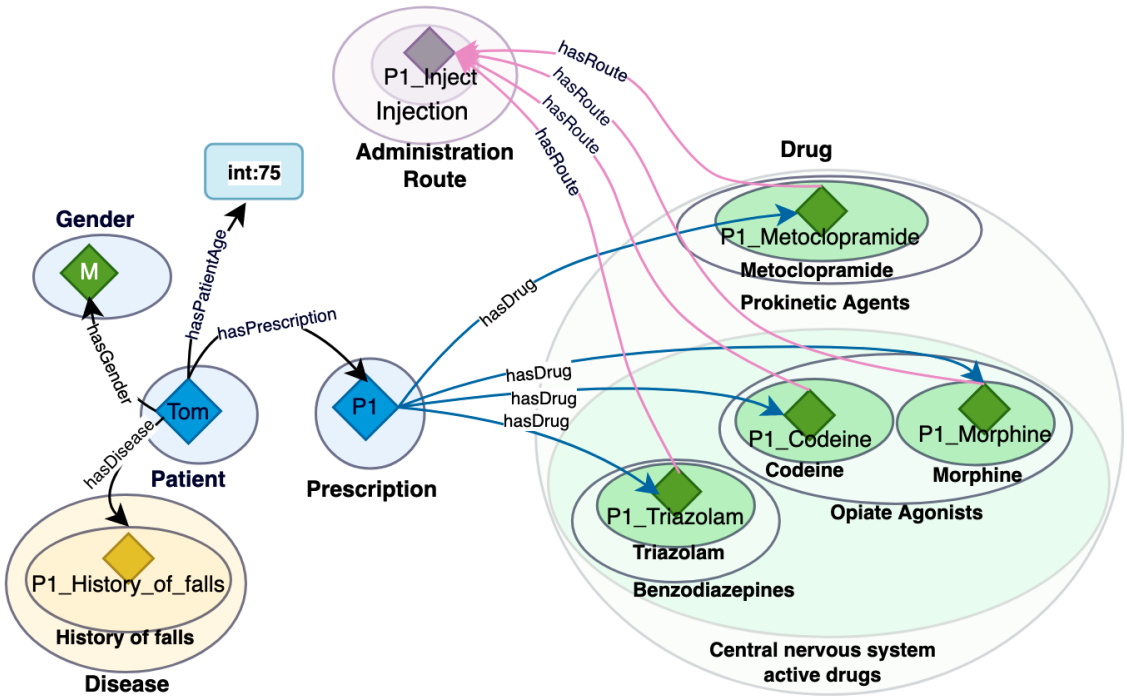


Figure 4.17: Individual Prescription

belong to the drug category class of *central nervous system active drugs*. All the drugs are administered by injection route, seen by the fact that they are linked by the object property hasRoute to the individual P1_Inject, which belongs to the classes Injection and Administration Route.

Summarising the above Tbox scenario shown in Figure. 4.17, patient Tom, 75, male, suffers from a history of falls and has a prescription for four drugs (Metoclopramide, Triazolam, Codeine and Morphine) all to be administered by injection.

To check whether our patient Tom has PIMs in his prescription, which corresponds to the individuals and relationships described in Figure 4.17, we use the ontology inference rules to check. Figure 4.18 illustrates the inference rules outcome (ABox) from the performed rules. As a result of the inference rules, we obtain the Beers Criteria, Recommendation, Strength of recommendation and Quality of evidence classes, with their respective subclasses.

Figure 4.18 shows that for the four prescribed drugs, PIMs could be asserted as shown by the arrows that link the individual drugs with the Beers Classes from Table 4.1.

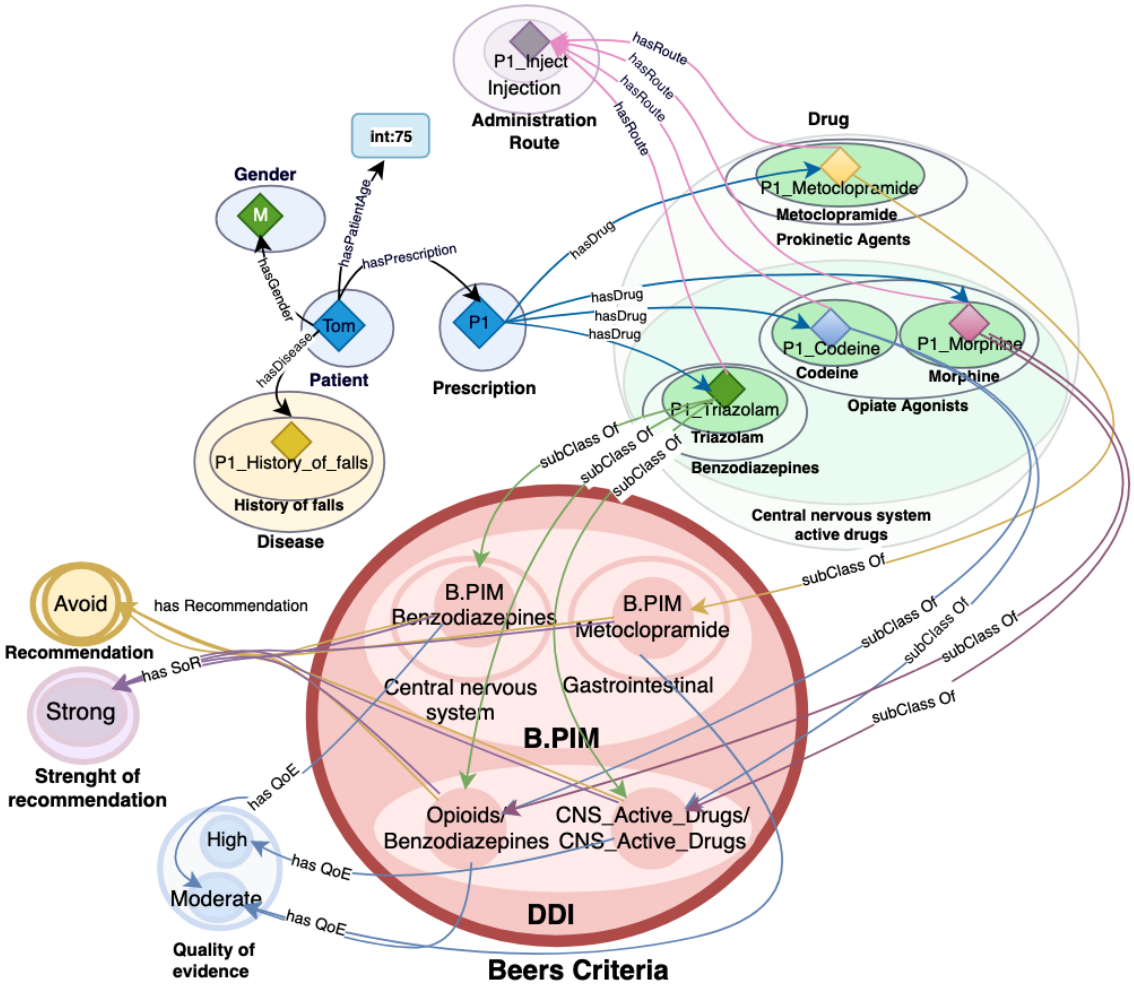


Figure 4.18: Abox Inferred PIMs for the patient Tom

These classes are divided into two main superclasses B.PIM and DDI. Moreover, PIM_Metoclopramide is a subclass of Gastrointestinal while PIM_Benzodiazepines is a subclass of the Central nervous system, both of which belong to class PIM. In the DDI subclasses, OpioidsBenzodiazepines means an interaction between these drug categories, while the CNS_Active_Drugs/CNS_Active_Drug means an interaction between drugs from the same drug category.

The interaction between the individual drugs contained in Tom's prescription is illustrated more clearly in Figure 4.19. The red arrows represent the object property hasInteractionWith, which when given as a double-headed arrow means that the interaction is bidirectional. For example, the individual P1_Triazoalm interacts with P1_Codeine and P1_Morphine, and P1_Codeine interacts with P1_Morphine.

Individual drugs	PIM category	PIM Class
P1_Metoclopramide	B.PIM	PIM_Metoclopramide
P1_Triazolam	DDI	CNS_Active_Drugs/CNS_Active_Drugs
	DDI	Opioids/Benzodiazepines
	B.PIM	PIM_Benzodiazepines
P1_Codeine	DDI	Opioids/Benzodiazepines
	DDI	CNS_Active_Drugs/CNS_Active_Drugs
P1_Morphine	DDI	Opioids/Benzodiazepines
	DDI	CNS_Active_Drugs/CNS_Active_Drugs

Table 4.1: Ontology Abox

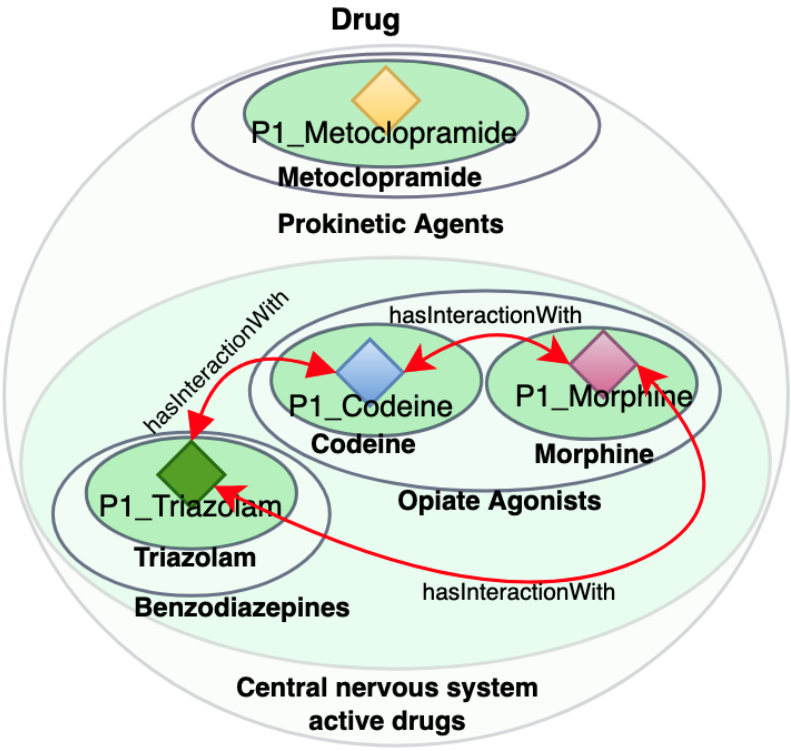


Figure 4.19: Abox Drug interaction

The inappropriate drug assertions for this case emerged from the inference rules that compose the ontology. For each Beers Criteria class drug or drug category, a rule has been defined. Therefore, to understand how the ontology classified these individual drugs as PIMs, we give the details for the inference rules that compose this example. In the following, let p denote a patient, pr a prescription, d a drug or

drug category, and a an integer.

The PIM_Metoclopramide[14] inference Rule 4.12 is defined as follows:

$$\begin{aligned}(&hasPrescription(p,pr) \wedge hasDrug(pr,d) \wedge Metoclopramide(d) \\ \wedge &hasPatientAgeValue(p,a) \wedge a \geq 65) \Rightarrow PIM\_Metoclopramide(d)\end{aligned} \tag{4.12}$$

The logical expression formulated for the PIM_Metoclopramide defines that for an elderly patient p with a prescription pr containing the drug d of type Metoclopramide, drug d is classified as a potentially inappropriate medication (PIM).

Similarly, the PIM_Benzodiazepines[15] inference Rule 4.13 applies if d is a drug of type Benzodiazepines instead:

$$\begin{aligned}(&hasPrescription(p,pr) \wedge hasDrug(pr,d) \wedge Benzodiazepines(d) \\ \wedge &hasPatientAgeValue(p,a) \wedge a \geq 65) \Rightarrow PIM\_Benzodiazepines(d)\end{aligned} \tag{4.13}$$

The drug-drug interaction $DDI\_Opioids/Benzodiazepines$[16] is formulated in the inference Rule 4.14, where we consider two drugs d_1 and d_2 of each type.

$$\begin{aligned}(&Opiate\_Agonists(d_1) \wedge Benzodiazepines(d_2) \wedge hasPrescription(p,pr) \\ \wedge &hasDrug(pr,d_1) \wedge hasDrug(pr,d_2) \wedge hasPatientAgeValue(p,a) \wedge a \geq 65) \\ \Rightarrow &DDI\_Opioids/Benzodiazepines(d_1) \wedge DDI\_Opioids/Benzodiazepines(d_2) \wedge \\ &hasInteractionWith(d_1,d_2) \wedge hasInteractionWith(d_2,d_1)\end{aligned} \tag{4.14}$$

The meaning of the statement is that for elderly patients p with a prescription pr for an Opiate agonist drug d_1 and a Benzodiazepines drug d_2, d_1 and $d2$ are classified as DDI_Opioids/Benzodiazepines when used in combination with each other in patients 65 or above. Additionally, it states that there is a (bidirectional) interaction between these drugs.

[14]Refers to Table 2 -> Gastrointestinal -> Metoclopramide - of the Beers Criteria from the 2019 version.

[15]Refers to Table 2 -> Central nervous system -> Benzodiazepines - of the Beers Criteria from the 2019 version.

[16]Refers to Table 5 -> Opioids/Benzodiazepines - of the Beers Criteria from the 2019 version.

For the following rule, let p be a patient with prescription pr, three different drugs $d_1 \neq d_2 \neq d_3$ of type Central_nervous_system_active_drugs, and $i \neq j \in \{1,2,3\}$. The inference Rule 4.15 - DDI_CNS_Active_Drugs/CNS_Active_Drugs defines the interaction between CNS Active Drugs[17]

$$\begin{aligned} & hasPrescription(p, pr) \wedge hasDrug(pr, d_1) \wedge hasDrug(pr, d_2) \\ & \wedge hasDrug(pr, d_3) \wedge hasPatientAgeValue(p, a) \wedge a \geq 65 \\ \Rightarrow & \forall_{i \neq j \in \{1,2,3\}} DDI\_CNS\_Active\_DrugsCNS\_Active\_Drugs(d_i) \\ & \wedge hasInteractionWith(d_i, d_j) \wedge hasInteractionWith(d_j, d_i) \end{aligned} \tag{4.15}$$

According to this rule, for an elderly patient p with prescription pr containing distinct drugs d_1, d_2 and d_3 of type Central_nervous_ system_active_drugs, then each of the three drugs has a (bidirectional) drug-drug interaction (DDI) with each other as defined in the object property hasInteractionWith and these drugs belong to the PIM category DDI_CNS_Active_Drugs /CNS_Active_Drugs.

Through the integration of patient data with the ontology, this example showcases how we can utilise the ontology to detect PIMs. Additionally, the ontology can be integrated with a Clinical Decision Support System (CDSS) or other ontologies to enhance its usability and facilitate the sharing of knowledge. It also keeps the knowledge base separate and makes it easier to accommodate changes to the knowledge associated with the Beers Criteria (e.g., through newly revised criteria) without impacting on the remaining components and reasoning engines.

4.2 Summary

This chapter elicited the ontology requirements, how it was built and how to classify a drug as PIM. We explained the elements that compose the ontology and how the rules were defined.

Our ontology can be used to detect drug interactions and consequently support clinical decision-making. Nevertheless, this is just one step of the decision process. Further recommendation steps include preferred alternatives with equal/similar

[17] Refers to Table 5 -> Antidepressants (TCAs, SSRIs, and SNRIs) Antipsychotics Antiepileptics Benzodiazepines and nonbenzodiazepine, benzodiazepine receptor agonist hypnotics (ie, "Z-drugs") Opioids / Any combination of three or more of these CNS-active drugs - of the Beers Criteria from the 2019 version.

therapeutic value, revised timed schedules for medications to avoid interactions when no alternative is present, as well as revision of medications to check whether there is still a therapeutic need for certain medications over time. The example introduced in this chapter to identify PIMs will be revisited in later chapters.

Details on how the ontology was developed are demonstrated in Chapter 7. Furthermore, the experiments we conducted with real data are discussed in Chapter 8. It includes integrating real patient data in the ontology and analysing the outcomes as well as comparing our approach with current practice in the hospital.

CHAPTER FIVE

ALTERNATIVE DRUGS RECOMMENDATION AND VALIDATION

In the previous chapter, we showed how PIMs can be detected in a purpose built Beers Criteria Ontology and underlying inference rules. This chapter goes one step further and describes the concepts and implementation associated to determining preferable alternative drugs, if these exist. In addition, to determining whether a prescription with alternative drugs is satisfiable, we detail the validation of the recommended alternatives for patient prescriptions using an SMT solver-based approach. Concretely, we explain the implementation of rules on the Beers Criteria ontology used to provide alternative drugs for prescription drugs. Moreover, how patient data and ontology recommendations, consisting of prescription drugs, interactions, and alternative drugs, are integrated into an SMT solver to validate the alternative drugs and find prescriptions that fit all the interaction requirements.

5.1 Beers Criteria Drug alternatives

The choices of drug treatments health professionals have at their disposal are numerous and can be challenging to select for complex cases. When prescribing drugs for older adults, it is essential to consider the probability of the treatment being both safe and effective. One solution to this challenge is providing

healthcare professionals with a list of evidence-based medications considered safer alternatives to those listed in the Beers Criteria. However, it is important to note that these drug recommendations are not intended to replace the expertise and knowledge of healthcare professionals [73, 63]. Therefore, when implemented in a CDSS, the system can suggest these drugs in specific circumstances. However, the decision to change a drug for an alternative drug is always defined by physicians.

At this point, the Beers Criteria ontology is able to detect inappropriate medications in prescriptions, which is the first stage of the decision-making process that the proposed CDSS is supporting health professionals. This first stage can support health professionals in avoiding prescribed PIMs; however, it does not support finding alternative drugs to sort out PIMs problems. Thus, the next stage is to provide alternative drug recommendations when inappropriate medications are found. Consequently, alternative drug recommendations were implemented in the Beers Criteria ontology to support the decision process to tackle this problem.

However, the ontology is not able to check if the recommended alternative drugs have interactions with other alternative or prescribed drugs. Consequently, the alternative drug recommendations have to be validated by another tool. Thus, we proposed an SMT model-based approach to validate the prescription consistency considering the alternative drugs. The model checks if there are interactions between drugs and provides all the possible prescriptions, combining the prescribed drugs with the alternative drugs.

5.1.1 Alternative drugs knowledge acquisition

The knowledge acquisition provides the necessary elements to define how to structure and formalise the alternative rules in the ontology. Thus, for each described interaction in the Beers Criteria, we scrutinised how the alternative drugs suggested by the AGS Health in Aging Foundation [73] and by Hanlon et al. (2015) [63] could be formalised and which additional ontology elements would be necessary. The suggested alternative drugs follow a similar hierarchy as the one defined for the Beers Criteria list, as shown in Figures 5.2 and 5.1 for illustration.

The process of determining the required components of the ontology to establish the rules for alternative drugs involved utilising the same methodology employed in defining PIMs. This approach involved employing semantic triples consisting of subject, predicate and object.

Therapeutic Class	High-Risk Medications	Alternatives
Anticholinergic		
First-generation antihistamine	Brompheniramine Carbinoxamine Chlorpheniramine Clemastine Cyproheptadine Dexbrompheniramine Dexchlorpheniramine Diphenhydramine (oral) Doxylamine Hydroxyzine Promethazine Triprolidine	Intranasal normal saline Second-generation antihistamine (e.g., cetirizine, fexofenadine, loratadine) Intranasal steroid (e.g., beclomethasone, fluticasone, over the counter)
Parkinson disease	Benztropine (oral) Trihexyphenidyl	Carbidopa/levodopa
Antiplatelets	Dipyridamole (oral immediate release) Ticlopidine	Antithrombotic therapy for the secondary prevention of noncardioembolic stroke Clopidogrel, aspirin 25 mg with extended-release dipyridamole 200 mg
Cardiovascular		
Alpha agonists, central	Guanabenz Guanfacine Methyldopa	Thiazide-type diuretic, ACEI, ARB, long-acting dihydropyridine CCB In black individuals—thiazide-type diuretic, CCB For heart failure, diabetes mellitus, chronic kidney disease—ACEI or ARB preferred
Other	Disopyramide	Atrial fibrillation: For rate control—nondihydropyridine CCB (e.g., diltiazem), beta-blocker For rhythm control—dofetilide flecainide, propafenone
	Nifedipine (immediate release)	Long-acting dihydropyridine CCB (e.g., amlodipine)
Central nervous system		
Tertiary tricyclic antidepressant	Amitriptyline Clomipramine Imipramine Trimipramine	For depression—SSRI (except paroxetine), SNRI, bupropion (also see Appendix 3) For neuropathic pain—SNRI, gabapentin, capsaicin topical, pregabalin, lidocaine patch

Figure 5.1: Example of alternative drugs suggested by Hanlon et al. (2015) [63].

Medication Class/Examples	Possible Alternatives to Discuss with your Healthcare Provider
First Generation Antihistamines (used for allergies) ■ chlorpheniramine (AllerChlor) ■ diphenhydramine (Benadryl)	■ saline nasal rinse ■ steroid nasal sprays ■ such as fluticasone (Flonase) ■ Allergy products such as: - cetirizine (Zyrtec) - fexofenadine (Allegra) - loratadine (Claritin)
Tricyclic Antidepressants used for depression ■ amitriptyline (Elavil) ■ imipramine (Tofranil)	■ selective serotonin reuptake inhibitors (SSRIs) such as: - citalopram (Celexa) - sertraline (Zoloft) ■ buproprion (historically known as Wellbutrin)
Barbiturates ■ phenobarbital ■ other drugs ending in "barbital"	For epilepsy, anticonvulsants such as: ■ lamotrigine (Lamictal) ■ levetiracetam (Keppra)

Figure 5.2: Example of alternative drugs suggested by the AGS Health in Aging Foundation [73].

Furthermore, when defining the rules for the alternative drug, previously established components in the Beers Criteria, such as classes, objects, and data properties, were taken into account, and additional components were included to properly formalise the rules if and as needed.

To exemplify how the elements were defined, Figure 5.3 illustrates a scenario in which a patient has been prescribed the drug Amobarbital, which belongs to the Barbiturates drug category. Furthermore, this patient is over the age of 65 and suffers from Epilepsy. According to the Beers Criteria, Amobarbital is considered a PIMs, which means an alternative drug could be prescribed to avoid undesired effects. Therefore, alternative drugs from the Anticonvulsant category can be considered. Lamotrigine and Levetiracetam, which belong to the Anticonvulsant drug category, are examples of alternative drugs to Amobarbital for this prescription. To represent the link between the prescribed drug and the alternative drugs, the object property has alternative and its inverse *is alternative of* were created.

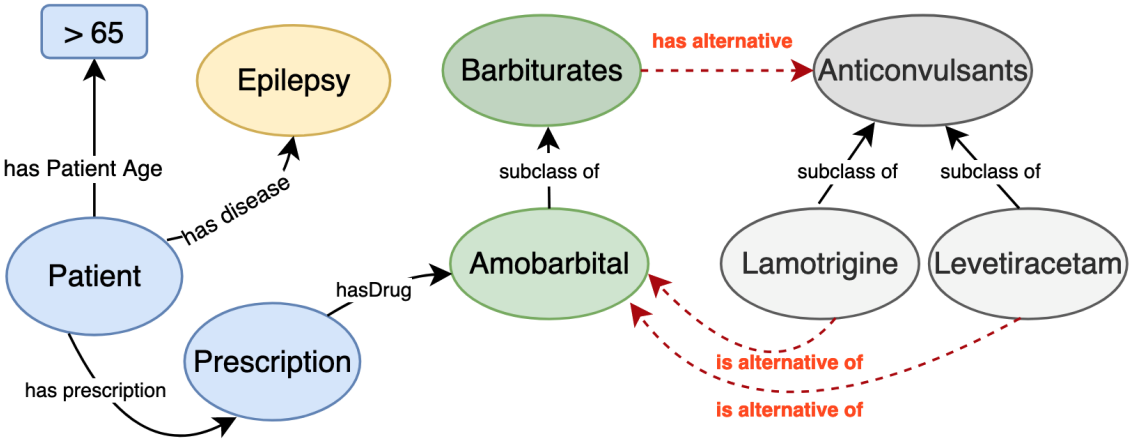


Figure 5.3: Alternative drugs semantic triple

The elements in Figure 5.3 could be represented in semantic triples according to Table 5.1. The table is composed of three semantic triple columns (Subject, Predicate and Object). Additionally, the column Statement is added to clarify when the relation between the elements happens, which can be either in a TBox or a ABox. As previously explained, TBox statements provide the framework for understanding the Beers Criteria taxonomy while ABox statements are the assertions or factual statements that relate to the conceptual model defined by the TBox. In Figure 5.3, the TBox is represented by the black arrows and the ABox by

the red dotted arrows. From Table 5.1, column Statement, it is possible to deduce that for the ABox to be true, all TBox statements must be true.

Subject	Predicate	Object	Statement
Patient	has age higher than	65	TBox
Patient	has disease	Epilepsy	TBox
Patient	has prescription	Prescription	TBox
Prescription	has drug	Amobarbital	TBox
Amobarbital	subclass of to	Barbiturates	TBox
Lamotrigine	subclass of to	Anticonvulsants	TBox
Levetiracetam	subclass of to	Anticonvulsants	TBox
Barbiturates	has alternative	Anticonvulsants	Abox
Lamotrigine	is alternative of	Amobarbital	Abox
Levetiracetam	is alternative of	Amobarbital	Abox

Table 5.1: Semantic triple for alternative drugs

The knowledge acquisition process resulted in new properties and classes that were utilised to establish alternative rules. Classes must be organised into a hierarchical taxonomy that adheres to the same organisational logic employed in the Beers Criteria ontology. The upcoming section will provide further information on the specific elements that were discovered and how they are structured.

5.2 Alternative drug recommendation ontology rules

The alternative drug recommendation rules aim to provide alternative drugs for inappropriate medications. These rules are implemented on the Beers Criteria ontology based on the elements defined in the knowledge acquisition. In this section, we detail the necessary elements to build the rules and how the rules are formalised.

5.2.1 Ontology elements

In addition to the previously described elements from Chapter 4, a new collection of classes, objects and data properties were defined with the semantic triple approach in order to develop the alternative drug recommendation rules in the Beers Criteria ontology.

5.2.1.1 Drug alternative classes

To systematise the alternative drug classes, a new main class was created named Alternative_drugs. The alternative drugs were defined by Hanlon et al. (2015) [63]in two main classes: High-Risk_Medications class and Medications Drug-Disease_Interactions. Therefore to define the ontology taxonomy, these two alternative drug category classes were employed in the ontology as shown in Figure 5.4. Each alternative drug category class is composed of subclasses in which the High-Risk_Medications subclasses are based on the Therapeutic Class, and the Drug-Disease_Interactions subclasses are based on the Diseases Class. The newly defined class names start with the initials "Alt" to indicate that they belong to the category of alternative drugs and to distinguish them from previously established classes.

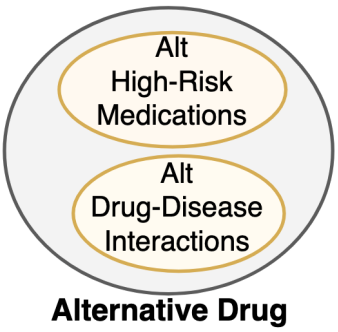


Figure 5.4: Main classes for the alternative drugs hierarchy

The Alt High-Risk_Medications class is illustrated in Figure 5.5. The class comprises six main subclasses: Alt Central nervous system, Alt Endocrine system, Alt Anticholinergic, Alt Pain medication, Alt Cardiovascular and Alt Antithrombotic anti-platelets. Each of these subclasses contains drug categories subclasses and their corresponding medication classes. For instance, the Alt Pain medication subclass comprises three subcategories of drugs, while the Alt Antithrombotic anti-platelets class consists of three specific medications. As depicted in Figure 5.5, the medications from Alt Antithrombotic anti-platelets start with the initials "Alt", as

they were previously defined for the drug class and do not belong exclusively to the Alternative_drugs taxonomy.

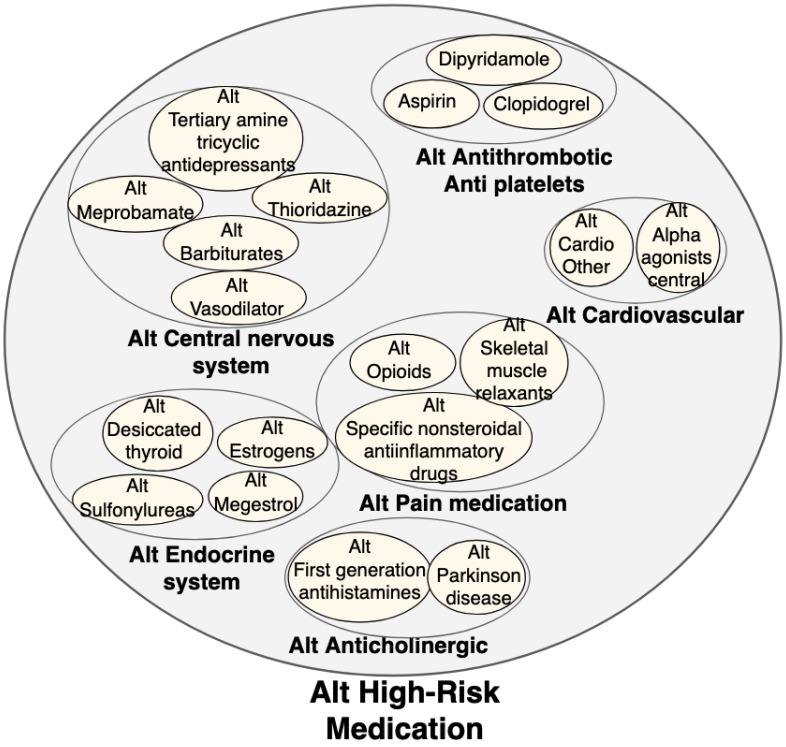


Figure 5.5: Alternative drugs hierarchy for High-Risk Medications.

The Alt Medications Drug-Disease_Interactions is composed of three main classes: Alt Falls, Alt Dementia and Alt Kidney diseases. These classes have drug category subclasses linked with drug classes. Moreover, there are drug category classes that can be an alternative for more than one disease class. For example, Figure 5.6 illustrates an intersection between the classes Alt Fall and Alt Dementia, where five drug category subclasses belong to both disease classes.

The Beers Criteria PIMs classes and alternative drugs have a similar taxonomy and similar class names; thus, finding which drug may be an alternative for an interaction drug or group of drugs is straightforward. The alternative drugs taxonomy aims to facilitate the creation of rules and to find a possible alternative drug. However, particular constraints have to be considered for each case; for example, some drugs are considered alternatives only when the patient suffers from a specific disease, as exemplified in Table 5.1. Therefore, a rule that specifies the criteria for a drug to be considered an alternative must be defined for each alternative drug or drug category.

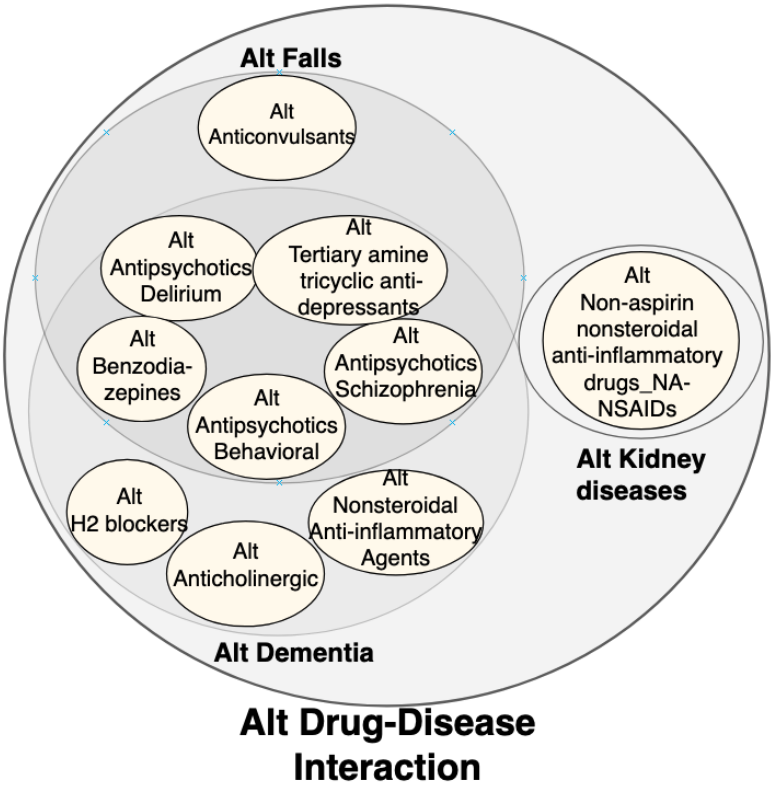


Figure 5.6: Alternative drugs hierarchy for drug-disease interactions

5.2.1.2 Object and data property

The relation that defines if a drug is an alternative to the other is stated by object properties. Moreover, there are individual drugs that belong to a drug class that might not be considered alternative; hence, we set a data property to define whether an individual drug is or is not an alternative. The object and data properties used to define the rules for determining alternative drugs are detailed below:

Object property:

- **hasAlternative**: defines the relationship between a drug and an alternative. Therefore if a drug has an alternative, the hasAlternative is the predicate between the subject drug and the object alternative drug. Its domain is the class Drug, and its inverse property is isAlternativeOf;
- **isAlternativeOf**: works in the inverse direction of object property hasAlternative. It is the predicate between the subject alternative drug and the object

drug.

Data property:

- **isAlternative**: defines if an individual drug can be considered an alternative to other drugs.

The defined elements are essential for the definition of the drug alternative rules. Most of the elements previously defined in the Beers Criteria ontology were used, as well as the taxonomy. The next step for the ontology to suggest alternative drugs is to define the drug alternative rules according to each drug specification, which was mapped in the semantic triple approach.

5.2.2 Drug alternative rules

The alternative drug recommendation rules in the Beers Criteria ontology aims to provide alternative drugs in cases where alternatives are available and could be offered instead. As before, our rules consist of two parts: an antecedent and a consequent. The antecedent establishes the requirements to define whether a drug is (or is not) an alternative to other drugs. The consequent links a drug with an alternative when the antecedent is true. The rules are given in predicate logic. In the following, we use p to denote an arbitrary patient, pr a prescription, d and $altD$ drugs, pd a disease, and a an integer. Additional variables may be introduced in the rules as needed and are explained in the context of the rule.

Two main categories of drugs, namely Alt High-Risk_Medications and Alt Medications Drug-Disease_Interactions, are used to make alternative drug recommendations. Thus, to demonstrate how the drug alternative rules are defined, we explain one rule for each category.

In the Rule 5.1, we define an alternative drug recommendation rule for the drug category Histamine H2 Antagonists, a group of drugs (which can be) used to treat symptoms of Dementia. The rule states that under certain conditions (see below) and if there is an alternative drug from class Alt H2 blocker, then the Histamine H2

Antagonists prescribed drug can be replaced by the alternative.

$$\begin{aligned} &Patient(p) \wedge Prescription(pr) \wedge hasPrescription(p, pr) \\ &\wedge hasDrug(pr, d) \wedge Drug(d) \wedge Histamine\_H2Antagonists(d) \\ &\wedge Drug(altD) \wedge Alt\_H2\_blockers(altD) \wedge hasPatientAgeValue(p, a) \\ &\wedge (a \geq 65) \wedge isAlternative(altD, true) \wedge Dementia(pd) \\ &\wedge hasDisease(p, pd) \Rightarrow hasAlternative(d, altD) \end{aligned} \tag{5.1}$$

The rule can be broken down as follows:

- $Patient(p) \wedge Prescription(pr) \wedge hasPrescription(p, pr) \wedge hasDrug(pr, d)$: specifies that p is a patient with a prescription pr containing drug d.
- $Histamine\_H2Antagonists(d) \wedge Drug(d)$: specifies that d is a drug of type Histamine H2 Antagonists.
- $Drug(altD) \wedge Alt\_H2\_blockers(altD)$: establishes that $altD$ is a drug that belongs to the class Alt_H2_blocker.
- $hasPatientAgeValue(p, a) \wedge (a \geq 65)$: states that patient p is aged 65 or above.
- $isAlternative(altD, true)$: specifies that $altD$ is an alternative drug.
- $Dementia(pd) \wedge hasDisease(p, pd)$: determines that patient p has a diagnosis (pd) of dementia.
- $hasAlternative(d, altD)$: this predicate is used to indicate that $altD$ is an alternative drug for d.

In summary, this rule states that for a patient aged 65 or above, suffering from dementia, with prescription containing drug d of type Histamine_H2Antagonist, we can replace d with $altD$ of type Alt_H2_blocker.

The Rule 5.2 defines an alternative drug for the drug category Alt First Generation Antihistamines, which belongs to the Alt Anticholinergics and Alt High-Risk Medication classes, as follows:

$$\begin{aligned} &Patient(p) \wedge Prescription(pr) \wedge hasPrescription(p,pr) \wedge \\ &hasDrug(pr,d) \wedge First\_Generation\_Antihistamines(d) \wedge Drug(d) \wedge \\ &Drug(altD) \wedge Alt\_First\_generation\_antihistamines(altD) \wedge \\ &hasPatientAgeValue(p,a) \wedge (a \geq 65) \wedge isAlternative(altD,true) \\ &\Rightarrow hasAlternative(d,altD) \end{aligned} \tag{5.2}$$

The rule is similar to the one given earlier. It states that for a patient aged 65 or above, if the patient has been prescribed a medication d that is a first-generation antihistamine, then if there is an alternative $altD$ in the same group of medications, then d can be replaced by $altD$.

5.2.2.1 Applying the ontology alternative rules

By establishing the rules for alternative drug recommendations, the ontology can provide alternative drugs for the prescribed drugs. To illustrate how the ontology recommends alternative drugs, we recall the example from Figure 4.18 which introduced patient Tom, who suffers from a history of falls, is 75 years old and male. As we saw before, all the prescribed drugs (P1_Metoclopramide, P1_Codeine, P1_Morphine, P1_Triazolam) that Tom was given in a prescription P1 were classified as inappropriate.

The Beers Criteria ontology aims to find alternative drugs to replace the inappropriate drugs prescribed for P1, in accordance with the defined rules. Figure 5.7 shows diagrammatically the ABox result after the alternative inference rules were performed. The result shows that the ontology could find four alternative individual drugs alt_Pregabalin, alt_Escitalopram, alt_Fluoxetine and alt_Levetiracetam, for drug P1_Triazolam. These individual drugs are represented in orange lozenges and are linked with the individual P1_Triazolam by the object properties has Alternative (green arrows) and by its inverse property is alternative of(orange arrows). Each individual alternative drug belongs to its respective drug class. For example, alt_Escitalopram belongs to drug class Escitalopram. This drug class also belongs to the drug category Selective_Serotonin-reuptake_Inhibitors and to the alternative classes Alt Anticonvulsants, Alt Falls, Alt Drug-Disease Interaction and Alternative drugs.

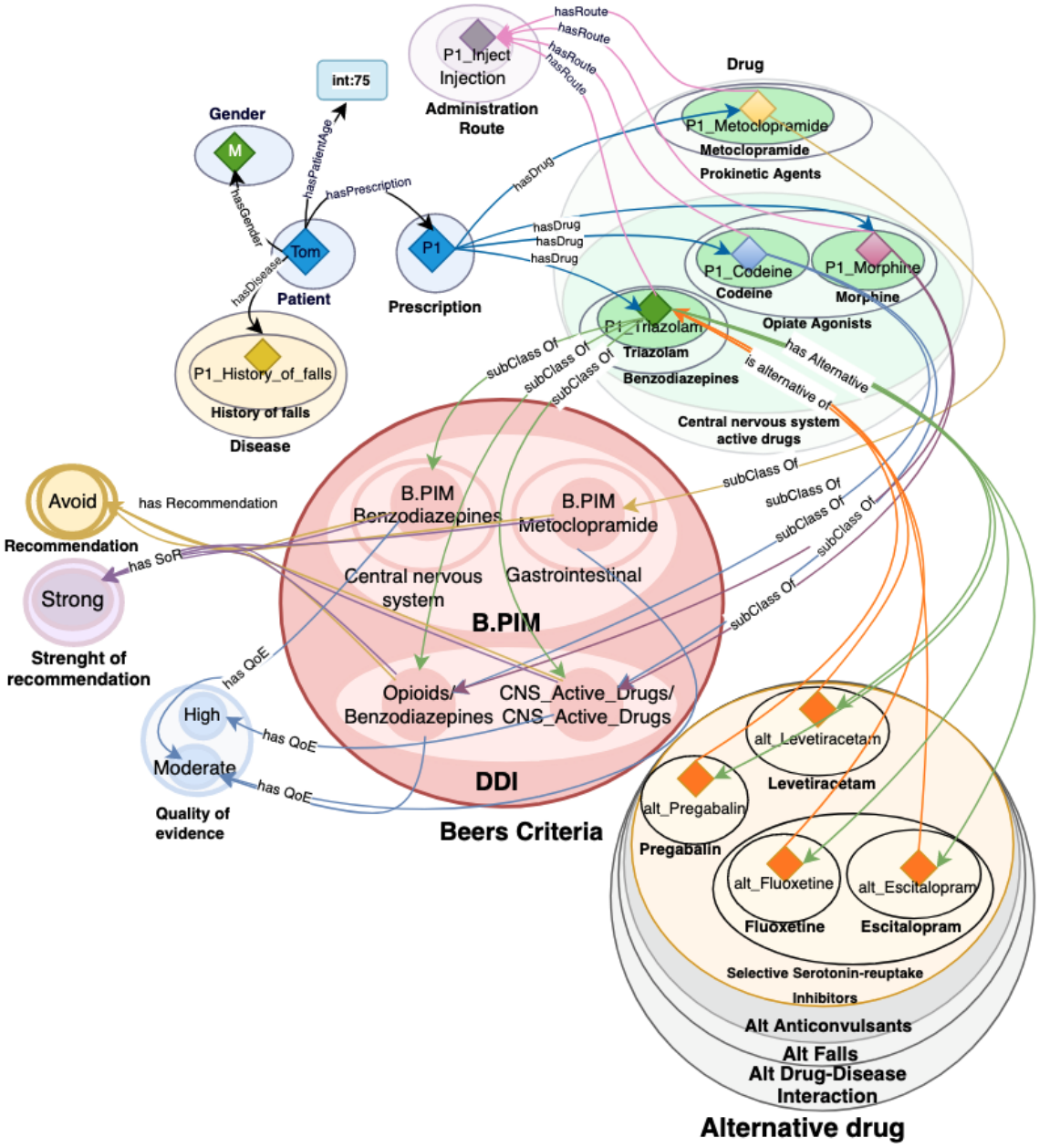


Figure 5.7: Alternative drugs ABox example.

To comprehend how these alternative drugs were inferred for the individual drug P1_Triazolam, we will scrutinise the corresponding rule. Regarding the patient parameters, we know that Tom is 75, suffers from falls and has a prescription for the individual drug P1_Triazolam, which belongs to drug class Triazolam. Additionally, the drug class Triazolam also belongs to the drug category classes Benzodiazepines and Anticonvulsants. These alternatives are defined in the

inference Rule 5.3:

$$\begin{aligned} &Patient(p) \wedge Prescription(pr) \wedge hasPrescription(p,pr) \wedge \\ &hasDrug(pr,d) \wedge Anticonvulsants(d) \wedge Drug(d) \wedge Drug(altD) \wedge \\ &Alt\_Anticonvulsants(altD) \wedge isAlternative(altD,true) \wedge \\ &hasPatientAgeValue(p,a) \wedge (a \geq 65) \wedge History\_of\_falls(pd) \\ &\wedge hasDisease(p,pd) \Rightarrow hasAlternative(d,altD) \end{aligned} \tag{5.3}$$

The first part of the formula above is similar to earlier rules. The second part of the formula is an implication ($\Rightarrow$) that connects the antecedent (the conditions specified in the first part) with the consequent (the statement that follows the arrow). The consequent states that if the patient has a prescription of an *Anticonvulsants* drug d, then this drug could be switched to a drug that belongs to the class *Alt_Anticonvulsants*. In other words, the formula states that for patients aged 65 or above with a history of falls, who have been prescribed an Anticonvulsants drug, there exists an alternative drug that can be prescribed instead of the original drug. Therefore, all the drugs that belong to the class *Alt_Anticonvulsants* are considered alternative drugs.

This example shows how the rules were created and how the ontology provides alternative drug recommendations. Rules were created for all the alternative drugs listed by Hanlon et al. (2015) [63] and by the AGS Health in Aging Foundation [73]. However, the alternative drug recommendations do not consider whether there is interaction with other prescribed drugs or alternative drugs. Therefore, a new approach is proposed in the following sections to validate the alternative drugs.

5.3 Alternative drugs validation by an SMT model

The Beers ontology supports health professionals in detecting PIMs and finding alternative drugs. Thereafter, the next decision that health professionals have to make is defining which alternative drugs can be selected without introducing new interactions with other drugs. Consequently, the alternative drug recommendations have to be validated to support health professionals in finding (better alternative) prescriptions without interactions. To facilitate this we propose a novel SMT model based approach for validating and finding alternative drugs

that do not interact with other prescribed drugs.

The patient prescription, alternative and inappropriate drugs are provided by the ontology. The SMT model will incorporate this information in order to be able to validate it. The prescription validation process happens in three steps. First, the data is extracted from the ontology, then converted and declared in SMT, and finally, the SMT solver checks whether there are valid prescriptions. These steps will be explained in the following sections. The result of the SMT model (a solution) is a prescription containing drugs without interactions (and suitable for patients aged 65 or above) or no solution (unsat - the model is unsatisfiable) can be found because there are no prescriptions without interactions.

5.3.1 Retrieving data from the Beers Criteria ontology

The first step outlined to determine valid alternative prescriptions corresponds to retrieving the necessary information from the ontology. The prescribed drugs for the patient, the suggested alternative drugs and the interaction between drugs are obtained from the ontology.

Concerning drug interactions, these are extracted from the Ontology by the object property hasInteractionWith. As for any binary relation between elements of two sets, it can be seen as a subset of pairs of elements from the corresponding sets. For hasInteractionWith, we have $hasInteractionWith \subseteq Drug \times Drug$. For inappropriate drugs from the group Potentially Clinically Important Drug-Drug Interactions (DDI), the interaction happens between two different drugs. For example, for drugs $d_1 \neq d_2$ in DDI we have d_1 $hasInteractionWith$ d_2 which we represent equivalently as a pair $(d_1, d_2) \in hasInteractionWith$.

In contrast, inappropriate drugs classified as DDDS, B.PIM, UWC, and VLKF usually do not have interactions with other drugs. These groups of drugs are considered to interact for other reasons, such as age, diagnosed disease (e.g., comorbidities that the patient has) or clinical condition. If these parameters affect a given patient, these drugs must be switched to other suitable alternative drugs. To encode the problem of taking such drugs in these cases, we assume an interaction of the drug with itself. For instance, if d is a drug within DDDS (or any of these classifications), then we assume that d $hasInteractionWith$ d which we represent equivalently as a pair $(d, d) \in hasInteractionWith$.

Similarly, alternative drugs are extracted from the ontology by the object property

hasAlternative which links the prescribed drug with (one or more) alternative drugs. For hasAlternative, we have $hasAlternative \subseteq Drug \times Drug$ where for $(d, ad) \in hasAlternative$ is such that d is the prescribed drug whereas ad is one possible alternative. If d has several alternatives $ad_1 \ldots ad_n$, then $(d, ad_i) \in hasAlternative$ for $1 \leq i \leq n$.

To explain the data structure that composes the data extracted from the Beers Criteria ontology, we use the example from Figure 5.7 for patient Tom. This example can be summarised in three sets:

Prescription set = {P1_Metoclopramide, P1_Codeine, P1_Morphine, P1_Triazolam}

Interaction set = {(P1_Metoclopramide, P1_Metoclopramide), (P1_Codeine, P1_Morphine), (P1_Codeine, P1_Triazolam), (P1_Morphine, P1_Triazolam)}

Alternative set= {(P1_Triazolam, alt_Pregabalin), (P1_Triazolam, alt_Fluoxetine), (P1_Triazolam, alt_Escitalopram), (P1_Triazolam, alt_levetiracetam)}

The prescription set represents the 4 drugs that our patient Tom was prescribed. The interaction set shows pairs of drugs that are problematic together. In the case of *Metoclopramide*, this drug cannot be taken by Tom and is hence excluded by imposing an interaction with itself. The alternative set describes possible replacements for drugs listed in the original prescription $P1$. For instance, *Pregabalin* can be used instead of $Triazolam$, but so can $Fluoxetine, Escitalopram$ or *Levetiracetam*.

The data retrieved from the ontology, as shown above, now needs to be passed to an SMT solver as described next.

5.3.2 Converting rules into SMT

Once the information is extracted from the ontology, drug details, interaction, and alternative rules can be passed to the solver. The process consists of adding the three sets (described through an example in Section 5.3.1) to the SMT solver: prescription, interaction and alternative sets. The underlying SMT model implemented in Python(as used at present - arithmetic will be useful later on) is based on Boolean logic, which means the solver will find Boolean assignments for all the variables: in our case, whether a drug is present/taken or not, which can be true or false. Drugs with true values correspond to those that should be prescribed, and drugs with false values will not be considered. For example, since

the alternative set gives several options for individual drugs (e.g., the case of options for *Pregabalin*), one solution will pick one of these options and assign it a true value, whereas all others are set to false.

5.3.2.1 Drug Declaration

To proceed with the drug declaration for the model, each drug is declared as belonging to the Drug Datatype, a custom data structure for our purposes. Custom data structures are useful because they allows us to introduce new data types, useful for our domain and beyond the built-in ones the solver provides, such as integers, Boolean, and arrays. Once a data type is defined, it can be used to create instances of it. In the alternative model, each drug that composes the model is an instance of the datatype Drug.

Considering that the drug instances would ideally be defined as Boolean (e.g., true if picked, false if not) and the Drug datatype is not the same as Boolean, a function named ***choice*** was created to attach a Boolean value to drug instances. The function ***choice*** as defined below

$$choice : Drug \rightarrow BoolSort()$$

takes a parameter of type ***Drug*** and returns a value of type ***BoolSort()***, which represents the Boolean values. For example, to express the choice of a drug Paracetamol, we indicate this through the function choice as follows `choice(Drug.Paracetamol) == True`.

Algorithm 1: Algorithm for declaring drugs

```
1 def declare_drug(drugSet):
2     Drug = Datatype
3     for d ∈ drugSet do
4         Drug.declare(d)
5     Drug.create()
6     choice = Function('choice', Drug, BoolSort())
7 ,
```

The Algorithm 1 above (shown in pseudocode) represents how instance drugs are declared in the SMT model through the function ***declare_drug***. The function receives a set of ***drugs*** that are declared as datatype ***Drug***. Thereafter the drugs are created in the SMT model, and the function ***choice*** is defined. Notice that up

to this point, no Boolean value has been assigned for the drug instances by the function ***choice***.

5.3.2.2 Drug constants

The SMT alternative drug model was set to provide a prescription that includes the exact number of prescribed drugs, considering alternative drugs. It means that if five drugs were prescribed, the solver will provide a solution with five drugs. Otherwise, the solver could provide a solution with more or less drugs than prescribed, which could not be ideal. Hence, we have to define a constraint forcing the solver to provide the same number of drugs as prescribed. To formalise this constraint we declare *Constants* of type *Drug*(which is a datatype previously explained) with the exact number of prescribed drugs.By declaring these Constants and declare that they *Exist* and are *True*, we force the model to provide a solution with the prescribed number of drugs. Moreover, that the drugs assigned are *Distinct* for each *Constants*. The Algorithm 2 was defined to define this constraint.

Algorithm 2: Algorithm for declaring drug constants

```
1 def declare_constant(numOfPrescribedDrugs):
2     distinctRules = list()
3     for x ∈ numOfPrescribedDrugs do
4         drug(x) = Const(drug(x), Drug)
5         sol.add(Exists(drug(x),choice(drug(x))))
6         sol.add(And(choice(drug(x))) == True)
7         distinctRules.append(drug(x))
8     sol.add(Distinct([x for x in distinct_rules]))
```

In Algorithm 2, the function *declare_constant* receives the variable *numOfPrescribed Drugs* with the total number of prescribed drugs. Then in a for loop, the *Constants* are created. First, it is assigned for a sequential variable drug(x), a Constant named drug(x)(the same name as the variable) of type Drug. Then is defined that *Exists* a $choice(drug(x))$ for the drug constant $drug(x)$. Next, we define that $choice(drug(x))$condition should be equal to *True* and add $drug(x)$ in the *distinctRules* list. Finally, we define that all constants in *distinctRules* are *Distinct*, which means the constants of type Drug are not equal to each other.

To exemplify how it works, let us imagine that we have a prescription with three drugs(A, B, C), the interaction between A and B and the alternatives (X, Y) for drug A. Three variables (drug1, drug2, drug3) would be instantiated with *Const*

(with the same name) of type Drug. Then we declare to the solver that Exists a choice(drug1), choice(drug2), choice(drug3). Next, we define that choice as true. Finally, we define that drug1, drug2, and drug3 must be distinct.

So the solver could provide solutions such as (X, B, C) or (Y, B, C). If this constraint would not be defined, the solver could provide several solutions, for example, (B, C), (X, B) or (B).

5.3.2.3 Mandatory true drugs rules

The SMT model has to provide the same number of prescribed drugs. However, it is not only the number of drugs that matter but also which drugs the SMT model selects for the prescription. Some drugs do not have interaction, and alternative drugs available. Thus, these drugs have always to be considered in the prescription.

For defining these drugs as always *True* in the SMT model, we consider the following definition in Algorithm 3.

Algorithm 3: Algorithm for declaring mandatory drugs

```
1 def declare_true_drugs(drugSet):
2     sequence = 0
3     for d ∈ drugSet do
4         sol.add(And(choice(Drug.d)) == True))
5         sol.add(Exists(Drug(sequence),choice(Drug.d)))
6         sequence += 1
```

Algorithm 3 receives as a parameter a set of drugs that have to be considered *True*, i.e., drugs that cannot be replaced and need to be kept in the prescription (for some reason). To do so, we iterate through the set of drugs and add a constraint stating that the drug is chosen (for each of the drugs in the set), i.e., function *choice* associates the value *True* to all the drugs in the set.

5.3.2.4 Interaction rules

The definition of an interaction rule refers to drugs that can not be prescribed together. Therefore, the SMT model has to choose only one drug with interaction. There are cases where a drug has interactions with more drugs. In this case, several rules are created. For example, if A interacts with B and C and B interacts with C, then the solver can choose either A or B or C, but not more than one.

Algorithm 4 defines the interaction rules for a pair of drugs. The function ***declare_interactions*** receives as parameters the interactions sets, which are sets of two drugs. For example {(A,B),(A,C),(B,C)}. Then a constraint with the interaction drugs is added in the solver.

Algorithm 4: Algorithm for interactions

```
1 def declare_interaction(drugPairsSet):
2     for (d_1,d_2) ∈ drugPairsSet do
3         sol.add(Or(Not(choice(Drug.d_1))),(Not(choice(Drug.d_2))))
```

The rule represents a logical expression which in this case makes use of the logical operators "Or" and "Not" to combine two Boolean values, representing the selection of one of the two drugs. In this rule, if $Drug.d_1$ and $Drug.d_2$ are both not $True$, then the expression holds. If only one of the drugs, $Drug.d_1$ or $Drug.d_2$, is declared $True$, then the expression also holds. However, note that if both $Drug.d_1$ and $Drug.d_2$ would be declared as $True$, the expression would no longer hold. In other words, only three possible assignments for d_1 and d_2 are satisfiable.

5.3.2.5 Alternative rules

In the SMT model, alternative drugs are considered drugs that can substitute prescribed drugs. Thus, when a constraint does not allow a drug to be prescribed, for example, due to an interaction, then the solver can switch this drug for an alternative drug (when available) based on the defined alternative rule. Considering the example from the interaction drugs, where A interacts with B and C and B interacts with C, if A has the alternative X, and B has the alternative Y, then the solver could define a valid prescription (X, Y, C).

The Algorithm 5 declares the rule for alternative drugs. The function ***declare_alternative*** receives as parameters a set of two drugs (prescribed and alternative drug) and adds them into the solver. The constraint rule comprises a Xor operator in which either choice(Drug.d1) or choice(Drug.d2) can be assigned as $True$, but not both drugs.

The operator Xor stands for exclusive or. This logical operator takes two Boolean operands and returns true if and only if exactly one of the operands is $True$ and the other is $False$. It indicates that it is impossible to prescribe both an alternative drug and a prescription drug simultaneously. For instance, the combination of

Algorithm 5: Algorithm for declaring alternative drugs

```
1 def declare_alternative(alternativePairsSet):
2     rule = ''
3     for (drug1, drug2) ∈ alternativePairsSet do
4         rule = (Xor(choice(Drug.d1), choice(Drug.d2),rule))
5     sol.add(rule)
```

A and X or B and Y can not be recommended concurrently since they may have identical therapeutic effects for the patient. Thus, the solver will define only one drug as true. In the example from the alternative set, we have the alternatives (P1_Triazolam, alt_Pregabalin), (P1_Triazolam, alt_Fluoxetine), (P1_Triazolam, alt_Escitalopram), (P1_Triazolam, alt_levetiracetam). In this case, four drugs can be alternatives to P1_Triazolam, but only one drug from all the alternatives can be defined as *True*.

5.3.2.6 Checking prescriptions

After declaring the drugs and specifying the SMT model interactions, alternatives, and requirements, the solver verifies if there is a prescription that meets all the constraints. More prescriptions may be possible, but unless we enforce this, the solver only provides one solution. Indeed, here in case a prescription that fits all the constraints is found, the algorithm checks if other valid prescriptions can be found, in order to provide health professionals with more prescription options.

Algorithm 6 checks all the possible prescriptions without interaction or drugs classified as PIMs, considering the predefined constraints (interaction, alternative, mandatory drugs). The function ***check_prescription*** receives as parameters a set with the mandatory drugs (drugs that must be picked and hence are necessarily *True*). First, it is checked if the constraints are satisfiable ($sol.check() == sat$), which means there are prescriptions without interactions. Next, if the constraints are satisfiable, then the solver tries to find more prescriptions without interactions in a *while* loop. These prescriptions are stored in a model set. Then the function get_true_drugs returns a list of drug names assigned as *True* by the solver, disregarding the mandatory drugs, and assigns this list for the variable *trueDrugs*. If the list *trueDrugs* is not null, then a new constraint is added to the solver. This constraint determines that one of the *True* drugs should be assigned as *False* next time the solver checks for prescriptions without interactions in the *while* loop. For

example, if the model has three drugs (X, B, C) assigned as *True*, this constraint will determine that Or X, Or B, Or C has to be $\neq True$. So this new constraint force the solver to find another valid drug. This loop happens until prescriptions without interactions cannot be found. Thus, this algorithm provides all the possibilities of prescriptions considering the alternative drugs available.

Algorithm 6: Algorithm for declaring interaction constraints

```
1  def check_prescription(mandatoryDrugs):
2      model = set()
3      if sol.check() == sat then
4          while sol.check() == sat do
5              model.add (sol.model())
6              solution = "False"
7              trueDrugs =
                get_true_drugs(sol.model()).difference(mandatoryDrugs)
8              if trueDrugs then
9                  for d in trueDrugs do
10                     solution = Or((choice(drug.d) != True), solution)
11                 sol.add(solution)
12             else
13                 break
14     return (model)
```

Considering again the example of Tom's prescription, only alternatives for drug P1_Triazolam were available, which would result in an unsat prescription. Therefore, in this example, to get a prescription that would satisfy the constraints, it would be necessary to have alternative drugs for P1_Metoclopramide, P1_Codeine or P1_Morphine, P1_Morphine or P1_Triazolam and P1_Codeine or P1_Triazolam. In particular, if there were alternatives for drugs P1_Metoclopramide, P1_Codeine and P1_Triazolam without interaction with other drugs, the prescription would be valid.

When a valid prescription cannot be found, there are two options: either prescribe the drugs with interaction (e.g., there may be a tradeoff between using a drug and avoiding a potentially severe side effect which may not occur) or not prescribe them. In case we choose to prescribe drugs that have interaction, we can investigate if we can attempt to minimise the interaction in some way. Our novel proposed approach is to minimise this interaction by maximising the distance between the peak (highest concentration in the blood) of both drugs and offering scheduling

options for these drugs accordingly. This approach will be detailed in the next chapter.

5.4 Summary

This chapter explained how alternative rules are formulated and added to the Beers Criteria ontology, and validated by an SMT model. We added new elements to the ontology in order to define the alternative rules. Moreover, we detailed how an SMT model can support health professionals in selecting drugs that do not interact with other prescribed drugs. With this approach, we aim to support the decision process beyond simply identifying PIMs. However, a prescription without interactions can not always be found. Therefore, in the next chapter, we describe a novel approach to minimise drug interactions when interacting drugs cannot be replaced and have to be prescribed.

CHAPTER SIX

DRUG SCHEDULING OPTIMISATION FOR MINIMISING DRUG INTERACTIONS

This chapter describes an SMT solver based approach for optimising drug schedules which minimises the effect of drug interactions. As described in Chapter 5, there may be cases where drugs cannot be replaced to avoid interactions. In such cases, our aim is to find solutions which maximise the distance between the maximum serum concentration of the drugs in the body after the drug has been administered. In other words, by scheduling interacting drugs in ways that avoid having their maximum concentration at the same time, we are trying to reduce the impact of their interaction. The proposed solution makes use of a known TMAX value for drugs and adds it to the ontology. TMAX corresponds to the time when the maximum concentration is reached after their administration. We formalise the scheduling and detail all relevant constraints for the SMT model, including administration times, drug frequency, meal and drug administration time rules. Overall, we demonstrate how this approach can be used to support health professionals in defining drug schedules that minimise interaction when the prescribed drugs themselves cannot be replaced for good reasons.

6.1 Drug administration scheduling

The drug scheduling process defines the time and frequency of drug administration. Each drug has recommended administration schemes that are commonly prescribed by physicians. The *frequency* refers to the number of times that a drug should be administered. For example, three times a day usually means this drug should be given every eight hours. The *time* indicates precisely when the drug has to be administered. For example, some drugs should be taken at particular times, such as after or before meals and before going to sleep.

As mentioned in earlier chapters, prescriptions often involve taking several medications which may interact with each other and even have contraindications when taken together. These drug interactions can sometimes yield positive outcomes, such as enhanced effectiveness. However, they more frequently lead to adverse effects such as reduced effectiveness and increased toxicity [44, 122].

In a hospital routine, the nurse plays a crucial role in ensuring the safe and accurate scheduling of medication administration. They are responsible for adhering to fixed routine hours and recording the specific times medications are given. However, using standardised and fixed schedules can inadvertently lead to the simultaneous administration of multiple drugs to the same patient. As mentioned before, this situation increases the risk of accidental drug interactions, as different medications may interact negatively when taken together [37].

Specially in cases where medications cannot be dropped or replaced, implementing rescheduling measures is a preventive approach to minimise the occurrence of drug interactions. The risk of adverse interactions can be reduced by scheduling medications with the potential for interaction at different times. However, effectively monitoring and implementing rescheduling activities poses a significant challenge, particularly when dealing with patients at a higher risk of drug interactions [8]. It is a common practice to maximise the distance between the administration times of drugs that have interaction. However, this approach is sometimes ineffective as a drug's effect does not always start right after it has been administered.

An enhanced approach to rescheduling measures to minimise the interaction could consider CMAX as a key factor. CMAX is the peak concentration of a drug in the body after administration, and it is a crucial parameter in determining dosing schedules. A drug needs to reach a high enough concentration for effectiveness

while avoiding side effects. In pharmacokinetic studies, CMAX is widely used to understand drug-body interactions, as it represents the maximum concentration of the drug [34]. CMAX does not indicate the drug's distribution throughout the body. Different organs may have varying concentrations, with organs like the liver potentially having higher values due to the metabolic breakdown. Conversely, TMAX refers to the time it takes for the drug to reach its peak concentration after administration, measured in units of time [135, 84]. Opting for a strategy that maximises the distance between the TMAX values of interacting drugs proves more effective than solely focusing on maximising administration time.

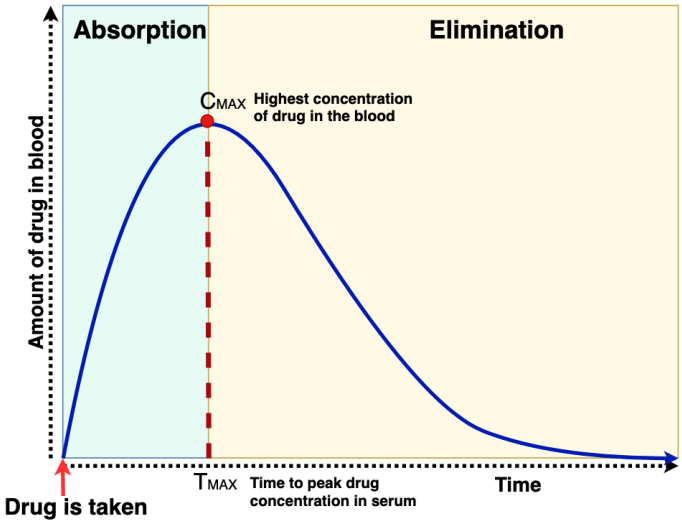


Figure 6.1: CMAX parameter after drug administration.

Figure 6.1 show how the maximum concentration is reached at time TMAX for an arbitrary drug. The drug is taken at time zero, where the amount of the drug in the blood is also zero. After the drug is administered, it is absorbed and metabolized by the body until it reaches CMAX represented by the red dot at time TMAX. After reaching the highest concentration, the level of the drug in the blood starts to decrease. Absorption and elimination times vary for different drugs and different people. When finding the right time for administering a drug we should consider the time the highest concentration is reached (TMAX), and use this information when trying to avoid the effects of interactions between this and other drugs. Even though understanding the relationship between CMAX and TMAX for a drug and how this may vary when given to different people (e.g., how genetics may affect drug absorption, how taking drugs over prolonged periods of time may change, etc) is outside the scope of the present thesis, we provide an approach that given such values (and the more we may know about them in time) can

offer personalised advice on how to schedule drugs to minimise the effect of their interactions.

6.2 Ontology pharmacokinetic parameters

The Beers Criteria ontology provides a knowledge base to support health professionals in taking decisions and provides relevant data for considering several situations. As seen in earlier chapters, our framework so far uses the ontology to identify PIMs, classify them according to the Beers Criteria PIMs type, and provide alternative drugs (if available) to avoid them. A further approach to enhance our framework consists of allowing drug scheduling to maximise drug efficacy whilst minimising the interactions. The SMT scheduling model considers the TMAX values as a parameter for the scheduling process; hence, these values have to be formalised in the ontology in order for the solver to be able to consider them.

Each drug class in the ontology has to be enriched with a TMAX value, that is, a value which corresponds to the time when CMAX is expected to be reached for that drug class. This is given by the data property hasCmaxTime (cf. Figure 3.4) which assigns the value TMAX to a drug class. For example, after taking Paracetamol, it takes on average 60 min for paracetamol to reach CMAX. Consequently, in its ontology class hasTmaxTime is assigned the value 60.

6.3 Rescheduling measures to minimise the interaction

Our SMT scheduling model aims to minimise the interaction among drugs by rescheduling them, maximising the distance between their corresponding TMAX values. This is different from current practice where the usual way of minimising drug interaction is by maximising the distance between administration times. To exemplify the usual way of minimising drug interaction, let us consider the following scenario: two drugs (Drug A and Drug B) have been prescribed to a patient with the recommendation to be administered twice a day with an interval of twelve hours. That means for example that if one of the drugs is given at 8 AM, then it has to be given again at 8 PM. However, let us assume that these drugs have an interaction and they are rescheduled to maximise their administration times. Considering that each drug has to be given twice daily, the best we can

do is to keep their administration six hours appart. Figure 6.2 illustrates the administration of Drug A and Drug B in one day (24 hour period) in a way that maximises the distance between their administration. In this solution, Drug A would be administered at 12 AM and 12 PM, and Drug B at 06 AM and 06 PM.

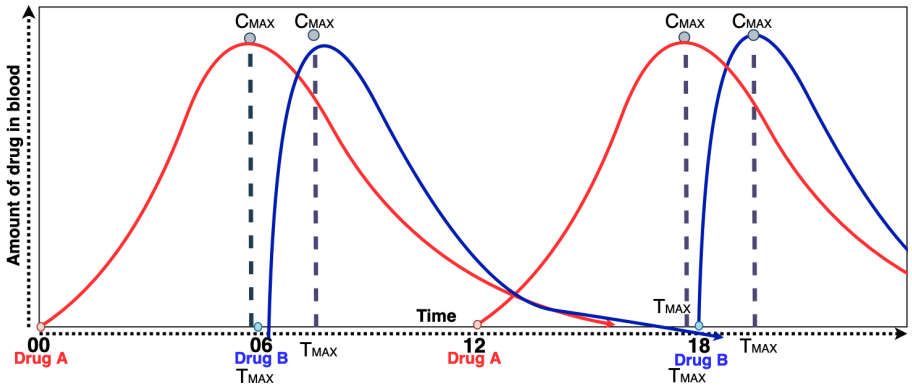


Figure 6.2: Prescriptions schedule.

Figure 6.2 also shows when each drug instance reaches its corresponding CMAX value. For Drug A (red curve) this is around six hours after being given, whereas for Drug B (blue curve) this is in less than two hours. When considering the times in which drugs achieve their CMAX value (given by their TMAX), it is clear that maximising drug administration in this case has the opposite effect than desired, as the distance between the drug's TMAX values was not maximised. In other words, ignoring TMAX values when rescheduling drugs may lead to worse outcomes. Conversely, an approach that considers TMAX values for interacting drugs is likely to be more accurate and safer.

Figure 6.3 shows the result of rescheduling the drug instances from the previous example, considering TMAX. In this scenario, Drug A is administered at 00 and 12, while Drug B is administered at 10 and 22. The administration's distance is two hours; however, the distance between TMAX points is six hours.

The rescheduling measure to minimise the interaction that considers TMAX was shown to be more effective than only considering administration times. It can be straightforward to reschedule two drugs; however, when more drugs are added and more constraints must be considered (such as fixed administration times or meal times), this quickly becomes complex and time-consuming.

To enable automated drug rescheduling considering all necessary constraints and drugs, we adopt an SMT-based approach in line with what we have done before

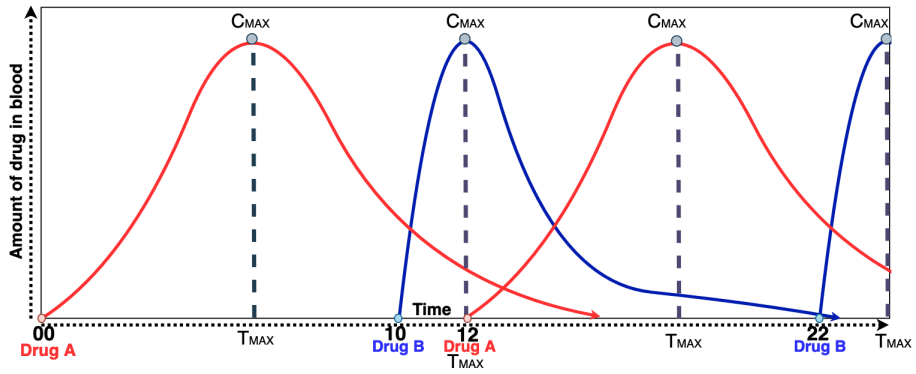


Figure 6.3: Rescheduling prescriptions.

for detecting interactions but where the power of arithmetic's of SMT solvers is now been used. Before explaining the approach, we describe all constraints that must be considered in the model.

6.4 Rescheduling constraints definition

The rescheduling model includes a set of constraints that must be satisfied to optimise the distance between TMAX points. Constraints are logical conditions or restrictions inflicted on the variables in their formulae, and can be used to specify relationships, conditions, or limitations that must be satisfied by the variables to find a satisfying solution to the formulae. Here, we can use them to formulate conditions over our TMAX values.

The first and main constraint is the interaction between drugs. An interaction can lead to undesired effects, such as increasing or decreasing the concentration of a drug, which may result in clinical toxicity or decrease the therapeutic efficacy of the individual drugs. Interactions can happen among two or several drugs. For example, consider that a patient has a prescription containing a set of drugs $\{A,B,C,D\}$. Let their assigned daily administration times (in a 24 hour period) be given by *ha*, *hb*, *hc* and *hd* respectively. Let us assume that A and B interact, as do C and D. Consider the following constraints:

Constraint 1 $\exists_{h_a,h_b}(A(ha) \wedge B(hb) \wedge \neg B(ha) \wedge \neg A(hb))$

Constraint 2 $\exists_{h_c,h_d}(C(hc) \wedge D(hd) \wedge \neg D(hc) \wedge \neg C(hd))$

Constraint 1 above makes sure that drugs A and B are not administrated at the

same time. This can also we expressed as:

$$\exists_{h_a \neq h_b}(A(ha) \wedge B(hb))$$

Similarly, **Constraint 2** does this for drugs C and D. There are also cases where more than two drugs interact simultaneously. For example, if we now assume that A, B and C interact, **Constraint 3** below enforces that the drug times for A, B and C do not coincide, though nothing is said about hd which can potentially coincide with either of the other administration times.

Constraint 3 $\exists_{h_a,h_b,h_c,h_d}(A(ha) \wedge B(hb) \wedge C(hc) \wedge D(hd) \wedge \neg A(hb) \wedge \neg A(hc) \wedge \neg B(ha) \wedge \neg B(hc) \wedge \neg C(ha) \wedge \neg C(hb))$

Besides formalising drug interactions as above (and further formalising distances between drug administration times and making sure they are maximal), additional constraints have to be considered for each drug in a prescription, such as the drug frequency and fixed times. As previously mentioned, drug frequency refers to the number of times the drug will be administered in 24 hours, and fixed times are composed of the hours that this drug must be administered, for example, before or after meals. The following First order logic (FOL) formula was defined to formalise the frequency constraint of three times per day for drug A (ha) and imposes fixed times for drug B, here $hb1$ and $hb2$, at 10 and 22 respectively.

Let $ha, hb1, hb2$ be times (hours) within a day, i.e., $0 \leq ha, hb1, hb2 \leq 24$.

$$\begin{aligned}\exists_{ha,hb1,hb2}(&A(ha) \wedge A(ha+8) \wedge A(ha+16) \wedge \\ &B(hb1) \wedge (hb1 == 10) \wedge B(hb2) \wedge (hb2 == 22))\end{aligned}$$

In the formula above, and as in the earlier constraints, we use $D(hd)$ to indicate the time, hd, when drug D is administered. In this example, drug A is administered at time ha, time $ha+8$ and time $ha+16$. For drug B the values are shown explicitly (though could be specified in relation to each other).

Another essential constraint that has to be considered concerns the TMAX time. The rescheduling process aims to maximise the distance between TMAX values of interacting drugs. Therefore, interacting drugs have to be administered at times such that together with their TMAX value, they do not clash on times when their

concentration in the blood (CMAX values) are the highest.

Below, let $Tmax(D,hd) = hd + TMAX$ denote the time that drug D reaches its CMAX value if given at time hd. For instance, $Tmax(A,ha) = ha + 2$ would suggest that the CMAX for A is reached within two hours of administration and hence at time $ha + 2$ with respect to administration time ha. The following constraints were added to the previous formulae:

$$Tmax(A,ha) = ha + 2 \ \wedge \ Tmax(B,hb1) = hb1 + 1 \ \wedge \ Tmax(B,hb2) = hb2 + 1$$

which describe when CMAX is reached for A and B with respect to their administration times (for A as mentioned before, whereas two hours for B).

In addition, when scheduling these two drugs we want to ensure that their CMAX values are not happening at the same time. This means that our solver will find assignments for the variables in such a way that the following constraint is met:

$$\exists_{ha,hb1,hb2}((Tmax(A,ha) \neq Tmax(B,hb1)) \wedge (Tmax(A,ha) \neq Tmax(B,hb2)))$$

As mentioned before, we also have to ensure that interacting drugs are spaced out as much as possible. In other words, we do not just want possible values for ha, $hb1$ and $hb2$ (for our example), but values that mean that the TMAX for drugs A and B are as distant as possible. Hence, we need to assign values for each interval between drug Tmax instances that have interaction and must be maximised. Therefore, we defined the following predicates:

- **Drug(x)** : x is a drug
- **Interval(x)** : x is an interval
- **LessThanOrEqual720(x)** : $x \leq 720$

The Formula 6.1 defines the value for each interval between drug instances that have interaction. The interval is always defined between two drug instances; therefore, if there are two drug instances of two drugs that have interaction, four intervals will be defined.

$$\begin{aligned}
&\forall x \forall y \Big(\text{Drug}(x) \land \text{Drug}(y) \land \text{Interval}(\text{interval}) \\
&\implies \Big(\text{LessThanOrEqual720}((x-y) \mod 1440) \\
&\qquad \implies \text{interval} = (x-y) \mod 1440 \Big) \\
&\lor \Big(\neg \text{LessThanOrEqual720}((x-y) \mod 1440) \\
&\qquad \implies \text{interval} = (y-x) \mod 1440 \Big) \Big)
\end{aligned} \tag{6.1}$$

The Formula 6.1 can be breakdown as follows:

1. Quantifiers:
   - $\forall x$: For all values of x.
   - $\forall y$: For all values of y.
2. Predicates:
   - $\text{Drug}(x)$: x is a drug.
   - $\text{Drug}(y)$: y is a drug.
   - $\text{Interval}(\text{interval})$: `interval` is an interval.
   - $\text{LessThanOrEqual720}(x)$: x is less than or equal to 720.
3. Implication:
   - $\implies$: Implies or implies that.
4. Logical Structure:
   - The formula asserts that for all drugs x and y and for any interval `interval`, if x and y are drugs and `interval` is an interval, then either:
     - If $(x-y) \mod 1440$ is less than or equal to 720, then `interval` is equal to $(x-y) \mod 1440$.
     - Otherwise, if $(x-y) \mod 1440$ is greater than 720, then `interval` is equal to $(y-x) \mod 1440$.

This formula captures a logical relationship between drugs x and y and an interval `interval`. It states that the interval is determined based on the value of $(x-y)$

mod 1440 and whether it is less than or equal to 720. The OR condition allows for two possibilities depending on the comparison result. If $(x-y) \mod 1440$ is less than or equal to 720, then `interval` is set accordingly. If it is not less than or equal to 720, then `interval` is set based on $(y-x) \mod 1440$.

In summary, our examples have highlighted the types of rules that we have to formalise and convert to the SMT solver:

- Drug frequency and the interval between drugs
- Drug with fixed time
- Drug administration must be in a 24-hour windows time
- Interval between the administration time and the Tmax time

The examples above demonstrate the process of formalising constraints that have to be given to the SMT solver before a solution (rescheduling) can be determined. As mentioned, additional constraints (meal times, patient preference, hospital routines) are formalised (if required) with the same principles. The following section details how these constraints are converted into the SMT code and how the distance between TMAX points for interacting drugs are maximised.

6.5 A SMT model for rescheduling drugs

The formalisation of the schedule constraints in FOL offers a thorough explanation of how the rules must be specified for the SMT model. In this section, we will detail how the constraints were converted into SMT.

6.5.1 Model constraints

The SMT model consists of variables, constraints, and rules or axioms that define the model assumptions. The SMT solver analyses the model and attempts to find a satisfying assignment of values to the variables that satisfies all the constraints. Additionally, to handle constraints, SMT optimisation extends the capabilities of SMT solvers to find optimal solutions that maximise or minimise an objective function. SMT solver optimisation searches for the optimal solution by iteratively refining the assignment of values to the variables, considering both the constraints and the objective function.

The SMT rescheduling model consists of a sequence of steps. First, we define the drug instances with their constraints, then we define the interaction rules, and finally, we define the objective function to maximise the distance between TMAX points and check the results.

6.5.1.1 Declaring drugs and constraints

The drug declarations consist of declaring an instance for each time that a drug is administered. For example, if drug A is administered three times a day, then three instances A1, A2 and A3 of this drug will be declared. Additionally, for each drug instance is also declared a Tmax instance, in which for drug A would be A1Tmax, A2Tmax and A3Tmax. Moreover, the Tmax value is the difference between the drug and Tmax instances, for example, $A1 - A1Tmax$. Finally, there are drugs in which the administration time is fixed, therefore, a value is assigned for the drug instance. For example, let us suppose that drug "B" has to be administered at 8 AM and 8 PM, hence, the instances would be assigned as $B1 = 8$ and $B2 = 20$. The process of declaring drugs and constraints is implemented in Algorithm 7.

The Algorithm 7 receives the variables *drugName*, *interval*, *schedule*, *fixedtime*, and *Tmax*. The *drugName* refers to the name of the drug, *interval* refers to the number of times that the drug has to be administered, *schedule* is a list of administration times for the drug instances in the prescription. *fixedtime* if the *schedule* times are fixed, that means, need to be administered at the defined times and finally *Tmax* refers to the value of the Tmax.

The Algorithm 7 starts by checking if the drug has to be administered at fixed times(row 2). If it is not, then the interval(in minutes) between each instance is assigned to the variable *drugInterval*(row 3). Next, it is defined the *drugTmax* and the *drug* for each instance, both variables are defined as Integer (rows 6 and 7). Then, it is checked if there is more than one instance of the drug; if there is, then the distance between the previous and current instances must be equal to the *drugInterval* (rows 8 and 9). After, is defined the value of *drug* considering the Tmax value and 24-hour windows, hence, it is calculated with the mod(%) operator in 1440 minutes, which corresponds to 24 hours (row 10). Thereafter, it is defined that drug and drugTmax, must be assigned with values between 24-hour windows (rows 11 and 12). Finally, is defined that the drug should be administered in even hours (row 13), however, it is not a mandatory constraint hence, a soft constraint was defined for it.

Algorithm 7: Algorithm for creating drug instances

```
1  def createDrugInstances(drugName, interval, schedule, fixedtime,
     Tmax):
2      if fixedtime == 'N' then
3          drugInterval = 24/interval × 60
4          interval = interval + 1
5          for x in range(1, interval) do
6              drugTmax = Int(drugName(x)Tmax)
7              drug = Int(drugName(x))
8              if x > 1 then
9                  s.add(drug − drugName(x − 1) == drugInterval)
10             s.add(drug == (drugTmax − Tmax)%1440)
11             s.add(drug ≤ 1440, drug ≥ 1)
12             s.add(drugTmax ≤ 1440, drugTmax ≥ 1)
13             s.add_soft(drug%120 == 0)
14     else
15         schedindex = 0
16         interval = interval + 1
17         for x in range(1, interval) do
18             drugTmax = Int(drugName(x)Tmax)
19             drug = Int(drugName(x))
20             s.add(drug == (drugTmax − Tmax)%1440)
21             s.add_soft(drug%120 == 0)
22             s.add(drug == (int(schedule[schedindex])) × 60)
23             schedindex+ = 1
```

Drugs that are administered in fixed time are declared from rows 15 to 23. The *drug* and *drugTmax* instances are declared in rows 18 and 19 , the distance between the *drug* and the *drugTmax* is calculated in row 20, and the even hours soft constraint are defined in row 21. Finally, it is assigned the fixed time from the *schedule* list for each drug instance in row 22.

6.5.1.2 Drug interaction Rules

The drug interaction rules establish the assertions that aim to increase the distance (interval) between the administration of drugs with interactions. Therefore, the administration time interval between drug instances is calculated. Additionally, for each interval between interacting drugs, the interval is calculated and assigned a variable named Interval.

The Algorithm 8 details how the intervals are created. The function receives the interval name *name* as a parameter. Then, an interval of type integer is defined in which the interval *name+intervalIndex* is a sequential number. Finally, the function returns the variable *interval*.

Algorithm 8: Algorithm for creating an interval

```
1 def createInterval(name):
2     intervalIndex+ = 1
3     interval = name(intervalIndex)
4     interval = Int(interval)
5     returninterval
```

The Algorithm 9 assigns the interaction constraint for each interval as previously explained. The algorithm receives as a parameter a dictionary *dicDrug* with a pair of drugs, with the drug name and administration frequency, for example, Drug A, three times a day. These drug's name and frequency are assigned to variables(rows 2 to 6). Then, the $drug1\ Tmax$ and $drug2\ Tmax$ names are defined in the for loops(rows 7 to 10). Next, a variable *interval* is defined for each interval between two drug instances. Finally, it is assigned the constraint for each *interval* and declared a constraint(Distinct) that the drug instances cannot have the same value.

Algorithm 9: Algorithm for creating drug rules

```
1  def createDrugRules(dicDrug):
2      drugs = list(dicDrug.keys())
3      drug1Name = drugs[0]
4      drug1Freq = dicDrug[drug1Name]
5      drug2Name = drugs[1]
6      drug2Freq = dicDrug[drug2Name]
7      for x in range(1, int(drug1Freq)+1) do
8          drug1 = drug1Name(x)Tmax
9          for i in range(1, int(drug2Freq)+1) do
10             drug2 = drug2Name(i)Tmax
11             interval = createInterval('interval')
12             s.add(If((drug1 − drug2)%1440 ≤ 720, interval ==
                 (drug1 − drug2)%1440, interval == (drug2 − drug1)%1440))
13             s.add(Distinct(drug1, drug2))
```

6.5.1.3 Maximising the distance between interacting drugs

Once all the constraints and intervals are defined, the objective function *maximise* is defined to maximise the distance between intervals. The intervals are grouped in a list, which is passed as a parameter to the Algorithm 10. In the algorithm, the intervals are summed and assigned to the variable obj(row 2). Then, this variable is defined in the object function maximise in row 3, in which the SMT solver aims to solve the constraints and maximise the value of each interval.

Algorithm 10: Algorithm for maximising distance between intervals

```
1 def createDrugRules(intervalList):
2     obj = Sum(intervalList)
3     s.maximize(obj)
```

6.6 Example of rescheduling

To provide a realistic example of how the solver defines each interval, we defined an example in Table 6.1. In columns D1 and D2, we give some fictional values (in minutes) for drugs D1 and D2. Then, we calculate the difference between D1 and D2 in column 3. Next, we calculate the modulo of the result considering 1440 minutes (24 hours). The modulo operation can be used for calculations involving cycles or repetitions; in this case, it represents a 24-hour window.

	D1	D2	- D1 D2	mod (- D1 D2) 1440	(ite (<= (mod (- D1 D2) 1440) 720)	THEN	ELSE	Result
1	60	1440	-1380	60	THEN	**60**	1380	60
2	120	1380	-1260	180	THEN	**180**	1260	180
3	420	1080	-660	780	ELSE	780	**660**	660
4	480	1020	-540	900	ELSE	900	**540**	540
5	1020	480	540	540	THEN	**540**	900	540
6	1080	420	660	660	THEN	**660**	780	660
7	1380	120	1260	1260	ELSE	1260	**180**	180
8	1440	60	1380	1380	ELSE	1380	**60**	60

Table 6.1: Rescheduling interaction rule

Afterwards, we define the *ite* (if-then-else) function to compare if the result of $(mod(-D1D2)1440)$ is lower or equal to 720 (12 hours), which corresponds

to the maximum possible distance between two instances. If the result of the *ite* function is lower than 720, then the value from column THEN is assigned to the variable interval. The THEN column corresponds to the assertion $(= interval1(mod(-D1D2)1440))$, which is the one defined in the *ite* function. However, if the result is higher than 720, then the value from the column ELSE is assigned. The ELSE column corresponds to the inverse assertion of the *ite* function.

Figure 6.4 below illustrates three examples of drug scheduling. The first and the second correspond to rows 2 and 4 from Table 6.1. In the first example, the interval to maximise is between D2 and D1, with value 180 (THEN column). In the second example, the interval that has to be maximised is between D1 and D2, with value -540 (column ELSE). In the third example, the drugs are already optimised and do not require further maximisation.

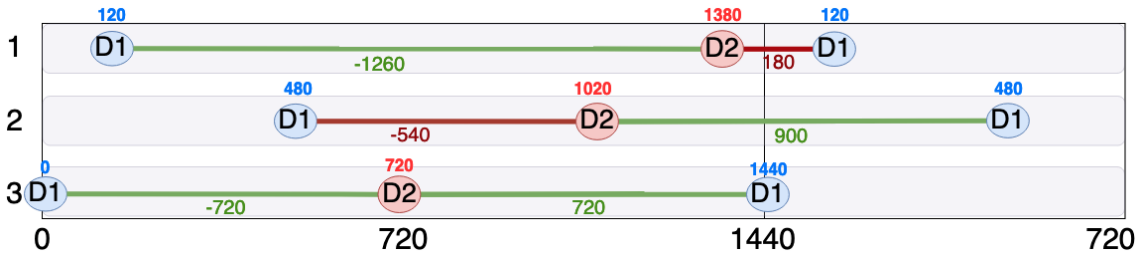


Figure 6.4: Drug Scheduling.

Now, let us consider a set of three drugs, $\{A, B, C\}$, with the following constraints:

- **Constraint 1:** Drug A has to be administered three times in 24 hours with an interval of 8 hours
- **Constraint 2:** Drug A reaches Tmax in 2 hours
- **Constraint 3:** Drug B has to be administered at fixed times 8 AM and 8 PM
- **Constraint 4:** Drug B reaches Tmax in 1 hour
- **Constraint 5:** Drug C has to be administered two times in 24 with an interval of 12 hours
- **Constraint 6:** Drug C reaches Tmax in 1 hour
- **Constraint 7:** Drug A interacts with Drug B
- **Constraint 8:** Drug A interacts with Drug C

- **Constraint 9:** Drug B interacts with Drug C
- **Constraint 10:** The administration time has to be preferably in even hours in accordance with the hospital routine

In the given example, interactions exist amongst Drug A, Drug B and Drug C. According to the constraints, Drug A must be administered thrice daily, which is declared by three instances. There are also TMAX variables for each drug instance. For example, Drug A has instances A1, A2 and A3 with corresponding TMAX instances A1Tmax, A2Tmax and A3Tmax, and similarly for the other drugs. The drug instance A1 is equals A1Tmax minus 120. The definition of the constraints for Drug B has to consider fixed times instead of intervals. Therefore, instances have to be defined as B1 (9 hours) and B2 (21 hours).

Drug A has three instances and Drug B two, consequently, six intervals between all drug instances have to be created. The rescheduling model has to assume a 24-hour (1440 minutes) window to calculate the interval difference between drugs. For example, if Drug A is administered at 11 PM and Drug B at 2 AM, the difference between these instances could be either 3 or 21 hours. In other words, if we calculate the distance between 11 PM to 2 AM we get 3 hours, while if we calculate it between 2 AM and 11 PM we obtain 21 hours. Nevertheless, the rescheduling model considers the lowest distance, which can be a maximum of 12 hours (or equivalently 720 minutes). The same approach has to be defined for the interaction between Drug B and Drug C

A possible solution(s) for the example consists of value assignments for each instance that satisfy all constraints (cf. Table 6.2). The table values are represented visually in Figure 6.5. The results demonstrated that the solver was able to avoid overlapping TMAX values. The minimum distance between Drug A and Drug B instances is 1 hour, between Drug A and Drug C is 2 hours and between Drug B and Drug C is 5 hours. The soft constraint, which states that the administration time should be in even hours, was partially satisfied as Drug C instances were assigned to be administered on odd hours.

Drug	Instance	Administration time	Tmax time
A	1	4	6
	2	12	14
	3	20	22
B	1	8	9
	2	20	21
C	1	3	4
	2	15	16

Table 6.2: Rescheduling solver result

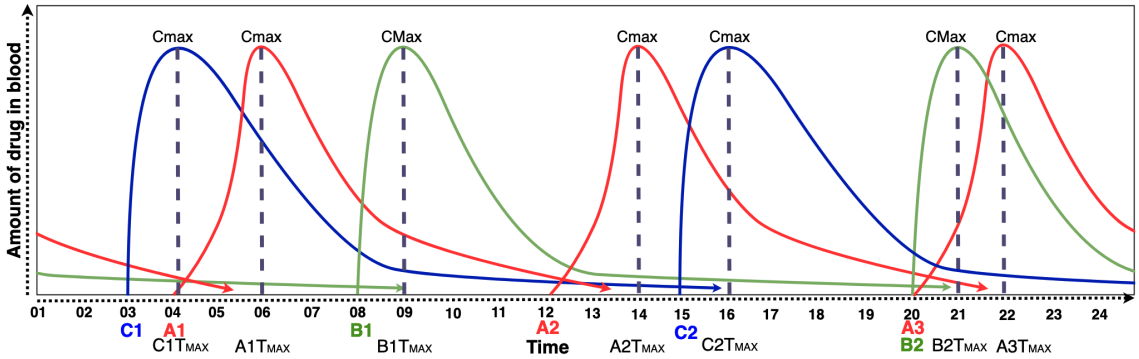


Figure 6.5: Rescheduling prescription result.

6.7 Summary

In this chapter, we presented a novel rescheduling SMT solver-based approach to minimise drug interaction problems. Additionally, we explained how the schedule constraints were formalised in FOL and converted into SMT code. Finally, we illustrate and discuss the solver results for a hypothetical scenario.

Details on the experiments we conducted are discussed separately in the next chapter. This includes all experiment's inputs and results for the rescheduling approach discussed in this chapter.

CHAPTER SEVEN

FRAMEWORK DEVELOPMENT AND TESTING

In this chapter, we discuss the development and integration of the various components that make up the CDSS framework. We previously covered the formalisation and requirements of each individual component. Here, we demonstrate how both the requirements and formalised rules were implemented into the knowledge base and the inference engines. Additionally, we describe the validation of the adopted approach by executing test cases in order to ensure the correctness and completeness of the framework.

7.1 Framework development

The development of the CDSS framework involved defining requirements, creating a taxonomy, formalising inference rules and constraints, and integrating several reasoning engines to address different possible outcomes. Figure 7.1 illustrates the CDSS framework internal information flow. The dataflow starts from the Input module, in which the information that composes the Knowledge base is provided and from the hospital EMR, in which the patient data is submitted to the CDSS framework.

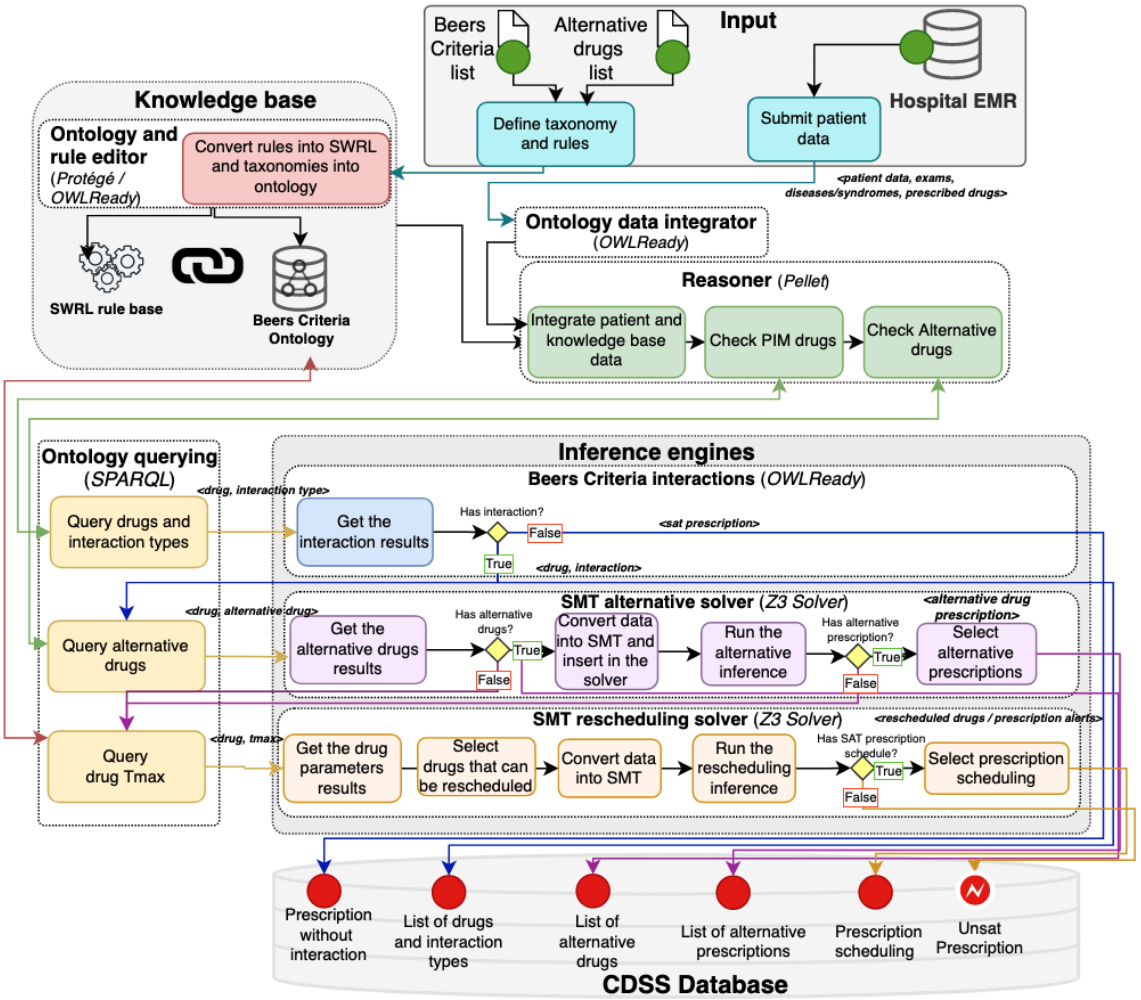


Figure 7.1: The internals of the CDSS framework

The knowledge base comprises the Beers Criteria Ontology and the SWRL rule base. The Beers Criteria Ontology defines the taxonomy, objects and data properties and its rules (e.g. disjoint, equivalence, domain, range). The SWRL rule base consists of inference rules for drug inappropriate drugs and alternative drugs. Both components were developed and overseen by the Ontology and Rule Editor within the Protégé tool [1]. As the ontology/rules can be edited, and this is a shared ontology, it would need version control and an update protocol to ensure correctness when updated. The process of converting the Beers Criteria list and Alternative Drugs list is done manually.

The Ontology data integrator then converts these data using the OWLReady[1] [85]

[1]Owlready is an OO Programming library in Python with automatic classification and high-level constructs for biomedical ontologies

library to integrate them with the inference engine components. Next, the ontology Reasoner integrates these data with the Knowledge base. The Pellet[2] [142] reasoner was selected due to its functionality to check the consistency of ontologies, compute the classification hierarchy, incorporate Semantic Web Rule Language (SWRL) rules in reasoning, explain inferences, and answer Simple Protocol and RDF Query Language (SPARQL) queries. Moreover, Pellet does reasoning over object and data properties, unlike, for example, Hermit, which only considers object properties. The Reasoner then infers patient data over the Knowledge base aiming to detect interactions and alternative drugs.

After performing the reasoner, a SPARQL query is performed into the reasoner assertions to check if interactions were detected. Thereafter, the CDSS checks if interactions are not found, and then a *sat prescription* message is sent to the CDSS Database. However, if interactions are found, they are sent to the SMT alternative solver component. The SMT alternative solver first queries the reasoner to check whether there are alternative drugs. If alternative drugs are found, then interaction rules, the prescription drugs and the alternative drugs are converted for the SMT solver, which was developed in Z3[3] [40]. Z3 solver was selected because it supports datatypes and quantifiers, which were necessary for modelling the alternative constraints model and due to its API with Python. The solver checks which alternative drugs do not have interaction with other drugs and if prescriptions that satisfy all the constraints can be found. If prescriptions are found, they are sent to the CDSS Database; otherwise, the drug interactions are sent to the SMT rescheduling solver component. The component first selects only the interaction between drugs, which are the ones that can be rescheduled, and queries the ontology to get the Tmax value of each drug. Next, the solver reschedule the drugs. If a valid schedule can be found, the solver sends it to the CDSS Database. Additionally, a list of all the interactions that were found and a list of valid alternative drugs are also sent to the CDSS Database. The CDSS framework source code developed in Python can be seen in the Appendix F.

Once the results of our approach are stored in the CDSS database, they can be retrieved and displayed for health professionals to support the decision process. As the Beers Criteria provide the Quality of evidence and the strength of

[2]Pellet is the OWL 2 Description Logics reasoner

[3]Z3 is a theorem prover for solving satisfiability modulo theories (SMT) problems. It supports various input formats, including SMT-LIB, SMTLIB2, and programmatically using APIs in languages such as C++, C#, Java and Python.

recommendations, the PIM alerts could be customised to display only relevant alerts for the professionals. Moreover, the rationale of the PIM drug could be displayed when requested in order to explain the reasons for the drug to be classified as PIM.

7.1.1 Beers Criteria ontology

In Chapter 4, we detail the formalization of the Beers Criteria through an ontology. This section will detail how this ontology was developed and how the inference rules were defined.

Figure 8.1 illustrate the ontology classes taxonomy, object and data properties. To detail how the taxonomy was defined, we expanded some BeersCriteria classes. The BeersCriteria class is composed of the main classes DDDS, DDI, B.PIM, UWC and VLKF. The interaction between drugs and disease or syndrome is defined in the DDDS class. This class is composed of four subclasses which represent a group of diseases. The DDDS_Cardiovascular has two disease subclasses DDDS_Heart_-Failure and DDDS_Syncope. The DDDS_Heart_Failure comprises six subclasses representing the drugs associated with this disease. The same approach was used to define the taxonomy of all other classes that compose the ontology. Additional parameters were defined in the ontology, such as disjoint classes, which means that an individual cannot belong to both classes. For example, a drug cannot be both Ibuprofen and Paracetamol; therefore, a disjoint rule is defined.

The object properties are listed in the second column, and the data properties in the third in Figure 8.1. For these elements, parameters were defined as domain and range. For example, the object property hasDisease has the domain class Prescription and the range class Drug. It means that this object can only link prescriptions with drugs. The same approach was used for data property, but instead of linking two classes, it linked a class with datatypes(e.g. float, int, string).

After defining the ontology elements, inference rules were defined in SWRL to detect and categorise the Beers Criteria interactions. In Section 4.1.8, we provide a fictional example to explain how the link between ontology elements happens and how inappropriate drugs are classified. Here, we will use the same to demonstrate the ontology implementation. To recollect, we had a patient named Tom, 75, male, suffers from a history of falls, has a prescription with four drugs, Metoclopramide, Triazolam, Codeine and Morphine which are administered by the route injection.

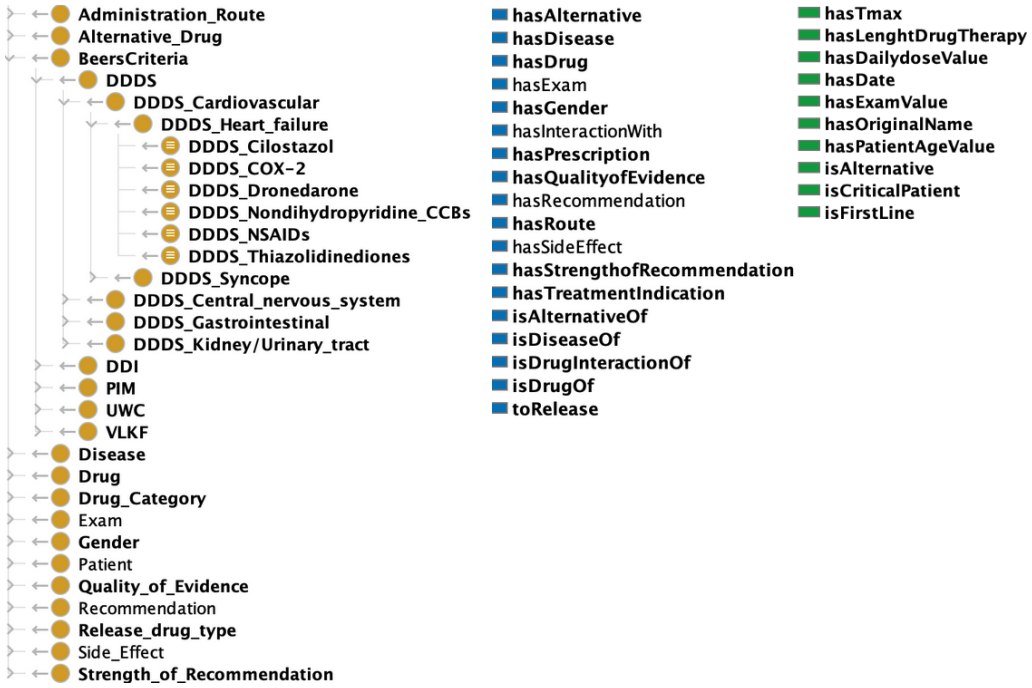


Figure 7.2: Ontology elements

First, to exemplify how all the interaction inference rules were translated from FOL to SWRL(all the ontology SWRL are available in Appendix C), we will consider the DDI_Opioids/Benzodiazepines rule as follows:

```
1Patient(?p) ^ Prescription(?pr) ^ Opiate_Agonists(?d) ^ Benzodiazepines(?d2) ^
2hasPatientAgeValue(?p, ?a) ^ hasPrescription(?p, ?pr) ^ hasDrug(?pr, ?d2) ^
3hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
4-> DDI_Opioids/Benzodiazepines(?d2) ^ hasInteractionWith(?d2, ?d) ^
5DDI_Opioids/Benzodiazepines(?d) ^ hasInteractionWith(?d, ?d2)
```

Listing 7.1: Interaction SWRL rule

.

This inference rule aims to detect interactions between the drug categories Opiate_Agonists and Benzodiazepines. $Patient(?p)$ represents an individual patient, and $Prescription(?pr)$ represents an individual prescription. $Opiate\_Agonists(?d)$ and $Benzodiazepines(?d2)$ represent classes of drugs, with the individual drugs denoted by $?d$ and $?d2$, respectively. $hasPatientAgeValue(?p, ?a)$ indicates the age value of the patient $?p$, while $hasPrescription(?p, ?pr)$ means that the patient $?p$ has a prescription $?pr$. If a prescription $?pr$ includes drugs $?d2$ and $?d$, they are represented by $hasDrug(?pr, ?d2)$ and $hasDrug(?pr, ?d)$, respectively.

The rule states that if a patient is over 64 years old (checked by $swrlb:greaterThan(?a, 64)$), and they are prescribed both *Opiate_Agonists* and *Benzodiazepines*, then there is a drug-drug interaction between these medications represented by $DDI\_Opioids/Benzodiazepines()$. Additionally, the interaction between $?d2$ and $?d$ is bidirectional, represented by $hasInteractionWith(?d2, ?d)$ and $hasInteractionWith(?d, ?d2)$.

After defining the inference rules, the reasoner was executed to assess if inappropriate drugs could be found. The ontology results and the link between classes and individuals are depicted in Figure 7.3. In this example, we illustrated the link between individuals of classes Beers Criteria and Drug Categories. For example, the individual P1_Codeine is linked to class Codeine, and the same happens to other drugs. The items highlighted in yellow represent the assertions made by the reasoner. For example, the individual P1_Codeine is linked to classes DDI_CNS_Active_Drugs/CNS_Active_Drugs and DDI_Opioids/Benzodiazepines in the Description:P1_Codeine. Moreover, the Property assertions:P1_Codeine, shows that this individual also has interaction with P1_Morphine and P1_-Triazolam.

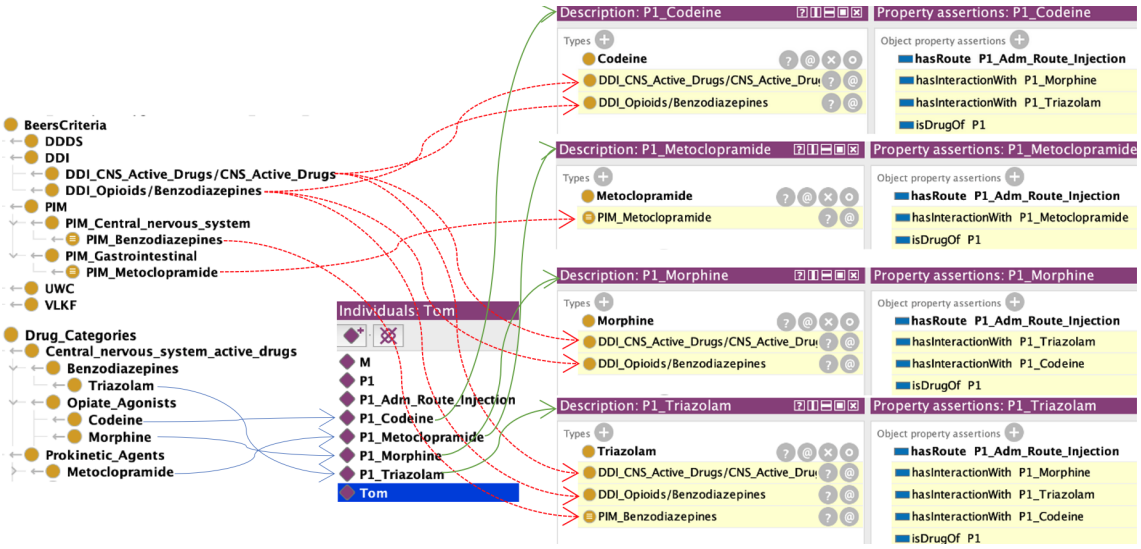


Figure 7.3: Classes and individuals links

7.1.2 Alternative drug solver

The development of the inference engine to find alternative drugs for the prescription involves a two-step process. Firstly, a taxonomy was defined in the ontology and then, inference rules were developed into the Beers Criteria in order to suggest

alternative drugs. Following this, the suggested drugs underwent validation by the SMT model to ensure that they did not conflict with any other prescribed or alternative drugs.

To exemplify how these rules were formalised, we will still consider the example from the previous section. In the example, four alternative drugs were found for the individual P1_Triazolam. The reasoner used the following inference rule to assert these drugs:

```
1Patient(?p) ^ Prescription(?pr) ^ Drug(?d) ^ Anticonvulsants(?d)
2^ Alt_Anticonvulsants(?alt) ^ Drug(?alt) ^ History_of_falls(?pd)
3^ isAlternative(?alt, true) ^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
4^ hasPrescription(?p, ?pr) ^ hasDisease(?p, ?pd) ^ hasPatientAgeValue(?d, ?a)
5  –> hasAlternative(?d, ?alt)
```

Listing 7.2: Alternative SWRL rule

This rule defines the relationship between a patient, a prescription, a specific drug, the patient's age, and the availability of an alternative drug. All the SWRL alternative rules defined in the Beers Criteria ontology are available in Appendix D.

The rule can be breakdown as follows:

- Patient ?*p* represents an individual patient
- Prescription ?*pr* represents an individual prescription
- Drug ?*d* represents the class Drug
- Anticonvulsants ?*d* represents a drug category class
- Alt_Anticonvulsants ?*alt* represents a class of alternative drugs for the drug category Anticonvulsant drug
- History_of_falls ?*pd* represents a medical condition
- isAlternative(?*alt*, *true*) specifies that the drug ?*alt* is an alternative option
- hasDrug(?*pr*, ?*d*) indicates that the prescription ?*pr* includes the drug ?*d*
- swrlb:greaterThan(?*a*, 64) checks if the age value ?*a* is greater than 64, implying that the patient is over 64 years old

- hasPrescription(?p, ?pr) states that the patient ?p has a prescription ?pr
- hasDisease(?p, ?pd) indicates that the patient ?p has a specific disease or condition ?pd
- hasPatientAgeValue(?p, ?a) specifies that the patient ?p has age value ?a

According to the rule, if all the conditions are met, then drugs that belong to class Alt_Anticonvulsants are considered alternative to drugs that belong to the drug category Anticonvulsants. Specifically, the relationship between an Anticonvulsants drug and an Alt_Anticonvulsants drug is represented by the hasAlternative(?d, ?alt) notation. It indicates the alternative drug ?alt can be used instead of the prescribed drug ?d.

Figure 7.4 displays the drug alternatives recommended by the ontology reasoner. The taxonomy Alternative_drug of the selected alternative drugs and the individuals are also displayed It is important to note that in this example, only drugs identified as an alternative to prescription drugs were included in the taxonomy. P1_Triazolam is the only individual drug that has alternatives alternative drugs as illustrated in Property assertions: P1_Triazolam, named alt_Escitalopram, alt_Fluoxetine, alt_Leviteracetam, and alt_Pregabalin,linked by the object property hasAlternative.

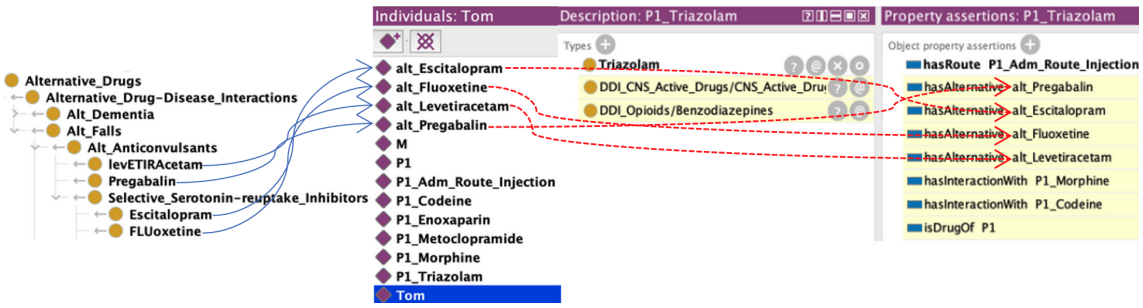


Figure 7.4: Drug alternatives recommended by the reasoner.

Based on the prescribed drugs and the ontology reasoner assertions, the following constraints have to be considered in the SMT Alternative Solver:

Drug Interactions:

- P1_Codeine x P1_Enoxaparin
- P1_Codeine x P1_Morphine

- P1_Codeine x P1_Triazolam
- P1_Metoclopramide x Patient age > 65

Alternative drugs:

- P1_Triazolam x alt_Escitalopram
- P1_Triazolam x alt_Fluoxetine
- P1_Triazolam x alt_Leviteracetam
- P1_Triazolam x alt_Pregabalin

Precription:

- Number of drugs = 4

These constraints are extracted from the ontology reasoner by a SPARQL query demonstrated in the Apendix E.4. Then these constraints are inserted in the SMT Alternative solver as explained in Chapter 5. In Chapter 5, we demonstrated how the data is converted and inserted into the SMT solver with Python algorithms. Here, we will demonstrate how drugs and constraints are declared directly in SMT Lib. Initially, the drug variables of the prescription and alternative are declared in Listing 7.3, which are defined as a datatype Drug (row 1). Then, are declared the constants variables to define that the result must have the same number of drugs as the prescription (rows 2-5). Finally is defined the function Choice which takes a single argument of type Drug and returns a Boolean value (row 6).

```
1 declare-datatypes ((Drug 0)) (((alt_Pregabilin) (P1_Metoclopramide) (P1_Triazolam) (
      P1_Codeine) (alt_Escitalopram) (alt_Levetiracetam) (alt_Fluoxetine) (P1_Morphine
      ))))
2 (declare-fun Drug0 () Drug)
3 (declare-fun Drug1 () Drug)
4 (declare-fun Drug2 () Drug)
5 (declare-fun Drug3 () Drug)
6 (declare-fun choice (Drug) Bool)
```

Listing 7.3: Declaring Drug datatype instances

A constraint is defined thereafter in Listing 7.4, asserting that the drug constants have to exist with a true value (rows 1-9). Then are defined drugs which do not have alternatives and must be true (rows 10-15). Next, are defined the drug interactions constraints (rows 16-18) and finally are defined the alternative drugs constraints (rows 19-22).

```
1 (assert (exists ((Drug0 Drug)) (choice Drug0)))
2 (assert (and (= (choice Drug0) true)))
3 (assert (exists ((Drug1 Drug)) (choice Drug1)))
4 (assert (and (= (choice Drug1) true)))
5 (assert (exists ((Drug2 Drug)) (choice Drug2)))
6 (assert (and (= (choice Drug2) true)))
7 (assert (exists ((Drug3 Drug)) (choice Drug3)))
8 (assert (and (= (choice Drug3) true)))
9 (assert (distinct Drug0 Drug1 Drug2 Drug3))
10 (assert (and (= (choice P1_Metoclopramide) true)))
11 (assert (exists ((Drug0 Drug)) (choice P1_Metoclopramide)))
12 (assert (and (= (choice P1_Codeine) true)))
13 (assert (exists ((Drug1 Drug)) (choice P1_Codeine)))
14 (assert (and (= (choice P1_Morphine) true)))
15 (assert (exists ((Drug2 Drug)) (choice P1_Morphine)))
16 (assert (or (not (choice P1_Codeine)) (not (choice P1_Morphine))))
17 (assert (or (not (choice P1_Codeine)) (not (choice P1_Triazolam))))
18 (assert (or (not (choice P1_Morphine)) (not (choice P1_Triazolam))))
19 (assert (xor (choice P1_Triazolam) (choice alt_Escitalopram)))
20 (assert (xor (choice P1_Triazolam) (choice alt_Fluoxetine)))
21 (assert (xor (choice P1_Triazolam) (choice alt_Levetiracetam)))
22 (assert (xor (choice P1_Triazolam) (choice alt_Pregabilin)))
```

Listing 7.4: Declaring drug constraints

After declaring the variables and the constraints, the solver checks whether these alternative drugs satisfy all the constraints. In this example, the solver returns a UNSAT result, meaning a valid prescription could be found. Therefore, drugs that do not have alternatives have to be rescheduled, aiming to minimise the interaction.

7.1.3 Rescheduling solver

The rescheduling inference engine uses the TMAX value as a parameter to maximise the time between CMAX of the different drugs. Hence, the TMAX values and the administration times or frequency of administration of each drug are added to the SMT solver. The solver seeks to maximize the distance between the TMAX values of the drug interaction to minimize the interaction and, consequently, the adverse effects that can be caused as detailed on Section 6.3. Moreover, this

value was added to the Beers Criteria ontology by the data property hasTmax. Figures 7.5 and 7.6 show how this value(in minutes) was added to the drug classes Codeine and Morphine.

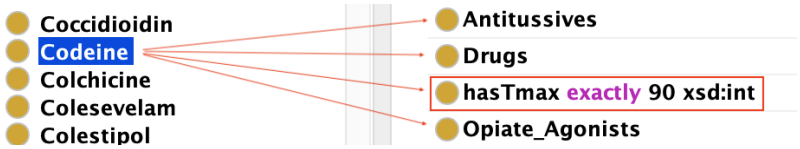


Figure 7.5: Codeine Tmax data property

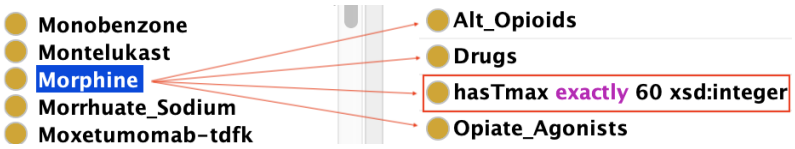


Figure 7.6: Morphine Tmax data property

For the rescheduling solver, it is considered only interaction between drugs(DDI). Hence, these drugs are selected from the ontology using the query demonstrated in Appendix E.3. In the following example, we will detail the rescheduling model in SMT Lib for drugs Codeine and Morphine. In this scenario, both drugs were prescribed to be administered twice a day; thus, two instances of each drug (P1Codeine(1/2) and P1Morphine(1/2))and of the Tmax P1Codeine(1/2)Tmax and P1Morphine(1/2)Tmax and the intervals between these drug instances were declared as follows:

```
1 (declare-fun P1Codeine1Tmax () Int)
2 (declare-fun P1Codeine1 () Int)
3 (declare-fun P1Codeine2Tmax () Int)
4 (declare-fun P1Codeine2 () Int)
5 (declare-fun P1Morphine1 () Int)
6 (declare-fun P1Morphine1Tmax () Int)
7 (declare-fun P1Morphine2 () Int)
8 (declare-fun P1Morphine2Tmax () Int)
9 (declare-fun interval1 () Int)
10 (declare-fun interval2 () Int)
11 (declare-fun interval3 () Int)
12 (declare-fun interval4 () Int)
```

Listing 7.5: Drug instances declaration

Thereafter, the values of Tmax were defined for each drug. In row 1 of Listing 7.6, $P1Codeine1$ was assigned with $mod(-P1Codeine1Tmax90)1440$. It means that Tmax is reached after 90 minutes of being administrated. The drug administration

schedule considered 24 hours(1440 minutes); hence, the *mod* operator was used to calculate the value within this time period.

```
(assert (= P1Codeine1 (mod (- P1Codeine1Tmax 90) 1440)))
(assert (<= P1Codeine1 1440))
(assert (>= P1Codeine1 1))
(assert (<= P1Codeine1Tmax 1440))
(assert (>= P1Codeine1Tmax 1))
(assert (= (- P1Codeine2Tmax P1Codeine1Tmax) 720))
(assert (= P1Codeine2 (mod (- P1Codeine2Tmax 90) 1440)))
(assert (<= P1Codeine2 1440))
(assert (>= P1Codeine2 1))
(assert (<= P1Codeine2Tmax 1440))
(assert (>= P1Codeine2Tmax 1))
(assert (and (<= P1Morphine1 1440) (>= P1Morphine1 1)))
(assert (<= P1Morphine1Tmax 1440))
(assert (>= P1Morphine1Tmax 1))
(assert (= P1Morphine1 (mod (- P1Morphine1Tmax 60) 1440)))
(assert (= P1Morphine1 480))
(assert (and (<= P1Morphine2 1440) (>= P1Morphine2 1)))
(assert (<= P1Morphine2Tmax 1440))
(assert (>= P1Morphine2Tmax 1))
(assert (= P1Morphine2 (mod (- P1Morphine2Tmax 60) 1440)))
(assert (= P1Morphine2 1200))
```

Listing 7.6: Declaration of Tmax for drugs

The interaction constraints between these drug instances are defined in Listing 7.7. Rows 1, 2 and 3 define the interaction between $P1Codeine1Tmax$ and $P1Morphine1Tmax$.

```
(assert (ite (< (mod (- P1Codeine1Tmax P1Morphine1Tmax) 1440) 720)
        (= interval1 (mod (- P1Codeine1Tmax P1Morphine1Tmax) 1440))
        (= interval1 (mod (- P1Morphine1Tmax P1Codeine1Tmax) 1440))))
(assert (distinct P1Codeine1Tmax P1Morphine1Tmax))
(assert (ite (< (mod (- P1Codeine1Tmax P1Morphine2Tmax) 1440) 720)
        (= interval2 (mod (- P1Codeine1Tmax P1Morphine2Tmax) 1440))
        (= interval2 (mod (- P1Morphine2Tmax P1Codeine1Tmax) 1440))))
(assert (distinct P1Codeine1Tmax P1Morphine2Tmax))
(assert (ite (< (mod (- P1Codeine2Tmax P1Morphine1Tmax) 1440) 720)
        (= interval3 (mod (- P1Codeine2Tmax P1Morphine1Tmax) 1440))
        (= interval3 (mod (- P1Morphine1Tmax P1Codeine2Tmax) 1440))))
(assert (distinct P1Codeine2Tmax P1Morphine1Tmax))
(assert (ite (< (mod (- P1Codeine2Tmax P1Morphine2Tmax) 1440) 720)
        (= interval4 (mod (- P1Codeine2Tmax P1Morphine2Tmax) 1440))
        (= interval4 (mod (- P1Morphine2Tmax P1Codeine2Tmax) 1440))))
(assert (distinct P1Codeine2Tmax P1Morphine2Tmax))
(assert-soft (= (mod P1Codeine1 120) 0) :weight 1)
(assert-soft (= (mod P1Codeine2 120) 0) :weight 1)
(assert-soft (= (mod P1Morphine1 2) 0) :weight 1)
(assert-soft (= (mod P1Morphine2 2) 0) :weight 1)
(assert-soft (>= interval1 360) :weight 1)
```

```
22 (assert-soft (>= interval2 360) :weight 1)
23 (assert-soft (>= interval3 360) :weight 1)
24 (assert-soft (>= interval4 360) :weight 1)
```

Listing 7.7: Rescheduling drug interaction declaration

Next, the solver checks for a valid drug schedule considering all the constraints. The result of the solver is listed in Table 7.1.

Drug/Tmax	Minutes	Hours
P1Codeine	90	1:30
P1Codeine1Tmax	180	3
P1Morphine1	480	8
P1Morphine1Tmax	540	9
P1Codeine2	810	13:30
P1Codeine2Tmax	900	15
P1Morphine2	1200	20
P1Morphine2Tmax	1260	21
interval1	360	6
interval2	360	6
interval3	360	6
interval4	360	6

Table 7.1: Rescheduling solver result example

7.2 Inference engine testing

The CDSS Framework validation consists of analysing the correctness and completeness of the inference engines. Several tests were executed in each inference engine, considering the required parameters to validate the framework results. Some examples of data input are listed in the Appendix H

7.2.1 Beers Criteria ontology

In ontology testing, patient and prescriptions inputs were provided to check if the ontology outputs were correct. This test can be classified as Black Box testing [14]. The black box testing focuses on the system's external behaviour. In this case, provide inputs (e.g., ontology terms, relationships between terms) and observe the outputs (e.g., inferred relationships, logical consequences) to check if they align with the expectations and the defined ontology rules. In this context, black box testing aims to validate that the ontology behaves correctly according to its intended design and specification. For example, we could provide inputs like patient age = 65 and prescribed drug = Meperidine and expect the output PIM_Pain_medications. This partially automated process involves checking if the ontology content covers the Beers Criteria list. Additionally, ontology reasoners are used to ensure the consistency of the ontology.

7.2.1.1 Ontology consistency

The consistency test verifies if the ontology is free of inconsistencies and unsatisfactory classes. The Pellet [142] reasoner was employed with the Protégé 5.0 [1] to test the ontology consistency. The reasoner revealed no discrepancies regarding the Beer Criteria ontology.

For example, a class can be considered inconsistent when there is a taxonomy problem. Figure 4.9 illustrated how a Disjoint Classes rule was formalised. For instance, an administration route cannot be both Nasal and Injection at the same time. Therefore, a disjoint rule was created to avoid an individual belonging to these administration routes simultaneously.

To illustrate how the Pellet [142] reasoner checks the ontology consistency, we created a new class named NasalIntection. Figure 7.7 shows that this class is equivalent to Nasal and Injections.

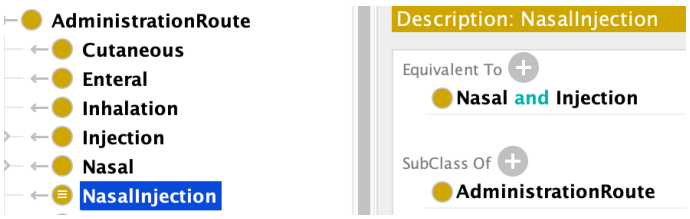


Figure 7.7: Inconsistent Class example

However, these classes belong to a disjoint rule. Hence, a class can be either Nasal or Injection but not both of them at the same time. Consequently, the reasoner highlighted in red an inconsistency in class NasalInjection as shown in Figure 7.8.

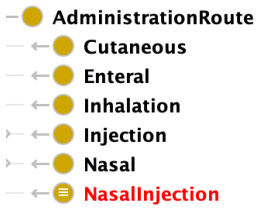


Figure 7.8: Inconsistent class highlighted

Figure 7.9 depicts an individual named InconsistentRoute from type NasalInjection. This individual is also considered inconsistent as it belongs to an inconsistent class.

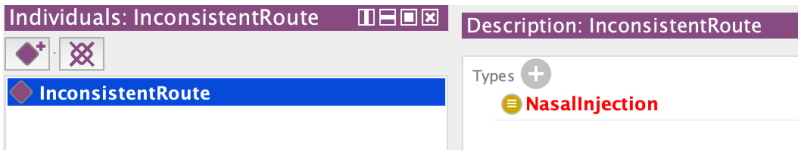


Figure 7.9: Inconsistent individual

As the ontology is inconsistent, it is not possible to do reasoning. Therefore, the reasoner provided an explanation of the inconsistencies that have to be fixed. In the NasalInjection example, it is shown in Figure 7.10 that there is an individual InconsistentRoute from type NasalInjection. There is also a DisjointClassses rule that the classes Intection and Nasal belong to. Finally, it shows the class NasalInjection is equivalent to Nasal and Injection.

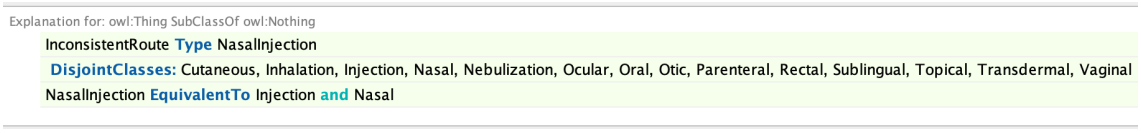


Figure 7.10: Inconsistencies explanation

In order to detect some of the most common pitfalls occurring in the development of ontologies that could lead to modelling errors, the Ontology Pitfall Scanner! (OOPS!) [125] was employed. This tool supports the test activity, providing mechanisms to automatically detect several pitfalls, such as wrong inverse relationships, unconnected ontology elements, and missing annotations. The results demonstrate that no pitfall was detected in the Beers Criteria ontology.

7.2.1.2 Completeness of content coverage - inappropriate medication

We evaluated the ontology content against the defined Beers Criteria. The results showed that the ontology is 100% complete, as it accurately identifies inappropriate medications based on its knowledge.

In order to conduct the test, multiple prescriptions were formulated to replicate each rule of the Beers Criteria. The prescriptions contain the necessary parameters for a drug to be classified as potentially inappropriate. For example, considering the criteria from Figure 4.4, there are listed drugs that belong to the drug class First-generation antihistamines. For this particular group, the only parameter that matters is the patient age being higher or equal to 65. Therefore, a prescription was created with all these drugs and a patient with 65 years. Another prescription was explicitly created for the drug Diphenhydramine, as it is potentially inappropriate only when the administration route is oral.

These prescriptions were then added into the Beers Criteria ontology and integrated with the reasoner. After that, a query was performed with SPARQL to get the information from the ontology(Rationale, recommendation, quality of evidence and Strength of recommendation). Finally, the query results were compared with the Beers Criteria information to find inconsistent or missing information.

7.2.2 Alternative drug solver

The evaluation of the Alternative solver inference engine comprises two steps. The first is evaluating the ontology inference rules to check whether the alternatives suggested by the solver are accurate. The second refers to the SMT model to check if the prescriptions provided by the solver with the alternative drugs are valid. The consistency of the ontology has already been checked, so there is no need to repeat the process.

7.2.2.1 Completeness of content coverage - alternative drugs

To evaluate alternative drugs provided by the ontology inference rules, we adopted a similar approach to check the inappropriate medications. The alternative drug rules were analysed to collect the necessary constraints to classify a drug as an alternative. Then prescriptions were created and integrated into the ontology. Next, the ontology reasoner was performed, and then queries were performed with SPARQL to get the alternative drug for each prescribed drug.

7.2.2.2 SMT model test - alternative drugs

The SMT model consists of three inputs, prescriptions, interactions and alternative drugs. Therefore, a set of prescriptions, interactions and alternative drugs were created to simulate a realistic scenario to check if the SMT model would provide valid prescriptions.

To evaluate the SMT model, we consider the following constraints:

- Number of prescribed drugs x number of drugs suggested
- Interactions not detected
- Suggestion of invalid alternative drugs
- Valid alternative drugs were not suggested

The first constraint refers to the number of drugs suggested by the solver. We assumed that the number of drugs the alternative solver provides must be the same as the original prescription. That means if five drugs were prescribed, the alternative solver has to provide a valid prescription with five drugs too.

In the second constraint, we checked if the solver could detect all the interactions and not allow a drug that two drugs that interact belong to the same prescription. In the following constraint, we check if there are alternative drugs that should not be suggested for a drug. Finally, in the last constraint, we checked if there were valid alternatives that the solver did not consider.

7.2.3 Rescheduling solver

The Rescheduling solver test consists of checking if the results provided are consistent. The following three constraints were validated:

- Drug frequency and the interval between drugs
- Drug with fixed time
- Drugs interaction: drugs where the Tmax times were not maximised

The frequency and interval between drugs refer to the number of times (instances) a drug has to be administered in 24 hours and the interval between each

administration. This test aims to check if the number of drug instances and intervals correspond to the prescription.

The drug with a fixed time constraint refers to a pre-defined administration time. In this case, the solver cannot change the administration time. Only drugs that do not have a fixed time can be rescheduled. In this test, we check if the prescription's fixed time corresponds to the rescheduled prescription.

Finally, the drug interactions constraint refers to drugs that cannot be administered simultaneously. Consequently, the administration time between these drugs has to be maximised. Thus the test aims to check the conflicts in the administration time of these drugs.

To perform the test, we simulate scenarios that comprise these constraints. Several prescriptions with different sets of drugs were defined. For example, with different drug frequencies, fixed times and number of interactions.

7.3 Summary

This chapter demonstrates how the CDSS framework was developed and the performed tests. We detail the information flow between components and how they were integrated. Moreover, we list the tools/libs employed to develop the knowledge base and the inference engines.

Regarding the Beers Criteria ontology, we demonstrate how the requirements and formalised rules from the previous chapters were implemented in the ontology. Moreover, we illustrate the relationship between classes, individuals and properties. Finally, we show how the ontology information was obtained from the ontology and integrated with the inference engines.

We detail how the requirements were converted into constraints in the solvers. First, we detail the alternative solver constraints of drug interactions, and alternative drugs were formalised in the SMT model. Then we demonstrate how the rescheduling solver was implemented and how the requirements related to drug interaction, drug administration and Tmax time were formalised in a SMT model optimisation.

Finally, we detail how each component of the framework was tested in order to consist the correctness and completeness of the framework. The tests aimed to

check whether the results provided by each element of the framework were consistent. The results demonstrated that the framework is detecting the inappropriate drugs according to the Beers Criteria list, suggesting alternative drugs without having interaction with other prescribed drugs and considering all the constraints when drugs are rescheduled.

CHAPTER EIGHT

CDSS EXPERIMENTS AND EVALUATION

This chapter details the experiments conducted with a hospital EMR. Moreover, we evaluate and compare our CDSS framework with other prescription screening tools and research. First, we describe the hospital EMR dataset and introduce some characteristics of patient and prescription profiles. Then some case studies will be conducted with selected patients to demonstrate how our CDSS framework can support health professionals and to compare the results with the hospital CDSS. Moreover, we will evaluate the hospital EMR prescriptions, explain and discuss the outcomes. Finally, we will compare our CDSS framework with other prescription screening tools and with the approaches listed earlier in related work.

8.1 Dataset Analysis

Our dataset was provided by a Brazilian hospital Eletronic Medical Record (EMR). The records are composed of prescriptions, patient data, laboratory exams, vital signs, main underwent procedures, and previous diseases as shown in the Entity Relationship diagram in Figure 8.1. The data is restricted to inpatients over 65 years old and was anonymized by the hospital, excluding any possibility of patient identification. The results may have some impact (positive or negative), which could affect the hospital's image. Therefore, the name of the hospital is also not disclosed.

The dataset tables can be described as follows:

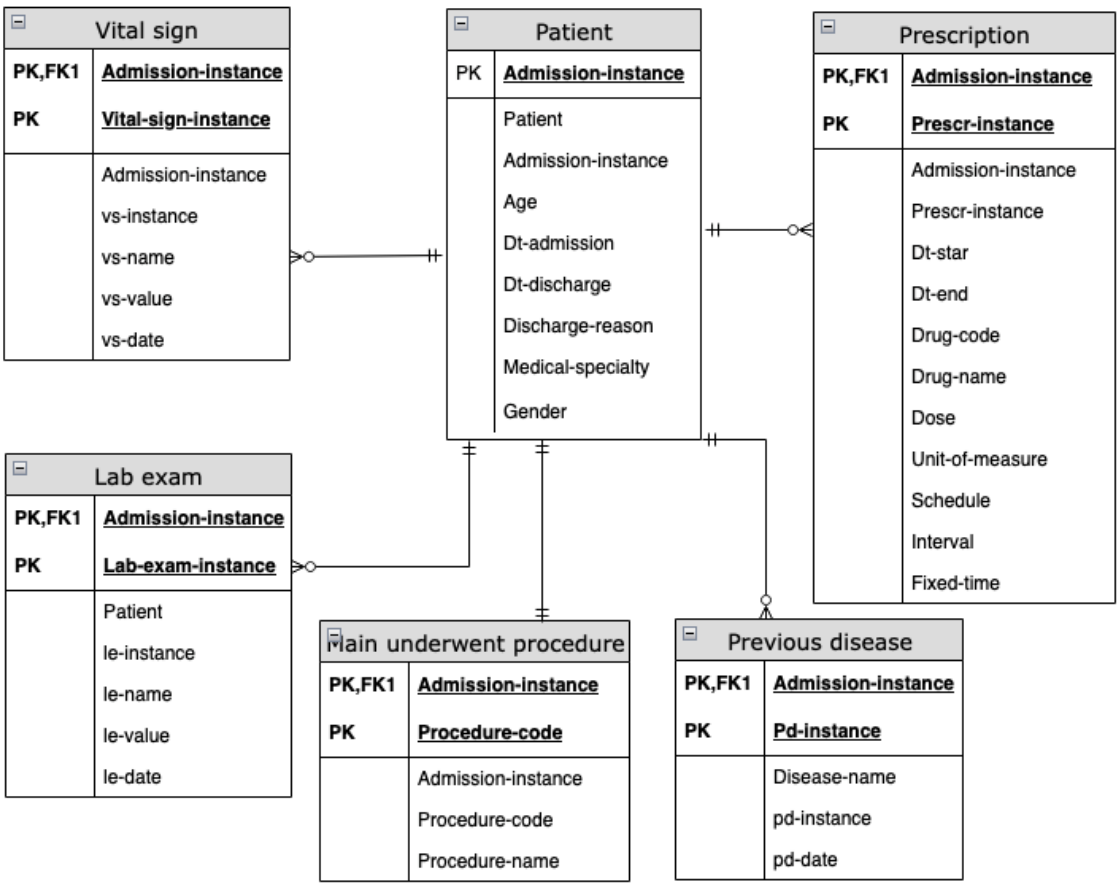


Figure 8.1: DataSet tables

- **Patient data**: patient characteristics (age, gender) and the information about the hospitalisation (admission, discharge) discharge reason (e.g., recovery, transfer, death) and medical speciality (e.g., Cardiology, Oncology, Neurology, Pulmonology). These records are filled out each time a patient is hospitalised.

- **Prescription**: Drugs administered at the hospital must be prescribed and registered in the prescription table. Information such as drug name, dose, schedule, interval and expiration date is stored in this table. A prescription can contain one or several drugs. In the hospital routine, a prescription is usually valid for 24 hours.

- **Lab exams**: Exam results are stored in this table with their results.

- **Main underwent procedure**: This table stores the procedure that is the main cause of the patient's hospitalisation.

- **Previous disease**: register of diseases that the patient has suffered before the hospitalisation
- **Vital sign**: physiological parameters such as body temperature, heart rate and blood pressure measurements.

Table 8.1 shows the general characteristics of the research patients. The data suggests that Male patients are more likely to be hospitalised. Considering that only patients over 64 years were considered, the mean age is 77.6. The length of stay refers to the average number of days a patient is hospitalised. For discharge reasons, we grouped them into Recovery/Transfer and Death.

Hospitalisation	
Gender	Male: 9,531
	Female: 8,450
Age(mean, SD):	77.6 - 7.4
Length of Stay(mean, SD):	4.7 - 5.8
Discharge reasons	Recovery/Transfer: 15,965
	Death: 2,016

Table 8.1: General characteristics of the considered patients in the dataset

Table 8.2 lists the general characteristics of the prescriptions. The dataset contains 92,273 prescriptions with a total of 1.3 million drugs. This means that, on average, 5.1 prescriptions with 14 drugs are prescribed per patient during the hospitalisation.

Total number of prescriptions:	92,273
Total number of prescribed drugs:	1,3M
Prescription per patient (mean, SD):	5.3 - 5.8
Drugs per prescription (mean, SD):	13 - 5.1

Table 8.2: Prescription profile

Table 8.3 lists the most prescribed drugs. These drugs are part of several drug classes but are mainly related to painkillers, heart diseases and antiemetics.

Drug	Quantity
Dipyrone Sodium	80,706
Metoclopramide	77,109
Human Regular Insulin	61,272
Glucose	51,194
Morphine	38,627
Captopril	33,939
Omeprazole	33,509
Enoxaparin	28,515
Furosemide	27,970
Ondansetron	27,628

Table 8.3: Leading administered drugs

The 10 most common diseases listed in Table 8.4 provide an overview of the different types of hospitalisations and treatments. The leading causes of hospitalisation are related to oncology treatment, as 4 out of 10 procedures are linked to it.

Disease	Quantity
Hospitalisation for continuous administration chemotherapy	1640
Treatment of other bacterial diseases	934
Treatment of pneumonia or influenza (flu)	908
Treatment of clinical complications in oncology patients	637
Coronary angioplasty with stent implantation	562
Treatment of heart failure	533
Excision and suturing with Z-plasty in oncology	484
Treatment of acute ischemic or hemorrhagic stroke	404
Treatment with multiple surgeries	372
Multiple excisions of skin or subcutaneous tissue lesions in oncology	343

Table 8.4: Leading diagnosed diseases

8.2 Experiments

The CDSS framework aims to support health professionals in tackling drug interactions, particularly when faced with elderly patients. To contextualise how the system could work in a hospital environment, we explain a typical patient journey in accordance with the hospital dataset. It is noteworthy again that the hospital employed the Beers Criteria guideline in its CDSS. Moreover, we present a few real patient cases to introduce how the framework was developed and demonstrate the obtained results in each case. It also highlights the benefits our approach would have had in real-life applications.

8.2.1 Patient journey

A typical patient journey illustrated in Figure 8.2 starts with the hospital admission, which can happen if the patient needs to undergo a scheduled surgery, treatment or arrives at the emergency department (A&E) as as a result of an emergency. The patient data is collected, and the patient is admitted to a ward department. There, physicians may prescribe drugs and care, and request exams in the hospital Eletronic Medical Record (EMR) system. For prescriptions, these are analysed by a pharmacist in order to detect and, when necessary, adjust inconsistencies such as dosage, interactions, repeated drugs, conflicts with the treatment plan, laboratory exams and scheduling problems. Some inconsistencies are automatically detected by the CDSS and others manually. After prescriptions have been analysed, the prescribed drugs are sent to the ward department to be administered by nurses. The drug administration, whenever possible, is administered at odd times following hospital rules.

Along the described process, there are several steps in which our framework could be incorporated to support health professionals. During the prescription process, the framework could alert the physician when an interaction is detected and provide an alternative solution, such as another drug or by providing the scheduled times to minimise the interaction. This same approach can be adopted in the pharmacist process, supporting the automated detection of prescription inconsistencies. In cases where the drug scheduling is not in according with the hospital routines, nurses occasionally also reschedule the drug administration directly. The framework rescheduling process of our CDSS can support nurses in their time adjustments in accordance with drug constraints that sometimes nurses are unaware of.

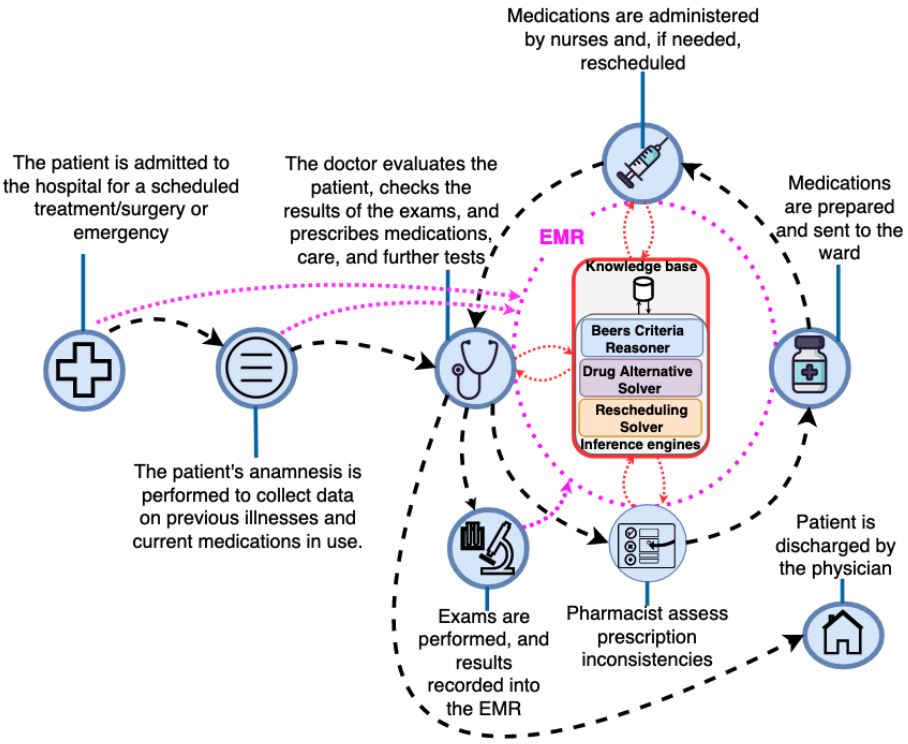


Figure 8.2: Patient journey

8.2.2 Experiments - Patient Case Studies

In order to demonstrate how the CDSS framework works with real data, we pick three patients from the hospital's EMR. These patients have different profiles of hospitalisation and disease. With these experiments, we want to evaluate if the proposed CDSS framework is able to detect inappropriate medications and compare the outcomes with the systems currently used by the hospital, the hospital's CDSS.

The hospital's CDSS outcome includes the drugs' names that were classified as inappropriate during the hospitalisation. Information on the inappropriate Beers Criteria classification or the number of prescriptions during the hospitalisation were not available/provided. Therefore, the comparison between the hospital system and the proposed CDSS is limited to the number of different types of drugs classified as inappropriate.

8.2.2.1 Case patient 1

The first patient was a 75 years old man who was hospitalised for the treatment of Parkinson's disease in the Medical Specialty of Neurology. During his

hospitalisation, 25 drugs were prescribed in two prescriptions corresponding to 12,5 drugs per day. His discharge reason was upon request.

Our CDSS framework found the inappropriate medications listed in Table 8.5. Six different prescribed drugs were classified as inappropriate in four interaction groups. In total, 17 cases of inappropriate drugs were identified. From the listed drugs, the hospital CDSS could only detect two drugs: Quetiapine fumarate and Promethazine.

Inappropriate medications		**Drug**	**Cases**
Group	**Subgroup**		
DDDS	All antipsychotics	Haloperidol	1
	Antiemetics	Promethazine	1
DDI	Anticholinergic /Anticholinergic	Clozapine	1
		Promethazine	1
	CNS Active Drugs /CNS Active Drugs	Clozapine	1
		Escitalopram	1
		Haloperidol	1
B.PIM	Antipsychotics first and second generation	Clozapine	1
		Haloperidol	1
		Quetiapine fumarate	1
	First-generation antihistamines	Promethazine	1
	Insulin sliding scale	Regular Human Insulin	2
UWC	Antipsychotics	Clozapine	1
		Quetiapine fumarate	1
	SSRIs	Escitalopram	2

Table 8.5: Patient 1: Inappropriate medications

After detecting inappropriate medications, the next step is to check if alternative drugs would have been available. The Beers Criteria ontology outcome revealed alternative drugs for drug Promethazine:alt_Beclomethasone, alt_Cetirizine, alt_-Fexofenadine, alt_Fluticasone, alt_Loratadine, alt_Saline_nasal_rinse and alt_-Steroid_nasal_sprays. This would mean that the interactions DDDS - Antiemetics, DDI- Anticholinergic/Anticholinergic and PIM - First-generation antihistamines

could have been solved. However, not all the interactions the given prescriptions had could have been solved, and consequently, a prescription without interactions does not exist. Instead, for drug-drug interactions, a rescheduling approach can be employed to minimise their interaction. In this case, DDI - CNS Active Drugs/CNS Active Drugs (CloZAPina, ESCitalopram and Haloperidol) can be rescheduled.

The following constraints composed the rescheduling model:

- **Drug: CloZAPina**
  - Interval : 2 x per day
  - Tmax: 150
- **Drug: ESCitalopram**
  - Interval : 1 x per day
  - Tmax: 300
- **Drug: Haloperidol**
  - Interval : 4 x per day
  - Tmax: 270

The rescheduling solver provided the drug administration times illustrated in Figure 8.3. The schedule does not have TMAX conflicts, all the constraints were considered, and the interval between drug TMAX times was maximised.

8.2.2.2 Case patient 2

The second patient was a man aged 89 who was hospitalised for surgical treatment of subdural hematoma in the Medical Specialty of Internal Medicine. During his hospitalisation, 495 drugs were prescribed in 26 prescriptions which corresponds to 19 drugs per day. His discharge reason was deceased.

Our CDSS framework found the inappropriate medications listed in Table 8.6. In total, 14 different prescribed drugs were classified as inappropriate in three interaction groups. In total, 190 cases of inappropriate drugs were identified. From the listed drugs, the hospital CDSS could only detect eight of these drugs: Prometazina, Quetiapine fumarate, Metilprednisolona succinate, Midazolam, Omeprazole, Risperidona, Sulfato de Atropina and Zolpidem.

Inappropriate medications		**Drug**	**Cases**
Group	**Subgroup**		
DDI	Anticholinergic/ Anticholinergic	Chlorpromazine	1
		Promethazine	1
	CNS Active Drugs/ CNS Active Drugs	Chlorpromazine	1
		Diazepam	1
		Fentanyl	7
		Haloperidol	17
		Midazolam	5
		Morphine	16
		Quetiapine fumarate	10
		Risperidone	6
		Magnesium Sulfate	1
		Zolpidem	3
	Opioids/Benzodiazepines	Fentanyl	5
		Midazolam	5
		Morphine	5
B.PIM	Benzodiazepines	Diazepam	1
		Midazolam	5
	First-generation antihistamines	Promethazine	2
	Insulin sliding scale	Regular Human Insulin	25
	Metoclopramide	Metoclopramide	26
	Nonbenzodiazepine benzodiazepine receptor agonist hypnotics	Zolpidem	3
UWC	Antipsychotics	Chlorpromazine	1
		Haloperidol	21
		Quetiapine fumarate	15
		Risperidone	6
	Diuretics	Mannitol	1

Table 8.6: Patient 2: Inappropriate medications

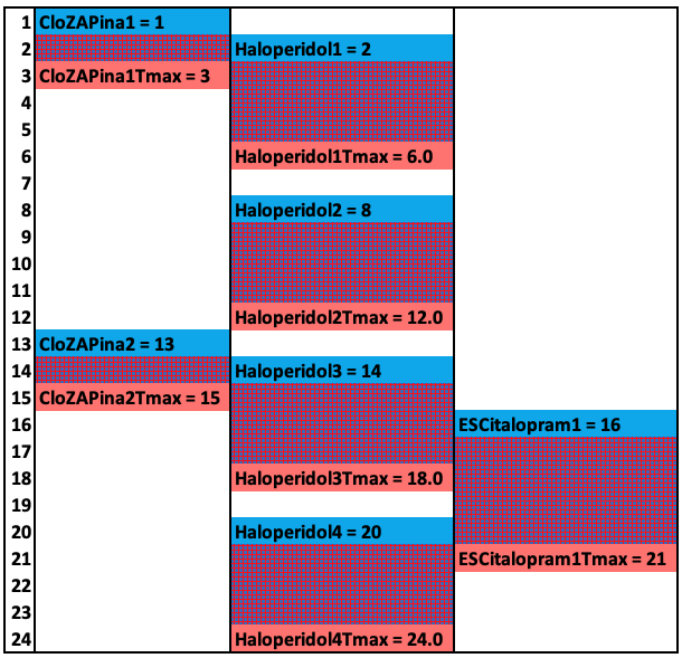


Figure 8.3: Patient 1 rescheduled drugs

For this patient, the CDSS the Beers Criteria ontology found two alternative drugs (alt_glipiZIDE and alt_metFORMIN) for ClorproMAZINA and seven (alt_-Beclomethasone, alt_Cetirizine, alt_Fexofenadine, alt_Fluticasone, alt_Loratadine, alt_Saline_nasal_rinse and alt_Steroid_nasal_sprays) for Prometazina. With these alternatives, it would have been possible to solve the DDI Anticholinergic/Anticholinergic, PIM-First-generation antihistamines and UWC-Antipsychotics for ClorproMAZINA. Overall, it would not have been possible to find a prescription without inappropriate drugs. However, it is possible to minimise the DDI - CNS Active Drugs/CNS Active Drugs by rescheduling the drugs. Hence, we selected a prescription with four drugs that belong to the category CNS Active Drugs, which are listed below:

- **Drug: Morfina**
  - Interval : 4 x per day
  - Tmax: 60
- **Drug: Risperidona**
  - Interval : 2 x per day
  - Tmax: 60

- **Drug: Haloperidol**
  - Interval : 2 x per day
  - Tmax: 270
- **Drug: Quetiapina, fumarato**
  - Interval : 2 x per day
  - Tmax: 90

Through the use of the solver underlying our solution we could reschedule all drugs maximising the distance between the TMAX times as shown in Figure 8.4.

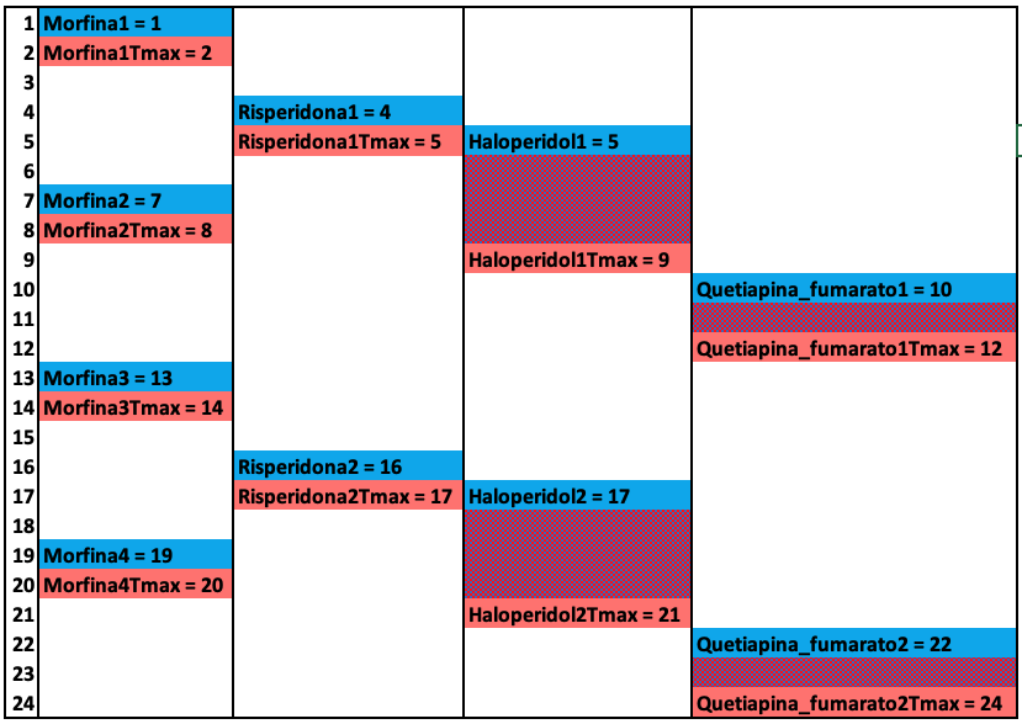


Figure 8.4: Patient 2 rescheduled drugs

8.2.2.3 Case patient 3

The third patient was a man aged 79 who was hospitalised for treatment of heart failure in the Medical Specialty of Cardiology. During his hospitalisation, 38 drugs were prescribed in 3 prescriptions which corresponds to 12,6 drugs per day. His discharge reason was recovered.

Inappropriate medications		Drug	Cases
Group	**Subgroup**		
DDDS	**NSAIDs**	Sodium Dipyrone	3
DDI	**Potassium-sparing diuretics**	Enalapril	3
		Spironolactone	3
	RAS inhibitor	Enalapril	3
	RAS inhibitor	Spironolactone	3
B.PIM	**Insulin sliding scale**	Regular Human Insulin	3
	Metoclopramide	Metoclopramide	3
UWC	**Diuretics**	Spironolactone	3
		Furosemide	3
	SSRIs	Escitalopram	3

Table 8.7: Patient 3: Inappropriate medications

The CDSS framework found the inappropriate medications listed in Table 8.7. Seven different prescribed drugs were classified as inappropriate in three interaction groups. In total, 30 cases of inappropriate drugs were identified. In this case, the hospital CDSS did not find any inappropriate drug. No drugs were found in the Beers Criteria ontology regarding the alternative drugs. Therefore, drugs belonging to DDI (Enalapril and Espironolactona) were rescheduled as follows:

- **Drug: Enalapril**
  - Interval : 4 x per day
  - Tmax: 240
- **Drug: Espironolactona**
  - Interval : 2 x per day
  - Tmax: 200

The drugs were rescheduled according to the TMAX value as illustrated in Figure 8.5. There is no conflict between TMAX times. However, there is an interaction between both drugs in six hours of twenty-four. For example, in

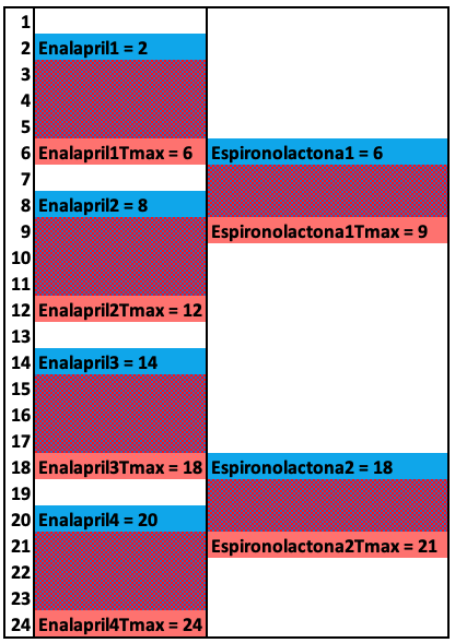


Figure 8.5: Patient 3 rescheduled drugs

hours 6, 8 and 9, both drugs interact; the same happens in hours 18, 20 and 21. Nevertheless, no solution could avoid the interaction between these drugs due to the number of intervals and the duration until the drugs reach their CMAX.

8.3 Results evaluation

In this section, we first compare our CDSS framework outcomes with the hospital CDSS. Then we evaluate the list and discuss the results of the prescription screening of our CDSS framework for the entire hospital EMR database. Finally, we compare the CDSS framework features with existing PIMs prescription-screening tools and with PIMs prescription-screening research.

A hospital pharmacist validated the results presented by our approach. The results of drugs classified as PIMs, the suggestion of alternative drugs and rescheduled drugs were validated. The validation showed that drugs classified as PIMs exactly followed the Beers Criteria. Moreover, alternative drugs had no interactions with other prescribed drugs and were in accordance with the AGS Health in Aging Foundation [73] and by Hanlon et al. (2015) [63] guidelines. Finally, the rescheduled drugs were maximized according to TMAX values appropriately.

Our approach was also demonstrated to a geriatric physician. The doctor

highlighted that in medical practice, drugs classified as PIMs are often ignored, resulting in some situations that result in negative results that could be avoided with a tool that helps in decision-making. In this way, he emphasized the importance of automating the detection of drugs classified as PIMs and suggested finding alternative drugs, which are tasks of significant complexity. If implemented in medical practice, the doctor assessed that our tool would prevent drugs classified as PIMs from being prescribed, avoiding adverse clinical outcomes.

8.3.1 Our CDSS Framework versus the Hospital's CDSS

The CDSS currently used by the hospital was implemented recently. Therefore, the system was not used to assess most of the prescriptions of the hospital EMR. Nevertheless, the hospital provided a set of results for 249 patients assessed by their CDSS. The results include a list of drugs classified as inappropriate according to the Beers Criteria for each patient during hospitalisation. The patient profile of this set is detailed in Table 8.8 and the prescription profile in Table 8.9.

Hospitalisation	
Gender	Male: 142
	Female: 107
Age(mean, SD):	77 - 6.9
Length of Stay(mean, SD):	14,4 - 15,2

Table 8.8: Patient profile

Total number of prescriptions:	3,696
Total number of prescribed drugs:	77,290
Prescription per patient(mean, SD):	14,8 - 15,3
Drugs per prescription(mean, SD):	20,9 - 7,4

Table 8.9: Prescription profile

Compared to the EMR data analysis, there was no significant change in the age mean and standard deviation. However, the length of stay significantly increased, impacting the number of prescriptions for each patient during hospitalisation. The number of drugs per prescription is also higher, which may impact the number

of interactions per patient. As previously mentioned, the hospital just provides a list of drugs detected during the patient's hospitalisation. Other information, for instance which Beers Criteria category the prescribed drugs belong to, the hospital's CDSS is not able to provide. Thus, the comparison between CDSS outcomes is limited to the number of different drugs that were classified as inappropriate.

Our CDSS framework could find 2864 inappropriate drugs for the 249 patients, which corresponds to an average of 11.5 drugs per patient during hospitalisation. This number does not consider the number of times a drug was prescribed or the different inappropriate categories a drug could belong to. In total, 70 different drugs were detected (cf. Table I.1 in the appendix).

Table 8.10 below lists the cases per Beers Criteria categories. It considers the number of times that a drug was prescribed and the categories that a drug belongs to. For example, if one inappropriate drug is prescribed three times during hospitalisation and belongs to two categories, it leads to six cases of inappropriate drugs. The higher incidence of inappropriate drugs is related to drug-drug interactions, followed by potentially inappropriate drugs, drugs that have to be used with caution and drug-disease or syndrome interaction.

Inappropriate drugs	**Cases**
DDI	16321
B.PIM	14432
UWC	7041
DDDS	1823
Total	**39617**

Table 8.10: Inappropriate drug cases

By contrast, the hospital's CDSS could find 839 inappropriate drugs, corresponding to an average of 3.7 per patient. Only sixteen different drugs were detected (cf. listed in the appendix on Table I.2). Five drugs were detected by the hospital's CDSS which were not detected by our framework CDSS: Two of these drugs (Nitrofurantoin and Atropine Sulfate) refer to version 2023 of the Beers Criteria, and the other three drugs (PrednisONE, Propafenone and Methylprednisolone succinate) are not listed in the Beers Criteria.

CDSS	Inappropriate drugs	Number of different drugs
CDSS framework	2864	70
Hospital CDSS	839	16

Table 8.11: Comparative results of Hospital x framework CDSS

When comparing the number of inappropriate drugs detected during hospitalisation, our framework outperformed the hospital's system by detecting more instances. Additionally, the hospital's CDSS misclassified some drugs that were not directly relevant to the Beers Criteria, leading to inaccuracies in the provided information. As such, our CDSS framework provides more reliable and accurate medication recommendations for health professionals. We also note that our framework is based on the information contained in Beers Criteria from 2019, but it can be extended to more recent versions and updated with additional information that does not appear in the Beers Criteria if there are valid reasons for doing so. The correctness of the sources used, in other words the information contained in the Beers Criteria and used by our CDSS, is nevertheless outside of the scope of this thesis.

8.3.2 Hospital EMR prescription-screening results

8.3.2.1 Results of the Beers Criteria ontology

A prescription screening was performed by our CDSS framework in the hospital's EMR to check and tackle occurrences of inappropriate medications. As previously introduced in Section 8.1, the EMR consists of 1,3 million drugs from 92,273 prescriptions and 17,981 patients aged 65 or above. Table 8.12 summarises the CDSS framework outcomes.

Inappropriate drugs	Inappropriate types	Non-inappropriate drugs	Total prescribed drugs
483,204	606,274	822,555	1,305,759

Table 8.12: Summary of Hospital EMR prescription-screening results

In total, 483,204 drugs were classified with at least one category of the Beers Criteria, which represents 37% of the total drugs prescribed. It means, on

average, that 5,23 of 13 drugs of each prescription have at least one type of inappropriate drug. Moreover, 606,274 inappropriate classifications were assigned to inappropriate drugs.

Table 8.13 lists the number of cases in each Beers Criteria category. The majority of inappropriate drugs were classified as B.PIM, DDI and UWC. DDDS was not as representative due to its restriction to specific diseases. In the case of VLKF, this is due to the creatinine clearance lab results that are a parameter for a drug to be classified as inappropriate, when the majority of patients did not have this exam in their EMR.

Interaction Group	Interaction cases	%
B.PIM	259983	42.9
DDI	226821	37.4
UWC	103053	17
DDDS	16416	2.7
VLKF	1	0
Total	606,274	100

Table 8.13: Hospital EMR Inappropriate medications by group

Table 8.14 lists the five more common subcategories of inappropriate medication cases. The first is the DDI_CNS_Active_Drugs, corresponding to drugs prescribed for the central nervous system. Drugs composing this drug class are considered inappropriate when three are prescribed together. In the EMR, 54 different types and 110,229 cases of these drugs were prescribed. In Table 8.15, this class is represented by drugs Tramadol, Morphine, Midazolam, Haloperidol, Clonazepam and Diazepam. The second is the PIM_Metoclopramide, which refers to the Metoclopramide drug used to treat nausea and vomiting and increase gastric motility. It is the second most prescribed drug, as shown in Table 8.3 and the first commonly prescribed inappropriate drug listed in Table 8.15.

The following subcategory is DDI_Opioids/Benzodiazepines, which refers to the interaction between the drug classes Opioids and Benzodiazepines. Opioids are prescribed mainly to treat moderate to severe pain, while Benzodiazepines are primarily prescribed for treating anxiety and other mental health conditions. This inappropriate subcategory corresponds to 19 different drug types, and 65,841 cases

Inappropriate SubGroup medications	Cases	%
DDI_CNS_Active_Drugs/CNS_Active_Drugs	110229	18.2
PIM_Metoclopramide	74213	12.2
DDI_Opioids/Benzodiazepines	65841	9.1
PIM_Insulin_sliding_scale	55219	8.4
PIM_Proton-pump_inhibitors	51123	7

Table 8.14: Hospital EMR Inappropriate medications by subgroup

Inappropriate drug	**Cases**	%
Metoclopramide	74338	12.3
Regular Human Insulin	48732	8
Tramadol	40368	6.7
Morphine	37158	6.1
Omeprazole	33509	5.5
Midazolam	30463	5
Haloperidol	29131	4.8
Clonazepam	28474	4.7
Diazepam	24255	4
Furosemide	23627	3.9

Table 8.15: The ten most commonly prescribed inappropriate medications

of these drugs were prescribed in the EMR. In Table 8.15, the drugs Tramadol, Morphine, Midazolam, Clonazepam and Diazepam correspond to this subcategory. PIM_Insulin_sliding_scale corresponds to the prescription of Human Insulin, which is prescribed for regulating blood sugar levels in patients with diabetes. It is the second most prescribed inappropriate drug listed in Table 8.15. Finally, PIM_Proton-pump_inhibitors refers to the drugs Omeprazole and Pantoprazole prescribed for treating stomach acid-related conditions.

In Table 8.16, we compared the incidence of inappropriate medications between different discharge reasons. The column Inappropriate Types lists the average inappropriate classifications that were assigned to inappropriate drugs. The

table shows that the incidence of inappropriate medications is higher for patients who died than those who recovered. Moreover, in Table 8.17, we compared the mortality rate in patients with a prescription of inappropriate medication versus those without inappropriate medication. Once again, patients prescribed inappropriate medications had a higher mortality rate.

Discharge reason	Inappropriate types	Total patients
Death	60.05	1,976
Recovered	32.78	14,876

Table 8.16: Hospital EMR Inappropriate medications by discharge reason

	Mortality rate	Death patients	Total patients
Patient with inappropriate medication	11.7%	1,976	16,852
Patient without inappropriate medication	3.4%	39	1,129

Table 8.17: Mortality rate among patients with inappropriate medication and patients without inappropriate medication

These values could indicate that prescribing inappropriate medication is associated with a higher mortality rate. However, we cannot reach this conclusion, as other variables must be evaluated to analyse the correlation between mortality and inappropriate medication that is not included in the scope of this research.

8.3.2.2 Results of the Alternative Drug Solver

We now discuss results on the detection of available alternatives for the prescribed inappropriate medication and the number of solved prescriptions. In total our CDSS framework detected 102 different drugs classified as inappropriate which can be replaced by alternative drugs. Table 8.18 summarises the context of the alternative drugs. The Beers Criteria ontology provided 106 alternative drugs for 22 different prescribed drugs.

In our approach, an alternative drug becomes a prescribed drug when it does not have interactions with other prescribed or alternative drugs. If all inappropriate

	Cases
Prescription with inappropriate drugs	89,602
Satisfiable prescription	152
Unsatisfiable prescription	89,450
Alternatives drug available for prescription	15,173
Drugs that have alternative	22
Alternative drugs available	106

Table 8.18: Summary of Hospital alternative drugs

drugs are solved, a prescription is considered satisfiable. Nevertheless, if an inappropriate drug cannot be solved, then the prescription is considered unsatisfiable. In total, 89,450 prescriptions had one or more drugs classified as inappropriate, which represents 97,1% of all EMR prescriptions. For 15,173 prescriptions, one or more alternative drug were available. However, only 152 prescriptions were considered satisfiable. It means that less than 1% of the prescriptions could be entirely solved. We note that this is as expected from experience when elderly patients take large numbers of medications. It is hence more important to prioritise and reduce the severity of interactions as opposed to completely eliminate them.

Table 8.19 provides a list of drugs with a significant number of alternative options. Upon comparing this list to Table 8.15, which details the most commonly prescribed inappropriate drugs, it is observed that none of the alternative drugs listed in the former table are present in the latter. It implies that the primary alternative drugs are not applicable to the most commonly prescribed inappropriate drugs.

The low number of satisfiable prescriptions implies that the available alternative is for specific cases that differ from the EMR inappropriate drugs. For example, most alternative drugs are available for a particular disease or clinical condition, such as dementia and history of falls. Moreover, the average number of 5,23 inappropriate drugs per prescription and 13 drugs per prescription is also a reason that more prescriptions could not be solved.

Drug	Alternatives
Indomethacin	33
Cyclobenzaprine	33
Thiopental	29
Primidone	29
Phenobarbital	29
Phenobarbital Sodium	29
Haloperidol	13
Quetiapine Fumarate	13
Olanzapine	13
Clomipramine	11

Table 8.19: Drugs with a greater number of alternative drug options

8.3.2.3 Results of the Rescheduling Solver

We now discuss results of the use of the underlying solver in our framework to reschedule prescriptions. For the rescheduling process we only considered interaction between drugs (DDI). In total, 47,133 prescriptions, 161,265 drugs and 439,126 drug instances were available to be rescheduled. We recall that we use *drug instance* to refer to the number of times the same drug is administered in 24 hours.

The rescheduling solver could maximise the distance of TMAX values from 32973 prescriptions, and 97,305 drugs (or 240,591 drug instances) were rescheduled. In other words, 54,78% of the available drug instances were rescheduled. The main reason for not rescheduling drugs in some prescriptions were cases such as drugs with continuous administration (e.g., saline infusion). In this case, the CMAX is constant as the drug is administered 24 hours daily. Therefore, there is no way of maximising the distance with other drugs.

8.3.3 Our CDSS Framework versus Other Existing Tools

In addition to comparing our CDSS framework with the hospital's most recent CDSS, we compared it with existing tools that (claim to) address inappropriate drugs. We defined parameters based on the Beers Criteria to consider patient

and prescription drug profiles. Moreover, we evaluated if the tools provided alternative drugs and if they considered the patient profile. We also looked for additional approaches to address inappropriate drugs, such as drug rescheduling. Finally, we checked some essential features, such as being reusable and supporting the integration with EMR systems. Below are the tools that have been compared:

- AGS Beers Criteria [145]: is an app provided by the American Geriatrics Society to tackle potentially inappropriate drugs which supports health professionals in implementing prescribing recommendations.
- MALPIP, STOPP Start, Beers [51]: is a screening tool to issue alerts and in this way to support health professionals in dealing with inappropriate drugs.
- GlobalRPh [2]: is a website to generate a report of medications from the Beers criteria based on the patient's current conditions.
- PIM Check [47]: is a prescription-screening website checklist to detect potentially inappropriate medications.

Table 8.20 compares these tools and the CDSS framework. The first item evaluated is the decision based on the patient profile, which means that if a drug is classified as inappropriate, we consider the patient profile. The only parameter that is considered by AGS Beers Criteria, GlobalRPh and PIM Check is the patient's disease. In contrast, our CDSS framework, considers not only the patient's disease, but the age, gender, criticality and laboratory exams, which are required parameters to classify a drug as inappropriate. The next item refers to the prescription drug profile. According to the Beers Criteria, some drugs are classified as inappropriate depending on the patient's age and the dose, duration of therapy or if it is a first-line treatment. Therefore, these parameters must be considered to provide accurate information. None of the tools considers these parameters, only our CDSS framework.

Another dimension that was evaluated is if the tool suggests alternative drugs and if this suggestion considers the patient profile. Besides our CDSS framework, the AGS Beers Criteria tool is the only one which provides a list of alternatives. However, it is a static list that does not consider the patient profile. Some drugs are considered alternatives in specific conditions, for example, when a patient suffers from a disease. Therefore, our CDSS framework provides a straightforward

Dimension	CDSS Framework	AGS Beers Criteria	MALPIP, STOPP Start, Beers	GlobalRPh	PIM Check
Decisions based on the patient profile	**Yes**	**Yes**	No	**Yes**	**Yes**
Age	**Yes**	No	No	No	No
Gender	**Yes**	No	No	No	No
Disease	**Yes**	**Yes**	No	**Yes**	**Yes**
Criticality	**Yes**	No	No	No	No
Exams	**Yes**	No	No	No	No
Decisions based on the prescription drug profile	**Yes**	No	No	No	No
Dose	**Yes**	No	No	No	No
Duration therapy	**Yes**	No	No	No	No
First line	**Yes**	No	No	No	No
Suggest alternative drugs	**Yes**	**Yes**	No	No	No
Suggest alternative drugs based on patient profile	**Yes**	No	No	No	No
Disease	**Yes**	No	No	No	No
Check alternative drugs interaction with prescribed drugs	**Yes**	No	No	No	No
Reschedule drugs for minimising interaction	**Yes**	No	No	No	No
Based on standard knowledge (e.g., collected from CPGs)	**Yes**	Yes	Yes	Yes	Yes
Available for reuse	**Yes**	No	No	No	No
Interoperable with EMR systems	**Yes**	No	No	No	No
Multi-languages label	**Yes**	No	No	No	**Yes**

Table 8.20: Comparison between prescription-screening tools

suggestion of alternative drugs, as it considers the patient profile to provide the suggestions. Additionally, to provide alternative drugs, our framework also checks if the alternatives do not conflict with other prescribed or alternative drugs, providing a more accurate suggestion. The AGS Beers Criteria tool does not take into account the patient's prescription. If it is impossible to switch an inappropriate drug with an alternative drug, our CDSS framework provides an additional solution that minimises the interaction by rescheduling drugs that have interaction. All the other mentioned tools do not provide a mechanism to address interactions further.

The standard knowledge incorporated in these tools was also analysed. All the existing tools consider the Beers Criteria list, whereby some of them consider the 2015 version of the criteria, whereas our CDSS framework is based on the 2019 version. The tool with the most updated version of Beers is the AGS Beers Criteria which considers the 2023 version. The tools evaluation also includes three essential characteristics: availability of a knowledge base for reuse by other systems, integration with EMR systems, and multi-language label options. Out of all the options, only our CDSS framework offers the ability to reuse and integrate. Additionally, PIM Check is the only tool besides our framework that offers information in two languages. However, the architecture of our framework allows the information to be defined in any number of languages.

We conducted a comparative analysis of studies aiming to accomplish similar objectives as our research, as presented earlier in Table 2.1. The purpose was to identify areas that require further improvement or resolution. Based on that table, we list the elements that were evaluated with the results of our research as follows:

- **Guideline**: Beers Criteria
- **Shareable knowledge base**: Yes
- **Alternative drug recommendation**: Yes
- **Conflict management**: Yes
- **Drug scheduling recommendation**: Yes
- **Pharmacokinetics parameters**: Yes, Tmax value
- **Input**: Patient data, disease, exams, prescriptions

- **Output**: Inappropriate criteria, rationale, recommendation, strength of recommendation, quality of evidence, alternative drugs, rescheduled drug times.

Compared with the studies listed in Table 2.1, our CDSS framework addresses all the compared elements. For example, it is the only one that provides alternative drug recommendations with a conflict management tool to select only the alternatives that do not interact with other drugs. Moreover, it provides another approach to minimise the interaction by rescheduling the drugs considering the TMAX value. Finally, it embraces more input data that can be used to provide more accurate information and provide more outputs to support clinical decision-making.

The comparison conducted in Table 8.20 and Table 2.1 indicates the current tools and approaches available for handling PIMs do not provide a comprehensive solution. It means that these tools/approaches cover only a part of the process of supporting the decision process, such as the identification of PIMs. Consequently, it may affect the clinical decision process, leaving healthcare professionals in charge of finding solutions for medications identified as PIMs. This highlights the importance of our approach, which presents a complete solution for detecting and resolving PIMs issues.

8.3.4 Performance Evaluation

To conduct all the experiments, a MacBook with a 2,3 GHz Dual-Core Intel Core i5 CPU and 16 gigabytes of memory was used. The experiments were conducted using a database with a total size of 600 MB. This database contained the tables listed in the Appendix G, which constitute all patient data and prescriptions, as well as the results of experiments, such as interactions, prescriptions with alternative medications and rescheduled prescriptions.

Table 8.21 lists the performance of each inference engine to run the experiments. For comparison purposes, we selected five prescriptions from the hospital database with the highest number of medications and five with the lowest number of medications.

Based on the data analyzed, it was found that the average time required to run a prescription with a smaller number of medications was 20 seconds. In comparison,

Prescription	Number of Drugs	Execution Time	Avg Time per Drug
Prescription 1	69	00:00:43	00:00:01
Prescription 2	66	00:00:38	00:00:01
Prescription 3	65	00:00:29	00:00:00
Prescription 4	65	00:00:38	00:00:01
Prescription 5	63	00:00:54	00:00:01
Prescription 6	1	00:00:20	00:00:20
Prescription 7	1	00:00:20	00:00:20
Prescription 8	1	00:00:20	00:00:20
Prescription 9	1	00:00:22	00:00:22
Prescription 10	1	00:00:20	00:00:20
Total	333	00:05:04	00:00:11

Table 8.21: Summary of prescription performance evaluation

for a prescription with a greater number of medications, the average time was 40 seconds. On calculating the average execution time of the total number of medications with the total execution time, the average value obtained was 11 seconds. It can be observed that there is a significant difference between the average time taken to run the first five prescriptions as compared to the next five prescriptions. This indicates that the ontology requires approximately 19 seconds to be loaded and executed, while the impact of the number of drugs on the execution time is around 1 second per drug. We could not find information regarding the execution performance of the studies/tools we use as a benchmark to evaluate our approach. Therefore, we were unable to carry out a comparative performance analysis.

8.4 Summary

This chapter demonstrated how the CDSS framework performed over a real hospital EMR dataset. We showed concrete cases of patients from the EMR to detail the outcomes of each component of the CDSS and compare them with the hospital's CDSS outcomes. Moreover, we showed and compared the results of the

prescription-screening assessment on the hospital's EMR by the CDSS. The results demonstrated that our CDSS framework could detect more inappropriate drugs and provide accurate information based on patient and prescription profiles.

Our CDSS framework was also compared with other prescription-screening tools, and compare the features and scope of each tool. The results demonstrated that our CDSS framework is much more comprehensive, the only one that considers all the Beers Criteria parameters and consequently can provide accurate personalised information to support clinical decision-making. Moreover, it is the only available approach which contains additional mechanisms to tackle inappropriate prescribing (here in the form of drug rescheduling). Finally, we evaluate studies with objectives similar to our research. The CDSS framework stands out as the sole solution that comprehensively addresses all the evaluated items, making it the most effective personalised approach for addressing inappropriate prescribing.

Chapter Nine

Conclusions

Supporting health professionals in taking better decisions has been the key motivation for our research to improve the quality of patient care and treatment. Avoiding patient harm that could lead to death due to inappropriate medications is a must when defining treatments. Medications should have the purpose of treating and improving the quality of life and not be a risk to health and life. However, as shown in our thesis through the complexity of the ontology alone, it is hard to keep an oversight of the complexity of conditions, drugs, and potential intolerances patients may have without the aid of automated solutions.

This thesis looked into this problem by exploring how different approaches within computer science could be combined to provide novel and realistic solutions currently unavailable to healthcare practitioners. As demonstrated, the current tools for handling Potentially Inappropriate Medications only cover a part of the decision process, leaving healthcare professionals responsible for finding comprehensive solutions for solving Potentially Inappropriate Medications issues. Through our developed approach, we make a novel contribution to supporting healthcare providers and avoiding inappropriate prescribing. In this research, we designed and implemented a CDSS approach that tackles potentially inappropriate drugs from the Beers Criteria guideline to be in line with best medical practice. The core of our approach combines formal reasoning activities to detect interactions, find alternative drugs or minimise the effect of drug interactions and inappropriate medications in accordance with a recognised knowledge base.

We have covered three sub-questions that summarise our research questions: (1) how can we formalise PIMs in a clinical knowledge base for reasoning? (2) which

reasoning methods best support clinical decision-making for PIMs, and (3) can the proposed approach be used in practice to identify and solve PIMs in real scenarios?

To address our first sub-questions, we proposed a novel knowledge base that gathers information regarding potentially inappropriate medications, alternative drugs and drug parameters of the Beers Criteria guideline to support reasoning activities. No formalisation of Beers Criteria can be found in the literature that reasoners (or other automated tools) can interpret. Therefore, we formalised the Beers Criteria into an ontology that can be shared and integrated with other knowledge bases or systems and can be continuously updated with new information. This process involved knowledge acquisition from each Beers criterion to specify the requirements for determining the classes, building the taxonomy and defining the object, data and annotation properties that constitute the ontology.

The second sub-questions was first formalised (and then built) by adding inference rules to the ontology to determine when a drug was classified as inappropriate and to suggest alternative drugs. Moreover, we propose an SMT model (though at this point a SAT-based approach would suffice) to check and suggest only alternative drugs that do not have interaction with other prescribed drugs. Different from the evaluated tools, these inference rules considered the patient and prescription parameters, providing a precise prescription screening result. Additionally, considering that an alternative drug is not always available, we proposed an SMT solver based rescheduling approach to minimise the effect of drug interactions more efficiently than currently done in practice. By considering the TMAX value, we aimed to avoid/minimise the interaction between drugs at the peak of drug concentration in the blood(CMAX). We also considered additional constraints regarding drug administration, such as fixed time of administration and hospital routines.

Finally, for our last sub-questions, we conducted a proof of concept with an real hospital EMR to evaluate and compare our CDSS approach with other tools and studies. The results demonstrated that our CDSS approach could detect a greater number of inappropriate drugs and offer more precise solutions to address them. Moreover, we validated the correctness and completeness of the inference engines. First, we check ontology consistencies with the Pellet [142] reasoner. Then to test the completeness of content coverage, we provided inputs by formulating prescriptions that covered each criterion and checked if the outputs

were in accordance with what was expected.

9.1 Key Contributions

There are several contributions in this thesis:

- We have demonstrated how to formalise clinical guidelines in an ontology. Having the ontology, we can use it further to support prescription screening to tackle inappropriate medications.
  - The formalisation process converted the Beers Criteria guideline into FOL and next into ontology inference rules. The process ranged from defining guideline requisites to creating inference rules to detect and propose alternative drugs to inappropriate medications. Formalising a guideline in an ontology facilitates the reuse of its knowledge by other systems. The same process can be applied to formalise other guidelines, reusing the object and data properties employed in the ontology. Further knowledge can be incorporated similarly, adding new relationships if and as needed.
  - The detection of inappropriate medications that considers patient and prescription profiles to fulfil guideline requirements is a complex task to be performed by humans. To the best of our knowledge, this approach addressed all the Beers Criteria requirements and can personalise the detection and suggestion of alternative drugs for inappropriate drugs. As Beers Criteria change over time, our approach can be extended and adapted to cover these changes.
- Supporting healthcare professionals in decision making can encompass several steps. The identification of a drug interaction or inappropriate medication is the standard approach of a CDSS . Yet, addressing a medication issue necessitates more than just detection; a CDSS must also aid healthcare professionals in resolving it.
  - The suggestion of alternative drugs that consider the patient a prescription profile is an approach that supports addressing medication problems. Manually doing it when several parameters have to be considered, is time consuming and error prone. The alternative solver

can support health professionals to optimise and mitigate errors by defining alternative drugs.

- ➳ The process of rescheduling drugs to minimise is a practice that is already adopted in hospitals. However, taking into account the TMAX to provide a more accurate solution is not part of the rescheduling routines. Thus, our rescheduling solver provides a novel approach that enhances and optimises the rescheduling process providing a more accurate way to minimise the effect of drug interactions.

* ❋ The prescription of inappropriate medications is still a concern that has to be addressed in Brazilian hospitals (where our dataset is from) and beyond.
  - ➳ The results of the evaluation of the dataset from the hospital's EMR demonstrated that the incidence of inappropriate medications is very high.
  - ➳ The current hospital's CDSS detected only a small number of inappropriate drugs, which may have led health professionals to prescribe possible harmful drugs for patients without knowing the side effects when some of these cases could have been avoided.

In 2017, the World Health Organization (WHO) launched the third Global Patient Safety Challenge to tackle unsafe medication practices and medication errors [113]. It emerged from the recognition that medication errors and associated harm remains a problem worldwide. Even though it includes aspects that we cannot address (like incorrect drug dispensing), we make a contribution to reducing medication-related harm by focusing on PIMs in the elderly. Indeed, we address this challenge by offering a digital solution that can be used to provide guidance to medical practitioners and consequently improve the choice of prescribed medications in the interest to enhancing patient safety.

9.2 Threats to Validity

Our approach has external dependencies that limit the validity of our tool and, thus, the validity of our results. For instance, we assume that the drugs defined as PIMs by the Beers Criteria are correct. However, there are several limitations, for example, Beers Criteria 2019 states: "Evidence for the benefits and harms of medications in older adults is often limited, particularly from randomized

clinical trials, and so decisions on the composition of the criteria were often made in the context of best-available, rather than definitive evidence. Moreover, evidence assessment frameworks are not perfectly tuned to drug safety evaluation, particularly for observational studies from which much relevant evidence derives." Hence, if the guideline has any errors, the same error will be applied to our approach since we fully replicate the rules of the guideline into the ontology. Moreover, we assume that the results of the ontology reasoner and the SMT solver are correct, and then we use that information to conduct our evaluation. However, should the ontology or SMT solver misbehave at some point, the trust in our results would have to be reassessed.

We have evaluated our approach through a proof-of-concept with a hospital EMR. However, for this approach to be validated in a real-world environment before being approved for use, the CDSS would have to undergo some trials to ensure accuracy and correctness. For example, randomized controlled trials to evaluate the effectiveness of the CDSS by comparing outcomes between a group using the CDSS and a control group without it in order to assess the clinical outcomes, patient safety, and adherence to guidelines. Moreover, accuracy and validation trials to verify the accuracy of the CDSS in providing relevant information and recommendations to compare the CDSS outputs against established clinical guidelines and expert opinions. The feedback from the hospital will go a long way towards addressing "threats to external validity".

In addition to external threats, there are also some threats to the internal validity of our approach. For example, the conversion of the Beers Criteria rules to the ontology is performed manually, hence, errors that could affect the results are possible. Furthermore, the ontology update process is also carried out manually and lacks a system that guarantees correct versioning and interoperability with other systems.

Although the ontology provides multi-language labelling and allows multiple names to be registered for the same class, such as alternative (commercial) names of medications, it is subject to misspelling errors. The approach does not address or validate any spelling errors relating to patient data, medications and prescription details, either in the ontology or in a patient's data entry. This may represent a threat to its applicability in a real-world scenario where data entry is not previously validated.

9.3 Future Work

There are many possible directions for future work.

Regarding the Beers Criteria ontology, it was developed in accordance with version 2019. However, a new version was released in May 2023. Therefore, it has to be updated with this version. The formalisation and update of the Beers Criteria ontology requires a manual analysis and translation to the ontology. An approach with machine analysis and translation could automate and improve this process without human effort.

On average, a new version of Beers Criteria is released every four years. As mentioned, importing information regarding Beers Criteria directly into a CDSS is not possible. A tool for automating the conversion process of the Beers Criteria file into an ontology or another format that allows data to be structured to be imported or compared with previous versions could be an opportunity for future work. On the other hand, it would be handy if AGS released the versions in a structured way, thus avoiding this rework.

Besides the Beers Criteria, other guidelines cover potentially inappropriate drugs. To provide a more comprehensive analysis, detect more inappropriate drugs, and suggest more alternative drugs, other guidelines could be incorporated, such as the STOPP/START [111]. The ontology was built in a standalone environment, which means it is not, at present, integrated with other ontologies. In the future, it could be integrated with the SNOMED ontology, which is a medical terminology and coding system that covers a wide variety of clinical concepts such as diseases, procedures, drugs, and active ingredients. A further benefit of doing so is that it would facilitate integration with other ontologies, CDSS and EMR systems. Moreover, to enhance the traceability of the information captured from the ontology, the source table of Beers Criteria could be incorporated into the ontology. For example, the criterion PIM -> Anticholinergics -> First-generation antihistamines ->Brompheniramine refers to Table 2. Therefore, this information could be stored in an annotation property, facilitating the traceability of the ontology rules and the Beers Criteria. The same approach could be applied to alternative drugs. Running the drug interaction rules backwards, given a patient's known prescribed medications and interaction symptoms, for identifying the possible drug interactions for unknown medications of that patient would be another field that could be explored. The reverse process could be a way of

supporting the discovery of unknown PIM drugs.

The alternative drug solver considers only hard constraints, which means that all drugs have the same weight in the definition of an alternative. Soft constraints could be incorporated to add different drug weights (scores) or PIMs according to specific parameters, as proposed by [24, 23]. For example, scores could be defined to separate mild from serious interactions or side-effects/harm, and to find alternative drugs that may have interactions but are still less severe. We could also consider the patient choices when defining the alternative, for example, to avoid certain side effects. Moreover, the cost of a drug could also be a parameter for the definition of an alternative drug.

Currently, the rescheduling solver considers only the TMAX parameter. In the current model, we know when a drug is administered and when it reaches the CMAX at time given by TMAX. However, we do not know when the effect of a drug stops or how different people react differently. Therefore, we could for instance incorporate the time to reach the minimum concentration in the bloodstream, aka Tmin. This measure would provide the solver with information on which distance between drugs could be considered safe (potentially useful for very severe drug interactions concerning drugs which can nonetheless not be removed). For example, if a drug reaches the Tmin after 1 hour of the TMAX, there is no need for the rescheduling to maximise the distance between drugs by more than 1 hour. In addition to Tmin, we could adopt other parameters such as T50 or T10 (when the concentration drops to 50% or 10%) to find the best time to administer the medications. Additionally, the rescheduling process was considered over a 24-hour window. Additionally, constraints could be added to the model to consider drugs administered in an interval of more than 24 hours, as currently the rescheduling process considers only a 24-hour window.

One of the future objectives is to integrate our approach with the hospital's EMR or CDSS. To properly integrate patient and prescription data, we must validate the integration process with other systems. Moreover, we must develop a front-end layer to display the processed data and results.

REFERENCES

[1] Stanford Center for Biomedical Informatics Research (BMIR), "Protégé,". https://protege.stanford.edu. [Online; accessed 2021-01-10].

[2] GlobalRPh - Beers Criteria Patient-Specific Reporting. https://globalrph.com/medcalcs/beers-criteria-patient-specific-reporting-available/. Accessed: August 7, 2023.

[3] Erika Ábrahám and Gereon Kremer. Satisfiability checking: Theory and applications. In *Software Engineering and Formal Methods: 14th International Conference, SEFM 2016, Held as Part of STAF 2016, Vienna, Austria, July 4-8, 2016, Proceedings 14*, pages 9–23. Springer, 2016.

[4] European Medicines Agency. Guideline on the investigation of drug interactions, 2012.

[5] Agência Nacional de Vigilância Sanitária (ANVISA). Boletim de farmacovigilância aborda erros de medicação, 2022. URL http://antigo.anvisa.gov.br/resultado-de-busca?p_p_id=101&p_p_lifecycle=0&p_p_state=maximized&p_p_mode=view&p_p_col_id=column-1&p_p_col_count=1&_101_struts_action=%2Fasset_publisher%2Fview_content&_101_assetEntryId=5765434&_101_type=content&_101_groupId=219201&_101_urlTitle=boletim-de-farmacovigilancia-aborda-erros-de-medicacao&inheritRedirect=true. Accessed: November 8, 2023.

[6] Ahmad Al-Azayzih, Rawan Alamoori, and Shoroq M Altawalbeh. Potentially inappropriate medications prescribing according to beers criteria among elderly outpatients in jordan: a cross sectional study. *Pharmacy Practice (Granada)*, 17(2), 2019.

[7] Kannayiram Alagiakrishnan, Patricia Wilson, Cheryl A Sadowski, Darryl Rolfson, Mark Ballermann, Allen Ausford, Karla Vermeer, Kunal Mohindra,

Jacques Romney, and Robert S Hayward. Physicians' use of computerized clinical decision supports to improve medication management in the elderly–the seniors medication alert and review technology intervention. *Clinical interventions in aging*, 11:73, 2016.

[8] Cassia Amorim Rodrigues Araújo and Isabel Cristina Fonseca da Cruz. Evidence-based nursing practice about risk of adverse drug interaction in icu–systematic review. *Journal of Specialized Nursing Care*, 13(1), 2021.

[9] Imran Sarwar Bajwa, Bushra Ramzan, and Shabana Ramzan. Markov logic based inference engine for cdss. *Mehran University Research Journal of Engineering & Technology*, 36(1):55–66, 2017.

[10] Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, et al. cvc5: A versatile and industrial-strength smt solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 415–442. Springer, 2022.

[11] Karen Barnett, Stewart W Mercer, Michael Norbury, Graham Watt, Sally Wyke, and Bruce Guthrie. Epidemiology of multimorbidity and implications for health care, research, and medical education: a cross-sectional study. *The Lancet*, 380(9836):37–43, 2012.

[12] Mark H Beers, Joseph G Ouslander, Irving Rollingher, David B Reuben, Jacqueline Brooks, and John C Beck. Explicit criteria for determining inappropriate medication use in nursing home residents. *Archives of internal medicine*, 151(9):1825–1832, 1991.

[13] Eta S Berner and Tonya J La Lande. Overview of clinical decision support systems. In *Clinical decision support systems*, pages 1–17. Springer, 2016.

[14] Ralf Bierig, Stephen Brown, Edgar Galván, and Joe Timoney. *Essentials of Software Testing*. Cambridge University Press, 2021.

[15] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. *νz*-an optimizing smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 21*, pages 194–199. Springer, 2015.

[16] Emilia Błeszyńska, Łukasz Wierucki, Tomasz Zdrojewski, and Marcin Renke. Pharmacological interactions in the elderly. *Medicina*, 56(7):320, 2020.

[17] BMJ Newsroom. 237 million medication errors made every year in england, 2022. URL https://www.bmj.com/company/newsroom/237-million-medication-errors-made-every-year-in-england/. Accessed: November 8, 2023.

[18] Miquel Bofill, Joan Espasa, and Mateu Villaret. Relaxed e-step plans in planning as smt. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 563–570, 2017.

[19] Ettore Bolisani and Constantin Bratianu. The elusive definition of knowledge. In *Emergent knowledge strategies*, pages 1–22. Springer, 2018.

[20] Paolo Bouquet, Chiara Ghidini, Fausto Giunchiglia, and Enrico Blanzieri. Theories and uses of context in knowledge representation and reasoning. *Journal of pragmatics*, 35(3):455–484, 2003.

[21] Jean Bousquet. Electronic clinical decision support system (ecdss) in the management of asthma: from theory to practice, 2019.

[22] Malaz Boustani, Noll Campbell, Stephanie Munger, Ian Maidment, and Chris Fox. Impact of anticholinergics on the aging brain: a review and practical application. 2008.

[23] Juliana Bowles, Marco B Caminati, Suhyun Cha, and Juan Mendoza. A framework for automated conflict detection and resolution in medical guidelines. *Science of computer programming*, 182:42–63, 2019.

[24] Juliana KF Bowles and Marco B Caminati. Balancing prescriptions with constraint solvers. In *Automated Reasoning for Systems Biology and Medicine*, pages 243–267. Springer, 2019.

[25] Karin Koogan Breitman, Marco Antonio Casanova, and Walter Truszkowski. Ontology in computer science. *Semantic Web: Concepts, Technologies and Applications*, pages 17–34, 2007.

[26] Marshall N Brunden, Thomas J Vidmar, and Joseph W McKean. *Drug Interaction and Lethality Analysis*. Chapman and Hall/CRC, 2019.

[27] Teresa Cristina Jahn Cassoni, Ligiana Pires Corona, Nicolina Silvana Romano-Lieber, Silvia Regina Secoli, Yeda Aparecida de Oliveira Duarte, and Maria Lúcia Lebrão. Use of potentially inappropriate medication by the elderly in são paulo, brazil: Sabe study. *Cadernos de saude publica*, 30(8): 1708–1720, 2014.

[28] Ellen Carolina D Castilho, AMM Reis, TL Borges, LDC Siqueira, and AI Miasso. Potential drug–drug interactions and polypharmacy in institutionalized elderly patients in a public hospital in brazil. *Journal of psychiatric and mental health nursing*, 25(1):3–13, 2018.

[29] Xing Chen, Chenggang Clarence Yan, Xiaotian Zhang, Xu Zhang, Feng Dai, Jian Yin, and Yongdong Zhang. Drug–target interaction prediction: databases, web servers and computational models. *Briefings in bioinformatics*, 17(4):696–712, 2016.

[30] Yuh-Jen Chen. Development of a method for ontology-based empirical knowledge representation and reasoning. *Decision Support Systems*, 50(1): 1–20, 2010.

[31] Antonio Cherubini, Andrea Corsonello, and Fabrizia Lattanzio. Polypharmacy in nursing home residents: what is the way forward? *Journal of the American Medical Directors Association*, 17(1):4–6, 2016.

[32] Wajanakorn Chivapricha, Varalak Srinonprasert, and Thanarat Suansanae. Impact of geriatric pharmacy specialist interventions to reduce potentially inappropriate medication among hospitalized elderly patients at medical wards: A prospective quasi-experimental study. *Drugs-Real World Outcomes*, 8(1):39–47, 2021.

[33] Insun Choi, Seung-Mi Lee, Linda Flynn, Chul-min Kim, Saerom Lee, Na-Kyung Kim, and Dong-Churl Suh. Incidence and treatment costs attributable to medication errors in hospitalized patients. *Research in Social and Administrative Pharmacy*, 12(3):428–437, 2016.

[34] SallyL Collins, ClaraC Faura, R Andrew Moore, and HenryJ McQuay. Peak plasma concentrations after oral morphine: a systematic review. *Journal of pain and symptom management*, 16(6):388–402, 1998.

[35] Erika Cornell, Michael Kwa, Amy S Paller, and Shuai Xu. Adverse events reported to the food and drug administration from 2004 to 2016 for cosmetics

and personal care products marketed to newborns and infants. *Pediatric dermatology*, 35(2):225–229, 2018.

[36] Kieran Dalton, Gary O'Brien, Denis O'Mahony, and Stephen Byrne. Computerised interventions designed to reduce potentially inappropriate prescribing in hospitalised older adults: a systematic review and meta-analysis. *Age and ageing*, 47(5):670–678, 2018.

[37] Flávia Fernanda Rosa D'Aquino, Carmen Maria Casquel Monti Juliani, Silvana Andrea Molina Lima, Wilza Carla Spiri, and Carmen Silva Gabriel. Incidentes relacionados a medicamentos em uma instituição hospitalar: subsídios para a melhoria da gestão [drug-related incidents in a hospital: input to improving management]. *Revista enfermagem UERJ*, 23(5):616–621, 2015.

[38] Fabiana Divina de Brito Amorim, Paula Vanessa Peclat Flores, Priscila Sanchez Bosco, Andréia Holanda Barbosa Menezes, and Kyra Vianna Alóchio. O aprazamento de medicamentos pautado na segurança do paciente: um alerta para prática de enfermagem. *Revista de Enfermagem UFPE on line*, 8(1):224–228, 2014.

[39] Giuseppe De Giacomo and Maurizio Lenzerini. Tbox and abox reasoning in expressive description logics. *KR*, 96(316-327):10, 1996.

[40] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[41] Leonardo de Moura, Bruno Dutertre, and Natarajan Shankar. A tutorial on satisfiability modulo theories: (invited tutorial). In *International conference on computer aided verification*, pages 20–36. Springer, 2007.

[42] Henrique Souza Barros de Oliveira, Jamile Rafaela Poltronieri de Sousa, Ana Carolina Gariba Donis, and Maria Elisa Gonzalez Manso. Utilização dos critérios de beers para avaliação das prescrições em idosos portadores de doenças crônicas vinculados a um plano de saúde. *Revista Brasileira de Ciências do Envelhecimento Humano*, 14(3), 2017.

[43] Luciana Mello de Oliveira, Juliana do Amaral Carneiro Diel, Alessandra Nunes, and Tatiane da Silva Dal Pizzol. Prevalence of drug interactions

in hospitalised elderly patients: a systematic review. *European Journal of Hospital Pharmacy*, 28(1):4–9, 2021.

[44] Kelly Cristina Batista de Queiroz, Maria de Fátima da Silva Nascimento, Vander Fernandes, and Fábio André Miotto. Análise de interações medicamentosas identificadas em prescrições da uti neonatal da icu-hgu. *Journal of Health Sciences*, 16(3), 2014.

[45] Jeffrey C Delafuente. Understanding and preventing drug interactions in elderly patients. *Critical reviews in oncology/hematology*, 48(2):133–143, 2003.

[46] Kim H DeRhodes. The dangers of ignoring the beers criteria—the prescribing cascade. *JAMA Internal Medicine*, 179(7):863–864, 2019.

[47] Aude Desnoyer, Anne-Laure Blanc, Valérie Pourcher, Marie Besson, Caroline Fonzo-Christe, Jules Desmeules, Arnaud Perrier, Pascal Bonnabry, Caroline Samer, and Bertrand Guignard. Pim-check: development of an international prescription-screening checklist designed by a delphi method for internal medicine patients. *BMJ open*, 7(7):e016070, 2017.

[48] James G Dolan. Multi-criteria clinical decision support: a primer on the use of multiple-criteria decision-making methods to promote evidence-based, patient-centered healthcare. *The Patient: Patient-Centered Outcomes Research*, 3:229–248, 2010.

[49] Bruno Dutertre. Yices 2.2. In *International Conference on Computer Aided Verification*, pages 737–744. Springer, 2014.

[50] Shaker H El-Sappagh and Samir El-Masri. A distributed clinical decision support system architecture. *Journal of King Saud University-Computer and Information Sciences*, 26(1):69–78, 2014.

[51] CoolApps Entertainment. Malpip, stopp/start, beers, 2023. URL **https://play.google.com/store/apps/details?id=pack.geridea&hl=pt&gl=US**.

[52] Joan Espasa Arxer et al. Smt techniques for planning problems. 2018.

[53] Mary Ane Lessa Etelvino, Noemi Duque dos Santos, Beatriz Gerbassi Costa Aguiar, and Tamyris Garcia de Assis. Segurança do paciente: uma análise do aprazamento de medicamentos. *Enfermagem em Foco*, 10(4), 2019.

[54] Rui Feng, Qiping Hu, and Yingan Jiang. Unknown disease outbreaks detection: A pilot study on feature-based knowledge representation and reasoning model. *Frontiers in Public Health*, 9:535, 2021.

[55] Donna M Fick, James W Cooper, William E Wade, Jennifer L Waller, J Ross Maclean, and Mark H Beers. Updating the beers criteria for potentially inappropriate medication use in older adults: results of a us consensus panel of experts. *Archives of internal medicine*, 163(22):2716–2724, 2003.

[56] Eliana Frutos, Martin Kakazu, Matias Tajerian, Alejandro Gaiera, Luciana Rubin, Carlos Otero, and Daniel Luna. Clinical decision support system for pim in elderly patients: Implementation and initial evaluation in ambulatory care. *Studies in Health Technology and Informatics*, 294:475–479, 2022.

[57] Tomás M García-Caballero, Juan Lojo, Carlos Menéndez, Roberto Fernández-Álvarez, Raimundo Mateos, and Alejandro Garcia-Caballero. Polimedication: applicability of a computer tool to reduce polypharmacy in nursing homes. *International psychogeriatrics*, 30(7):1001–1008, 2018.

[58] Simona Ghibelli, Alessandra Marengoni, Codjo D Djade, Alessandro Nobili, Mauro Tettamanti, Carlotta Franchi, Silvio Caccia, Flavio Giovarruscio, Andrea Remuzzi, and Luca Pasina. Prevention of inappropriate prescribing in hospitalized older patients using a computerized prescription support system (intercheck®). *Drugs & aging*, 30(10):821–828, 2013.

[59] Christoph D Gladisch. Satisfiability solving and model generation for quantified first-order logic formulas. In *International Conference on Formal Verification of Object-Oriented Software*, pages 76–91. Springer, 2010.

[60] Adela Grando, Susan Farrish, Cynthia Boyd, and Aziz Boxwala. Ontological approach for safe and effective polypharmacy prescription. In *AMIA Annual Symposium Proceedings*, volume 2012, page 291. American Medical Informatics Association, 2012.

[61] Hima Bindu Gujjarlamudi. Polytherapy and drug interactions in elderly. *Journal of mid-life health*, 7(3):105, 2016.

[62] Akash Gupta, Tieming Liu, and Scott Shepherd. Clinical decision support system to assess the risk of sepsis using tree augmented bayesian networks and electronic medical record data. *Health informatics journal*, 26(2):841–861, 2020.

[63] Joseph T Hanlon, Todd P Semla, and Kenneth E Schmader. Alternative medications for medications in the use of high-risk medications in the elderly and potentially harmful drug–disease interactions in the elderly quality measures. *Journal of the American Geriatrics Society*, 63(12):e8–e18, 2015.

[64] A Harugeri, J Joseph, G Parthasarathi, M Ramesh, S Guido, et al. Potentially inappropriate medication use in elderly patients: a study of prevalence and predictors in two teaching hospitals. *Journal of postgraduate medicine*, 56(3): 186, 2010.

[65] Phyllis Heintz and Malcolm Buchholz. After rescue: The importance of beers criteria for medication assessment in older adults. *Critical Care Nursing Quarterly*, 38(3):312–316, 2015.

[66] Mohammad Hekmatnejad, Andrew M Simms, and Georgios Fainekos. Model checking clinical decision support systems using smt. *arXiv preprint arXiv:1901.04545*, 2019.

[67] Heinrich Herre. Formal ontology and the foundation of knowledge organization. *KO KNOWLEDGE ORGANIZATION*, 40(5):332–339, 2014.

[68] Lisa E Hines and John E Murphy. Potentially harmful drug–drug interactions in the elderly: a review. *The American journal of geriatric pharmacotherapy*, 9 (6):364–377, 2011.

[69] PC Hsiu, HC Yeh, PH Tsai, CS Shih, DH Burkhardt, TW Kuo, JWS Liu, TY Huang, et al. A general model for medication scheduling. *Institute of Information Science, Academia Sinica, Taiwan, Technical Report TR-IIS-05-008*, 2005.

[70] Chi-Hsien Huang, Hiroyuki Umegaki, Yuuki Watanabe, Hiroko Kamitani, Atushi Asai, Shigeru Kanda, Hideki Nomura, and Masafumi Kuzuya. Potentially inappropriate medications according to stopp-j criteria and risks of hospitalization and mortality in elderly patients receiving home-based medical services. *Plos one*, 14(2):e0211947, 2019.

[71] Jialiang Huang, Chaoqun Niu, Christopher D Green, Lun Yang, Hongkang Mei, and Jing-Dong J Han. Systematic prediction of pharmacodynamic drug-drug interactions through protein-protein-interaction network. *PLoS computational biology*, 9(3):e1002998, 2013.

[72] Ian Hyland and Renate A Schmidt. Protege-ts: An owl ontology term selection tool. In *Description Logics*, 2020.

[73] AGS Health in Aging Foundation. Tip sheet: Alternatives for medications listed in the ags beers criteria® for potentially inappropriate medication use in older adults. URL https://www.healthinaging.org/tools-and-tips/tip-sheet-alternatives-medications-listed-ags-beers-criteriar-potentially.

[74] Pengli Jia, Longhao Zhang, Jingjing Chen, Pujing Zhao, and Mingming Zhang. The effects of clinical decision support systems on medication safety: an overview. *PloS one*, 11(12):e0167683, 2016.

[75] Yicheng Jiang, Bensheng Qiu, Chunsheng Xu, Chuanfu Li, et al. The research of clinical decision support system based on three-layer knowledge base model. *Journal of healthcare engineering*, 2017, 2017.

[76] Rose-Marie Johansson-Pajala, Lene Martin, and Kerstin Jorsäter Blomgren. Registered nurses' use of computerised decision support in medication reviews: Implications in swedish nursing homes. *International Journal of Health Care Quality Assurance*, 31(6):531–544, 2018.

[77] Hanumanthrao Kannan. Formal reasoning of knowledge in systems engineering through epistemic modal logic. *Systems Engineering*, 24(1): 3–16, 2021.

[78] Hanumanthrao Kannan. Knowledge representation and reasoning in the context of systems engineering. In *Recent Trends and Advances in Model Based Systems Engineering*, pages 217–227. Springer, 2022.

[79] Masoudeh Keshavarzi and Hamid Reza Ghaffary. An ontology-driven framework for knowledge representation of digital extortion attacks. *Computers in Human Behavior*, page 107520, 2022.

[80] Ahlem Chérifa Khadir, Hassina Aliane, and Ahmed Guessoum. Ontology learning: Grand tour and challenges. *Computer Science Review*, 39:100339, 2021.

[81] Mohamed Khalifa. Clinical decision support: Strategies for success. *Procedia Computer Science*, 37:422–427, 2014.

[82] J Kim and O De Jesus. Medication routes of administration. statpearls, 2022.

[83] Andrii Kovalov and Juliana Küster Filipe Bowles. Avoiding medication conflicts for patients with multimorbidities. In *International Conference on Integrated Formal Methods*, pages 376–390. Springer, 2016.

[84] Butch KuKanich. Clinical interpretation of pharmacokinetic and pharmacodynamic data in zoologic companion animal species. *Veterinary Clinics: Exotic Animal Practice*, 14(1):1–20, 2011.

[85] Jean-Baptiste Lamy. Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine*, 80:11–28, 2017.

[86] Louis Létinier, Sébastien Cossin, Yohann Mansiaux, Mickaël Arnaud, Francesco Salvo, Julien Bezin, Frantz Thiessard, and Antoine Pariente. Risk of drug-drug interactions in out-hospital drug dispensings in france: results from the drug-drug interaction prevalence study. *Frontiers in Pharmacology*, 10:265, 2019.

[87] Leonardo Lezcano, Miguel-Angel Sicilia, and Carlos Rodríguez-Solano. Integrating reasoning and clinical archetypes using owl ontologies and swrl rules. *Journal of biomedical informatics*, 44(2):343–353, 2011.

[88] Yi Li, Aws Albarghouthi, Zachary Kincaid, Arie Gurfinkel, and Marsha Chechik. Symbolic optimization with smt solvers. *ACM SIGPLAN Notices*, 49(1):607–618, 2014.

[89] Bárbara Heather Lutz, Vanessa Irribarem Avena Miranda, and Andréa Dâmaso Bertoldi. Potentially inappropriate medications among older adults in pelotas, southern brazil. *Revista de saude publica*, 51:52, 2017.

[90] Alpana Mair, Martin Wilson, and Tobias Dreischulte. Addressing the challenge of polypharmacy. *Annual review of pharmacology and toxicology*, 60: 661–681, 2020.

[91] Sharad Malik and Georg Weissenbacher. Boolean satisfiability solvers: techniques and extensions. *Software Safety & Security Tools for Analysis and Verification; IOS Press: Amsterdam, The Netherlands*, 2012.

[92] Louise Mallet, Anne Spinewine, and Allen Huang. The challenge of managing drug interactions in elderly people. *The Lancet*, 370(9582):185–191, 2007.

[93] Arduino A Mangoni and Stephen HD Jackson. Age-related changes in pharmacokinetics and pharmacodynamics: basic principles and practical applications. *British journal of clinical pharmacology*, 57(1):6–14, 2004.

[94] Alessandra Marengoni, Alessandro Nobili, and Graziano Onder. Best practices for drug prescribing in older adults: a call for action. *Drugs & aging*, 32(11):887–890, 2015.

[95] Emily G McDonald, Peter E Wu, Babak Rashidi, Alan J Forster, Allen Huang, Louise Pilote, Louise Papillon-Ferland, André Bonnici, Robyn Tamblyn, Rachel Whitty, et al. The medsafer study: a controlled trial of an electronic decision support tool for deprescribing in acute care. *Journal of the American Geriatrics Society*, 67(9):1843–1850, 2019.

[96] Juan Jose Mendoza Santana and Juliana Kuster Filipe Bowles. Formal reasoning over class models with tomm. *Journal of Object Technology*, 2019.

[97] Aniello Minutolo, Massimo Esposito, and Giuseppe De Pietro. A pattern-based knowledge editing system for building clinical decision support systems. *Knowledge-Based Systems*, 35:120–131, 2012.

[98] Luís Monteiro, Tiago Maricoto, Isabel Solha, Inês Ribeiro-Vaz, Carlos Martins, Matilde Monteiro-Soares, et al. Reducing potentially inappropriate prescriptions for older patients using computerized decision support tools: systematic review. *Journal of medical Internet research*, 21(11):e15385, 2019.

[99] Manuel Montero-Odasso, Yanina Sarquis-Adamson, Hao Yuan Song, Nick Walter Bray, Frederico Pieruccini-Faria, and Mark Speechley. Polypharmacy, gait performance, and falls in community-dwelling older adults. results from the gait and brain study. *Journal of the American Geriatrics Society*, 67(6):1182–1188, 2019.

[100] Francisca Sueli Monte Moreira, Javier Jerez-Roig, Lidiane Maria de Brito Macedo Ferreira, Ana Patricia de Queiroz Medeiros Dantas, Kenio Costa Lima, and Maria Ângela Fernandes Ferreira. Use of potentially inappropriate medications in institutionalized elderly: prevalence and associated factors. *Ciência & Saúde Coletiva*, 25:2073–2082, 2020.

[101] Abdelmalek Mouazer, Karima Sedki, Rosy Tsopra, and Jean-Baptiste Lamy. Speak-pim, towards a framework for the automatic detection of potentially inappropriate prescriptions. *Studies in health technology and informatics*, 294: 460–464, 2022.

[102] K Nagai, S Fukuno, R Moriwaki, H Kuroda, S Omotani, T Miura, Y Hatsuda, M Myotoku, and H Konishi. Influence of concurrent and staggered dosing of semi-solid nutrients on the pharmacokinetics of orally administered carbamazepine in rats. *Die Pharmazie-An International Journal of Pharmaceutical Sciences*, 77(3-4):118–120, 2022.

[103] Tabbasum Naz, Muhammad Akhtar, Syed Khuram Shahzad, Maria Fasli, Muhammad Waseem Iqbal, and Muhammad Raza Naqvi. Ontology-driven advanced drug-drug interaction. *Computers & Electrical Engineering*, 86: 106695, 2020.

[104] M Neugebauer, M Ebert, and R Vogelmann. A clinical decision support system improves antibiotic therapy for upper urinary tract infection in a randomized single-blinded study. *BMC health services research*, 20(1):1–10, 2020.

[105] Kristina M Niehoff, Nallakkandi Rajeevan, Peter A Charpentier, Perry L Miller, Mary K Goldstein, and Terri R Fried. Development of the tool to reduce inappropriate medications (trim): a clinical decision support system to improve medication prescribing for older adults. *Pharmacotherapy: The Journal of Human Pharmacology and Drug Therapy*, 36(6):694–701, 2016.

[106] Robert Nieuwenhuis and Albert Oliveras. On sat modulo theories and optimization problems. In *Theory and Applications of Satisfiability Testing-SAT 2006: 9th International Conference, Seattle, WA, USA, August 12-15, 2006. Proceedings 9*, pages 156–169. Springer, 2006.

[107] Yasiru Nilan, Darika Sellahewa, Shalith Fernando, Lakshan Gamage, and Dulani Meedeniya. A clinical decision support system for drug conflict identification. In *2018 Moratuwa Engineering Research Conference (MERCon)*, pages 126–131. IEEE, 2018.

[108] Jin Niu, Robert M Straubinger, and Donald E Mager. Pharmacodynamic drug–drug interactions. *Clinical Pharmacology & Therapeutics*, 105(6):1395–1406, 2019.

[109] Paulo Roque Obreli-Neto, Alessandro Nobili, André de Oliveira Baldoni, Camilo Molino Guidoni, Divaldo Pereira de Lyra Júnior, Diogo Pilger, Juliano Duzanski, Mauro Tettamanti, Joice Mara Cruciol-Souza, Walderez Penteado Gaeti, et al. Adverse drug reactions caused by drug–drug interactions in elderly outpatients: a prospective cohort study. *European journal of clinical pharmacology*, 68(12):1667–1676, 2012.

[110] Scottish Government Polypharmacy Model of Care Group. Polypharmacy guidance, realistic prescribing 3rd edition. 2018.

[111] Denis O'Mahony, David O'Sullivan, Stephen Byrne, Marie Noelle O'Connor, Cristin Ryan, and Paul Gallagher. Stopp/start criteria for potentially inappropriate prescribing in older people: version 2. *Age and ageing*, 44 (2):213–218, 2014.

[112] 'World Health Organization and others'. Multimorbidity. 2016.

[113] 'World Health Organization and others'. Medication without harm. Technical report, World Health Organization, 2017.

[114] Martin O'connor, Holger Knublauch, Samson Tu, and Mark Musen. Writing rules for the semantic web using swrl and jess. *Protégé With Rules WS, Madrid*, 2005.

[115] American Geriatrics Society 2012 Beers Criteria Update Expert Panel. A merican g eriatrics s ociety updated b eers c riteria for potentially inappropriate medication use in older adults. *Journal of the American Geriatrics Society*, 60 (4):616–631, 2012.

[116] American Geriatrics Society 2015 Beers Criteria Update Expert Panel, Donna M Fick, Todd P Semla, Judith Beizer, Nicole Brandt, Robert Dombrowski, Catherine E DuBeau, Woody Eisenberg, Jerome J Epplin, Nina Flanagan, et al. American geriatrics society 2015 updated beers criteria for potentially inappropriate medication use in older adults. *Journal of the American Geriatrics Society*, 63(11):2227–2246, 2015.

[117] American Geriatrics Society 2019 Beers Criteria® Update Expert Panel, Donna M Fick, Todd P Semla, Michael Steinman, Judith Beizer, Nicole Brandt, Robert Dombrowski, Catherine E DuBeau, Lynn Pezzullo, Jerome J Epplin, et al. American geriatrics society 2019 updated ags beers criteria®

for potentially inappropriate medication use in older adults. *Journal of the American Geriatrics Society*, 67(4):674–694, 2019.

[118] American Geriatrics Society 2023 Beers Criteria® Update Expert Panel. American geriatrics society 2023 updated ags beers criteria® for potentially inappropriate medication use in older adults. *Journal of the American Geriatrics Society*, 2023.

[119] Dominic Papandria, Jeremy G Fisher, Brian D Kenney, Michael Dykes, Abigail Nelson, and Karen A Diefenbach. Orientation in perpetuity: an online clinical decision support system for surgical residents. *Journal of Surgical Research*, 245:649–655, 2020.

[120] Maxime Peralta, Pierre Jannin, and John SH Baxter. Machine learning in deep brain stimulation: A systematic review. *Artificial intelligence in medicine*, 122:102198, 2021.

[121] Francisco Gilberto Fernandes Pereira, GA Melo, Nelson Miguel Galindo Neto, RE Carvalho, ED Néri, and JA Caetano. Interações medicamentosas induzidas pelo aprazamento e os erros no preparo de antibacterianos. *Revista Rene*, 19, 2018.

[122] Francisco Gilberto Fernandes Pereira, Geórgia Alcântara Alencar Melo, Nelson Miguel Galindo Neto, Rhanna Emanuela Fontenele Lima de Carvalho, Eugenié Desireé Rabelo Néri, and Joselany Áfio Caetano. Drug interactions resulting from scheduling and errors in the preparation of antibacterials. 2018.

[123] Leonard Petnga. Ontology-driven knowledge modeling and reasoning for multi-domain system architecting and configuration. In *Recent Trends and Advances in Model Based Systems Engineering*, pages 229–239. Springer, 2022.

[124] Rachel Phillips, Lorna Hazell, Odile Sauzet, and Victoria Cornelius. Analysis and reporting of adverse events in randomised controlled trials: a review. *BMJ open*, 9(2):e024537, 2019.

[125] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):7–34, 2014.

[126] Vanida Prasert, Aiko Shono, Farsai Chanjaruporn, Chanuttha Ploylearmsang, Keerataphan Boonnan, Apinan Khampetdee, and Manabu Akazawa. Effect of a computerized decision support system on potentially inappropriate medication prescriptions for elderly patients in thailand. *Journal of Evaluation in Clinical Practice*, 25(3):514–520, 2019.

[127] Zhenisgul Rakhmetullina, Raushan Mukhamedova, Roza Mukasheva, and Elvira Aitmukhanbetova. Mathematical model for clinical decision support system using genetic algorithm. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–5. IEEE, 2020.

[128] Guilherme Redeker and Juliana Bowles. Tackling polypharmacy: A multisource decision support system. In *Digital Personalized Health and Medicine*, pages 688–692. IOS Press, 2020.

[129] Anna Renom-Guiteras, Gabriele Meyer, and Petra A Thürmann. The eu (7)-pim list: a list of potentially inappropriate medications for older people consented by experts from seven european countries. *European journal of clinical pharmacology*, 71(7):861–875, 2015.

[130] José A Riaza and Ginés Moreno. Using sat/smt solvers for efficiently tuning fuzzy logic programs. In *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2020.

[131] Sarmad Riazi and Bengt Lennartson. Using cp/smt solvers for scheduling and routing of agvs. *IEEE Transactions on Automation Science and Engineering*, 18(1):218–229, 2020.

[132] Eloisa Rogero-Blanco, Juan A Lopez-Rodriguez, Teresa Sanz-Cuesta, Mercedes Aza-Pascual-Salcedo, M Jose Bujalance-Zafra, Isabel Cura-Gonzalez, et al. Use of an electronic clinical decision support system in primary care to assess inappropriate polypharmacy in young seniors with multimorbidity: observational, descriptive, cross-sectional study. *JMIR medical informatics*, 8 (3):e14130, 2020.

[133] Sabino Francesco Roselli, Kristofer Bengtsson, and Knut Åkesson. Smt solvers for job-shop scheduling problems: Models comparison and performance evaluation. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 547–552. IEEE, 2018.

[134] Elizabeth E Roughead, Susan J Semple, and Ellie Rosenfeld. The extent of medication errors and adverse drug reactions throughout the patient journey in acute care in australia. *International journal of evidence-based healthcare*, 14 (3-4):113–122, 2016.

[135] Malcolm Rowland and Shaikh B Matin. Kinetics of drug-drug interactions. *Journal of Pharmacokinetics and Biopharmaceutics*, 1(6):553–567, 1973.

[136] Serhad Sarica and Jianxi Luo. Design knowledge representation with technology semantic network. *Proceedings of the Design Society*, 1:1043–1052, 2021.

[137] Stefan Schulz and Ludger Jansen. Formal ontologies in biomedical knowledge representation. *Yearbook of medical informatics*, 22(01):132–146, 2013.

[138] Oshani Seneviratne, Amar K Das, Shruthi Chari, Nkechinyere N Agu, Sabbir M Rashid, Ching-Hua Chen, James P McCusker, James A Hendler, and Deborah L McGuinness. Enabling trust in clinical decision support recommendations through semantics. In *SeWeBMeDa@ ISWC*, pages 55–67, 2019.

[139] Amir Mohammad Shahsavarani, Esfandiar Azad Marz Abadi, Maryam Hakimi Kalkhoran, Saeideh Jafari, and Shirin Qaranli. Clinical decision support systems (cdsss): state of the art review of literature. *International Journal of Medical Reviews*, 2(4):299–308, 2015.

[140] Rishabh Sharma, Malika Arora, Ravinder Garg, and Parveen Bansal. A closer look at the 2019 beers criteria. *Drugs & Therapy Perspectives*, 36(3): 116–122, 2020.

[141] Rishabh Sharma, Parveen Bansal, Ravinder Garg, Ravi Ranjan, Rakesh Kumar, and Malika Arora. Prevalence of potentially inappropriate medication and its correlates in elderly hospitalized patients: A cross-sectional study based on beers criteria. *Journal of Family & Community Medicine*, 27(3):200, 2020.

[142] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5 (2):51–53, 2007.

[143] David M Smith, Charlotte Friend, and Joanne Reeve. Polypharmacy and rationalisation of medications. *InnovAiT*, 13(2):87–93, 2020.

[144] Ben D Snyder, Thomas M Polasek, and Matthew P Doogue. Drug interactions: principles and practice. 2012.

[145] American Geriatrics Society. Ags beers criteria® for potentially inappropriate medication use in older adults, the ags beers criteria®, 2023. URL https://geriatricscareonline.org/ProductAbstract/ags-beers-criteria-for-potentially-inappropriate-medications-for-older-adults-mobile-app/B067.

[146] Andreas Sönnichsen, Ulrike S Trampisch, Anja Rieckert, Giuliano Piccoliori, Anna Vögele, Maria Flamm, Tim Johansson, Aneez Esmail, David Reeves, Christin Löffler, et al. Polypharmacy in chronic diseases-reduction of inappropriate medication and adverse drug events in older populations by electronic decision support (prima-eds): study protocol for a randomized controlled trial. *Trials*, 17(1):1–9, 2016.

[147] Robert Stevens, Carole A Goble, and Sean Bechhofer. Ontology-based knowledge representation for bioinformatics. *Briefings in bioinformatics*, 1(4): 398–414, 2000.

[148] Reed T Sutton, David Pincock, Daniel C Baumgart, Daniel C Sadowski, Richard N Fedorak, and Karen I Kroeker. An overview of clinical decision support systems: benefits, risks, and strategies for success. *NPJ Digital Medicine*, 3(1):1–10, 2020.

[149] Mohammad Mustafa Taye. Understanding semantic web and ontologies: Theory and applications. *arXiv preprint arXiv:1006.4567*, 2010.

[150] Kristine Thorell, Patrik Midlöv, Johan Fastbom, and Anders Halling. Use of potentially inappropriate medication and polypharmacy in older adults: a repeated cross-sectional study. *BMC geriatrics*, 20(1):1–9, 2020.

[151] Carlos Toro, Manuel Graña, Eider Sanchez, Cesar Sanin, and Edward Szczerbicki. Experience based clinical decision support systems: An overview and case studies. *Knowledge Management and Engineering with Decisional DNA*, pages 151–188, 2020.

[152] Johannes P van den Berg, Hugo EM Vereecke, JH Proost, Douglas J Eleveld, JK Götz Wietasch, Anthony R Absalom, and Michel MRF Struys. Pharmacokinetic and pharmacodynamic interactions in anaesthesia. a review of current knowledge and how it can be used to optimize anaesthetic drug administration. *BJA: British Journal of Anaesthesia*, 118(1):44–57, 2017.

[153] Heleen van der Sijs, Laureen Lammers, Annemieke van den Tweel, Jos Aarts, Marc Berg, Arnold Vulto, and Teun van Gelder. Time-dependent drug–drug interaction alerts in care provider order entry: software may inhibit medication error reductions. *Journal of the American Medical Informatics Association*, 16(6):864–868, 2009.

[154] Eduard E Vasilevskis, Avantika S Shah, Emily K Hollingsworth, Matthew S Shotwell, Amanda S Mixon, Susan P Bell, Sunil Kripalani, John F Schnelle, and Sandra F Simmons. A patient-centered deprescribing intervention for hospitalized older patients with polypharmacy: rationale and design of the shed-meds randomized controlled trial. *BMC health services research*, 19(1): 1–13, 2019.

[155] Sanne Verdoorn, Henk-Frans Kwint, Petra Hoogland, Jacobijn Gussekloo, and Marcel L Bouvy. Drug-related problems identified during medication review before and after the introduction of a clinical decision support system. *Journal of clinical pharmacy and therapeutics*, 43(2):224–231, 2018.

[156] ATM Wasylewicz and AMJW Scheepers-Hoeks. Clinical decision support systems. *Fundamentals of clinical data science*, pages 153–169, 2019.

[157] Nedra S Whitehead, Laurina Williams, Sreelatha Meleth, Sara Kennedy, Nneka Ubaka-Blackmoore, Michael Kanter, Kevin J O'Leary, David Classen, Brian Jackson, Daniel R Murphy, et al. The effect of laboratory test–based clinical decision support tools on medication errors and adverse drug events: A laboratory medicine best practices systematic review. *The journal of applied laboratory medicine*, 3(6):1035–1048, 2019.

[158] Szymon Wilk, Wojtek Michalowski, Martin Michalowski, Ken Farion, Marisela Mainegra Hing, and Subhra Mohapatra. Mitigation of adverse interactions in pairs of clinical practice guidelines using constraint logic programming. *Journal of biomedical informatics*, 46(2):341–353, 2013.

[159] David S Wishart, Craig Knox, An Chi Guo, Dean Cheng, Savita Shrivastava, Dan Tzur, Bijaya Gautam, and Murtaza Hassanali. Drugbank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic acids research*, 36 (suppl_1):D901–D906, 2008.

[160] World Health Organization. Who calls for urgent action by countries for achieving medication without harm, 2022. URL https://www.who.int/news/item/16-09-2022-who-calls-for-urgent-action-by-countries-for-achieving-medication-without-harm. Accessed: November 8, 2023.

[161] World Wide Web Consortium (W3C). Swrl: A semantic web rule language combining owl and ruleml, 2004. URL https://www.w3.org/submissions/SWRL/. Accessed on October 13, 2023.

[162] Fran Wu, Mitch Williams, Peter Kazanzides, K Brady, and J Fackler. A modular clinical decision support system. In *Proc. of 3rd International Conference on Pervasive Computing Technologies for Healthcare,(PervasiveHealth) and Workshops*, 2009.

[163] Guizhen Yang, Michael Kifer, and Chang Zhao. Flora-2: A rule-based knowledge representation and inference infrastructure for the semantic web. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 671–688. Springer, 2003.

[164] Jiansong Yang, Maria Kjellsson, Amin Rostami-Hodjegan, and Geoffrey T Tucker. The effects of dose staggering on metabolic drug–drug interactions. *European journal of pharmaceutical sciences*, 20(2):223–232, 2003.

[165] Hyeong Won Yu, Maqbool Hussain, Muhammad Afzal, Taqdir Ali, June Young Choi, Ho-Seong Han, and Sungyoung Lee. Use of mind maps and iterative decision trees to develop a guideline-based clinical decision support system for routine surgical practice: case study in thyroid nodules. *Journal of the American Medical Informatics Association*, 26(6):524–536, 2019.

[166] Mariam Zahwe, Hadi Skouri, Samar Rachidi, Maurice Khoury, Samar Noureddine, Hussain Isma'eel, Hani Tamim, and Amal Al-Hajje. Potentially inappropriate medications in elderly patients with heart failure: Beers criteria-based study. *International Journal of Pharmacy Practice*, 28(6):652–659, 2020.

[167] Yael Zenziper, Daniel Kurnik, Noa Markovits, Amitai Ziv, Ari Shamiss, Hillel Halkin, and Ronen Loebstein. Implementation of a clinical decision support system for computerized drug prescription entries in a large tertiary care hospital. *The Israel Medical Association Journal: IMAJ*, 16(5):289–294, 2014.

[168] BT Zhong, LY Ding, Peter ED Love, and HB Luo. An ontological approach for technical plan definition and verification in construction. *Automation in Construction*, 55:47–57, 2015.

Appendix A

Ethics Approvals

University of St Andrews | FOUNDED 1413

School of Computer Science Ethics Committee

31 May 2021

Dear Guilherme,

Thank you for submitting your ethical application which was considered by the School Ethics Committee.

The School of Computer Science Ethics Committee, acting on behalf of the University Teaching and Research Ethics Committee (UTREC), has approved this application:

Approval Code:	CS15485	**Approved on:**	31.05.2021	**Approval Expiry:**	31.05.2026
Project Title:	Searching for hospital prescriptions with drug interactions when treating elderly patients				
Researcher(s):	Guilherme Alfredo Redeker				
Supervisor(s):	Dr Juliana Bowles				

The following supporting documents are also acknowledged and approved:

1. Application Form
2. Consent and Authorisation Documents

Approval is awarded for 5 years, see the approval expiry data above.

If your project has not commenced within 2 years of approval, you must submit a new and updated ethical application to your School Ethics Committee.

If you are unable to complete your research by the approval expiry date you must request an extension to the approval period. You can write to your School Ethics Committee who may grant a discretionary extension of up to 6 months. For longer extensions, or for any other changes, you must submit an ethical amendment application.

You must report any serious adverse events, or significant changes not covered by this approval, related to this study immediately to the School Ethics Committee.

Approval is given on the following conditions:

- that you conduct your research in line with:
  - the details provided in your ethical application
  - the University's Principles of Good Research Conduct
  - the conditions of any funding associated with your work
- that you obtain all applicable additional documents (see the 'additional documents' webpage for guidance) before research commences.

You should retain this approval letter with your study paperwork.

Yours sincerely,

SEC Administrator

School of Computer Science Ethics Committee
Dr Juan Ye/Convenor, Jack Cole Building, North Haugh, St Andrews, Fife, KY16 9SX
Telephone: 01334 463252 Email: ethics-cs@st-andrews.ac.uk

APPENDIX B

ONTOLOGY DETAILS

B.1 Accessing ontology files

The ontology relevant files are available in the GitHub repository:
https://github.com/gr60/ThesisFiles

The ontology components, comprising classes, objects, and data properties, along with inference rules and relevant details, can be readily accessed and viewed in the file named TheBeersCriteriaOntology.owl. This file can be conveniently opened using the Protégé software Protégé.

APPENDIX C

SEMANTIC WEB RULE LANGUAGE (SWRL) RULES TO DETECT INAPPROPRIATE DRUGS

This appendix provides the SWRL rules developed on the Beers Criteria ontology to detect and classify prescribed inappropriate drugs. The rules are listed according to the Beers Criteria categories.

C.1 Potentially Clinically Important Drug-Drug Interactions (DDI) Rules

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Central_nervous_system_active_drugs(?d)
3^ hasDrug(?pr, ?d2) ^ Central_nervous_system_active_drugs(?d2)
4^ hasDrug(?pr, ?d3) ^ Central_nervous_system_active_drugs(?d3)
5^ hasPatientAgeValue(?p, ?a) ^ swrlb:greaterThan(?a, 64)
6^ differentFrom(?d, ?d2) ^ differentFrom(?d2, ?d3)
7^ differentFrom(?d, ?d3) -> DDI_CNS_Active_Drugs/
     CNS_Active_Drugs(?d)
8^ DDI_CNS_Active_Drugs/CNS_Active_Drugs(?d2)
9^ DDI_CNS_Active_Drugs/CNS_Active_Drugs(?d3) ^
     hasInteractionWith(?d, ?d2)
10^ hasInteractionWith(?d2, ?d) ^ hasInteractionWith(?d, ?d3)
11^ hasInteractionWith(?d3, ?d) ^ hasInteractionWith(?d3, ?d2)
12^ hasInteractionWith(?d2, ?d3
```

Listing C.1: DDI_CNS_Active_Drugs/CNS_Active_Drugs

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^hasDrug(?pr,?d)^Drugs_With_Strong_Anticholinergic_Properties(?d)

3^hasDrug(?pr,?d2)^Drugs_With_Strong_Anticholinergic_Properties(?
    d2)
4^hasPatientAgeValue(?p, ?a) ^ swrlb:greaterThan(?a, 64)
5^differentFrom(?d,?d2) -> DDI_Anticholinergic/Anticholinergic(?d)

6^DDI_Anticholinergic/Anticholinergic(?d2) ^ hasInteractionWith(?
    d, ?d2)
7^hasInteractionWith(?d2, ?d)
```

Listing C.2: DDI_Anticholinergic/Anticholinergic

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Corticosteroids(?d)^Nonsteroidal_Anti-inflammatory_Agents(?d2)
3^ hasDrug(?pr,?d) ^ hasDrug(?pr,?d2) ^ hasPatientAgeValue(?p,?a)
4^ swrlb:greaterThan(?a, 64) ^ Oral(?r) ^ hasRoute(?d, ?r) ->
5DDI_Corticosteroids/NSAIDs/Oral(?d) ^ hasInteractionWith(?d2,?d)
6^ DDI_Corticosteroids/NSAIDs/Oral(?d2)^hasInteractionWith(?d,?d2)
```

Listing C.3: DDI_Corticosteroids/NSAIDs/Oral

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Corticosteroids(?d)^Nonsteroidal_Anti-inflammatory_Agents(?d2)
3^ hasDrug(?pr,?d) ^ hasDrug(?pr,?d2) ^ hasPatientAgeValue(?p,?a)
4^ swrlb:greaterThan(?a, 64) ^ Parenteral(?r) ^ hasRoute(?d, ?r)
5-> DDI_Corticosteroids/NSAIDs/Parenteral(?d)
6^ DDI_Corticosteroids/NSAIDs/Parenteral(?d2)
7^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.4: DDI_Corticosteroids/NSAIDs/Parenteral

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Lithium(?d) ^ hasDrug(?pr, ?d2)
3^ Angiotensin-Converting_Enzyme_Inhibitors(?d2)
4^ hasDrug(?pr,?d) ^ hasPatientAgeValue(?p,?a)
5^ swrlb:greaterThan(?a, 64)
6-> DDI_Lithium/ACEIs(?d) ^ DDI_Lithium/ACEIs(?d2)
7^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.5: DDI_Lithium/ACEIs

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Lithium(?d) ^ hasDrug(?pr, ?d2) ^ Loop_Diuretics(?d2)
3^ hasDrug(?pr, ?d) ^ hasPatientAgeValue(?p, ?a)
4^ swrlb:greaterThan(?a, 64)
5-> DDI_Lithium/Loop_diuretics(?d)
6^ DDI_Lithium/Loop_diuretics(?d2)
7^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.6: DDI_Lithium/Loop_diuretics

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Opiate_Agonists(?d) ^ hasDrug(?pr,?d2) ^ Benzodiazepines(?d2)
3^ hasDrug(?pr, ?d) ^ hasPatientAgeValue(?p, ?a)
4^ swrlb:greaterThan(?a, 64) -> DDI_Opioids/Benzodiazepines(?d)
5^ DDI_Opioids/Benzodiazepines(?d2)
6^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.7: DDI_Opioids/Benzodiazepines

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Opiate_Agonists(?d) ^ hasDrug(?pr, ?d2) ^ Gabapentin(?d2)
3^ hasDrug(?pr, ?d) ^ hasPatientAgeValue(?p, ?a)
4^ swrlb:greaterThan(?a, 64) -> DDI_Opioids/Gabapentin(?d)
5^ DDI_Opioids/Gabapentin(?d2)
6^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.8: DDI_Opioids/Gabapentin

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Opiate_Agonists(?d) ^ hasDrug(?pr, ?d2) ^ Pregabalin(?d2)
3^ hasDrug(?pr, ?d) ^ hasPatientAgeValue(?p, ?a)
4^ swrlb:greaterThan(?a, 64) -> DDI_Opioids/Pregabalin(?d)
5^ DDI_Opioids/Pregabalin(?d2) ^ hasInteractionWith(?d, ?d2)
6^ hasInteractionWith(?d2, ?d)
```

Listing C.9: DDI_Opioids/Pregabalin

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Non-selective_alpha-1-Adrenergic_Blocking_Agents(?d)
3^ Loop_Diuretics(?d2) ^ hasDrug(?pr, ?d) ^ hasDrug(?pr, ?d2)
4^ hasPatientAgeValue(?p, ?a) ^ swrlb:greaterThan(?a, 64)
```

```
5^ Parenteral(?r) ^ hasRoute(?d, ?r)
6-> DDI_Peripheral_alpha-1_blockers/Loop_diuretics(?d)
7^ DDI_Peripheral_alpha-1_blockers/Loop_diuretics(?d2)
8^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.10: DDI_Peripheral_alpha-1_blockers/Loop_diuretics

```
1Phenytoin(?d) ^ Trimethoprim_sulfamethoxazole(?d2) ^ hasDrug(?pr
    ,?d)
2^ Patient(?p) ^ hasPatientAgeValue(?p, ?a) ^ Prescription(?pr)
3^ swrlb:greaterThan(?a, 64) ^ hasPrescription(?p, ?pr)
4^ hasDrug(?pr, ?d2) -> hasInteractionWith(?d, ?d2) ^
    DDI_Phenytoin/Trimethoprim_sulfamethoxazole(?d2)
5^ hasInteractionWith(?d2, ?d)
6^ DDI_Phenytoin/Trimethoprim_sulfamethoxazole(?d)
```

Listing C.11: DDI_Phenytoin/Trimethoprim_sulfamethoxazole

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Renin-Angiotensin-Aldosterone_System_Inhibitors(?d)
3^ Potassium-sparing_Diuretics(?d2) ^ hasDrug(?pr, ?d)
4^ hasDrug(?pr, ?d2) ^ hasPatientAgeValue(?p, ?a)
5^ swrlb:greaterThan(?a, 64)
6-> DDI_Potassium-sparing_diuretics(?d)
7^ DDI_Potassium-sparing_diuretics(?d2)
8^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.12: DDI_Potassium-sparing_diuretics

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ hasDrug(?pr, ?d)^ hasDrug(?pr, ?d2)
3^ Renin-Angiotensin-Aldosterone_System_Inhibitors(?d)
4^ Renin-Angiotensin-Aldosterone_System_Inhibitors(?d2)
5^ differentFrom(?d, ?d2) ^ hasPatientAgeValue(?p, ?a)
6^ swrlb:greaterThan(?a, 64) -> DDI_RAS_inhibitor(?d)
7^ DDI_RAS_inhibitor(?d2)
8^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.13: DDI_RAS_inhibitor

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Theophyllines(?d) ^ Cimetidine(?d2) ^ hasDrug(?pr, ?d)
3^ hasDrug(?pr, ?d2) ^ hasPatientAgeValue(?p, ?a)
4^ swrlb:greaterThan(?a, 64) -> DDI_Theophylline/Cimetidine(?d)
5^ DDI_Theophylline/Cimetidine(?d2) ^ hasInteractionWith(?d, ?d2)
6^ hasInteractionWith(?d2, ?d)
```

Listing C.14: DDI_Theophylline/Cimetidine

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Theophyllines(?d) ^ Ciprofloxacin(?d2)
3^ hasDrug(?pr, ?d) ^ hasDrug(?pr, ?d2)
4^ hasPatientAgeValue(?p, ?a) ^ swrlb:greaterThan(?a, 64)
5-> DDI_Theophylline/Ciprofloxacin(?d)
6^ DDI_Theophylline/Ciprofloxacin(?d2)
7^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.15: DDI_Theophylline/Ciprofloxacin

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Warfarin(?d) ^ Amiodarone(?d2) ^ hasDrug(?pr, ?d)
3^ hasDrug(?pr, ?d2) ^ hasPatientAgeValue(?p, ?a)
4^ swrlb:greaterThan(?a, 64) -> DDI_Warfarin/Amiodarone(?d)
5^ DDI_Warfarin/Amiodarone(?d2)
6^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.16: DDI_Warfarin/Amiodarone

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Warfarin(?d) ^ Ciprofloxacin(?d2) ^ hasDrug(?pr, ?d)
3^ hasDrug(?pr, ?d2) ^ hasPatientAgeValue(?p, ?a)
4^ swrlb:greaterThan(?a, 64) -> DDI_Warfarin/Ciprofloxacin(?d)
5^ DDI_Warfarin/Ciprofloxacin(?d2)
6^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.17: DDI_Warfarin/Ciprofloxacin

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Warfarin(?d) ^ Macrolides(?d2) ^ hasDrug(?pr, ?d)
3^ hasDrug(?pr, ?d2) ^ Azithromycin(?az)
4^ differentFrom(?d2, ?az) ^ hasPatientAgeValue(?p, ?a)
5^ swrlb:greaterThan(?a, 64) -> DDI_Warfarin/Macrolides(?d)
6^ DDI_Warfarin/Macrolides(?d2)
7^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.18: DDI_Warfarin/Macrolides

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Warfarin(?d) ^ Nonsteroidal_Anti-inflammatory_Agents(?d2)
3^ hasDrug(?pr, ?d) ^ hasDrug(?pr, ?d2)
```

```
4^ hasPatientAgeValue(?p, ?a) ^ swrlb:greaterThan(?a, 64) ->
5DDI_Warfarin/NSAIDs(?d) ^ DDI_Warfarin/NSAIDs(?d2)
6^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.19: DDI_Warfarin/NSAIDs

```
1Trimethoprim_sulfamethoxazole(?d2) ^ hasDrug(?pr, ?d)
2^ Patient(?p) ^ hasPatientAgeValue(?p,?a) ^ Prescription(?pr)
3^ swrlb:greaterThan(?a, 64) ^ hasPrescription(?p, ?pr)
4^ Warfarin(?d) ^ hasDrug(?pr, ?d2)
5-> DDI_Warfarin/Trimethoprim_sulfamethoxazole(?d)
6^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
7^ DDI_Warfarin/Trimethoprim_sulfamethoxazole(?d2)
```

Listing C.20: DDI_Warfarin/Trimethoprim_sulfamethoxazole

C.2 Potentially Inappropriate Medication Due to Drug-Disease or Drug-Syndrome Interactions That May Exacerbate the Disease or Syndrome (DDDS) Rules

```
1hasExam(?p,?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)
2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
4^ swrlb:lessThan(?v, 30) ^ Creatinine_Clearancee(?e)
5^ Cyclooxygenase-2_COX-2_Inhibitors(?d) ^ hasRoute(?d, ?r) ^
6Oral(?r) ->
7DDDS_NSAIDs_non-COX_and_COX_selective_oral_and_parenteral_
    nonacetylated_salicylates(?d) ^ hasInteractionWith(?d, ?d)
```

Listing C.21: DDDS_NSAIDs_non-COX_and_COX - Cyclooxygenase - Oral

```
1hasExam(?p,?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)
2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
4^ swrlb:lessThan(?v, 30) ^ Creatinine_Clearancee(?e)
5^ Cyclooxygenase-2_COX-2_Inhibitors(?d) ^ hasRoute(?d, ?r) ^
6Parenteral(?r) ->
7DDDS_NSAIDs_non-COX_and_COX_selective_oral_and-
    parenteral_nonacetylated_salicylates(?d)
8^ hasInteractionWith(?d, ?d)
```

Listing C.22: DDDS_NSAIDs_non-COX_and_COX - Cyclooxygenase - Parenteral

```
1hasExam(?p,?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)
2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ hasDrug(?pr,?d) ^ swrlb:greaterThan(?a,64) ^ swrlb:lessThan(?v,
     30)
4^ Creatinine_Clearancee(?e) ^ Non_COX-2_selective_NSAIDS(?d)
5^ hasRoute(?d, ?r) ^ Oral(?r) ->
6DDDS_NSAIDs_non-COX_and_COX_selective_oral_and-
    parenteral_nonacetylated_salicylates(?d)
7^ hasInteractionWith(?d, ?d)
```

Listing C.23: DDDS_NSAIDs_non-COX_and_COX - Non_COX-2_selective_NSAIDS - Oral

```
1hasExam(?p,?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p, ?a)

2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64) ^ swrlb:lessThan(?
    v, 30)
4^ Creatinine_Clearancee(?e) ^ Non_COX-2_selective_NSAIDS(?d)
5^ hasRoute(?d, ?r) ^ Parenteral(?r) -> DDDS_NSAIDs_non-
    COX_and_COX_selective_oral_and-
    parenteral_nonacetylated_salicylates(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.24: DDDS_NSAIDs_non-COX_and_COX - Non_COX-2_selective_NSAIDS - Parenteral

```
1hasExam(?p,?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)
2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ hasDrug(?pr,?d) ^ swrlb:greaterThan(?a,64) ^ swrlb:lessThan(?v,
     30)
4^ Creatinine_Clearancee(?e) ^ Nonacetylated_salicylates(?d)
5^ hasRoute(?d, ?r) ^ Oral(?r) -> DDDS_NSAIDs_non-
    COX_and_COX_selective_oral_and-
    parenteral_nonacetylated_salicylates(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.25: DDDS_NSAIDs_non-COX_and_COX - Nonacetylated_salicylates - Oral

```
1hasExam(?p,?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)
2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64) ^ swrlb:lessThan(?
    v, 30)
```

```
4^ Creatinine_Clearancee(?e) ^ Nonacetylated_salicylates(?d)
5^ hasRoute(?d, ?r) ^ Parenteral(?r) ->
6DDDS_NSAIDs_non-COX_and_COX_selective_oral_and-
    parenteral_nonacetylated_salicylates(?d)
7^ hasInteractionWith(?d, ?d)
```

Listing C.26: DDDS_NSAIDs_non-COX_and_COX - Nonacetylated_salicylates - Parenteral

C.3 Medications That Should Be Avoided or Have Their Dosage Reduced With Varying Levels of Kidney Function (VLKF) Rules

```
1hasExam(?p, ?e) ^ hasExamValue(?e, ?v) ^ hasPatientAgeValue(?p,?
    a)
2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
4^ swrlb:lessThan(?v, 30) ^ Creatinine_Clearancee(?e)
5^ aMILoride(?d) -> VLKF_Amiloride(?d) ^ hasInteractionWith(?d,?d)
```

Listing C.27: VLKF_Amiloride

```
1hasExam(?p, ?e) ^ swrlb:lessThan(?v, 25) ^ hasExamValue(?e,?v)
2^ hasPatientAgeValue(?p, ?a) ^ Patient(?p) ^ Prescription(?pr)
3^ hasPrescription(?p, ?pr) ^ hasDrug(?pr, ?d)
4^ swrlb:greaterThan(?a, 64) ^ Apixaban(?d)
5^ Creatinine_Clearancee(?e) -> VLKF_Apixaban(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.28: VLKF_Apixaban

```
1hasExam(?p, ?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)

2^ swrlb:lessThan(?v, 50) ^ Patient(?p) ^ Prescription(?pr)
3^ hasPrescription(?p, ?pr) ^ hasDrug(?pr, ?d)
4^ swrlb:greaterThan(?a, 64) ^ Cimetidine(?d)
5^ Creatinine_Clearancee(?e) -> VLKF_Cimetidine(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.29: VLKF_Cimetidine

```
1hasExam(?p, ?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)

2^ Ciprofloxacin(?d) ^ Patient(?p) ^ Prescription(?pr)
3^ hasPrescription(?p, ?pr) ^ hasDrug(?pr, ?d)
4^ swrlb:greaterThan(?a, 64) ^ swrlb:lessThan(?v, 30)
5^ Creatinine_Clearancee(?e) -> VLKF_Ciprofloxacin(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.30: VLKF_Ciprofloxacin

```
1hasExam(?p, ?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)

2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
4^ swrlb:lessThan(?v, 30) ^ Colchicine(?d)
5^ Creatinine_Clearancee(?e) -> VLKF_Colchicine(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.31: VLKF_Colchicine

```
1hasExam(?p, ?e) ^ Dofetilide(?d)^ swrlb:lessThan(?v, 60)
2^ hasExamValue(?e, ?v) ^ hasPatientAgeValue(?p, ?a)
3^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
4^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
5^ Creatinine_Clearancee(?e) -> VLKF_Dofetilide(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.32: VLKF_Dofetilide

```
1hasExam(?p, ?e) ^ hasExamValue(?e,?v) ^ DULoxetine(?d)
2^ hasPatientAgeValue(?p, ?a) ^ Patient(?p)
3^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
4^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
5^ swrlb:lessThan(?v, 30) ^ Creatinine_Clearancee(?e)
6-> VLKF_Duloxetine(?d) ^ hasInteractionWith(?d, ?d)
```

Listing C.33: VLKF_Duloxetine

```
1hasExam(?p, ?e) ^ swrlb:lessThan(?v, 51) ^swrlb:greaterThan(?v
    ,14)
2^ hasExamValue(?e, ?v) ^ hasPatientAgeValue(?p, ?a) ^ Patient(?p)

3^ Prescription(?pr) ^ hasPrescription(?p, ?pr) ^ hasDrug(?pr, ?d)

4^ swrlb:greaterThan(?a, 64) ^ Edoxaban_Tosylate(?d)
5^ Creatinine_Clearancee(?e) -> VLKF_Edoxaban(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.34: VLKF_Edoxaban 15-50

```
1hasExam(?p, ?e) ^ swrlb:lessThan(?v,15) ^ hasExamValue(?e,?v)
2^ hasPatientAgeValue(?p, ?a) ^ Patient(?p) ^ Prescription(?pr)
3^ hasPrescription(?p, ?pr) ^ hasDrug(?pr, ?d)
4^ swrlb:greaterThan(?a, 64) ^ Edoxaban_Tosylate(?d)
5^ Creatinine_Clearancee(?e) -> VLKF_Edoxaban(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.35: VLKF_Edoxaban <15

```
1hasExam(?p, ?e) ^ swrlb:greaterThan(?v,95) ^ hasExamValue(?e,?v)
2^ hasPatientAgeValue(?p, ?a) ^ Patient(?p) ^ Prescription(?pr)
3^ hasPrescription(?p, ?pr) ^ hasDrug(?pr, ?d)
4^ swrlb:greaterThan(?a, 64) ^ Edoxaban_Tosylate(?d)
5^ Creatinine_Clearancee(?e) -> VLKF_Edoxaban(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.36: VLKF_Edoxaban >95

```
1hasExam(?p, ?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)

2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ Enoxaparin(?d) ^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
4^ swrlb:lessThan(?v, 30) ^ Creatinine_Clearancee(?e) ->
5VLKF_Enoxaparin(?d) ^ hasInteractionWith(?d, ?d)
```

Listing C.37: VLKF_Enoxaparin

```
1hasExam(?p, ?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)

2^ swrlb:lessThan(?v, 50) ^ Patient(?p) ^ Prescription(?pr)
3^ hasPrescription(?p, ?pr) ^ Famotidine(?d) ^ hasDrug(?pr, ?d)
4^ swrlb:greaterThan(?a, 64) ^ Creatinine_Clearancee(?e) ->
5VLKF_Famotidine(?d) ^ hasInteractionWith(?d, ?d)
```

Listing C.38: VLKF_Famotidine

```
1hasExam(?p, ?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)

2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
4^ swrlb:lessThan(?v, 30) ^ Fondaparinux(?d)
```

```
5^ Creatinine_Clearancee(?e) -> VLKF_Fondaparinux(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.39: VLKF_Fondaparinux

```
1hasExam(?p,?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)
2^ Gabapentin(?d) ^ Patient(?p) ^ Prescription(?pr)
3^ hasPrescription(?p, ?pr) ^ hasDrug(?pr, ?d)
4^ swrlb:greaterThan(?a, 64) ^ swrlb:lessThan(?v, 30)
5^ Creatinine_Clearancee(?e) -> VLKF_Gabapentin(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.40: VLKF_Gabapentin

```
1hasExam(?p,?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)
2^ swrlb:lessThan(?v, 81) ^ Patient(?p) ^ Prescription(?pr)
3^ hasPrescription(?p, ?pr) ^ hasDrug(?pr, ?d)
4^ swrlb:greaterThan(?a, 64) ^ levETIRAcetam(?d)
5^ Creatinine_Clearancee(?e) -> VLKF_Levetiracetam(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.41: VLKF_Levetiracetam

```
1hasExam(?p,?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)
2^ swrlb:lessThan(?v, 50) ^ Patient(?p) ^ Prescription(?pr)
3^ hasPrescription(?p, ?pr) ^ hasDrug(?pr, ?d)
4^ swrlb:greaterThan(?a, 64) ^ Creatinine_Clearancee(?e)
5^ Nizatidine(?d) -> VLKF_Nizatidine(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.42: VLKF_Nizatidine

```
1hasExam(?p,?e) ^ swrlb:lessThan(?v,60) ^ Pregabalin(?d)
2^ hasExamValue(?e, ?v) ^ hasPatientAgeValue(?p, ?a)
3^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
4^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
5^ Creatinine_Clearancee(?e) -> VLKF_Pregabalin(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.43: VLKF_Pregabalin

```
1hasExam(?p,?e) ^ Probenecid(?d) ^ hasExamValue(?e,?v)
2^ hasPatientAgeValue(?p, ?a) ^ Patient(?p)
3^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
4^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
```

```
5^ swrlb:lessThan(?v, 30) ^ Creatinine_Clearancee(?e) ->
6VLKF_Probenecid(?d) ^ hasInteractionWith(?d, ?d)
```

Listing C.44: VLKF_Probenecid

```
1hasExam(?p,?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)
2^ raNITIdine(?d) ^ swrlb:lessThan(?v, 50) ^ Patient(?p)
3^ Prescription(?pr) ^ hasPrescription(?p, ?pr) ^ hasDrug(?pr, ?d)

4^ swrlb:greaterThan(?a, 64) ^ Creatinine_Clearancee(?e)
5-> VLKF_Ranitidine(?d) ^ hasInteractionWith(?d, ?d)
```

Listing C.45: VLKF_Ranitidine

```
1hasExam(?p,?e) ^ Rivaroxaban(?d) ^ hasExamValue(?e,?v)
2^ hasPatientAgeValue(?p, ?a) ^ swrlb:lessThan(?v, 50)
3^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
4^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
5^ Creatinine_Clearancee(?e) -> VLKF_Rivaroxaban(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.46: VLKF_Rivaroxaban

```
1Spironolactone(?d) ^ hasExam(?p,?e) ^ hasExamValue(?e,?v)
2^ hasPatientAgeValue(?p, ?a) ^ Patient(?p)
3^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
4^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
5^ swrlb:lessThan(?v, 30) ^ Creatinine_Clearancee(?e)
6-> VLKF_Spironolactone(?d) ^ hasInteractionWith(?d, ?d)
```

Listing C.47: VLKF_Spironolactone

```
1hasExam(?p,?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)
2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
4^ swrlb:lessThan(?v, 30) ^ Creatinine_Clearancee(?e)
5^ traMADol(?d) -> VLKF_Tramadol(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.48: VLKF_Tramadol

```
1hasExam(?p,?e) ^ hasExamValue(?e,?v) ^ Triamterene(?d)
2^ hasPatientAgeValue(?p, ?a) ^ Patient(?p)
3^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
4^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
```

```
5^ swrlb:lessThan(?v, 30) ^ Creatinine_Clearancee(?e)
6-> VLKF_Triamterene(?d) ^ hasInteractionWith(?d, ?d)
```

Listing C.49: VLKF_Triamterene

```
1hasExam(?p,?e) ^ Trimethoprim(?d) ^ hasExamValue(?e,?v)
2^ hasDrug(?pr, ?d) ^ hasPatientAgeValue(?p, ?a)
3^ Patient(?p) ^ Prescription(?pr)
4^ swrlb:greaterThan(?a, 64) ^ swrlb:lessThan(?v, 30)
5^ hasPrescription(?p, ?pr) ^ Creatinine_Clearancee(?e)
6-> VLKF_Trimethoprim_sulfamethoxazole(?d)
7^ hasInteractionWith(?d, ?d)
```

Listing C.50: VLKF_Trimethoprim

C.4 Potentially Inappropriate Medications (PIMs) rules

```
1hasExam(?p, ?e) ^ hasExamValue(?e,?v) ^ hasPatientAgeValue(?p,?a)

2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
4^ swrlb:lessThan(?v, 30) ^ Creatinine_Clearancee(?e)
5^ Nitrofurantoin(?d) -> PIM_Anti-infective(?d)
6^ hasInteractionWith(?d, ?d)
```

Listing C.51: PIM_Anti-infective - Creatinine_Clearancee

```
1hasLenghtDrugTherapie(?p,?v) ^ hasPatientAgeValue(?p,?a)
2^ Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
3^ hasDrug(?pr, ?d) ^ swrlb:greaterThan(?a, 64)
4^ swrlb:greaterThan(?v, 30) ^ Nitrofurantoin(?d)
5-> PIM_Anti-infective(?d) ^ hasInteractionWith(?d, ?d)
```

Listing C.52: PIM_Anti-infective- LenghtDrugTherapie

C.5 Drugs To Be Used With Caution (UWC) rules

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Angiotensin-Converting_Enzyme_Inhibitors(?d)
3^ Trimethoprim_sulfamethoxazole(?d2) ^ hasDrug(?pr, ?d)
```

```
4^ hasDrug(?pr, ?d2) ^ hasPatientAgeValue(?p, ?a)
5^ swrlb:greaterThan(?a, 64) ->
6UWC_Trimethoprim_sulfamethoxazole(?d)
7^ UWC_Trimethoprim_sulfamethoxazole(?d2)
8^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.53: UWC_Trimethoprim_sulfamethoxazole - Angiotensin-Converting_Enzyme_-Inhibitors

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ Angiotensin_II_Receptor_Antagonists(?d)
3^ Trimethoprim_sulfamethoxazole(?d2) ^ hasDrug(?pr, ?d)
4^ hasDrug(?pr, ?d2) ^ hasPatientAgeValue(?p, ?a)
5^ swrlb:greaterThan(?a, 64) ->
6UWC_Trimethoprim_sulfamethoxazole(?d)
7^ UWC_Trimethoprim_sulfamethoxazole(?d2)
8^ hasInteractionWith(?d, ?d2) ^ hasInteractionWith(?d2, ?d)
```

Listing C.54: UWC_Trimethoprim_sulfamethoxazole - Angiotensin_II_Receptor_-Antagonists

APPENDIX D

SEMANTIC WEB RULE LANGUAGE (SWRL) RULES TO FIND ALTERNATIVE DRUGS

This appendix provides the SWRL rules developed on the Beers Criteria ontology to find alternative drugs for prescribed drugs.

D.1 Alternative drugs rules to drugs included in the Potentially Harmful Drug-Disease Interactions

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ hasDrug(?pr, ?d) ^ First_Generation_Antihistamines(?d)
3^ Drugs(?d) ^ Drugs(?snp) ^ isAlternative(?snp, true) ^
     Alt_First_generation_antihistamines(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ swrlb:greaterThan(?a, 64)
5^ Dementia(?pd) ^ hasDisease(?p, ?pd)
6-> hasAlternative(?d, ?snp)
```

Listing D.1: Alt_DDI_Dementia_Anticholinergic_First-generation_antihistamines

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ hasDrug(?pr, ?d) ^ First_Generation_Antihistamines_Oral(?d)
3^ Drugs(?d) ^ Drugs(?snp) ^ isAlternative(?snp, true) ^
     Alt_First_generation_antihistamines(?snp)
```

```
4^ hasPatientAgeValue(?d, ?a) ^ swrlb:greaterThan(?a, 64)
5^ Dementia(?pd) ^ hasDisease(?p, ?pd)
6-> hasAlternative(?d, ?snp)
```

Listing D.2: Alt_DDI_Dementia_Anticholinergic_First-generation_antihistamines_Oral

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ hasDrug(?pr, ?d) ^ Benztropine(?d) ^ Drugs(?d)
3^ hasRoute(?d, ?o) ^ Oral(?o) ^ Drugs(?snp)
4^ Alt_Parkinson_disease(?snp) ^ hasPatientAgeValue(?d, ?a)
5^ isAlternative(?snp, true) ^ Parkinson(?pd)
6^ hasDisease(?p, ?pd) ^ swrlb:greaterThan(?a, 64)
7^ Dementia(?pd) -> hasAlternative(?d, ?snp)
```

Listing D.3: Alt_DDI_Dementia_Anticholinergic_Parkinson_Benztropine

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ hasDrug(?pr, ?d) ^ Trihexyphenidyl(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Parkinson_disease(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ Parkinson(?pd) ^ hasDisease(?p, ?pd)
6^ swrlb:greaterThan(?a, 64) ^ Dementia(?pd)
7-> hasAlternative(?d, ?snp)
```

Listing D.4: Alt_DDI_Dementia_Anticholinergic_Parkinson_Trihexyphenidyl

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ hasDrug(?pr, ?d) ^ Antipsychotics(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Antipsychotics_Behavioral(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) ^ Dementia(?pd)
6^ hasDisease(?p, ?pd) -> hasAlternative(?d, ?snp)
```

Listing D.5: Alt_DDI_Dementia_Antipsychotics_Behavioral

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ hasDrug(?pr, ?d) ^ Histamine_H2-Antagonists(?d)
3^ Drugs(?d) ^ Drugs(?snp) ^ Alt_H2_blockers(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) ^ Dementia(?pd)
6^ hasDisease(?p, ?pd) -> hasAlternative(?d, ?snp)
```

Listing D.6: Alt_DDI_Dementia_H2Blocker

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
```

```
2^ hasDrug(?pr, ?d)
3^ Non-aspirin_nonsteroidal_anti-inflammatory_drugs_NA-NSAIDs(?d)
4^ Drugs(?d) ^ Drugs(?snp)
5^ Alt_Non-aspirin_nonsteroidal_anti-inflammatory_drugs_NA-NSAIDs
     (?snp)
6^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
7^ swrlb:greaterThan(?a, 64) ^ Dementia(?pd)
8^ hasDisease(?p, ?pd) -> hasAlternative(?d, ?snp)
```

Listing D.7: Alt_DDI_Dementia_NA-NSAIDs

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Eszopiclone(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Benzodiazepines(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) ^ Dementia(?pd)
6^ hasDisease(?p, ?pd) -> hasAlternative(?d, ?snp)
```

Listing D.8: Alt_DDI_Dementia_Nonbenzodiazepine_Eszopiclone

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d)
3^ Tricyclics_and_Other_Norepinephrine-_reuptake_Inhibitors(?d)
4^ Drugs(?d) ^ Drugs(?snp) ^ isAlternative(?snp, true) ^
     Alt_Tertiary_amine_tricyclic_antidepressants(?snp)
5^ hasPatientAgeValue(?d, ?a) ^ swrlb:greaterThan(?a, 64)
6^ Dementia(?pd) ^ hasDisease(?p, ?pd)
7-> hasAlternative(?d, ?snp)
```

Listing D.9: Alt_DDI_Dementia_Tricyclic antidepressants - secondary

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Tertiary_amine_tricyclic_antidepressants(?d)

3^ Drugs(?d) ^ Drugs(?snp) ^ isAlternative(?snp, true)
4^ Alt_Tertiary_amine_tricyclic_antidepressants(?snp)
5^ hasPatientAgeValue(?d, ?a) ^ swrlb:greaterThan(?a, 64)
6^ Dementia(?pd) ^ hasDisease(?p, ?pd) -> hasAlternative(?d, ?snp)
```

Listing D.10: Alt_DDI_Dementia_Tricyclic antidepressants - tertiary

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Anticonvulsants(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Anticonvulsants(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) ^ History_of_falls(?pd)
```

```
6^ hasDisease(?p, ?pd) -> hasAlternative(?d, ?snp)
```

Listing D.11: Alt_DDI_Falls_Anticonvulsants

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Antipsychotics(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Antipsychotics_Behavioral(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) ^ History_of_falls(?pd)
6^ hasDisease(?p, ?pd) -> hasAlternative(?d, ?snp)
```

Listing D.12: Alt_DDI_Falls_Antipsychotics_Behavioral

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Antipsychotics(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Antipsychotics_Delirium(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) ^ History_of_falls(?pd)
6^ hasDisease(?p, ?pd) -> hasAlternative(?d, ?snp)
```

Listing D.13: Alt_DDI_Falls_Antipsychotics_Delirium

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Antipsychotics(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Antipsychotics_Schizophrenia(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) ^ History_of_falls(?pd)
6^ hasDisease(?p, ?pd) -> hasAlternative(?d, ?snp)
```

Listing D.14: Alt_DDI_Falls_Antipsychotics_Schizophrenia

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Benzodiazepines(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Benzodiazepines(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) ^ History_of_falls(?pd)
6^ hasDisease(?p, ?pd) -> hasAlternative(?d, ?snp)
```

Listing D.15: Alt_DDI_Falls_Benzodiazepines

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Zaleplon(?d) ^ Drugs(?d) ^ Drugs(?snp)
3^ Alt_Benzodiazepines(?snp) ^ hasPatientAgeValue(?d, ?a)
4^ isAlternative(?snp, true) ^ swrlb:greaterThan(?a, 64)
5^ History_of_falls(?pd) ^ hasDisease(?p, ?pd)
```

```
6-> hasAlternative(?d, ?snp)
```

Listing D.16: Alt_DDI_Falls_Benzodiazepines_Zaleplon

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Zolpidem(?d) ^ Drugs(?d) ^ Drugs(?snp)
3^ Alt_Benzodiazepines(?snp) ^ hasPatientAgeValue(?d, ?a)
4^ isAlternative(?snp, true) ^ swrlb:greaterThan(?a, 64)
5^ History_of_falls(?pd) ^ hasDisease(?p, ?pd)
6-> hasAlternative(?d, ?snp)
```

Listing D.17: Alt_DDI_Falls_Benzodiazepines_Zolpidem

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Eszopiclone(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Benzodiazepines(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) ^ History_of_falls(?pd)
6^ hasDisease(?p, ?pd) -> hasAlternative(?d, ?snp)
```

Listing D.18: Alt_DDI_Falls_Nonbenzodiazepine_Eszopiclone

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d)
3^ Tricyclics_and_Other_Norepinephrine-_reuptake_Inhibitors(?d)
4^ Drugs(?d) ^ Drugs(?snp) ^ isAlternative(?snp, true) ^
    Alt_Tertiary_amine_tricyclic_antidepressants(?snp)
5^ hasPatientAgeValue(?d, ?a) ^ swrlb:greaterThan(?a, 64)
6^ History_of_falls(?pd) ^ hasDisease(?p, ?pd)
7-> hasAlternative(?d, ?snp)
```

Listing D.19: Alt_DDI_Falls_Tricyclic antidepressants - secondary

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Tertiary_amine_tricyclic_antidepressants(?d)

3^ Drugs(?d) ^ Drugs(?snp) ^ isAlternative(?snp, true)
4^ Alt_Tertiary_amine_tricyclic_antidepressants(?snp)
5^ hasPatientAgeValue(?d, ?a) ^ swrlb:greaterThan(?a, 64)
6^ History_of_falls(?pd) ^ hasDisease(?p, ?pd)
7-> hasAlternative(?d, ?snp)
```

Listing D.20: Alt_DDI_Falls_Tricyclic antidepressants - tertiary

D.2 Alternative drugs rules to drugs Included in the High-Risk Medications

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p,?pr)
2^ hasDrug(?pr, ?d) ^ First_Generation_Antihistamines(?d)
3^ Drugs(?d) ^ Drugs(?snp) ^ isAlternative(?snp, true)
4^ Alt_First_generation_antihistamines(?snp)
5^ hasPatientAgeValue(?d, ?a) ^ swrlb:greaterThan(?a, 64)
6-> hasAlternative(?d, ?snp)
```

Listing D.21: Alt_HRM_Anticholinergic_First-generation_antihistamines

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ First_Generation_Antihistamines_Oral(?d)
3^ Drugs(?d) ^ Drugs(?snp) ^ isAlternative(?snp, true)
4^ Alt_First_generation_antihistamines(?snp)
5^ hasPatientAgeValue(?d, ?a) ^ swrlb:greaterThan(?a, 64)
6-> hasAlternative(?d, ?snp)
```

Listing D.22: Alt_HRM_Anticholinergic_First-generation_antihistamines_Oral

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Benztropine(?d) ^ Drugs(?d)
3^ hasRoute(?d, ?o) ^ Oral(?o) ^ Drugs(?snp)
4^ Alt_Parkinson_disease(?snp) ^ hasPatientAgeValue(?d, ?a)
5^ isAlternative(?snp, true) ^ Parkinson(?pd)
6^ hasDisease(?p, ?pd) ^ swrlb:greaterThan(?a, 64)
7-> hasAlternative(?d, ?snp)
```

Listing D.23: Alt_HRM_Anticholinergic_Parkinson _disease_Benztropine

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Trihexyphenidyl(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Parkinson_disease(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ Parkinson(?pd) ^ hasDisease(?p, ?pd)
6^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.24: Alt_HRM_Anticholinergic_Parkinson _disease_Trihexyphenidyl

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Dipyridamole(?d) ^ Drugs(?d)
3^ hasRoute(?d, ?o) ^ Oral(?o) ^ Drugs(?snp)
4^ Alt_Antithrombotic_Anti_platelets(?snp)
5^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
```

```
6^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.25: Alt_HRM_Antithrombotic/Anti platelets_Dipyridamole

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Ticlopidine(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Antithrombotic_Anti_platelets(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.26: Alt_HRM_Antithrombotic/Anti platelets_Ticlopidine

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Barbiturates(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Barbiturates(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.27: Alt_HRM_CNS_Barbiturates

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Meprobamate(?d) ^ Drugs(?d) ^ Drugs(?snp)
3^ Alt_Meprobamate(?snp) ^ hasPatientAgeValue(?d, ?a)
4^ isAlternative(?snp, true) ^ swrlb:greaterThan(?a, 64)
5-> hasAlternative(?d, ?snp)
```

Listing D.28: Alt_HRM_CNS_Other_Meprobamate

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Thioridazine(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Thioridazine(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.29: Alt_HRM_CNS_Other_Thioridazine

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Tertiary_amine_tricyclic_antidepressants(?d)

3^ Drugs(?d) ^ Drugs(?snp) ^ isAlternative(?snp, true) ^
    Alt_Tertiary_amine_tricyclic_antidepressants(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ swrlb:greaterThan(?a, 64)
5-> hasAlternative(?d, ?snp)
```

Listing D.30: Alt_HRM_CNS_Tertiary_TCAs

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Ergoloid_Mesylates(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Vasodilator(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.31: Alt_HRM_CNS_Vasodilator_Ergot mesylates

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Isoxsuprine(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Vasodilator(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.32: Alt_HRM_CNS_Vasodilator_Isoxsuprine

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Central_alpha-Agonists(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Alpha_agonists_central(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.33: Alt_HRM_Cardio_Alpha agonists

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Disopyramide(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Cardio_Other(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.34: Alt_HRM_Cardio_Other_Disopyramide

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ NIFEdipine(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Cardio_Other(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.35: Alt_HRM_Cardio_Other_NIFEdipine

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Desiccated_thyroid(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Desiccated_thyroid(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.36: Alt_HRM_Endocrine_Desiccated thyroid

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Estrogens(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Estrogens(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.37: Alt_HRM_Endocrine_Estrogens

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ chlorproMAZINE(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Sulfonylureas(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.38: Alt_HRM_Endocrine_Sulfonylureas_chlorproMAZINE

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ glyBURIDE(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Sulfonylureas(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.39: Alt_HRM_Endocrine_Sulfonylureas_glyBURIDE

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Meperidine(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Opioids(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.40: Alt_HRM_Pain_Opioids_Meperidine

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Meperidine(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Opioids(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
```

```
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.41: Alt_HRM_Pain_Opioids_Pentazocine

```
1Patient(?p) ^ Prescription(?pr)
2^ hasPrescription(?p, ?pr) ^ hasDrug(?pr, ?d)
3^ Centrally_Acting_Skeletal_Muscle_Relaxants(?d)
4^ Drugs(?d) ^ Drugs(?snp)
5^ Alt_Skeletal_muscle_relaxants(?snp)
6^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
7^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.42: Alt_HRM_Pain_Skeletal muscle relaxants

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Orphenadrine(?d) ^ Drugs(?d)
3^ Drugs(?snp) ^ Alt_Skeletal_muscle_relaxants(?snp)
4^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
5^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.43: Alt_HRM_Pain_Skeletal muscle relaxants_Orphenadrine

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Indomethacin(?d) ^ Drugs(?d) ^ Drugs(?snp)
    ^ Alt_Specific_nonsteroidal_antiinflammatory_drugs(?snp)
3^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
4^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.44: Alt_HRM_Pain_Specific nonsteroidal antiinflammatory drugs_Indomethacin

```
1Patient(?p) ^ Prescription(?pr) ^ hasPrescription(?p, ?pr)
2^ hasDrug(?pr, ?d) ^ Ketorolac(?d) ^ Drugs(?d) ^ Drugs(?snp) ^
    Alt_Specific_nonsteroidal_antiinflammatory_drugs(?snp)
3^ hasPatientAgeValue(?d, ?a) ^ isAlternative(?snp, true)
4^ swrlb:greaterThan(?a, 64) -> hasAlternative(?d, ?snp)
```

Listing D.45: Alt_HRM_Pain_Specific nonsteroidal antiinflammatory drugs_Ketorolac

APPENDIX E

SPARQL INTERACTIONS QUERYING

This appendix provides the SPARQL queries performed on the Beers Criteria ontology to select prescribed inappropriate drugs, alternative drugs, and TMAX value.

E.1 SPARQL interactions querying

```
1     SELECT DISTINCT ?Drug ?Interaction1 ?Interaction2
2     WHERE OntoBC:P1 OntoBC:hasDrug ?Drug .
3        ?Drug rdf:type ?Interaction1 .
4        ?Interaction1 rdfs:subClassOf ?Interaction2 .
5        ?Interaction2 rdfs:subClassOf OntoBC:BeersCriteria .
```

Listing E.1: SPARQL group interaction query

```
1    SELECT DISTINCT ?Drug ?Interaction1 ?Interaction2 ?
         Interaction3
2    WHERE OntoBC:P1 OntoBC:hasDrug ?Drug .
3       ?Drug rdf:type ?Interaction1 .
4       ?Interaction1 rdfs:subClassOf ?Interaction2 .
5       ?Interaction2 rdfs:subClassOf ?Interaction3 .
6       ?Interaction3 rdfs:subClassOf OntoBC:BeersCriteria .
```

Listing E.2: SPARQL group and subgroup interaction query

```
1    SELECT DISTINCT ?Drug ?DrugInt
2    WHERE OntoBC:P1 OntoBC:hasDrug ?Drug .
3          ?Drug {ontology}:hasInteractionWith ?DrugInt
```

Listing E.3: SPARQL drug-drug interaction query

E.2 SPARQL alternative querying

```
1SELECT DISTINCT ?Drug ?Alternative
2WHERE OntoBC:P1 OntoBC:hasDrug ?Drug .
3    ?Drug OntoBC:hasAlternative ?Alternative
```

Listing E.4: SPARQL alternative drug query

E.3 SPARQL drug parameters querying

```
1SELECT DISTINCT ?value
2WHERE {{{ontology}:{labeldrug} rdfs:subClassOf ?object.
3    ?object a owl:Restriction .
4    ?object owl:onProperty {ontology}:hasTmax.
5    object owl:qualifiedCardinality ?value }}
```

Listing E.5: SPARQL Tmax query

E.4 SPARQL all PIM parameters querying

```
1 prefix = "http://gr60.st-andrews.ac.uk/BeersCriteria#"
2 graph.bind("pre", prefix)
3 result = list(graph.query(
4  SELECT *
5  WHERE { {
6    SELECT ?Prescription ?Drug ?Interaction ?InteractionGroup
7             ?QualityofEvidence ?StrengthofRecommendation
8    WHERE {
9      pre:Patient1 pre:hasPrescription ?Prescription .
10     ?Prescription pre:hasDrug ?Drug .
11     ?Drug rdf:type ?Interaction .
12     ?Drug rdf:type ?InteractionProp .
13     ?Interaction rdfs:subClassOf ?InteractionGroup .
14     ?InteractionGroup rdfs:subClassOf pre:BeersCriteria .
15     ?InteractionProp rdfs:subClassOf ?QoF .
16     ?InteractionProp rdfs:subClassOf ?SoR .
```

```
17      ?QoF a owl:Restriction .
18      ?QoF owl:onProperty pre:hasQualityofEvidence.
19      ?QoF owl:someValuesFrom ?QualityofEvidence .
20      ?SoR a owl:Restriction .
21      ?SoR owl:onProperty pre:hasStrengthofRecommendation.
22      ?SoR owl:someValuesFrom ?StrengthofRecommendation . }
23     }
24     UNION {
25       SELECT ?Prescription ?Drug ?Interaction ?InteractionGroup
26              ?QualityofEvidence ?StrengthofRecommendation
27       WHERE {pre:Patient1 pre:hasPrescription ?Prescription .
28         ?Prescription pre:hasDrug ?Drug .
29         ?Drug rdf:type ?InteractionSubGroup .
30         ?Drug rdf:type ?InteractionProp .
31         ?InteractionSubGroup rdfs:subClassOf ?Interaction .
32         ?Interaction rdfs:subClassOf ?InteractionGroup .
33         ?InteractionGroup rdfs:subClassOf pre:BeersCriteria .
34         ?InteractionProp rdfs:subClassOf ?QoF .
35         ?InteractionProp rdfs:subClassOf ?SoR .
36         ?QoF a owl:Restriction .
37         ?QoF owl:onProperty pre:hasQualityofEvidence.
38         ?QoF owl:someValuesFrom ?QualityofEvidence .
39         ?SoR a owl:Restriction .
40         ?SoR owl:onProperty pre:hasStrengthofRecommendation.
41         ?SoR owl:someValuesFrom ?StrengthofRecommendation . }
42       }
43     }
44))
```

Listing E.6: SPARQL all PIM parameters query

Appendix F

Python Code

This appendix provides the inference engines developed in Python that are integrated with the Beers Criteria ontology.

F.1 Main file

The main file integrates the inference engines and the CDSS database. Moreover, it defines the CDSS workflow.

Filename: main.py

```
from ie_beers_criteria_interactions import checkInteraction
from ie_smt_alternative_solver import checkAlternative
from ie_smt_rescheduling_solver import checkRescheduling
from Utils.dbInsert import importPatients
from Utils.dbCreate import resetDB
from result_queries import results
import sqlite3, os
import pandas as pd
from config import DBNAME
import time


importPatients(True) # Insert patient data into the DB – Folder: /data/patient_data

conn = sqlite3.connect(DBNAME, timeout=10)

def getPatients(conn): # Get patients from the CDSS DB
    df = pd.read_sql_query("""
                                SELECT distinct (A.CD_PATIENT)
                                FROM PATIENT A""", conn)
    patients = df.values.tolist()
    return(patients)

```

```
patients = getPatients(conn)

for patient in patients:
    os.system('cls' if os.name == 'nt' else 'clear')
    checkInteraction(patient[0]) #Check if there are inappropriate medications in each
        prescription
    os.system('cls' if os.name == 'nt' else 'clear')
    checkAlternative(patient[0]) #Check if SAT prescriptions can be found with
        alternative drugs
    os.system('cls' if os.name == 'nt' else 'clear')
    checkRescheduling(patient[0]) #Reschedule drug-drug interaction

results() #Folder: /results/CDSS
```

F.2 Inference engine Beers Criteria

The Beers Criteria inference engine integrates the patient data with the ontology and the reasoner Pellet. Moreover, it selects the interactions and alternative drugs from the reasoners and inserts them in the CDSS database.

Filename: ie_beers_criteria_interactions.py

```
import sys, io, glob
import time
from owlready2 import *
from dateutil import parser
from import_files import importFile, listOccurence, listPatients, listExams,
    listPreviousDisease, listPrescrDays
from datetime import datetime
import pandas as pd
import sqlite3
from config import DBNAME, BEERSVERSION
os.system('cls' if os.name == 'nt' else 'clear')

conn = sqlite3.connect(DBNAME, timeout=10)
c = conn.cursor()
default_world.set_backend(filename = fr'inference_engines/ontology_files/
    t.sqlite3', exclusive = False)

version = BEERSVERSION


ONTOLOGY = fr'inference_engines/ontology_files/BeersOntologyV{
    version}.owl'
ONTOLOGY_NAME = f"BeersOntologyV{version}."
ONTO = get_ontology(ONTOLOGY).load() #Import the Beers Criteria Ontology
DRUGS = 'Drugs'
ROUTE = 'AdministrationRoute'
DISEASE = 'Disease'
BEERS = 'BeersCriteria'
EXAMS = 'Exams'
PATIENT = 'Patient'
```

```
28 PROCESSED_ROWS = 0
29
30 DrugNotRegistered = set()
31 RouteNotRegistered = set()
32 DiseaseNotRegistered = set()
33 ExamNotRegistered = set()
34 PacientNotRegisted = set()
35 Exceptions = set()
36
37
38
39
40 #ONTOLOGY FUNCTIONS
41
42 def checkLabel(name, type): #check if exist the label drug on the onotlogy and return
       the class name
43     classObj = ONTO.search(label = (name))
44     if not classObj:
45         classObj = ONTO.search(label = (name.replace('-','_')), _case_sensitive =
               False)
46     if not classObj:
47         classObj = ONTO.search(label = (name.replace('_','-')), _case_sensitive =
               False)
48     if classObj:
49         classObj = str(classObj[0]).replace(ONTOLOGY_NAME,'')
50
51         if isinstance(ONTO[classObj], ONTO[type]):
52             return classObj
53     else:
54         return(None)
55
56 def checkIndividual(name): #check if exist the label drug on the onotlogy and return the
        class name
57     classObj = ONTO.search(iri = '*'+str(name), _case_sensitive = False)
58     if classObj:
59         return(True)
60     else:
61         return(False)
62
63
64 def addOntologyPatients(patientList): #Select patient data and send to the function
       addIndividualPatient
65         for index, row in patientList.iterrows():
66             name = row['CD_PATIENT']
67             age = int(row['AGE'])
68             gender = row['GENDER']
69             composeName = str(name)+"-"+str(age)+"-"+str(gender)
70             hospitalizationDate = row['DT_Admission']
71             dischargeDate = row['DT_discharge']
72             procedureName = row['MAIN_PROCEDURE']
73             duration = pd.to_datetime(row['DT_discharge']) - pd.to_datetime(
                   row['DT_Admission'])
74             lenghtTreatment = duration.days
75             #Insert parameters into the ontology
76             individual = ONTO.Patient(composeName)
77             individual.hasPatientAgeValue = age
```

```
78                  individual.hasLenghtTreatment = lenghtTreatment
79                  addOntologyDisease(procedureName, composeName)
80                  if gender == 'M':
81                      individual.hasGender.append(ONTO.iMale)
82                  else:
83                      individual.hasGender.append(ONTO.iFemale)
84
85 def addOntologyDisease(name, patient):
86      if checkIndividual(patient):
87          diseaseName = (str(patient)+"-"+str(name))
88          if checkIndividual(diseaseName):
89              ONTO[patient].hasDisease.append(ONTO[diseaseName])
90              return('Disease already registered')
91          else:
92              classObj = checkLabel(name, DISEASE)
93              if classObj:
94                  individualdisease = ONTO[classObj](diseaseName)
95                  ONTO[patient].hasDisease.append(individualdisease)
96                  return('Disease registered')
97              else:
98                  DiseaseNotRegistered.add(name)
99                  return(None)
100     else:
101         PacientNotRegisted.add(patient)
102         print(f'{PacientNotRegisted} PacientNotRegisted')
103         return(None)
104
105 def addOntologyRelease(name, patient):
106     if checkIndividual(patient):
107         diseaseName = (str(patient)+"-"+str(name))
108         if checkIndividual(diseaseName):
109             classObj = checkLabel(name, 'ReleaseDrug')
110             return(ONTO[classObj](diseaseName))
111         else:
112             classObj = checkLabel(name, 'ReleaseDrug')
113             individualRelease = ONTO[classObj](diseaseName)
114             return(individualRelease)
115
116 def addOntologyRoute(name, patient):
117     if checkIndividual(patient):
118         routeName = (str(patient)+"-"+str(name))
119         if checkIndividual(routeName):
120             classObj = checkLabel(name, 'AdministrationRoute')
121             return(ONTO[classObj](routeName))
122         else:
123             classObj = checkLabel(name, 'AdministrationRoute')
124             individualRelease = ONTO[classObj](routeName)
125             return(individualRelease)
126
127 def add_ontology_exam(examList, age, gender):
128     for index, row in examList.iterrows():
129         patient = row['CD_PATIENT']
130         seqResult = row['seq_Result']
131         exam = row['nm_Exam']
132         result = float(row['qt_Result'])
133         date = row['dt_Result']
```

```
134             composeName = str(patient)+"-"+str(age)+"-"+str(gender)
135             examName = str(composeName) +"-"+ exam+"-"+str(seqResult)
136
137             if checkIndividual(examName):
138                 ONTO[examName].hasExamValue.append(result)
139                 ONTO[composeName].hasExam.append(ONTO[examName])
140                 return('Exam already registered')
141             else:
142                 classObj = checkLabel(exam, EXAMS)
143                 if classObj:
144                     if checkIndividual(composeName):
145                         individualexam = ONTO[classObj](examName)
146                         ONTO[composeName].hasExam.append(individualexam)
147                         individualexam.hasExamValue.append(result)
148                     else:
149                         ExamNotRegistered.add(exam)
150
151
152## Prescription functions
153def addOntologyDrug(name, patient, prescription, age, route, dose, gender,
        drugNameOriginal, typeDrug, drugLenght,criticalPatient,firstLine,release ):
154     classDrug = checkLabel(name, DRUGS)
155     classRoute = checkLabel(route, ROUTE)
156     classRelease = checkLabel(release, 'ReleaseDrug')
157     global PROCESSED_ROWS
158     PROCESSED_ROWS += 1
159     if typeDrug == 'S' or typeDrug =='C':
160         if classDrug and classRoute:
161             individual = ONTO[classDrug](str((str(prescription)+"-"+str(name))).
                    replace('/', "-"))
162             individual.hasPatientAgeValue = age
163             individual.hasLenghtDrugTherapie = drugLenght
164             individual.isAlternative.append(False)
165             individual.hasGender.append(gender)
166             currentDose = individual.hasDailydoseValue
167             if criticalPatient == 'False' or criticalPatient == 'false' or
                    criticalPatient == 'F':
168                 individual.isCriticalPatient.append(False)
169             else:
170                 individual.isCriticalPatient.append(True)
171
172             if firstLine == 'False' or firstLine == 'false' or firstLine == 'F':
173                 individual.isFirstLine.append(False)
174             else:
175                 individual.isFirstLine.append(True)
176             individualRelease = addOntologyRelease(release,patient)
177             individual.toRelease.append(individualRelease)
178             if currentDose:
179                 dose = int(currentDose) + dose
180             individual.hasDailydoseValue = dose
181             individualRoute = addOntologyRoute(route, patient)
182             individual.hasRoute.append(individualRoute)
183             if checkIndividual(prescription):
184                 ONTO[patient].hasPrescription.append(ONTO[prescription])
185             else:
186                 individualPrescr = ONTO['Prescription'](str(prescription))
```

```
187                     ONTO[patient].hasPrescription.append(individualPrescr)
188                 ONTO[prescription].hasDrug.append(ONTO[(str((str(prescription)+"-"
                        +str(name))).replace('/',"-"))])
189                 return(individual)
190             else:
191                 if not classDrug:
192                     DrugNotRegistered.add(name)
193                     print(f'{DrugNotRegistered} DrugNotRegistered')
194                 if not classRoute:
195                     RouteNotRegistered.add(route)
196                     print(f'{RouteNotRegistered} RouteNotRegistered')
197                 return(None)
198         else:
199             individual = ONTO[name](str((str(prescription)+'_ALT_'+str(name))).
                    replace('/',"-"))
200             individual.hasPatientAgeValue = age
201             individual.hasLenghtDrugTherapie = drugLenght
202             individual.isAlternative.append(False)
203             currentDose = individual.hasDailydoseValue
204             if currentDose:
205                 dose = currentDose + dose
206             individual.hasDailydoseValue.append(dose)
207             individual.hasRoute.append(ONTO[classRoute])
208             if checkIndividual(prescription):
209                 ONTO[patient].hasPrescription.append(ONTO[prescription])
210             else:
211                 individualPrescr = ONTO['Prescription'](str(prescription))
212                 ONTO[patient].hasPrescription.append(individualPrescr)
213             ONTO[prescription].hasDrug.append(ONTO[(str(str(prescription)+'_ALT_'+
                    str(name)).replace('/',"-"))])
214
215
216 def add_ontology_drug_assertions(patient, day, age, gender): #Add an individual
        patient
217             if gender == 'M':
218                 Indvgender = ONTO.iMale
219             else:
220                 Indvgender = ONTO.iFemale
221
222             presc_day = str(patient)+"-"+str(age)+"-"+str(gender) +"-"+str(day).
                    replace("-", "").replace("/", "")
223             patientName = str(patient)+"-"+str(age)+"-"+str(gender)
224             if patient:
225                 drugs = ONTO[presc_day].hasDrug
226                 diseases = ONTO[patientName].hasDisease
227                 for drug in drugs:
228                     drug.hasGender.append(Indvgender)
229                     for disease in diseases:
230                         drug.hasTreatmentIndication.append(disease)
231             alternatives = ONTO.search(iri = '*alt_*', _case_sensitive = False,
                    isAlternative = True )
232             for alternative in alternatives:
233                 if isinstance(alternative, ONTO['Alternative_Drugs']):
234                     alternative.hasPatientAgeValue = age
235                     alternative.hasGender.append(Indvgender)
236
```

```
237 def add_prescription(prescription_df, age, gender): #Select patient data and send to the
        function
238         if gender == 'M':
239             Indvgender = ONTO.iMale
240         else:
241             Indvgender = ONTO.iFemale
242         prescrdata={}
243         header = {}
244         body = []
245         index = prescription_df.index.tolist()
246         counter = int(prescription_df["CD_PATIENT"][index[0]])
247         header = {"Counter": counter, "Gender": gender,"Age": age}
248         for i in index:
249             drugName = ''.join(str(prescription_df["DRUG"][i]).rstrip().lstrip()).
                    replace(" ", "_")
250             drugNameOriginal = ''.join(str(prescription_df["DS_DRUG_ORIGINAL"
                    ][i]).rstrip().lstrip()).replace(" ", "_")
251             composeName = str(counter)+"-"+str(age)+"-"+str(gender)
252             route = ''.join(str(prescription_df["ROUTE"][i]).rstrip().lstrip()).replace(
                    " ", "_")
253             startDateInt = str(prescription_df['START_DATE'][i]).replace("/", "").
                    replace("-", "")
254             startDate = str(prescription_df['START_DATE'][i])
255             endDate = str(prescription_df['END_DATE'][i])
256             schedule = str(prescription_df['SCHEDULE'][i])
257             frequency = str(prescription_df['FREQUENCY'][i])
258             typeDrug = str(prescription_df['TYPE_DRUG'][i])
259             criticalPatient = str(prescription_df['CRITICAL_PATIENT'][i])
260             firstLine = str(prescription_df['FIRST_LINE'][i])
261             release = str(prescription_df['RELEASE'][i])
262             drugLenght = int(prescription_df['Drug_Lenght'][i])
263             presc_day = str(counter)+"-"+str(age)+"-"+str(gender) +"-"+str(
                    startDateInt)
264             if typeDrug == 'S':
265                 dose = int(prescription_df['DOSE'][i])
266                 addOntologyDrug(drugName, composeName, presc_day, age,
                        route, dose, Indvgender, drugNameOriginal, typeDrug,
                        drugLenght,criticalPatient,firstLine,release )
267                 body.append({'DrugName': drugName , 'DrugNameOriginal':
                        drugNameOriginal, 'Route' :route, 'Dose' :dose, '
                        TypeDrug':typeDrug, 'StartDate':startDate, 'EndDate':
                        endDate, 'Schedule':schedule, 'Frequency':frequency, '
                        CriticalPatient':criticalPatient , 'FirstLine':firstLine ,
                        'Release':release })
268             else:
269                 drugName = ''.join(str(prescription_df["DRUG"][i]).rstrip().lstrip()).
                        replace(" ", "-")
270                 dose = int(prescription_df['DOSE'][i])
271                 addOntologyDrug(drugName, composeName, presc_day, age,
                        route, dose, Indvgender, drugNameOriginal, typeDrug,
                        drugLenght,criticalPatient,firstLine,release )
272                 body.append({'DrugName': drugName , 'DrugNameOriginal':
                        drugNameOriginal, 'Route' :route, 'Dose' :dose, '
                        TypeDrug':typeDrug, 'StartDate':startDate, 'EndDate':
                        endDate, 'Schedule':schedule, 'Frequency':frequency, '
                        CriticalPatient':criticalPatient , 'FirstLine':firstLine ,
```

```
                    'Release':release })
273                 try:
274                     for x in range(3):
275                         if len(str(prescription_df['DS_COM_DRUG'+str(x)][i])) > 3:
276                             drugName = ''.join(str(prescription_df['
                                    DS_COM_DRUG'+str(x)][i]).rstrip().lstrip()).
                                    replace(" ", "-")
277                             dose = int(prescription_df['DOSE_COM'+str(x)][i])
278                             addOntologyDrug(drugName, composeName,
                                    presc_day, age, route, dose, Indvgender,
                                    drugNameOriginal, typeDrug, drugLenght,
                                    criticalPatient,firstLine,release )
279                             body.append({'DrugName': drugName , '
                                    DrugNameOriginal':drugNameOriginal, '
                                    Route' :route, 'Dose' :dose, 'TypeDrug':
                                    typeDrug, 'StartDate':startDate, 'EndDate'
                                    :endDate, 'Schedule':schedule, 'Frequency'
                                    :frequency, 'CriticalPatient':
                                    criticalPatient , 'FirstLine':firstLine , '
                                    Release':release })
280                 except Exception as e: Exceptions.add('')
281         prescrdata = {'Header': header, 'Drugs': body}
282         return (prescrdata)
283
284 def add_disease(diseaseList, age, gender): #Select patient data and send to the function
        addIndividualPatient
285         for index, row in diseaseList.iterrows():
286             patient = row['CD_PATIENT']
287             disease = row['nm_Disease']
288             composeName = str(patient)+"-"+str(age)+"-"+str(gender)
289             addOntologyDisease(disease, composeName)
290
291
292
293 #get Results - SPARQL
294
295 def get_prescription_interactions(patient, prescription, ontology, dbcon):
296     print("Selecting inappropriate drugs from the ontology")
297     queryDrug1 = list(default_world.sparql(f"""
298             SELECT DISTINCT ?Drug ?Interaction1 ?Interaction2 ?
                    QualityofEvidence ?StrengthofRecommendation ?quant
                     ?recommendation ?detail
299             WHERE {{{ontology}:{prescription} {ontology}:hasDrug ?
                    Drug .
300                 ?Drug rdf:type ?Interaction1 .
301                 ?Interaction1 rdfs:subClassOf ?Interaction2 .
302                 ?Interaction2 rdfs:subClassOf {ontology}:
                        BeersCriteria .
303                 ?Interaction1 rdfs:subClassOf ?QoF .
304                 ?Interaction1 rdfs:subClassOf ?SoR .
305                 ?QoF a owl:Restriction .
306                 ?QoF owl:onProperty {ontology}:
                        hasQualityofEvidence.
307                 ?QoF owl:someValuesFrom ?QualityofEvidence .
308                 ?SoR a owl:Restriction .
309                 ?SoR owl:onProperty {ontology}:
```

```
                hasStrengthofRecommendation.
310             ?SoR owl:someValuesFrom ?StrengthofRecommendation.
311             optional {{?Interaction1 {ontology}:
                    intDrugQuantity ?quant}}
312             optional {{?Interaction1 {ontology}:recommendation
                     ?recommendation}}
313             optional {{?Interaction1 rdfs:comment ?detail}}
314             FILTER( CONTAINS( STR(?Interaction1), STR(?
                    Interaction2) ) )
315             }} """))
316 queryDrug2 = list(default_world.sparql(f"""
317         SELECT DISTINCT ?Drug ?Interaction1 ?Interaction2 ?
                Interaction3 ?QualityofEvidence ?
                StrengthofRecommendation ?quant ?recommendation ?
                detail
318         WHERE {{{ontology}:{prescription} {ontology}:hasDrug ?
                Drug .
319             ?Drug rdf:type ?Interaction1 .
320             ?Interaction1 rdfs:subClassOf ?Interaction2 .
321             ?Interaction2 rdfs:subClassOf ?Interaction3 .
322             ?Interaction3 rdfs:subClassOf {ontology}:
                    BeersCriteria .
323             ?Interaction1 rdfs:subClassOf ?QoF .
324             ?Interaction1 rdfs:subClassOf ?SoR .
325             ?QoF a owl:Restriction .
326             ?QoF owl:onProperty {ontology}:
                    hasQualityofEvidence.
327             ?QoF owl:someValuesFrom ?QualityofEvidence .
328             ?SoR a owl:Restriction .
329             ?SoR owl:onProperty {ontology}:
                    hasStrengthofRecommendation.
330             ?SoR owl:someValuesFrom ?StrengthofRecommendation.
331             optional {{?Interaction1 {ontology}:
                    intDrugQuantity ?quant}}
332             optional {{?Interaction1 {ontology}:recommendation
                     ?recommendation}}
333             optional {{?Interaction1 rdfs:comment ?detail}}
334             FILTER( CONTAINS( STR(?Interaction1), STR(?
                    Interaction3) ) )
335             }} """))
336 queryDrug3 = list(default_world.sparql(f"""
337         SELECT DISTINCT ?Drug ?Interaction1 ?Interaction2 ?
                Interaction3 ?Interaction4 ?QualityofEvidence ?
                StrengthofRecommendation ?quant ?recommendation ?
                detail
338         WHERE {{{ontology}:{prescription} {ontology}:hasDrug ?
                Drug .
339             ?Drug rdf:type ?Interaction1 .
340             ?Interaction1 rdfs:subClassOf ?Interaction2 .
341             ?Interaction2 rdfs:subClassOf ?Interaction3 .
342             ?Interaction3 rdfs:subClassOf ?Interaction4 .
343             ?Interaction4 rdfs:subClassOf {ontology}:
                    BeersCriteria .
344             ?Interaction1 rdfs:subClassOf ?QoF .
345             ?Interaction1 rdfs:subClassOf ?SoR .
346             ?QoF a owl:Restriction .
```

```
347                 ?QoF owl:onProperty {ontology}:
                        hasQualityofEvidence.
348                 ?QoF owl:someValuesFrom ?QualityofEvidence .
349                 ?SoR a owl:Restriction .
350                 ?SoR owl:onProperty {ontology}:
                        hasStrengthofRecommendation.
351                 ?SoR owl:someValuesFrom ?StrengthofRecommendation.
352                 optional {{?Interaction1 {ontology}:
                        intDrugQuantity ?quant}}
353                 optional {{?Interaction1 {ontology}:recommendation
                         ?recommendation}}
354                 optional {{?Interaction1 rdfs:comment ?detail}}
355                 FILTER( CONTAINS( STR(?Interaction1), STR(?
                        Interaction4) ) )
356                 }}
357                 """))
358     queryDrug4 = list(default_world.sparql(f"""
359             SELECT DISTINCT ?Drug ?Interaction2 ?Interaction3 ?
                    Interaction4 ?QualityofEvidence ?
                    StrengthofRecommendation ?quant ?recommendation ?
                    detail
360             WHERE {{{ontology}:{prescription} {ontology}:hasDrug ?
                    Drug .
361                 ?Drug rdf:type ?Interaction1 .
362                 ?Interaction1 rdfs:subClassOf ?Interaction2 .
363                 ?Interaction2 rdfs:subClassOf ?Interaction3 .
364                 ?Interaction3 rdfs:subClassOf ?Interaction4 .
365                 ?Interaction4 rdfs:subClassOf {ontology}:
                        BeersCriteria .
366                 ?Interaction2 rdfs:subClassOf ?QoF .
367                 ?Interaction2 rdfs:subClassOf ?SoR .
368                 ?QoF a owl:Restriction .
369                 ?QoF owl:onProperty {ontology}:
                        hasQualityofEvidence.
370                 ?QoF owl:someValuesFrom ?QualityofEvidence .
371                 ?SoR a owl:Restriction .
372                 ?SoR owl:onProperty {ontology}:
                        hasStrengthofRecommendation.
373                 ?SoR owl:someValuesFrom ?StrengthofRecommendation.
374                 optional {{?Interaction2 {ontology}:
                        intDrugQuantity ?quant}}
375                 optional {{?Interaction2 {ontology}:recommendation
                         ?recommendation}}
376                 optional {{?Interaction2 rdfs:comment ?detail}}
377                 FILTER( CONTAINS( STR(?Interaction2), STR(?
                        Interaction4) ) )
378                 }} """))
379     queryDrug5 = list(default_world.sparql(f"""
380             SELECT DISTINCT ?Drug ?Interaction2 ?Interaction3 ?
                    QualityofEvidence ?StrengthofRecommendation ?quant
                     ?recommendation ?detail
381             WHERE {{{ontology}:{prescription} {ontology}:hasDrug ?
                    Drug .
382                 ?Drug rdf:type ?Interaction1 .
383                 ?Interaction1 rdfs:subClassOf ?Interaction2 .
384                 ?Interaction2 rdfs:subClassOf ?Interaction3 .
```

```
385                 ?Interaction3 rdfs:subClassOf ?Interaction4 .
386                 ?Interaction3 rdfs:subClassOf {ontology}:
                        BeersCriteria .
387                 ?Interaction2 rdfs:subClassOf ?QoF .
388                 ?Interaction2 rdfs:subClassOf ?SoR .
389                 ?QoF a owl:Restriction .
390                 ?QoF owl:onProperty {ontology}:
                        hasQualityofEvidence.
391                 ?QoF owl:someValuesFrom ?QualityofEvidence .
392                 ?SoR a owl:Restriction .
393                 ?SoR owl:onProperty {ontology}:
                        hasStrengthofRecommendation.
394                 ?SoR owl:someValuesFrom ?StrengthofRecommendation.
395                 optional {{?Interaction2 {ontology}:
                        intDrugQuantity ?quant}}
396                 optional {{?Interaction2 {ontology}:recommendation
                         ?recommendation}}
397                 optional {{?Interaction2 rdfs:comment ?detail}}
398                 FILTER( CONTAINS( STR(?Interaction2), STR(?
                        Interaction3) ) )
399                 }} """))
400     queryAlternative1 = list(default_world.sparql(f"""
401             SELECT DISTINCT ?Alternative ?Interaction1 ?
                    Interaction2 ?QualityofEvidence ?
                    StrengthofRecommendation ?quant ?recommendation ?
                    detail
402             WHERE {{{ontology}:{prescription} {ontology}:hasDrug ?
                    Drug .
403                 ?Drug {ontology}:hasAlternative ?Alternative
404                 ?Alternative rdf:type ?Interaction1 .
405                 ?Interaction1 rdfs:subClassOf ?Interaction2 .
406                 ?Interaction2 rdfs:subClassOf {ontology}:
                        BeersCriteria .
407                 ?Interaction1 rdfs:subClassOf ?QoF .
408                 ?Interaction1 rdfs:subClassOf ?SoR .
409                 ?QoF a owl:Restriction .
410                 ?QoF owl:onProperty {ontology}:
                        hasQualityofEvidence.
411                 ?QoF owl:someValuesFrom ?QualityofEvidence .
412                 ?SoR a owl:Restriction .
413                 ?SoR owl:onProperty {ontology}:
                        hasStrengthofRecommendation.
414                 ?SoR owl:someValuesFrom ?StrengthofRecommendation.
415                 optional {{?Interaction1 {ontology}:
                        intDrugQuantity ?quant}}
416                 optional {{?Interaction1 {ontology}:recommendation
                         ?recommendation}}
417                 optional {{?Interaction1 rdfs:comment ?detail}}
418                 FILTER( CONTAINS( STR(?Interaction1), STR(?
                        Interaction2) ) )
419                 }} """))
420     queryAlternative2 = list(default_world.sparql(f"""
421             SELECT DISTINCT ?Alternative ?Interaction1 ?
                    Interaction2 ?Interaction3 ?QualityofEvidence ?
                    StrengthofRecommendation ?quant ?recommendation ?
                    detail
```

```
422             WHERE {{{ontology}:{prescription} {ontology}:hasDrug ?
                    Drug .
423                 ?Drug {ontology}:hasAlternative ?Alternative
424                 ?Alternative rdf:type ?Interaction1 .
425                 ?Interaction1 rdfs:subClassOf ?Interaction2 .
426                 ?Interaction2 rdfs:subClassOf ?Interaction3 .
427                 ?Interaction3 rdfs:subClassOf {ontology}:
                        BeersCriteria .
428                 ?Interaction1 rdfs:subClassOf ?QoF .
429                 ?Interaction1 rdfs:subClassOf ?SoR .
430                 ?QoF a owl:Restriction .
431                 ?QoF owl:onProperty {ontology}:
                        hasQualityofEvidence.
432                 ?QoF owl:someValuesFrom ?QualityofEvidence .
433                 ?SoR a owl:Restriction .
434                 ?SoR owl:onProperty {ontology}:
                        hasStrengthofRecommendation.
435                 ?SoR owl:someValuesFrom ?StrengthofRecommendation.
436                 optional {{?Interaction1 {ontology}:
                        intDrugQuantity ?quant}}
437                 optional {{?Interaction1 {ontology}:recommendation
                         ?recommendation}}
438                 optional {{?Interaction1 rdfs:comment ?detail}}
439                 FILTER( CONTAINS( STR(?Interaction1), STR(?
                        Interaction3) ) )
440                 }} """))
441     queryAlternative3 = list(default_world.sparql(f"""
442             SELECT DISTINCT ?Alternative ?Interaction1 ?
                    Interaction2 ?Interaction3 ?Interaction4 ?
                    QualityofEvidence ?StrengthofRecommendation ?quant
                     ?recommendation ?detail
443             WHERE {{{ontology}:{prescription} {ontology}:hasDrug ?
                    Drug .
444                 ?Drug {ontology}:hasAlternative ?Alternative
445                 ?Alternative rdf:type ?Interaction1 .
446                 ?Interaction1 rdfs:subClassOf ?Interaction2 .
447                 ?Interaction2 rdfs:subClassOf ?Interaction3 .
448                 ?Interaction3 rdfs:subClassOf ?Interaction4 .
449                 ?Interaction4 rdfs:subClassOf {ontology}:
                        BeersCriteria .
450                 ?Interaction1 rdfs:subClassOf ?QoF .
451                 ?Interaction1 rdfs:subClassOf ?SoR .
452                 ?QoF a owl:Restriction .
453                 ?QoF owl:onProperty {ontology}:
                        hasQualityofEvidence.
454                 ?QoF owl:someValuesFrom ?QualityofEvidence .
455                 ?SoR a owl:Restriction .
456                 ?SoR owl:onProperty {ontology}:
                        hasStrengthofRecommendation.
457                 ?SoR owl:someValuesFrom ?StrengthofRecommendation.
458                 optional {{?Interaction1 {ontology}:
                        intDrugQuantity ?quant}}
459                 optional {{?Interaction1 {ontology}:recommendation
                         ?recommendation}}
460                 optional {{?Interaction1 rdfs:comment ?detail}}
461                 FILTER( CONTAINS( STR(?Interaction1), STR(?
```

```
                Interaction4) ) )
462             }}
463             """))
464 queryAlternative4 = list(default_world.sparql(f"""
465         SELECT DISTINCT ?Alternative ?Interaction2 ?
                Interaction3 ?Interaction4 ?QualityofEvidence ?
                StrengthofRecommendation ?quant ?recommendation ?
                detail
466         WHERE {{{ontology}:{prescription} {ontology}:hasDrug ?
                Drug .
467             ?Drug {ontology}:hasAlternative ?Alternative
468             ?Alternative rdf:type ?Interaction1 .
469             ?Interaction1 rdfs:subClassOf ?Interaction2 .
470             ?Interaction2 rdfs:subClassOf ?Interaction3 .
471             ?Interaction3 rdfs:subClassOf ?Interaction4 .
472             ?Interaction4 rdfs:subClassOf {ontology}:
                    BeersCriteria .
473             ?Interaction2 rdfs:subClassOf ?QoF .
474             ?Interaction2 rdfs:subClassOf ?SoR .
475             ?QoF a owl:Restriction .
476             ?QoF owl:onProperty {ontology}:
                    hasQualityofEvidence.
477             ?QoF owl:someValuesFrom ?QualityofEvidence .
478             ?SoR a owl:Restriction .
479             ?SoR owl:onProperty {ontology}:
                    hasStrengthofRecommendation.
480             ?SoR owl:someValuesFrom ?StrengthofRecommendation.
481             optional {{?Interaction2 {ontology}:
                    intDrugQuantity ?quant}}
482             optional {{?Interaction2 {ontology}:recommendation
                     ?recommendation}}
483             optional {{?Interaction2 rdfs:comment ?detail}}
484             FILTER( CONTAINS( STR(?Interaction2), STR(?
                    Interaction4) ) )
485             }} """))
486 queryAlternative5 = list(default_world.sparql(f"""
487         SELECT DISTINCT ?Alternative ?Interaction2 ?
                Interaction3 ?QualityofEvidence ?
                StrengthofRecommendation ?quant ?recommendation ?
                detail
488         WHERE {{{ontology}:{prescription} {ontology}:hasDrug ?
                Drug .
489             ?Drug {ontology}:hasAlternative ?Alternative
490             ?Alternative rdf:type ?Interaction1 .
491             ?Interaction1 rdfs:subClassOf ?Interaction2 .
492             ?Interaction2 rdfs:subClassOf ?Interaction3 .
493             ?Interaction3 rdfs:subClassOf ?Interaction4 .
494             ?Interaction3 rdfs:subClassOf {ontology}:
                    BeersCriteria .
495             ?Interaction2 rdfs:subClassOf ?QoF .
496             ?Interaction2 rdfs:subClassOf ?SoR .
497             ?QoF a owl:Restriction .
498             ?QoF owl:onProperty {ontology}:
                    hasQualityofEvidence.
499             ?QoF owl:someValuesFrom ?QualityofEvidence .
500             ?SoR a owl:Restriction .
```

```
501                 ?SoR owl:onProperty {ontology}:
                        hasStrengthofRecommendation.
502                 ?SoR owl:someValuesFrom ?StrengthofRecommendation.
503                 optional {{?Interaction2 {ontology}:
                        intDrugQuantity ?quant}}
504                 optional {{?Interaction2 {ontology}:recommendation
                         ?recommendation}}
505                 optional {{?Interaction2 rdfs:comment ?detail}}
506                 FILTER( CONTAINS( STR(?Interaction2), STR(?
                        Interaction3) ) )
507                 }} """))
508
509     for data in queryDrug1:
510         patientDb = patient
511         drug = str(data[0]).replace(f"{ontology}.","").replace(f"{prescription
                -","")
512         interaction1 = str(data[2]).replace(f"{ontology}.","")
513         interaction2 = str(data[1]).replace(f"{ontology}.","")
514         interaction3 = str(data[1]).replace(f"{ontology}.","")
515         interaction4 = str(data[1]).replace(f"{ontology}.","")
516         QoF = str(data[3]).replace(f"{ontology}.","")
517         SoR = str(data[4]).replace(f"{ontology}.","")
518         intDrugNum = str(data[5]).replace(f"{ontology}.","")
519         #intDrugCat = str(data[6]).replace(f"{ontology}.","")
520         recommendation = str(data[6]).replace(f"{ontology}.","")
521         detail = str(data[7]).replace(f"{ontology}.","")
522         params = (patientDb, prescription, drug,intDrugNum, interaction1,
                interaction2, interaction3, interaction4, QoF, SoR, detail, recommendation
                , "F")
523         format_string = "Patient DB: {}\nPrescription: {}\nDrug: {}\
                nInt Drug Num: {}\nInteraction 1: {}\nInteraction 2:
                {}\nInteraction 3: {}\nInteraction 4: {}\nQoF: {}\nSoR
                : {}\nDetail: {}\nRecommendation: {}\nAlternative: {}\
                n"
524         # Print the formatted string with tuple values
525         formatted_params = format_string.format(*params)
526         print(formatted_params, end='\n')
527         dbcon.execute("INSERT INTO PRESC_INTERACTION values
                (?,?,?,?,?,?,?,?,?,?,?,?,?)", params)
528
529     for data in queryDrug2:
530         patientDb = patient
531         drug = str(data[0]).replace(f"{ontology}.","").replace(f"{prescription
                -","")
532         interaction4 = str(data[1]).replace(f"{ontology}.","")
533         interaction3 = str(data[1]).replace(f"{ontology}.","")
534         interaction2 = str(data[2]).replace(f"{ontology}.","")
535         interaction1 = str(data[3]).replace(f"{ontology}.","")
536         QoF = str(data[4]).replace(f"{ontology}.","")
537         SoR = str(data[5]).replace(f"{ontology}.","")
538         intDrugNum = str(data[6]).replace(f"{ontology}.","")
539         #intDrugCat = str(data[6]).replace(f"{ontology}.","")
540         recommendation = str(data[7]).replace(f"{ontology}.","")
541         detail = str(data[8]).replace(f"{ontology}.","")
542         params = (patientDb, prescription, drug,intDrugNum, interaction1,
                interaction2, interaction3, interaction4, QoF, SoR, detail, recommendation
```

```
            , "F")
543         format_string = "Patient DB: {}\nPrescription: {}\nDrug: {}\
                nInt Drug Num: {}\nInteraction 1: {}\nInteraction 2:
                {}\nInteraction 3: {}\nInteraction 4: {}\nQoF: {}\nSoR
                : {}\nDetail: {}\nRecommendation: {}\nAlternative: {}\
                n"
544         # Print the formatted string with tuple values
545         formatted_params = format_string.format(*params)
546         print(formatted_params, end='\n')
547         dbcon.execute("INSERT INTO PRESC_INTERACTION values
                (?,?,?,?,?,?,?,?,?,?,?,?,?)", params)
548
549     for data in queryDrug3:
550         patientDb = patient
551         drug = str(data[0]).replace(f"{ontology}.",'').replace(f"{prescription
                }-",'')
552         interaction4 = str(data[1]).replace(f"{ontology}.",'')
553         interaction3 = str(data[1]).replace(f"{ontology}.",'')
554         interaction2 = str(data[2]).replace(f"{ontology}.",'')
555         interaction1 = str(data[4]).replace(f"{ontology}.",'')
556         QoF = str(data[5]).replace(f"{ontology}.",'')
557         SoR = str(data[6]).replace(f"{ontology}.",'')
558         intDrugNum = str(data[7]).replace(f"{ontology}.",'')
559         #intDrugCat = str(data[6]).replace(f"{ontology}.",'')
560         recommendation = str(data[8]).replace(f"{ontology}.",'')
561         detail = str(data[9]).replace(f"{ontology}.",'')
562         params = (patientDb, prescription, drug,intDrugNum, interaction1,
                interaction2, interaction3, interaction4, QoF, SoR, detail, recommendation
                , "F")
563         format_string = "Patient DB: {}\nPrescription: {}\nDrug: {}\
                nInt Drug Num: {}\nInteraction 1: {}\nInteraction 2:
                {}\nInteraction 3: {}\nInteraction 4: {}\nQoF: {}\nSoR
                : {}\nDetail: {}\nRecommendation: {}\nAlternative: {}\
                n"
564         # Print the formatted string with tuple values
565         formatted_params = format_string.format(*params)
566         print(formatted_params, end='\n')
567         dbcon.execute("INSERT INTO PRESC_INTERACTION values
                (?,?,?,?,?,?,?,?,?,?,?,?,?)", params)
568
569     for data in queryDrug4:
570         patientDb = patient
571         drug = str(data[0]).replace(f"{ontology}.",'').replace(f"{prescription
                }-",'')
572         interaction4 = str(data[1]).replace(f"{ontology}.",'')
573         interaction3 = str(data[1]).replace(f"{ontology}.",'')
574         interaction2 = str(data[2]).replace(f"{ontology}.",'')
575         interaction1 = str(data[3]).replace(f"{ontology}.",'')
576         QoF = str(data[4]).replace(f"{ontology}.",'')
577         SoR = str(data[5]).replace(f"{ontology}.",'')
578         intDrugNum = str(data[6]).replace(f"{ontology}.",'')
579         #intDrugCat = str(data[6]).replace(f"{ontology}.",'')
580         recommendation = str(data[7]).replace(f"{ontology}.",'')
581         detail = str(data[8]).replace(f"{ontology}.",'')
582         params = (patientDb, prescription, drug,intDrugNum, interaction1,
                interaction2, interaction3, interaction4, QoF, SoR, detail, recommendation
```

```
            , "F")
583         format_string = "Patient DB: {}\nPrescription: {}\nDrug: {}\
                nInt Drug Num: {}\nInteraction 1: {}\nInteraction 2:
                {}\nInteraction 3: {}\nInteraction 4: {}\nQoF: {}\nSoR
                : {}\nDetail: {}\nRecommendation: {}\nAlternative: {}\
                n"
584         # Print the formatted string with tuple values
585         formatted_params = format_string.format(*params)
586         print(formatted_params, end='\n')
587         dbcon.execute("INSERT INTO PRESC_INTERACTION values
                (?,?,?,?,?,?,?,?,?,?,?,?,?)", params)
588
589     for data in queryDrug5:
590         patientDb = patient
591         drug = str(data[0]).replace(f"{ontology}.",'').replace(f"{prescription
                }-",'')
592         interaction1 = str(data[2]).replace(f"{ontology}.",'')
593         interaction2 = str(data[1]).replace(f"{ontology}.",'')
594         interaction3 = str(data[1]).replace(f"{ontology}.",'')
595         interaction4 = str(data[1]).replace(f"{ontology}.",'')
596         QoF = str(data[3]).replace(f"{ontology}.",'')
597         SoR = str(data[4]).replace(f"{ontology}.",'')
598         intDrugNum = str(data[5]).replace(f"{ontology}.",'')
599         #intDrugCat = str(data[6]).replace(f"{ontology}.",'')
600         recommendation = str(data[6]).replace(f"{ontology}.",'')
601         detail = str(data[7]).replace(f"{ontology}.",'')
602         params = (patientDb, prescription, drug,intDrugNum, interaction1,
                interaction2, interaction3, interaction4, QoF, SoR, detail, recommendation
                , "F")
603         format_string = "Patient DB: {}\nPrescription: {}\nDrug: {}\
                nInt Drug Num: {}\nInteraction 1: {}\nInteraction 2:
                {}\nInteraction 3: {}\nInteraction 4: {}\nQoF: {}\nSoR
                : {}\nDetail: {}\nRecommendation: {}\nAlternative: {}\
                n"
604         # Print the formatted string with tuple values
605         formatted_params = format_string.format(*params)
606         print(formatted_params, end='\n')
607         dbcon.execute("INSERT INTO PRESC_INTERACTION values
                (?,?,?,?,?,?,?,?,?,?,?,?,?)", params)
608
609     for data in queryAlternative1:
610         patientDb = patient
611         drug = str(data[0]).replace(f"{ontology}.",'').replace(f"{prescription
                }-",'')
612         interaction1 = str(data[2]).replace(f"{ontology}.",'')
613         interaction2 = str(data[1]).replace(f"{ontology}.",'')
614         interaction3 = str(data[1]).replace(f"{ontology}.",'')
615         interaction4 = str(data[1]).replace(f"{ontology}.",'')
616         QoF = str(data[3]).replace(f"{ontology}.",'')
617         SoR = str(data[4]).replace(f"{ontology}.",'')
618         intDrugNum = str(data[5]).replace(f"{ontology}.",'')
619         #intDrugCat = str(data[6]).replace(f"{ontology}.",'')
620         recommendation = str(data[6]).replace(f"{ontology}.",'')
621         detail = str(data[7]).replace(f"{ontology}.",'')
622         params = (patientDb, prescription, drug,intDrugNum, interaction1,
                interaction2, interaction3, interaction4, QoF, SoR, detail, recommendation
```

```
            , "T")
623         format_string = "Patient DB: {}\nPrescription: {}\nDrug: {}\
                nInt Drug Num: {}\nInteraction 1: {}\nInteraction 2:
                {}\nInteraction 3: {}\nInteraction 4: {}\nQoF: {}\nSoR
                : {}\nDetail: {}\nRecommendation: {}\nAlternative: {}\
                n"
624         # Print the formatted string with tuple values
625         formatted_params = format_string.format(*params)
626         print(formatted_params, end='\n')
627         dbcon.execute("INSERT INTO PRESC_INTERACTION values
                (?,?,?,?,?,?,?,?,?,?,?,?,?)", params)
628
629     for data in queryAlternative2:
630         patientDb = patient
631         drug = str(data[0]).replace(f"{ontology}.",'').replace(f"{prescription
                }-",'')
632         interaction4 = str(data[1]).replace(f"{ontology}.",'')
633         interaction3 = str(data[1]).replace(f"{ontology}.",'')
634         interaction2 = str(data[2]).replace(f"{ontology}.",'')
635         interaction1 = str(data[3]).replace(f"{ontology}.",'')
636         QoF = str(data[4]).replace(f"{ontology}.",'')
637         SoR = str(data[5]).replace(f"{ontology}.",'')
638         intDrugNum = str(data[6]).replace(f"{ontology}.",'')
639         #intDrugCat = str(data[6]).replace(f"{ontology}.",'')
640         recommendation = str(data[7]).replace(f"{ontology}.",'')
641         detail = str(data[8]).replace(f"{ontology}.",'')
642         params = (patientDb, prescription, drug,intDrugNum, interaction1,
                interaction2, interaction3, interaction4, QoF, SoR, detail, recommendation
                , "T")
643         format_string = "Patient DB: {}\nPrescription: {}\nDrug: {}\
                nInt Drug Num: {}\nInteraction 1: {}\nInteraction 2:
                {}\nInteraction 3: {}\nInteraction 4: {}\nQoF: {}\nSoR
                : {}\nDetail: {}\nRecommendation: {}\nAlternative: {}\
                n"
644         # Print the formatted string with tuple values
645         formatted_params = format_string.format(*params)
646         print(formatted_params, end='\n')
647         dbcon.execute("INSERT INTO PRESC_INTERACTION values
                (?,?,?,?,?,?,?,?,?,?,?,?,?)", params)
648
649     for data in queryAlternative3:
650         patientDb = patient
651         drug = str(data[0]).replace(f"{ontology}.",'').replace(f"{prescription
                }-",'')
652         interaction4 = str(data[1]).replace(f"{ontology}.",'')
653         interaction3 = str(data[1]).replace(f"{ontology}.",'')
654         interaction2 = str(data[2]).replace(f"{ontology}.",'')
655         interaction1 = str(data[4]).replace(f"{ontology}.",'')
656         QoF = str(data[5]).replace(f"{ontology}.",'')
657         SoR = str(data[6]).replace(f"{ontology}.",'')
658         intDrugNum = str(data[7]).replace(f"{ontology}.",'')
659         #intDrugCat = str(data[6]).replace(f"{ontology}.",'')
660         recommendation = str(data[8]).replace(f"{ontology}.",'')
661         detail = str(data[9]).replace(f"{ontology}.",'')
662         params = (patientDb, prescription, drug,intDrugNum, interaction1,
                interaction2, interaction3, interaction4, QoF, SoR, detail, recommendation
```

```
            , "T")
663         format_string = "Patient DB: {}\nPrescription: {}\nDrug: {}\
                nInt Drug Num: {}\nInteraction 1: {}\nInteraction 2:
                {}\nInteraction 3: {}\nInteraction 4: {}\nQoF: {}\nSoR
                : {}\nDetail: {}\nRecommendation: {}\nAlternative: {}\
                n"
664         # Print the formatted string with tuple values
665         formatted_params = format_string.format(*params)
666         print(formatted_params, end='\n')
667         dbcon.execute("INSERT INTO PRESC_INTERACTION values
                (?,?,?,?,?,?,?,?,?,?,?,?,?)", params)
668
669     for data in queryAlternative4:
670         patientDb = patient
671         drug = str(data[0]).replace(f"{ontology}.",'').replace(f"{prescription
                }-",'')
672         interaction4 = str(data[1]).replace(f"{ontology}.",'')
673         interaction3 = str(data[1]).replace(f"{ontology}.",'')
674         interaction2 = str(data[2]).replace(f"{ontology}.",'')
675         interaction1 = str(data[3]).replace(f"{ontology}.",'')
676         QoF = str(data[4]).replace(f"{ontology}.",'')
677         SoR = str(data[5]).replace(f"{ontology}.",'')
678         intDrugNum = str(data[6]).replace(f"{ontology}.",'')
679         #intDrugCat = str(data[6]).replace(f"{ontology}.",'')
680         recommendation = str(data[7]).replace(f"{ontology}.",'')
681         detail = str(data[8]).replace(f"{ontology}.",'')
682         params = (patientDb, prescription, drug,intDrugNum, interaction1,
                interaction2, interaction3, interaction4, QoF, SoR, detail, recommendation
                , "T")
683         format_string = "Patient DB: {}\nPrescription: {}\nDrug: {}\
                nInt Drug Num: {}\nInteraction 1: {}\nInteraction 2:
                {}\nInteraction 3: {}\nInteraction 4: {}\nQoF: {}\nSoR
                : {}\nDetail: {}\nRecommendation: {}\nAlternative: {}\
                n"
684         # Print the formatted string with tuple values
685         formatted_params = format_string.format(*params)
686         print(formatted_params, end='\n')
687         dbcon.execute("INSERT INTO PRESC_INTERACTION values
                (?,?,?,?,?,?,?,?,?,?,?,?,?)", params)
688
689     for data in queryAlternative5:
690         patientDb = patient
691         drug = str(data[0]).replace(f"{ontology}.",'').replace(f"{prescription
                }-",'')
692         interaction1 = str(data[2]).replace(f"{ontology}.",'')
693         interaction2 = str(data[1]).replace(f"{ontology}.",'')
694         interaction3 = str(data[1]).replace(f"{ontology}.",'')
695         interaction4 = str(data[1]).replace(f"{ontology}.",'')
696         QoF = str(data[3]).replace(f"{ontology}.",'')
697         SoR = str(data[4]).replace(f"{ontology}.",'')
698         intDrugNum = str(data[5]).replace(f"{ontology}.",'')
699         #intDrugCat = str(data[6]).replace(f"{ontology}.",'')
700         recommendation = str(data[6]).replace(f"{ontology}.",'')
701         detail = str(data[7]).replace(f"{ontology}.",'')
702         params = (patientDb, prescription, drug,intDrugNum, interaction1,
                interaction2, interaction3, interaction4, QoF, SoR, detail, recommendation
```

```
            ,"T")
703         format_string = "Patient DB: {}\nPrescription: {}\nDrug: {}\
                nInt Drug Num: {}\nInteraction 1: {}\nInteraction 2:
                {}\nInteraction 3: {}\nInteraction 4: {}\nQoF: {}\nSoR
                : {}\nDetail: {}\nRecommendation: {}\nAlternative: {}\
                n"
704         # Print the formatted string with tuple values
705         formatted_params = format_string.format(*params)
706         print(formatted_params, end='\n')
707         dbcon.execute("INSERT INTO PRESC_INTERACTION values
                (?,?,?,?,?,?,?,?,?,?,?,?,?)", params)
708
709
710 def get_alternatives(patient,prescription, ontology, dbcon):
711     print("Select alternative drugs from the ontology")
712     response = list(default_world.sparql(f"""
713             SELECT DISTINCT ?Drug ?Alternative
714             WHERE {{{ontology}:{prescription} {ontology}:hasDrug ?
                    Drug .
715                 ?Drug {ontology}:hasAlternative ?Alternative }}""")
                    )
716
717     for data in response:
718         drug = str(data[0]).replace(f"{ontology}.",'').replace(f"{prescription
                }-",'')
719         alternative = str(data[1]).replace(f"{ontology}.",'')
720         params = (patient, prescription, drug,alternative)
721         format_string = "Patient: {}\nPrescription: {}\nDrug: {}\
                nAlternative: {}\n"
722         # Print the formatted string with tuple values
723         formatted_params = format_string.format(*params)
724         print(formatted_params, end='\n')
725
726         dbcon.execute("INSERT INTO DRUG_ALTERNATIVE values (?,?,?,?)
                ", params)
727
728
729 def set_prescription_as_processed(patient, prescription, ontology, prescFile, dbcon):
730         params = (patient, prescription)
731         dbcon.execute("INSERT INTO PRESCRIPTION_PROCESSED values
                (?,?)", params)
732
733
734 def get_drug_drug_interaction(patient, prescription, ontology, dbcon):
735     print("Selecting drug x drug interactions from the ontology")
736     response = list(default_world.sparql(f"""
737         SELECT DISTINCT ?Drug ?DrugInt
738         WHERE {{{ontology}:{prescription} {ontology}:hasDrug ?
                Drug .
739             ?Drug {ontology}:hasInteractionWith ?DrugInt }} """))
740     for data in response:
741         drug1 = str(data[0]).replace(f"{ontology}.",'').replace(f"{
                prescription}-",'')
742         drug2 = str(data[1]).replace(f"{ontology}.",'').replace(f"{
                prescription}-",'')
743         params = (patient, prescription, drug1, drug2)
```

```
744         format_string = "Patient: {}\nPrescription: {}\nDrug: {}\
                nDrug interaction: {}"
745         # Print the formatted string with tuple values
746         formatted_params = format_string.format(*params)
747         print(formatted_params, end='\n')
748         dbcon.execute("INSERT INTO DRUG_DRUG_INTERACTION values
                (?,?,?,?)", params)
749
750
751 def current_time():
752     now = datetime.now()
753     current_time = now.strftime("%H:%M:%S")
754     return(current_time)
755
756
757 def get_prescriptions(patient):
758     df = pd.read_sql_query(f"""SELECT *
759                                FROM PRESCRIPTION A
760                                WHERE CD_PATIENT = {patient}
761                                AND CD_PRESCRIPTION NOT IN (
                                       SELECT DISTINCT(
                                       CD_PRESCRIPTION) FROM
                                       PRESCRIPTION_PROCESSED WHERE
                                        CD_PATIENT = {patient})""",
                                       conn)
762     return(df)
763
764 def get_patient(patient):
765     df = pd.read_sql_query(f"SELECT * FROM PATIENT WHERE CD_PATIENT = {
            patient}", conn)
766     return(df)
767
768 def get_exam(patient, day):
769     df = pd.read_sql_query(f"SELECT * FROM PATIENT_EXAMS WHERE
            CD_PATIENT = {patient} and dt_Result = '{day}'", conn)
770     return(df)
771
772 def get_previousDiseases(patient):
773     df = pd.read_sql_query(f"SELECT * FROM PATIENT_PREVIOUS_DISEASES
            WHERE CD_PATIENT = {patient}", conn)
774     return(df)
775
776 def getpatientProcessed(conn):
777     df = pd.read_sql_query("SELECT DISTINCT CD_PATIENT from
            PRESCRIPTION_PROCESSED", conn)
778     patients = df.values.tolist()
779     patientlist = []
780     for patientl1 in patients:
781         for patientl2 in patientl1:
782             patientlist.append(int(patientl2))
783     return(patientlist)
784
785
786 def getNumDrugprocessed(conn):
787     df = pd.read_sql_query("SELECT count( DISTINCT CD_PRESCRIPTION)
            from PRESCRIPTION_PROCESSED", conn)
```

```
788     patients = df.values.tolist()
789     return(patients)
790
791 def getNumDrugs(conn):
792     df = pd.read_sql_query("SELECT count( DISTINCT CD_PRESCRIPTION)
            from PRESCRIPTION", conn)
793     patients = df.values.tolist()
794     return(patients)
795
796
797 def get_patient_age(patient):
798     df = pd.read_sql_query(f'SELECT distinct(age) from PATIENT WHERE
            CD_PATIENT = {patient}', conn)
799     patients = df.values.tolist()
800     if patients:
801         return(int(patients[0][0]))
802     else:
803         return(999)
804
805 def get_patient_gender(patient):
806     df = pd.read_sql_query(f'SELECT distinct(gender) from PATIENT WHERE
             CD_PATIENT = {patient}', conn)
807     patients = df.values.tolist()
808     if patients:
809         return(str(patients[0][0]))
810     else:
811         return('M')
812
813 ##################MAIN APPLICATION#################
814
815 def checkInteraction(key):
816     print(f'########################Inference engine: Beers
            Criteria Interactions######################### \n')
817     patient_prescriptions = get_prescriptions(key)
818     if not patient_prescriptions.empty:
819         startTime = datetime.now()
820         totalRows = getNumDrugs(conn)
821         print(f'Total prescription:{totalRows} / Prescription
                processed: {getNumDrugprocessed(conn)}')
822         prescDays = listPrescrDays(patient_prescriptions)
823         age = get_patient_age(key)
824         gender = get_patient_gender(key)
825         for day in prescDays:
826             print(f"Patient:{key} - Day:{day} Time: {time.ctime(time.
                    time())} - Inserting patient data into the
                    ontology")
827             daystr = str(day).replace('/','').replace("-",'')
828             #Add patient into the ontology
829             patientData = get_patient(key)
830             patient = addOntologyPatients(patientData)
831
832             #Add exam into the ontology filter by patient and date
833             examData = get_exam(key, day)
834             add_ontology_exam(examData, age, gender)
835
836             #Add disease into the ontology filter by patient and date
```

```
837             diseaseData = get_previousDiseases(key)
838             add_disease(diseaseData, age, gender)
839
840             #Add prescription drugs into the ontology filter by patient and date
841             dtFilterPrescriptions = patient_prescriptions.loc[patient_prescriptions.
                    START_DATE == day, :]
842             index = dtFilterPrescriptions.index.tolist()
843             patient_prescr = add_prescription(dtFilterPrescriptions, age, gender)
844
845             add_ontology_drug_assertions(key, daystr, age, gender)
846             composeName = str(key)+"-"+str(age)+"-"+gender
847             classObjDrugs = ONTO.search(iri = '*'+str(composeName)+'*',
                    _case_sensitive = False)
848
849             diff = datetime.now() - startTime
850             days, seconds = diff.days, diff.seconds
851             hours = days * 24 + seconds // 3600
852             minutes = (seconds % 3600) // 60
853
854             AllDifferent(classObjDrugs)
855
856             #Save a copy of the ontology with patient data
857             ONTO.save(rf'inference_engines/ontology_files/{key}.owl'
                    , format = "rdfxml")
858
859             print(f"Patient:{key} - Day:{day} Time: {time.ctime(time.
                    time())} - Sync reasoner pellet")
860             sync_reasoner_pellet(infer_property_values = True,
                    infer_data_property_values = True)
861             os.system('cls' if os.name == 'nt' else 'clear')
862
863
864             print(f"Patient:{key} - Day:{day} Time: {time.ctime(time.
                    time())} - SPARQL - Querying results and inserting
                     into the CDSS DB ")
865             presc_day = str(key)+"-"+str(age)+"-"+gender+"-"+str(daystr)
866             print('Inappropriate drugs \n')
867             get_prescription_interactions(key,presc_day, ONTOLOGY_NAME.
                    replace('.',''), c) #Get interactions from the ontology
868             print('Alternative drugs\n')
869             get_alternatives(key, presc_day,ONTOLOGY_NAME.replace('.',''), c)
                     # Get the alternative drugs
870             print('Drug-drug interaction\n')
871             get_drug_drug_interaction(key,presc_day, ONTOLOGY_NAME.replace(
                    '.',''), c) # Get interaction between drugs
872             set_prescription_as_processed(key,presc_day, ONTOLOGY_NAME.
                    replace('.',''),patient_prescr, c) # Define the prescription as
                    processed
873             print(f"Patient:{key} - Day:{day} Time: {time.ctime(time.
                    time())} - Deleting patient data from the ontology
                     \n")
874
875             classObj = ONTO.search(iri = '*'+str(composeName)+'*',
                    _case_sensitive = False)
876             for individual in classObj: #Delete patient data
877                 destroy_entity(individual)
```

```
878             os.system('cls' if os.name == 'nt' else 'clear')
879         conn.commit()
```

F.3 Inference engine Alternative solver

The Alternative solver inference engine collects the patient data, interactions and alternative drugs from the CDSS database to integrate them with the SMT solver (z3_alternativeDrug) to then get the valid alternatives. These alternatives are then stored in the CDSS database.

Filename: ie_smt_alternative_solver.py

```
1 import json, sys, glob, io, os
2 import sqlite3
3 import pandas as pd
4 import time
5 from config import DBNAME
6
7
8 from z3_alternativeDrug import Solver_obj, check_prescription
9
10 conn = sqlite3.connect(DBNAME, timeout=10)
11
12
13 def getProcessedPatients(conn, patient):
14     df = pd.read_sql_query(f""" SELECT DISTINCT CD_PATIENT,
           CD_PRESCRIPTION
15                                from PRESC_INTERACTION
16                                where CD_PATIENT = {patient}
17                                and CD_PRESCRIPTION not in(SELECT
                                      CD_PRESCRIPTION FROM
                                      PRESCRIPTION_MODELS)
18                                and CD_PRESCRIPTION not in(SELECT
                                      CD_PRESCRIPTION FROM
                                      TIMEOUT_PRESCRIPTIONS)""", conn)
19     patients = df.values.tolist()
20     return(patients)
21
22
23 def getInteractionList(prescriptionPar, conn):
24     df = pd.read_sql_query(f"""
25             SELECT DRUG1, DRUG2 from DRUG_DRUG_INTERACTION
26             WHERE CD_PRESCRIPTION = "{prescriptionPar}" """, conn)
27     prescriptions = df.values.tolist()
28     prescriptionList = []
29     for prescription1 in prescriptions:
30         temp = []
31         for prescription2 in prescription1:
32             temp.append(prescription2.replace(',','').replace('.','').replace('-',
                   '_').replace('/','_').replace("'",''))
33         prescriptionList.append(temp)
```

```
34      return(prescriptionList)
35
36  def getPrescrDrugsList(prescriptionPar, conn):
37      df = pd.read_sql_query(f"""
38              SELECT DISTINCT DRUG from PRESCRIPTION
39              WHERE CD_PRESCRIPTION = "{prescriptionPar}" """, conn)
40      prescriptions = df.values.tolist()
41      prescriptionList = []
42      for prescription1 in prescriptions:
43          for prescription2 in prescription1:
44              prescriptionList.append(prescription2.replace(',','').replace('.','').
                    replace('-','_').replace('/','_').replace("'",''))
45      return(prescriptionList)
46
47  def getPrescrAlternativeList(prescriptionPar, conn):
48      df = pd.read_sql_query(f"""
49              SELECT DISTINCT DRUG from DRUG_ALTERNATIVE
50              WHERE CD_PRESCRIPTION = "{prescriptionPar}" """, conn)
51      drugs = df.values.tolist()
52      alternativeList = []
53
54      for druglist in drugs:
55          for drug in druglist:
56              alternatives = []
57              alternatives.append(drug.replace(',',''))
58              df = pd.read_sql_query(f"""
59                  SELECT DISTINCT ALTERNATIVE from DRUG_ALTERNATIVE
60                  WHERE CD_PRESCRIPTION = "{prescriptionPar}"
61                  AND DRUG = "{drug}" """, conn)
62              for alterlist in df.values.tolist():
63                  for alter in alterlist:
64                      alternatives.append(alter.replace(',','').replace('.','').
                            replace('-','_').replace('/','_').replace("'",''))
65              alternativeList.append(alternatives)
66      return(alternativeList)
67
68  def getNumDrugprocessed(conn):
69      df = pd.read_sql_query("SELECT count(CD_PATIENT) from
            PRESCRIPTION_MODELS", conn)
70      patients = df.values.tolist()
71      return(patients)
72
73  def getNumDrugs(conn):
74      df = pd.read_sql_query("SELECT count(CD_PATIENT) from PRESCRIPTION
            ", conn)
75      patients = df.values.tolist()
76      return(patients)
77
78  def checkAlternative(patient):
79      print(f'##########################Inference engine: SMT
            alternative solver########################## \n')
80      totalDrugs = getNumDrugs(conn)
81      drugsProcessed = getNumDrugprocessed(conn)
82      registers = getProcessedPatients(conn, patient)
83      for patient, prescription in registers:
84          print(f"Patient:{patient} - Prescription:{prescription}
```

```
            Time: {time.ctime(time.time()) } - Selecting patient
            inappropriate and alterantive drugs")
85          drugs = getPrescrDrugsList(prescription, conn)
86          alternative = getPrescrAlternativeList(prescription, conn)
87          interaction = getInteractionList(prescription, conn)
88          solverObject = Solver_obj
89          solverObject.nr_prescription = prescription
90          solverObject.prescription = drugs
91          solverObject.interaction = interaction
92          solverObject.alternative = alternative
93
94          print(f"Patient:{patient} - Prescription:{prescription}
                Time: {time.ctime(time.time()) } - Inserting data into
                 the SMT Solver")
95          # Print the solverObject and its attributes
96          print(f"Solver Object Information:\nPrescription number: {
                solverObject.nr_prescription}\nPrescription: {
                solverObject.prescription}\nInteraction: {solverObject
                .interaction}\nAlternative: {solverObject.alternative}
                ")
97          models = check_prescription(solverObject)
98
99          print(f"Patient:{patient} - Prescription:{prescription}
                Time: {time.ctime(time.time()) } - Inserting results
                into the CDSS DB")
100         if models == 'canceled':
101             params = (patient, prescription)
102             conn.execute("INSERT INTO TIMEOUT_PRESCRIPTIONS values
                    (?,?)", params)
103         elif models:
104             for model in models:
105                 params = (patient, prescription,str(model))
106                 format_string = "Patient: {}\nPrescription number: {}\
                        nValidPrescriptions: {}\n"
107                 formatted_params = format_string.format(*params)
108                 print(formatted_params, end='\n')
109                 conn.execute("INSERT INTO PRESCRIPTION_MODELS values
                        (?,?,?)", params)
110         else:
111             params = (patient, prescription,'null')
112             format_string = "Patient: {}\nPrescription number: {}\
                    nValidPrescriptions: {}\n"
113             formatted_params = format_string.format(*params)
114             print(formatted_params, end='\n')
115             conn.execute("INSERT INTO PRESCRIPTION_MODELS values
                    (?,?,?)", params)
116         conn.commit()
```

F.4 Inference engine Rescheduling Solver

The Rescheduling Solver inference engine collects the patient data and interactions from the CDSS database to integrate them with the SMT solver (z3_scheduling) to get the schedules the solver defines. These scheduled times are then stored in the

CDSS database.

Filename: ie_smt_rescheduling_solver.py

```
import sqlite3, sys
import pandas as pd
import time
import re
from owlready2 import *
from z3_scheduling import schedulingDrugs
from config import DBNAME, BEERSVERSION


version = BEERSVERSION
ONTOLOGY = fr'inference_engines/ontology_files/BeersOntologyV{version}.owl'
ONTOLOGY_NAME = f"BeersOntologyV{version}"
ONTO = get_ontology(ONTOLOGY).load()

conn = sqlite3.connect(DBNAME, timeout=10)


def checkLabel(name, type): #check if exist the label drug on the onotlogy and return the class name
    ONTOLOGY = fr'inference_engines/ontology_files/BeersOntologyV{version}.owl'
    ONTOLOGY_NAME = f"BeersOntologyV{version}"
    ONTO = get_ontology(ONTOLOGY).load()
    ch = '.'
    pattern = ".*" + ch
    classObj = ONTO.search(label = (name), _case_sensitive = False)
    if not classObj:
        classObj = ONTO.search(label = (name.replace('-', '_')), _case_sensitive = False)
    if not classObj:
        classObj = ONTO.search(label = (name.replace('_', '-')), _case_sensitive = False)
    if classObj:
        classObj = str(classObj[0]).split('.', 1)[-1]

        if isinstance(ONTO[classObj], ONTO[type]):
            return classObj
    else:
        return(None)



def getTmax(drug, ontology):
    value = 0
    labeldrug = checkLabel(drug, 'Drugs')
    response = list(default_world.sparql(f"""
            SELECT DISTINCT ?value
            WHERE {{{ontology}:{labeldrug} rdfs:subClassOf ?object .
                ?object a owl:Restriction .
```

```
46                 ?object owl:onProperty {ontology}:hasTmax.
47                 ?object owl:qualifiedCardinality ?value }}"""))
48
49     for data in response:
50             value = str(data[0])
51     if value == 0:
52         value = 120
53     return(value)
54
55
56
57 def getPatients(conn, patient):
58     df = pd.read_sql_query(f""" SELECT DISTINCT CD_PATIENT FROM
           PRESCRIPTION_MODELS
59                                 WHERE CD_PATIENT = {patient}
60                                 and CD_PATIENT NOT IN (SELECT
                                       CD_PATIENT FROM
                                       PRESCRIPTION_RESCHEDULED)
61                                 AND MODEL = "null"
62                                 """, conn)
63     patients = df.values.tolist()
64     patients = [patient for patientList in patients for patient in patientList]
65     return(patients)
66
67
68 def getUnsatPrescriptions(conn, patientInput):
69     df = pd.read_sql_query(f""" SELECT DISTINCT CD_PRESCRIPTION FROM
           PRESCRIPTION_MODELS
70                                 WHERE CD_PATIENT = "{patientInput}"
71                                 AND MODEL = "null"
72                                 AND CD_PRESCRIPTION NOT IN (SELECT
                                       CD_PRESCRIPTION FROM
                                       PRESCRIPTION_RESCHEDULED)
73                                                                """,
                                                                    conn
                                                                    )
74
75     prescriptions = df.values.tolist()
76     prescriptions = [prescription for prescriptionList in prescriptions for prescription in
           prescriptionList]
77     return(prescriptions)
78
79
80 def getInteractions(conn, prescription):
81     df = pd.read_sql_query(f""" SELECT B.DRUG1, B.DRUG2
82                                 FROM DRUG_DRUG_INTERACTION AS B
83                                 WHERE B.CD_PRESCRIPTION = "{
                                       prescription}"
84                                 AND B.DRUG2 <> B.DRUG1""", conn)
85     interactions = df.values.tolist()
86     return(interactions)
87
88 def getInteractionsWithoutAlternative(conn, prescription):
89     df = pd.read_sql_query(f""" SELECT B.DRUG1, B.DRUG2
90                                 FROM DRUG_DRUG_INTERACTION AS B
91                                 WHERE B.CD_PRESCRIPTION = "{
```

```
                                    prescription}"
92                                  AND B.DRUG2 <> B.DRUG1
93                                  and B.DRUG2 not in (SELECT DISTINCT
                                        A.DRUG from DRUG_ALTERNATIVE AS
                                        A
94                                                      WHERE A.
                                                            CD_PRESCRIPTION
                                                             = "{prescription}
                                                            ")
95                                                      """, conn)
96      interactions = df.values.tolist()
97      return(interactions)
98
99
100 def getPrescDrugDetails(conn, prescription, drug):
101     df = pd.read_sql_query(f""" SELECT DISTINCT FREQUENCY, SCHEDULE,
            FIXEDTIME
102                                 FROM PRESCRIPTION
103                                 WHERE CD_PRESCRIPTION = "{prescription}"
104                                 AND DRUG = "{drug}" """, conn)
105     interactions = df.values.tolist()
106     return(interactions)
107
108
109 def getNumDrugprocessed(conn):
110     df = pd.read_sql_query("SELECT count(CD_PATIENT) from
            PRESCRIPTION_MODELS", conn)
111     patients = df.values.tolist()
112     return(patients)
113
114 def getNumDrugs(conn):
115     df = pd.read_sql_query("SELECT count(CD_PATIENT) from PRESCRIPTION
            ", conn)
116     patients = df.values.tolist()
117     return(patients)
118
119
120
121
122 def checkRescheduling(patient):
123     print(f'##########################Inference engine: SMT
            rescheduling solver########################## \n')
124     totalDrugs = getNumDrugs(conn)
125     patients = getPatients(conn, patient)
126
127     for patient in patients:
128         prescriptions = getUnsatPrescriptions(conn, patient)
129         for prescription in prescriptions:
130             print(f"Patient:{patient} - Prescription:{prescription}
                    Time: {time.ctime(time.time()) } - Selecting
                    inappropriate drugs")
131             interaction = getInteractionsWithoutAlternative(conn, prescription)
132             druglist = []
133             drugdic = {}
134             if interaction:
135                 for drug1, drug2 in interaction:
```

```
136                         if sorted([drug1, drug2]) not in druglist:
137                             druglist.append(sorted([drug1, drug2]))
138                             print(f"Patient:{patient} - Prescription:{
                                    prescription} Time: {time.ctime(time.
                                    time()) } - Selecting drug parameters")
139                             details = getPrescDrugDetails(conn, prescription, drug1)
140                             drugdic[drug1] = [details]
141                             print(f"Patient:{patient} - Prescription:{
                                    prescription} Time: {time.ctime(time.
                                    time()) } - SPARQL - Selecting Tmax
                                    value")
142                             drugdic[drug1].append(getTmax(drug1,
                                    ONTOLOGY_NAME.replace('.','')))
143                             details = getPrescDrugDetails(conn, prescription, drug2)
144                             drugdic[drug2] = [details]
145                             drugdic[drug2].append(getTmax(drug2,
                                    ONTOLOGY_NAME.replace('.','')))
146                             patientPref = []
147                 print(f'Drug list: {druglist}\n')
148                 print(f'Drug details: {drugdic}\n')
149                 print(f'Patient preferences: {patientPref}\n')
150                 print(f"Patient:{patient} - Prescription:{
                        prescription} Time: {time.ctime(time.time()) }
                         - Inserting drug into Z3 model")
151                 sat, result = schedulingDrugs(druglist, drugdic, patientPref)
152                 params = (patient, prescription,sat,str(result))
153                 format_string = "Patient: {}\nPrescription number: {}\
                        nSat Schedule: {}\n Drug schedule: {}\n"
154                 # Print the formatted string with tuple values
155                 formatted_params = format_string.format(*params)
156                 print(formatted_params, end='\n')
157                 print(f"Patient:{patient} - Prescription:{
                        prescription} Time: {time.ctime(time.time()) }
                         - Inserting results into the CDSS DB")
158                 conn.execute("INSERT INTO PRESCRIPTION_RESCHEDULED
                        values (?,?,?,?)", params)
159         conn.commit()
```

F.5 SMT solver - Alternative Drugs

The SMT solver - Alternative Drugs gets the data from the Alternative solver inference engines to find valid alternative drugs for the prescription by the Z3 SMT model.

Filename: z3_alternativeDrug.py

```
1 from z3 import (Solver, And, Or, Not, sat, Datatype, Function, BoolSort, Exists, Distinct,
      Const, Xor)
2 import re
3 #import signal
4 from contextlib import contextmanager
5
```

```
6
7class Solver_obj(object):
8
9       def __init__(self, interaction = [], variables = [], alternatives = [], prescription= [],
            opt_false= [], opt_true= [], nr_prescription = int):
10              self.interaction = interaction
11              self.alternatives = alternatives
12              self.prescription = prescription
13              self.nr_prescription = nr_prescription
14              self.opt_false = opt_false
15              self.opt_true = opt_true
16              self.variables = variables
17
18
19def check_prescription( Solver_obj):
20      obj_solve = Solver_obj
21      result = False
22      if len(obj_solve.interaction) == 0:
23              return ([True])
24      else:
25              result = solve_alternatives(obj_solve)
26              return (result)
27
28
29def get_true_values(model, num):
30      characters_to_remove = "[" + "]=[==,"+ "]"
31      data_string = str(model)
32      data_string = re.sub(characters_to_remove, "", data_string)
33      split_data = data_string.split()
34      drugs = set()
35      try:
36              for i in range (num):
37                      index = split_data.index('Drug'+str(i))
38                      drugs.add( str(split_data[index+1]))
39      except Exception as e: print('err')
40      return(drugs)
41
42def remove_duplicate_sets(set_p): #remove alternative if it has inconsistency with itself,
         for example [[Aspirin, Aspirin]]
43      result = []
44      [result.append(x) for x in set_p if x not in result]
45      return(result)
46
47def solve_alternatives(self):
48      sol = Solver()
49      Drug = Datatype('Drug')
50# Inserting variables
51#Add drug names to a set drug (prescription, interaction and alternatives)
52      set_drugs = set()
53      set_alternative = set()
54      set_prescription = set()
55
56      self.interaction = remove_duplicate_sets(self.interaction)
57
58      for prel1 in self.prescription: #prescription
59              set_drugs.add(prel1)
```

```
60              set_prescription.add(prel1)
61
62      for intl1 in self.interaction: #interaction
63              for intl2 in intl1:
64                      set_drugs.add(intl2)
65
66      for altl1 in self.alternative: #alternatives
67              for altl2 in altl1:
68                      set_drugs.add(altl2)
69                      set_alternative.add(altl2)
70
71 #Declare drugs i1n Z3
72      for drug in set_drugs:
73              Drug.declare(str(drug))
74
75      Drug = Drug.create()
76      distinct_rules = []
77
78      choice = Function('choice', Drug, BoolSort())
79
80      for x in range(len(self.prescription)):
81              exec("Drug"+str(x)+" = Const('Drug"+str(x)+"', Drug)")
82              sol.assert_and_track(Exists([eval("Drug"+str(x))],choice(eval("Drug"+str(x)))
                        ), 'Exists_' + str(x))
83              sol.assert_and_track(And(choice(eval("Drug"+str(x))) == True), '
                        ExistsTrue_' + str(x))
84              distinct_rules.append(eval("Drug"+str(x)))
85      sol.assert_and_track(Distinct([x for x in distinct_rules]), 'Distinct_' + str(x))
86
87
88 #Insert true drugs on Z3 - drugs without alternatives
89      sequence = 0
90      trueDrugs = set_prescription.difference(set_alternative)
91      for drug in trueDrugs:
92              sol.assert_and_track(And(choice(eval("Drug."+str(drug))) == True), str(f'
                        TRUE_{drug}'))
93              sol.assert_and_track(Exists([eval(f"Drug{str(sequence)}")],choice(eval(f
                        "Drug.{str(drug)}" ))), f'Exists_Drug{str(sequence)} / {
                        str(drug)}')
94              sequence += 1
95
96 # Inserting interaction rules
97      for interactList in self.interaction:
98              sol.assert_and_track(Or(Not(choice(eval("Drug."+str(interactList[0])))), (Not
                        (choice(eval("Drug."+str(interactList[1]))))),str(f'NOT_{str(
                        interactList[0])}/{str(interactList[1])}'))
99
100 #Insert alternative rules on Z3
101     drugAlternaitve = [["choice(Drug."+drug+")" for drug in group] for group in
                self.alternative]
102     altSequence = 0
103     for alternative in drugAlternaitve:
104             if len(alternative) > 2:
105                     for value in alternative:
106                             altSequence += 1
107                             tempList = alternative.copy()
```

```
108                     tempList.remove(value)
109                     tempList2 = ',' .join(str(e) for e in tempList)
110                     sol.assert_and_track(Xor(eval(value), Or(eval(tempList2))), str('XOR
                            ' +str(altSequence)))
111             else:
112                 altSequence += 1
113                 sol.assert_and_track(Xor(eval(alternative[0]),eval(alternative[1])), str('
                        XOR' +str(altSequence)))
114
115
116     model = []
117     check = sol.check()
118     if check == sat:
119         sequence = 0
120         while sol.check() == sat:
121             sequence += 1
122             prescrModel = sorted(frozenset(get_true_values(sol.model(), len(self.
                        prescription))))
123             if len(self.prescription) == len(prescrModel):
124                 if prescrModel not in model:
125                     model.append(prescrModel)
126             else:
127                 break
128             solution = "False"
129             trueValues = get_true_values(sol.model(), len(self.prescription)).
                        difference(trueDrugs)
130             if trueValues:
131                 for i in trueValues:
132                     i = 'choice(Drug.'+str(i)+')'
133                     solution = f"Or(({i} != {True}), {solution})"
134                 f2 = eval(solution)
135                 sol.assert_and_track((f2), f'Models{sequence}')
136             else:
137                 break
138         #[print(sublist) for sublist in model]
139     if sol.reason_unknown() == 'canceled':
140         return (sol.reason_unknown())
141     else:
142         return(model)
```

F.6 SMT solver - Rescheduling Drugs

The SMT solver - Rescheduling Drugs, gets the data from the Rescheduling Solver inference engines to find optimised drug schedules for the prescription by the Z3 SMT model.

Filename: z3_scheduling.py

```
1 from z3 import *
2 import re
3 s = Optimize()
4 s.set("timeout", 15000)
```

```
intervalIndex = 0
intervalIndexavg = 0
drugList = []
intervalList = []
intervalAvg = []

def createDrugInstances(drugName, interval, schedule, fixedtime, Tmax):
    try:
        global drugList
        if fixedtime == 'N' or (interval != len(schedule) and fixedtime == 'F'):
            drugInterval = (24/interval)*60
            interval = interval + 1
            for x in range(1,interval):
                drugTmax = str(drugName+str(x)+'Tmax')
                drugList.append(drugTmax)
                globals()[f"{drugTmax}"] = Int(drugTmax)
                drugTmax = eval(drugTmax)
                drug = str(drugName+str(x))
                globals()[f"{drug}"] = Int(drug)
                drugList.append(drug)
                if x > 1:
                    prevDrug = str(drugName+str(x-1))
                    s.add(eval(drug) - eval(prevDrug) == drugInterval)
                s.add(eval(drug) == (drugTmax - Tmax)%1440)
                s.add(eval(drug) <= 1440,eval(drug) >= 1)
                s.add(drugTmax <= 1440, drugTmax>= 1)
                s.add_soft(eval(drug) % 120 == 0)

        elif interval == len(schedule) and fixedtime == 'F':
            schedindex = 0
            interval = interval + 1
            for x in range(1,interval):
                drugTmax = str(drugName+str(x)+'Tmax')
                drugList.append(drugTmax)
                globals()[f"{drugTmax}"] = Int(drugTmax)
                drugTmax = eval(drugTmax)
                drug = str(drugName+str(x))
                globals()[f"{drug}"] = Int(drug)
                drugList.append(drug)
                s.add(And(eval(drug) <= 1440, eval(drug) >= 1))
                s.add(drugTmax <= 1440, drugTmax>= 1)
                s.add(eval(drug) == (drugTmax - Tmax)%1440)
                s.add_soft(eval(drug) % 120 == 0)
                s.add(eval(drug) == (int(schedule[schedindex]))*60)
                schedindex += 1
    except Exception as e: print(drugName,drug, interval, schedule, fixedtime, Tmax)

def definePatientPreference(drugName, list):
    interval = int(list[0]) + 1
    for x in range(1,interval):
        drug = str(drugName+str(x))
        globals()[f"{drugName}"] = Int(drugName)
        s.add_soft(And(eval(drugName) == (list[x])*60))

def createInterval(name, freq):
    global intervalIndex
```

```
61      intervalIndex += 1
62      interval = str(name+str(intervalIndex))
63      globals()[f"{interval}"] = Int(interval)
64      interval = eval(interval)
65      s.add_soft(interval >= 720/freq, 5)
66      return(interval)
67
68
69  def createDrugRules(dicDrug, total_intervals):
70          global intervalList
71          global intervalIndexavg
72          global intervalAvg
73          drugs = list(dicDrug.keys())
74          drug1Name = drugs[0]
75          drug1Freq = dicDrug[drug1Name]
76          drug2Name = drugs[1]
77          drug2Freq = dicDrug[drug2Name]
78          if drug2Freq> drug1Freq:freq = drug2Freq
79          else: freq = drug1Freq
80          for x in range(1, int(drug1Freq)+1):
81              drug1 = eval(drug1Name+str(x)+'Tmax')
82              for i in range(1, int(drug2Freq)+1):
83                  drug2 = eval(drug2Name+str(i)+'Tmax')
84                  interval = createInterval('interval', freq)
85                  intervalList.append(interval)
86                  s.add( If( (drug1 − drug2)%1440 <= 720, interval == (drug1 − drug2)
                            %1440, interval == (drug2 − drug1)%1440))
87                  s.add(Distinct(drug1,drug2))
88
89  def schedulingDrugs(interactions, drugDetails, patientPref):
90      global s
91      global intervalIndex
92      global drugList
93      global intervalList
94      s.set("timeout", 15000)
95      drugDic = {}
96      total_intervals = 0
97      for drug, values in drugDetails.items():
98              inner_lists, tmax = values
99              #for i in inner_lists: # iterate over each nested list in the value
100             sum_of_first_values = sum(int(item[0]) for item in inner_lists)
101             # Concatenate the second values of each nested set
102             concatenated_second_values = "  ".join(item[1] for item in inner_lists)
103             last_element = inner_lists[0][2]
104             # Create a new merged list with the sum of the first values and
                    concatenated second values
105             merged_list = [sum_of_first_values, concatenated_second_values,
                    last_element]
106             # iterate over each nested list in the value
107             freq, schedule, fixedTime = merged_list
108             schedule = schedule.rstrip().split()
109             drug = drug.replace(',','')
110             createDrugInstances(drug,int(freq),schedule,fixedTime,tmax)
111             drugDic[drug] = freq
112             total_intervals = int(freq) +total_intervals
113
```

```
114     total_intervals = (24/total_intervals)*60
115
116     if patientPref:
117         for drug, values in patientPref.items():
118             definePatientPreference(drug, values)
119
120
121     for interaction in interactions:
122         interacDic = {}
123         for drug in interaction:
124             drug = drug.replace(',','')
125             interacDic[drug] = drugDic[drug]
126         createDrugRules(interacDic, total_intervals)
127     obj = Sum(intervalList)
128     s.maximize(obj)
129     checkSat = s.check()
130     m = s.model()
131     result = []
132     if len(m)> 0:
133         for drug in drugList:
134             value = int(str(m[eval(drug)]))/60
135             result.append(drug)
136             result.append(round(value))
137     s = Optimize()
138     if len(m) == 0:
139         result = 'null'
140     intervalIndex = 0
141     drugList = []
142     intervalList = []
143     return(str(checkSat), str(result))
```

APPENDIX G

CDSS DATABASE

This appendix lists the tables that compose the CDSS database.

G.1 CDSS database

```
1TABLE PRESCRIPTION_PROCESSED: (
2    CD_PATIENT,
3    CD_PRESCRIPTION
4);
```

Listing G.1: Table Prescription Processed

```
1TABLE PRESC_INTERACTION (
2    CD_PATIENT,
3    CD_PRESCRIPTION,
4    DRUG,
5    INTER_DRUG_NUM,
6    INTERACTION1,
7    INTERACTION2,
8    INTERACTION3,
9    INTERACTION4,
10   QOE,
11   SOR,
12   DETAIL,
13   RECOMMENDATION,
14   ALTERNATIVE CHAR(3)
15);
```

Listing G.2: Table Prescription Interaction

```
1TABLE DRUG_DRUG_INTERACTION (
2    CD_PATIENT,
```

```
3    CD_PRESCRIPTION,
4    DRUG1,
5    DRUG2
6);
```

Listing G.3: Table Drug-Drug Interaction

```
1TABLE DRUG_ALTERNATIVE (
2    CD_PATIENT,
3    CD_PRESCRIPTION,
4    DRUG,
5    ALTERNATIVE
6);
```

Listing G.4: Table Drug Alternative

```
1TABLE PRESCRIPTION_MODELS (
2    CD_PATIENT,
3    CD_PRESCRIPTION,
4    MODEL
5);
```

Listing G.5: Table Prescription Models

```
1TABLE PRESCRIPTION_RESCHEDULED (
2    CD_PATIENT,
3    CD_PRESCRIPTION,
4    MODEL
5);
```

Listing G.6: Table Prescription Rescheduled

```
1TABLE PATIENT (
2    CD_PATIENT,
3    AGE,
4    GENDER,
5    DT_ADMISSION,
6    DT_DISCHARGE,
7    CLINIC,
8    DISCHARGE_REASON,
9    MAIN_PROCEDURE,
10   CID
11);
```

Listing G.7: Table Patient

```
1TABLE PATIENT_EXAMS (
2    CD_PATIENT,
3    SEQ_RESULT,
4    NM_EXAM,
5    QT_RESULT,
6    DT_RESULT
7);
```

Listing G.8: Table Patient Exams

```
1TABLE PATIENT_PREVIOUS_DISEASES (
2    CD_PATIENT,
3    NM_DISEASE
4);
```

Listing G.9: Table Patient Previous Diseases

```
1TABLE PRESCRIPTION (
2    CD_PATIENT,
3    CD_PRESCRIPTION,
4    DRUG,
5    DOSE,
6    FREQUENCY,
7    SCHEDULE,
8    START_DATE,
9    END_DATE,
10   FIXEDTIME,
11   DS_INTERVAL,
12   TYPE_DRUG,
13   DRUG_UNIT,
14   DS_DRUG_ORIGINAL,
15   DRUG_LENGTH,
16   ROUTE,
17   COMPOSENAME,
18   PRESC_DAY
19);
```

Listing G.10: Table Prescription

APPENDIX H

INPUT AND OUTPUT TEST TABLE FOR THE BEERS CRITERIA

This Appendix lists the main tables used to perform validation tests. The data in the table was created based on information from the Beers Criteria table to simulate all possible results of medications classified as PIM. In the tables, we list only a sample of the data.

H.1 Input test tables

Table H.1 refers to the patient's hospitalization data, such as date of entry, exit, and diagnosis.

Table H.1: Patient Information

Patient	Gender	Age	Entry Date	Discharge Date	Procedure Description	ICD 10
111111	M	89	2022-09-26	2022-10-22	Surgical Treatment of Subdural Hematoma	I620
555555	M	75	2022-09-15	2022-09-16	Treatment of Parkinson's Disease	G20
888888	M	79	2022-12-02	2022-12-05	Treatment of Heart Failure	I500

Table H.2 refers to the illnesses the patient suffered before the hospitalisation.

Table H.2: Patient previous diseases

Patient	Disease
111111	Parkinson
555555	Delirium
888888	History of Falls

Table H.3 refers to the result of the Creatinine Clearance exam during the hospitalisation.

Table H.3: Lab Test Results

Patient	Exam Name	Result Value.	Result Date
111111	Creat. Clearance	20	2022-09-16
111111	Creat. Clearance	30	2022-09-15
555555	Creat. Clearance	20	2022-10-01
555555	Creat. Clearance	30	2022-10-02
888888	Creat. Clearance	20	2022-12-02
888888	Creat. Clearance	30	2022-12-03

Table H.4 refers to the drugs prescribed for the patient during the hospitalization.

Table H.4: Patient prescriptions

Patient ID	Start Date	Dose Unit	Schedule	Route	Frequency Time	Fixed Date	End Date	Drug Name	Dose	Critical Patient	First Line	Release Drug	Drug Length	Treatment Length
111111	16/09/22	mg	16 00 08	Nasoenteral Tube	3	N	17/09/22	Amantadine	10000	FALSE	FALSE	Immediate	1	1
555555	02/10/22	amp	05:55	Intravenous	1	F	03/10/22	Atracurium	20	FALSE	FALSE	Immediate	0	1
555555	02/10/22	amp	06:20	Intravenous	1	F	03/10/22	Atropine	0.5	FALSE	FALSE	Immediate	0	1
555555	01/10/22	mg	SN	Nasoenteral Tube	4	N	02/10/22	Captopril	625	FALSE	FALSE	Immediate	0	0
555555	02/10/22	mg	SN	Nasoenteral Tube	4	N	03/10/22	Captopril	625	FALSE	FALSE	Immediate	1	1
888888	02/12/22	mg	ACM	Oral	2	N	03/12/22	Warfarin	625	FALSE	FALSE	Immediate	0	0
888888	03/12/22	mg	18	Oral	2	N	04/12/22	Warfarin	312.5	FALSE	FALSE	Immediate	1	1
555555	02/10/22	FA	SN	Intravenous	2	N	03/10/22	Fentanyl	0.1	FALSE	FALSE	Immediate	0	1
888888	02/12/22	mg	16 00 08	Intravenous	3	N	03/12/22	Furosemide	400	FALSE	FALSE	Immediate	0	0
888888	03/12/22	mg	16 00 08	Intravenous	3	N	04/12/22	Furosemide	400	FALSE	FALSE	Immediate	1	1
111111	15/09/22	amp	SN	Intravenous	4	F	16/09/22	Glucose	150	FALSE	FALSE	Immediate	0	0
555555	01/10/22	amp	SN	Intravenous	4	F	02/10/22	Glucose	150	FALSE	FALSE	Immediate	0	0
555555	02/10/22	amp	SN	Intravenous	2	F	03/10/22	Glucose	150	FALSE	FALSE	Immediate	1	1
888888	02/12/22	amp	SN	Intravenous	2	F	03/12/22	Amiodarone	150	FALSE	FALSE	Immediate	0	0
888888	03/12/22	amp	SN	Intravenous	4	F	04/12/22	Amiodarone	150	FALSE	FALSE	Immediate	1	1
555555	01/10/22	mg	SN	Intravenous	2	N	02/10/22	Gabapentin	50	FALSE	FALSE	Immediate	0	0
555555	02/10/22	mg	SN	Intravenous	2	N	03/10/22	Gabapentin	50	FALSE	FALSE	Immediate	1	1
888888	02/12/22	mg	SN	Intravenous	2	N	03/12/22	Trimethoprim	50	FALSE	FALSE	Immediate	0	0
888888	03/12/22	mg	SN	Intravenous	2	N	04/12/22	Trimethoprim	50	FALSE	FALSE	Immediate	1	1
888888	02/12/22	mg	08	Oral	1	F	03/12/22	Metoprolol succinate	1250	FALSE	FALSE	Immediate	0	0
888888	03/12/22	mg	08	Oral	1	F	04/12/22	Metoprolol succinate	1250	FALSE	FALSE	Immediate	1	1
555555	01/10/22	mg	SN	Intravenous	8	N	02/10/22	Morphine	40	FALSE	FALSE	Immediate	0	0
888888	02/12/22	mg	08	Oral	1	N	03/12/22	Pramipexole	0.015625	FALSE	FALSE	Immediate	0	0
888888	03/12/22	mg	08	Oral	1	N	04/12/22	Pramipexole	0.015625	FALSE	FALSE	Immediate	1	1
111111	16/09/22	mg	SN	Intramuscular	1	N	17/09/22	Promethazine	1250	FALSE	FALSE	Immediate	0	1

H.2 Output test table

Table H.4 refers to the list of PIMs drugs detected by the ontology.

Table H.5: Drug Interactions

Patient ID	Prescription ID	Drug	Interaction 1	Interaction 2	Interaction 3	Interaction 4
111111	111111-89-M-20220916	Promethazine	DDDS	DDDS_Parkinson_disease	DDDS_Antiemetics	DDDS_Antiemetics
111111	111111-89-M-20220916	Promethazine	PIM	PIM_Anticholinergics	PIM_First-generation_antihistamines	PIM_First-generation_antihistamines
555555	555555-75-M-20220110	Gabapentin	DDI	DDI_Opioids/Gabapentin	DDI_Opioids/Gabapentin	DDI_Opioids/Gabapentin
555555	555555-75-M-20220110	Morphine	DDI	DDI_Opioids/Gabapentin	DDI_Opioids/Gabapentin	DDI_Opioids/Gabapentin
555555	555555-75-M-20220210	Atropine	PIM	PIM_Anticholinergics	PIM_Antispasmodics	PIM_Antispasmodics
555555	555555-75-M-20220210	Fentanyl	DDI	DDI_Opioids/Gabapentin	DDI_Opioids/Gabapentin	DDI_Opioids/Gabapentin
555555	555555-75-M-20220210	Gabapentin	DDI	DDI_Opioids/Gabapentin	DDI_Opioids/Gabapentin	DDI_Opioids/Gabapentin
888888	888888-79-M-20220212	Amiodarone	DDI	DDI_Warfarin/Amiodarone	DDI_Warfarin/Amiodarone	DDI_Warfarin/Amiodarone
888888	888888-79-M-20220212	Amiodarone	PIM	PIM_Cardiovascular	PIM_Amiodarone	PIM_Amiodarone
888888	888888-79-M-20220212	Furosemide	UWC	UWC_Diuretics	UWC_Diuretics	UWC_Diuretics
888888	888888-79-M-20220212	Warfarin	DDI	DDI_Warfarin/Amiodarone	DDI_Warfarin/Amiodarone	DDI_Warfarin/Amiodarone
888888	888888-79-M-20220312	Amiodarone	DDI	DDI_Warfarin/Amiodarone	DDI_Warfarin/Amiodarone	DDI_Warfarin/Amiodarone
888888	888888-79-M-20220312	Amiodarone	PIM	PIM_Cardiovascular	PIM_Amiodarone	PIM_Amiodarone
888888	888888-79-M-20220312	Furosemide	UWC	UWC_Diuretics	UWC_Diuretics	UWC_Diuretics
888888	888888-79-M-20220312	Warfarin	DDI	DDI_Warfarin/Amiodarone	DDI_Warfarin/Amiodarone	DDI_Warfarin/Amiodarone

APPENDIX I

HOSPITAL CDSS X FRAMEWORK CDSS - INAPPROPRIATE DRUG TABLES

This appendix provides the tables of inappropriate cases detected by our CDSS framework and by the hospital CDSS.

I.1 Hospital CDSS x Framework CDSS - inappropriate drug tables

Table I.1: Framework CDSS Inappropriate drugs

Drug	Cases	Drug	Cases
Acetazolamida	54	HidroCLOROTiazida	386
Acido_Valproico	197	Insulina_Humana_NPH	685
Alprazolam	180	Insulina_Humana_Regular	2997

Continued on next page

Table I.1 – Continued from previous page

Drug	Cases	Drug	Cases
Amilorida	4	Lorazepam	64
AmioDARONA	58	Losartana_Potassica	81
AmiTRIPtilina	859	Metadona	487
Captopril	191	Metildopa	31
CarBAMazepina	71	Metoclopramida	2999
Carbonato_de_litio	44	Midazolam	1693
Ciclobenzaprina	37	Morfina	2690
Cilostazol	21	NORTriptilina	28
Citalopram	211	Olanzapina	814
Clonazepam	1489	Oleo Mineral	265
ClorproMAZINA	561	Olmesartana_medoxomila	7
Clortalidona	22	Omeprazol	660
CloZAPina	130	Pantoprazol	2006
Codeina	612	Paroxetina	106
Dexclorfeniramina	358	Periciazina	3
Diazepam	1614	Pregabalina	204
Dimenidrato	601	Prometazina	622
DipiRONA_Sodica	174	Quetiapina,_fumarato	2344
Doxazosina	4	Risperidona	763
DULoxetina	381	Sertralina	401
Enalapril	399	Sol_Manitol	50

Continued on next page

Table I.1 – Continued from previous page

Drug	**Cases**	**Drug**	**Cases**
ESCitalopram	411	SUFentanila	4
Escopolamina	301	Sulfato_de_Magnesio	145
Espironolactona	912	Tiopental	1
Fenitoina	163	Topiramato	2
Fenobarbital	18	Tramadol	2132
FentaNILA	1371	Valsartana	49
Flunitrazepam	12	Varfarina	31
FLUoxetina	220	Venlafaxina	4
Furosemida	1667	Zolpidem	557
Gabapentina	583	Nalbufina	1

Table I.2: Hospital Inappropriate drugs

Inappropriate drugs	**Cases**
Lorazepam	10
Metadona	25
Metildopa	3
Metilprednisolona succinato	20
Midazolam	69
Nalbufina	1
Nitrofurantoina	2
NORTriptilina	3
Olanzapina	17
Oleo Mineral	53
Omeprazol	89
Pantoprazol	187
PrednisONA	51
Prometazina	67
Propafenona	1
Quetiapina, fumarato	147
Risperidona	39
Sulfato de Atropina	5
Zolpidem	50