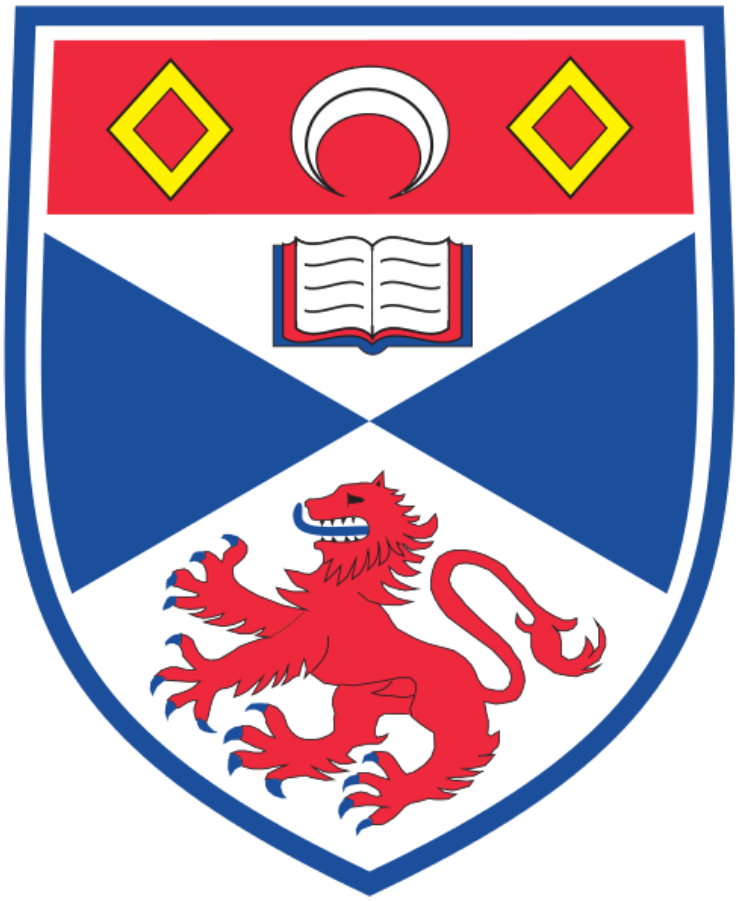


University of St. Andrews

School of Computer Science

EMR SMT: Using Constraint Programming for Cancer Data Fabrication

Austin Hatch 160026981 MSc. Computer Science Supervisor: Juliana Bowles

August 2021

Abstract

Electronic Medical Records (EMRs) are a critical part of the modern healthcare data infrastructure. EMRs contain confidential patient data protected by data privacy laws such as GDPR and HIPAA. This patient data is extremely valuable for many research purposes, especially in the context of Machine Learning approaches for targeted medicine and treatment. Simple anonymization, shuffling and encryption of patient data has been proven to be inefficient in preventing reverse extraction of the original patient data and can be incompliant with the patient privacy laws. Data fabrication provides a solution by creating completely synthetic data sets, while also allowing for enhancement and scaling for that data. This project explores the use of SMT solvers, a form of constraint solver, to model data patterns and treatment decisions as a set of constraint rules. Through the implementation and testing of *EMR SMT* , the project evaluates how rules can be extracted from input datasets, knowledge modules and simplified patient pathway input to create data that preserves the statistical generalities of input datasets, enhances the output data, and guarantees the security of the original patient data.

Acknowledgements

I would like to thank Dr. Juliana Küster Filipe Bowles for her continued help and support as a supervisor for this project, especially given the circumstances. I would like to thank Agastya Silvina and the SERUMs team for their guidance and foundational work for the research conducted in this report.

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 13,279 words long, including project specification and plan.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Contents

Chapter 1

Introduction

1.1 Background and Related Work

The use of patient data comes with ethical and privacy concerns. However, increased use of machine learning and AI methods in medical research has created a need for high-quality test data. Constraint programming can be used to fabricate test data based on metadata analysis of existing patient data alongside rules designated by a variety of sources. An example of such a system can be found in IBM's InfoSphere Optim[1].

The ability to augment existing datasets through fabrication allows for researchers to build large enough datasets for more intensive machine learning analyses and also allows researchers to synthesize necessary characteristics which may not be evident in an existing dataset.[2] The goal of the EMR SMT project was develop a platform for constructing fabricated test data based on breast cancer patient data that can accurately handle user specified rules and rules extracted from patient data sets. The platform will also look to handle constraints based on more complicated oncological relationships that may not be evident in the dataset such as stages of treatment pathways, prior cancer history, risk factors etc. This project will attempt to build on existing CSP solvers, such as IBM's, to accommodate more specific rules to more accurately model cancer patient data via the inclusion of patient pathway data rules.

1.2 Goals and Objectives

Using constraint solvers for data fabrication is not novel, but the implementation in the space of healthcare data records is still very sparse. Constraint Solvers are used frequently for software and systems testing, but there is a certain level of understanding required by a user to create constraint rules for a desired outcome. This primary goal of this project was to simplify the use of an SMT solver for data fabrication by developing a software platform which can:

1. Extract existing rules from input datasets that can be viewed and modified by a user
2. Model unique breast cancer data relationship rules based on user provided oncological principles that can be translated into constraints for a CSP solver.
3. Leverage a CSP solver to take a set of constraints from a variety of sources and create a fabricated breast cancer patient dataset based on an input dataset.
4. Implements a GUI for the solver which helps to simplify the creation of rules for a user.

These goals were met in part to deliver the required output of a completely fabricated set of output data based on the various levels of constraints. While a successful software was created, there is no single correct solution to such a problem. Instead, EMR SMT represents one example of building a CSP Solver centric data fabrication platform.

1.3 Document Structure

This report contains the following 6 chapters as well as an Appendix. All code, instructions and documentation for *EMR SMT* can be found in a GitHub repository noted in Appendix I.

1. **Introduction** - Presents the motivation, scope and goals for designing EMR SMT
2. **Context Survey -** Provides an overview of constraint solvers and their use in practice for data fabrication. Data fabrication in the context of medical records is discussed with specific focus on the inspiration for this project coming out of SERUM's use of the IBM Infosphere Data Fabrication Platform.
3. **Ethics –** A brief declaration of the ethical considerations for this project
4. **Design -** Outlines key considerations for the design of *EMR SMT* regarding desired functionality and usability within the context of the given EMR datasets and desired outcomes.
5. **Implementation -** In depth exploration of the data types, data hierarchies, objects, rule types and algorithms which comprise the *EMR SMT.*
6. **System Testing and Evaluation –** Analyze the output of a synthetic dataset compared to the input datasets.
7. **Conclusions and Future Work –** Summarize the successes and challenges of building the EMR SMT while proposing future enhancements and changes

Chapter 2

Context Survey

This chapter will introduce the technical and scientific background necessary for understanding this project. It will begin by introducing the concepts of constraint solvers (2.1.1) and their use in test data fabrication software (2.1.2). Then the current scope of test data fabrication within the domain of medical data will be explored (2.1.3).

We will then look specifically at the SERUMS project (2.2.1) and how it makes use of the IBM Data Fabrication software (2.2.2) for data fabrication, concluding with current use cases for the fabricated breast cancer patient data in predicting treatment outcomes, as well as the areas of Breast Cancer Patient Data which will be necessary to capture as constraints (2.2.3).

2.1 Background and Related Work

2.1.1 Constraint Problem Solvers

Constraint Programming is an area of computer science that focuses on solving complex sets of variable assignments using search methods. A constraint satisfaction problem (CSP) involves the definition of variables and the rules which must hold within a variable's domain, and amongst relationships of variables. Constraint Programming can utilize a variety of techniques to reason about rules that have been defined for the set of variables in the scope of the constraint satisfaction problem (Rossi, van Beek, & Walsh, 2008). Constraint solving generally utilizes one of two methods for finding a solution to a CSP – variable elimination or propagation (Fruhwirth & Abdennadher, 2003). For the purpose of this project, we will focus on the use of propagation for constraint solving. Propagation relies on a concept of local consistency to guarantee that any subset of constraints is valid with given variable assignments and implements search to guarantee the combinatorial satisfaction of all local consistency subsets, achieving global consistency (Fruhwirth & Abdennadher, 2003).

SAT solvers are an example of a solver that utilizes search and consistency. SAT solvers utilize propositional logic in conjunctive normal form to establish consistency, typically combined with a backtracking search algorithm (Kautz, Gomes, Sabharwal, & Selman, 2008). SAT solvers deal with boolean logic and a knowledge base of the domain rules to check that all constraints are satisfied in a solution. Satisfiability modulo theories, or SMT, builds on the core of a SAT solver by introducing additional complexity via formulas, or theories, written in first order logic (De Moura & Bjorner, 2011). As a result, SMTs can handle advanced concepts such as arithmetic, functions, bit-vectors, arrays, quantifiers etc. on top of the boolean logic of SAT solvers (Barrett, Kroening, & Melham, 2014).

SMT solvers such as Microsoft's Z3 have a SAT solver as the core mechanism for Boolean constraint propagation, while also implementing theory solvers and satellite solvers which handle functions, arithmetic, arrays and other complexities representable in first order logic (Moura & Bjorner, 2008).

2.1.2 SMT Solvers for Test Data Generation

Generating synthetic test data can be modelled as a CSP problem. SMT solvers in particular have been utilized to generate test data for complex model checking (Peleska, Vorobev, & Lapschies, 2011). Gotlieb et. al introduced automatic test data generation through constraint solving, using constraints to limit the domain of values within a test set to match the needs of the system (Gotlieb, Botella, & Rueher, 1998). Adorf and Verendorff demonstrated the ability to modify a generic SMT solver to generate complex test data sets for systems requiring thousands of fields (Adorf & Varendorff, 2014). IBM's InfoSphere Optim Test Data Fabrication software also implements a CSP solver as the engine for data fabrication (Figure 2). This tool has been used to fabricate breast cancer patient data based on extracted constraints from patient data relationships, as well as user defined and clinical data (Bowles, Silvina, Bin, & Vinov, 2020). This will be discussed further in Section 2.2.

2.1.3 Health Record Data Fabrication

Accurate and effective datasets are crucial for modern research, applications and analysis, particularly machine learning. However, laws surrounding data privacy restricts the usage of human data in many instances, especially in healthcare under HIPAA and GDPR regulations.

While simplistic methods such as data shuffling or omitting sensitive fields have been attempted, there are still ways to re-identify individuals when data is combined with information from other datasets (Goncalves, et al., 2020). For healthcare data in particular, the increased use of electronic healthcare records (EHR) and increased amount of patient data collected provides an excellent opportunity to leverage data for research, education and patient care. Patient data fabrication provides a way to generate synthetic patient records which comply with data protection laws (Choi, et al., 2017). Synthetic test data generation also allows for augmentation of datasets with smaller sample sizes, creating large enough inputs for machine learning purposes, but also representing important medical or scientific concepts that may not be fully represented in an existing dataset (Bowles, Silvina, Bin, & Vinov, 2020).

EMR data synthesis has been explored and evaluated using a range of artificial intelligence techniques including linear regression, decision trees and neural networks (Gonclaves, et al., 2020).

Tucker et. al (2020) developed a methodology of capturing complex data relationships from the United Kingdom's CRPD patient database in a Bayesian Network, then utilizing those network dependencies to generate a synthetic dataset (Tucker, Wang, Rotalinti, & Myles, 2020). Choi et. al proposed a methodology of using Generative Adversarial Networks (GAN) in their system *medGan*. Their AI system generates synthetic data and continuously tests the feasibility of the generated data compared to real data, allowing the generator to learn the appropriate distribution over time (Choi, et al., 2017). While this approach only relies on real patient data for checking how realistic the test data is, platforms such as PADARASER attempt to generate realistic, synthetic EHRs without ever using patient level data (Dube & Gallagher, 2013). Technologies, such as *Synthea*, leverage this framework to generate synthetic EHRs based exclusively on general healthcare statistics and clinical standards, eliminating risks of using any individual patient information (Walonoski, et al., 2018). Scalfani and Bhada were able to successfully leverage the open-source Synthea platform to develop a model of analyzing breast cancer patient survivorship based on health insurance loss of service (Scalfani & Bhada, 2020).

Despite the successful use of Synthea to build synthetic patient datasets, limits in modeling health outcomes have been expressed (Chen, Chun, Patel, Chiang, & James, 2019). Likewise, concerns over the inability for the popular GAN method of generation to capture additional constraints that are unseen by the GAN network and therefore not involved in the adversarial

algorithm (Yan, Zhang, Nyemba, & Malin). Yan et. al (2021) have begun work on augmenting the GAN framework to include feature constraints across data sources, implementing a constraint violation penalty in the adversarial acceptance algorithm.

The CareMap project also looks to enhance synthetic EHR generation by creating a constraint model for the desired EHR domain and executing generation via a State Transition Machine (McLachlan, Dube, & Gallagher, 2016). Existing projects in the synthetic EHR generation space demonstrate the power of computing in achieving domain specific datasets which can protect patient data and enhance machine learning applications. However, there are limitations and complexities within current methodologies which we will hope to explore and alleviate with this project.

2.2 Context

2.2.1 SERUMS Overview

The SERUMS tool chain looks to build a platform for the future of EHR storage, access and analysis with patient privacy and protection at the core (Janic, et al., 2019). SERUMS implements a suite of technologies including a blockchain database for user permissions, deep learning for diagnosis and outcome predictions, data encryption and masking as well as integrating data from multiple users and sources to form a *Smart Patient Record* (Figure 1). The SERUMS project utilizes the IBM Infosphere Data Fabrication Platform (DFP) to create data for system development and testing, while preserving patient privacy. Use of the CSP solver based platform helps to create data that fits the complex format of a *Smart Patient Record*. The DFP also extracts constraint rules and relationships from input datasets, allowing for additional user rules to be included in the CSP, all while maintaining database requirements within the complex SERUMS toolchain.

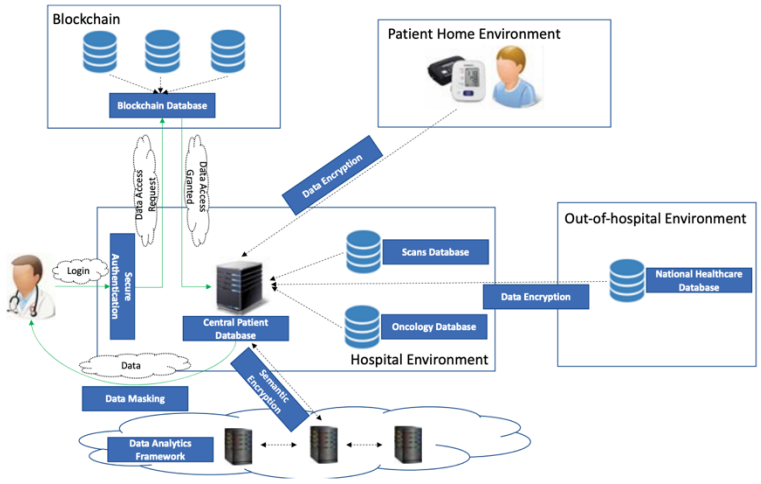


Figure 1: Overview of the SERUMS System. Originally published as "Figure 1. The overview of SERUMS" on pg. 28 of (Janic, et al., 2019)

2.2.2 IBM Data Fabrication Platform

The IBM Infosphere Optim Test Data Fabrication Platform described as part of the SERUMS project utilizes an CSP solver at its core (IBM InfoSphere Optim Test Data Fabrication, 2020). The DFP handles rules from users of different types, namely:

- **Constraint rules** – domain restrictions, mathematical and arithmetic relations, string relations etc.
- **Knowledge base rules** – rules derived from existing data sources
- **Analytics rules** – classifications, value, and pattern distributions
- **Declarative Transformation rules** – constraints on relationships between targets and sources
- **Programmatic rules** – code/ script functions for generating target values.

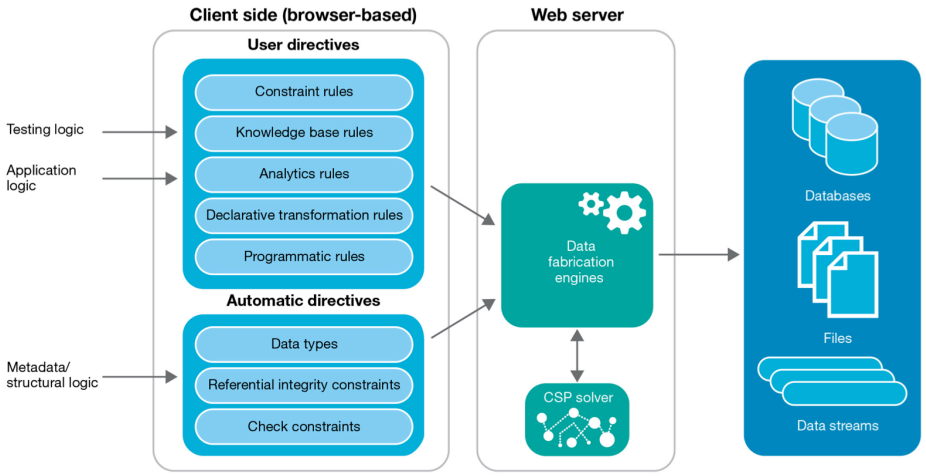


Figure 2: Overview of the IBM InfoSphere Optim Test Data Fabrication Platform. Originally published as "Figure 2. The InfoSphere Optim Test Data Fabrication workflow." in (IBM InfoSphere Optim Test Data Fabrication, 2020)

Using the set of rules provided by the user as well as constraints automatically derived from the input dataset, the DFP uses its SMT solver to build a synthetic dataset which satisfies all of the constraints (Bowles, Silvina, Bin, & Vinov, 2020).

2.2.3 Fabricating Breast Cancer Patient Data

Bowles et. al (2020) demonstrated how the tool could be used not only to generate test data for SERUMS, but to generate synthetic patient data for breast cancer patients that can be used in machine learning applications. Using the DFP, they were able to formulate rules for generating treatment data for breast cancer patients based on data from the Scottish health board as well as domain specific knowledge, while preserving important relations between different treatment row entries for the same patient (Bowles, Silvina, Bin, & Vinov, 2020). This project will look to build on this existing framework of SMT solvers for cancer patient treatment data, while finding ways to handle more complex, domain specific rules and implementing tests to verify the accuracy of rules in creating a realistic dataset.

The use of the synthetic cancer patient treatment data can play a key role in predicting patient outcomes based on treatments. Machine learning techniques for predicting toxicity outcomes for breast cancer patients have been explored (Silvina, Bowles, & Hall, 2019). Silvina, Bowles and Hall (2019) concluded that the accuracy of toxicity prediction modelling can be

improved with additional information surrounding comorbidities, cancer stages (TNM), breast cancer hormone receptor characteristics etc. This project will further explore the oncological and pharmaceutical principles that may affect treatment toxicity and determine how to capture and evaluate these principles as constraints for data fabrication.

The input datasets used for creating constraints, including those used for domain knowledge, will be discussed further in Section 4.2.

Chapter 3

Ethics

Working with patient data, particularly EMRs, is subject to a high level of ethical consideration. In the context of this project however, all data used is synthetically generated in compliance with appropriate patient privacy standards. As such, for the scope of this report there are no additional ethical considerations needed.

Chapter 4

Design

This chapter will cover the design considerations used to create EMR SMT including the critical functions needed, evaluation of input and required output, user considerations and finally the proposed design of the system.

4.1 System Design and Goals

The design of the EMR SMT focuses on four major tasks which are essential to the EMR fabrication process:

1. **Data Import**: How does existing patient data get input into the system and what are the properties of this data?
2. **Data Structuring:** How can we create Object classes in Java which will allow us to represent the data hierarchy within the EMR system, as well as successfully use the imported data for future tasks.
3. **Rule Generation and Import:** How do we extract patterns from the imported datasets and understand rules prevalent to specific data types? How can we take rules from external sources and apply them to the existent data structure?
4. **Constraint Solving and Data Fabrication:** How can we translate rules objects into logic that is executable in the Z3 SMT environment? What is the process flow that allows us to utilize our imported and rules to build synthetic data objects that satisfy rules at all levels of the data hierarchy?

The process flow below shows the movement of data within the system and where the tasks above fall within the process.

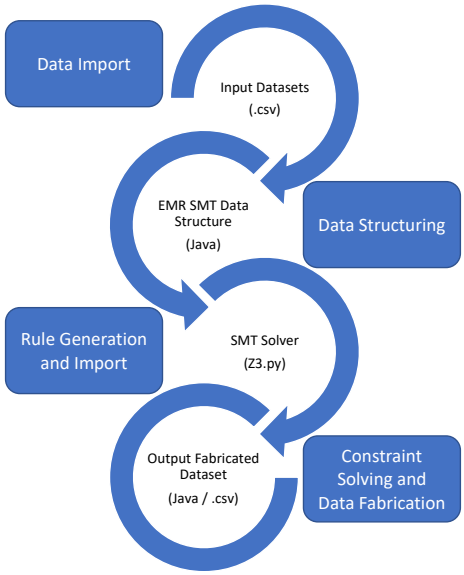


Figure 3: Data flow for EMR SMT with system tasks.

4.2 Data Import

The *EMR SMT* software is designed to read in multiple CSV files containing patient data, which all have a different set of patient IDs (CHI numbers), different Fields and a different number of rows. The datasets each contain unique data relationships which are critical to maintaining information about the patient's treatment pathway. For the purposes of this project, the solution was modelled to handle a series of datasets that are synthetic representations of the NHS Scotland's SMR datasets, as well as additional datasets which capture important patient information regarding their cancer treatment pathways.

The datasets used for this project can be found in the *EMR_SMT/Test Input Datasets* folder (Appendix A). A brief list and description can be found below:

1. **SMR01** → Fabricated representation of the NHS Scotland's "General / Acute Inpatient and Day Case" database, filtered for breast cancer patients. [1]
2. **SMR06** → Fabricated representation of the NHS Scotland's "Scottish Cancer Registry" database, filtered for breast cancer patients. [2]
3. **General** → Contains patient personal detail, demographic information, general cancer histology, comorbidities (Charleston Score), hospital and GP information. This serves as the definitive list of Patient EMRs to be used throughout the system.
4. **Patients** → A list of Patient CHIs, initial treatment intention and initial appointment date in the NHS system
5. **Intentions** → A list of treatment intentions for each Patient. Includes the number of treatment regimens and cycles for the treatment intention as well as the initial regimen and initial appointment date for that treatment intention.
6. **Regimes** → A list of regimes for chemotherapy treatments. Contains the regime type, number of cycles, intention,
7. **Cycles** → Captures the treatment cycles of a chemotherapy regime. It includes the treatment type, dosage and recorded side effects.

The data from the import datasets is a combination of quantitative and qualitative fields. Defining the scope of field types is necessary for this project because the generation and handling of constraint rules varies based on field type, and the scripting needed for Z3 is dependent on the type of data. This will be discussed further in the proceeding sections of this report. The data used in this project was categorized into 4 major types:

Type	Example
Integer	general.age
Double	cycles.required_doses
Date	smr01.admission_date
String	patients.first_intention

Table 1: Field Types in the EMR SMT based on input requirements

[1] https://www.ndc.scot.nhs.uk/Data-Dictionary/SMR-Datasets/SMR01-General-Acute-Inpatient-and-Day-Case/

[2] https://www.ndc.scot.nhs.uk/National-Datasets/data.asp?SubID=8

While this set of types was satisfactory for the scope of this project, it can certainly be expanded. Additions of Boolean, Binary, Categorical etc. could all be implemented to adjust constraint creation and handling.

4.2.1 SMR Datasets

The SMR01 and SMR06 datasets used for this project were synthetic generated datasets of the Scottish Morbidity Record (SMR) datasets that were generated to only include patient entries with breast cancer as their main condition.

The SMR01 dataset is the General/ Acute Inpatient and Day Case dataset. The dataset captures the occurrence of a day case for a patient, as well as the condition being treated and the treatment administered.[3] It contains the following fields:

Field	Type	Description
smr01.chi	Integer	CHI Number for the Patient
smr01.incedence_date	Date	Date of diagnosis (Breast Cancer in this case)
smr01.admission_date	Date	Date of admittance to the hospital/ place of care
smr01.length_of_stay	Integer	Duration of stay at hospital/ place of care
smr01.other_condition1	String	ICD-10 Code for Other Condition besides C509
smr01.other_condition2	String	ICD-10 Code for Other Condition besides C509
smr01.other_condition3	String	ICD-10 Code for Other Condition besides C509
smr01.main_operation_b	String	ICD-10 Code for main operation b
smr01.discharge_date	Date	Date of discharge from hospital/ place of care
smr01.waiting_list_type	Integer	Enumerated value for waiting list type
smr01.main_condition	String	ICD-10 Code for Main Condition (C509)
smr01.main_operation_a	String	ICD-10 Code for main operation a
smr01.marital_status	String	Encoded value for Marital Status
smr01.ethnic_group	String	Encoded value for Ethnic Group

Table 2: Field Dictionary for SMR01 Dataset

[3] https://www.ndc.scot.nhs.uk/Dictionary-A-Z/Definitions/index.asp?Search=S&ID=997&Title=SMR01%20-%20Summary%20of%20Rules

The SMR06 dataset is the Scottish Cancer Registry. It is used to capture information on every new diagnosis of cancer in Scottish residents, dating back to 1958.[4] The synthetic SMR06 dataset used for this project contains the following fields:

Field	Type	Description
smr06.index	Integer	Index number for the entry in the database
smr06.chi	Integer	CHI number for the patient
smr06.incidence_date	Date	Initial date of diagnosis
smr06.er_status	Integer	Oestrogen Receptor (ER) Status (0=negative, 1=Positive)
smr06.her2_status	Integer	Human Epidermal Growth Factor Receptor (HER2) Status (0=negative, 1=Positive)
smr06.stage_clinical_t	String	Staging for Primary Tumor
smr06.stage_clinical_n	String	Staging for Regional Lymph Nodes
smr06.stage_clinical_m	String	Staging for Distant Metastasis
smr06.num_positive	Integer	Number of Positive Regional Lymph Nodes
smr06.pathological_tum_size	Integer	Maximum tumor diameter size (mm)

Table 3: Field Dictionary for SMR06 Dataset

4.2.2 Regimes, Cycles and Intentions

The Regimes, Cycles and Intentions datasets work in combination to capture the chemotherapy treatment pathways for patients using a structure first outlined by Bowles et. al (2020). In essence, a patient is treated with on Intention at a time, given their progress within a treatment pathway. This is either:

- Neo-Adjuvant → Therapy delivered before the main treatment or surgery to reduce tumor size or kill cancer cells.

[4] https://www.ndc.scot.nhs.uk/National-Datasets/data.asp?SubID=8

- Adjuvant → Therapy delivered after the main treatment or surgery to reduce tumor size or kill cancer cells.
- Palliative → Therapy to manage/ reduce symptoms given at any stage of the treatment pathway

A patient being treated with a given intention will be treated using several regimens to achieve that given intention. Each regimen is comprised of one or more cycles of treatment which represents the length of treatment (weeks or months), often with an interval break between.

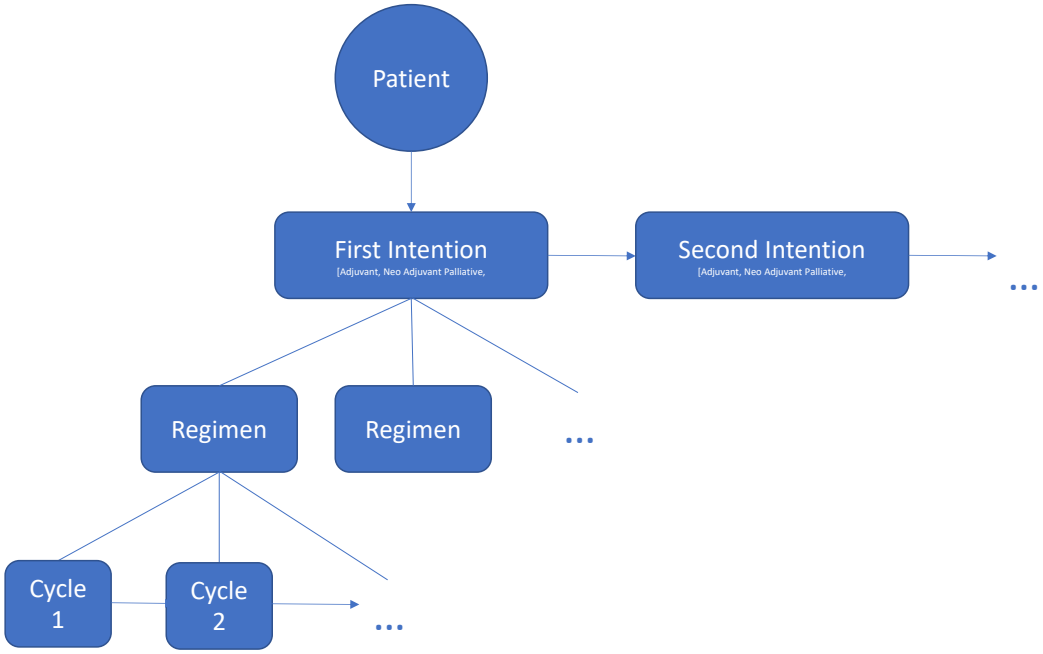


Figure 4: The Relationship Between Patients, Intentions, Regimens and Cycle

The structure depicted in Figure 2 is preserved across the datasets by using *regime_id*, *intention_id* and *cycle_id* fields. There are also fields for *regime_ratio* and *cycle_ratio* which capture the appropriate number of regimens for an intention and the appropriate number of cycles for a regimen. This hierarchy is necessary to be maintained throughout the fabrication process. The rules which define this structure, initially an output of Bowles et. al (2020), were extracted and used to demonstrate a rule import package found in

Test_Import_Rule_Sets/Cycles-Regimes-Intentions.dfproj. The use of this rule set will be further discussed in sections 4.4.2 and 5.4.4.

4.2.3 Clinical, Oncological and Pharmaceutical Data

To better capture complex relationships between patient attributes, cancer properties and treatment pathways, this project looked to build additional rules in the system based on guidelines for breast cancer patient treatment published by leading organizations. This project made use of the following resources:

Organization	Resource Name	URL
NCCN (National Comprehensive Cancer Network)	NCCN Guidelines Version 5.2021 Breast Cancer	https://www.nccn.org/professionals/physician_gls/pdf/breast.pdf
NICE (National Institute for Healthcare and Excellence)	Early and Locally Advanced Breast Cancer Overview	https://pathways.nice.org.uk/pathways/early-and-locally-advanced-breast-cancer
NICE (National Institute for Healthcare and Excellence)	Advanced Breast Cancer Overview	https://pathways.nice.org.uk/pathways/advanced-breast-cancer
ESMO (European Society for Medical Oncology)	Early Breast Cancer ESMO Clinical Practice Guidelines for diagnosis, treatment and follow-up	https://www.esmo.org/content/download/284512/5623447/1/Clinical-Practice-Guidelines-Slideset-Early-Breast-Cancer.pdf
ESMO (European Society for Medical Oncology)	5th ESO-ESMO International Consensus Guidelines for Advanced Breast Cancer	https://www.esmo.org/content/download/395915/7703117/1/Clinical%20Practice%20Guidelines%20Slideset%20Advanced%20Breast%20Cancer.pdf

Table 4: Clinical Treatment Pathway Decision Trees

The treatment pathways presented by each of the resources offer the organization's recommended patient treatments based on characteristics of the patient, tumor histology and attributes, prior treatment regimens etc. This information can be interpreted as a "loose" constraint for how an EMR should represent the relationship between treatments and patient demographics, comorbidities, and tumor attributes. The pathways also help to formalize the timeline of patient care from diagnosis, to neo-adjuvant therapies, to surgery, to adjuvant therapies, to continued evaluation.

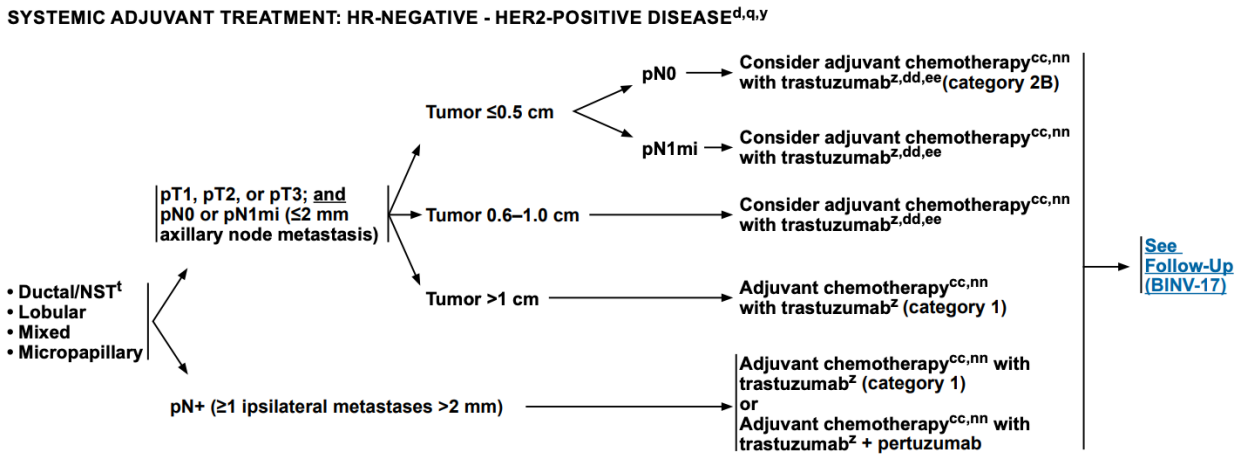


Figure 5: Example Patient Pathway for Adjuvant Treatment of HR-/ HER+ Breast Cancer. Originally appears as BINV-9 in NCCN Guidelines Version 5.2021 Breast Cancer (2021)

Further information about the extraction of rules from Patient Pathways will be discussed in section 4.4 and 5.4.

4.2 Data Structuring

Importing and manipulating data within the *EMR SMT* system requires a defined data structure which allows for rule creation and data fabrication to be partitioned across different levels of the data hierarchy. Since a patient's EMR is evaluated wholistically, and constraints may be applied to information within and among datasets, the structure must maintain the original import dataset designations, while also allow for parsing of information from other datasets. As such, the general structure of the data within the EMR System is as follows:

- **DataBase** → Maintains a collection of all DataSets and EMRs within the system
- **DataSet** → Maintains a collection of all of the Fields in the DataSet as well as the Patients in the DataSet
- **DataSetEntry** → An entry (row) for a patient in a given DataSet that maintains a map of Fields and Values
- **EMR** → Maintains a collection of all of the DataSetEntry objects for a patient
- **Field** → Represents a column of data from a DataSet
- **Value** → Represents a datapoint in a column from a DataSet

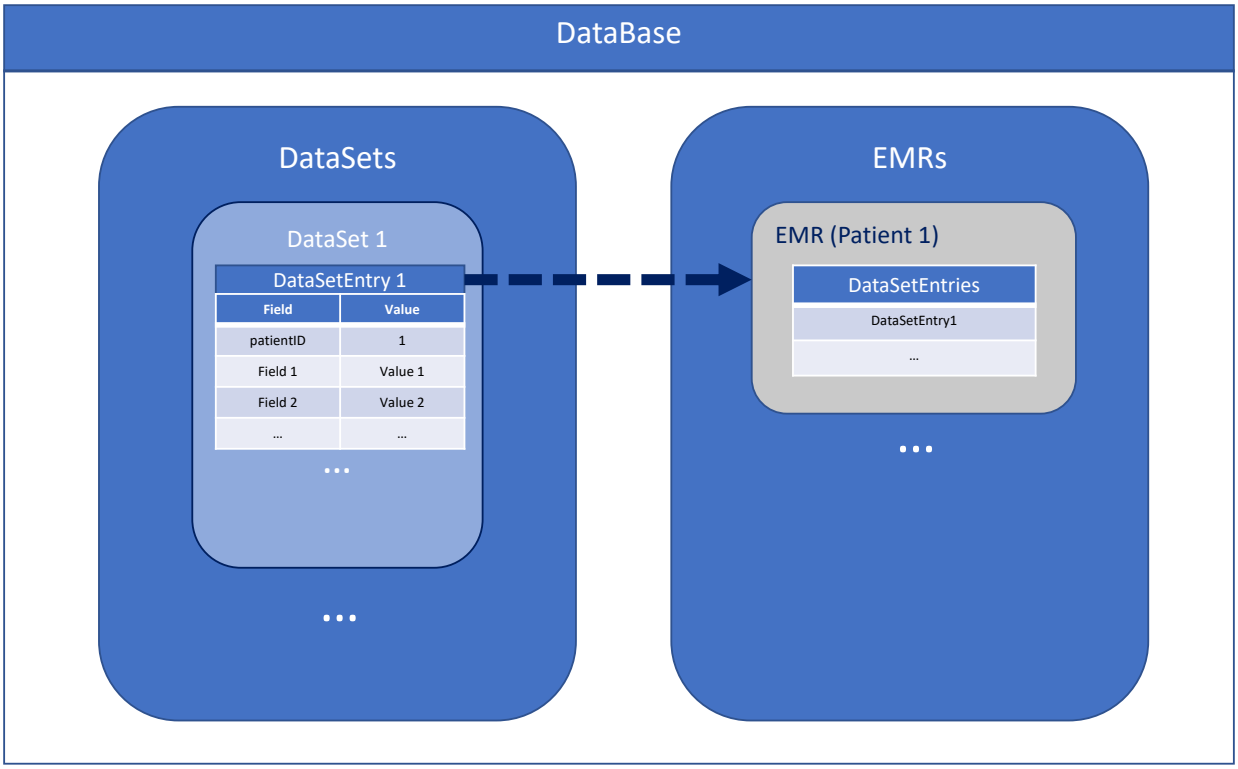


Figure 6: Concept Data Structure within the EMR SMT system

Under this proposed structure, EMRs can have multiple entries per dataset, or none at all. This is especially important given that patients can show up in the input datasets zero, once or multiple times. Table 5 below shows the number of unique patients, number of entries and entries per EMR range for each dataset. This will be a critical aspect to solve for in our implementation of EMR SMT.

Dataset	Number of Patients	Number of Entries	Entries per EMR
general	3000	3000	1
patients	3000	3000	1
smr01	776	4573	0 to 45
smr06	933	1034	0 to 3
intentions	3000	3087	0 to 2
regimes	2869	5229	0 to 5
cycles	2871	43916	0 to 51

Table 5: Number of Patients, Entries and Entries per Patient in Input Datasets

For each DataSetEntry, there is a map of Fields to Values. One important consideration for this project is that input data will not be perfect and there may be missing values for an entry. These missing values will need to be handled, and for the scope of this project, data imputation will not be employed. As such, the inclusion of blank values should be maintained in the implementation.

4.3 Rule Generation and Import

Modelling the input datasets and resulting data structure as a series of constraints requires the following types of rules as laid out in Section 2.2.2:

- **Constraint rules** – domain restrictions, mathematical and arithmetic relations, string relations etc.
- **Knowledge base rules** – rules derived from existing data sources
- **Analytics rules** – classifications, value, and pattern distributions
- **Declarative Transformation rules** – constraints on relationships between targets and sources
- **Programmatic rules** – code/ script functions for generating target values.

Rules could be applied at any level of the data structure. For example, a rule can define the domain of a Value, whether the values of a Field are all unique and how many Dataset

Entries a patient has in a dataset. Rule types within the EMR SMT fall into the following buckets to capture the scope of the constraint:

1. **Value Rules** → Rules applied directly to a value. Includes the domain, distribution, and any conditional distributions. Handles the generation of values.
2. **Field Rules** → Rules applied across all values of a given field. Includes all different, monotonic sequences and equating two Fields.
3. **Record Rules** → Rules applied to an EMRs data to capture unique relationships among and across dataset entries.
4. **Dataset Rules** → Rules that define which EMRs must be included in a given dataset, how many times an EMR will appear in a dataset etc.

Based on an output rules set from the IBM Infosphere tool as part of the work done by Bowles et. al (2020), a baseline for the general rules for breast cancer dataset were defined. This file can be found in *EMR_SMT/Resources/CHEMO_REDO_WITH_VIRTUAL.dfproj*. The rules defined in the file contained a set of operators used to build rules which will need to be recreated or solved for in the EMR SMT.

Operation Type	Mechanism	Rule Level
allDiff	All values in a given field are unique	Field, Record
numOf	The number of values for a certain Field in an EMR	Dataset
<=, =, >=	Defines the bounds/ domain of numerical and date values	Value
+, - , *, %	Defines the arithmetic relations between Fields in a Record	Record
concat	Defines the concatenation of String values	Value, Field, Record
reuseData	Sets two Fields in different datasets to use the same Values	Field
randomWeightedValue	Defines a weighted distribution for value assignment	Value
monotonic	Values for a field are monotonically increasing/ decreasing by a given step size	Field
lastTableValue	Finds an occurrence of a value that has been previously assigned in an EMR	Record

normalDistribution	Defines a normal distribution for value assignment	Value
? _ : _	Defines a conditional assignment of value based on another	Value
randomNumber	Defines a range for random value assignment	Value
-> (Implies)	Defines a required value assignment based on a different Field	Value
items	Applies constraint to a group of items. Used to create relationships across more than 2 fields	Record
&&, ||	And, Or conditionals	All
==, is, isNot	Equals or Not Equals conditionals	Field, Value

Table 6: Operator Types Derived from IBM Infosphere Data Fabrication Rules Set

While the rule operators can be derived from the IBM Tool rules file, the execution of such rules, translation to an SMT language, and order of rule execution to achieve fabrication is unknown. These will all be decisions and tasks for implementation within the EMR SMT data structure.

The source of rules for this project fell into three major buckets: rules generated automatically from datasets via the EMR SMT platform, rules imported from XML files that could be the output of the IBM Infosphere tool and rules designed via a user – either through CSV import of pathways or via a simple GUI design. The following sections will describe the design considerations for utilizing each of these rule creation sources.

4.3.1. Extracted Rules

One major benefit for a user of the EMR SMT platform is the automatic creation of constraint rules from input data. While it is difficult to automatically extract rule relationships between columns specific to cancer patients without creating a restrictive platform, analysis of input data can help create the foundation for data fabrication.

To preserve the statistical patterns of the input data in the fabricated data output, rules should be created that enforce a similar distribution of data. The fabrication process will rely on generating random variable assignments and ensuring those assignments satisfy the constraint rules for the dataset. Value rules within the system will handle defining the domain of possible value assignments, as well as the distribution for random assignment. The system offers 5 types of automatically extracted value rules based on the domain of values extracted from the input data:

1. **Equals →** If the domain of a value in the input dataset is a single value, all values will be that value or blank.
2. **Range →** Defines the bounds of the range for integer, double and date values based on minimum and maximum values from the input dataset.
3. **Random Number →** The value is a random number within the range, with no distribution.
4. **Normal Distribution →** The value is a random sample from a normal distribution based on the mean and variance of input data.
5. **Weighted Distribution →** The value assignment is based on a weighted distribution calculated by frequency in the input data. This is used for fields with a domain of 15 or fewer values.
6. **Conditional Weighted Distribution →** The value assignment is based on a weighted distribution, calculated conditionally based on the assignment of another variable. This is used for fields with a domain of 15 or fewer values conditional on a second field with 15 or fewer values.
7. **Conditional Normal Distribution →** The value is a random assignment from a normal distribution based on a sample of values given the assignment of another variable in the input dataset. This is used for fields with a large domain conditional on a second field with 15 or fewer values.

Input data analysis can also help to extract field rules for patterns among the value assignments in an input column. The field rules that are extracted automatically using the software are:

1. **Equals →** If the field name and values for each entry are equal across two fields, the fields are equal and values for those two fields should be equal for all dataset entries.
2. **All Different →** If all non-blank values in an input column are unique, and the Field is not a double type, values in the fabricated data should be all different.
3. **Monotonic →** If all values in an integer column are increasing by a consistent step, this pattern should be reflected in the fabricated dataset.

Lastly, DataSet rules are extracted automatically in the EMR SMT system to replicate the frequency of entries per EMR reflect the size of the original dataset relative to the new patient base. The DataSet rules extracted are:

1. **Number of Entries** → If there is a set number of entries per patient in the input dataset, this rule will ensure the same number of entries applies in the fabricated dataset.
2. **Range of Entries** → If the number of entries per patient is variable in the input dataset, this rule will randomly select a number of entries for each patient to be included.
3. **Weighted Number of Entries** → This rule will select a number of entries per patient that is weighted based on the frequency table of number of entries in the input dataset. This is preferred to Range of Entries in the case that no other rule is proposed.

Automatically generated rules should be presented to the user, allowing them to remove any rules that do not apply. Similarly, rules should only be created when they are applicable, and there should be a natural hierarchy so that not every applicable rule is created in case of clashes. The programmatic selection process for automatic rule creation is discussed further in section 5.4.3.

4.3.2 Imported Rules

Automatically generated rules allow for users to not have to focus on capturing the statistical model of the datasets in rule creation, as this can be time consuming. However, the user should be able to offer contextual rules based on the nature of cancer patient data that is not extractable via generated rules. The system should therefore allow for users to create rules based on knowledge of relationships within the data, or import rules generated from other sources. Since this project looks to build off of the work done in IBM's Infosphere platform, the EMR SMT platform was designed to accept and parse rules from the XML format of the *.dfproj* files. To do so, being able to parse the XML file, retrieve the applicable Fields for the rule, and parse operators to form an appropriate constraint rule for the appropriate level (value, field, record etc.) must be implemented.

For the purpose of this project, critical rules related to the input datasets from the provided *CHEMO_REDO_WITH_VIRTUAL.dfproj* were broken up into separate files, designated rules packages, based on function.

The rules packages created can be found in *EMR_SMT/import_rules_packages* and include the following unique rules sets created by Bowles et. al (2020):

- *CHI.dfproj* → contains rules for valid CHI creations (Replicated in SMTSolver class)
- *General.dfproj* → contains rules for accurate date relations, healthcare information that would not be captured by automatically generated rules
- *Cancer.dfproj* → contains rules for basic cancer logic (ex. a second metastasis cannot occur without a first) and domain specific rules (ex. If a cancer has metastasized it is a CCI index score of 6, if not it is a 2).
- *Cycles-Regimes-Intentions.dfproj* → Contains the rules for maintaining the treatment structure of the dataset outlined in Figure 3.

In practice, these sets of rules could either be defaults for the system, or importable packages provided to scientists/ researchers by individuals who understand the unique structures of their data, particularly in the way that cycles, regimens, and intentions are recorded in the system. Additional work can be done to create a simplified user interface for rule creation which does not require an intimate understanding of constraints. This concept has been explored in the IBM Infosphere Tool as illustrated in their user manual.

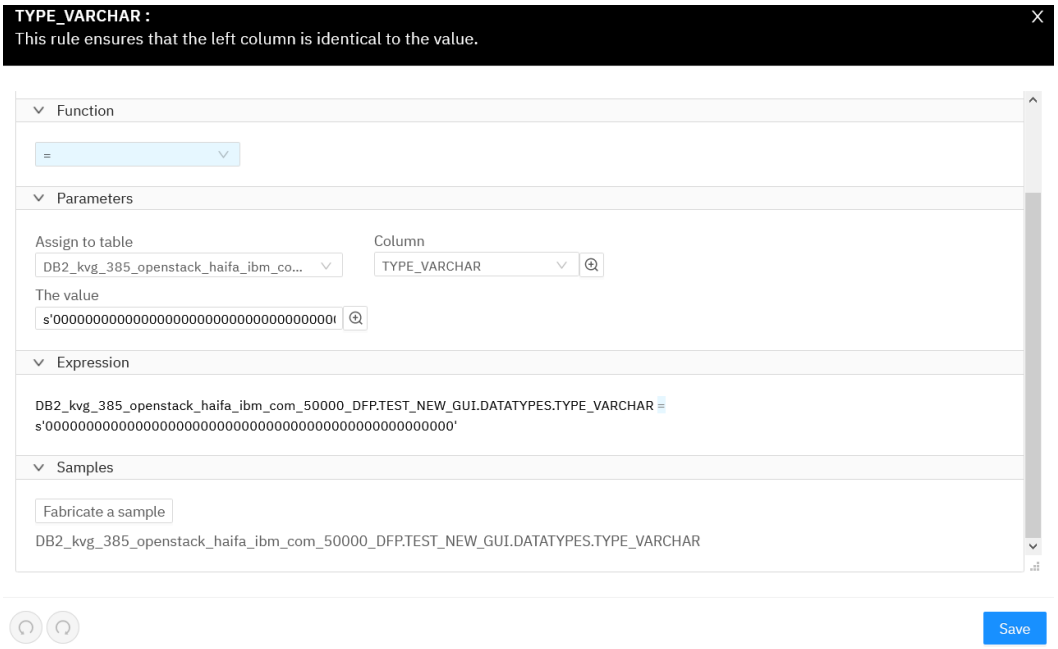


Figure 7: Example Rule Creation Window from IBM InfoSphere Optim Platform. Originally published on pg. 30 of "IBM InfoSphere Optim Test Data Fabrication User Manual"

One important dimension for imported rules which needed to be considered was the use of auxiliary fields. For example, the creation of a CHI index in the general database is a concatenation of 10 digits, the first 6 of which are the date of birth of the patient, and the 9th of which depends on the gender of the patient.

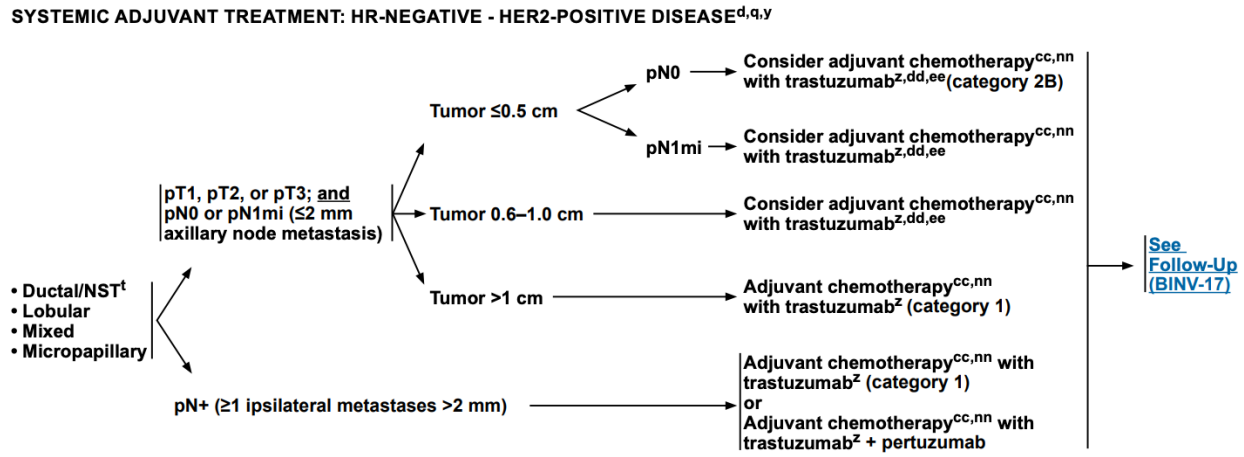
4.3.3 Patient Pathway Rules

One major contribution of this project is defining novel rules to build upon the already existing rules provided. The automatically generated rules maintain proper value assignments and dataset composition to uphold statistical properties, while the imported rules help define the relations within the data that are not easily extractable. The addition of Patient Pathway Rules looks to add underlying insight to patterns that may be deeper than the datapoints provided, which can help reason about a patient's treatment schedule, options, and outcomes. As a result, fabricated data based on pathway constraints could more accurately represent patient experience than just extracted and imported rules.

For example, extracted rules will give a patient certain histological properties based on frequency distributions from the input data, as well as generate an. A generated patient may be HER2+ and ER- in smr06, and a separate set of extracted rules will give that patient an entry in smr01 with a main operation indicating a mastectomy.

Patient pathway rules can help logically create time series data for cycles, regimes and intentions. As in the basis example above, we can formalize that a patient who has a mastectomy or breast conserving surgery listed in SMR01, and has a regimen for chemotherapy that is Neo-Adjuvant in Regimes, that regimen must occur on a date greater than the date of the surgery + any recommended buffer time. Likewise, pathway rules can help to guarantee the logical application of certain drug types to certain patients based on health factors.

Three sets of patient pathway rules sets were developed for this project, based on the guidelines from the NCCN, ESMO and NICE resources outlined in table. A complete set of rules extracted from these pathway resources can be found in Appendix 2. Patient pathway rules are designed to be a collapsed version of the decision tree format presented in each guideline resource. For example, the tree in Figure 4 from NCCN, recreated below:



This figure demonstrates Adjuvant Treatment for HR- / HER2+ tumors with histology Ductal, Lobular, Mixed, Micropapillary. To break down this tree into pathways, we can follow each of the branches from the root and develop the pathway rules given the fields presented using simple logical operators of ==, <=, >=, <, >, |, &. Each pathway starts with the logical expression of:

AND((HER2 == +), (HR == -), (Histology = Ductal | Lobular | Mixed | Microcapillary))

This indicates that the patient is HER2+, HR- and has a histology of ductal, lobular, mixed or micropapillary. This is the root of all pathways in figure. The tree above has 5 terminal nodes which leads to the 5 separate pathways:

1. *AND(OR(AND(TStage = 1 | 2 | 3), (NStage == 0)), (NStage == 1mi)), (Size <= 0.5), (NStage == 0), AND(ADJUVANT, trastuzumab == YES, required == NO))*
2. *AND(OR(AND(TStage = 1 | 2 | 3), (NStage == 0)), (NStage == 1mi)), (Size <= 0.5), (NStage == 1mi), AND(ADJUVANT, trastuzumab == YES, required == NO))*
3. *AND(OR(AND(TStage = 1 | 2 | 3), (NStage == 0)), (NStage == 1mi)), AND(Size <= 1, Size >= 0.6), AND(ADJUVANT, trastuzumab == YES, required == NO))*
4. *AND(OR(AND(TStage = 1 | 2 | 3), (NStage == 0)), (NStage == 1mi)), (Size > 1), AND(ADJUVANT, trastuzumab == YES, required == YES))*
5. *AND((NStage >= 1)), OR(AND(ADJUVANT, trastuzumab == YES, required == YES), AND(ADJUVANT, trastuzumab == YES, pertuzumab == YES, required == YES))*

The pathway rules are conjunctions of individual rules for Field such as the NStage of the tumor, the tumor's Size and the recommended treatment. Evidently there will be simplifications that can be made. For example, pathways 1 and 2 have the same recommended terminal node treatments. As a result, the constraint can be an OR(Pathway1, Pathway 2). This can be handled by the Z3 solver, and will be discussed in section 5.4. Ultimately, a pathway rule is a logical constraint that combines a series of Field level rules such as "TStage = 1 | 2 |3" using AND to form a branch of the decision tree called a pathway. Each pathway is combined using an OR to form the constraint for the entire set of treatment pathways.

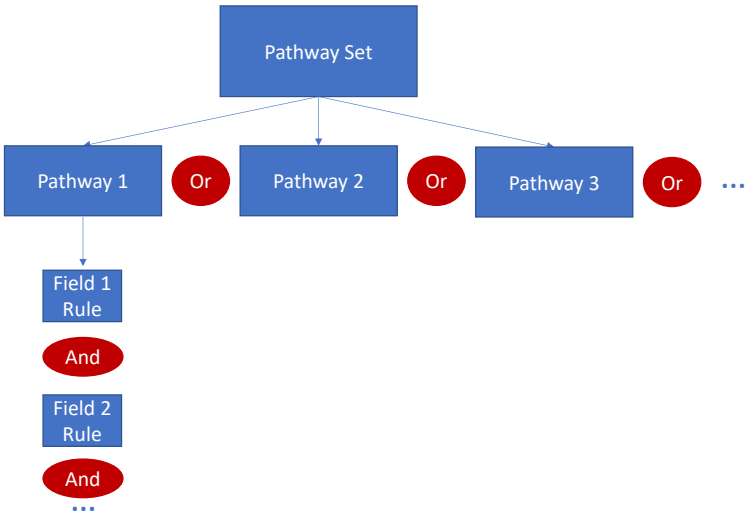


Figure 8: Structure of a Pathway Set Rule as a combination of Field level rules to form a Pathway, and the combination of Pathways to form the Pathway Set

The patient pathways in each of the resources can create quite large constraints, containing the concatenation of up to $n * m$ atomic field constraints, where n is the number of terminal nodes in the decision tree and m is the depth of the tree, or the number of fields considered in each pathway. As such, writing the constraints in any type of logic syntax would be extremely burdensome on a user, especially one not necessarily versed in logical operators. The EMR SMT platform looked to solve for this complexity by allowing for pathways to be represented in .csv files which will be imported into the system. The appropriate constraint can then be constructed in accordance with the structure in Figure 7 programmatically. The basic structure of a .csv file import can be seen below:

	Fields									
	HER2	HR	Histology	T Stage	N Stage	Size	Chemo	Trastuzumab	Pertuzumab	Required
Pathways	+	-	Ductal | Lobular | Mixed | Micropapillary	1 | 2 | 3	0	<= 0.5	ADJUVANT	Y	N	N
	+	-	Ductal | Lobular | Mixed | Micropapillary		1mi	<= 0.5	ADJUVANT	Y	N	N
	+	-	Ductal | Lobular | Mixed | Micropapillary	1 | 2 | 3	0	>= 0.6 & <= 1	ADJUVANT	Y	N	N
	+	-	Ductal | Lobular | Mixed | Micropapillary		1mi	>= 0.6 & <= 1	ADJUVANT	Y	N	N
	+	-	Ductal | Lobular | Mixed | Micropapillary	1 | 2 | 3	0	> 1	ADJUVANT	Y	N	Y
	+	-	Ductal | Lobular | Mixed | Micropapillary		1mi	> 1	ADJUVANT	Y	N	Y
	+	-	Ductal | Lobular | Mixed | Micropapillary		>= 1		ADJUVANT	Y	Y | N	Y

Figure 9: Example .csv flattening of the decision tree from Figure 4. This can be imported into the EMR SMT tool to develop the rules noted above.

A set of Patient Pathway .csv files can be found in *EMR_SMT/Test Patient Pathways*. Each file contains a set of pathways with Fields pertinent to data fabrication, as well as citations for which page in the resource the rule was derived from. Reading in pathways and creating rules requires a mapping of the Fields and value constraints expressed in .csv table format to the appropriate fields and values from the input datasets. This process will be illustrated in section 5.3.

One important consideration for the addition of pathway rules is that by including pathway rules, it will impact the fabrication process with respect to how closely the fabricated data matches the input data. If pathway rules are used in the fabrication process, but entries in the input dataset do not always follow pathway rules, then the fabricated data will not be representative of the input. As a solution, a **compliance score** can be implemented. Based on input data EMRs, we can test how many EMRs follow any given pathway within the pathway set. If 80% of the input EMRs follow the recommended pathways, the EMR SMT system can will apply pathway constraints to 80% of fabricated EMRs for example. This solution can be explored in further research and can be analyzed on a more specific level. For example, we can find which type of cancer patients tend to fall outside of recommended guidelines, and explore why might that be the case. This becomes particularly interesting when investigating the healthcare system administering care, the guidelines they follow and perhaps guidelines from a different healthcare system.

4.4 Constraint Solving and Data Fabrication

The process of fabricating EMRs based on the data structure and rule types implemented the following steps:

1. Generate a set of *n* valid CHI numbers based on rules for CHI
2. For each valid CHI:
    - create an empty EMR with for that CHI
3. For each EMR *e*:
    - For each input dataset *d*:
        - Use dataset rules to generate the number of entries *k* that *e* has in *d*

- For each entry *k*:
    - Create a solvable constraint model *m* for the fields in *d*
    - If *m* is satisfiable, add the solution as entry *k*

This process works well for rules that are strictly contained within a dataset. However, EMR data relies heavily on relationships across datasets. As such, in step 3 we will often be creating satisfiable field assignments for fields not in dataset *d*. These assignments must be upheld to maintain constraint satisfiability, so they must be stored in a list of unassigned fields, to be properly re-used in other models for additional datasets that either contain that field, or call on that field for a constraint.

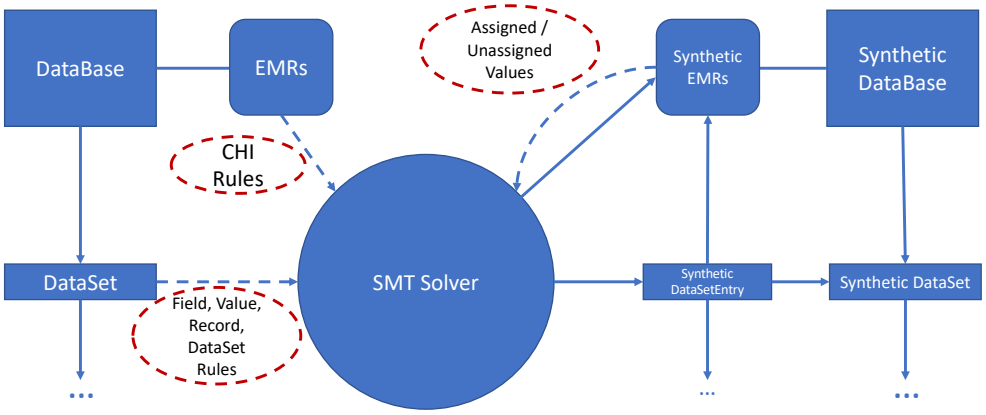


Figure 10: Process for Synthetic DataBase creation using an SMT Solver

As depicted in Figure 7, CHI rules are passed from the database into the SMT solver to generate a set of synthetic EMRs. The database then passes in the set of constraint rules for each dataset it has. Along with those constraints, the synthetic database passes in the assigned values in those constraints from previously created synthetic dataset entries and unassigned values for each EMR. Based on those previous value assignments, another synthetic dataset entry can be created. All fields in the model dataset are added as an entry, and all new assignments which do not have a synthetic dataset created yet will be passed to unassigned values to be used in future constraints and eventually assigned once that field's dataset is being fabricated.

The Z3 solver[5], an SMT Solver developed by Microsoft served as the CSP solver engine behind the creation of fabricated EMRs. Z3 can either run scripts written in the SMT-LIB2 format – a universal language for SMT solver constraints[6], or use APIs through higher level programming languages (Bjørner, de Moura, Nachmanson, & Wintersteiger, 2019). For the purpose of this program the Python front-end, Z3.py, was utilized to build Z3 scripts based on the availability of documentation and the simplicity of structuring and syntax vs. .NET and Java APIs.

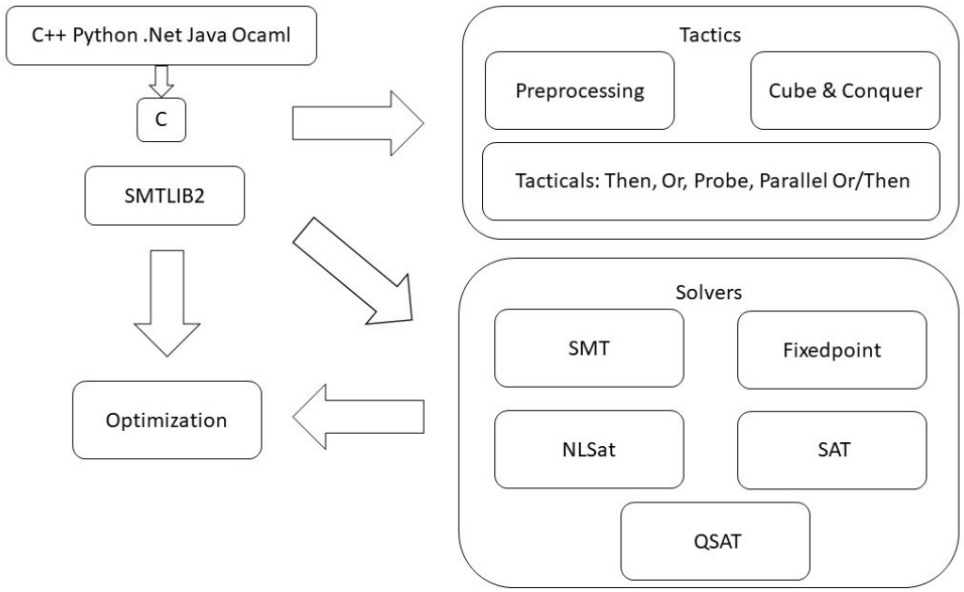


Figure 11: The system architecture of Microsoft's Z3 Solver. Originally published as "Figure 1: Overall system architecture of Z3" in (Bjørner, de Moura, Nachmanson, & Wintersteiger, 2019)

To use Z3 for creating fabricated data, constraint rules and data structures from the EMR SMT database needed to be translated into Z3.py format. Z3 offers a series of default data types, or Sorts, that are used to build constants for a constraint model. It also offers the ability to define unique sorts or utilize generic sorts to define relationships. Since the EMR SMT needs to handle Z3 output as real values, the following default sorts were used for the EMR SMT field types:

- Int Sort → covers all integer values and fields
- Real Sort → covers all double and date values and fields

[5] https://github.com/Z3Prover/z3
[6] http://smtlib.cs.uiowa.edu/

- String Sort → covers all string values and fields

Additionally, Z3 implements Arrays which allow for functions and constraints to be applied across a set of constants. This structure is useful when looking at Record and Field constraints.

Scripts are created for each dataset to model the creation of an entry within that dataset. The Z3 scripts created for the EMR SMT platform are mainly created using the following format:

1. Read in all rules from the database that include constraints related to any field in the dataset
2. Create constants for each field mentioned in those constraints with Sort based on the field type.
3. Initialize a Solver object
4. Add all constants to the Solver
5. If any constant has already been assigned a value from a previous model, create a rule that assigns the constant the assigned value
6. If a constant has not previously been assigned a value, add the value and field rules for that field in Z3 syntax
7. Add all record rules in Z3 syntax
8. Add all rules and assignments to Solver
9. Check the model using Solver.check()
10. If the model is satisfied, read in the constants and values that satisfied the model to EMR SMT
11. If the model is not satisfied, a new script will be created causing new value assignments to unassigned constants until the script is satisfied.

Chapter 5

Implementation

This chapter will cover the technologies, techniques, structures and algorithms used to achieve all of the design tasks highlighted in Chapter 4. It will also explore the limited set of user features built into the EMR SMT platform's GUI.

5.1 System Overview and Technologies Used

The EMR SMT platform is primarily built using the Java programming language in an Object Oriented Programming approach. A basic GUI was created for the project using Java Swing. The platform also utilized Python as the top-level language for Z3 scripting. A complete list of technologies used can be found below, with instructions for installing all necessary components in Appendix A.

Name	Use
Java (Version 8) – Default Packages	GUI, General Object Creation
Java Commons Math (.jar)	Statistical Analysis of Input Data and Random Sampling
Java Apache Commons Lang	Enhanced utilities for Sting parsing and Replacement
Python (Version 3.9.1)	Used for running Z3.py
Z3.py (z3-solver)	Used as main API for Z3 solver

Table 7: Technologies used to build and run EMR SMT

The Java classes designed for this project are broken up into 4 major packages:

- GUI → Contains the Swing classes that comprise the GUI
- Data → Contains the classes that comprise the Data Structure (Figure 5)
- Pathways → Contains the classes that comprise Pathway Sets (Figure 7)

- Solver → Contains the packages for each Constraint Rule Type (see Sect. 4.3) as well as the primary interface with Z3.py

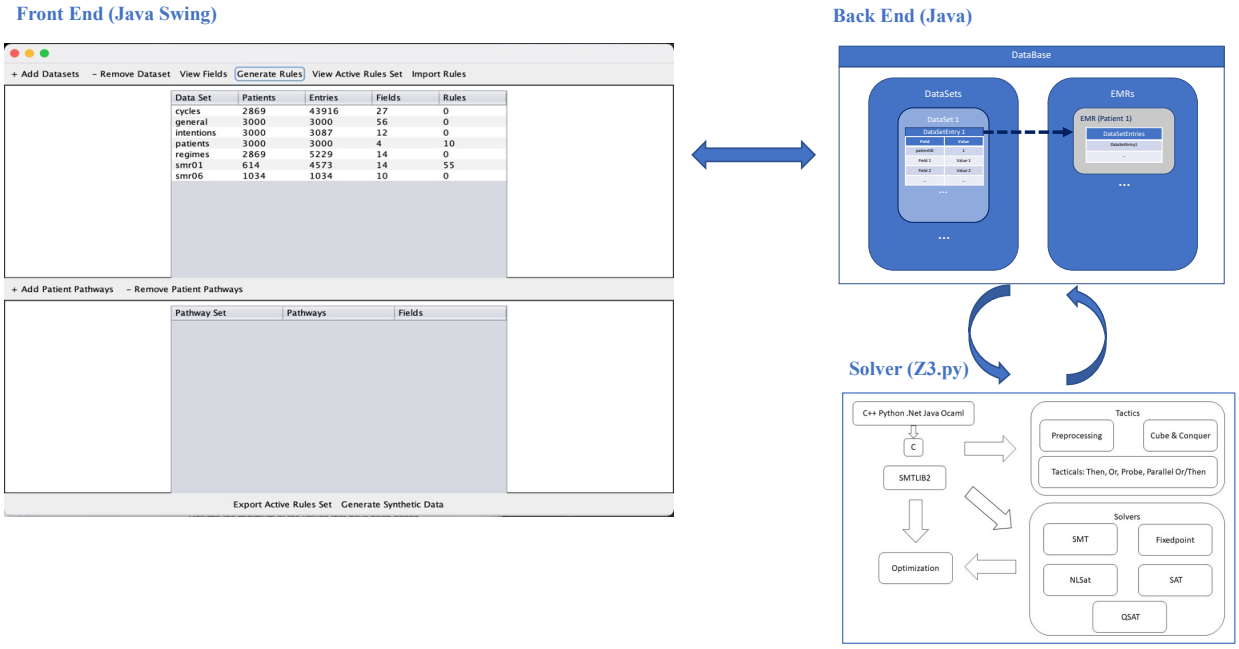


Figure 12: EMR SMT Implemented System Components

5.2 Data Import Implementation

When the EMR SMT is run, it instantiates a new DataBase class which has a empty list of EMR objects and a an empty list of DataSet objects. Data import into the system is an important step in the process flow because it takes a .csv file and instantiates the necessary data structures. Upon import, a new DataSet object is created with an identifier based on the name of the file. The system reads in the top row of the file as new Field objects. Each Field is subsequently given an attribute which links it to the DataSet, and a name that combines the name of the DataSet with the name of the Field (column header). For example, the CHI field in the patients.csv file will be named *patients.chi*. This is important because it helps to distinguish between repeat column names across datasets which may or may not contain the same information.

5.2.1 EMR Creation

The remaining rows of the .csv file are parsed. When a .csv file is imported into the system via the user interface, the user identifies the column containing the patient id field (in this case CHI Number). This patient id column is important because it's value will map each row in the imported file back to an EMR in the database. If an EMR for that patient ID does not exist already exist in the database, a new one will be created.

Once the appropriate EMR is found in the database or a new one is created, the rest of the data in the row is added to a DataSetEntry object, which links the Field from the column to the string value in that row. It is important that the value is initially set as a String, as these will later be changed to match the value patterns for the dataset.

5.2.2 Field and Value Type Parsing

Once all the rows are parsed, and DataSetEnry objects created and added to EMR objects, Field types are parsed. This is done by reading all the String values for a given Field from each DataSetEntry. Field types are assigned to either Integer, Double, Date, Time or String based on the following algorithm:

1. Identify a list of accepted blank values for the dataset (i.e "NA", " ", "-" etc.)
2. For each DataSetEntry *e* in DataSet get String Value *v* mapped to Field *f*
3. Attempt to parse a Java Double object from the String Value
    a. If a double can be parsed (ie. no NumberFormatException thrown), parse the String value for a decimal point '.'.
    b. If not, break the loop.
4. If all of the values were either a blank, or successfully parsed to a number then the Field is an Integer or Double type. If there was a decimal found in any of the non-blank values it is a Double. If not, it is an Integer Field.
5. If not, the same process is performed with a set of DateFormats. If all values can be parsed to a date format, or are blank, then the Field is a Date Field.
6. If none of the field types are assigned after the previous steps, the Field is a String type.

Based on the parsed Field type, all Values mapped to that Field in the entries for the DataSet will be converted to that type prior to their addition to the DataBase. The Value class contains a series of functions which convert a Value to any one of the subclasses of Value types:

- IntegerValue
- DoubleValue
- DateValue
- TimeValue
- StringValue

Each of the subclasses of Value contain unique attributes for the functional value of the objects (ex: a DoubleValue has an attribute that is a Java Double, a DateValue has an attribute that is a Java Date). They also contain unique comparators which are essential for input analysis.

Assigning the appropriate value type is important because it changes how the data is analyzed and how automatic rules are created. Additionally it impacts how Fields and Rules are translated into Z3 for fabrication. The user has the ability to adjust Field types using the user interface, which automatically converts all associated Values in the database.

5.3 Data Structure Implementation

Once the input data has been successfully imported, it is important that the data is organized in the data hierarchy proposed in Figure 5. Objects within the system maintain lists and attributes which connect all the data structures to one another. The following rules outline the implementation of the data structure using object attributes and lists to create the linked hierarchy.

- A DataBase has many DataSets and many EMRs
- Each DataSet has many Fields, and many DataSetEntries
- Each DataSetEntry has a single EMR and many Fields with one Value per Field
- Each EMR has many DataSetEntries and may have more than one DataSetEntry per DataSet

5.4 Rule Generation and Import Implementation

5.4.1 Rule Objects

All rules within the EMR SMT extend a class called *ConstraintRule.java* in the Solver package. A rule object is used to capture trends within the data, or proposed for the data, designate the requisite Fields for which the rule applies and have methods for presenting the field in both String (mirroring the .dfproj format) or Z3 formats.

Rule objects are broken up into major sub packages, as mentioned in section 4.4, which allows for rules to be applied at different stages of fabrication, and handled in different ways. In addition to Value, Field, DataSet and Record rules, rules imported from files and rules derived from pathways are stored as separate data structures. Based on string parsing algorithms, the pathway and imported rules are translated into usable rule objects within the system which are passable to Z3.

5.4.2 Rule Types

The following rule types were implemented for the purpose of this project:

Rule Type	Data Structure	Function
ValueRandomNumberRule*	Value	Sets a Value to a random numeric value between a min and max value
ValueEqualsRule*	Value	Sets a Value equal to a value of any type
ValueDistributionRule*	Value	Sets a Value to a random sample from a Normal Distribution
ValueWeightedDistributionRule*	Value	Sets a Value to a value of any type selected via Enumerated Distribution
ValueConditionalDistributionRule	Value	Sets a Value to a value of any type based on an Enumerated Distribution conditional on the assigned Value of another Field

ValueConditionalNormalDistributionRule	Value	Sets a Value to a random sample of a Normal Distribution conditional on the assigned Value of another Field
FieldEqualsRule*	Field	The values of the Field are exactly equal to the values of another
FieldAllDiffRule*	Field	The values of the Field are all different
FieldMonotonicRule*	Field	The values of the Field are all different and have a monotonic sequencing
RecordAllDiffRule	Record	The Values for a set of Fields in an EMR are all different
RecordConcateRule	Record	The Value for a Field is the concatenation of the Values of a set of Fields
RecordConditionalRule	Record	The Value of a Field is assigned via an "if, else if, else" conditional
RecordLastValueRule	Record	The Value of a Field is based on the last assigned Value of another Field
RecordNumOfRule	Record	The Value of a Field is based on the number of occurences of some other Field or Value within the EMR
RecordArithmeticRule	Record	The Value of a Field is based on an arithmetic calculation using the Value of other Fields in the Record
DataSetNumEntriesRule*	DataSet	Sets the number of DataSetEntries per EMR for a DataSet to a single number
DataSetRangeEntriesRule*	DataSet	Sets the minimum and maximum number of DataSetEntries per EMR for a DataSet
DataSetWeightedEntriesRule*	DataSet	Sets the number of DataSetEntries in a DataSet for a given EMR based on a sample from an EnumeratedDistribution

Table 8: ConstraintRule Subclasses implemented in EMR SMT

5.4.3 Rule Generation

EMR SMT generates statistical rules from the input data that serve as a baseline for generating data representative of the input sample. The system handles the creation of all of the rule types indicated with an * next to the name in Table 7.

The generated rules for Values are used to generate a random assignment based on a Valuem distribution or range. As such, only a single Value rule will be generated per Field. The system chooses which rule to use based on the type of data and number of unique values.

The generation of Field rules is dependent on all data in a given column of input data. The FieldEqualsRule parses all existing Fields in the database and attempts to find any Field with the same name, and set of values per EMR. If this holds for all EMRs, then a FieldEqualsRule is created. The algorithm then checks to see if all non-blank values increase monotonically when sorted in order to create a FieldMonotonicRule. If not, the algorithm checks whether all non-blank values are different and creates a FieldAllDiffRule. Not every Field will have an automatically generated FieldRule.

Field and Value rules are generated within the Field class. DataSetRules are generated in the DataSet object by gathering counts of EMRs and entries per EMR in the dataset. The rules generated for the DataSet, as well as all of the Fields in a DataSet are stored in a list for that DataSet to be called upon during the fabrication process.

5.4.4 Import Rule Parsing

Imported rules for this project were reused from the *CHEMO_REDO_WITH_VIRTUAL* file as discussed in Section 4.4.4. Imported Rules are read in via an XML parser that captures Fields used in each rule and the String constraint. An ImportRule object is create using these attributes. Once an ImportRule object is created, the system parses the String constraint and uses the size, ordering and operators (Table 5) to determine the appropriate Rule Type (Table 7) to create for the Imported Rule.

If an imported rule is parsed to a Field or Value rule, it must be compared to any rules already in the system. If the rule conflicts with an existing rule (ie. a ValueConditionalDistRule vs. a ValueWeightedRule), the imported rule will replace the existing rule.

For the purpose of this project, one additional critical patient Field was added via import rules. The binary integer value *general.menopause* was created using the .dfproj format and is in

General.dfproj. This Field will prove key in many of the Pathway Rules, and as such could not remain omitted from the database.

The rule created is:

```
SERUMS.USTAN.general.menopause = (
SERUMS.USTAN.general.age < 40 ? randomWeightedNumber(
  1 ? 1,
  99 ? 0),
 40 <= SERUMS.USTAN.general.age < 65 ? randomWeightedNumber(
 85 ? 1,
 15 ? 0),
 SERUMS.USTAN.general.age >= 65 ? randomWeightedNumber(
 998 ? 1,
 2 ? 0))
```

This rule accounts for the fact that around 1% of women experience menopause prior to 40[7]. Accurate figures for menopausal age could be added to determine the percentage of women achieving menopause within the median range of 40 and 65 years. This was set to 95% for the purpose of this project, with all women over the age of 65 having a menopause probability of 99.8%. This is based on normal distribution values for 2 and 3 standard deviations from the mean respectively

An additional conditional rule was needed to ensure that a menopause value of 0 applies to all male patients:

```
(SERUMS.USTAN.general.gender == 1) -> (SERUMS.USTAN.general.menopause == 0)
```

Menopause is an important factor in treatment pathway decisions. This field itself should be explored further in Breast Cancer EMR fabrication, particularly considering the link between chemotherapy treatment types and medical menopause.[8]

[7] https://www.nhs.uk/conditions/menopause/
[8] https://www.breastcancer.org/tips/menopausal/types/treatment-induced

5.4.5 Pathway Rule Parsing

Pathway rules are imported into the system via a .csv file as noted in Section 4.4.5. Each file is a PathwaySet which contains rows of Pathways. Within a Pathway, each cell contains a simple rule or relation, or PathwayValue, for the PathwayField in that column. Rule relations can contain the following operators:

Operator	Meaning
|	Or
&	And
+,-,*, /	Mathematical Operators
<= , >=, <, >	Comparators

Table 9: Pathway File Operators

In addition to the operators in Table 8, a PathwayValue rule will contain values which are either String values, numerical values or references to the name of another PathwayField.

When a PathwaySet file is imported to EMR SMT, the user will be asked to map the PathwayFields from the PathwaySet to Fields already within the system. If not Field exists to map, that PathwayField can remain unassigned, but will not be included in any created constraint. Once a Field has been mapped, the user will be able to map the values from any PathwayValue rule to any Values within the system. For example, a "Primary Surgery" PathwayField may occur in a Pathway with a PathwayValue of "Masectomy". The user can map this PathwayField to smr01.main_operation_a and the value of "Masectomy" from the pathway to the Values [B27.2, Z94.3,Z94.2] [9].

A PathwayRule can then be created using the mapped Field, mapped Values and the operators parsed for each PathwayField. A PathwayRule is the conjunctive AND of all of these individually parsed rules as discussed in 4.4.5.

[9] See https://www.ndc.scot.nhs.uk/Dictionary-A-Z/Definitions/index.asp?Search=O&ID=993&Title=OPCS4%20Coding%20of%20Operations/Procedures/Interventions for Patient Intervention Codes

A PathwaySet's compliance percentage is calculated using a quick Z3 script. For each input EMR in the database, the Field assignments are tested to see if any PathwayRule from a PathwaySet is satisfied. The percentage of input EMRs which satisfy a PathwayRule will dictate how many fabricated EMRs will be created using the PathwaySet guidelines.

5.5 Constraint Solver Implementation

5.5.1 Z3 Solver

The Z3 solver is accessed via the SMTSolver class. The SMTSolver class generates a SyntheticDatabase based on an existing DataBase. As discussed in Section 4.5, the process begins by using a basic Z3 script to create a satisfiable set of SyntheticEMRs based on CHI rules provided in *CHI.dfproj*. Once the SyntheticEMRs are created and added to the SyntheticDatabase, the process of fabricating data begins by creating valid DataSetEntries for each DataSet in the existing DataBase.

One important aspect of the fabrication process is the order of DataSet fabrication. Since RecordRules allow for the assignment of variables via constraints across DataSets, the ordering of variable assignments have a cascading effect on which constraints will be applied, and which values assigned in later datasets.

Based on the structure of the input data for this project, the natural ordering of DataSet fabrication was as follows:

1. **General** → Creates a set of CHIs as well as diagnosis dates, deceased dates, histology, comorbidities, gender, age etc. which will all be used for future datasets and constraints.
2. **Patients** → Generates the First Intention of the patient which is necessary for the Cycles-Regimes-Intentions structure
3. **SMR01** → Creates a set of hospital visits and procedures which may be dependent on histology, dates and intentions previously defined.
4. **SMR06** → Creates additional properties for the cancer diagnosis which are dependent on values from general and will impact the treatment properties in Cycles, Regimes and Intentions

5. **Intentions** → Generates the details surrounding the intentions for each CHI, as well as the number of regimens and cycles needed for that Intention.
6. **Regimes** → Builds the details of each regimen for the previously generated Intentions
7. **Cycles** → Builds the details for each cycle of treatment for each previously generated regimen

For each DataSet, a number of DataSetEntries per EMR is chosen by satisfying all of the DataSetRules for that DataSet. A Z3 script is then built for the EMR which takes into account any previously assigned Fields in the EMR, any outstanding Fields to be assigned, and all of the necessary Value, Field and Record constraints.

The script is then run using Z3. The script either returns a set of Fields mapped to values that satisfy the constraint model outlined in the scrip, or returns "unsat" if no assignment of Values satisfies the model. The Z3 solver does not actually handle random Value generation. Rather, the Field and Value rules generate new assignments. If a script returns "unsat", a new script is generated, with new generated assignments and passed into the solver again. This occurs until enough satisfiable DataSetEntries are created to satisfy the DataSetRules. A policy for setting a maximum number of unsatisfied scripts before aborting the fabrication process should be implemented to prevent any infinite loops occurring from conflicting rules not caught by the system prior to fabrication.

5.5.2 Building Z3 Scripts from Rules

As previously mentioned, the EMR SMT generates python scripts for the Z3 solver in order to perform the fabrication process. The script is broken up into the following main components:

1. **Constant Declarations** → Any Field which is included as part of a constraint for the DataSet being fabricated will be declared as a constant. The constant takes the form of:

   constant_name = Sort('constant_name')

   Where the constant name is the Field name, and sort will be Int, String, or Real
2. **Solver Initialization** → A Solver object is created using the line:

   s = Solver()

3. **Value Assignments** → If an Field has been previously assigned a value, an assign rule will be added for that constraint and added to the solver's model using:

   constant_name_assign = (constant_name == v)

   s.add(constant_name_assign)

   Where *v* is the previously assigned value from the SyntheticDatabase.

4. **Constraint Declarations** → All Value, Field and Record constraint rules that are applicable will be added to the solver's model using:

   rule_name = [rule text]

   s.add(rule_name)

   Where the rule name and rule text come from the ConstraintRule object.

Once a all of the constants and rules have been added to the solver in the script, *solver.check()* can be ran to check whether there exists a set of value assignments for the constants which can satisfy the solver's model. If there is, *solver.model()* can be used to extract all of the assignments.

These assignments are then passed back to EMR SMT via a Java via a buffered reader and the constants are mapped back to Fields within the database, assigning the model's value. A sample script can be found in *EMR_SMT/z3Scripts*.

5.5.3 Generating Data using Z3

Generating synthetic data relies on the use of a SyntheticDataBase object which mirrors the functionality of the main database for the most part. It maintains a set of SyntheticDataSets and SyntheticEMRs which allow for Z3 scripts to access data from previously fabricated datasets, entries and records (Figure 7). This allows for Z3 to not need to maintain any record of data which has previously been assigned. Instead, required data is accessed at the point of scripting, and passed on via a value assignment statement.

By implementing the fabrication process in this way, an entire EMR is never solved for in one massive constraint, but rather built iteratively as previously discussed. Z3 acts as a stateless check for satisfaction of all rules given previous assignments and any additional random assignments generated by the rules.

This process replicates a sort of search for satisfiable values by selecting random, or weighted random values based on data distribution, or assigning based on other rules. Rather than utilizing a traditional search tree that CSP solvers may use for variable assignment. This is due to the difference in required variable assignment strategies which are needed to help replicate data occurrence patterns from the entry dataset. For example, if we asked Z3 to find a hospital entry date for a surgery that is greater than the date of their last chemotherapy cycle + 20 days, it will by default return the earlies possible date within those bounds. Using the approach laid out by this project, we can create data that replicates distribution of real world values, rather than the first solution, or "best solution" found by a solver.

This is by no means the only way to implement a CSP based solver. Further exploration could be done into utilizing search within the solver for value assignments, rather than a more rule satisfaction-based approach utilized here.

Chapter 6

System Testing and Evaluation

The main output goal for this project was to create a fabricated dataset for cancer patient data using an SMT and a variety of input data sources. Unfortunately, the project did not achieve full completion on this front. The system is capable of data synthetization and does produce output datasets, but can only do so through generated rules. The translation of imported rules and pathway rules was not successfully achieved at the time of writing this paper, but the framework has been built to do so, as presented in sections 5.4.4 and 5.4.5 above.

6.1 System Output

The system output achieved is a set of .csv files, each representing an input dataset. The user has the ability to select the desired amount of EMRs to export at the time of fabrication. The system can also export an XML .dfproj file containing all of the rules created and used for fabrication which reflects the same format as the .dfproj file imports.

6.2 Testing Data Feasibility

A major component of this project that was not implemented due to time constrains is a methodology for testing output datasets for feasibility using a machine learning alogorithm. The idea for future implementation would be to provide a tagged set of input data entries and a tagged set of output fabricated entries as training data, then have the system tag a test set. If the accuracy is close to 50%, then the fabricated data is a close representation of the input. This process would need to take into account the inclusion of additional rules from pathways as this will require an accurate compliance score.

6.3 Critical Analysis

Overall, this project proved how complicated creating a system with just 3 separate rule creation methodologies and 18 different rule types can be. The goals created prior to the project were ambitious, and multifaceted. Looking at the project wholistically, there was great progress made in the areas of all four design goals, but tying the entire system together proved to be a difficult final step within the timeframe of this project.

Especially in the context of the IBM Infosphere Data Fabrication Platform, this project showed how multifaceted a solution for data fabrication needs to be. IBMs tool can operate on top of its own CSP platform. This project attempted to take data and rules, translating them into an EMR SMT data structure, and then ultimately translate them again to Z3 format.

Despite the shortcomings of this project, additional work can continue to build on this system and help build out rulesets that

6.3.1 Limitations and Rules that Need to Be Captured in the Future

The major limitation to this project was time. The goals set out to develop a system for data fabrication using an SMT and a process for developing EMR rules for breast cancer patients within that system proved to be a big undertaking. Implementing a solver proved to be a large development task, taking up a large proportion of the allotted time for this project. With the fabrication framework implemented, the next steps would be to grow the usability of data from other sources, focusing on translating imported rule sets of multiple formats into ConstraintRule objects executable in Z3.

Chapter 7

Conclusions and Future Work

7.1 Project Summary

The objectives set forward for this project were achieved in part, but not in full. The process of looking at a patient EMR fabrication system wholistically exposed the total complexities of implementing such a system from scratch, and why no simple solution exists in the market. While the objectives were not achieved to the level initially imagined, substantial thought and development was put into EMR SMT which can lay the groundwork for future development. When evaluating progress versus the primary objectives for the project and the

7.1.1 Primary Objectives

1. **Extract existing rules from input datasets that can be viewed and modified by a user.**

   This was achieved in part. Input datasets are structured in an Object Oriented way which allows for flexible analysis and rule generation. Baseline statistical rules are generated by EMR SMT but these can be expanded further to capture causality and relationships through the use of correlation and regression. Machine Learning may be an interesting area of expansion for automatic rule creation.

2. **Model unique breast cancer data relationship rules based on user provided oncological principles that can be translated into constraints for a CSP solver.**

   This goal was achieved in part. The EMR SMT presents a framework for capturing patient pathway recommendations from a set of guidelines provided by leading healthcare and cancer organizations. There was a process created for importing .csv files based on the decision tree format of the treatment pathway guidelines. There was also a framework implemented for applying these rules to fabrication using a compliance score.

These rules can be improved in the future, by creating a more user friendly input format and introducing rules more closely tied patient outcomes.

3. **Leverage a CSP solver to take a set of constraints from a variety of sources and create a fabricated breast cancer patient dataset based on an input dataset.**

   This goal was achieved in part. The Z3 solver was implemented and works with the EMR SMT system to create synthetic output datasets. However, the use of all rule types (generated, imported, pathway) was not successfully implemented. Further work needs to be done to prioritize the use of certain rule types (via a weighting system or user input) to guarantee that data can be properly generated.

4. **Implements a GUI for the solver which helps to simplify the creation of rules for a user.**

   This goal could use the most additional work. While a very simplified prototype GUI was implemented that allows for a user to navigate the fabrication process, additional work should be done to handle user missteps in the process, improve the intuitiveness of user flows and visualize complex rule relationships. An enhanced user rule creation platform should also be implemented to allow for writing of rules without being constrained to the .dfproj and .csv file formats.

7.2 Successes and Challenges

Overall, this project proved to be more difficult and complicated than originally anticipated. Building a system from scratch with the appropriate data structure, rule formats and ability to handle rules from multiple sources with different logical syntax proved to be a massive undertaking. The effort is evidence in the robustness of data structures and though behind the entire process but does not represent a final product. Instead, it begins to expose the difficult nature of building a generic fabrication platform designed for specific types of data.

One of the successes of this project was the framework for patient pathways and translating those guidelines into constraint rules. Particularly with data that is meant to measure patient outcome, understanding the recommended treatment frequency, timing and application

for different patient considerations is critical for generating effective data. Another success was creating a data structure that could successfully handle the required data relationships of an EMR and could be recreated using a SMT based fabrication process.

The major challenge the project was being able to create a flexible enough framework for data fabrication that could handle a wide range of data types, rule types and relationships between records and their data to demonstrate a patient's treatment. This task is relatively novel, but will be important as data driven research on healthcare records grows. While this project does not offer a solution, it does present a novel approach and useable insight moving forward.

7.3 Future Work

Future work on the EMR SMT platform should focus first on creating the desired interaction between rule types to achieve an output based on the various rule sources. Beyond that, expansion of the platform should look to achieve the following goals.

7.3.1 Additional Treatment Types

Cancer treatment often involves multiple types of therapies including radiation therapy and endocrine therapies which can be either preferred to chemotherapies or administered in conjunction with chemotherapies which are the main treatments captured by the current input datasets. While pathway rules capture some of these other therapy types, they are not properly used within the system since much of the information surrounding them is missing from input patient EMRs.

Additionally, information around the availability of certain treatments based on regional and national differences could be an interesting dimension to create fabricated rules upon. The input data for this project was based solely on Scottish medical records, but expanding the system to handle records from different countries with different approved treatments, healthcare providers and the inclusion of private vs. public health insurance could change the way a patient's data is analyzed, and the rules created.

7.3.2 Additional Cancer Types

The prospect of building a simplified application that can fabricate patient records is evidently a complex process – particularly when expanding to other disease types with different treatment structures. However, it would be beneficial to build out the platform to capture additional cancer types, each with their own unique oncological principles and histology. By expanding the breadth of rule creation within the EMR_SMT, it should be feasible to expand the fabrication process pretty easily to other cancer types which may follow a similar intention-cycle-regime structure, but with a different set of factors determining their care plan.

7.3.3 Additional Rule Types

The last, and perhaps most influential future work, would be to expand rule types to improve efficiency and cohesiveness of EMR fabrication. As noted earlier, the process for fabricating in EMR SMT is an iterative building process across datasets. It may be beneficial to explore solving all constraints at one time using a search tree, or creating a process that assigns the most or least influential – occurring the most or fewest times in other constraints – first. The scope and need for quality, secure patient data is expanding with the increased desire for analytics and a contrasting demand for data privacy. Being able to better represent the complexities of medical data as a series of statistical and practical constraints will become even more essential.

References

Adorf, H.-M., & Varendorff, M. (2014). Constraint-Based Automated Generation of Test Data. *Software Quality. Model-Based Approaches for Advanced Software and Systems Engineering*, (pp. 199-213).

Barrett, C., Kroening, D., & Melham, T. (2014). Problem Solving for the 21st Century Efficient Solvers for Satisfiability Modulo Theories.

Bjørner, N., de Moura, L., Nachmanson, L., & Wintersteiger, C. M. (2019, April 14). Programming Z3. *Lecture Notes in Computer Science, 11430*. Retrieved from https://theory.stanford.edu/~nikolaj/programmingz3.html

Bowles, J. K., Silvina, A., Bin, E., & Vinov, M. (2020). On Defining Rules for Cancer Data Fabrication. *Rules and Reasoning. RuleML+RR 2020*, *12173*, pp. 168-176.

Chen, J., Chun, D., Patel, M., Chiang, E., & James, J. (2019). The validity of synthetic clinical data: a validation study of a leading synthetic data generator (Synthea) using clinical quality measures. *BMC Medical Informatics and Decision Making, 19*.

Choi, E., Biswal, S., Malin, B., Duke, J., Stewart, W. F., & Sun, J. (2017). Generating Multi-label Discrete Patient Records using Generative Adversarial Networks. *Proceedings of Machine Learning for Healthcare, 68*.

De Moura, L., & Bjorner, N. (2011, September). Satisfiability modulo theories: introduction and applications. *Communications of the ACM, 54*(9), 69-77.

Dube, K., & Gallagher, T. (2013). Approach and Method for Generating Realistic Synthetic Electronic Healthcare Records for Secondary Use. *Foundations of Health Information Engineering and Systems*, 69-86.

Fruhwirth, T., & Abdennadher, S. (2003). Principles of Constraint Systems and Constraint. In *Essentials of Constraint Programming* (pp. 53-62).

Goncalves, A., Priyadip, R., Braden, S., Jennifer, S., Coyle, L., & Sales, A. P. (2020). Generation and evaluation of synthetic patient data. *BMC Medical Research Methodology*.

Gonclaves, A., Ray, P., Soper, B., Stevens, J., Coyle, L., & Sales, A. P. (2020). Generation and evaluation of synthetic patient data. *BMC Medical Research Methodology*.

Gotlieb, A., Botella, B., & Rueher, M. (1998). Automatic test data generation using constraint solving techniques. *ISSTA '98: Proceedings of the 1998 ACM SIGSOFT international symposium on Software testing and analysis*, (pp. 53-62).

IBM InfoSphere Optim Test Data Fabrication. (2020). Retrieved from IBM: https://www.ibm.com/downloads/cas/A0NE7EGV

Janic, V., Bowles, J. K., Vermeulen, A. F., Silvina, A., Belk, M., Fidas, C., . . . Huang, W. (2019). The SERUMS tool-chain: Ensuring Security and Privacy of Medical Data in Smart Patient-Centric Healthcare Systems. *2019 IEEE International Conference on Big Data*, (pp. 2726-2735).

Kautz, H., Gomes, C. P., Sabharwal, A., & Selman, B. (2008). Chapter 2 Satisfiability Solvers. In *Foundations of Artificial Intelligence* (Vol. 3, pp. 89-134). Elsevier.

McLachlan, S., Dube, K., & Gallagher, T. (2016). Using the CareMap with Health Incidents Statistics for Generating the Realistic Synthetic Electronic Healthcare Record. *IEEE International Conference on Healthcare Informatics (ICHI)*.

Moura, L., & Bjorner, N. (2008). Z3: An Efficient SMT Solver. *Tools and Algorithms for the Construction and Analysis of Systems, 4963*.

Peleska, J., Vorobev, E., & Lapschies, F. (2011). Automated Test Case Generation with SMT-Solving and Abstract Interpretation. *NASA Formal Methods*, 298-312.

Rossi, F., van Beek, P., & Walsh, T. (2008). Chapter 4 Constraint Programming. In *Foundations of Artificial Intelligence* (Vol. 3, pp. 181-211). Elsevier.

Scalfani, R., & Bhada, S. V. (2020). Health insurance and its impact on the survival rates of breast cancer patients in Synthea. *Risk Management and Insurance Review*, 7-29.

Silvina, A., Bowles, J., & Hall, P. (2019). On Predicting the Outcomes of Chemotherapy Treatments in Breast Cancer. *AIME 2019: Artificial Intelligence in Medicine*, 180-190.

Tucker, A., Wang, Z., Rotalinti, Y., & Myles, P. (2020). Generating high-fidelity synthetic patient data for assessing machine learning healthcare software. *npj Digital Medicine, 3*.

Walonoski, J., Kramer, M., Nichols, J., Quina, A., Moesel, C., Hall, D., . . . McLachlan, S. (2018). Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association, 23*(3), 230-238.

Yan, C., Zhang, Z., Nyemba, S., & Malin, B. (n.d.). *Generating Electronic Health Records with Multiple Data Types and Constraints.*

Appendix A Repository of Code

A link to the GitHub repository containing the complete code, installation instructions, requisite packages and README files can be found below:

https://github.com/austinhatch/EMR_SMT