TEMPORAL CASE-BASED REASONING FOR INSULIN DECISION SUPPORT

Daniel Brown

A thesis submitted in partial fulfilment of the requirements of the award of

Doctor of Philosophy

The Department of Computing and Communication Technologies

Oxford Brookes University

June 2015

Acknowledgements

I would like to thank Oxford Brookes University for funding this research, and also for giving me the opportunity to delve into the world of research.

This research would not have been possible without the brilliant support of my supervisors Prof Rachel Harrison, Dr Clare Martin, and Dr Ian Bayley. Their expertise, patience and assistance throughout has helped me to greatly improve both my research and presentation skills. I would also like to mention my tutors Dr Cigdem Sengul, Prof Khaled Hayatleh, and Dr Rob Beale. Both my supervisors and tutors have been there every step of the way to guide me in right direction. Thank you for your continuous support throughout.

I also extend my gratitude to the Department of Computing and Communication Technologies at Oxford Brookes University. Their questions, suggestions and friendly presence throughout have greatly enriched my experience.

A special mention also to Imperial College London for suggesting the UVa/Padova Type 1 Diabetes Mellitus simulator, which was used extensively in this research.

I would also like to thank my family and friends for their support throughout. My mother Pat has shown great patience over the last 4 years, and has greatly helped me with her understanding of diabetes from a nursing perspective. My partner and best friend Natalie, for having to dealing with me at the most stressful of times, throughout she has supported and encouraged me. Additionally, I would like to thank Russell and Bedour for taking the time to proof read my thesis.

Finally, I would like to thank my father David, who sadly passed away in July 2014 whilst I was undertaking this research. Without his enthusiasm for technology, it is probable I would not be in the position I am today. His support and encouragement throughout both this research and my life is forever appreciated.

Contents

List of Tables

List of Figures

List of Abbreviations

AI Artificial intelligence

ANN Artificial neural network

BGRI Blood glucose risk index

CBR Case-based reasoning

FDA Food and Drug Administration

FR Functional requirement

HBGI High blood glucose index

IDE Integrated development environment

IOB Insulin on board

IQR Interquartile range

ISF Insulin sensitivity factor

LBGI Low blood glucose index

MSS Main success scenario

NCD Non-communicable disease

NFR Non-functional requirement

NN Nearest neighbour

R2R Run-to-run

T1DM Type 1 diabetes mellitus

T2DM Type 2 diabetes mellitus

TDD Total daily (insulin) dose

TOP Thematic organisation packet

UML Unified Modeling Language

WHO World Health Organisation

Abstract

Type 1 diabetes mellitus is an autoimmune disease resulting in insufficient insulin to regulate blood glucose levels. The condition can be successfully managed through effective blood glucose control, one aspect of which is the administration of bolus insulin. Formulas exist to estimate the required bolus, and have been adopted by existing mobile expert systems. These formulas are shown to be effective but are unable to automatically adapt to an individual.

This research resolves the limitations of existing formula based calculators by using case-based reasoning to automatically improve bolus advice. Case-based reasoning is a method of artificial intelligence that has successfully been adopted in the diabetes domain previously, but has primarily been limited to assisting doctors with therapy adjustments. Here case-based reasoning is instead used to directly assist the patient.

The case-based reasoning process is enhanced for bolus advice through a temporal retrieval algorithm coupled with domain specific automated adjustment and revision. This temporal retrieval algorithm includes factors from previous events to improve the prediction of a bolus dose. The automated adjustment then refines the predicted bolus dose, and automated revision improves the prediction for future advice through the evaluation of the resulting blood glucose level.

Analysis of the temporal retrieval algorithm found that it is capable of predicting bolus advice comparable to closed-loop simulation and existing formulas, with adapted advice resulting in improvements to simulated blood glucose control. The learning potential of the model is made evident through further improvements in blood glucose control when using revised advice.

The system is implemented on a mobile device with a focus on safety using formal methods to help ensure actions performed do not violate the system constraints. Performance analysis demonstrated acceptable response times, providing evidence that this approach is viable. The research demonstrates how formula based mobile bolus calculators can be replaced by an artificially intelligent alternative which continuously learns to improve advice.

Chapter 1

Introduction

Diabetes Mellitus is the term used for a metabolic disorder resulting from defective insulin secretion, insulin action, or both; leading to the subject's inability to control blood glucose levels [Alberti and Zimmet, 1998]. People living with early onset *Type 1 Diabetes Mellitus* (T1DM) require regular insulin doses to control their blood glucose levels, a complex task even for the most motivated individual [Borus and Laffel, 2010]. One aspect of blood glucose control for T1DM is the use of bolus insulin doses to control the blood glucose following a meal. The bolus insulin dosage is often calculated using established formulas with parameters defined by the subject's doctor, blood glucose readings, and carbohydrate counting. A drawback of these formulas is the inability to automatically learn and improve future bolus advice. This leads to the hypothesis that artificial intelligence (AI) in the form of *case-based reasoning* (CBR) can be used to overcome the inability to learn and improve future advice. Case-based reasoning applies the philosophy that solutions which are known to have worked in similar situations can be used to solve new problems [Kolodner, 1993]. This research seeks to investigate whether CBR is a viable approach for predicting and improving bolus insulin doses for T1DM subjects.

Diabetes Mellitus has two classifications: early onset - T1DM also known as Insulin-Dependent Diabetes Mellitus, and late onset - Type 2 Diabetes Mellitus (T2DM) also known as Non-Insulin-Dependent Diabetes Mellitus [Alberti and Zimmet, 1998]. T1DM refers to cases arising as a result of a defective autoimmune system and requires insulin injections several times a day to control blood glucose levels. T2DM occurs due to defects in insulin secretion and insulin resistance. T2DM is treated by changes to diet, the use of oral glucose lowering drugs and in some cases insulin injections.

The World Health Organisation (WHO) states that 347 million people worldwide have Diabetes Mellitus, of which 5% have T1DM [WHO, 2014a]. WHO predicts Diabetes Mellitus to become the seventh leading cause of deaths worldwide by 2030. Statistics presented in a 2014 WHO report on Non-Communicable Diseases (NCDs) [WHO, 2014b] show that 4% of NCD deaths under the

age of 70 are a result of Diabetes Mellitus, with NCDs resulting in 52% of all deaths below the age of 70. Poor management of Diabetes Mellitus is a leading cause for amputation, blindness and kidney failure. Studies have shown that good blood glucose control reduces the risk of long-term complications [DCCT, 1993, 2005], and serves as the motivation for this research. This research proposes and implements an intelligent personalised insulin adviser on a mobile device for people with T1DM.

To establish the current state of apps available to assist T1DM management on mobile devices a search of the iTunes app store was conducted. Using the search term *diabetes* over 1,100 apps were returned, with the majority functioning as logbooks, advice reference guides, or to count carbohydrates. A small minority of the apps provide bolus dose suggestions through variations of a common formula, and provide a foundation for this research to build upon. The step forward this research will take is to use artificial intelligence in the form of CBR to move away from basing decisions solely upon the use of general formulas, and use instead the subject's history to provide bolus insulin advice; tailoring the app to the user.

1.1 Aim and objectives

The question this research will investigate is whether an intelligent and robust mobile app for predicting bolus insulin advice for people with T1DM can be developed. Six objectives are outlined below to answer this question and fulfil the aim of developing such a system.

The objectives are to:

1. Identify the state-of-the-art approaches for T1DM bolus advice.
2. Discover how the state-of-the-art approaches predict bolus advice and evaluate what safety mechanisms are in place.
3. Discover how CBR provides intelligence and how it has been utilised in the T1DM domain previously.
4. Develop a CBR system for predicting user tailored T1DM bolus advice.
5. Implement the system on a mobile device.
6. Evaluate the implemented system for suitability on a mobile device.

Investigating the research question through these objectives will result in a convenient and cost effective solution for T1DM self-management. The approach provides novelty firstly through formally defining state-of-the-art approaches to both understand existing methods and also to help ensure appropriate safety mechanisms are included by the implemented system. This is coupled with the development of a novel mobile CBR system for user specific bolus insulin prediction.

In Chapter 2 we discover that existing state-of-the-art approaches rely on established formulas to predict bolus insulin advice. Variables for these formulas should only be tuned to the user by expert recommendation, making the user reliant on professionals. Even if these variables are tuned to the user, there is a chance that in different circumstances they may not reflect the current situation. Additionally, many of these state-of-the-art mobile apps suffer from design flaws and potential safety issues.

This research will use CBR to predict bolus insulin advice instead of the existing formulas. In Chapter 3 we discuss how CBR has been used in the T1DM domain to assist practitioners, but rarely the subject themselves. Case-based reasoning will allow the resulting app to learn and improve future advice by reusing, adapting and refining previous predictions which were successful. As CBR is constantly learning and refining predictions, the user experience should improve with pro-longed use of the app.

1.2 Motivation

The motivation for this research is to provide a contribution to the advancement of mobile apps for T1DM self-management. Several of my friends, family and acquaintances have been diagnosed with T1DM or T2DM providing an insight into the difficulty of managing the condition. Additionally my mother works with T1DM subjects and frequently says that her patients are always seeking convenient technology which will make self-management easier.

This research will extend existing work by Christine Poerschke in the early 2000s [Poerschke, 2004]. Poerschke's research sought to revise an existing hand-held insulin adviser for T1DM patients. The result of the research was the Patient-Oriented Insulin Regimen Optimiser (POIRO) MK4, a rule-based and statistical expert system. POIRO MK4 is an extension of POIRO, which was jointly developed by Oxford Brookes University and the Diabetes Research Laboratories at Oxford University.

In Poerschke's thesis, the idea of using AI methods such as Artificial Neural Networks (ANNs) and CBR are explored. These approaches were ruled out by Poerschke as they either fail to demonstrate how rules are learnt, or exceed the technological limits of the time. The first problem remains true for ANNs, which can be viewed as a black box approach to learning [Tu, 1996]. The latter is less likely to be an issue with modern mobile devices. Poerschke's research was undertaken during the era of palm PCs, and since then we have had the advent of smart phones and modern tablets, with substantially greater computational ability.

A notable contribution from Poerschke is the use of formal methods to develop the revised expert system. Poerschke used the B-Method [Abrial et al., 1991; Abrial, 1996] to formally specify the application [Poerschke et al., 2003]. Formal specification is important in safety critical domains to prove the system models the problem correctly and to significantly reduce the possibility of errors.

The safety aspect of formal methods is important to the medical domain; firstly to ensure that user inputs are realistic and secondly to ensure that actions performed by the user and system do not invalidate the state of the system.

In this research a formal specification is created to model and understand state-of-the-art bolus calculators available on the market. The formal specifications produced by Poerschke contained proof obligations which were not discharged. The research in this thesis aims to discharge all of the proof obligations generated by the specifications. In addition Event-B will be used instead of the B-Method. Event-B is an evolution of the B-Method and was also developed by Jean-Raymond Abrial [Abrial, 2010].

By extending Poerschke's research to develop a robust and intelligent mobile app for T1DM bolus insulin advice it is hoped that portable and reliable advice can be made available to greater number of individuals with T1DM.

1.3 Context

The domain of T1DM management includes state-of-the-art bolus calculators and the use of CBR to improve T1DM therapies. This section provides a brief outline of this related work which will be examined in more detail later in this thesis.

State-of-the-art bolus calculators produce their bolus advice through the use of established formulas. Barnard et al. [2012] conducted research on the benefits of automated bolus advisers in young T1DM patients. The research discovered that 89.3% of participants found that an automated bolus adviser made calculating bolus doses easy or very easy compared to manual calculations. Additionally, 78.8% stated that these advisers improved their confidence in dose calculations. An example of such a bolus calculator is the Accu-Chek® Aviva Expert developed by Roche [Roche, 2012], a blood glucose meter which doubles up as an expert system for suggesting bolus advice.

With the increased popularity of smart phones and tablets, a selection of mobile apps to aid T1DM have been developed. A small number of these offer bolus advice similar to the Accu-Chek® Aviva Expert. However, Klonoff [2012] raised safety concerns over mobile apps that offer bolus advice, stating that often there is no explanation of the formula used and safety mechanisms included. Some examples of the mobile apps available for bolus advice include: RapidCalc [RapidCalc, 2015], Diabetes Personal Calculator [iTenuto Soft, 2015], Diabetic Dosage [Diabetic Dosage, 2015], and Insulin Calc [Insulin Calc, 2013]. These mobile applications vary in terms of features available, with RapidCalc containing a large number of settings to help increase the accuracy of the bolus calculator and data visualisation, whilst Diabetic Dosage allows for minimal customisation and does not retain data.

Artificial intelligence in the form of CBR has been utilised for several research projects in the domain of T1DM management, mainly to aid doctors with therapy adjustments for their patients.

Montani et al. [2000] utilised CBR to produce a system for classifying T1DM subjects, allowing doctors to identify the most suitable therapy based on patient classification. The research proved successful, with the CBR system able to correctly classify subjects 83% of the time.

Case-based reasoning was also successfully adopted in the 4 Diabetes Support System^{TM} [Marling et al., 2011]. The research undertaken by this project also attempted to aid doctors in suggesting the most appropriate therapy. The case-base (database) was constructed by recording subject information and identifying key problems with the assistance of a doctor. Each case consists of a problem with blood glucose control and a therapy to solve the problem. Evaluation of the system was conducted by a panel of doctors to see whether they agreed with the suggestions produced by the CBR system. The results were positive with the panel agreeing with 80% of the suggestions.

Most recently, the Imperial NIHR Biomedical Research Centre [Herrero et al., 2014] have incorporated CBR alongside the Run-to-Run (R2R) control algorithm to aid self-management through intelligent bolus dose calculations. The intent of this research is to provide a mobile app to assist the self-management of T1DM. Case-based reasoning is utilised in this project to optimise the parameters of a bolus insulin calculation. The app learns to improve future bolus advice through revision of the parameters in the event of suboptimal solutions. Simulation showed that using CBR in conjunction with R2R improved blood glucose control, with a complete elimination of hypoglycaemia.

The aforementioned research projects for CBR in the T1DM domain highlight the positive effects CBR can have for T1DM therapeutics. The research described in this thesis aims to add to this collection of work by using CBR to improve bolus insulin dose suggestions, aiding self-management.

1.4 Chapter overview

The thesis begins with a look at state-of-the-art bolus calculators currently available for assisting T1DM self-management. These tools are analysed in detail to help identify key aspects involved in bolus insulin advice. A formal specification in then created to ensure the understanding of these tools is modelled correctly, and to assist the implementation of the resulting mobile app.

This is followed by a look at CBR in general, with a look at existing CBR models and their components. This is coupled with a look at some of the seminal CBR applications developed as well as the use of CBR in the T1DM domain.

A CBR model for T1DM is then introduced based upon the findings from the state-of-the-art bolus calculators and CBR research. The model includes the use of temporal sequences, dynamic feature weighting, adaptation and evaluation rules to improve the predictions made by the system.

Finally, the CBR model is implemented as a mobile app with the assistance of the formal specification to help ensure the safety of the resulting app. The final app is then subjected to unit

testing and performance evaluation.

A brief outline of the chapters and their contents are presented below.

Chapter 2 looks at existing bolus advisers on the market. This chapter seeks to identify the features present in both a blood glucose meter with a built-in bolus adviser and mobile apps publicly available. The features and constraints of these state-of-the-art advisers are then used to produce a formal specification in Event-B. The chapter discusses developments in the formal methods community, with a focus on the Rodin Platform for Event-B.

Chapter 3 is an in-depth discussion into CBR, the origins of which are discussed alongside models proposed for implementing CBR systems. The chapter looks at the seminal CBR work, other reasoning methods, and concludes with a look at the use of CBR in the domain of T1DM.

Chapter 4 discusses a case-based reasoner for T1DM bolus advice. A retrieval method is proposed using temporal sequences to aid the identification of the most suitable cases. This is coupled with a method for weighting distance functions through feature selection. An adaptation rule is then proposed to reduce the effect of insulin stacking. This is followed by an automated method to evaluate and revise a suggested solution in order to improve future bolus insulin advice.

Chapter 5 analyses and evaluates the bolus advice produced by the CBR system proposed in Chapter 4. The chapter begins with an explanation of the methodology applied to testing the CBR system, and identifies appropriate statistical measures for evaluation. The chapter concludes with a comparison of the CBR results to those obtained by a T1DM simulator and formulas used by a state-of-the-art bolus calculator, and the effectiveness of the adaptation and evaluation approaches.

Chapter 6 discusses the implementation of a mobile prototype for bolus advice using CBR, and goes on to outline its requirements and design. The formal specification devised in Chapter 2 is used to aid the implementation of a robust mobile app for bolus decision support. The chapter concludes with an evaluation of CBR performance on a mobile device.

Chapter 7 concludes this thesis with a discussion of research aim and objectives, research impact, researching limitations, and recommendations for future research in the area.

Chapter 2

Formal specification of a bolus calculator

The self-management of T1DM is a complex task requiring good numeracy skills to help ensure successful glycaemic control [Borus and Laffel, 2010; Kerr, 2010; Marden et al., 2012]. Individuals with T1DM often maintain a diary for recording various factors such as meals, blood glucose readings and insulin doses. This information can then be used to manually calculate a suitable bolus dose [Sussman et al., 2012]. More recently artificial pancreases are being researched and developed as an effective alternative for self-management, where glycaemic control is managed through the automated delivery of insulin [Harvey et al., 2010]. However, this solution is not yet openly available to T1DM subjects, with clinical trials and refinement still ongoing [Kovatchev et al., 2014; Nimri et al., 2014]. To bridge this gap between manual bolus insulin calculation and artificial pancreases a number of bolus calculators have been developed. These bolus calculators allow T1DM subjects to quickly calculate bolus insulin doses in addition to functioning as an electronic diary [Sussman et al., 2012; Barnard et al., 2012]. However, some concerns have been raised of the safety of some bolus calculators [Klonoff, 2012]. We aim to address through the use of a formal specification in this research.

In this chapter some of the state-of-the-art bolus calculators available on the market are examined. Firstly the Accu-Chek® Aviva Expert [Roche, 2012], a blood glucose meter with built in bolus dose decision support. This is followed by a look at existing mobile apps which provide the ability to calculate a bolus insulin dose. The information obtained through the assessment of these state-of-the-art bolus calculators is then used to create a formal specification in Event-B. Prior to the formal specification, the use of formal methods in software engineering and developments in the formal methods community are discussed. The formal specification in this chapter seeks to aid understanding of the problem through reverse engineering, and ensure that the model derived from

the calculators is correct. The formal specification will also aid the implementation of a mobile prototype discussed in Chapter 6.

2.1 Roche Accu-Chek® Aviva Expert blood glucose meter

Accu-Chek® blood glucose meters are one of the most prominent on the market and have been produced by Roche for over 35 years [Roche, 2015a]. Historically, blood glucose meters are used in combination with test strips to accurately inform the user of their blood glucose level before and after meals to aid insulin therapy. Roche have released the Accu-Chek® Aviva Expert [Roche, 2012] blood glucose meter to not only accurately measure blood glucose levels, but also to provide bolus dose advice to the user. In this section the Accu-Chek® Aviva Expert's customisable settings and bolus dose calculation method are discussed.

Figure 2.1: Accu-Chek® Aviva Expert

Settings

The Accu-Chek® Aviva Expert includes a selection of customisable features to tailor the device to the subject's condition. These settings are initially defined using the setup wizard and can be modified through the interface at any time. The settings primarily define values to be used for bolus advice or to warn the user if their blood glucose level breaches user defined thresholds. The settings and their purpose are defined in the list below.

Acting time

Defines the duration that a bolus insulin dose remains active in minutes.

Offset time

Defines the delay in blood glucose reduction following a bolus dose in minutes.

Basal insulin

Allows the user to specify their daily basal insulin[1] dose in insulin units.

Maximum bolus dose

Limits the maximum bolus insulin dose displayed to the user following a calculation.

Meal rise

Defines the maximum increase in the subject's blood glucose level following a bolus, which if exceeded will prompt for an additional correction bolus.

Health factors

Allows the user to define percentage reductions or increases in bolus suggestions for different health events such as stress, illness and exercise.

Insulin increment

Defines the accuracy of the insulin suggestions to correspond with the user's equipment (e.g. insulin pen or syringe).

Insulin sensitivity

Defines the reduction in blood glucose levels in mmol/L[2] for a user defined quantity of insulin units.

Carbohydrate unit

Defines the carbohydrate unit to be used by the device. Available options are: grams, bread equivalent (equal to 12 grams of carbohydrate), kohlenhydrateinheit (equal to 10 grams of carbohydrate), and carbohydrate choice (equal to 15 grams of carbohydrate).

Carbohydrate-to-insulin ratio

Defines the number of carbohydrate units covered by a user defined insulin unit quantity.

Target blood glucose range

Defines the upper and lower bounds of the subject's target blood glucose range in mmol/L.

Hyperglycaemic and hypoglycaemic warnings

Defines blood glucose values in mmol/L for triggering hyperglycaemic and hypoglycaemic

[1] Intermediate or long acting background insulin to keep blood glucose levels constant during periods of fasting [Diabetes.co.uk, 2015]

[2] Blood glucose concentration millimoles per litre, advised UK standard [Joint Formulary Committee, 2010].

warnings for the user. The warnings are triggered if a blood glucose reading is above the hyperglycaemic value or below the hypoglycaemic value.

Table 2.1 states the minimum, maximum and default values for the settings described for the Accu-Chek® Aviva Expert [Roche, 2012]. As the device is Food and Drug Administration (FDA) approved, the values in Table 2.1 will be used in the formal specification discussed later to aid the safety of the system.

Data type	Min	Max	Default
Acting time (minutes)	90	480	240
Offset time (minutes)	45	Acting time	60
Basal insulin (IU)	0	99	0
Maximum bolus dose (IU)	0	50	-
Meal rise (mmol/L)	2.8	11.1	2.8
Health factors (%)	-50	50	0
Insulin sensitivity			
Insulin units (IU)	0.1	50	1
Blood glucose (mmol/L)	0.1	55.4	-
Carbohydrate-to-insulin ratio			
Insulin units (IU)	0.1	50	1
Carbohydrates (grams)	1	240	-
Target blood glucose range			
Upper value (mmol/L)	5.5	15	8
Lower value (mmol/L)	3	8	4
High BG threshold (mmol/L)	6.5	19.5	16.5
Low BG threshold (mmol/L)	3	5.5	4
Hyperglycaemia limit (mmol/L)	10	19.5	16.5
Hypoglycaemia limit (mmol/L)	3	5	4

Table 2.1: Accu-Chek® Aviva Expert settings, value limit and default values

Bolus calculation

The Accu-Chek® Aviva Expert uses the subject's blood glucose level alongside planned carbohydrate intake, exercise, insulin sensitivity factor and carbohydrate-to-insulin ratio to predict a suitable bolus insulin dose [Roche, 2013]. The total bolus dose suggested is calculated through the sum of a correction bolus (Eq. 2.1) and meal bolus (Eq. 2.3). The correction bolus component calculates the bolus insulin required to correct the subject's blood glucose back to their target blood glucose level, and if applicable, also factors in the effects of meal rise and active insulin[3] time. The meal bolus calculates the insulin units required to correct a specified carbohydrate intake.

The correction dose *cb* is defined by Eq. 2.1, let *cbg* define the current blood glucose level, *abg* be the currently allowed blood glucose value (calculated by Eq. 2.2). The subject's insulin sensitivity factor is determined by the user defined number of insulin units $isfi$ required to reduce the subject's blood glucose level by the user defined blood glucose reduction $isfbg$, e.g. if 1 insulin unit reduces the subject's blood glucose by 2 mmol/L then the insulin sensitivity factor is 0.5.

[3]Active insulin is also referred to as *insulin on board* in this thesis.

$$cb = (cbg - abg) \times \frac{isfi}{isfbg} \tag{2.1}$$

Calculation of the currently allowed blood glucose value abg used in Eq. 2.1 is calculated using Eq. 2.2. This formula factors in the target range mean trm, the effects of a postprandial blood glucose meal rise mr, and the sum of blood glucose covered by currently active insulin a. The postprandial blood glucose meal rise mr is only applicable if the correction dose is calculated within a specified time period following a meal.

$$abg = trm + mr + \sum a \tag{2.2}$$

The meal bolus mb (Eq. 2.3) is calculated by multiplying the planned carbohydrate intake c by the carbohydrate-to-insulin ratio, which is determined by calculating the user defined insulin unit quantity cri required to correct the a user defined quantity of carbohydrates in grams crc.

$$mb = c \times \frac{cri}{crc} \tag{2.3}$$

Finally, the total bolus dose is calculated by Eq. 2.4 through the sum of the correction bolus cb (Eq. 2.1) and meal bolus mb (Eq. 2.3).

$$\text{suggested bolus dose} = \begin{cases} cb + mb, & cb + mb > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

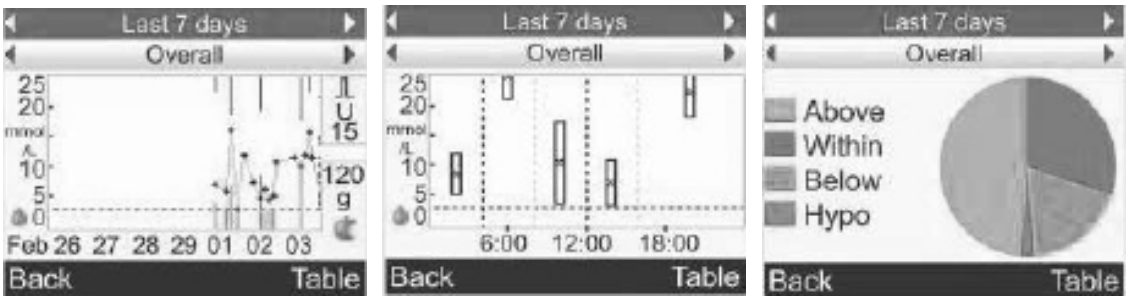
Negative correction boluses are permitted in order to reduce the suggested insulin if there is either residual active insulin or the subject's current blood glucose level is below their target value. The overall suggested bolus cannot be negative, if the sum of the correction bolus and meal bolus is negative then the suggested bolus is 0 insulin units.

Additional features

The primary purpose of the device is to allow the user to test and record their blood glucose level using test strips. Although a unique selling point of this model is the built-in bolus adviser, it is considered an optional feature. Other notable features include the ability to set reminders and visualise data.

The device includes a selection of customisable reminders to inform the user when to perform blood glucose tests and also to notify of upcoming appointments with their doctor. Blood glucose test reminders include the ability to enable daily reminders at set times, post hypoglycaemia and hyperglycaemia episode reminders, and post meal reminders. Data visualisation is another feature

included and allows data to be displayed as trend graphs, a representation of the subject's standard day, or as a breakdown of targets. This trend graph displays blood glucose, carbohydrate intake and bolus dose information together (Fig. 2.2a). The trend graph displays time along the x-axis, blood glucose readings along the left y-axis, and a split of bolus doses and carbohydrate intake along the right y-axis. The right y-axis is split such that the top half displays the bolus dose and the bottom half displays the carbohydrate intake. This view can be filtered by type to display overall, pre-meal, post-meal or bedtime data only over a period of 8 hours, 24 hours, 48 hours, or 7 days.



(a) Trend visulisation (b) Standard day visulisation (c) Target visulisation

Figure 2.2: Accu-Chek® Aviva Expert data visualisation

The standard day visualisation allows the user to quickly see an overview of their blood glucose in time blocks over an average day. This aids in identifying periods of the day with high variations and to view average trends. An example is shown in Fig. 2.2b, where the x-axis represents time and the y-axis represents the subject's blood glucose level. The time blocks depict the variation in readings for each block with the average marked with an "X". This view can be filtered over a period of 7, 14, 30, 60, or 90 days.

Target visualisation allows the user to view the percentage of blood glucose readings that fall into specified blood glucose ranges: within, above, below and hypoglycaemic. The device does not state that hypoglycaemic readings are displayed. This view allows the user to quickly visualise the percentage of time spent within different blood glucose ranges over a period of 7, 14, 30, 60 or 90 days. An example of this visualisation is shown in Fig. 2.2c.

2.2 Mobile apps for bolus insulin decision support

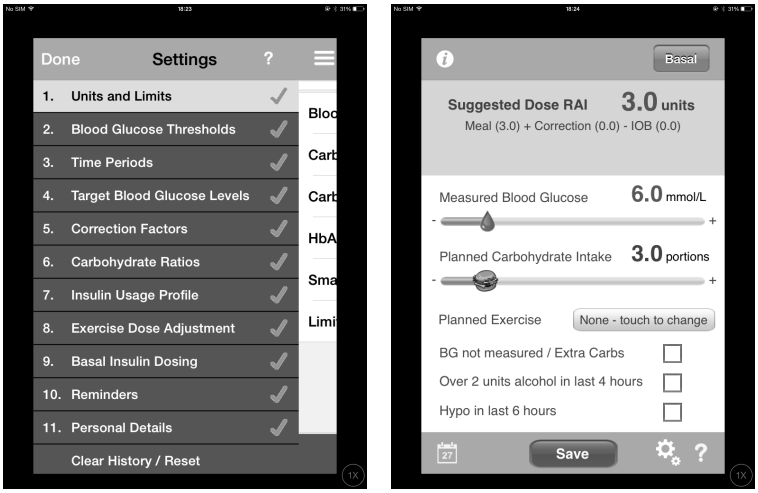
A selection of mobile apps to aid people with T1DM already exist on the market. These apps vary with regards to their purpose, with most offering the ability to record carbohydrate intake, blood glucose readings and insulin usage. Other apps provide general information to help the user manage their condition, acting as a pocket guide. Although beneficial, they do not actively provide insulin decision support. However, a small percentage of the apps available do offer insulin decision support, allowing bolus insulin doses to be calculated using variations of established formulas.

Four of these apps have been selected for analysis and evaluation: *RapidCalc* [RapidCalc, 2015], *Diabetes Personal Calculator* [iTenuto Soft, 2015], *Diabetic Dosage* [Diabetic Dosage, 2015] and *Insulin Calc* [Insulin Calc, 2013]. The selection of these apps was guided by a previous usability study of T1DM mobile apps [Martin et al., 2011]. All of the selected apps allow for bolus insulin doses to be calculated using a variety of bolus insulin formulas, with some of the apps supporting the ability to log meals and visualise data. Each app will be analysed to identify user definable settings, the calculation method, and additional features such as meal logging and data visualisation.

2.2.1 RapidCalc

Analysis and evaluation begins with RapidCalc [RapidCalc, 2015], an insulin decision support app available for iOS. RapidCalc is the most sophisticated of the insulin calculators evaluated in this research, supporting a large number of user definable settings and an easy to use interface. RapidCalc provides support for time period variations of user settings, the ability to include exercise in the calculation, and the inclusion of insulin on board reduction. Insulin on board factors in insulin which is still active following a preceding bolus dose. Configuring the app is not a quick process due to the number of user definable settings available, however these settings allow for an improved user experience in the long-term.

The analysis and evaluation of RapidCalc will firstly look at the settings available to the user. This is followed by a discussing into RapidCalc's bolus insulin recommendation formula. Finally, additional features of the application are discussed.



(a) User settings (b) Calculator

Figure 2.3: RapidCalc application for iOS

Settings

Before RapidCalc can be used to calculate a bolus insulin dose, the user must first define their personal settings (Fig. 2.3a). These settings include details about their condition, units, and

exercise and time variations.

Units and Limits:

Blood glucose units

Defines the blood glucose unit for the application to use (mmol/L or mg/dL[4]). This cannot be modified without a full application reset.

Carbohydrate units

Defines the the carbohydrate unit to be used in grams or gram portions (10, 12, or 15 gram portions).

HbA_{1c} units

Defines the unit used to display HbA_{1c}[5] estimates in percentage or blood glucose units.

Smallest pen dose size

Defines the smallest dose size available to the user's insulin pen or syringe (0.1, 0.5, 1.0, or 2.0 insulin units).

Limit maximum bolus dose

Sets a maximum bolus dose, which if the calculation exceeds will be highlighted in red.

Blood glucose thresholds:

Ideal blood glucose range

Defines the user's ideal blood glucose level in mmol/L or mg/dL.

Warning blood glucose range

Defines blood glucose thresholds at which hypoglycaemia ([3,10] mmol/L) and hyperglycaemia ([3,20] mmol/L) warnings are activated.

Time periods

All the inputs in this section allow the user to set the start time of each period of the day (Breakfast, Morning, Lunch, Afternoon, Dinner, and Overnight). The application automatically enforces a 70 minute difference between time periods, and will always cover a 24 hour period.

Target blood glucose levels

Sets the target blood glucose levels ([3,10] mmol/L) for each time period.

Insulin sensitivity factors

Sets the insulin sensitivity factor ([0.1,20] mmol/L) for each time period.

[4]Blood glucose concentration in milligrams per decilitre, standard in some European countries and the US [Joint Formulary Committee, 2010].

[5]Glycated Haemoglobin, measures the average plasma glucose over a period of 8 to 12 weeks [WHO, 2011]. The measure provides an overall picture of blood glucose control.

Carbohydrate ratios

Carbohydrate ratios for each time period, which states the number of insulin units required to cover 1 carbohydrate portion, or carbohydrates in grams covered by 1 insulin unit. No range limit is enforced.

Insulin usage profile

Sets the percentage of active insulin used as a percentage each hour over a total of 6 hours. All hours must total 100%.

Exercise adjustment

Sets the percentage a bolus insulin dose is reduced for light, moderate and intense exercise. Each intensity of exercise allows for separate reduction percentages for short, medium and long exercise duration.

Basal insulin dosing

Sets the how many basal doses per day [0,3], the time of each basal dose, and the quantity of insulin units in each basal dose.

Reminders:

Basal dose reminders

Reminds the user to administer their basal dose(s).

Postprandial blood glucose reminder

Reminds the user to test their postprandial blood glucose level.

Postprandial blood glucose check delay

Sets the delay time of the postprandial blood glucose check in hours and minutes after a meal.

Personal details

Stores the user's contact information and allows for a password to be applied to prevent others from changing the application settings.

Bolus calculation

RapidCalc's bolus insulin calculator (Eq. 2.5) (Fig. 2.3b) features six inputs for the user to specify. The three main inputs are a preprandial blood glucose reading in mmol/L or mg/dL, the planned carbohydrate intake in grams or the number of portions, and the intensity and duration of any planned exercise. In addition there are three options: blood glucose not measured or extra carbohydrates, over two units of alcohol in the last 4 hours, and hypoglycaemic event in the last 6 hours. If blood glucose not measured or extra carbohydrates is selected then insulin on board is removed from the bolus calculation (Eq. 2.5). If alcohol has be consumed in the last 4 hours or if

a hypoglycaemic event has occurred in the last 6 hours, the correction dose (Eq. 2.7) is removed from the bolus calculation (Eq. 2.5).

$$\text{suggested bolus dose} = \text{meal dose} + \text{correction dose} - \text{insulin on board} \tag{2.5}$$

The meal dose is defined by Eq. 2.6, let ci be the carbohydrate-to-insulin ratio, c be the planned carbohydrate intake, and e be the exercise percentage reduction.

$$\text{meal dose} = ci \times c \times (1 - e) \tag{2.6}$$

The correction dose is defined by Eq. 2.7, let pbg be the preprandial blood glucose reading, tbg be the target blood glucose, isf be the insulin sensitivity factor, and e be the exercise percentage reduction.

$$\text{correction dose} = \left(\frac{pbg - tbg}{isf}\right) \times (1 - e) \tag{2.7}$$

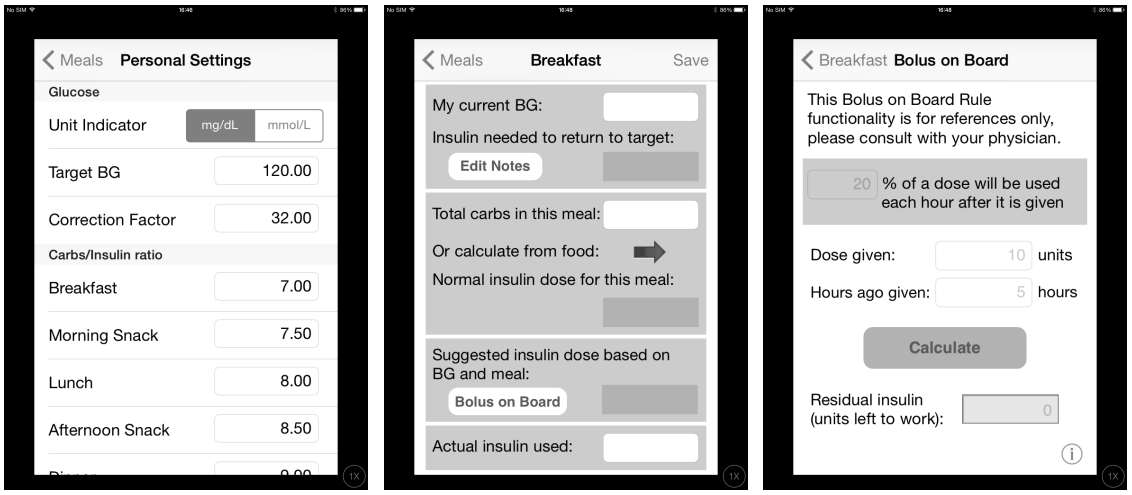
Insulin on board in Eq. 2.5 is calculated using the user defined hourly percentage scale for all meals occurring within 6 hours of any previously recorded bolus calculation.

Additional features

RapidCalc allows all calculations to be logged and viewed in the history. The log entry will display all associated user settings relating to that meal in addition to the calculation inputs and any additional notes. Statistics and graphs can also be generated by the application for 7, 30, and 90 days in the past. The statistics displayed over the selected time period include the number of readings, hypoglycaemic and hyperglycaemic warnings, average bolus dose per day, and average daily carbohydrate intake. RapidCalc includes two data visualisation options: a 24 hour blood glucose profile scatter graph, and a bar chart displaying the average preprandial and postprandial blood glucose readings recorded for each type of meal (breakfast, lunch, and dinner).

2.2.2 Diabetes Personal Calculator

Diabetes Personal Calculator [iTenuto Soft, 2015] is a mobile app available for iOS developed by iTenuto Soft. The app is available as a free or purchased version, with the purchased version evaluated in this research, since it includes additional features. The app allows for time period based bolus insulin calculations to be conducted using a combination of user defined settings and meal specific inputs.



(a) User settings (b) Calculator (c) Insulin on board

Figure 2.4: Diabetes Personal Calculator application for iOS

Settings

The application has a minimal number of settings which must be defined prior to using the calculator. The settings are divided into two categories: glucose, and carbohydrate-to-insulin ratios.

Glucose settings:

Unit indicator

Sets whether the application uses mmol/L or mg/dL units.

Target blood glucose

Defines the target blood glucose level for the user.

Insulin sensitivity factor

Defines the reduction in blood glucose for 1 unit of insulin.

Carbohydrate-to-insulin ratios:

Defines the quantity of carbohydrates covered by 1 unit of insulin. Diabetes Personal Calculator provides carbohydrate-to-insulin ratios for six periods of the day (breakfast, morning snack, lunch, afternoon snack, dinner, and evening snack).

A potential problem with the application is that should the user change the unit indicator setting, then all other glucose settings do no change to reflect the change of units. This may be beneficial if the user realises afterwards that the wrong unit indicator is selected as the values entered will not be changed. Equally, the values could not reflect the specified unit. As the application does not specify the units on the calculation screen, then it is unlikely to be an issue if the user is consistent with their inputs.

More significantly, the application does not impose any restrictions on the values when the subject defines their settings. This allows for potentially unrealistic or incorrect values to be

entered, which would prove dangerous to the subject. The only input validation performed is to ensure that a value has been entered.

Bolus calculation

To calculate a suggested bolus insulin dose with Diabetes Personal Calculator, a preprandial blood glucose reading and the quantity of carbohydrates to be consumed in grams must be input by the user at the time of calculation. The application allows for time period based carbohydrate-to-insulin ratios, which are defined in the user's settings. For each time period a separate calculation screen is available, which are visually the same.

The formula used by Diabetes Personal Calculator is defined by Eq. 2.8, let pbg be the preprandial blood glucose level in mmol/L or mg/dL, tbg be the target blood glucose level in mmol/L or mg/dL, isf be the insulin sensitivity factor, c be the planned carbohydrate intake in grams, and ci be carbohydrate-to-insulin ratio for a selected time period.

$$\text{suggested bolus dose} = \frac{pbg - tbg}{isf} + \frac{c}{ci} \tag{2.8}$$

Diabetes Personal Calculator also provides some support for insulin on board. The app requires the user to remember their previous time and insulin dose, and must be manually deducted from the bolus suggestion acquired by Eq. 2.8. The app's insulin on board calculator reduces the active insulin using an hourly percentage reduction. This calculation is described by Eq. 2.9, let i be the previous bolus insulin dose, p be the percentage reduction of the active insulin over the period of 1 hour, and h be the hours since i was administered.

$$\text{insulin on board} = i - \frac{i}{100} \times p \times h \tag{2.9}$$

Additional features

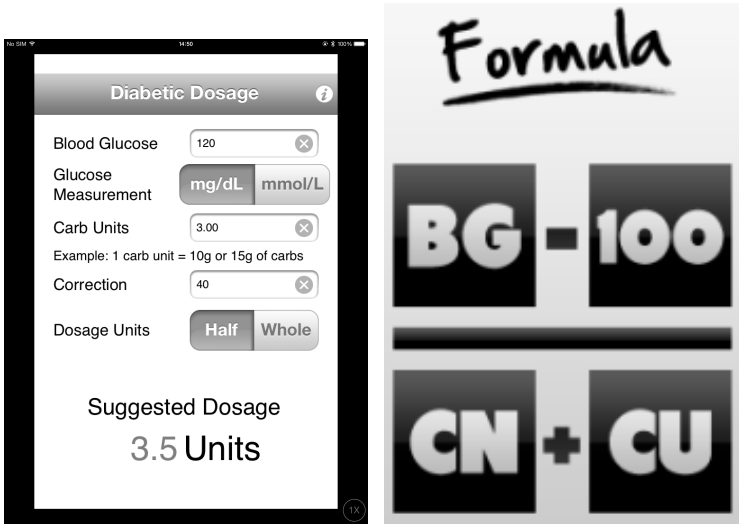
Diabetes Personal Calculator includes a list of food which allows the user to add carbohydrate estimations based on the portion size of the food they are about to consume. The user can add multiple food items for each meal and may also specific their own custom foods and meals.

The user can also record meal and insulin dose information within the app for reviewing at a later date. The application allows the user to view this information for a maximum of 31 days prior to the current date. One negative aspect of the app is that the user may only record one meal for each time period of the day. This limits the user to only six unique meals and snacks each day, which may not be desirable.

Other features available to the application include the ability to record basal insulin doses, and an average blood glucose reading to HbA_{1c} conversion tool.

2.2.3 Diabetic Dosage

Diabetic Dosage [Diabetic Dosage, 2015] is the most basic of the bolus calculators reviewed in this research, and is available for both iOS and Android devices. Unlike RapidCalc and Diabetes Personal Calculator, Diabetic Dosage does not require any settings to be configured prior to use. Instead the user specifies five parameters to calculate a suggested bolus dose: preprandial blood glucose level, blood glucose unit (mg/dL or mmol/L), quantity of carbohydrate unit (1 unit = 10 or 15 grams of carbohydrate), insulin sensitivity factor, and insulin dose unit (half and whole units).



(a) Calculator (b) Published formula

Figure 2.5: Diabetic Dosage application for iOS and Android

The app has some notable issues. One issue is that the app assumes a target blood glucose of 100 mg/dL (approximately 5.5 mmol/L), which is not treated as a constant across all subjects by the other bolus calculators. Another issue is that the formula published on the app's website (Fig. 2.5b and Eq. 2.10) does not represent the actual calculation performed by the app (Eq. 2.11). In both Eq. 2.10 and Eq. 2.11, let pbg be the preprandial blood glucose level in mg/dL, isf be the insulin sensitivity factor, and c be the planned carbohydrate intake in carbohydrate units (1 carbohydrate unit = 10 or 15 grams of carbohydrates).

$$\text{published suggested bolus dose} = \frac{pbg - 100}{isf + c} \tag{2.10}$$

$$\text{implemented suggested bolus dose} = \frac{pbg - 100}{isf} + c \tag{2.11}$$

Both versions of the calculation produce different results to those used by the other apps. Figure 2.6 illustrates the difference in bolus insulin suggestions for Diabetic Dosage in comparison to Diabetes Personal Calculator and RapidCalc (ignoring insulin on board and exercise factors).

$$
\begin{aligned}
pbg &= 6.6 \text{ mmol/L} \\
tbg &= 5.5 \text{ mmol/L} \\
isf &= 2.0 \text{ mmol/L} \\
c &= 60 \text{ grams or 6 carbohydrate units} \\
ci &= 13 \text{ grams per insulin unit} \\
&\quad \text{or } 0.769 \text{ insulin units per carbohydrate unit} \\
&\quad (0.0769 \text{ insulin units per gram})
\end{aligned}
$$

$$
\begin{aligned}
\text{RapidCalc} &= \frac{6.6 - 5.5}{2.0} + (0.769 \times 6) \\
&= 5.0 \text{ insulin units (to the nearest half unit)}
\end{aligned}
$$

$$
\begin{aligned}
\text{Diabetes Personal Calculator} &= \frac{6.6 - 5.5}{2.0} + \frac{60}{13} \\
&= 5.0 \text{ insulin units (to the nearest half unit)}
\end{aligned}
$$

$$
\begin{aligned}
\text{Diabetic Dosage (published)} &= \frac{6.6 - 5.5}{2.0 + 6} \\
&= 0.0 \text{ insulin units (to the nearest half unit)}
\end{aligned}
$$

$$
\begin{aligned}
\text{Diabetic Dosage (implemented)} &= \frac{6.6 - 5.5}{2.0} + 6 \\
&= 6.5 \text{ insulin units (to the nearest half unit)}
\end{aligned}
$$

Figure 2.6: Diabetic Dosage calculation comparison

Figure 2.6 demonstrates that the version of the calculation published on the app's websites is incorrect. The implemented version relies on the user having a carbohydrate-to-insulin ratio of 1 carbohydrate portion to 1 insulin unit, which as shown can have a large bearing on the outcome.

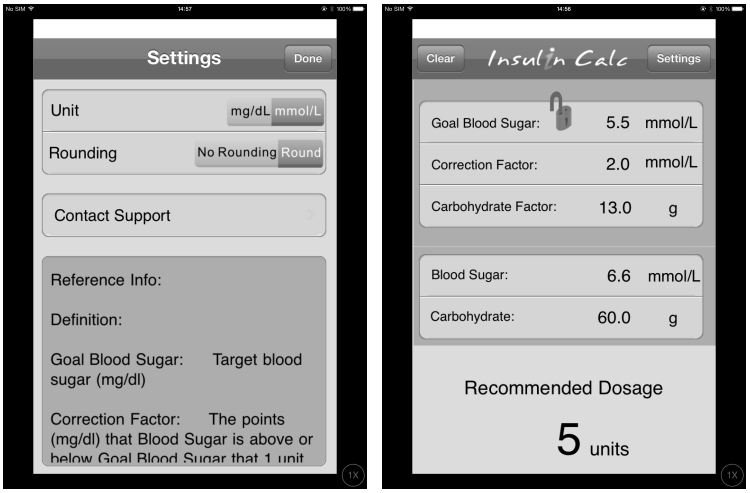
The story behind the development of Diabetic Dosage can be found on the app's website which helps to explain the differences discovered. Development of the application was motivated by a sister who wanted to help her brother with calculating bolus insulin doses. Previously a chart was used to obtain a bolus insulin suggestion based on the brother's preprandial blood glucose reading and carbohydrates in the meal or snack. This resulted in the sister taking the chart and designing a mobile application to help her brother and others with bolus insulin suggestions. This story helps explain why the calculation is different to the other apps, as the information used to derive the formula was tailored to her brother. However, this story illustrates how mobile technology can be used to simplify obtaining bolus advice and removing the need to manually sift through tables of numbers.

2.2.4 Insulin Calc

Insulin Calc [Insulin Calc, 2013] is the last of the mobile apps reviewed in this research. The app is similar to Diabetic Dosage in terms of simplicity but does allow for a customisable target blood

glucose value. The app does not record data and instead works as a quick calculator. To reduce the time taken to perform a calculation the app allows the target blood glucose level, insulin sensitivity factor and carbohydrate factor to be stored and locked to prevent accidental change. Once these three inputs are defined and saved, the user only needs to input their preprandial blood glucose level and planned carbohydrate intake at the time of calculation.

The settings view of the application allows the user to specify which blood glucose unit to use (mmol/L or mg/dL) and whether the suggested bolus dose should be rounded to whole units or half units (Fig. 2.7a). The view also provides information about the application including how a bolus suggestion is calculated.



(a) User settings (b) Calculator

Figure 2.7: Insulin Calc application for iOS

To perform a successful bolus calculation, the user must firstly define their target blood glucose level, insulin sensitivity factor, and carbohydrate-to-insulin ratio (Fig. 2.7b). This information is then used alongside the two calculation specific inputs of planned carbohydrate intake and preprandial blood glucose reading to produce a bolus suggestion. The formula used by the application is specified in Eq. 2.12, let pbg be the preprandial blood glucose level, tbg be the defined target blood glucose level, isf be the blood glucose reduction per unit of insulin, c be the planned carbohydrate intake, and ci be the carbohydrates to correct one unit of insulin.

$$\text{suggested bolus dose} = \frac{pbg - tbg}{isf} + \frac{c}{ci} \tag{2.12}$$

The app updates the recommended dose when any input field is amended and does not require all parameters to be defined prior to displaying a bolus insulin suggestion. This raises safety concerns as the subject may use the bolus advice prior to defining all the parameters required for an accurate dose to be calculated. Insulin Calc's main drawback is the inability to factor active insulin into the suggestion. In cases where there is no active insulin, the application replicates the behaviour of the Accu-Chek® Aviva Expert, and such results can be considered reliable.

2.2.5 Summary of the mobile apps

The mobile apps described in this section all provide a means to calculate bolus insulin doses. However, it can be seen that the sophistication of the calculation and the features provided by the apps vary drastically. A comparison of the features in the mobile apps and the Accu-Chek® Aviva Expert are shown in Table 2.2. The column headers in this table are abbreviations denoted as follows: AE - Accu-Chek® Aviva Expert, RC - RapidCalc, DPC - Diabetes Personal Calculator, DD - Diabetic Dosage, IC - Insulin Calc.

	AE	RC	DPC	DD	IC
Carbohydrate intake	✓	✓	✓	✓	✓
Preprandial blood glucose	✓	✓	✓	✓	✓
Target blood glucose	✓	✓	✓		✓
Insulin sensitivity factor	✓	✓	✓	✓	✓
Time period variations	✓	✓			
Carbohydrate-to-insulin ratio	✓	✓	✓		✓
Time period variations	✓	✓	✓		
Insulin on board	✓	✓	✓		
Exercise	✓	✓			
Record meals	✓	✓	✓		
Configurable settings	✓	✓	✓		
Data visualisation	✓	✓			

Table 2.2: Comparison of features present in the state-of-the-art mobile apps and Accu-Chek® Aviva Expert

The most sophisticated of the apps is RapidCalc, which uses a bolus calculation formula similar to the FDA approved Accu-Chek® Aviva Expert blood glucose meter with the addition of factors such as alcohol, stress, and automated active insulin consideration, which the others apps do not. Additionally, RapidCalc provides visualisations of data recorded and a large range of customisable settings.

Diabetes Personal Calculator lacks some of the more sophisticated features of RapidCalc such as data visualisation, alcohol, stress and exercise. However, Diabetes Personal Calculator still provides more functionality than Insulin Calc and Diabetic Dosage by allowing active insulin to be calculated, records to be stored, and configurable settings. A major pitfall of Diabetes Personal Calculator is the limitation on how many meals can be recorded each day, which reduces the flexibility of the app.

Insulin Calc is much more limited than both RapidCalc and Diabetes Personal Calculator, only allowing a simple bolus calculation to be performed with no functionality for recording meal information. Despite the simplistic nature of Insulin Calc, the formula mirrors that used by the FDA approved Accu-Chek® Aviva Expert with the exclusion of active insulin consideration. This provides some reassurance that the app itself is capable of producing reasonable advice, but could still be improved.

At the low end of the sophistication scale is Diabetic Dosage, an app which simply allows

bolus calculation with no means to record the information. Diabetic Dosage enforces a constant target blood glucose level in the calculation, a factor which is likely to vary from subject to subject, exposing users to inaccurate bolus calculations. This app also raised concerns due to the associated website publishing the formula used by the app, which would result in incorrect bolus calculations and is inconsistent with all the other products assessed in this research. Fortunately, the formula implemented by the app does perform the calculation correctly, but is limited to a static target blood glucose level as previously stated.

In summary, it can be concluded that RapidCalc is the benchmark for determining the features which should be included in the app developed by this research. All the other other apps assessed contain less features, providing no note-worthy contributions above RapidCalc. Below is a list of considerations that will be used when designing the app:

- Produce bolus advice, to tailor the app to the user. Case-based reasoning (Chapter 3 - 5) will replace the formulas described by these apps in order to allow the app to learn and improve bolus advice over time.
- The ability to store meal records.
- Provide a comprehensive set of customisable settings to ensure the app can be tailored to all subjects.
- Cater for both the European and US market by providing a suitable selection of units.
- Enforce constraints on user input to reduce the chance of errors and prevent potentially dangerous advice.
- Provide a means to view previous records.
- Visualise previous records to help the user identify trends and spot outliers.

To ensure the understanding of the domain, data types and functionality of existing apps, a formal specification will now be devised to model the current state-of-the-art approaches.

2.3 Formal specification

In this section a formal specification is defined from the features and constraints discussed earlier in this chapter. The specification sets out to define the restrictions on user inputs and settings to aid the safety of the app. Formal methods provide a mathematical framework for software engineering, and help to ensure that the resulting system is correct through modelling the system prior to implementation.

The Event-B language [Abrial, 2010] was chosen to create the formal specification for the mobile app. An initial formal specification was produced by reverse engineering the Diabetes Personal

Calculator application for iOS to learn and understand the Event-B language and its integrated development environment (IDE) the Rodin Platform [Abrial et al., 2010]. This work was presented (Appendix F) at the 2012 DEPLOY Federated Event to evaluate the Rodin Platform from a new user's perspective; with the conclusion that the tool, language and available resources simplified the process of formalising the specification [Brown et al., 2012].

The formal specification included in this research is primarily based on the functionality of the Accu-Chek® Aviva Expert as opposed to Diabetes Personal Calculator. This is due to approval of the device by the FDA, which provides reassurance that the constraints and formulas used by the device are safe.

2.3.1 Introduction to formal methods and Event-B

The use of formal methods in industry is generally limited to safety critical systems as the software industry as a whole has yet to embrace its use [Woodcock et al., 2009]. Any application designed for medical use should be considered safety-critical and developed using a formal approach. Woodcock et al. [2009] state that that lack of evidence supporting the cost-benefit of formal methods is a fundamental reason why they are not widely adopted in industry. This is in addition to the learning curve required for its adoption. Snook and Harrison [2001] state that the adoption of formal methods in the long-term does not have a negative implication on project costs and any increased time spent on correctly modelling and specifying the system is balanced out by a reduction in the resources required for testing and debugging.

A significant effort to promote the adoption of formal methods has been undertaken by the RODIN (2004-2007) project, later funded by the DEPLOY (2008-2012) and ADVANCE (2011-2014) projects. This project led to the development of the Rodin Platform, an Eclipse-based IDE for Event-B modelling. Alongside the IDE, the project resulted in a repository of resources to aid existing and new users of the platform. The resources available include a selection of plug-ins designed to aid the modelling process. Most notably, a plug-in to integrate ProB into the IDE. ProB provides a method to systematically check the model for errors through animation. The animation process checks that the state of the model does not violate the invariants. In addition, animation provides a method to visualise the activation of events, and the result the event actions have on the state of model. Other plug-ins include automated code generation from the Event-B specification and external provers to assist the automatic proofs.

Event-B is a formal modelling language developed by Jean-Raymond Abrial as an evolution of the B-Method [Abrial, 2010; Abrial et al., 1991]. The purpose of modelling with Event-B is to prove that a model will work prior to implementation and for early identification of problems with the system requirements. Event-B models are broken down into two main components *contexts* and *machines*. Contexts define constants of the model and use *axioms* to impose restrictions on the

constants. These constants can include the definition of numerical values and sets. Machines model the dynamic behaviour of the system. *Variables* represent the dynamic properties of the machine's state and can be altered by *events* within the machine. The behaviour allowed by the machine is defined by *invariants*, and in order for the machine to successfully *prove* these invariants must not be violated by changes to the machine's state. The events allow for changes to be made to the state of the machine and consist of *parameters*, *guards* and *actions*. The parameters define values used by the event, providing a means to define event inputs. The guards express the conditions which must be true to allow the event to be enabled and are used to preserve the invariant of the machine. Adequate guards will prevent the event from altering the machine state in such a way which would violate the invariants. If a machine is enabled, the actions defined by the machine can be performed. These actions allow the state of the machine to be changed through the update of variables.

An Event-B model is validated through the successful discharge of *proof obligations*, which imply that the invariants and type declarations of the model are not violated by any aspect of the model. For example, if a variable x is of type $\mathbb{N}1$, the assignment of 0 to x would not discharge the proof obligation created by $x \in \mathbb{N}1$, since x must be greater than or equal to 1. The default prover provided by the Rodin Platform often struggles with relatively simple proofs and it is recommended that the Atelier B prover plug-in is installed to aid automatic proofs.

A key feature of the Event-B language is its incremental modelling process through *refinement*. Refinement allows gradual development of the specification starting at an abstract level, which can then be refined to include new functionality and gradually move the model towards the concrete implementation. This type of refinement is known as horizontal refinement [Abrial, 2010]. There is a second type of refinement called vertical refinement to transforms the specification into a format which is easier to implement, and occurs after the completion of horizontal refinement.

2.3.2 Event-B example

Before formally specifying the bolus calculator a small example is presented. This example will model a basic banking system and demonstrates some of the concepts introduced in Section 2.3.1.

To begin with the context of the banking system needs to be defined. The system will consist of two custom sets $PERSON$ and $ACCOUNT$ which are defined in a context called bankc0 (Fig. 2.8). These sets describe all possible people and all possible accounts respectively.

CONTEXT bankc0
SETS
    PERSON
    ACCOUNT
END

Figure 2.8: Bank context bankc0

With the context defined a new machine is introduced which sees this context. In the machine variables of the system defined, alongside its initialisation and events. The variables of the banking system will reflect the accounts owned by people.

Firstly, a variable *account* is defined, which is a subset of all possible accounts ($ACCOUNT$). This denoted by describing a new invariant which must be preserved by the system. In this case the standard subset notation is used to say that *account* is a subset of $ACCOUNT$: $account \subseteq ACCOUNT$.

Secondly, a variable *owner* is introduced to map the account to a person, which is done using a total function from *account* to $PERSON$: $owner \in account \rightarrow PERSON$.

Thirdly, a variable *balance* is introduced to map the account to their bank balance. In this model the money in their account (the balance) will be a natural number ($\mathbb{N}$). *balance* is defined as a total function from *account* to $\mathbb{N}$: $balance \in account \rightarrow \mathbb{N}$

Any variables defined by the machine need initialising, this done using the special event called 'Initialisation'. In this case the initialisation is assumed to be a fresh start, so all variables have the initial value of the empty set ($\varnothing$). All assignments in Event-B are done using the := notation, and will also be seen in the event actions later. Figure 2.9 shows the machine called bankm0, with the variable definitions just described as well as their their initialisation.

MACHINE bankm0
SEES bankc0
VARIABLES
`account`
`owner`
`balance`
INVARIANTS
`inv1` : $account \subseteq ACCOUNT$
`inv2` : $owner \in account \rightarrow PERSON$
`inv3` : $balance \in account \rightarrow \mathbb{N}$
EVENTS
Initialisation
begin
`act1` : $account := \varnothing$
`act2` : $owner := \varnothing$
`act3` : $balance := \varnothing$
end

Figure 2.9: Bank machine bankm0, variables and initialisation

Now the context, variables, invariants and initialisation of the machine are defined some events can be added. The first event defined is $OPEN\_ACCOUNT$ (Fig. 2.10) which provides the ability to open a new account, since a person cannot perform any tasks until they create an account. To perform this event some parameters must be specified. Firstly the person p who is a member of the set $PERSON$ ($p \in PERSON$), and secondly a new account a which is a member of the set $ACCOUNT$ but does not already exist in the variable *account* ($a \in ACCOUNT \land a \notin account$).

With the parameters and guards in place, the actions of the event are defined. The new account a needs to be added to *account* using the set union ($\cup$) operator ($account := account \cup \{a\}$). It is important to note that all set operators can only be applied to sets, since a is not a set it must be represented as if it is a set by using the curly brackets $\{a\}$. This assignment alone will violate inv2 and inv3 shown in Figure 2.9, since both *owner* and *balance* are total functions with the domain *account*, meaning that all elements of *account* must be included. To resolve this *owner* and *balance* must also be updated. In the case of *owner*, the function maps the account to the person who owns the account. The new account is defined by parameter a and the person owning the account p, resulting in the assignment $owner(a) := p$. This operation will override the range of the total function, changing the value. The same is done to *balance*, but this time the new account a maps to a $\mathbb{N}$. Since the account will have no money in it, the assignment will map a to 0 ($balance(a) := 0$).

The full $OPEN\_ACCOUNT$ event is shown in Fig. 2.10, where the *any* section defines the parameters, the *where* section defines the event guards, and the *then* section performs any actions should the guards not be violated.

EVENTS
Event $OPEN\_ACCOUNT \mathrel{\widehat{=}}$
any
p
a
where
`grd1` : $p \in PERSON$
`grd2` : $a \in ACCOUNT$
`grd3` : $a \notin account$
then
`act1` : $account := account \cup \{a\}$
`act2` : $balance(a) := 0$
`act3` : $owner(a) := p$
end

Figure 2.10: Bank machine bankm0, open account event

Now the ability to open a new account is included in the model, a new event can be defined to close the account $CLOSE\_ACCOUNT$ (Fig. 2.11). This event will reverse the process of $OPEN\_ACCOUNT$, removing an account specified by parameter a from the machine variables *account*, *owner* and *balance*. To allow the event to be active the account a must be a member of *account*, which in turn means it is in the domain of *owner* and *balance* due to these variables being total functions. Another guard $balance(a) = 0$ is also added to this event in order to prevent the account being closed if it stills contains money, where $balance(a)$ will return the balance for account a.

The first action described by the event will remove a from *account* using the set subtraction ($\setminus$) operator ($account := account \setminus \{a\}$). The *owner* and *balance* machine variables must also be updated to close the account. To remove from these mappings the domain subtraction operator

($\lhd$) is used (e.g. $owner := \{a\} \lhd owner$). The $CLOSE\_ACCOUNT$ (Fig. 2.11) event will now remove account a whilst successfully preserving the machine invariants and ensuring the account balance is 0 prior to closure.

EVENTS
Event $CLOSE\_ACCOUNT \mathrel{\widehat{=}}$
any
a
where
grd1 : $a \in account$
grd2 : $balance(a) = 0$
then
act1 : $account := account \setminus \{a\}$
act2 : $owner := \{a\} \lhd owner$
act3 : $balance := \{a\} \lhd balance$
end

Figure 2.11: Bank machine bankm0, open account event

The model so far allows a person to open and close an account but not perform any useful tasks such as depositing and withdrawing money. To include this functionality two new events are added $DEPOSIT$ (Fig. 2.12) and $WITHDRAW$ (Fig. 2.13), which will increase and reduce the account balance respectively.

The $DEPOSIT$ event (Fig. 2.12) requires two parameters, the account a which is in the set $account$, and the money m to be added to the account's balance which is a $\mathbb{N}$. Since this event only changes the account balance, the only action required is an update to $balance$ for account a to the value of $balance(a) + m$.

EVENTS
Event $DEPOSIT \mathrel{\widehat{=}}$
any
a
m
where
grd1 : $a \in account$
grd2 : $m \in \mathbb{N}$
then
act1 : $balance(a) := balance(a) + m$
end

Figure 2.12: Bank machine bankm0, deposit event

The $WITHDRAW$ event (Fig. 2.13) is almost identical to $DEPOSIT$ with the same parameters, but instead of updating the account balance $balance(a)$ to increase by m, $balance(a)$ is reduced by m. Additionally a new guard is added stating that the current account balance must be greater than or equal to the withdrawal amount m to prevent the balance going below 0.

EVENTS
Event *WITHDRAW* $\widehat{=}$
any
a
m
where
`grd1` : $a \in account$
`grd2` : $m \in \mathbb{N}$
`grd3` : $balance(a) \geq m$
then
`act1` : $balance(a) := balance(a) - m$
end

Figure 2.13: Bank machine bankm0, withdraw event

The specification described so far describes a simple banking system where people can open a new account, and if they have an account can deposit, withdraw, and close the account. The example has so far demonstrated the use of a context to define custom types, and machines which provide the dynamic aspects of the model. Additionally several common aspects of Event-B modelling have been described including the creation of variables, invariants, initialisation and events. To conclude the example a simple refinement of the system will be done to add two different types of account.

Refinement prevents the need for creating a complex system in one go, and allows functionality to be gradually added. Any refined machine will inherit all aspects of the parent machines, meaning that the refined machine must not violate the invariants of the parent machines. In this refinement two account types will be defined *current* and *savings* which will be assigning to an *account* by a new total function *type*. The $OPEN\_ACCOUNT$ and $CLOSE\_ACCOUNT$ events will be refined to cater for this new functionality.

To begin this refinement a new context *bankc*1 (Fig. 2.14) is created which extends *bankc*0. This context will define an enumerated set called $TYPE$ with the constant elements *current* and *savings*. This is achieved by defining $TYPE$ as a new set, *current* and *savings* as constants, and adding an axiom which states that $TYPE$ is a partition containing the two constants.

CONTEXT bankc1
EXTENDS bankc0
SETS
`TYPE`
CONSTANTS
`normal`
`savings`
AXIOMS
`axm1` : $partition(TYPE, \{normal\}, \{savings\})$
END

Figure 2.14: Bank context bankc1

A new machine is now created called *bankm*1 (Fig. 2.15) which refines the machine *bankm*0. Firstly a new variable *type* is defined to create a relationship between the account and its type. This done in the same way as *owner* and *balance* through an invariant describing *account* to $TYPE$ as a total function ($type \in account \rightarrow TYPE$). Additionally, *type* is initialised in the refined machine as an empty set.

MACHINE bankm1
REFINES bankm0
SEES bankc1
VARIABLES
`account`
`owner`
`balance`
`type`
INVARIANTS
`inv1` : $type \in account \rightarrow TYPE$
EVENTS
Initialisation
extended
begin
`act1` : `account` $:= \varnothing$
`act2` : `owner` $:= \varnothing$
`act3` : `balance` $:= \varnothing$
`act4` : $type := \varnothing$
end

Figure 2.15: Refined bank machine bankm1, variables, invariant and initialisation

The addition of the new variable *type* and its associated invariant will result in the machine not successfully proving. To rectify this the $OPEN\_ACCOUNT$ and $CLOSE\_ACCOUNT$ events must be refined so that they add and remove from *type* respectively. For the $OPEN\_ACCOUNT$ (Fig. 2.16) event a new parameter t of type $TYPE$ is specified. Additionally, the action of the event will assign t to $type(a)$.

EVENTS
Event $OPEN\_ACCOUNT \mathrel{\widehat{=}}$
extends $OPEN\_ACCOUNT$
any
p
a
t
where
grd1 : $\mathtt{p} \in \mathtt{PERSON}$
grd2 : $\mathtt{a} \in \mathtt{ACCOUNT}$
grd3 : $\mathtt{a} \notin \mathtt{account}$
grd4 : $t \in TYPE$
then
act1 : $\mathtt{account} := \mathtt{account} \cup \{\mathtt{a}\}$
act2 : $\mathtt{balance}(\mathtt{a}) := 0$
act3 : $\mathtt{owner}(\mathtt{a}) := \mathtt{p}$
act4 : $type(a) := t$
end

Figure 2.16: Refined bank machine bankm1, open account event

For the refined $CLOSE\_ACCOUNT$ (Fig. 2.17) event no new parameters are required, but the account must be removed from the domain of *type*.

EVENTS
Event $CLOSE\_ACCOUNT \mathrel{\widehat{=}}$
extends $CLOSE\_ACCOUNT$
any
a
where
grd1 : $\mathtt{a} \in \mathtt{account}$
grd2 : $\mathtt{balance}(\mathtt{a}) = 0$
then
act1 : $\mathtt{account} := \mathtt{account} \setminus \{\mathtt{a}\}$
act2 : $\mathtt{owner} := \{\mathtt{a}\} \lhd \mathtt{owner}$
act3 : $\mathtt{balance} := \{\mathtt{a}\} \lhd \mathtt{balance}$
act4 : $type := \{a\} \lhd type$
end

Figure 2.17: Refined bank machine bankm1, close account event

A refinement can also provide additional functionality to the machine. To demonstrate this a new event to change the account's type called $CHANGE\_TYPE$ (Fig. 2.18) is added to the machine $bankm1$. This event requires two parameters, the account in question a and the new type of the account t. Since there is little point in changing the account type to the same type, a guard is included stating that the account's current type must not be the same as the new type ($type(a) \neq t$).

EVENTS
Event $CHANGE\_TYPE \mathrel{\widehat{=}}$
any
a
t
where
grd1 : $a \in account$
grd2 : $t \in TYPE$
grd3 : $t \neq type(a)$
then
act1 : $type(a) := t$
end

Figure 2.18: Refined bank machine bankm1, change account type event

This concludes the introductory Event-B example which covers the fundamental aspects of the notation. For further examples please refer to Abrial's book 'Modeling in Event-B: system and software engineering' [Abrial, 2010]. In the next section Event-B is used to formally specify a bolus calculator, a task which enables a better understanding of the problem and will also aid the implementation of the mobile app for bolus decision support.

2.3.3 Formal specification of a bolus calculator

The specification of the bolus calculator begins by defining type and invariant conditions for the variables of the system alongside abstract events for configuring the system and the bolus calculation. The variables and constraints used by the specification are derived from those presented by the Accu-Chek® Aviva Expert in Section 2.1. The machine will start at an abstract level and be refined twice to include new functionality to the model. The full machine specifications can be found in Appendices A-C.

The bolus calculation used in this specification is based on that defined by the Accu-Chek® Aviva Expert. The calculation has been adjusted to omit meal rise as no detailed information is available in the documentation, and is not used in the formulas presented by the mobile apps or that presented in research by Herrero et al. [2014]. Active insulin has also been changed to be deducted from the total bolus sum as opposed to inclusion in the correction dose component. This decision is taken as the Accu-Chek® Aviva Expert provides no detailed information on how to calculate the sum of the blood glucose range covered by active insulin. This change also reflects the formula presented Herrero et al. [2014] and the RapidCalc app. The bolus formula used in the specification is presented in Eq. 2.13, let cd be the correction dose, md be the meal dose, and a be the active insulin.

$$\text{bolus suggestion} = \begin{cases} cd + md - a, & \text{if } cd + md - a > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.13}$$

The correction dose cd for Eq. 2.13 is calculated by multiplying the user's insulin sensitivity factor isf by the difference in current blood glucose cbg and the user's target blood glucose level tbg (Eq. 2.14).

$$cd = (cbg - tbg) \times isf \tag{2.14}$$

The target blood glucose tbg (Eq. 2.15) is calculated from the average of the user's upper target blood glucose range level tru and the lower target blood glucose range level trl. The insulin sensitivity factor isf (Eq. 2.16) is the reduction in blood glucose $isfbg$ per a defined number of insulin units $isfi$.

$$tbg = \frac{tru + trl}{2} \tag{2.15}$$

$$isf = \frac{isfi}{isfbg} \tag{2.16}$$

The meal dose md for Eq. 2.13 is calculated by multiplying the planned carbohydrate intake c in grams by the carbohydrate-to-insulin ratio cir (Eq. 2.17).

$$md = c \times cir \tag{2.17}$$

The carbohydrate-to-insulin ratio cir required to calculate the meal dose is defined as the quantity of carbohydrates in grams crc covered by a defined number of insulin units cri (Eq. 2.18).

$$cir = \frac{cri}{crc} \tag{2.18}$$

Finally, the active insulin a is the insulin which remains active from all recorded prior bolus insulin doses. Equation 2.19 [Campbell and Abramovich, 2012] defines the active insulin calculation, let C be a sequence of cases c, ci be the previous bolus insulin dose at the time ct in minutes of the case c, t be the time of the current calculation in minutes, and at be the active insulin duration in

minutes.

$$a = \sum_{c \in C} \begin{cases} ci \times \left(1 - \dfrac{t - ct}{at}\right), & \text{if } at > t - ct > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.19}$$

2.3.3.1 Abstract machine

The specification begins with an abstract specification of the system which focuses on the constraints of the system variables. Figure 2.19 provides an extract of the variables and invariants that includes the maximum bolus dose, and upper and lower target blood glucose ranges. A full specification of the variables and invariants are included in Appendix A due to the number present. Event-B is limited to natural and integer numerical types, which presents a problem when decimal values are required. To overcome this, all natural numbers or integers in the specification are represented through multiplication by the power of 10, allowing precision to one decimal place (e.g. 5.5 is represented as 55). Comments in the specification will be used help to reinforce this representation.

MACHINE t1dm_m0
VARIABLES
`maxBolus` Maximum bolus dose limit
`targetRangeUpper` Target blood glucose range upper value
`targetRangeLower` Target blood glucose range lower value
INVARIANTS
`inv21` : $maxBolus \in \mathbb{N}$
`inv22` : $maxBolus \leq 500$
The maximum bolus dose must be in the range [0.0,50.0] IU
`inv23` : $targetRangeUpper \in \mathbb{N}$
`inv24` : $targetRangeUpper \geq 55 \wedge targetRangeUpper \leq 150$
Target blood glucose range upper value must be in the range [5.5,15.0]
`inv25` : $targetRangeLower \in \mathbb{N}$
`inv26` : $targetRangeLower \geq 30 \wedge targetRangeLower \leq 80$
Target blood glucose range upper value must be in the range [3.0,8.0]

Figure 2.19: Extract of machine variables and invariants

Figure 2.19 defines the types of *maxBolus*, *targetRangeUpper* and *targetRangeLower* as that of a natural number ($\mathbb{N}$). This invariant alone implies that the value of these variables must be greater than or equal to 0, and that negative values are not permitted. *inv22* states that *maxBolus* is restricted to a maximum bolus value of 50.0. Both of the invariants for *maxBolus* state that $0 \leq maxBolus \leq 50$ insulin units. Additionally, the invariants for *targetRangeUpper* and *targetRangeLower* also define the constraints: $5.5 \leq targetRangeUpper \leq 15.0$ mmol/L, and $3.0 \leq targetRangeLower \leq 8.0$ mmol/L. These invariants prevent the variables from breaching the constraints imposed by the Accu-Chek® Aviva Expert blood glucose meter.

Variables must be initialised by the machine using a special event called *initialisation*. The

initialisation event defines the initial state of the machine, this state must comply with the invariants of the machine in order to successfully discharge all the proof obligations. Figure 2.20 demonstrates the initialisation for *maxBolus*, *targetRangeLower*, and *targetRangeHigher*. The initial values are derived from those specified by the Accu-Chek® Aviva Expert default values. The full initialisation can be found in Appendix A.

MACHINE t1dm_m0
EVENTS
Initialisation
Initilialises the machine
begin
`act11` : $maxBolus := 10$
Default maximum bolus suggestion
`act12` : $targetRangeUpper := 80$
Default target blood glucose range upper value
`act13` : $targetRangeLower := 40$
Default target blood glucose range lower value
end

Figure 2.20: Extract of machine initialisation

With the variables and invariants of the machine specified and initialised, the events to model the bolus calculation can be added. The event called *bolusCalc* is initially included as an abstract event, which updates a variable *bolusSuggestion* and ensures it does not violate the invariant.

The specification for the *bolusSuggestion* states that it must be greater than or equal to 0, and must be less than the maximum blood glucose value:

$$bolusSuggestion \in \mathbb{N} \land bolusSuggestion \leq maxBolus$$

An abstract event for the bolus suggestion *bolusCalc* is described by Figure 2.21. One parameter is specified for the event, the bolus suggestion s. The event has two guards which check that parameter s is a natural number and that s is less than or equal to the maximum bolus dose $maxBolus$.

The *bolusCalc* event does not satisfy the requirements of the application alone as it is possible that the bolus suggestion may be negative or greater than the maximum bolus dose. In these circumstances, the bolus dose should be set to 0 and the *maxBolus* respectively. To model this, two new events are added to the model to allow for these circumstances *bolusCalcNeg* and *bolucCalcMax*. These events are shown in Fig. 2.22.

MACHINE t1dm_m0
EVENTS
Event *bolusCalc* $\widehat{=}$
Bolus calculator event for instances where s is > 0 and $<$ maxBolus
any
s
s is the parameter of bolus suggestion
where
`grd1` : $s \in \mathbb{N}$
`grd2` : $s \leq maxBolus$
The bolus solution suggestion is $\leq$ maxBolus
then
`act1` : $bolusSuggestion := s$
Sets the bolus suggestion value to parameter s
end

Figure 2.21: Abstract bolus calculation event

MACHINE t1dm_m0
EVENTS
Event *bolusCalcNeg* $\widehat{=}$
Bolus calculator event for instances where s is < 0
any
s
s is the parameter of bolus suggestion
where
`grd1` : $s \in \mathbb{Z}$
`grd2` : $s < 0$
The bolus solution suggestion is < 0
then
`act1` : $bolusSuggestion := 0$
Sets the bolus suggestion to 0 as negative values are not permitted
end
Event *bolusCalcMax* $\widehat{=}$
Bolus calculator event for instances where s is $>$ maxBolus
any
s
s is the parameter of bolus suggestion
where
`grd1` : $s \in \mathbb{N}$
`grd2` : $s > maxBolus$
The bolus solution suggestion is $>$ the maximum bolus dose
then
`act1` : $bolusSuggestion := maxBolus$
Sets the bolus suggestion to maxBolus as values $>$ maxBolus are not permitted
end

Figure 2.22: Additional abstract bolus calculation events

An additional event $defineSettings$ to perform modifications to the system variables is also defined in the machine and can been seen in Appendix A. Including these events concludes an abstract model of the bolus calculator which will now be refined into a more concrete model. At present, the model does not reflect the calculation to be performed but instead defines abstract the

events required. All proofs are automatically discharged by the prover and animation with ProB indicates that the events are only activated by parameters which satisfy the guards.

2.3.3.2 First refinement

The first refinement of the model will change the parameter s of the bolus calculation events to the result of the bolus calculation formula. This requires the introduction of three variables not defined in the first machine: planned carbohydrate intake *carbs*, preprandial blood glucose reading *preBG*, and the active insulin *activeInsulin*.

The specification of the Accu-Chek® Aviva Expert states that the carbohydrate value has the range [0,240]. A blood glucose range is not specified, but it is known that this value cannot be negative. A maximum value for the blood glucose range is derived from the maximum value allowed for insulin sensitivity of 55.4 mmol/L. These variables could be included as parameters in the events, however to ensure consistency and to prevent activation of multiple calculation events during animation, these will be defined as variables of the machine. In this refinement active insulin will be modelled at an abstract level with an event that will allow the active insulin to be set to a natural number.

MACHINE t1dm_m1
VARIABLES
`carbs` Carbohydrates for the new problem
`preBG` Preprandial blood glucose reading for the new problem
`inputsDefined` Used to control the activation of bolus calculator events
`activeInsulin` The quantity of insulin units current active in the subject
INVARIANTS
`inv1` : $carbs \in \mathbb{N}$
`inv2` : $carbs \leq 2400$
Carbohydrate input must be $\leq$ 240 grams and $\geq$ 0 grams
`inv3` : $preBG \in \mathbb{N}$
`inv4` : $preBG \leq 554$
Preprandial blood glucose input must be $\leq$ 55.4 mmol/L and $\geq$ to 0 mmol/L
`inv5` : $inputsDefined \in BOOL$
inputsDefined informs the machine carbs and preBG have been set.
`inv6` : $activeInsulin \in \mathbb{N}$ Active insulin is $\geq$ to 0.
EVENTS
Initialisation
extended
Initilialises the machine
begin
`act16` : $carbs := 0$ Default carbohydrate input set to 0 grams.
`act17` : $preBG := 60$ Default preprandial blood glucose input set to 6.0 mmol/L.
`act18` : $inputsDefined := FALSE$ Inputs have not been defined by the user.
`act19` : $activeInsulin := 0$ Default active insulin set to 0 insulin units.

Figure 2.23: Extract of the variables and invariants from the first machine refinement

Figure 2.23 displays the declaration and initialisation of the new variables with the new invari-

ants to comply with the specification. In addition, a Boolean variable $inputsDefined$ is declared, which will prevent the bolus calculation events from activating unless the inputs required for the bolus calculation have been defined. A new guard will be added to the bolus calculator events to ensure they cannot be activated unless $inputsDefined$ is $TRUE$.

The bolus calculator events are now updated to define the concrete calculation. A new guard is added which states that the user inputs must be defined before the event can become active. The parameter s from the original abstract machine also includes a new guard stating that the value of s must be equal to the result of the calculation. The *bolusCalc* event is shown in Fig. 2.24, *bolusCalcNeg* and *bolusCalcMax* contain the same changes in this refinement.

MACHINE t1dm_m1
EVENTS
Event *bolusCalc* $\widehat{=}$
Bolus calculator event for instances where the result is > 0 and $\leq$ maxBolus
extends *bolusCalc*
any
`s`
where
`grd1` : $\texttt{s} \in \mathbb{N}$
`grd2` : $\texttt{s} > 0$ The bolus solution is > 0
`grd3` : $\texttt{s} \leq \texttt{maxBolus}$
The bolus solution suggestion is $\leq$ maxBolus
`grd4` :
$s = (preBG - (targetRangeLower + (targetRangeUpper - targetRangeLower)/2)) * (isfI/isfBG) + carbs * (carbRatioI/carbRatioC) - activeInsulin$
The bolus solution is equal to the calculation formula
`grd5` : $inputsDefined = TRUE$ The user inputs must be defined first
then
`act1` : $\texttt{bolusSuggestion} := \texttt{s}$
Sets the bolus suggestion value
end

Figure 2.24: Bolus calculation event in the first machine refinement

To activate the bolus calculation events *bolusCalc*, *bolusCalcNeg*, and *bolusCalcMax*, a new event is required to define the user inputs. The parameters for this event must not violate the invariants imposed on *carbs* and *preBG*. The resulting event $defineInput$ is shown in Fig. 2.25. The event updates the state of the machine to set new values for the machine variables *carbs* and *preBG*. In addition, the event sets $inputsDefined$ to $TRUE$, allowing the updated bolus calculator events to be activated if the other guard conditions are also met.

MACHINE t1dm_m1
EVENTS
Event *defineInput* $\widehat{=}$
Modifies user inputs for the bolus calculation
any
c Planned carbohydrate intake
bg Preprandial blood glucose level (mmol/L)
where
grd1 : $c \in \mathbb{N}$
grd2 : $c \leq 2400$
Carbohydrate value must be *leq* to 240 grams and ≥ 0 grams
grd3 : $bg \in \mathbb{N}$
grd4 : $bg \leq 554$
Carbohydrate value must be *leq* to 55.4 mmol/L and ≥ 0 grams
then
act1 : $carbs := c$
act2 : $preBG := bg$
act3 : $inputsDefined := TRUE$ The user inputs have been defined
end

Figure 2.25: Event to define user inputs

The final inclusion in the first refinement is an event (Fig. 2.26) to update the variable *activeInsulin* at an abstract level. This event sets *activeInsulin* to be equal to any natural number defined by the parameter a, and is refined to a concrete event in the second refinement.

MACHINE t1dm_m1
EVENTS
Event *calcActiveInsulin* $\widehat{=}$
Calculates active insulin
any
a Active insulin
where
grd1 : $a \in \mathbb{N}$ Active insulin must be $\geq$ to 0
then
act1 : $activeInsulin := a$
end

Figure 2.26: Abstract active insulin event

2.3.3.3 Second refinement

In order to model active insulin, there needs to be a record of previous cases. To incorporate this into the model, a formal representation of past cases is required. The case-base will be represented as total functions from a natural number representing the case id to a feature of the case. The use of a total function implies that members of *caseBase* must also be a member of the function's domain. The case features that will be stored in this specification include the input variables of carbohydrates and preprandial blood glucose reading. In addition to calculate active insulin, the used bolus dose and time of the dose also need to be recorded. This requires the new machine

variable of *time* for the current problem to be included, and the refinement of the $defineInput$ event to allow *time* to be modified. Additionally a new Boolean variable *solutionObtained* is defined which must be $TRUE$ to activate the event for adding a new case. Figure 2.27 shows the new variables, and the associated invariants and initialisation for this refinement.

MACHINE t1dm_m2
REFINES t1dm_m1
VARIABLES

`caseBase` Sequence of case IDs
`caseCarbs` Function mapping a case ID to the carbohydrate intake
`casePreBG` Function mapping a case ID to the preprandial blood glucose reading
`caseUsedBolus` Function mapping a case ID to the bolus suggestion
`caseTime` Function mapping a case ID to time
`time` Time of the calculation
`activeInsulinCases` Set of applicable cases active insulin has been calculated for
`solutionObtained` Boolean defining if a bolus solution has been obtained

INVARIANTS

`inv1` : $caseBase \subseteq \mathbb{N}_1$ Case IDs are a subset of natural numbers
`inv2` : $finite(caseBase)$ Case-base (case IDs) has cardinality
`inv3` : $caseCarbs \in caseBase \rightarrow \mathbb{N}$
Total function mapping every case ID to carbohydrate intake
`inv4` : $\forall c \cdot c \in ran(caseCarbs) \Rightarrow c \leq 2400$
All retain carbohydrate values must be $\leq$ 240 grams
`inv5` : $casePreBG \in caseBase \rightarrow \mathbb{N}$
Total function mapping every case ID to preprandial blood glucose reading
`inv6` : $\forall bg \cdot bg \in ran(casePreBG) \Rightarrow bg \leq 554$
All retained preprandial blood glucose readings must be $\leq$ 55.4 mmol/L
`inv7` : $caseUsedBolus \in caseBase \rightarrow \mathbb{N}$
Total function mapping every case ID to a bolus suggestion
`inv8` : $caseTime \in caseBase \rightarrow \mathbb{N}$ Total function mapping every case ID to case time
`inv9` : $time \in \mathbb{N}$ Time is ≥ 0
`inv10` : $activeInsulinCases \subseteq caseBase$
Applicable active insulin cases are a member of caseBase
`inv11` : $solutionObtained \in BOOL$

EVENTS
Initialisation
extended
Initilialises the machine
begin
`act20` : $caseBase := \varnothing$ No cases initially
`act21` : $caseCarbs := \varnothing$ No cases initially
`act22` : $casePreBG := \varnothing$ No cases initially
`act23` : $caseUsedBolus := \varnothing$ No cases initially
`act24` : $caseTime := \varnothing$ No cases initially
`act25` : $time := 0$ Initial time value is 0 minutes
`act26` : $activeInsulinCases := \varnothing$ No active insulin cases initially
`act27` : $solutionObtained := FALSE$ Solution has not been obtained
end

Figure 2.27: Variables and invariant for the second machine refinement

Two events allowing cases to be added and removed from the case-base are now defined. Figure 2.28 displays the event for adding cases *retainCase*, the case removal event can be found in the full machine in Appendix C. The information for each feature is stored in the case must be compliant with those defined by the specification, e.g. that carbohydrates must be in the range [0,240] grams. An exception to this is the constraint limiting the maximum bolus dose $maxBolus$, as the maximum allowed bolus dose may changed after retaining a case.

MACHINE t1dm_m2
REFINES t1dm_m1
Event *retainCase* $\widehat{=}$
Retains a case in the case-base
any
c New case ID
where
grd1 : $solutionObtained = TRUE$
A solution must be obtained first
grd2 : $c \in \mathbb{N}_1$
The caseID must be ≥ 0
grd3 : $c \notin caseBase$
The caseID must not already exist in the case-base
grd4 : $caseBase = \varnothing \Rightarrow c = 1$
If the case-base is empty, then the caseID is 0
grd5 : $caseBase \neq \varnothing \Rightarrow c = card(caseBase) + 1$
If the case-base is not empty then the caseID is the cardinality of the case-base + 1
then
act1 : $caseBase := caseBase \cup \{c\}$
Adds the caseID to the case-base
act2 : $caseCarbs := caseCarbs \cup \{c \mapsto carbs\}$
Maps the caseID to carbohydrates
act3 : $casePreBG := casePreBG \cup \{c \mapsto preBG\}$
Maps the caseID to the preprandial blood glucose reading
act4 : $caseUsedBolus := caseUsedBolus \cup \{c \mapsto bolusSuggestion\}$
Maps the caseID to the bolus suggestion
act5 : $caseTime := caseTime \cup \{c \mapsto time\}$
Maps the caseID to the time of the case
end

Figure 2.28: Event to retain cases

The *calcActiveInsulin* event is also refined in this machine to model to increase the machine variable *activeInsulin* for instances where there exists a case c in *caseBase* that is not in the set *activeInsulinCases*, and c has insulin which remains active (Fig. 2.29). Active insulin - also known as insulin on board - is determined by Eq. 2.20 [Campbell and Abramovich, 2012], let ci be the planned carbohydrate intake in grams, t be the time of the new problem in minutes, ct be the case time in minutes, and a be the constant active insulin duration in minutes. As the event will only calculate the active insulin against one previous case $grd6$ is introduced to keep the event active until there are no retained cases which will contribute towards the active insulin sum.

$$\text{iob} = \begin{cases} ci \times \left(1 - \dfrac{t - ct}{a}\right), & \text{if } a > t - ct > 0 \\ 0.0, & \text{otherwise.} \end{cases} \tag{2.20}$$

MACHINE t1dm_m2
REFINES t1dm_m1
Event *calcActiveInsulin* $\widehat{=}$
extends *calcActiveInsulin*

any

a Active insulin remaining from a previous case

c The caseID which active insulin is to be calculated for

where

`grd1` : $a \in \mathbb{N}$
Active insulin must be ≥ 0

`grd2` : $c \in caseBase$
The caseID must exist in the case-base

`grd3` : $c \notin activeInsulinCases$
The caseID must not already be accounted for in terms of active insulin

`grd4` : $time - caseTime(c) > 0$
The current time must be after the previous case time

`grd5` : $time - caseTime(c) \leq active$
The different between the current time and case time must be $\leq$ to the active insulin duration time (active)

`grd6` : $a = activeInsulin + (caseUsedBolus(c) * (1 - ((time - caseTime(c))/active)))$
Active insulin calculation

then

`act1` : $activeInsulin := a$

`act2` : $activeInsulinCases := activeInsulinCases \cup \{c\}$
Adds the caseID into the cases where active insulin has already been accounted for

end

Figure 2.29: Refined event to increment active insulin

The final step in the refinement is to alter the guards on the bolus calculator events (Fig. 2.30) to ensure it is only enabled when there exists no cases in the case-base which would enable the *calcActiveInsulin* event. This is enforced by *grd*6 (Fig. 2.30), which states that each case which would contribute to the active insulin sum must be a member of the set $activeInsulinCases$.

This concludes the second refinement and the model of the bolus calculator. The formal specification presented in Appendices A - C successfully discharged all proof obligations, indicating that the event guards ensure that the invariants will not be violated. The machines are animated using the ProB plug-in to ensure that the events are enabled and disabled as expected. In addition, the animation provides a way to ensure the calculations produce the expected results. In both cases the animation was successful, with events enabled or disabled correctly and the calculations producing the expected results based on the parameters and machine variables.

MACHINE t1dm_m2
REFINES t1dm_m1
Event *bolusCalc* $\widehat{=}$
Bolus calculator event for instances where the result is ≥ 0 and $\leq$ maximum bolus dose
extends *bolusCalc*
any
`s` Bolus suggestion
where
`grd1` : $\texttt{s} \in \mathbb{N}_1$ The bolus suggestion is > 0
`grd2` : $\texttt{s} \leq \texttt{maxBolus}$ The bolus solution suggestion is $\leq$ maxBolus
`grd4` :
$\texttt{s} = (\texttt{preBG} - (\texttt{targetRangeLower} + (\texttt{targetRangeUpper} - \texttt{targetRangeLower})/2)) * (\texttt{isfI}/\texttt{isfBG}) + \texttt{carbs} * (\texttt{carbRatioI}/\texttt{carbRatioC}) - \texttt{activeInsulin}$
The bolus suggestion equals the bolus calculation result
`grd5` : $\texttt{inputsDefined} = \texttt{TRUE}$
The user inputs for the calculation must be done first
`grd6` : $caseBase \neq \varnothing \Rightarrow (\forall c \cdot c \in caseBase \land time - caseTime(c) > 0 \land c \leq active \Rightarrow c \in activeInsulinCases)$
All cases where active insulin is applicable must be a member of activeInsulinCases
then
`act1` : $\texttt{bolusSuggestion} := \texttt{s}$ Sets the bolus suggestion value
`act2` : $solutionObtained := TRUE$ A solution has been obtained
end

Figure 2.30: Refined bolus calculation event

2.4 Summary

In this chapter, state-of-the-art tools for bolus insulin advice were analysed to determine their functionality. This process identified the differences between the tools, and also highlighted the basic implementation of some mobile apps available for bolus advice on the market. The Accu-Chek® Aviva Expert blood glucose meter provided detailed information on the constraints of system variables which were used to produce a formal specification.

The creation of the formal specification not only aids in the implementation of the application, but also helps to understand the problem. By modelling the problem using mathematics, issues with the model can be identified prior to implementation. Primarily this focuses around ensuring that the preconditions (guards) for any actions performed by the machine do not violate the constraints (invariants). The formal specification outlined in this chapter will be used to aid the mobile implementation discussed in Chapter 6 to ensure that the application is robust.

The chapter also highlighted the steps taken by the formal methods community to increase both awareness and encourage developers to use such methods. The Rodin Platform and the repository of resources available significantly aid the process of defining a formal specification. With continued research in this area, there is hope that formal methods will be widely adopted in industry for all system development and not just those of a safety critical nature.

Next we look at case-based reasoning (CBR), the artificial intelligence method which will replace

the formulas used by the bolus calculators discussed in this chapter. The chapter will discuss the history, models, components, and seminal work in the field to provide a general understanding of CBR. Additionally, we will see how CBR has already been used successfully in the T1DM domain to assist professionals.

Chapter 3

Case-based reasoning

This chapter introduces case-based reasoning (CBR), an AI technique which will be used to predict bolus advice instead of the formulas presented in Chapter 2. The chapter begins with an overview of CBR and its origins. This is followed by a discussion on different CBR models which have been proposed, with a detailed look at the processes within these models. Examples of some seminal CBR systems are then presented to demonstrate how CBR can be applied to different tasks. This is followed by a comparison of CBR to other reasoning methods. Finally, the chapter concludes with a look into the use of CBR in the domain of T1DM.

The management of a T1DM subject is based on medical records and historical events containing information such as meals consumed, blood glucose readings, and hypoglycaemic or hyperglycaemic occurrences. This use of historical data for providing guidance suggests that an application which uses historical events is a logical approach to implementing a self-management system.

Case-based reasoning is a form of artificial intelligence (AI) which allows historical events to be used for the purpose of prediction. The CBR process begins with a description of a new problem which requires solving, with a set of *features* describing individual aspects of the new problem. Case-based reasoning uses a similarity algorithm on a new problem to retrieve and reuse solved historical problems known as *cases*. Relevant cases are reused and if necessary adapted to predict a solution to the new problem. Finally, before retaining the new problem and its solution in the case-base, the solution is revised through evaluating the real-world or simulated use of the solution.

This functionality will allow a subject to aid their self-management through the use of similar historical cases based on the philosophy of CBR - that solutions which are known to have worked for historical events can be reused and adapted to solve new problems [Kolodner, 1993].

As these cases are stored within the CBR system's case-base, this information will also be available for doctors to interpret. This is a useful benefit of CBR in the T1DM domain as subjects often have regular medical visits with their doctor. In order for these medical visits to be most effective, accurate logs must be maintained by the subject. The more detailed the information

provided by the subject, the better this guidance can be.

Case-based reasoning has been utilised previously in the domain of T1DM management with positive results [Montani et al., 2000; Marling et al., 2008; Herrero et al., 2014]. The T-IDDM [Montani et al., 2000] and 4 Diabetes Support System^{TM} [Marling et al., 2008] projects resulted in systems to aid a doctor with general guidance for a subject, whereas this research uses CBR as an aid for self-management on a meal by meal basis. More recently research funded by the Imperial NIHR Biomedical Research Centre has looked at using CBR in conjunction with continuous glucose monitoring for bolus insulin advice to aid self-management [Herrero et al., 2014].

3.1 An overview of case-based reasoning

Case-based reasoning is a form of AI which employs historical problems known as cases to predict an outcome for a new problem. These problems may represent anything, but are usually specific to the domain of the CBR system. The aim of CBR is to solve or classify these new problems using previous cases retained within the case-base. An example of a problem may be deciding what should be served at a dinner party. Another example in the context of T1DM is how much insulin should be administered for a meal.

The outcome of a CBR system will vary depending on its purpose. Possible outcomes include an explanation of a new problem based on previous cases, criticising a proposed solution using previous cases, and suggesting a solution to a new problem through the reuse of previous case solutions [Kolodner, 1993].

The foundation of CBR systems can be traced back to research at Yale University in the late 1970s on dynamic memory [Kolodner, 1983; Schank, 1983; Riesbeck and Schank, 2013]. This research focuses on the principle that we use our past experiences to understand and adapt to new problems. When faced with a new problem, we will recall a similar experience in memory aid to us to overcome the problem. We reason that if a solution to a similar problem in the past worked, then the same solution, possibly with some adaptation, will allow us to overcome the new problem. In the event that we fail to resolve the problem, we will learn from the experience and apply what we have learnt to future problems. This concept resulted in the development of CYRUS (Computerized Yale Retrieval and Update System) which implemented the concepts of indexing, reorganising, generalising and searching a model of human memory [Kolodner, 1983].

CYRUS served as a basis for the CBR systems which emerged in the 1980s. Some of these systems, which are discussed as examples later, include: MEDIATOR for conflict resolution [Simpson Jr, 1985], CHEF for meal dish planning [Hammond, 1986], CASEY [Koton, 1988] for diagnosing heart problems, and JULIA for meal designing [Hinrichs and Kolodner, 1991].

3.2 Case-based reasoning models

Case-based reasoning systems have been in development since the 1980s [Simpson Jr, 1985; Hammond, 1986; Koton, 1988]; however, publications of CBR models did not surface until the mid-1990s [Kolodner, 1993; Aamodt and Plaza, 1994; Allen, 1994; Hunt, 1995]. These models describe the steps taken for implementing CBR, and all share the same fundamental aspects of reusing historical cases to propose a solution, and evaluating the proposed solution to improve future reasoning. In this section four models are presented: Aamodt and Plaza's R^4 model [Aamodt and Plaza, 1994], Kolodner's model [Kolodner, 1993], Hunt's model [Hunt, 1995], and Allen's model [Allen, 1994].

3.2.1 Aamodt and Plaza's R^4 model

The R^4 model is one of the most recognised models for CBR [Aamodt and Plaza, 1994], and is based on previous CBR implementations. The model provides an abstract representation of a CBR cycle, describing a four step cycle consisting of the four Rs: *retrieve*, *reuse*, *revise*, and *retain* (Fig. 3.1).

RETRIEVE The most similar cases to a new problem are retrieved from the case-base.

REUSE The retrieved cases are then reused and adapted if necessary to aid in solving the problem.

REVISE The proposed solution is evaluated and revised if necessary.

RETAIN The case and its solution are then retained in the case-base to serve the purpose of solving future problems.

The R^4 CBR cycle begins with a description of a new problem, which is comprised of features. In the domain of T1DM the problem will consist of features including carbohydrates, time and current blood glucose level. Once the new problem has been defined, similar cases to this new problem are then retrieved from the case-base using a similarity measure. The cases retrieved are then reused to propose a solution to the initial problem. If required, the solution will then be adapted in order to provide a better solution to the problem originally presented. Once adaptation is completed, the solution will evaluated and revised where necessary to correct poor solutions. The evaluated solution, alongside the problem are then retained as a new case in the case-base for future reuse.

The strength of the case-base in this model is largely dependent upon the knowledge available during the revision process. This knowledge can either be provided by the user with some domain knowledge, an expert, or an automated process. If solutions retained within the case-base do not solve the problem and yet are considered to be correct, there is a risk of these solutions being reused for new problems. This could occur through incorrectly retaining a poor solution without

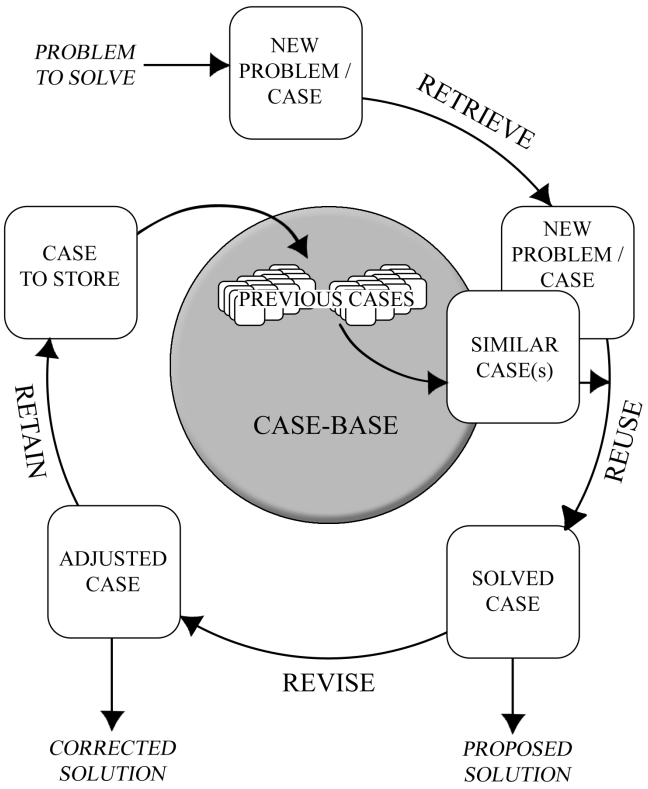


Figure 3.1: R^4 CBR cycle [Aamodt and Plaza, 1994]

correcting through revision, or not including an indication that the solution is poor to aid future reasoning.

For user-based revision, the user does not require specific knowledge of each case within the case-base, but some general knowledge of the domain is required to make informed revisions. This user revision will be a part of our CBR system for T1DM bolus advice, as the user will have some knowledge and guidance on how to manage their diabetes. An example of where user revision would occur would be in the event of a postprandial blood glucose test indicating a low or high blood glucose level. This may be as a result of too much insulin being administered, as a result the solution should be adapted to a lower insulin dose.

An example of automated revision is provided by CHEF, a case-based reasoner for meal dish planning [Hammond, 1986]. CHEF is able to adapt cases through the use of a separate knowledge base of abstract planning problems. These abstract planning problems contain strategies for revising the solution; the solution in the case of CHEF is a recipe to meet goals defined in the original problem. Should revision be required, then the cause of the recipe's failure is retained for future failure prediction. A problem with the automated adaptation approach used by CHEF is the requirement of expert domain knowledge in order to define these strategies. In Chapter 4 an automated evaluation and revision rule is proposed using a postprandial blood glucose reading to automatically revise the bolus suggestion [Becton, Dickinson and Company, 2005]. Additionally failure acknowledgement and revision can be undertaken by the user. Should a failure occur, such

as too much insulin predicted in the solution, then the user can adjust the bolus solution prior to retaining the new case for future reuse.

The R^4 model served as the basis for the similarity-based R^5 model [Finnie and Sun, 2003]. The emphasis of the R^5 model is on case-base building, adding the additional step of *repartition*, where the world of problems and solutions are partitioned using similarity relations.

In summary, the R^4 model provides the fundamental steps required in order to implement a CBR system. The R^4 model and the other CBR models discussed later all have the same limitation of not providing concrete methods for implementing each of the steps involved. This is due to the requirement of domain specific step implementation, as a single method may not be best suited to every domain. As a result, the R^4 model should be seen as a general outline for implementing CBR systems, allowing research and development to focus on the best method for each of the cycle's steps.

3.2.2 Kolodner's model

Kolodner [1993] describes CBR as serving two separate purposes, both using concrete cases rather than abstract rules. The first purpose is for remembering cases to aid in solving new problems (problem-solving CBR). The second is to remember cases in order to understand new problems (interpretive CBR). In complex domains both these purposes are often used together.

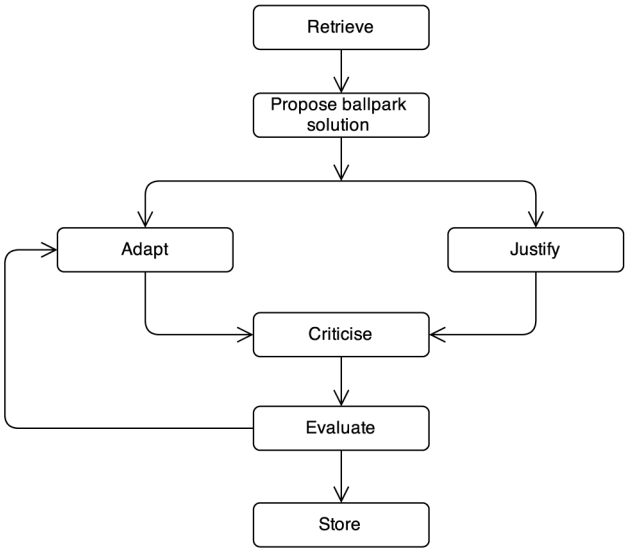


Figure 3.2: Kolodner's CBR model [Kolodner, 1993]

The model begins with case retrieval and the reuse of retrieved cases to propose a solution as described in the R^4 model, but then differentiates depending on the purpose of the CBR system. For problem-solving CBR, the predicted solution is subject to adaptation; as with the R^4 model. For interpretive CBR, the next step of the model is justifying the proposed solution. This justification is done through comparing the new problem with previous cases in order to identify similarities to reinforce the proposed solution, and to identify differences which need to be further assessed.

Once a proposed solution has been adapted in the instance of problem-solving CBR, or justified for interpretive CBR, the solution is now criticised. This involves either simulating the solution or testing the solution against hypothetical situations. The outcome of the simulation or testing is then assessed to determine if further adjustment is required prior to retaining the case.

Prior to retaining the new problem as a case for future reuse, the solution is evaluated in the real world. This step is considered important by Kolodner, as it facilitates providing feedback into why failure occurred and provides reasoning behind the failure. This failure and the reasons behind the failure can then be used to adapt the solution further, and to aid the prediction and solution of failures for future problems.

Kolodner's model concludes with a memory update for retaining the problem, solution and other information useful for future reasoning. This memory update step includes indexing the new case in such a way that it can be retrieved again when it would be helpful in assisting future problems. Indexes are discussed in more detail in Section 3.3.2.

3.2.3 Hunt's model

Fundamentally, Hunt's CBR model [Hunt, 1995] is similar to both the R^4 and Kolodner models; the main difference resides in the analysis of inputs [Avramenko and Kraslawski, 2008]. This analysis stage involves feature selection to determine which attributes are important in the retrieval process. By performing this feature selection, the obsolete attributes can be ignored in the retrieval process. The retrieved case is then adapted to fit the problem prior to evaluation in order to determine whether the solution is suitable. In the event the retrieved case cannot solve the new problem, the reasons behind why it failed are used as a way of repairing the case by forming a new solution. Once the problem is repaired it can then be stored for future use.

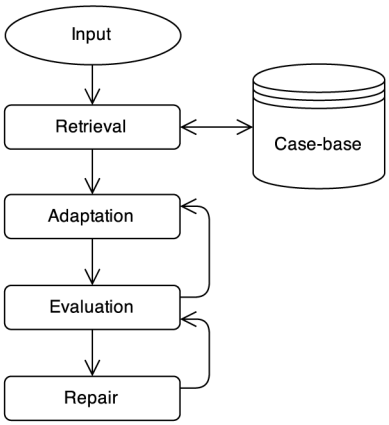


Figure 3.3: Hunt's CBR model [Avramenko and Kraslawski, 2008]

3.2.4 Allen's model

The model proposed by Allen [Allen, 1994] consists of five stages: presentation, retrieval, adaptation, validation, and update [Avramenko and Kraslawski, 2008]. Allen's model is similar to the models previously discussed, but includes an additional process prior to retrieval. This additional step explicitly states that a description of the current problem is created for input into the CBR system. This step is also present in the R^4 model but not as a process in its own right.

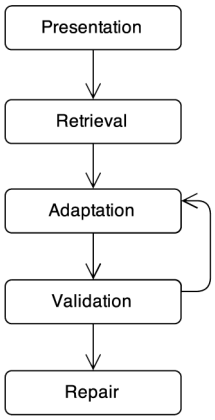


Figure 3.4: Allen's CBR model [Avramenko and Kraslawski, 2008]

3.2.5 Comparison of the models

All four of the models discussed previously are fundamentally similar in terms of the overall process cycle, and revolve around retrieving, adapting and revising cases from the case-base to obtain and improve solutions. The R^4 model is the most established and widely cited, most likely due to the publication's ease of access and easily remembered steps.

Kolodner's model is unique as it proposes the concept of problem-solving and interpretive CBR as two separate tasks. The model is also focused on the importance of learning from failures to anticipate them in the future. Hunt's model introduces the concept of pre-processing through feature selection. It may have been considered that this process is somewhat natural. However, the concept that retrieved cases should be relevant to solving the problem is important to the success of CBR, and it is beneficial to explicitly state this in the model. It is difficult to differentiate between the Allen and R^4 models other than the additional initial step of presentation in Allen's model and naming of the cycle phases.

A common problem with CBR and AI in general is that no single method is suitable for use across all domains [Aamodt and Plaza, 1994]. This leads to the challenge of devising appropriate methods for each step of the CBR cycle to meet the demands of the specific domain.

3.3 Case-based reasoning processes and components

The CBR models provide abstract guidance on how to implement a CBR system. In this section the components and processes used by these models are examined in more detail. Firstly, the cases and the case-base which retains them are discussed. This is followed by a detailed look at the steps presented in the R^4 model: retrieve, reuse, revise and retain.

3.3.1 Cases and the case-base

In CBR, records of historical events are known as cases, and represent specific knowledge relating to a context [Kolodner, 1993]. These cases act as a blueprint describing information known about a historical event called features, and the resulting outcome or solution. In the context of bolus advice for T1DM, these case features will include information such as the quantity of carbohydrates within a meal, the subject's blood glucose level, and the time of the meal. The solution of the case is the bolus insulin administered.

Cases are retained in a case-base for use by the CBR system. The complexity of the case-base depends on the complexity of the domain. In well-defined domains where the features of the cases are known, the structure of a case-base is simple. The complexity of the case-base structure increases when the system is working with multiple domains or the structure of each case varies greatly. Approaches for handling complex case-bases are discussed in the indexing section.

3.3.2 Indexing

In CBR, *indexes* are pointers to general episodes or concrete cases. Indexes facilitate the retrieval stage by identifying relevant cases to a new problem and so affect efficiency. Indexing a case-base was first introduced by Kolodner with the CYRUS system [Kolodner, 1983], based on Schank's dynamic memory model [Schank, 1983]. In CYRUS the case-base is a structure containing three types of object: norms, indexes, and cases (Fig. 3.5) [Kolodner, 1983; Aamodt and Plaza, 1994]. *Norms* are general episodes consisting of all features applicable to all cases in the norm. Indexes are a two component object consisting of the index and values to further discriminate the features within the norm. The index values point to concrete cases or other norms.

The creation of indexes is a system and domain-dependent task and can be created manually or automatically. Manual indexing requires an indexer to identify important features and any lessons learnt in a case. The indexer requires knowledge of the domain in order to produce informed indexes. For example, indexing a hyperglycaemic event with a cause such as stress. The next time a new problem is presented to the CBR system with the feature of stress, the retrieval process would then consider cases indexed by stress as plausible for reuse. Manual indexing can be a time-consuming process which can result in inconsistent indexing, especially where there are multiple

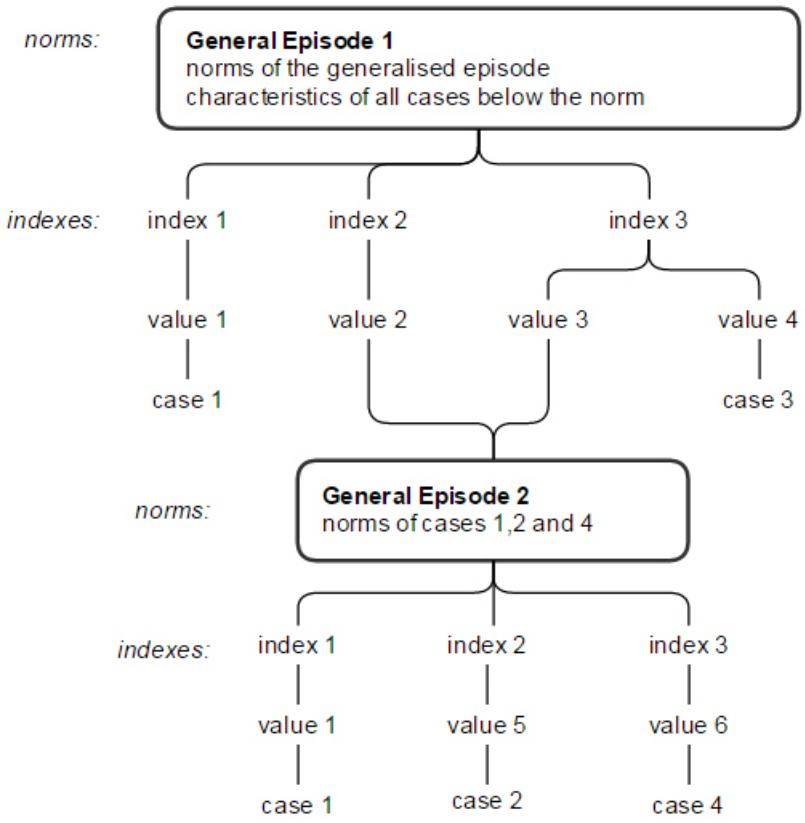


Figure 3.5: Example structure of CYRUS' case-base [Aamodt and Plaza, 1994]

indexers involved.

Indexing through abstraction in CBR aids the retrieval process by allowing problems and cases to be compared away from their concrete instances. This is useful in systems catering for multiple domains, and also for those systems with a large number of features or where the features of a case can vary. Increased levels of abstraction provide further reduction in complexity and constraints [Bergmann and Wilke, 1996].

With abstraction, the case-base can be seen as a hierarchical structure consisting of concrete cases and abstract cases, as with CYRUS. Each concrete case is a child of an abstract case, and depending on the levels of abstraction used, abstract cases are a child of other more abstract cases. Using different levels of abstraction provides benefits for retrieval as the degree of similarity a new problem has to a case can be determined through similarity at different levels of abstraction. The lower the level of abstraction a new problem matches a subset of cases, the higher the similarity and more relevant for this subset of cases is likely to be for reuse.

Kolodner [1993] provides examples of how CHEF [Hammond, 1986] is able to use both concrete and abstract indexing to help recall cases. A case could have an index of *dish includes beef*, however when recalling previous cases other *meat* dishes would not benefit from this index. Instead if the case is indexed more abstractly as *dish includes meat*, the case can then be recalled when the user wishes to create a dish cooked with another meat such as *lamb*. However, abstract indexes do have the disadvantage of important details being missed which could result in failure. For example,

CHEF adapts a vanilla soufflé recipe to create a strawberry soufflé with the result that it fell. The corrected recipe should then be indexed in a more concrete fashion such as *dish type is soufflé and dish includes fruit* so that in future recalls a fruit based soufflé will not result in the same failure.

Automated approaches for indexing include the creation of data structures such as a binary search tree, in addition to the indexing of cases by important features and lessons learnt. The use of a binary search tree or k-dimensional tree (k-d tree) for multidimensional problems removes the need for a traditional indexing approach [Wess et al., 1993]. As CBR is usually a multidimensional problem, the discussion focuses on the use of k-d trees. A k-d tree partitions the case-base recursively using a similarity measure to discriminate each dimension, each dimension being a feature of the case. This discrimination can be achieved through methods such as evaluating the dispersion of numerical or ordered nominal feature values, entropy or the similarity of cases to one another. The structure and success of the k-d tree is largely dependent upon the discrimination method used to partition the k-d tree. The leaves of the tree may be a single case or a bucket of cases, the size of this bucket is predetermined before creating the k-d tree. During the creation of the tree the recursion will end when the bucket size is reached. Once the k-d tree is created, retrieval can quickly traverse the tree until a leaf or bucket is reached.

In this research, the majority of case discrimination is on feature values, as the features of cases will remain constant. As a result, using CYRUS' indexing approach would lead to one general episode, and discrimination would only be achieved by the index-values. This approach of abstracting cases could prove useful if cases do not always contain certain features, such as stress, exercise, hypoglycaemic or hyperglycaemic events. The use of k-d trees for indexing is applicable to this domain as all features are naturally numerical or can be represented on a nominal scale; however, this approach focuses on retrieval efficiency for larger case-bases [Wess et al., 1993]. As this research will use a small personal case-base the need for efficiency is low, and the resources required to create and maintain the k-d tree may outweigh the benefits.

3.3.3 Retrieval

The retrieval process identifies the most similar case or subset of cases retained in the case-base to a new problem [Kolodner, 1993]. As a case-base can be extremely large, selecting the most appropriate retrieval method involves finding the balance between efficiency and the ability to locate the best matching cases within the case-base.

Retrieval begins with the identification of features within the new problem, and is the first stage of understanding the problem and its context [Aamodt and Plaza, 1994]. Unknown features within the problem require understanding or removal. For understanding, it is common for the user to explain the feature in order to link it to relevant category of the case-base, where for example a category can be an index node. The issue of feature identification is outside the scope of the

research undertaken as the features are already known, and only the values of the features will change.

Once the features of the new problem are understood, the case-base is searched to identify the best matching cases to the new problem. This matching process usually comprises two stages; *dimensional matching* and *aggregate matching* [Kolodner, 1993]. Dimensional matching identifies similar cases on a feature by feature basis and is usually facilitated by indexes. Aggregate matching uses a more elaborate similarity measure to match the whole problem to whole cases.

Dimensional matching is used to resolve the serial search efficiency problem by retrieving a subset of plausible cases from the case-base using each feature (dimension) of a new problem independently. One approach to dimensional matching is the traversal of the case-base structure such as a k-d tree and abstract hierarchies to locate a subset of relevant cases. For case-bases where such structures do not exist, a filter can be adopted, where the relevant cases are selected based on their features and corresponding values. An example of dimensional matching as a search filter is selecting only subjects within a certain age range from the case-base dependent upon the value of the feature age in the new problem.

Aggregate matching compares all the features of a new problem to the features in a retained cases in order to determine similarity. This similarity is determined through a appropriate similarity measure, discussed in Section 3.3.4. Aggregate matching is resource intensive and so is often used after dimensional matching to reduce the number of cases where it is required. Features present in the new problem are mapped to the features within cases retained by the case-base. The complexity of mapping these features is dependent upon the domain of the CBR system. In well-defined domains - such as T1DM - where the features of the new problem and cases remain constant the mapping process is simple. The features representing the new problem will correspond to features within a retained case. However, in systems where the features present in a new problem or case may vary, this mapping process becomes more difficult. This is most prominent with cross-domain CBR systems, or if the domain has a large scope. In the cross-domain systems, the features contained within each case are likely to differ. In addition, the meaning of features which appear to be similar can vary depending on the context. For these CBR systems, abstraction and heuristic rules are often used to provide relationships between features.

A serial search is the simplest retrieval method, where each case retained in the case-base is subject to an aggregate match. This method is not efficient, but does have the advantage that every case is considered, meaning that there is no risk of relevant cases being omitted. With the decreasing cost and availability of hardware capable of parallel processing since the advent of CBR, there is an argument that the need for indexing is unnecessary due to the increased complexity and maintenance required of the case-base. Advocates of indexed case-bases argue that it can be used as part of the matching process [Kolodner, 1993].

3.3.4 Similarity measures

Similarity measures are required to perform an aggregate match. A similarity measure can be a numerical evaluation function, heuristic evaluation, abstraction, or a combination.

Numeric evaluation involves determining how similar two cases are through the use of a distance function and is also known as nearest neighbour matching. The distance function takes each feature contained within a new problem and compares it to the corresponding feature within a stored case. It is likely that all features are not of equal importance; to overcome this, each dimension is weighted within the distance function to represent its importance [Cunningham and Delany, 2007]. Common distance metrics used for this purpose include variations on the Euclidean, Manhattan and Minkowski distance functions. Most commonly, some form of the Euclidean distance metric is used with continuous features [El Emam et al., 2001], with an investigation demonstrating the benefits of the weighted Euclidean distance [Mendes et al., 2003].

The Euclidean distance calculates the straight line distance between two points in feature space. In the context of CBR this relates to the distance between a feature of a problem and the corresponding feature in the retained case. Both the Euclidean distance and Manhattan distance are reliant on both the new problem and retained case having the same structure (identical features). Equation 3.1 defines the Euclidean distance calculation, let p be the new problem, c be a retained case, n be the total number of features in the new problem or retained case, and i be an index relating to the feature of the new problem or retained case (e.g. p_i and c_i will relate to the feature of time in both instances).

$$ed(p,c) = \sqrt{\sum_{i=1}^{n} (p_i - c_i)^2} \tag{3.1}$$

The weighted Euclidean distance is more suitable for CBR since it is likely that not all features will have the same impact on the similarity. Equation 3.2 defines the weighted Euclidean distance calculation, with one additional factor of feature weight w.

$$wed(p,c) = \sqrt{\sum_{i=1}^{n} w_i (p_i - c_i)^2} \tag{3.2}$$

The Manhattan distance in comparison determines distance through the sum of the absolute differences of their Cartesian coordinates. It is for this reason it is sometimes referred to as taxicab geometry, reflecting the route a taxicab would take around a grid road network (e.g. Manhattan, New York). The standard Manhattan distance calculation is defined in Eq. 3.3 and weighted Manhattan distance in Eq. 3.4.

$$md(p,c) = \sum_{i=1}^{n} |p_i - c_i| \tag{3.3}$$

$$wmd(p, c) = \sum_{i=1}^{n} w_i |p_i - c_i| \tag{3.4}$$

An example of the weighted Euclidean and Manhattan distance is presented to help illustrate the process. In this example a new problem p is presented in Table 3.1 and two cases c are retained in the case-base with the IDs *1* and *2*, as displayed in Table 3.2. Each case has two features A and B both of which has been normalised to a scale of 0.0 to 1.0. The weightings for each feature are displayed in Table 3.3.

Feature A p_1	Feature B p_2
0.6	0.6

Table 3.1: Example new problem p for the weighted Euclidean and Manhattan distance examples

Case ID	Feature A c_1	Feature B c_2
Case 1	0.8	0.5
Case 2	0.2	0.8

Table 3.2: Example cases c for the weighted Euclidean and Manhattan distance examples

Feature A w_1	Feature B w_2
0.4	0.6

Table 3.3: Feature weights w for the weighted Euclidean and Manhattan distance examples

Example 1 (Weighted Euclidean distance example)**.**

$$wed(p, c) = \sqrt{\sum_{i=1}^{n} w_i (p_i - c_i)^2}$$

$$\begin{aligned} wed(p, \text{Case 1}) &= \sqrt{0.4(0.6 - 0.8)^2 + 0.6(0.6 - 0.5)^2} \\ &= \sqrt{0.4 \times 0.04 + 0.6 \times 0.01} \\ &= \sqrt{0.0160 + 0.006} \\ &= \sqrt{0.022} \\ &= 0.148 \end{aligned}$$

$$\begin{aligned} wed(p, \text{Case 2}) &= \sqrt{0.4(0.6 - 0.2)^2 + 0.6(0.6 - 0.8)^2} \\ &= \sqrt{0.4 \times 0.16 + 0.6 \times 0.04} \\ &= \sqrt{0.064 + 0.024} \\ &= \sqrt{0.088} \\ &= 0.297 \end{aligned}$$

Example 2 (Weighted Manhattan distance example)**.**

$$wmd(p, c) = \sum_{i=1}^{n} w_i |p_i - c_i|$$

$$\begin{aligned} wmd(p, \text{Case 1}) &= 0.4|0.6 - 0.8| + 0.6|0.6 - 0.5| \\ &= 0.4 \times 0.2 + 0.6 \times 0.1 \\ &= 0.08 + 0.06 \\ &= 0.14 \end{aligned}$$

$$\begin{aligned} wmd(p, \text{Case 2}) &= 0.4|0.6 - 0.2| + 0.6|0.6 - 0.8| \\ &= 0.4 \times 0.4 + 0.6 \times 0.2 \\ &= 0.16 + 0.12 \\ &= 0.28 \end{aligned}$$

In both the weighted Euclidean and Manhattan examples it is determined that *Case 1* has the shortest distance to the new problem p and as a result can be considered the most similar case. As the example demonstrates, these two distance functions do not return the same result, meaning that in some situations the case retrieved can vary between the distance functions.

A distance function is not always suitable for determining similarity. Heuristic evaluation may be used in situations where the similarity cannot be gauged without knowledge of the domain. This evaluation process results in the exclusion of cases where it can be predicted that a case will not solve the problem, and when used in conjunction with numeric evaluation, this exclusion will occur first. If the new problem requires context, for example in cross-domain systems, then heuristic evaluation is also needed. Heuristic evaluation allows for specific features to be considered more important depending on the context. Where user interaction is required, the user will define which features are most important in order to filter the matches, or to adjust the ranking by the given preferences.

Abstraction provides a useful mechanism for retrieval when feature values require mapping, similar to the use of heuristic rules. In abstraction, the rules are not explicit; instead they are inferred through relationships and levels of abstraction. For example, in CHEF a new problem is presented with the feature *ingredient* and feature-value *beef sirloin steak*. At a high level of abstraction, beef sirloin steak is a *meat*. This would infer similarity between beef sirloin steak and all types of meat; excluding other factors such as texture. At a lower level of abstraction, beef sirloin steak is a *red meat*, and as a result has greater similarity to other red meats. The

abstraction level can again be lower, with the type of meat as *beef*, where other concrete instances include *sirloin steak*, *T-bone steak* and *fillet steak*. As a result, feature values at a lower level of abstraction infer greater similarity, with concrete instances being the most similar.

In situations where a feature is missing, the system needs to determine whether the absence of a feature can be ignored through the use of evidence [Koton, 1988]. This evidence principle may be used when multiple states relate to the same solution. For example, in a diagnostic application the system can ignore a feature which supports a certain diagnosis should there be another feature present which supports the same diagnosis. If the differences between the features in the new problem and old case can be reconciled, it may be suitable for reuse.

The problems stated above become even more prolific in cross-domain systems. The use of structure mapping is used to overcome this problem. In structure mapping, cases from two different domains can be compared on a more abstract level based on the structure of the cases. The theory is that if two cases share the same structure, then despite representing two different domains, the functional role will be similar.

Selecting an appropriate similarity measure is dependent upon the structure of cases and the case-base. In this research all features to be included are numerical so a nearest neighbour match is a logical approach to use. Other approaches such as heuristic rules and abstraction may be suited should subject's share a case-base in order to identify cases recorded by other similar subjects.

3.3.5 Reuse

A predicted solution is proposed through the reuse of the most similar case or cases from the retrieval stage. Reuse in the R^4 model is the stage of adaptation which determines what can be reused from the retrieved cases, and what needs adjusting to fit the new problem [Aamodt and Plaza, 1994]. As CBR is designed to find partial matches, it is common for an old case to not exactly match the new problem. In order to overcome this issue, the adaptation of cases is required. There are two primary strategies for adapting cases – *substitution adaptation* and *transformation adaptation* [Kolodner, 1993].

Substitution adaptation can be achieved in a variety of ways. The method called *parameter adjustment* is the most relevant to this research. Differences between the new problem and existing case are used to apply specialised rules to the solution. JUDGE [Bain, 1986] a case-based reasoner for sentencing crimes will be used to give an example of substitution through parameter adjustment. JUDGE is presented with a new problem to sentence, in which the victim is killed but the cause is accidental. The case retrieved as the best fit to the new problem also features that the victim is killed; however in this existing case, the death was caused by intent. As the cause of death differentiates, JUDGE adjusts the sentence to reflect this; in this case reducing the sentence by a percentage value.

In the context of this research, parameter adjustment can be used to adjust the insulin according to differences in features between the problem and existing case. An example of this is to adjust the bolus advice by reconciling differences between the active insulin in the new problem and the retrieved case. Active insulin is the insulin which remains active in the subject from a prior bolus dose, and the stacking effect of active insulin can lead to hypoglycaemia [Toffanin et al., 2013; Walsh and Roberts, 2015]. If the active insulin in the problem is greater than that of the retrieved case, the bolus insulin advice will need to be decreased to reflect the difference. In contrast, if the active insulin in the problem is lower than the retrieved case, the insulin dose will require an increase.

Re-instantiation is another method of substitution adaptation which substitutes the values of the new problem into the case selected for reuse. The values of the features can be adjusted where differences between the old case and the new problem can be used to justify changing a value. CHEF [Hammond, 1986] provides an example of re-instantiation, where if at an abstract level the ingredients within the reused case (a recipe) contains the same type of ingredients as the problem (goals of the meal), then the ingredient in the problem can be substituted into the reused case. In a concrete example, if an old case contained the ingredients *chicken* and *peas* and the new problem requires *pork* and *carrots*, CHEF would re-instantiated the old case (*chicken* and *peas*) as *pork* and *carrots* in order to fit the new problem. This substitution is based on the reasoning that these ingredients are interchangeable because both *pork* and *chicken* are *meat*, and both *carrots* and *peas* are *vegetables*. Re-instantiation requires some form of abstraction, such as an ontology to connect similar concrete instances. In this research, the features do not suit the use of re-instantiation; as for example, a blood glucose reading will always be a blood glucose reading and the carbohydrates will always be carbohydrates.

Transformation adaptation is used when substitution adaptation is not a viable option. Transformation is achieved either by a common sense approach, or through model guided repair. An example from JULIA [Hinrichs and Kolodner, 1991] is chosen to explain transformation and how it differs from substitution [Kolodner, 1993]. In JULIA, a meal is designed based on some constraints. JULIA is presented with a new problem with the constraints of *pasta dish* and *kosher meal*, which being kosher cannot contain a combination of meat and dairy ingredients. The most similar retained case JULIA can retrieve is *lasagne*, which contains both meat and dairy in the form of cheese. In order to satisfy the constraints of the problem, JULIA must transform the case to adapt to the new problem. JULIA selects two possible transformation heuristics to adapt the problem from the system. The first transformation heuristic is *remove meat* since meat is considered a secondary ingredient. The second is to substitute the meat based on a role it plays, which in this example is protein. Using this transformation the meat is replaced by the non-meat protein ingredient $Quorn^{TM}$. Common sense has to be applied to the transformation, as should the protein component be removed, the dish will no longer be lasagne. In this example, common

sense says the transformation rule: *substitute meat for Quorn*^{TM} is the best adaptation. JULIA's common sense for choosing the best transformation heuristic is based on user intervention; JULIA will ask a question and the user decides whether or not to proceed. With this transformation, the retrieved case fulfils the constraints of the new problem.

Model-guided transformation is an automated form of transformation adaptation achieved by evaluating the similarities and differences between a new problem and an old case to find a suitable repair heuristic. This repair heuristic is chosen based on general rules inferred by the similarities and differences in an attempt to reconcile the differences. For example, if similarities between the features of the new problem and existing case support the predicted solution then any features known to be redundant to this solution will be removed from the new problem.

The best adaptation strategy to adopt is circumstantial. In the case of this research, a viable adaptation strategy would be substitution through parameter adjustment, as presented in the example of active insulin adjustment mentioned earlier.

3.3.6 Revise

Revision seeks to improve the future knowledge of the CBR system using real-world experience. Aamodt and Plaza [1994] break the revision stage into two tasks of evaluate and repair. If the evaluated solution is considered successful, then the problem and solution are retained as a new case. If the solution is not successful, then the reason for the failure is used to repair the solution. The evaluation of a solution usually occurs outside the CBR system, most commonly in the real-world. In medical domains the results of evaluation may take several months or years. Prior to evaluation, the new case can be stored in the case-base, but it should be noted that the solution has not yet been evaluated [Aamodt and Plaza, 1994]. An alternative to real-world evaluation is the use of a simulator capable of generating a correct solution.

Faults identified during evaluation require repairing prior to retaining the new case. For example, in CHEF, problems not foreseen by the system cannot be repaired until after a dish is created. Looking again at the soufflé example from Section 3.3.2 Indexing [Kolodner, 1993][Hammond, 1986], if CHEF had not previously made a *fruit soufflé*, the system would not be able to anticipate that adapting the *vanilla soufflé* dish would result in the soufflé falling. As a result, this fault can only be detected after the dish has failed. It in these situations where the fault must be repaired through revising a solution which has already been accepted. In the soufflé example, the recipe will be revised to account for the extra liquid resulting from the use of *fruit* in order to prevent the soufflé from falling. CHEF not only uses these revisions to fix a recipe, but also retains these errors to anticipate and avoid faults in future tasks.

In this research, a postprandial blood glucose reading is used to evaluate the cases in order to determine if the solution requires repairing. The postprandial blood glucose reading informs the

subject of their blood glucose levels after the meal and bolus insulin dose. It is this stage which lets us known if the solution was successful or unsuccessful. If successful, the new case can be retained. If unsuccessful, the solution is revised to increase or decrease the bolus suggestion prior to retaining the new case.

3.3.7 Retain

Retaining cases is fundamental to the acquisition of knowledge for aiding future problems [Aamodt and Plaza, 1994]. The effort of identifying similar cases, reusing, evaluating and repairing the new problem would be wasted if this information is not retained.

It is important to retain cases whether the outcome was desired or not. One aspect of retaining is to store information of why failure occurred to facilitate future problem solving. The reasons for failure can then be used to modify solutions in the future. How these failures are stored is dependent upon the system, but they can be retained either as part of the new problem or be retained as separate failure cases. Failure cases can then be used to predict problems in advance, and adjust solutions to prevent this failure re-occurring.

If indexing is used, then it is important that a new case is correctly indexed during the retain step. It may also be necessary to adjust existing indexes to identify relevant cases during subsequent retrievals. For an example of appropriate abstract and concrete indexing please refer to Section 3.3.2 Indexing.

3.4 Examples

This section discusses seminal case-based reasoners developed for different purposes and domains to illustrate the models and concepts introduced so far. These purposes are conflict resolution (MEDIATOR) [Simpson Jr, 1985], diagnosis (CASEY) [Koton, 1988], planning (CHEF) [Hammond, 1986], and design (JULIA) [Hinrichs and Kolodner, 1991]. All the case studies share the fundamental characteristics described in the CBR models, but vary in emphasis. These fundamental characteristics are the use of historical knowledge stored within a case-base, a retrieval mechanism, the ability to reuse and adapt cases to fit a new problem, and a method to evaluate and revise the solution.

3.4.1 MEDIATOR: A case-based conflict resolver

MEDIATOR [Simpson Jr, 1985; Kolodner and Simpson, 1989] is a case-based reasoner developed by Robert Simpson during his PhD at Georgia Institute of Technology. MEDIATOR was designed to resolve disputes between two parties through inference, with the aim to determine resolutions in which both parties may agree. MEDIATOR uses case-based inference to understand the problem,

generate a plan, and recover in the event of failure. In the event MEDIATOR is unable infer solutions using existing cases, it resorts to rule-based methods.

MEDIATOR initially identifies characteristics of the dispute in order to classify it against known dispute types. MEDIATOR will ignore cases which fail to match any specified key features within the new dispute. Having established classifications for the dispute, similar cases are retrieved from the case-base. MEDIATOR then ranks the cases retrieved depending on how similar they are to the new dispute. With the most similar case identified, the program begins to infer a plan to resolve the dispute from retrieved cases. This plan is then subject to checks to ensure that it meets the criteria and goals of the disputants.

MEDIATOR relies on feedback from the solution plan in order to improve decision making. Should the solution be rejected, the system will attempt to explain the failure from a similar previous case which also failed. MEDIATOR will then attempt to correct the cause of failure through a remedy provided by the similar failure case, or through resolution plans. With the failure resolved, MEDIATOR draws up a new plan to solve the problem.

The major contribution of MEDIATOR is its ability to infer solutions to a variety of problem solving tasks through the reuse of experience, as opposed to relying on rule-based methods. The processes used by MEDIATOR were later adopted and improved by other case-based reasoners such as CASEY, CHEF, and JULIA.

3.4.2 CASEY: A case-based diagnostician

CASEY [Koton, 1988, 1989] is a case-based reasoner designed for the task of diagnosis. The goal of CASEY is to diagnose a new subject using the features which represent the subject. The features representing these subjects consist of details such as subject signs, medical history and symptoms.

Cases retained in CASEY's case-base contain the features of the subject and the successful diagnosis achieved through evaluating these features. The original case-base used by CASEY was created using diagnosis produced by the separate Heart Failure Program [Long and Naimi, 1987], a model-based reasoner. This Heart Failure Program is also used as a back-up for when CASEY itself is unable to successfully predict a diagnosis. The diagnoses produced by the Heart Failure Program are then added to CASEY's case-base for future use. The implementation of CASEY focuses primarily on the retrieval, justification and adaptation.

CASEY begins its diagnosis by first retrieving the most similar case to the new subject. If CASEY is unable to identify a similar case, it is passed directly to the Heart Failure Program for diagnosis. Following retrieval, CASEY uses evidence rules to justify whether a retrieved case is suitable for achieving a successful diagnosis. Evidence rules for CASEY specify the situations where differences can be resolved; providing CASEY with the ability to eliminate cases during retrieval, and also allow features of the new subject and retained cases to be removed, added or

adapted. For example, cases with a diagnosis which is known to be associated with a low heart rate would be eliminated if the new subject has a high heart rate. Where features are not present for the new subject or the retrieved case, evidence is found to suggest the feature may be present or is not relevant and can be ignored. In the event significant differences are found, CASEY will determine that the match is not suitable. Where CASEY finds all differences to be insignificant, the old case can be adapted to account for the differences identified.

Adaptation of cases that are deemed by CASEY to have insignificant differences is done based on the evidence principles used in the justification stage. This process involves adding, removing and substituting features or states in the retrieved case to fit the new problem should there be suitable supporting evidence.

If CASEY is able to reach a diagnosis for the new subject, the features of the subject and the resulting diagnosis are retained within the case-base for future reuse. Otherwise, when CASEY fails to achieve a diagnosis itself, it will then revert to passing the new subject to the Heart Failure Program for diagnosis. The resulting diagnosis is then stored within CASEY's case-base for future reuse.

The accuracy of CASEY's diagnoses are as a result of the Heart Failure Program's model, and the additional evidence rules included by CASEY do not reduce the accuracy [Kolodner, 1993]. The benefit of CASEY over the Heart Failure Program resides in its efficiency, with a two to three magnitude increase in response time. This demonstrates how CBR can be used to increase the efficiency of a model-based approach without negatively impacting the outcome.

3.4.3 CHEF: A case-based planner

CHEF was created to be a case-based reasoner for the purpose of planning in the domain of dish recipes [Hammond, 1986]. CHEF plans a dish through the use of previously known recipes, which consist of the sequence of events and the ingredients used to produce the dish. Failures which occurred when making the dishes are also stored by CHEF in order to predict failures in the future.

CHEF begins its planning process through the input of goals required for the dish. These goals can be the inclusion of certain ingredients, or desired characteristics of the dish. CHEF will always attempt to predict failures which may occur in the specified goals based on failures which occurred in previous cases. Should CHEF predict a failure, a new goal will be added to avoid this failure. An example of this failure prediction would be having historical knowledge that a combination of ingredients in the problems goals would contradict the desired texture of the dish.

The established goals representing the new problem are input into CHEF to begin the retrieval process. The best matching recipe retrieved from the case-base is that which achieves the most of the specified goals. The importance of these goals is also factored into the decision. In CHEF the

type of dish is considered more important than the type of meat in the dish for example.

With a recipe selected, CHEF is able to begin its adaptation phase. This firstly involves substituting ingredients where required. In order for this to occur, CHEF must have some knowledge about the ingredient, such as knowing that chicken is a type of meat. An example of this adaptation occurring would be when CHEF has identified a recipe where the dish type matches that specified in the problem, but the meat used in the retrieved dish is different to that specified in the problem. In this instance, CHEF will change the meat within the retrieved recipe to that which is specified in the problem. CHEF will then further adapt the recipe to include the specific steps required for certain ingredients, and steps for the ingredient when used in that type of dish. An example would be de-boning a chicken prior to use, a step which does not exist in the retrieved recipe as a different meat which did not require de-boning was present.

Having created a new recipe, CHEF then seeks feedback through the simulation of the recipe. This feedback allows CHEF to identify failures in order to repair the recipe. If a failure is identified during simulation, CHEF will use known strategies to try and resolve the failure. These strategies are stored in structures known as Thematic Organisation Packets (TOPs). Each TOP is an abstract planning problem and contains a set of strategies for resolving the problem. An example of a TOP in CHEF is the concurrent cooking of two ingredients. CHEF will identify steps in the recipe which relates to a TOP. If the identified TOP does not contribute towards a goal of the problem, then relevant strategies within the TOP will be applied in an attempt to resolve the failure. For the concurrent TOP example, a strategy to cook the ingredients separately could resolve the problem if doing so would not result in the failure of another goal.

Once the recipe meets the goals provided without failure, the recipe is stored in the case-base. This will include information about any failures encountered, and how they were resolved to prevent them occurring in future reuse.

CHEF's major contribution to CBR is the ability to predict failures of a new problem based upon experience. CHEF's failure prediction is a result of failure resolution using pre-defined domain knowledge stored in abstract planning problems called TOPs. The TOPs contain various strategies to overcome the abstract planning problem, which requires domain expertise to create. The use of TOPs for failure resolution works well in specific domains where expert knowledge is available. Although TOPs are abstract, and could cater a large scope, the strategies relating to the TOPs may require maintenance. This would be likely if the scope of the system increases, and additional TOPs and strategies are required. Without domain knowledge, and defining the strategies, CHEF would not be able to resolve failures; and as a result could not predict failures when planning a dish.

3.4.4 JULIA: A case-based designer

JULIA is a case-based reasoner for designing meals [Hinrichs and Kolodner, 1991]. A case-based designer produces a concrete outcome based on a set of constraints, and does not describe the sequence of steps taken (the plan) to reach this outcome. Designing involves adapting the constraints to produce a concrete outcome, removing contradicting constraints and adding new constraints where too many possibilities of a concrete outcome exist. In contrast, a case-based planner such as CHEF forms a plan of the required steps to be taken in order to reach a desired outcome [Kolodner, 1993]. As a result, JULIA is not be able to provide details on how to cook a dish as its purpose is not to produce a recipe. Instead the purpose is to determine what courses, cuisine and dishes would be suitable for the given constraints.

JULIA like CHEF not only uses previous cases to reach its conclusions, but also uses rules or constraints to aid the design process. This general knowledge of the domain allows for the meal plan, and meal dishes to be adjusted where previous cases may not be available to aid the design process. This general knowledge contains information on concepts such as types of meals, courses and social events.

When JULIA receives a new meal to plan, it will first attempt to define the structure of the meal by finding a similar case which fits the criteria provided. This meal structure is a general outline of the meal in terms of the number of people and types of courses included. Should JULIA fail to find a case which fulfils the criteria, it will resort to general knowledge to find an appropriate meal structure. With a meal structure in place, JULIA then breaks down each course into separate sub goals. From this point, JULIA will use the same cycle of attempting to use previous cases for each sub goal first, and then using general knowledge and rules known if necessary.

A unique feature of JULIA in comparison to CHEF is the way it interacts with the subject. Where CHEF is a fully automated system which works with the demands input by the subject, JULIA instead will ask for feedback of suggestions or ideas during runtime. For example, should JULIA determine that the ingredients specified are common to two types of cuisine; JULIA will ask the subject which of the cuisines they would like.

In addition to asking the subject questions, JULIA is also able to cope with constraints being added during runtime. If the new constraints breach the solutions already obtained for sub goals in the meal, JULIA will attempt to adapt and repair the solutions. This may be achieved by swapping ingredients for example. JULIA will only backtrack on solutions it has reached if the solutions cannot either be repaired, or the constraints cannot be relaxed.

3.5 Comparison to other reasoning approaches

Case-based reasoning provides a number of useful contributions to the creation of automated reasoning systems. Case-based reasoning provides the ability to reuse previous cases which worked in the past, and can advise of potential problems based on previous experiences. In addition, some CBR implementations such as CASEY have been shown to outperform existing expert systems in terms of speed whilst maintaining accuracy [Koton, 1988]. The expert system in the case of CASEY is a model-based Heart Failure Program. However, CBR does some have disadvantages. If no cases in the case-base are able to correctly solve the problem then adjustment rules are necessary to produce a potential solution, which often requires expert knowledge. Additionally optimal solutions may be omitted where decision trees are used to reduce the number of cases considered.

A comparison of CBR with rule-based and model-based reasoning is presented in this section. In the case of model-based reasoning, there is discussion into how it can be used in conjunction with CBR to improve the overall capability of a system.

3.5.1 Rule-based reasoning

Case-based reasoning can be seen as a form of rule-based reasoning [Kolodner, 1993]. The underlying difference between CBR and a rule-based approach is in the way knowledge is applied. In a rule-based system, knowledge is applied through defined rules inside the program, such as if-then-else rules [Golding and Rosenbloom, 1996]. These rules are formed by experts with the appropriate knowledge for the domain. In contrast, with CBR the content of the cases form the knowledge base, emphasising the focus on case features and adaptation. Another difference between the two approaches is that CBR is built around the concept of finding partial matches instead of requiring an exact match to the predefined rules. A rule-based approach explains how a solution was achieved by following the sequence of rules used. Case-based reasoning instead provides its explanation of the solution through the information within the reused cases.

Case-based reasoning systems have been shown to require significantly less development time as well as reduced maintenance. An example of this can be found in the comparison of two similar systems using the different approaches CANASTA [Register and Rewari, 1991] and CASCADE [Simoudis, 1992]. Both systems assist engineers in crash recovery of the Virtual Memory System operating system. CANASTA is a rule-based reasoning system for diagnosis and required 960 person days to implement the system. CASCADE, a CBR system, instead only required 105 person days. Rule-based systems require ongoing maintenance throughout their lifetime, due to the need for new rules to be added. Often this results in the added complication of modifying existing rules. In comparison, CBR systems require little maintenance, as the knowledge is updated through the

addition of new cases.

Systems and frameworks have been developed to combine both rule-based reasoning and CBR. This combination may be adopted to find solutions where the rules are unable to do so, and as a method to increase efficiency [Golding and Rosenbloom, 1996]. CABARET [Skalak and Rissland, 1990] provides an example of how rule-based reasoning and CBR have been combined to aid statutory interpretation. In CABARET, CBR is used to assist reasoning by identifying cases by broadening and discrediting rules to support a particular viewpoint.

The drawback of rule-based reasoning systems is that they are limited by the rules known to the system. Case-based reasoning instead provides a method to solve problems without rules, and can reach a solution without requiring an exact match. Whilst rule-based reasoning systems can be effective, the resources required and the limitations suggest that CBR can be more effective. This is reiterated by some rule-based reasoning systems adopting CBR to assist in situations where the rules alone cannot reach a solution.

3.5.2 Model-based reasoning

Model-based reasoning, as with CBR, was developed to remove the need to start the reasoning from the very beginning [Kolodner, 1993]. Both use knowledge to provide reasoning to their solutions, but the way the knowledge is represented is different. In CBR, knowledge is stored within cases, containing features of a specific episode. In model-based reasoning, knowledge is constructed from a good understanding of the domain and infers causal rules from observations to reach a solution.

Another difference between the two approaches is how the knowledge is used. In CBR, the knowledge is used to construct a solution, but is not able to validate or evaluate them. In contrast, model-based reasoning does not provide a method for producing a solution, but instead the knowledge to validate and evaluate solutions.

Case-based reasoning and model-based reasoning complement each other and are often used together. The ability to apply knowledge about a domain in order to validate and evaluate cases is an important part of the adaptation process. Examples of systems where these two concepts work together include CASEY and JULIA, both of which use general knowledge about the domain in order to perform adaptation. For example, CASEY uses a model-based approach to form evidence rules in order to evaluate and adapt cases. JULIA instead uses a model-based approach to construct different meal types in terms of the number of courses, and advises on the general content within courses. Ultimately, the benefit of using CBR and model-based reasoning together lies in the ability to reason about general situations whilst considering specific aspects of previous episodes.

3.6 Case-based reasoning in the T1DM domain

The rise in popularity of CBR has led to its use in many domains including assisting with T1DM management. The T-IDDM (Telematic Management of Insulin-Dependent Diabetes Mellitus) project contains the first notable work within the domain [Montani et al., 2000]. The project set about classifying T1DM subjects in order to find and suggest therapies based on historic data. Subsequent work was also undertaken at Ohio University as part of the 4 Diabetes Support System™ project [Marling et al., 2007].

Case-based reasoning has primarily been used in the T1DM domain to classify subjects in order to provide doctors with a system to assist the identification of appropriate treatment therapies as opposed to assisting self-management [Montani et al., 2000; Marling et al., 2007]. An exception to this trend is in the research and development of the CBR(R2R) advanced bolus calculator, funded by the Imperial NIHR Biomedical Research Centre [Herrero et al., 2014]. The research in this thesis also uses CBR to predict bolus advice, however the approach adopted differs from that used by Herrero et al. [2014].

3.6.1 T-IDDM Project

Research into T1DM therapy using CBR was undertaken as part of the T-IDDM project [Montani et al., 2000], which set out to implement a web-based telemedicine system, comprising of many IT services for managing chronic subjects. The CBR system aims to classify subjects in order to find appropriate past therapies. Abstraction is used to structure the case-base, where each concrete case is a leaf node of the lowest level of abstraction (basic class). Higher levels of abstraction exist in the hierarchy with the highest level being the root class. The root class contains all concrete cases and abstract classes. The use of this structure allows more flexibility in the retrieval stage and also provides allows cases with missing features to be classified more generally. Determining the basic class of the case is achieved through Bayesian classification, with the actual implementation using a Naive Bayes strategy due to the assumed independence of the features within the case. Naïve Bayes assumes that all features are independent of each other whether they are present or not [Kononenko, 1993]. For this approach, a subset of features considered most important by a doctor are utilised for determining the probability the case belongs to a certain basic class. The cases are represented by various features (e.g. sex, height, and weight) of the subject in general and T1DM relevant factors (e.g. glycated haemoglobin (HbA_{1c}), insulin regime, etc.).

During retrieval, both the most probable class and a set of classes can be considered by using the Heterogeneous Euclidean-Overlap Metric (HEOM) and Heterogeneous Value Difference Metric (HVDM) distance metrics respectively [Wilson and Martinez, 1997]. Both metrics cater for the use of both continuous and nominal values, where the standard Euclidean metric cannot appropriately

handle nominal values. The metrics also allow for missing data by defining the distance of missing features as 1 on the scale [0, 1] where 0 is an exact match, and 1 is the maximum distance.

Results from this research on 147 cases found the classification technique had an 83% success rate, with 98% of cases falling into the two most probable classes. The high success rate provides evidence that CBR can be appropriate in this domain. However, the system is not intended for use as self-management, but as a tool for doctors to use to find potential adjustments to the subject's therapy.

3.6.2 The 4 Diabetes Support System™ Project

The 4 Diabetes Support System™ [Marling et al., 2011] project is led by Marling at Ohio University and focuses on the use of CBR to suggest therapies for blood glucose control issues. Initial research for this project began with a clinical study between 2006 and 2007 with twenty T1DM subjects using insulin pumps [Schwartz et al., 2008]. The study lasted six weeks and required the participants to log information regarding their continuous blood glucose level, activities and general health. Doctors took the collected data to structure cases, each of which represented one problem with blood glucose control, and a recommended solution. A rule-based assessment routine was developed in conjunction with doctors to automatically identify problems in patient data. The doctor selects key problems resulting from this rule-based assessment to be input into the CBR system, with the output of a similar case for each key problem. The output cases are then reused to determine appropriate therapeutic adjustments.

The CBR process used in this project begins by identifying a subset of the most relevant cases based on the each key problem type. Each case within this subset is then subject to an aggregate match to score the similarity. If the score calculated is above a certain threshold, then the doctor is given the suggested therapy adjustment for the subject. The doctor can then decide whether or not to accept the suggestion.

The system was evaluated by a panel of three doctors and one specialist diabetic nurse who were asked to review a sample of 10 problems. These 10 problems were randomly selected from a total of 352 problems which were detected by the system. The panel were asked three questions is relation to the problems, and the results were mostly positive. One such result is that the evaluators agreed that the system identified the problem correctly 77.5% of the time.

Another evaluation of the system was also undertaken, where a similarity threshold to remove cases below a certain score was disabled. In this test, 10 out of a total of 50 cases were randomly selected and presented to a panel of three doctors. The doctors agreed 80% of the time that the cases identified were similar to the problem, despite the threshold being disabled.

Further research between 2009 and 2010 aimed to enlarge the case-base, devise rules for detecting additional problems and to enable adaptation of previous cases. During this research subjects

collected data using a continuous blood glucose monitor and insulin pump, and uploaded general day-to-day information on a daily basis via a Web browser. Later in the project subjects could upload data using a mobile app.

The 4 Diabetes Support System^{TM} project has continued evolve since the CBR tool was created, but the focus of the project has moved away from CBR [Marling et al., 2011]. The project has evolved to include work into the inclusion of other classification techniques for problems which could not be reliably detected. Alongside this, there have been improvements to the interface. Assessments of system performance have also been undertaken. The most recent addition to the project is the inclusion of blood glucose prediction through Support Vector Regression.

3.6.3 CBR(R2R) Advanced bolus insulin calculator

Recent research published in 2014 into the use of CBR for bolus insulin advice has been undertaken in a project funded by the Imperial NIHR Biomedical Research Centre [Herrero et al., 2014]. This research uses CBR in conjunction with Run-to-Run control (R2R) algorithm to provide insulin decision support on a mobile device. The R2R algorithm updates the insulin sensitivity factor and carbohydrate-to-insulin ratio through the evaluation of two post-postprandial glucose samples.

Cases in the CBR(R2R) approach are comprised of a triplet: parameters of the problem, solution, and outcome. The solution retains the carbohydrate-to-insulin ratio in addition to the insulin sensitivity factor. The outcome describes the results of applying the solution to the parameters of the problem.

In the CBR cycle, the most similar case is retrieved from the case-base using a weighted distance function. The solution from the retrieved case is then reused in a bolus calculator formula. During reuse, the hyperglycaemic correction is only applied if the preprandial blood glucose measurement is below or above set hypoglycaemia and hyperglycaemia thresholds. The solution is then revised the outcome to the new problem is unsatisfactory. A new case from the new problem are only created and retained if no similar case already existed.

Evaluation of CBR(R2R) using the UVa/Padova T1DM simulator [Kovatchev et al., 2009] showed improvements over the use of R2R alone, reinforcing the benefits CBR can provide. CBR(R2R) also resulted in improvements to mean blood glucose levels, and the complete elimination of hypoglycaemia, which was not achieved by R2R alone.

3.7 Summary

In this chapter, an overview of CBR has been described. Firstly, models proposed by Aamodt and Plaza, Kolodner, Hunt, and Allen to outline the steps required for CBR were described. This was followed by a more detailed look at the functionality of the steps in the R^4 model proposed by Aamodt and Plaza. This model is broken down into four phases: retrieve, reuse, revise, and

retain. The retrieve phase identifies similar existing cases to a new problem. These retrieved cases are then reused to propose a solution to the new problem. Reuse of the cases is often to subject to adaptation in order to resolve differences between the new problem and the retrieved cases. With a solution proposed, it is then subjected to real-world evaluation or simulation to determine if the proposed solution is correct. The evaluation and revision phase is crucial to CBRs learning process. Only through correcting and improving solutions can the system learn to improve future suggestions. Finally, the evaluated solution is retained in the case-base for future reuse.

Although models for implementing a CBR system exist, they are abstract and each phase requires implementation suitable to the desired task. Examples of the seminal CBR systems MEDIATOR, CASEY, CHEF, and JULIA were used to illustrate some approaches used for devising CBR systems in different domains. The overview of CBR concluded with a comparison of CBR to rule-based reasoning and model-based reasoning. Development and maintenance of rule-based reasoning systems was shown to be resource intensive in comparison to CBR systems. Whilst model-based reasoning has been shown to have good synergy with CBR when the domain is well understood.

The chapter concluded with a look into the use of CBR within the domain of T1DM. The T-IDDM project and 4 Diabetes Support System^{TM} utilised CBR in the T1DM domain to aid doctors with improvements to therapy. Both projects demonstrating positive results, and highlight the benefits of CBR in the T1DM domain. Most closely related to this research is the Imperial NIHR Biomedical Research Centre's CBR(R2R) project, which uses CBR to assist the optimisation of a bolus insulin calculation formula, aiding self-management. Simulation of CBR in conjunction with the R2R algorithm demonstrated positive results with improved blood glucose control, and provides evidence for the benefits of using CBR in the context of T1DM bolus advice.

Using the techniques identified in this chapter, we describe a CBR system for T1DM self-management in Chapter 4. The system follows the R^4 cycle introduced in this chapter, with each stage tailored towards the T1DM domain. The system includes a novel temporal retrieval algorithm, active insulin adjustment and automated advice evaluation. The system is then tested and evaluated in Chapter 5 to determine and evaluate the optimal retrieval, adaptation and evaluation configurations, and how the system compares to existing tools.

Chapter 4

A case-based reasoner for bolus decision support

This chapter applies the domain understanding acquired in Chapter 2 and the CBR methods from Chapter 3 to propose a CBR system for T1DM bolus decision support. The system is primarily based on the R^4 model, following the same cycle of retrieve, reuse, revise, and retain [Aamodt and Plaza, 1994]. The R^4 model provides an abstract procedure to follow, but lacks detail on how each step should be achieved. This chapter expands the work in progress paper presented at IEEE HealthCom 2013 (Appendix G) [Brown et al., 2013].

The chapter begins by describing an overview of the CBR system architecture. This system architecture frames the rest of the chapter, where each component is discussed in detail. Firstly the structure of cases and case-base are defined through use of the UVa/Padova T1DM simulator [Kovatchev et al., 2009], which is also used throughout this research to generate test data and to evaluate the bolus advice suggested by the system. This is coupled with the identification of case features from those present in the state-of-the-art bolus calculators discussed in Chapter 2.

The retrieval stage of the system is then presented, discussing the similarity measures used, and how it has been enhanced for this domain through the use of automated feature weightings obtained through feature selection algorithms, and expanding the scope of a new problem using temporal sequences.

Following retrieval is the reuse stage. A simple reuse strategy is initially introduced to facilitate testing of the retrieval algorithm. This is then expanded to include an adaptation rule for factoring insulin stacking into the reuse stage as a form of parameter adaptation. Insulin stacking occurs as a result of insulin remaining active for a period of time after a insulin dose, meaning that a reduced bolus dose should be suggested to prevent potential hypoglycaemia.

Finally, the new case is revised and retained by the CBR system. During the revise stage the

suggested solution is evaluated to determine if the solution was correct or if it should be revised prior to retaining the new case. Evaluation is achieved through the assessment of a postprandial blood glucose reading. By comparing this reading to the user's target blood glucose level, the bolus advice can be revised to help the system learn and improve future suggestions.

4.1 System architecture

The architecture of the system is presented in Fig. 4.1. The system follows the R^4 model proposed by Aamodt and Plaza [1994] with the explicit introduction of temporal sequences in the retrieval stage.

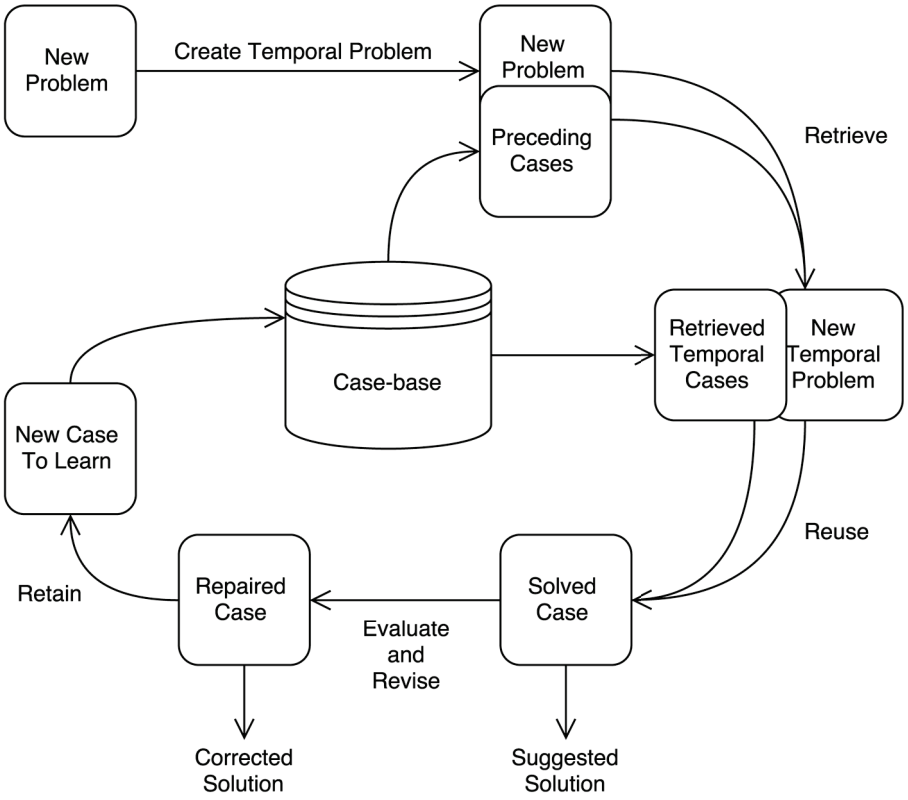


Figure 4.1: Case-based reasoning model for T1DM bolus insulin advice

The cycle of the system centres around the case-base, which stores the individual cases. The system begins when the user inputs information about a new problem, which consists of their blood glucose level, meal information and time. This new problem is then combined with preceding cases to form a temporal sequence. A temporal sequence is a sequence of consecutive cases which expands the new problem to include preceding factors.

This temporal problem sequence is then compared to temporal sequences contained by the case-base in the retrieval stage of the system, as opposed to comparing isolated events. Comparison is achieved in this system using a weighted distance function, with the feature weights computed by a feature selection algorithm. This automated weighting provides additional benefit by tailoring the similarity to the user. For example, one user's bolus advice may be more dependant on

carbohydrates than another user.

The most similar case is then reused in order to suggest a bolus insulin dose to the user. This suggestion is subject to adaptation in this stage, where the bolus dose may be modified based on differences in the active insulin of the retrieved cases and the current active insulin. The user may also choose to manually perform adaptation based on their domain expertise.

After adaptation the case is considered solved. It is now that the solution can be revised based on the actual outcome of using the suggestion. Revision can be achieved manually by the user or an automated approach to revise the suggestion based on a postprandial blood glucose reading and their target blood glucose range can be used.

The case and its corrected solution is then retained in the case-base for future cycles of the system. Through continuous revision of the CBR suggestions, the system can learn to improve bolus advice with each iteration.

4.2 UVa/Padova T1DM simulator

The FDA-approved UVa/Padova T1DM simulator [Kovatchev et al., 2009] is used in this research to create test data and to evaluate the proposed CBR model. The simulator allows blood glucose management to be assessed using *closed-loop* and *open-loop* control algorithms. The closed-loop algorithm acts as an artificial pancreas, where bolus insulin is automatically calculated and dosed to the subject as required [Farmer et al., 2008]. In contrast, the open-loop algorithm is reliant on the subject administering the bolus insulin themselves.

Data can be simulated in two ways. One method is through the user interface, which allows five meals at pre-defined times to be input over the course of multiple days. Alternatively, simulation can be conducted using a scenario file, which is a text file that allows various parameters of the simulation to be set with greater flexibility. Parameters include meal times and carbohydrates, simulation length, bolus times and dose, and closed-loop or open-loop control. The only inputs required for a meal to be simulated using closed-loop control is the quantity of carbohydrates in grams, and the time of the meal. When using open-loop control, the bolus insulin dose and time must also be specified. The results of the simulation can be exported as minute by minute information on signals such as bolus doses, basal rate, and blood glucose level. This allows for a continuous blood glucose reading to be obtained for the purpose of analysis and evaluation.

The simulator serves two important purposes for the research and development of the CBR system. Firstly, the simulator allows for the production of datasets to create test case-bases and problem sets through closed-loop simulation. Secondly, the simulator enables bolus suggestions obtained through CBR to analysed and evaluated using open-loop simulation. The simulator does have some limitations for this research, including the inability to model physical activity, stress, time period based insulin sensitivity factors, and time period based carbohydrate-to-insulin ratios.

4.3 Identifying case features

When working with a single domain and context such as bolus advice for T1DM subjects, the features representing a case remain constant. This allows the case and case-base structure to be defined at the start of the design process. The identification of key features begins with looking at the state-of-the-art bolus calculators discussed in Chapter 2, and those available for testing with the simulator. It is important that all case features considered can be simulated in order to reliably analysis and evaluate the CBR suggestions.

In Chapter 2, the *Accu-Chek® Aviva Expert* (AE) blood glucose meter and four mobile apps - *RapidCalc* (RC), *Diabetes Personal Calculator* (DPC), *Diabetic Dosage* (DD), and *Insulin Calc* (IC) - for bolus advice were analysed. Each app varied in regards to the level of detail required to perform a calculation. The features used in the bolus calculation by the blood glucose meter and mobile apps are shown in Table 4.1.

	AE	RC	DPC	DD	IC
Carbohydrate intake	✓	✓	✓	✓	✓
Preprandial blood glucose	✓	✓	✓	✓	✓
Target blood glucose	✓	✓	✓		✓
Insulin sensitivity factor	✓	✓	✓	✓	✓
Time period variations	✓	✓			
Carbohydrate-to-insulin ratio	✓	✓	✓		✓
Time period variations	✓	✓	✓		
Insulin on board	✓	✓	✓		
Exercise	✓	✓			

Table 4.1: Comparison of features used in the state-of-the-art bolus calculators

All the state-of-the-art bolus calculators are reliant on the input of a problem-specific carbohydrate intake and a preprandial blood glucose reading in order to obtain a bolus suggestion. The target blood glucose level, insulin sensitivity factor, and carbohydrate-to-insulin ratios are all likely to remain static as they are definable settings. In the case of *Diabetic Dosage* the target blood glucose level cannot be modified by the user. These static features will not aid case retrieval as each case will contain the same feature-value. Using a time period based insulin sensitivity factor and carbohydrate-to-insulin ratio would make these features useful to retrieval as the feature-value may change depending on the time of the case. However, as discussed in Section 4.2, time period based insulin sensitivity factors and carbohydrate-to-insulin ratios are not supported by the simulator. Insulin on board will be included using temporal sequences which are discussed in Section 4.7, and as an adaptation rule in Section 4.10.

The features in the simulator are divided into two categories: subject features and meal-specific features. There are a number of constant subject features included in the simulator, with each subject feature representing a different medical statistic. These subject features can be discarded, as the feature-value does not change and would not aid retrieval. Additionally, it is unlikely the

majority of the medical statistics would be known by the subject themselves. The meal features modelled by the simulator are carbohydrates, blood glucose level, and the time of the meal. The blood glucose level is calculated by the simulator at run time, with only the initial blood glucose level required as an input. These meal features present in the simulator are consistent with those identified in the state-of-the-art bolus calculators.

Analysis of the state-of-the-art bolus calculators and simulator determined that the case features will include planned carbohydrate intake, preprandial blood glucose level, time, and the bolus insulin solution. These case features are limited to those which can be modelled by the simulator. This is due to the simulator serving the purpose of creating the test case-base, problem sets, and providing a method analyse and evaluate CBR suggestions using open-loop control. The limitations of the simulator mean that time period variations of insulin sensitivity factor and carbohydrate-to-insulin ratios, physical activity and alcohol cannot be included in this research. Through this proof of concept, the same techniques discussed in this chapter can be applied to these features in future work.

4.4 Case-base structure

Each case is comprised of features describing information known about the case, and a corresponding solution (a bolus insulin dose). In this system, the features attached to this solution represent the corresponding meal information.

All cases in the initial model relate to the subject of the system. Therefore, subject-specific information on factors such as weight, sex, and age are omitted. The inclusion of these features would only be necessary when looking at either long-term use of the application, or if the cases are to be used by other users. With long-term use of the application, it is likely that the subject's characteristics will change. Although it could be considered that only retrieving recent cases is a viable solution, it could potentially lead to relevant cases being omitted. For instance, if the subject was to lose or gain weight, then the amount of insulin they require could change. Should their weight then revert back to a state previously observed, then these older cases may prove to be more relevant than more recent cases.

When looking at the potential for sharing these cases, the relevance of these features becomes even more important. The question may be asked why subjects would wish to have their insulin management based on the behaviour of other subjects. A strong reason to consider this approach is due to new users initially having no previous cases, meaning that no solution can be obtained. Reusing the cases of other users could overcome this problem. Alternatively, the a formula used by a state-of-the-art bolus calculator could be adopted until a sufficient case-base has been produced.

In a shared case environment, the number of cases to sift through during retrieval has the potential to become huge. For dealing with large case-bases, indexing would be required to ensure

efficient retrieval. In this research, all the cases will be within the same context and of the same structure. As a result, indexing would need to be based on the characteristics of a subject.

4.5 Retrieval

The retrieval stage seeks to identify the most relevant case(s) within the case-base for reuse. Case retrieval mechanisms fall into two categories: dimensional matching and aggregate matching. Dimensional matching identifies retained cases with similar features to those of the new problem, and the independent similarity of feature values. Dimensional matching would serve only as a means of improving efficiency by filtering the cases on a feature-value basis in this domain due to the structure of all cases being identical. Aggregate matching is used to provide a detailed comparison between the values of all features in the new problem and those of a retained case. How aggregate matching is performed is dependent upon the domain. It is common in numerical domains to use a nearest neighbour metric; in other domains, heuristic rules, or abstraction may be used to determine similarity.

Nearest neighbour classifiers for numerical domains typically use a distance metric, such as Euclidean and Manhattan (Taxicab geometry) distance [Wilson and Martinez, 1997]. Both the Euclidean and Manhattan distance metrics allow for the computation of the distance between two points in feature space, even with the presence of multiple features. Euclidean distance is a direct line connecting two points within a feature space, whereas the Manhattan distance is the distance by going horizontally or vertically along the axis of the feature space. In the case of nominal values, the Euclidean and Manhattan distance functions are inappropriate. Although nominal features can be represented on a continuous scale, the results would prove meaningless if there is no order to the scale. One solution to this problem is the use of a hybrid distance function, which simply reverts to using a distance function designed for nominal features where present. One such nominal distance function is the Value Difference Metric (VDM), which determines the probability of a feature's value relating to an output class. As all the features present in this model are numeric, a distance metric such as Euclidean or Manhattan is appropriate for this domain [El Emam et al., 2001].

A problem with the standard Euclidean and Manhattan metrics is the possibility of attributes having a greater influence over the result than desired. This may occur when the range of values between features varies by substantial amounts. To address this issue, all features are normalised prior to use within the distance function. By default, all features are considered to have equal importance within the function. This may not be desirable for achieving the best results, and certainly is not desirable within the T1DM domain where the importance of some variables are known to have a greater impact on the quantity of insulin required. To overcome this a weighted version of the distance function is used, which provides each feature with a multiplier denoting importance. Section 4.6 describes the acquisition of these weights using feature selection algorithms.

4.6 Feature weighting

To determine the similarity between a problem and an existing case, it is necessary to consider the importance of each feature. If the similarity comparison is conducted with each feature having equal importance, the result may not be desirable. Instead, each feature should be weighted in accordance to its ability to correctly predict the outcome. These weights can then been used in the similarity comparison to ensure accurate case retrieval.

Determining a weighting for each feature can be achieved by identifying the relevance between a feature-value and the outcome. An understanding of the domain may tell us that certain features are more important than others in determining the outcome. However, an estimate is not satisfactory, and may vary from subject to subject. Instead, weightings can be identified from information known to the system, which in this instance is available through the case-base. In data mining, the process of feature selection is used to determine which features are required to reliably predict the outcome [Guyon and Elisseeff, 2003]. This allows features of little or no importance to be ignored, which is useful in big data environments. There are two main approaches to performing feature selection: *subset evaluation* and *single-attribute evaluation* [Witten and Frank, 2005].

Subset evaluators aim to identify the smallest subset of features which can successfully predict the outcome [Witten and Frank, 2005]. This is usually driven by a random cycle, which depending on the number of features present, may or may not include every combination. In order to prevent excessive computation time, the process is often limited to a certain number of iterations. Each subset selected is tested to see how reliably the solution can be predicted, and the smallest possible subset which achieves this is returned.

Single-attribute evaluators do not attempt to identify the smallest subset of features, and instead evaluate each feature independently [Witten and Frank, 2005]. Each attribute is assigned a numerical result based on its ability to predict the outcome, which allows for the features to be ranked.

Both approaches are suitable for feature selection. However, for the purpose of weighting features only the single-attribute evaluators will be considered. Through performing single-attribute evaluation on each of the features, the results obtained can be used within the distance function for feature weighting. The use of feature selection using single-attribute evaluators is analysed during the experimental phase of this research. Several well-established attribute evaluators within the Weka [Hall et al., 2009] data mining application will be used to obtain these weightings.

The data mining tool Weka provides a number of single-attribute evaluation algorithms. Some single-attribute evaluators present in Weka include: *Chi-Squared*, *Information Gain*, *Gain Ratio*, *Symmetrical Uncertainty*, *One Rule*, and *RELIEF-F* [Witten and Frank, 2005]. Each of these attribute evaluators are described in the proceeding subsections. The performance of the feature selection algorithms for T1DM bolus advice case retrieval is discussed in Chapter 5.

4.6.1 Chi-Squared

In Weka, Chi-Squared is a single-attribute feature selection algorithm based on the χ^2 statistic [Witten and Frank, 2005]. The algorithm seeks to evaluate the χ^2 value for a feature in respect to predicting a class by comparing the number of observations to the excepted frequency. The χ^2 value for any feature is defined by Eq. 4.1, let E_{ij} be the expected frequency, and O_{ij} be the observed frequency for class i with the feature-value j [Hall, 1999].

$$\chi^2 = \sum_i \sum_j \frac{(E_{ij} - O_{ij})^2}{E_{ij}} \tag{4.1}$$

The excepted frequency E_{ij} is defined by Eq. 4.2, let n_j be the number of instances of the feature with value j, n_i be the number of instances of the class i, and n be the total number of instances.

$$E_{ij} = \frac{n_j n_i}{n} \tag{4.2}$$

Any continuous feature values should be transformed into nominal values prior to performing the χ^2 test; a process called *discretisation*. Liu and Setiono [1995] presented Chi2, an automated discretisation and feature selection algorithm using the χ^2 statistic. Chi2 is based upon ChiMerge [Kerber, 1992], an algorithm designed purely to discretise data. Equation 4.3 defines the ChiMerge algorithm for discretisation, let C be the total number of classes, A_{ij} be the number of examples in interval i of class j, E_{ij} be the expected frequency of $A_{ij} = R_i \times C_j / N$, R_i be the number of examples in interval $i = \sum_{j=1}^{C} A_{ij}$, C_j be the number of examples in class $j = \sum_{i=1}^{2} A_{ij}$, and N be the total number of examples $= \sum_{j=1}^{C} A_{ij}$ [Kerber, 1992; Hall, 1999].

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{C} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \tag{4.3}$$

Weka does not adopt the Chi2 or ChiMerge algorithms prior to applying the χ^2 statistic for feature selection [Witten and Frank, 2005]. Instead, Weka applies a discretisation method proposed by Fayyad and Irani [1993]; a minimum entropy heuristic which uses the Minimum Description Length Principle (MDLP) [Rissanen, 1978] to determine useful cut points for discretising the features. Liu et al. [2002] found using entropy with MDLP to be the best choice for data discretisation when compared to other methods including Chi2 and ChiMerge. The same discretisation filter is also applied to the entropy based feature selection algorithms Information Gain, Gain Ratio, and Symmetrical Uncertainty [Witten and Frank, 2005]; algorithms which also favour nominal values [Yu and Liu, 2003].

Equation 4.4 defines the entropy function for discretisation, which is applied recursively until a stop criterion is met. Let S_1 and S_2 be two intervals of set S bound by cut point T, A be the

feature, and Ent(S) (Eq. 4.5) be the entropy for a subset of S, with $P(C_i, S)$ is the proportion of examples in S with the class C_i [Fayyad and Irani, 1993; Hall, 1999].

$$E(A, T; S) = \frac{|S_1|}{|S|}\text{Ent}(S_1) + \frac{|S_2|}{|S|}\text{Ent}(S_2) \tag{4.4}$$

$$\text{Ent}(S) = -\sum_{i=1}^{C} P(C_i, S)\log_2(P(C_i, S)) \tag{4.5}$$

The criterion for stopping discretisation uses MDLP, where a partition induced by the cut point T is only accepted if encoding the partition costs less than the encoding prior to the split. This stop criterion is defined by Eq. 4.6, let N be the number of instances in the set S, and the distinct classes present in S, S_1, and S_2 be c, c_1, and c_2 respectively [Fayyad and Irani, 1993; Hall, 1999].

$$\text{Ent}(S) - E(A, T; S) > \frac{\log_2(N-1)}{N} + \frac{\log_2(3^c - 2) - [c\text{Ent}(S) - c_1\text{Ent}(S) - c_2\text{Ent}(S)]}{N} \tag{4.6}$$

4.6.2 Information Gain

Information Gain is a feature evaluation approach which uses entropy to evaluate the uncertainty of a feature [Quinlan, 1993]. This is achieved through evaluating how the inclusion of additional information provided by a feature reduces the entropy. The entropy of a feature X is determined by Eq. 4.7, with $P(x_i)$ being the frequency of value in X [Hall and Smith, 1999; Yu and Liu, 2003].

$$H(X) = -\sum_{i} P(x_i)\log_2(P(x_i)) \tag{4.7}$$

The entropy of feature X with the additional information provided by Y is calculated by Eq. 4.8, let $P(y_j)$ be the frequency of a value in feature Y, and $P(x_i|y_j)$ be the frequency of a value of X given the evidence of a value of Y. For classification, X is the class or outcome and Y is a feature of the problem.

$$H(X|Y) = -\sum_{j} P(y_j)\sum_{i} P(x_i|y_j)\log_2(P(x_i|y_j)) \tag{4.8}$$

Information Gain (Eq. 4.9) is calculated through the reduction in entropy of X following the information of X provided by Y.

$$IG(X|Y) = H(X) - H(X|Y) \tag{4.9}$$

A limitation of the Information Gain algorithm is the ability to only cater for nominal feature values. To resolve this, continuous data is first partitioned into nominal values. In Weka, this is achieved using the MDLP discretisation method.

4.6.3 Gain Ratio

A limitation of Information Gain is the algorithm's preference to select features containing a large quantity of values [Quinlan, 1993]. This is due to increased partitioning leading to an increased number of subsets which may only point to a single class. An example of this would be a unique identification feature for which each subset would only contain one case, and subsequently relate to only one class. This results in the maximal information gain $IG(X|Y) = 1$ despite the information being irrelevant for prediction. To overcome this, an extension of the Information Gain algorithm was developed called Gain Ratio. Gain Ratio attempts to normalise the information gain by factoring in the useful proportion of information.

The Gain Ratio (Eq. 4.10) is calculated through the division of the Information Gain (Eq. 4.9) for X given the evidence of Y by the entropy of feature Y (Eq. 4.7) [Hall, 1999].

$$GR(X,Y) = \frac{IG(X|Y)}{H(Y)} \tag{4.10}$$

Gain Ratio has been shown to be robust and provide consistently better results than Information Gain. However, evidence has shown that Gain Ratio can favour unbalanced splits when one particular subset is smaller than the others [Mingers, 1989].

4.6.4 Symmetrical Uncertainty

Symmetrical Uncertainty [Press et al., 1988] was developed to compensate for Information Gain's preference for features with more values, similar to Gain Ratio [Yu and Liu, 2003]. Symmetrical Uncertainty normalises the value to range $[0, 1]$, where 1 implies the knowledge provided by a value of Y always predicts the value of X, and 0 implies the values are completely independent of each other. Symmetrical Uncertainty is defined by Eq. 4.11.

$$SU(X,Y) = 2 \times \left[\frac{IG(X|Y)}{H(X) + H(Y)}\right] \tag{4.11}$$

4.6.5 One Rule

One Rule was designed with simplicity in mind [Holte, 1993]. The algorithm uses the error rate of a feature as opposed to the entropy approached used by Information Gain. Nevill-Manning et al. [1995] describe the pseudocode for the One Rule algorithm as displayed in Fig. 4.2.

One Rule was tested against C4.5 [Quinlan, 1993] (used to generate decision trees) on 16 datasets that are regularly used for evaluation. The results showed that despite the simple approach taken, One Rule was only marginally less accurate (by 3.1%) [Holte, 1993]. One Rule demonstrates that with most real-world data problems, rules can perform as well as more complex algorithms. In Weka, the One Rule feature selection algorithm performs discretisation using the same error

```
for each attribute a, form a rule as follows:
    for each value v from the domain of a,
        select the set of instances where a has value v.
        let c be the most frequent class in that set.
        add the following clause to the rule for a:
            if a has value v then the class is c
    calculate the classification accuracy of this rule.
use the rule with the highest classification accuracy
```

Figure 4.2: One Rule pseudocode [Nevill-Manning et al., 1995]

rate principle [Witten and Frank, 2005].

4.6.6 RELIEF-F

RELIEF-F [Kononenko, 1994] is an extension of the RELIEF [Kira and Rendell, 1992] instance based feature ranker. RELIEF was designed as an efficient method for estimating how the values of features are able to distinguish between instances which are near to each other; these values can either be discrete or continuous. This is achieved through finding a near-hit and near-miss instance. A near-hit is an instance which belongs to the same class and neighbourhood as the instance being evaluated. A near-miss is an instance in the same neighbourhood as the instance being evaluated but is not of the same class. A limitation of the original RELIEF algorithm is its ability to only deal with a two class problem, which would not be suitable for this research as there will be more than two classes of bolus insulin dose available. The RELIEF-F extension was devised to eliminate this constraint.

RELIEF-F is the result of incremental development of the RELIEF algorithm, starting with RELIEF-A [Kononenko, 1994]. RELIEF-A extended RELIEF to include more than one near-hit or near-miss. RELIEF-B, C and D implemented improvements upon RELIEF-A for dealing with incomplete datasets. Multi-class problems were introduced with RELIEF-E by including near-misses from each class present in the dataset. RELIEF-F improved this through averaging the contribution of the near-misses for each class. This was introduced to allow the algorithm to estimate the ability a feature has to separate two classes without considering if they are closest to each other.

Equation 4.12 defines RELIEF-F, let m be in the sample size, $diff(A, R, H)$ be the difference between the values of the features in instance R and H, $diff(A, R, M(C))$ be the the difference between the values of the features in instance R and the near miss instance $M(C)$, and $P(C)$ be the prior probability of the feature for the class.

$$W(A) := W(A) - \frac{diff(A, R, H)}{m} + \sum_{C \neq class(R)} \frac{P(C) \times diff(A, R, M(C))}{m} \tag{4.12}$$

4.6.7 Summary of the feature selection algorithms

This section introduced six feature selection algorithms provided by the Weka data-mining tool. Three of the algorithms - Information Gain, Gain Ratio and Symmetrical Uncertainty - use entropy to determine a feature's ability to classify. In contrast Chi-Squared applies the χ^2 statistic to predicting a class, One Rule applies error rate, and RELIEF-F applies a near-hit and near-miss method.

All the algorithms in this section with the exception of RELIEF-F require features and classes to be nominal values. As a result, all features with continuous values should be transformed into nominal values prior to applying the discussed feature selection algorithms, with the exception of RELIEF-F. Weka applies the entropy function described by Fayyad and Irani [1993] with MDLP to determine optimal cut points to perform discretisation. This method is used for all feature selection algorithms requiring nominal values with the exception of One Rule, which instead applies the same error rate principle used in the feature selection process.

It is expected that all the entropy based algorithms - Information Gain, Gain Ratio and Symmetrical Uncertainty - will result in similar feature weightings. However, it is possible that Gain Ratio and Symmetrical Uncertainty could produce more accurate weightings than Information Gain, as these algorithms improve upon Information Gain. The other algorithms - Chi-Squared, One Rule and RELIEF-F - all apply different methods to feature selection, and the results are likely to differ for each algorithm. In terms of execution speed, the One Rule algorithm is likely to out perform the other algorithms due to its simplistic approach. The optimal algorithm for this system is difficult to predict and will be determined during analysis and testing of the proposed CBR retrieval algorithm.

4.7 Temporal sequences

When seeking to solve a problem, it can be important to consider previous events. Events that precede a current problem may result in a change to the solution. This is a problem in the T1DM domain, as insulin remains active in the subject for several hours after the dose is administered. Any remaining active insulin in the subject needs to be considered when predicting a new solution. If the bolus from a previous a case is not considered, then there is a chance of insulin stacking resulting in potential hypoglycaemic events [Toffanin et al., 2013; Walsh and Roberts, 2015]. To address this and other temporal factors, an episode approach is used through the introduction of *temporal sequences* (episodes) to aid case retrieval [Sánchez-Marré et al., 2005].

In a standard approach, the new problem and retrieved cases are considered as isolated instances, and are independent of previous events [Sánchez-Marré et al., 2005]. Temporal sequences allow not only the current isolated instance of a problem to be considered, but also multiple previ-

ous events prior to the new problem. By factoring in previous events to the new problem, a similar sequence of events can be obtained. As the cases are the subject's continuous meal events ordered by time, these sequences occur naturally in the case-base.

Several different approaches for temporal case-based reasoning have been explored, with the majority focusing on the prediction of future events. Jaczynski [1997] proposed a CBR framework with the ability to retrieve cases composed of time series features. Each time series represents the evolution of a variable over time, catering for both numerical samples and changes of state. Jære et al. [2002] investigated the use of Allen's theory of temporal intervals [Allen, 1983] to avoid faults through prediction in the CBR system CREEK. Another approach to temporal reasoning is temporal projection, introduced by Branting and Hastings [1994]. Temporal projection shifts retrieved cases forwards or backwards through a method such as simulation to match the new problem. The case with the greatest similarity following a shift is then selected.

The aforementioned methods do not cater for the formation of temporal sequences from isolated cases. For this research, an approach similar to Sánchez-Marré et al. [2005] is used for temporal reasoning, which proposed that sequences of continuous temporal cases can be merged into a into a singular case. This method allows the temporal sequences to be compared using standard distance metrics without the need for additional rules. In the work conducted by Sánchez-Marré et al. [2005], plausible episodes are generated from a new problem, which are then compared to similar retrieved episodes in order to solve the new problem. Since the research in this thesis looks at preceding events, there is no need to generate a possible episode first, and instead the episode is created from the preceding cases.

For the CBR system presented in this research, a temporal sequence representing the new problem is formed by merging the isolated instance of the new problem with the preceding cases in the case-base. The temporal relation between these cases is defined by the time between the cases in the sequence, which is a feature of each case. Equation 4.13 defines the formation of a temporal sequence for a new problem, where the new problem P is a tuple of n number of f features, a retained case C is a tuple of n number of f features, the case-base CB is the sequence of retained cases C order by time, with $C_{|CB|}$ as the most recently retained case, and the temporal sequence TS_p is a sequence of the most recently retained cases in CB and the new problem P for a sequence

length of t.

$$1 \leq t \leq |CB| + 1$$

$$P = (f_1, f_2, ..., f_n)$$

$$C = (f_1, f_2, ..., f_n)$$

$$CB = \langle C_1, C_2, ..., C_{|CB|} \rangle$$

$$TS_p = \begin{cases} \langle P \rangle, & \text{if } t = 1 \\ \langle CB_{|CB|-((t-2)+0)}, CB_{|CB|-((t-2)+1)}, ..., CB_{|CB|}, P \rangle, & \text{otherwise.} \end{cases} \tag{4.13}$$

The case-base can be viewed as a sequence of retained case sequences order by time. Although the cases are not stored this way due to the repetition of cases, it helps to illustrate how the problem temporal sequence relates to the case-base during retrieval. Equation 4.14 describes a case-base of temporal sequences, let TS_{CB} be a sequence of sequences ordered by time, and each sequence contained by TS_{CB} consists of the retained cases $C_{i,...,t+(i-1)}$ ordered by time with a length of t.

$$1 \leq t \leq |CB|$$

$$TS_{CB} = \langle \langle C_1, C_2, ..., C_{t+0} \rangle \langle C_2, C_3, ..., C_{t+1} \rangle, ..., \langle C_{|CB|-((t-1)+0)}, C_{|CB|-((t-1)+1)}, ..., C_{|CB|} \rangle \rangle \tag{4.14}$$

The similarity between the problem sequence and a sequence within the case-base is calculated using a standard nearest neighbour algorithm on the singular cases formed from these sequences.

One issue with temporal sequences is the possibility of broken sequences where there are cases missing due to the user not recording a problem. A simple method for identifying such instances would be to impose a maximum time difference between cases. However, it is possible that this will not reflect the user's actual behaviour. Instead, outlier fences defined by Tukey [1977] (Eq. 4.15) are used to identify probable gaps. Any time difference in the sequence which falls outside the inner fence is considered an outlier and suggests a gap in the sequence.

$$\begin{aligned} \text{lower inner fence} &= Q_1 - 1.5 \times IQR \\ \text{upper inner fence} &= Q_3 + 1.5 \times IQR \end{aligned} \tag{4.15}$$

Simply limiting the sequence for instances where a gap is found would result in unexpected retrieval results, as the total number of features will be reduced in comparison to other sequences. Instead, for instances where a gap is found, the distance for the features is replaced by the maximum distance of 1 on the scale [0,1] as described by Montani et al. [2000].

4.8 Retrieval examples

In this section, worked examples of the retrieval process are described to illustrate how retrieval is achieved. The examples include both the Euclidean (Eq. 4.16) and Manhattan (Eq. 4.17) distance metrics on temporal sequences with the use of feature weightings. For both distance metrics, let p_i be a feature-value for the new problem, c_i be the corresponding feature-value for the retained case, w_i be the weighting for the feature, and n be the total number of features.

$$\begin{aligned} wed(p,c) &= \sqrt{\sum_{i=1}^{n} w_i(p_i - c_i)^2} \\ &= \sqrt{w_1(p_1 - c_1)^2 + w_2(p_2 - c_2)^2 + ... + w_n(p_n - c_n)^2} \end{aligned} \tag{4.16}$$

$$\begin{aligned} wmd(p,c) &= \sum_{i=1}^{n} w_i|p_i - c_i| \\ &= w_1|p_1 - c_1| + w_2|p_2 - c_2| + ... + w_n|p_n - c_n| \end{aligned} \tag{4.17}$$

Prior to calculating the distance the feature values must be normalised to a scale of [0,1] (Eq. 4.18). This normalisation process prevents features of different scales from distorting the result. In the normalisation function (Eq. 4.18), let f be the feature to normalise, f_i be the feature-value, f_{min} be the minimum value of the feature, and f_{max} be the maximum value of the feature.

$$n(f) = \frac{f_i - f_{min}}{f_{max} - f_{min}} \tag{4.18}$$

4.8.1 Example: Retrieval without temporal sequences

The first example describes the use of the similarity measure discussed in this chapter to identify the nearest neighbour case from a small example case-base (Table 4.2) in regards to a new problem (Table 4.3) without the use of temporal sequences.

The first step is to normalise the feature values. Normalisation factors in the feature values of both the case-base and the new problem. Example 4.8.1.1 demonstrates the normalisation of the preprandial blood glucose level. The normalisation process results in the values shown in Tables 4.4 and 4.5.

CB : case-base				
c_n : case	cc : carbohydrates (grams)	cbg : blood glucose level (mmol/L)	ct : time since previous meal (minutes)	ci : bolus insulin (insulin units)
c_1 : Case 1	50	4.00	120	4.00
c_2 : Case 2	30	6.00	240	2.00
c_3 : Case 3	60	5.00	180	4.50

Table 4.2: Example case-base

p : new problem		
pc : carbohydrates (grams)	pbg : blood glucose level (mmol/L)	pt : time since previous meal (minutes)
60	3.50	160

Table 4.3: Example new problem

Example 4.8.1.1 (Normalisation)**.**

$$\begin{aligned} f &= \langle pbg, c_1 \cdot cbg, c_2 \cdot cbg, c_3 \cdot cbg \rangle \\ &= \langle 3.50, 4.00, 6.00, 5.00 \rangle \end{aligned}$$

$$f_{min} = 3.50 \qquad f_{max} = 6.00$$

$$n(f_1) = \frac{3.50 - 3.50}{6.00 - 3.50} = 0.00 \qquad n(f_2) = \frac{4.00 - 3.50}{6.00 - 3.50} = 0.20$$

$$n(f_3) = \frac{6.00 - 3.50}{6.00 - 3.50} = 1.00 \qquad n(f_4) = \frac{5.00 - 3.50}{6.00 - 3.50} = 0.60$$

p : normalised new problem		
pc : carbohydrates	pbg : blood glucose level	pt : time since previous meal
1.00	0.00	0.33

Table 4.4: Example normalised new problem

CB : normalised case-base				
c_n : case	cc : carbohydrates	cbg : blood glucose level	ct : time since previous meal	bolus insulin
c_1 : Case 1	0.67	0.20	0.00	0.80
c_2 : Case 2	0.00	1.00	1.00	0.00
c_3 : Case 3	1.00	0.60	0.50	1.00

Table 4.5: Example normalised case-base

The last task to cover before performing the distance calculations is to weight each feature. Table 4.6 shows the weights to be used in this example for three features. The bolus insulin solution does not require weighting since it is only used when determining similarity with temporal sequences. The weights are normalised using the same method as the features; however, to avoid weights of 0.0, the minimum and maximum ranges of the feature selection algorithm are used instead of the minimum and maximum weights returned by the feature selection algorithm.

w : normalised weights		
wc : carbohydrates	wbg : blood glucose level	wt : time since previous meal
1.00	0.30	0.20

Table 4.6: Example normalised weights

Examples 4.8.1.2 and 4.8.1.3 illustrate the Euclidean and Manhattan distance calculations respectively for the new problem p against cases c_1, c_2, and c_3 in the example case-base. A lower distance implies greater similarity between the new problem and the case. In this example, both distance metrics state that c_3 has the lowest distance to the new problem, and c_2 has the highest distance. As a result c_3 is the nearest neighbour in this instance. The values used in the examples are rounded to two decimal places at each step, as a result some precision may be lost.

Example 4.8.1.2 (Euclidean distance)**.**

$$wed(p,c) = \sqrt{wc(pc-cc)^2 + wbg(pbg-cbg)^2 + wt(pt-ct)^2}$$

$$\begin{aligned} wed(p,c_1) &= \sqrt{1.00 \times (1.00-0.67)^2 + 0.30 \times (0.00-0.20)^2 + 0.20 \times (0.33-0.00)^2} \\ &= \sqrt{1.00 \times 0.11 + 0.30 \times 0.04 + 0.20 \times 0.11} \\ &= \sqrt{0.11 + 0.01 + 0.02} = \sqrt{0.14} = \mathbf{0.37} \end{aligned}$$

$$\begin{aligned} wed(p,c_2) &= \sqrt{1.00 \times (1.00-0.00)^2 + 0.30 \times (0.00-1.00)^2 + 0.20 \times (0.33-1.00)^2} \\ &= \sqrt{1.00 \times 1.00 + 0.30 \times 1.00 + 0.20 \times 0.45} \\ &= \sqrt{1.00 + 0.30 + 0.09} = \sqrt{1.39} = \mathbf{1.18} \end{aligned}$$

$$\begin{aligned} wed(p,c_3) &= \sqrt{1.00 \times (1.00-1.00)^2 + 0.30 \times (0.00-0.60)^2 + 0.20 \times (0.33-0.50)^2} \\ &= \sqrt{1.00 \times 0.00 + 0.30 \times 0.36 + 0.20 \times 0.03} \\ &= \sqrt{0.00 + 0.11 + 0.01} = \sqrt{0.12} = \mathbf{0.35} \end{aligned}$$

Example 4.8.1.3 (Manhattan distance)**.**

$$wmd(p,c) = wc|pc-cc| + wbg|pbg-cbg| + wt|pt-ct|$$

$$\begin{aligned} wmd(p,c_1) &= 1.00 \times |1.00-0.67| + 0.30 \times |0.00-0.20| + 0.20 \times |0.33-0.00| \\ &= 1.00 \times 0.33 + 0.30 \times 0.20 + 0.20 \times 0.33 \\ &= 0.33 + 0.06 + 0.07 = 0.46 = \mathbf{0.46} \end{aligned}$$

$$\begin{aligned} wmd(p,c_2) &= 1.00 \times |1.00-0.00| + 0.30 \times |0.00-1.00| + 0.20 \times |0.33-1.00| \\ &= 1.00 \times 1.00 + 0.30 \times 1.00 + 0.20 \times 0.67 \\ &= 1.00 + 0.30 + 0.13 = 1.43 = \mathbf{1.43} \end{aligned}$$

$$\begin{aligned} wmd(p,c_3) &= 1.00 \times |1.00-1.00| + 0.30 \times |0.00-0.60| + 0.20 \times |0.33-0.50| \\ &= 1.00 \times 0.00 + 0.30 \times 0.60 + 0.20 \times 0.17 \\ &= 0.00 + 0.18 + 0.03 = 0.21 = \mathbf{0.21} \end{aligned}$$

4.8.2 Example: Retrieval with temporal sequences

In this second example, the use of the distance metric with temporal sequences is demonstrated. The same new problem (Tables 4.3 and 4.4) and case-base (Tables 4.2 and 4.5) are used to form the temporal sequences, which in this example have a length of 2. The temporal sequence is ordered by the date and time of the cases such that the most recent case in the last element of the sequence. It is the last element of the sequence from which the reused solution is derived from. The cases are ordered such that c_2 occurs after c_1, c_3 after c_2, and the new problem after c_3. c_1 is omitted as there are no previous cases and temporal sequence cannot be formed. The process of creating the temporal sequences and transforming the sequences in to a new compound case is illustrated in Example 4.8.2.1. In the previous example there were three features, while in this example there are seven features in the compound case. This is because in the preceding cases the bolus insulin solution is also considered a feature.

Example 4.8.2.1 (Creating the temporal sequences)**.**

$$\begin{aligned} TS_p &= \langle c_3, p \rangle \\ &= \langle (1.00, 0.60, 0.50, 1.00), (1.00, 0.00, 0.33) \rangle \end{aligned}$$

$$\begin{aligned} TS_{c2} &= \langle c_1, c_2 \rangle \\ &= \langle (0.67, 0.20, 0.00, 0.80), (0.00, 1.00, 1.00) \rangle \end{aligned}$$

$$\begin{aligned} TS_{c3} &= \langle c_2, c_3 \rangle \\ &= \langle (0.00, 1.00, 1.00, 0.00), (1.00, 0.60, 0.50) \rangle \end{aligned}$$

In addition to the creation of the temporal sequences for this example, weights need to be defined for the new compound cases. A feature selection algorithm is again used to calculate these weights except with the compound cases instead of individual cases. The weights for this example are shown in Table 4.7.

w_1 : normalised weights for TS index 1			
wc_1 : carbohydrates	wbg_1 : blood glucose level	wt_1 : time since previous meal	wi_1 : bolus insulin
0.40	0.20	0.10	0.50

w_2 : normalised weights for TS index 2		
wc_2 : carbohydrates	wbg_2 : blood glucose level	wt_2 : time since previous meal
1.00	0.30	0.20

Table 4.7: Example normalised weights for the temporal sequence

Examples 4.8.2.2 and 4.8.2.3 demonstrate the aggregate matching for temporal sequences using the Euclidean and Manhattan distance metrics respectively. In this example both distance metrics

find that c_3 is the nearest neighbour when considering a temporal sequence length of 2 and the weightings from Table 4.7.

Example 4.8.2.2 (Euclidean distance)**.**

$$wed(p,c) = \sqrt{\begin{array}{l} wc_1(pc_1 - cc_1)^2 + wbg_1(pbg_1 - cbg_1)^2 + wt_1(pt_1 - ct_1)^2 \\ + wi_1(pi_1 - ci_1)^2 + wc_2(pc_2 - cc_2)^2 + wbg_2(pbg_2 - cbg_2)^2 \\ + wt_2(pt_2 - ct_2)^2 \end{array}}$$

$$wed(TS_p, TS_{c2}) = \sqrt{\begin{array}{l} 0.40 \times (1.00 - 0.67)^2 + 0.20 \times (0.60 - 0.20)^2 + 0.10 \times (0.50 - 0.00)^2 \\ + 0.50 \times (1.00 - 0.80)^2 + 1.00 \times (1.00 - 0.00)^2 + 0.30 \times (0.00 - 1.00)^2 \\ + 0.20 \times (0.33 - 1.00)^2 \end{array}}$$

$$= \sqrt{\begin{array}{l} 0.40 \times 0.11 + 0.20 \times 0.16 + 0.10 \times 0.25 \\ + 0.50 \times 0.04 + 1.00 \times 1.00 + 0.30 \times 1.00 \\ + 0.20 \times 0.45 \end{array}}$$

$$= \sqrt{0.04 + 0.03 + 0.03 + 0.02 + 1.00 + 0.30 + 0.09}$$

$$= \sqrt{1.51} = \mathbf{1.23}$$

$$wed(TS_p, TS_{c3}) = \sqrt{\begin{array}{l} 0.40 \times (1.00 - 0.00)^2 + 0.20 \times (0.60 - 1.00)^2 + 0.10 \times (0.50 - 1.00)^2 \\ + 0.50 \times (1.00 - 0.00)^2 + 1.00 \times (1.00 - 1.00)^2 + 0.30 \times (0.00 - 0.60)^2 \\ + 0.20 \times (0.33 - 0.50)^2 \end{array}}$$

$$= \sqrt{\begin{array}{l} 0.40 \times 1.00 + 0.20 \times 0.16 + 0.10 \times 0.25 \\ + 0.50 \times 1.00 + 1.00 \times 0.00 + 0.30 \times 0.36 \\ + 0.20 \times 0.03 \end{array}}$$

$$= \sqrt{0.40 + 0.03 + 0.03 + 0.50 + 0.00 + 0.11 + 0.01}$$

$$= \sqrt{1.08} = \mathbf{1.04}$$

Example 4.8.2.3 (Manhattan distance)**.**

$$\begin{aligned} wmd(p,c) = {} & wc_1|pc_1 - cc_1| + wbg_1|pbg_1 - cbg_1| + wt_1|pt_1 - ct_1| \\ & + wi_1|pi_1 - ci_1| + wc_2|pc_2 - cc_2| + wbg_2|pbg_2 - cbg_2| \\ & + wt_2|pt_2 - ct_2| \end{aligned}$$

$$\begin{aligned} wmd(TS_p, TS_{c2}) = {} & 0.40 \times |1.00 - 0.67| + 0.20 \times |0.60 - 0.20| + 0.10 \times |0.50 - 0.00| \\ & + 0.50 \times |1.00 - 0.80| + 1.00 \times |1.00 - 0.00| + 0.30 \times |0.00 - 1.00| \\ & + 0.20 \times |0.33 - 1.00| \\ \\ = {} & 0.40 \times 0.33 + 0.20 \times 0.40 + 0.10 \times 0.50 \\ & + 0.50 \times 0.20 + 1.00 \times 1.00 + 0.30 \times -1.00 \\ & + 0.20 \times -0.67 \\ \\ = {} & 0.13 + 0.08 + 0.05 + 0.10 + 1.00 + 0.30 + 0.13 \\ \\ = {} & \mathbf{1.79} \end{aligned}$$

$$\begin{aligned} wmd(TS_p, TS_{c3}) = {} & 0.40 \times |1.00 - 0.00| + 0.20 \times |0.60 - 1.00| + 0.10 \times |0.50 - 1.00| \\ & + 0.50 \times |1.00 - 0.00| + 1.00 \times |1.00 - 1.00| + 0.30 \times |0.00 - 0.60| \\ & + 0.20 \times |0.33 - 0.50| \\ \\ = {} & 0.40 \times 1.00 + 0.20 \times -0.40 + 0.10 \times -0.50 \\ & + 0.50 \times 1.00 + 1.00 \times 0.00 + 0.30 \times -0.60 \\ & + 0.20 \times -0.17 \\ \\ = {} & 0.40 + 0.08 + 0.05 + 0.50 + 0.00 + 0.18 + 0.03 \\ \\ = {} & \mathbf{1.24} \end{aligned}$$

4.9 Reuse

The purpose of the reuse stage is to produce a predicted solution for the new problem. This predicted solution is determined by the case(s) obtained from the retrieval stage. Following retrieval, the predicted solution (a bolus insulin dose) is simply formed from the bolus doses contained by the retrieved cases. This is extended in Section 4.10 to include an adaptation rule to help further reduce the effect of insulin stacking.

The predicted bolus insulin solution for the new problem is achieved by taking the solution from the retrieved cases. In a temporal sequence, the solution is taken from the last case in the sequence, which corresponds to the position of the new problem in the sequence. Prior to reuse, the retrieved cases are sorted by their degree of similarity, from which the defined k-nearest neighbours (k-NN)

of greatest similarity are selected for reuse. In the instance of 1-NN, the solution of the single retrieved case is used to solve the new problem. For more than 1-NN, a simple k-NN regression approach is used, taking the mean of the k retrieved bolus insulin values. Equation 4.19 defines the reuse strategy, let b_i define bolus insulin solution provided by a retrieved case.

$$\text{suggested bolus insulin dose} = \frac{1}{k}\sum_{i=1}^{k} b_i \tag{4.19}$$

4.10 Adaptation

The use of temporal sequences in the retrieval phase aims to reduce the effect of insulin stacking occurring through the inclusion of preceding bolus and time intervals in the retrieval algorithm. However, it is probable that the nearest neighbour(s) will not exactly match the active insulin profile of the new problem. The adaptation rule introduced in this section aims to adjust the bolus advice in order to reduce the negative effects of insulin stacking, which can result in hypoglycaemic episodes. This is applies a similar concept to the CBR system JUDGE [Bain, 1986], which reduces or increases the length of a sentence based on differences between the new problem and retrieved case.

Insulin on board (IOB) is included by insulin pumps and bolus calculators to account for insulin stacking. Insulin stacking is the accumulative effect of multiple bolus doses and can result in an increased risk of hypoglycaemic episodes [Toffanin et al., 2013; Walsh and Roberts, 2015]. Bolus insulin calculators deduct IOB from the suggested bolus dose to account for the insulin which remains active in the subject (Eq. 4.20 [RapidCalc, 2015]). Insulin on board will be included in the reuse step of the CBR model to adjust retrieved bolus dose solutions to better reflect the current problem.

$$\text{suggested insulin} = \text{correction dose} + \text{meal dose} - \text{insulin on board} \tag{4.20}$$

The duration and levels of active insulin in the subject is dependent upon the type and brand of the insulin, and also environmental factors [Heinemann, 2004]. Research into rapid action insulin activity has shown that the level of active insulin remaining over its duration is not linear [Heinemann, 2004]. Insulin activity usually has an offset of 15 to 20 minutes, and after approximately 3 hours only 40% of the insulin remains active [Walsh et al., 2011]. Other studies show a peak in insulin activity at 2 hours following the dose, followed by a rapid decline, with an estimated 6 hour duration [Woodworth et al., 1994]. Another source states that only 50% of the insulin remains active after 2 hours, which then gradually decreases over the next 4 hours, for a total duration of 6 hours [Walsh and Roberts, 2015].

Although the levels of active insulin have been shown not to follow a linear decline, many insulin

pumps and bolus calculators use a linear formula to calculate the active insulin remaining [Walsh and Roberts, 2015]. Campbell and Abramovich [2012] describe one such linear IOB formula for insulin pumps (Eq. 4.21) which will be used in this research. Walsh et al. [2011] state that the most effective active insulin duration in linear calculations should be between 4 and 5 ½ hours.

Insulin on board is calculated by multiplying the previous insulin dose by the time remaining of the active insulin time shown in Eq. 4.21. The time difference must be greater than 0 and less than the defined total active insulin time to avoid negative IOB values.

For the subsequent formulas in this section, the variables are defined as follows: the case-base CB is a sequence of cases c, with each case c a tuple of case time ct in minutes and the bolus insulin dose ci. t denotes time in minutes, pt is the time of a new problem in minutes, and rt is the time of the retrieved case in minutes. RC is a sequence of case times rt in minutes for $k > 1$ nearest neighbour retrievals. The active insulin time a is a constant to reflect the duration of a bolus insulin dose in minutes. The suggested bolus insulin dose i is the original bolus insulin dose to be adapted. The type definitions for these variables are described below.

$$t : \mathbb{N},\ pt : \mathbb{N},\ a : \mathbb{N}_1,\ k : \mathbb{N}_1,\ i : \mathbb{R}_0^+$$
$$rt : \mathbb{N},\ RC : \langle rt_1, rt_2, ..., rt_k \rangle$$
$$c = (ct, ci) : \mathbb{N} \times \mathbb{R}_0^+,\ CB = \langle c_1, c_2, ..., c_{|CB|} \rangle$$

$$\text{iob}(c, t, a) = \begin{cases} ci \times \left(1 - \dfrac{t - ct}{a}\right), & \text{if } a > t - ct > 0 \\ 0.0, & \text{otherwise.} \end{cases} \tag{4.21}$$

It is possible that there is more than one previous bolus insulin dose for which active insulin remains in the subject. As a result, IOB is calculated by for all cases where the time difference is less than the active insulin time (Eq. 4.22).

$$\text{iobsum}(CB, t, a) = \sum_{n=1}^{|CB|} \text{iob}(c_n, t, a) \tag{4.22}$$

The adjustment to be applied to the insulin solution is calculated by deducting the IOB for the retrieved case from the IOB of the new problem (Eq. 4.23).

$$\text{a}_a(pt, rt, CB, a) = \text{iobsum}(CB, pt, a) - \text{iobsum}(CB, rt, a) \tag{4.23}$$

Equation 4.24 adjusts the original insulin suggestion using the adjustment value calculated in Eq. 4.23. If the IOB for the new problem is greater than that of the IOB for the retrieved case, there is presently more active insulin within the subject. As a result, the suggested bolus insulin needs to be reduced to reflect the fact that the original bolus insulin suggestion was obtained when

there was less active insulin within the subject. In the reverse situation where the problem IOB is less than the retrieved case IOB, the bolus insulin suggestion needs to be increased.

$$\mathrm{ai}_a(i, pt, rt, CB, a) = \begin{cases} i - \mathrm{a}_a(pt, rt, CB, a), & \text{if } i - \mathrm{a}_a(pt, rt, CB, a) \geq 0 \\ 0.0, & \text{otherwise.} \end{cases} \tag{4.24}$$

When handling more than 1-NN, an average of the sum of the IOBs for all the retrieved cases is used instead (Eq. 4.25). In this situation, the retrieved case time is replaced by a sequence of retrieved case times, where the size of the sequence is equal to k-NN. Equation 4.26 revises the original bolus suggestion adjustment for $k > 1$ retrievals.

$$\mathrm{a}_b(pt, RC, CB, a) = \mathrm{iobsum}(CB, pt, a) - \frac{\sum_{n=1}^{k} \mathrm{iobsum}(CB, RC_n, a)}{k} \tag{4.25}$$

$$\mathrm{ai}_b(i, pt, RC, CB, a) = \begin{cases} i - \mathrm{a}_b(pt, RC, CB, a), & \text{if } i - \mathrm{a}_b(pt, RC, CB, a) \geq 0 \\ 0.0, & \text{otherwise.} \end{cases} \tag{4.26}$$

4.10.1 Examples

To illustrate how the IOB adjustment is applied to the original bolus insulin suggestion, three examples are given. Example 4.10.1.1 and example 4.10.1.2 demonstrates scenarios where the IOB adjustment will result in a decrease and increase of the suggested bolus insulin dose respectively. Example 4.10.1.3 demonstrates an IOB adjustment in a 2-NN scenario.

Example 4.10.1.1 (Decrease in suggested bolus insulin)**.**

This example illustrates the adjusted bolus insulin using IOB adjustment for the new problem with a time (pt) of 900 minutes, and a retrieved case of time (rt) 540 minutes. The retrieved case time relates to Case 3 ($c3$) within the case-base (CB). The case-base in this example contains four cases. The original bolus insulin suggestion (i) of 5.0 is also defined by Case 3 ($c3$). The active insulin time (a) is defined as 240 minutes. All values are rounded to one decimal place.

In this example, the IOB for the new problem is greater than the IOB of the retrieved case. As a result, the adjusted bolus insulin suggestion will be reduced as there is more active insulin present in the new problem than in the retrieved case's bolus insulin solution.

i : original bolus insulin suggestion (insulin units)	pt : problem time (minutes)	rt : retrieved case time (minutes)	a : active insulin time (minutes)
5.0	900	540	240

Case-base (CB)		
c_n : case	ct : time (minutes)	ci : bolus insulin (insulin units)
Case 1 (c_1)	180	5.0
Case 2 (c_2)	360	4.0
Case 3 (c_3)	540	5.0
Case 4 (c_4)	720	6.0

$$\begin{aligned}
\mathrm{a}_a(pt, rt, CB, a) &= \mathrm{iobsum}(CB, pt, a) - \mathrm{iobsum}(CB, rt, a) \\
&= (\mathrm{iob}((180, 5.0), 900, 240) + \mathrm{iob}((360, 4.0), 900, 240) \\
&\quad + \ \mathrm{iob}((540, 5.0), 900, 240) + \mathrm{iob}((720, 6.0), 900, 240)) \\
&\quad - \ (\mathrm{iob}((180, 5.0), 540, 240) + \mathrm{iob}((360, 4.0), 540, 240) \\
&\quad + \ \mathrm{iob}((540, 5.0), 540, 240) + \mathrm{iob}((720, 6.0), 540, 240)) \\
&= (0.0 + 0.0 + 0.0 + 1.5) - (0.0 + 1.0 + 0.0 + 0.0) \\
&= 1.5 - 1.0 = \mathbf{0.5}
\end{aligned}$$

$$\begin{aligned}
\mathrm{ai}_a(i, pt, rt, CB, a) &= \begin{cases} i - \mathrm{a}_a(pt, rt, CB, a), & \text{if } i - \mathrm{a}_a(pt, rt, CB, a) \geq 0 \\ 0.0, & \text{otherwise.} \end{cases} \\
&= 5.0 - 0.5 = \mathbf{4.5 \text{ insulin units}}
\end{aligned}$$

Example 4.10.1.2 (Increase in suggested bolus insulin)**.**

The second example illustrates the adjusted bolus insulin using IOB adjustment for the new problem with a the time (pt) of 800 minutes, and a retrieved case of time (rt) 320 minutes. The retrieved case time relates to Case 2 ($c2$) within the case-base (CB). The case-base in this example also contains four cases. The original bolus insulin suggestion (i) of 3.0 is also defined by Case 2 ($c2$). The active insulin time (a) is defined as 260 minutes. All values are rounded to one decimal place.

In this example, the IOB for the new problem is less than the IOB of the retrieved case. As a result the amended bolus insulin suggestion will be increased as there is less active insulin present than in the retrieved case's bolus insulin solution.

i : original bolus insulin suggestion (insulin units)	pt : problem time (minutes)	rt : retrieved case time (minutes)	a : active insulin time (minutes)
3.0	800	320	260

Case-base (CB)		
c_n : case	ct : time (minutes)	ci : bolus insulin (insulin units)
Case 1 (c_1)	200	7.0
Case 2 (c_2)	320	3.0
Case 3 (c_3)	520	5.0
Case 4 (c_4)	700	3.0

$$
\begin{aligned}
\mathrm{a}_a(pt, rt, CB, a) &= \mathrm{iobsum}(CB, pt, a) - \mathrm{iobsum}(CB, rt, a) \\
&= (\mathrm{iob}((200, 7.0), 800, 260) + \mathrm{iob}((320, 3.0), 800, 260) \\
&\quad + \ \mathrm{iob}((520, 5.0), 800, 260) + \mathrm{iob}((700, 3.0), 800, 260)) \\
&\quad - \ (\mathrm{iob}((200, 7.0), 320, 260) + \mathrm{iob}((320, 3.0), 320, 260) \\
&\quad + \ \mathrm{iob}((520, 5.0), 320, 260) + \mathrm{iob}((700, 3.0), 320, 260)) \\
&= (0.0 + 0.0 + 0.0 + 1.8) - (3.8 + 1.0 + 0.0 + 0.0) \\
&= 1.8 - 3.8 = \textbf{-2.0}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{ai}_a(i, pt, rt, CB, a) &= \begin{cases} i - \mathrm{a}_a(pt, rt, CB, a), & \text{if } i - \mathrm{a}_a(pt, rt, CB, a) \geq 0 \\ 0.0, & \text{otherwise.} \end{cases} \\
&= 3.0 - (-2.0) = \textbf{5.0 insulin units}
\end{aligned}
$$

Example 4.10.1.3 (Multiple retrieved cases (2-NN))**.**

The final example illustrates the adjusted bolus insulin using IOB adjustment for the new problem with a time (pt) of 1720 minutes, and a sequence of two retrieved cases (RC) with the times (ct) $\langle 520, 1020 \rangle$ minutes. The retrieved case time relate to Cases 3 and 6 (c_3, c_6) within the case-base (CB), which in this example contains ten cases. The original bolus insulin suggestion (i) of 5.5 is also defined by the average bolus insulin solution (ci) of Cases 3 and 6 (c_3, c_6). The active insulin time (a) is defined as 320 minutes. All values are rounded to one decimal place.

In this example, the IOB for the new problem is greater than the IOB of the retrieved case. As a result, the adjusted bolus insulin suggestion will be reduced as there is more active insulin present in the new problem than in the retrieved case's bolus insulin solution.

i : original bolus insulin suggestion (insulin units)	pt : problem time (minutes)	RC : retrieved case times (minutes)	a : active insulin time (minutes)	k : k-NN
5.5	1720	$\langle 520, 1020 \rangle$	320	2

Case-base (CB)		
c_n : case	ct : time (minutes)	ci : bolus insulin (insulin units)
Case 1 (c_1)	200	7.0
Case 2 (c_2)	320	3.0
Case 3 (c_3)	520	5.0
Case 4 (c_4)	700	2.5
Case 5 (c_5)	840	3.0
Case 6 (c_6)	1020	6.0
Case 7 (c_7)	1180	4.0
Case 8 (c_8)	1320	1.5
Case 9 (c_9)	1500	6.0
Case 10 (c_{10})	1600	0.5

$$
\begin{aligned}
\mathrm{a}_b(pt, RC, CB, a) &= \mathrm{iobsum}(CB, pt, a) - \frac{\sum_{n=1}^{k} \mathrm{iobsum}(CB, RC_n, a)}{k} \\
&= (\mathrm{iob}((200, 7.0), 1720, 320) + \mathrm{iob}((320, 3.0), 1720, 320) \\
&\quad + \ \mathrm{iob}((520, 5.0), 1720, 320) + \mathrm{iob}((700, 2.5), 1720, 320) \\
&\quad + \ \mathrm{iob}((840, 3.0), 1720, 320) + \mathrm{iob}((1020, 6.0), 1720, 320) \\
&\quad + \ \mathrm{iob}((1180, 4.0), 1720, 320) + \mathrm{iob}((1320, 1.5), 1720, 320) \\
&\quad + \ \mathrm{iob}((1500, 6.0), 1720, 320) + \mathrm{iob}((1600, 0.5), 1720, 320)) \\
&\quad - \ (\mathrm{iob}((200, 7.0), 520, 320) + \mathrm{iob}((320, 3.0), 520, 320) \\
&\quad + \ \mathrm{iob}((520, 5.0), 520, 320) + \mathrm{iob}((700, 2.5), 520, 320) \\
&\quad + \ \mathrm{iob}((840, 3.0), 520, 320) + \mathrm{iob}((1020, 6.0), 520, 320) \\
&\quad + \ \mathrm{iob}((1180, 4.0), 520, 320) + \mathrm{iob}((1320, 1.5), 520, 320) \\
&\quad + \ \mathrm{iob}((1500, 6.0), 520, 320) + \mathrm{iob}((1600, 0.5), 520, 320)) \\
&\quad + \ \mathrm{iob}((200, 7.0), 1020, 320) + \mathrm{iob}((320, 3.0), 1020, 320) \\
&\quad + \ \mathrm{iob}((520, 5.0), 1020, 320) + \mathrm{iob}((700, 2.5), 1020, 320) \\
&\quad + \ \mathrm{iob}((840, 3.0), 1020, 320) + \mathrm{iob}((1020, 6.0), 1020, 320) \\
&\quad + \ \mathrm{iob}((1180, 4.0), 1020, 320) + \mathrm{iob}((1320, 1.5), 1020, 320) \\
&\quad + \ \mathrm{iob}((1500, 6.0), 1020, 320) + \mathrm{iob}((1600, 0.5), 1020, 320)) \\
&\quad \div 2) \\
&= (0.0 + 0.0 + 0.0 + 0.0 + 0.0 \\
&\quad + \ 0.0 + 0.0 + 0.0 + 1.9 + 0.3) \\
&\quad - \ (0.0 + 1.1 + 0.0 + 0.0 + 0.0 \\
&\quad + \ 0.0 + 0.0 + 0.0 + 0.0 + 0.0 \\
&\quad + \ 0.0 + 0.0 + 0.0 + 0.0 + 1.3 \\
&\quad + \ 0.0 + 0.0 + 0.0 + 0.0 + 0.0) \\
&\quad \div 2) \\
&= 2.2 - \frac{2.4}{2} = 2.2 - 1.2 = \mathbf{1.0}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{ai}_b(i, pt, RC, CB, a) &= \begin{cases} i - \mathrm{a}_b(pt, RC, CB, a), & \text{if } i - \mathrm{a}_b(pt, RC, CB, a) \geq 0 \\ 0.0, & \text{otherwise.} \end{cases} \\
&= 5.5 - 1.0 = \mathbf{4.5 \text{ insulin units}}
\end{aligned}
$$

4.11 Revise

The revision step involves evaluation of a solution [Aamodt and Plaza, 1994]. This evaluation sets out to determine if the solution was desirable, and is key to CBRs ability to learn. One approach to performing revision is through the use of a domain expert to alter the solution, which may be human or machine. The CBR systems CHEF and CASEY demonstrate the use of expert systems to aid revision. In the case of CHEF, a simulator is used to try the proposed recipe, allowing the identification of any faults in its plan [Hammond, 1986]. CASEY also uses a third party application; in this instance, one built for heart condition diagnosis [Koton, 1988]. However, this heart condition diagnosis system is only used when CASEY itself fails to reach a solution. It is important that evaluation and revision occurs, as CBR is reliant on the quality of the solutions retained in the case-base [Aamodt and Plaza, 1994].

In context of this research, revision can be a manual process where the user specifies a different

bolus dose from their observations. However, an automated approach is more desirable. This section describes an algorithm which attempts to automate this process through the assessment of a postprandial blood glucose reading. A postprandial blood glucose reading is taken after a meal by the subject to provide an indication of the effect the preceding bolus dose had on their blood glucose control. An automated approach to evaluation will help to simplify this process for the user of the system.

At this stage of the CBR cycle, the system has produced a proposed solution (bolus advice), performed some form of adaptation if required, and is now awaiting validation of the solution. The solution can be validated using a simulator as is the case for testing this application. However, in real-world use this is not a feasible option, since the simulator may not accurately reflect the behaviour of the subject, and is reliant on the user having access to a simulator. This leads to evaluating the solution through the subject's own observations. A postprandial blood glucose reading allows the subject to observe the effect of a bolus dose on their blood glucose level. If the postprandial blood glucose reading deviates away from the target blood glucose level, the bolus advice can be improved. If the blood glucose level falls below the target level then a smaller dose may have been more successful, and if the result is a higher blood glucose reading than the target level then a larger dose may have been more successful. Postprandial blood glucose tests also allow the user to identify the risk of hyperglycaemic and hypoglycaemic episodes caused by either too little or too much bolus insulin administered respectively.

Becton, Dickinson and Company [2005] describe a method (Eq. 4.30) for correcting bolus doses based on the subject's *insulin sensitivity factor* (ISF). The ISF can be calculated using the subject's *total daily insulin dose* (TDD) [Davidson et al., 2008]. The inclusion of this correction method within the CBR model will allow for automated revision of a solution prior to retaining a case, aiding the learning process.

To calculate TDD based on guidelines by University College London Hospitals [2013], the sum of 4 days of bolus and basal insulin doses is divided by four. Equation 4.27 describes this calculation, let I represent the sequence of bolus and basal doses over a period of d days, I_i be an individual bolus or basal dose from the sequence of insulin doses I.

$$TDD = \frac{\sum_{i=1}^{|I|} I_i}{d} \tag{4.27}$$

In situations where there is not sufficient data to calculate the subject's TDD, then an estimate can be calculated (Eq. 4.28) using the subject's body weight w in pounds [Davidson et al., 2008].

$$TDD = 0.24 \times w \tag{4.28}$$

To calculate the ISF, University College London Hospitals [2013] suggests the use of the 100 rule for blood glucose measured in mmol/L, the equivalent for mg/dL is the 1800 rule. However,

Davidson et al. [2003] undertook extensive statistical studies of various constants and suggests the 1,700 rule, which was applied to active insulin management (AIM) system [Davidson et al., 2008]. The 1,700 rule is an established recommendation [King and Armstrong, 2007], and will be used in this research (Eq. 4.29). The ISF is converted to mmol/L by multiplying the result by the constant 0.0555.

$$ISF = \frac{1700}{TDD} \times 0.0555 \text{ mmol/L} \tag{4.29}$$

Finally, to calculate the bolus insulin revision the subject's postprandial blood glucose level *pbg*, target blood glucose level *tbg*, and ISF are applied to Eq. 4.30.

$$\text{insulin adjustment} = \frac{pbg - tbg}{ISF} \tag{4.30}$$

To demonstrate the use of the postprandial revision algorithm, two insulin adjustments are demonstrated in the following subsection. The approach will be analysed and evaluated in Chapter 5 to discover if this algorithm will allow the CBR system to improve future bolus suggestions through the revision of suboptimal suggestions.

4.11.1 Examples

To demonstrate the postprandial revision algorithm, two insulin adjustment examples are provided. The insulin doses used to calculate the total daily dose *TDD* and subsequently the insulin sensitivity factor *ISF* for these examples are displayed in Table 4.8. In this table both bolus and basal doses are shown over the period of four days, where each day has a single basal dose and between three and four bolus doses.

Insulin type	Day 1 (IU)	Day 2 (IU)	Day 3 (IU)	Day 4 (IU)
Bolus	4	5	4	6
	6	7	8	5
	8	7	6	5
		2		3
Basal	20	20	20	20

Table 4.8: Example insulin doses over a four day period

Table 4.9 shows the postprandial blood glucose reading *pbg* and target blood glucose level *tbg* for Examples 4.11.1.3 and 4.11.1.4.

pbg: postprandial blood glucose (mmol/L)	*tbg* : target blood glucose (mmol/L)
8.0	6.5
4.0	6.5

Table 4.9: Example postprandial blood glucose readings

With the values defined, the postprandial insulin adjustment can be calculated. Firstly the

TDD is calculated in Example 4.11.1.1. In this calculation the sequence I contains both the basal and bolus dose values from Table 4.8, and the number of days d is 4. In this example, the resulting TDD value is 39 IU.

Example 4.11.1.1 (Calculate total daily dose (TDD))**.**

$$
\begin{aligned}
I &= \langle 4, 6, 8, 20, 5, 7, 7, 2, 20, 4, 8, 6, 20, 6, 5, 5, 3, 20 \rangle \\
d &= 4 \\
TDD &= \frac{\sum_{i=1}^{|I|} I_i}{d} \\
&= \frac{156}{4} \\
&= \mathbf{39}
\end{aligned}
$$

Following the calculation of the TDD is the ISF calculation. This is demonstrated in Example 4.11.1.2, with the resulting ISF of 2.42. This ISF value will be used in both Example 4.11.1.3 and 4.11.1.4.

Example 4.11.1.2 (Calculate insulin sensitivity factor (ISF))**.**

$$
\begin{aligned}
ISF &= \frac{1700}{TDD} \times 0.0555 \\
&= \frac{1700}{39} \times 0.0555 \\
&= \mathbf{2.42}
\end{aligned}
$$

With all the information now acquired, the bolus insulin adjustment can be obtained. The calculation of the two example postprandial blood glucose readings and target blood glucose values from Table 4.9 are shown in Examples 4.11.1.3 and 4.11.1.4.

Example 4.11.1.3 demonstrates that the bolus insulin dose should be increased by 0.62 IU since the postprandial blood glucose reading is higher than the target blood glucose level. In contrast, Example 4.11.1.4 demonstrates that the bolus insulin dose should be decreased by 1.03 IU since the postprandial blood glucose reading is lower than the target blood glucose level.

Example 4.11.1.3 (Increase in bolus insulin)**.**

$$
\begin{aligned}
\text{insulin adjustment} &= \frac{pbg - tbg}{ISF} \\
&= \frac{8.0 - 6.5}{2.42} \\
&= \mathbf{0.62}
\end{aligned}
$$

Example 4.11.1.4 (Decrease in bolus insulin)**.**

$$\begin{aligned}\text{insulin adjustment} &= \frac{pbg - tbg}{ISF} \\ &= \frac{4.0 - 6.5}{2.42} \\ &= \textbf{-1.03}\end{aligned}$$

4.12 Retain

The retain step of the CBR cycle stores the solved problem and its solution in the case-base for use in future cycles. The retain step is simple in this system, since no indexing has been adopted due to the consistency of the case structure. The retained case is comprised of the features defining the problem (carbohydrates, time, and a preprandial blood glucose reading), and the revised solution (bolus dose).

Additional features may also be stored alongside these in order to store further information and help facilitate data visualisation. This may include the postprandial blood glucose readings and the bolus suggestion prior to revision.

4.13 Summary

In this chapter, a CBR system for T1DM bolus advice based on the R^4 was introduced [Aamodt and Plaza, 1994]. This initially involved identifying suitable case features from existing state-of-the-art bolus calculators and the UVa/Padova T1DM simulator. Features which remain static are discounted as they would not aid retrieval. Additionally, some features including physical activity, and time period based insulin sensitivity factors and carbohydrate-to-insulin ratios are also omitted due to limitations of the simulator. Through this process it was found that planned carbohydrate intake, preprandial blood glucose level, and time were required to represent a case.

To retrieve cases similar to a new problem a weighted nearest neighbour distance metric is used. This similarity measure was then enhanced through feature weightings obtained through single-attribute feature selection algorithms available to the Weka data mining tool. Finally, temporal sequences were introduced to include preceding events when determining similarity.

An adaptation rule to reduce the risk of insulin stacking and a method for automated bolus solution evaluation was introduced to the system to optimise the retrieved advice. This adaptation rule uses substitution adaptation to reconcile differences in active insulin between the new problem and the retrieved cases. Through introducing this adaptation rule, the system is able to adjust the solution in order to decrease the bolus dose if the new problem has a higher level of active insulin than the retrieved case, and increase the bolus advice if the new problem has a lower level level of active insulin than the retrieved case.

Postprandial evaluation was introduced to provide the CBR system with a method to repair and learn from faults. This is fundamental to the CBR process, as poor solutions need to be corrected in order to improve future solutions. Two methods of evaluation were proposed, the first a manual adjustment to the bolus insulin dose based upon the subject's observations following a bolus. The second is a method to automate this process to encourage users to perform an evaluation, and to provide confidence in the evaluation. The automated approach will increase or decrease the bolus solution based on the deviation away from the subject's target blood glucose level using a postprandial blood glucose reading. If the postprandial blood glucose reading is greater than the target blood glucose level, the bolus dose is revised to increase the dose. For the opposite situation where the postprandial blood glucose reading is lower than the target, the dose is decreased.

The next chapter analyses and evaluates the system proposed in this chapter. This process will test the effectiveness of the retrieval algorithm presented, the performance of the insulin on board adaptation, and the system ability to learn and improve through automated postprandial evaluation and revision. The analysis and evaluation will also compare the CBR system to existing methods of obtaining bolus advice.

Chapter 5

Analysis and evaluation

This chapter describes the analysis and evaluation of the CBR system proposed in Chapter 4. Firstly, statistical measures suitable for evaluating blood glucose management are identified. This is followed by a discussion of the testing methodology. Testing begins with the generation of problem sets and sample case-bases. These sample case-bases are then evaluated to determine feature weightings through the use of feature selection algorithms. The problem sets are tested against the sample case-bases with different retrieval configurations. The resulting CBR bolus insulin suggestions are then simulated using the UVa/Padova T1DM simulator to obtain continuous blood glucose levels for statistical analysis. The CBR retrieval results are then compared to those obtained through closed-loop simulation and a state-of-the-art bolus calculation formula.

Analysis and evaluation of the adaptation algorithm is then discussed, where retrieved solutions are adjusted through resolving any differences in the insulin on board from the new problem and retrieved case. Finally, the chapter concludes with analysis and evaluation of the automated revision algorithm, which aims to improve future bolus advice through the revision of suboptimal solutions.

5.1 Statistical measures for evaluating bolus insulin suggestions

Prior to devising a test plan for the CBR system, statistical measures need to be identified for analysing and evaluating the bolus advice obtained through case retrieval. The data available for testing is limited to information about the meals themselves and continuous blood glucose data retrieved from the simulator. For testing purposes, the blood glucose data provides the best representation of the subject's well-being, as T1DM management fundamentally revolves around maintaining safe blood glucose levels. The continuous blood glucose data is generated through

open-loop simulation of the meal information input into the CBR system alongside the bolus insulin suggested by the reuse of retrieved cases. This continuous blood glucose data provides a simulated blood glucose reading for every minute of the simulation.

Several statistical measures have been identified for use on continuous blood glucose data [Clarke and Kovatchev, 2009]: blood glucose risk index (BGRI), low blood glucose risk index (LBGI), high blood glucose risk index (HBGI), time within target blood glucose range, mean, and variance.

5.1.1 Blood glucose risk index

The BGRI can be applied to continuous blood glucose data to determine overall variance of LBGI and HBGI [Kovatchev et al., 1998, 2006; Clarke and Kovatchev, 2009]. LBGI is used as an early indicator for detecting potential hypoglycaemic events, whilst HBGI is used as an indicator of hyperglycaemic events. These risk indexes are obtained by splitting the data into low and high glucose values to assess the variance independently of each other. Defined boundaries (Table 5.1) have been outlined for LBGI and HBGI values to determine the risk level [Kovatchev et al., 1998, 2006; Roche, 2015b].

Risk level	LBGI	HBGI
Minimal	$x \leq 1.1$	$x < 5.0$
Low	$1.1 < x \leq 2.5$	$5.0 \leq x \leq 10.0$
Medium	$2.5 < x \leq 5.0$	$10.0 < x \leq 15.0$
High	$x > 5.0$	$x > 15.0$

Table 5.1: Low and high blood glucose risk index severity levels [Roche, 2015b]

The LBGI and HBGI are calculated by firstly applying Eq. 5.1 for blood glucose readings bg in mmol/L and Eq. 5.2 for blood glucose readings bg in mg/dL.

$$f(bg) = 1.509 \times (\ln(bg)^{1.084} - 5.381) \tag{5.1}$$

$$f(bg) = 1.509 \times (\ln(18 \times bg)^{1.084} - 5.381) \tag{5.2}$$

The risk function $r(bg)$ for each blood glucose is calculated by Eq. 5.3 of the readings.

$$r(bg) = 10 \times f(bg) \tag{5.3}$$

The results of the risk function are split into two branches rl for low blood glucose readings (Eq. 5.4), and rh for high blood glucose readings (Eq. 5.5).

$$rl(bg) = \begin{cases} r(bg), & \text{if } f(bg) < 0 \\ 0, & \text{otherwise.} \end{cases} \tag{5.4}$$

$$rh(bg) = \begin{cases} r(bg), & \text{if } f(bg) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{5.5}$$

To calculate LBGI and HBGI, two sequences of low blood glucose readings LBG and high blood glucose readings HBG are firstly determined by Eq. 5.4 and Eq. 5.5. LBGI is then calculated for the sequence of low blood glucose readings LBG using Eq. 5.6, and HBGI is calculated for the sequence of high blood glucose readings HBG using Eq. 5.7.

$$LBGI(LBG) = \frac{1}{|LBG|} \sum_{i=1}^{|LBG|} LBG_i \tag{5.6}$$

$$HBGI(HBG) = \frac{1}{|HBG|} \sum_{i=1}^{|HBG|} HBG_i \tag{5.7}$$

Finally, the overall BGRI is calculated through the sum of LBGI and HBGI as shown in Eq. 5.8.

$$BGRI = LBGI + HBGI \tag{5.8}$$

Lower values of LBGI and HBGI indicate a lower risk of hypoglycaemic and hyperglycaemic events respectively as illustrated by Table. 5.1. BGRI determines the overall risk of both hypoglycaemic and hyperglycaemic events, with lower values indicating a reduced chance of both hypoglycaemic or hyperglycaemic events occurring.

5.1.2 Time within target blood glucose level range

The percentage of time the subject spends within a pre-defined target range is a good indicator that the solutions provided by the CBR system are safe for the subject. The standard target range for T1DM subjects is between 4 mmol/L and 10.0 mmol/L [Rodbard, 2009; NICE, 2004]. The higher the percentage of time spent within the target range, the better the blood glucose control. However, the implications of extreme highs and lows are not represented by this.

5.1.3 Mean blood glucose level

The mean of the blood glucose data provides a simple approach to determining the subject's overall well-being, but does not aid in representing the variance, extreme lows, and extreme highs. The optimal target mean blood glucose is dependent upon the subject's target blood glucose level, which discounting the 2 hours following a meal is between 4 mmol/L and 7 mmol/L [NICE, 2004].

5.1.4 Variance and standard deviation

Calculating the variance and standard deviation of the continuous blood glucose data allows the overall stability of the subject's blood glucose to be assessed [Rodbard, 2009]. High variance indicates that the subject's blood glucose levels are varying greatly over time. In contrast, lower variance represents improved stability, and if the subject is within the target range, improved blood glucose control. Low variance combined with low or high blood glucose readings would indicate that the subject is consistently outside the safe zone.

5.2 Retrieval test plan

This section describes the testing process used to analyse and evaluate CBR case retrieval. Each stage of the testing process is listed below and visualised in Fig. 5.1. These stages are described in detail in the proceeding subsections.

1. Retrieve configuration to test and compare.
2. Generate test data.
   (a) Generate test case-bases.
   (b) Generate test problem sets.
3. Acquire feature weights through feature selection.
4. Retrieve bolus insulin suggestions.
5. Export and simulate the bolus insulin suggestions.
6. Analyse and evaluate the continuous blood glucose readings.

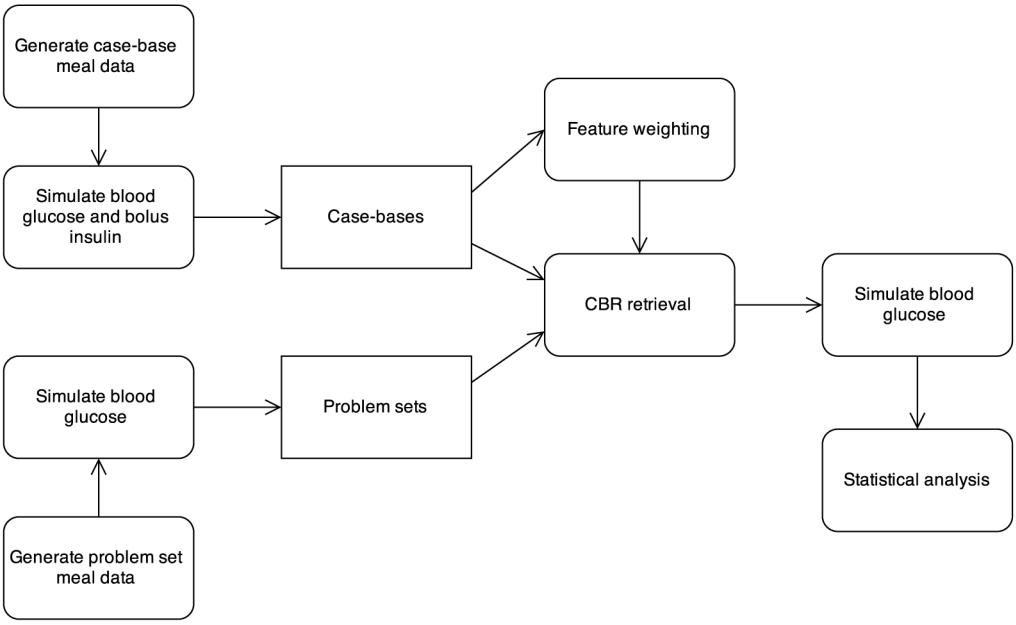


Figure 5.1: Testing process

5.2.1 Retrieval configurations

The first stage of the testing process identifies the different retrieval configurations which will be analysis and evaluated. This includes different distance metrics, k-NN, feature selection algorithms, and temporal sequence lengths. The two different distance metrics (Euclidean and Manhattan) are selected for testing. In addition, six different feature selection algorithms (Chi-Squared, Information Gain, Gain Ratio, One Rule, RELIEF-F, and Symmetrical Uncertainty) have been identified, alongside no feature weighting. The other factors of k-NN and temporal sequence length will be limited to five variations of values $1 \ldots 5$. The list below summarises the retrieval configurations to be tested.

Retrieval configurations

- 2 distance metrics: Euclidean and Manhattan.
- 6 feature selection algorithms for feature weighting: Chi-Squared, Gain Ratio, Info Gain, One Rule, RELIEF-F, and Symmetrical Uncertainty.
- No feature weighting.
- $1 \ldots 5$ nearest neighbours.
- $1 \ldots 5$ temporal sequence length.

In total there are 350 retrieval variations to tested. The goal of this testing is to determine which combination of distance metric, feature weighting, k-NN, and temporal sequence length is the most optimal for case retrieval in this CBR system.

5.2.2 Generation of test data

With the retrieval configurations established, data to test needs to be generated. This involves generating test case-bases and sets of test problems, and also computing the feature weights using the feature selection algorithms.

Case base and new problems

- 5 case-bases, each containing cases over a period of 6 months.
- 5 problem sets, each containing problems over the period of 1 month.

5.2.2.1 Generation of the case-bases

Test runs of the simulator determined that 6 months of meals can be simulated reliably at any one time. As a result it is was decided that the size of the test case-bases would be limited to 6 months. To generate the case-bases, 6 months of meal information are created. These meals are generated randomly with varying time intervals between 120 and 420 minutes, and carbohydrate

intakes of 0 to 240 grams [Roche, 2012]. A rule is imposed on these generated meals to make the earliest meal of the day occur on or after 7 a.m., this is in order to prevent meals occurring when the subject is likely to be asleep. The blood glucose level and bolus insulin dose for each cases are obtained from the simulation of the meal information. These generated meals provide a wide variety of realistic meal patterns over a 6 month period, with each case-base containing an average of 821 cases.

Following simulation of the meal information, the simulated blood glucose and bolus insulin signals require merging with the corresponding meal inputs of carbohydrates and time. These blood glucose and bolus insulin values are extracted from the minute by minute signals exported following simulation using the time the meal occurred during simulation.

5.2.2.2 Generating problem sets

The same process for generating the test case-bases is used to generate test problem sets. The problem sets contain 1 month of randomly generated meals using the same method applied to generating the test case-bases discussed in Section 5.2.2.1. Each problem set is simulated to obtain realistic blood glucose levels for the problem, and the blood glucose signal exported and merged to complete the problems.

5.2.3 Acquiring feature weights through feature selection

Each generated test case-base is run through through the six feature selection algorithms (Chi-Squared, Information Gain, Gain Ratio, One Rule, RELIEF-F, and Symmetrical Uncertainty) in Weka to acquire feature weights for use in the retrieval similarity measure. Prior to importing the case-bases into Weka, the case-bases are formatted to represent temporal sequences of up to 5 in length. Each of the six feature selection algorithms are then performed on the case-bases and the resulting weights extracted. All the test case-bases resulted in similar feature weights, and the mean feature weights from all five test case-bases with a temporal sequence length of 5 are shown in Table 5.2.

The mean feature weights displayed in Table 5.2 show similar results between all the entropy based feature selection algorithms (Information Gain, Gain Ratio, and Symmetrical Uncertainty). This is expected due to Gain Ratio and Symmetrical Uncertainty being evolutions of the Information Gain algorithm. Chi-Squared ranked the features simiarly to the entropy based algorithms but produced a lower weighting for each feature. RELIEF-F appears to be somewhere between the entropy based and Chi-Squared algorithms when looking at the magnitude of the weightings, but does differ in terms of ranking (e.g. $w4$ time has a lower weighting than $w4$ blood glucose). One Rule gave higher weightings to all features when compared to the other algorithms, but ranked the features most similarly to RELIEF-F. Based on these observations we can expect retrieval

using the entropy based and Chi-Squared weightings to result in similar nearest-neighbours being identified. RELIEF-F is likely to be retrieval similar cases to the entropy based algorithms, with potential for better or worse retrieval due to order the algorithm ranked the features. Another expectation is that One Rule will produce different results in retrieval with temporal sequences due to the higher weightings applied to all features, but due to the order the features are ranked, may perform similarly to RELIEF-F.

	Feature	Chi-Sq.	InfoGain	GainRatio	OneR	RELIEF-F	Sym. Uncert.
w_1	Bolus insulin	0.009	0.017	0.016	0.179	0.010	0.017
	Carbohydrates	0.012	0.023	0.018	0.182	0.017	0.021
	Time	0.016	0.024	0.022	0.194	0.008	0.023
	Blood glucose	0.008	0.015	0.017	0.186	0.008	0.016
w_2	Bolus insulin	0.008	0.015	0.016	0.181	0.008	0.015
	Carbohydrates	0.010	0.019	0.017	0.170	0.011	0.018
	Time	0.012	0.025	0.024	0.177	0.013	0.024
	Blood glucose	0.008	0.015	0.018	0.187	0.014	0.015
w_3	Bolus insulin	0.009	0.018	0.018	0.180	0.015	0.018
	Carbohydrates	0.011	0.021	0.018	0.184	0.009	0.019
	Time	0.013	0.027	0.023	0.183	0.016	0.026
	Blood glucose	0.010	0.019	0.020	0.195	0.012	0.019
w_4	Bolus insulin	0.009	0.018	0.019	0.194	0.012	0.018
	Carbohydrates	0.012	0.022	0.019	0.197	0.012	0.020
	Time	0.011	0.024	0.022	0.170	0.009	0.022
	Blood glucose	0.008	0.015	0.016	0.181	0.016	0.016
w_5	Carbohydrates	0.608	0.849	0.727	0.864	0.705	0.783
	Time	0.020	0.028	0.025	0.183	0.011	0.026
	Blood glucose	0.012	0.018	0.020	0.195	0.011	0.018

Table 5.2: Mean normalised feature weights

5.2.4 Retrieving the bolus insulin suggestions

For testing purposes the retrieval process is automated. The system's goal is to perform retrieval for each problem in the problem sets using each of the generated case-bases and retrieval configurations. The process begins by loading a case-base into memory. The system then performs retrieval for each problem in a problem set for a retrieval configuration. This is repeated for each retrieval configuration. Once a problem set has been tested using all retrieval configurations, the next problem set is tested. This process is repeated until all case-bases have been tested against.

5.2.5 Exporting and simulating the bolus insulin suggestions

The bolus insulin suggestions obtained must be run through the simulator in order to analyse and evaluate the retrieval algorithm. To achieve this, the meal information for each problem in the problem set alongside the bolus insulin suggestion must be exported into a scenario file which can be read by the simulator. To automate this process, after each problem set is tested against a test case-base with one retrieval configuration, a scenario file is automatically generated. The generated

scenario file tells the simulator to use open-loop control and contains all the solved problems in the problem set. Each problem in the scenario file is represented by the meal time, carbohydrate intake, and the corresponding bolus insulin dose.

Following simulation, continuous blood glucose readings are exported. This export provides minute-by-minute continuous blood glucose levels during simulation which are used to run statistical analysis upon.

5.2.6 Analysis and evaluation of the continuous blood glucose readings

The continuous blood glucose levels exported following simulation are processed to obtain the LBGI, HBGI, BGRI, time with target range, mean, and variance. Applying these statistical measures provides a method to analyse and evaluate the effect the retrieved bolus insulin suggestions had on the continuous blood glucose level. Through analysis and evaluation of the statistical results an optimal retrieval configuration of those tested will be identified.

The statistical measures are then applied to continuous blood glucose results obtained through closed-loop simulation and the Accu-Chek® Aviva Expert bolus calculator formula on the same problem sets. This comparison provides a benchmark to judge the performance of CBR retrieval against established methods for obtaining bolus advice, and determine if the proposed retrieval strategy is capable of identifying suitable cases.

5.3 Retrieval analysis and evaluation

The aim of the retrieval test plan is to determine how different retrieval configurations will affect the bolus insulin suggestions obtained. This section discusses the statistical results of the simulated continuous blood glucose levels as a result of the suggested bolus insulin suggestions. The results obtained are then compared to those achieved using closed-loop simulation, and the formula derived from the state-of-the-art Accu-Chek® Aviva Expert bolus calculator.

The suggested bolus insulin doses alone do not provide the ability to analyse the performance of the application. Instead the continuous blood glucose levels of the subject are required to assess the performance. The continuous blood glucose levels are obtained through open-loop control simulation. Open-loop control simulation is also applied to the bolus suggestions obtained from the bolus calculator formula.

5.3.1 Analysis of retrieval configurations

The statistical analysis of the retrieval results seeks to establish the optimal retrieval configurations from those explored in this research. The full statistical results are presented in Appendix D, Tables D.1 - D.7, with each table displaying the results for one feature selection algorithm when applied to different distance metrics (Euclidean and Manhattan), 1NN to 5NN, and temporal sequence

lengths of 1 to 5. All the retrieval configurations displayed in these tables present the mean across all problem sets and case-bases for BGRI, LBGI, HBGI, percentage of time below the target range ($<TR$), percentage of time above the target range ($>TR$), variance (σ^2), standard deviation (σ), and mean blood glucose level (μ) in mmol/L.

Initial observations showed some trends in the results. Across all test cases the mean across all statistical measures for the Euclidean and Manhattan distance metrics were almost identical. The Euclidean distance produced marginally better blood glucose control than the Manhattan distance, most notably in configurations where larger temporal sequence lengths are used. Observations on k-NN performance showed a trend of improved retrieval results when factoring in a greater number of nearest neighbours. Another observation is the performance of temporal sequences. Overall, the retrievals with a temporal sequence length of 1 (where no preceding cases are included) resulted in the poorest blood glucose control. This provides some evidence that including preceding events helps to identify better previous cases. A temporal sequence length of 5 demonstrated varied retrieval results depending on k-NN cases retrieved and the feature selection algorithm used. In 1-NN retrievals a larger temporal sequence resulted in the poorest blood glucose control, whilst in 5-NN retrievals a larger temporal sequence resulted in the better blood glucose control over smaller temporal sequences. The BGRI results of the different temporal sequence lengths and k-NN are presented in Fig. 5.2 and 5.3.

Figure 5.2 shows temporal sequence length along the x-axis and BGRI along the y-axis, with each line representing a different k-NN retrieval. The figure highlights that overall 1-NN retrieval resulted in the worst bolus predictions (due to a higher BGRI value) with longer temporal sequence lengths, whilst in $k > 1$-NN retrievals BGRI improved when using longer temporal sequence lengths. In contrast, Fig. 5.2 also shows that 5-NN retrieval performs the worst with temporal sequence lengths of 1, but equal best with 4-NN when using a temporal sequence length of 5.

Figure 5.3 shows the same information as Fig. 5.2, except in this graph the x-axis is k-NN, and lines are the temporal sequence length. The most notable observation from this figure is that in all cases an improvement in BGRI is observed when increasing the temporal sequence length from 1 (no previous cases) to 2 (including the previous case). This provides some evidence that in all NN variations including the previous case in the retrieval process does help to improve the predicted solution. Another observation from Fig. 5.3 is the deterioration or lack of improvement in longer temporal sequence where $k \leq 3$-NN retrieval is used.

The choice of feature selection algorithm used in most instances showed little variation in the retrieval result. This observation is not unexpected as the weights produced by the algorithms were similar with the exception of RELIEF-F and One Rule. Chi-Squared, Information Gain, Gain Ratio and Symmetrically Uncertainty all resulted in similar and blood glucose control all configurations. Overall RELIEF-F resulted in the poorest blood glucose control. One Rule produced interesting results, with the performance dependent upon other configuration factors. In 1-NN configurations,

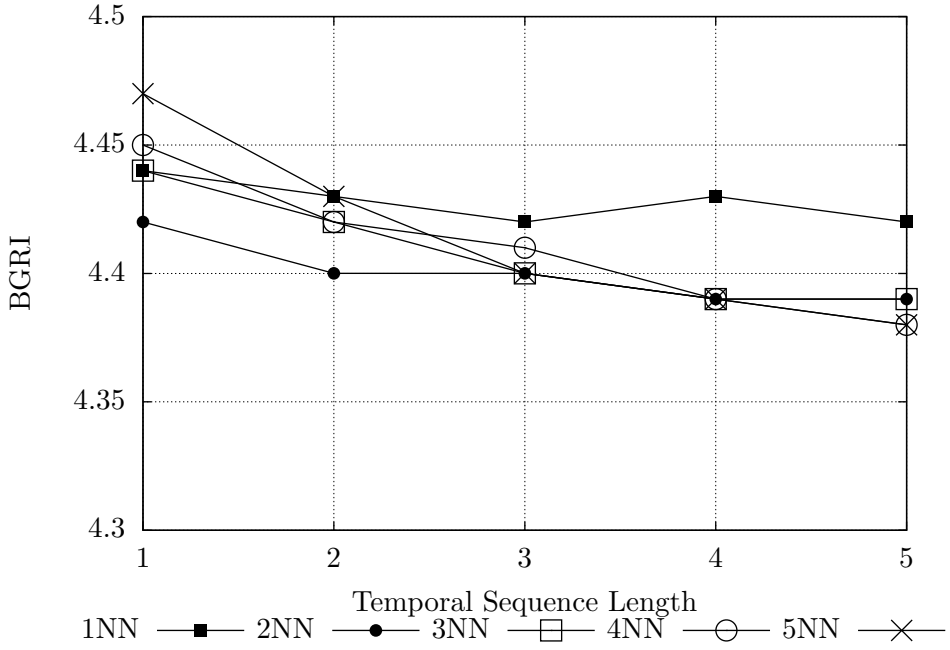


Figure 5.2: k-NN BGRI results for temporal sequence lengths 1 to 5

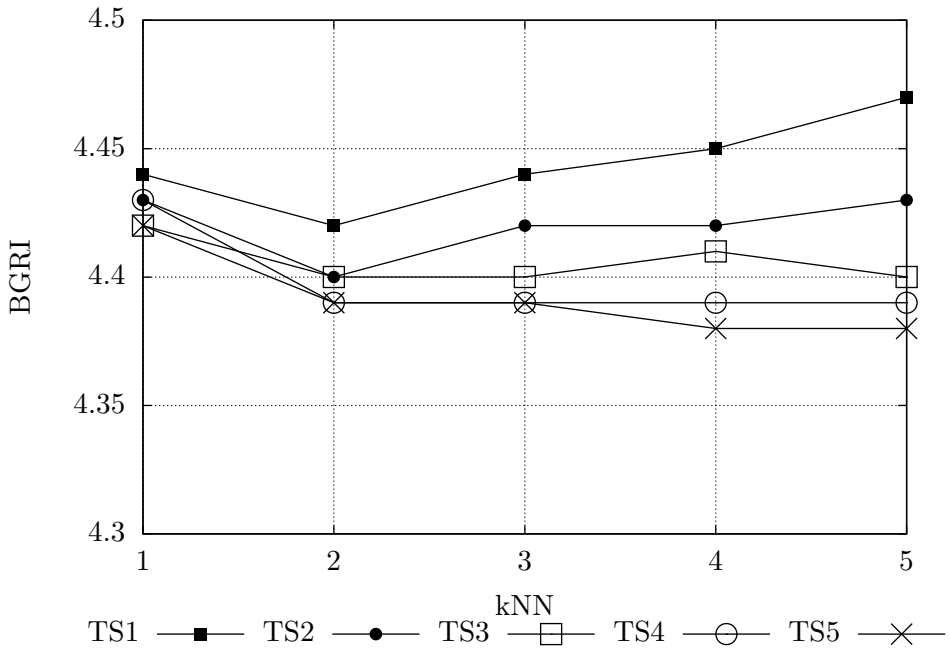


Figure 5.3: Temporal sequence BGRI results for 1-NN to 5-NN

One Rule resulted in the poorest blood glucose control, especially with a larger temporal sequence length. However, in the cases where $k > 1$, One Rule demonstrated the best blood glucose control, particularly for larger temporal sequences.

Appendix D, Table D.7 displays the results for retrieval where no feature weighting is used. The main observation of this data is large variations for different retrieval configurations compared to the consistency of the results where feature weightings are used. In some instances, the results where feature weightings are not used out-performed those with feature weighting. However, due to

the variation observed these positive results may be chance. Some retrieval configurations without feature weighting caused the LBGI to move from the low to medium risk category. Also observed is improved HBGI in comparison to retrieval where feature weightings are used. This is likely to be a result of the high LBGI; as should the LBGI value be high, the HBGI value will be lower. This observation indicates that the bolus suggestions in these instances resulted in larger bolus insulin doses to be suggested than required.

The mean (μ) percentage reduction in BGRI for each of the feature selection algorithms in comparison to no feature weighting are displayed in Fig. 5.4. The figure show the percentage *reduction* in the measure in comparison to when no feature weighting is used. This figure clearly shows that One Rule results in the highest percentage reduction of BGRI compared to no feature weighting, and RELIEF-F resulting in the smallest reduction. In all cases it can be seen that feature weighting benefits the retrieval process, but the selection of the algorithm can result in further improvements. For example, nearly a two-fold improvement is seen when using One Rule in comparison to RELIEF-F. Another interesting observation is the similarity between the Chi-Squared and Information Gain algorithms, despite their different approaches to feature weighting. Additionally it can be observed that the modifications introduced to improve the Information Gain algorithm by Gain Ratio and Symmetrical Uncertainty have resulted in improved BGRI results.

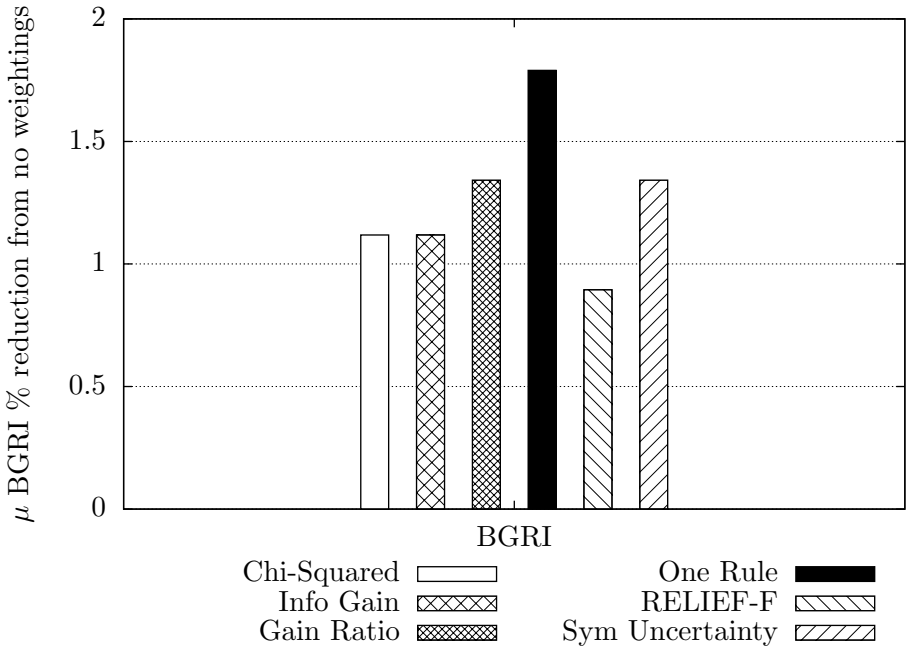


Figure 5.4: Mean BGRI % reduction for the weighting algorithms compared to no weighting

Figure 5.5 breaks down the BGRI results shown in Fig. 5.4 into its separate components of LBGI and HBGI. This figure again shows percentage reduction in comparison to no feature weightings. The most notable observation is the increase in HBGI (due to a negative percentage reduction) for all feature selection algorithms in comparison to no feature weightings. However, it can also be observed that there is a notable reduction in LBGI for all the feature selection algorithms which

significantly outweigh this HBGI percentage increase, as seen in the overall BGRI results in Fig. 5.4.

All feature selection algorithms with the exception of One Rule result in almost identical percentage increases in HBGI of just over 1%. One Rule instead shows a percentage increase of less than 0.5%. Looking at the LBGI statistics, it can be seen that Gain Ratio, One Rule and Symmetrical Uncertainty all result in the similar highest reduction of close to 4%. In contrast RELIEF-F only manages just over a 3% reduction in LBGI.

From the information visualised in Fig. 5.4 and Fig. 5.5 it can clearly be seen that One Rule results in the highest BGRI percentage reduction in comparison to no feature weightings. This is through both a comparably high LBGI reduction coupled with with a minimal increase in HBGI. This result is interesting due to the simplistic nature of the One Rule algorithm over the others used in this research.

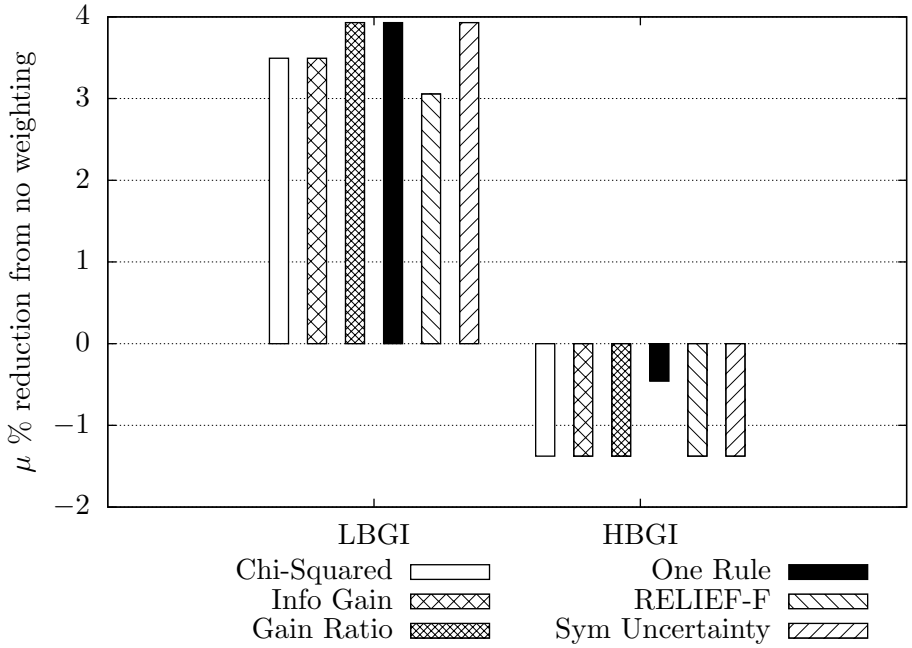


Figure 5.5: Mean LBGI and HBGI % reduction for the weighting algorithms compared to no weighting

The aim of the retrieval analysis and evaluation was to determine which of the retrieval configurations tested provide bolus insulin suggestions resulting in the best blood glucose control. The conclusion reached is that the optimal retrieval configuration of those tested is: One Rule feature selection, Euclidean distance metric, 5-NN, and a temporal sequence length of 5. This conclusion is reached due to the primary statistic for measuring blood glucose control BGRI being the lowest on average when tested with five problem sets on five sample case-bases. This configuration also resulted in the lowest variance of all those tested. However, only marginal differences between the configurations where feature weightings are included were observed.

5.3.2 Comparison to other methods

In this section the continuous blood glucose results from CBR retrieval are compared to bolus suggestions through closed-loop simulation, and the formula presented by the Accu-Chek® Aviva Expert bolus calculator. Both the closed-loop simulation and bolus calculator were applied to the same five problem sets used to evaluate CBR retrieval. The bolus calculator results were computed and then simulated using open-loop simulation to obtain continuous blood glucose levels. The results are compared to the optimal retrieval configuration, and are visualised in Fig. 5.6.

The mean statistical results across all five problems sets using closed-loop simulation can be seen in Table 5.3. As the sample case-bases were created using the simulator it is expected that the CBR results would be similar if the retrieval algorithm can correctly identify similar cases. The BGRI, LBGI and HBGI continuous blood glucose results for the CBR suggestions and closed-loop simulation (Fig. 5.6) yield, as expected, similar results. This provides evidence that the retrieval algorithm is identifying appropriate cases within the case-base.

BGRI	LBGI	HBGI	<TR %	>TR %	σ^2	σ	μ
4.34	2.11	2.2	0.12	0.00	0.79	0.89	6.37

Table 5.3: Closed-loop simulation statistics

Table 5.4 displays mean continuous blood glucose results for the bolus calculator advice. The results show that the bolus calculator outperforms both the simulator and the CBR retrievals. This provides evidence that the formulas used by the state-of-the-art bolus calculators can produce reliable and accurate results. These results provide reassurance, as the mobile prototype (Chapter 6) will build the initial case-base for the subject using an existing formula, since simulation cannot be incorporated into the app.

BGRI	LBGI	HBGI	<TR %	>TR %	σ^2	σ	μ
4.21	1.92	2.29	0.00	0.00	0.76	0.87	6.49

Table 5.4: Bolus calculator statistics

The state-of-the-art bolus calculator results question whether performing CBR retrieval on a case-base produced by the calculation formula will result in similar performance. To test this, the optimal retrieval configuration is tested against case-bases produced by the bolus calculator as opposed to closed-loop simulation. The case-bases were identical to those used previously, but the bolus solutions are replaced by those obtained from the bolus calculator. The results shown in Table 5.5 demonstrate similar CBR retrieval results to those produced by the bolus calculator, with a marginal improvement in some statistical measures. This is visualised in Fig. 5.7 where the original results are shown in addition to CBR retrieval on the bolus calculator case-base.

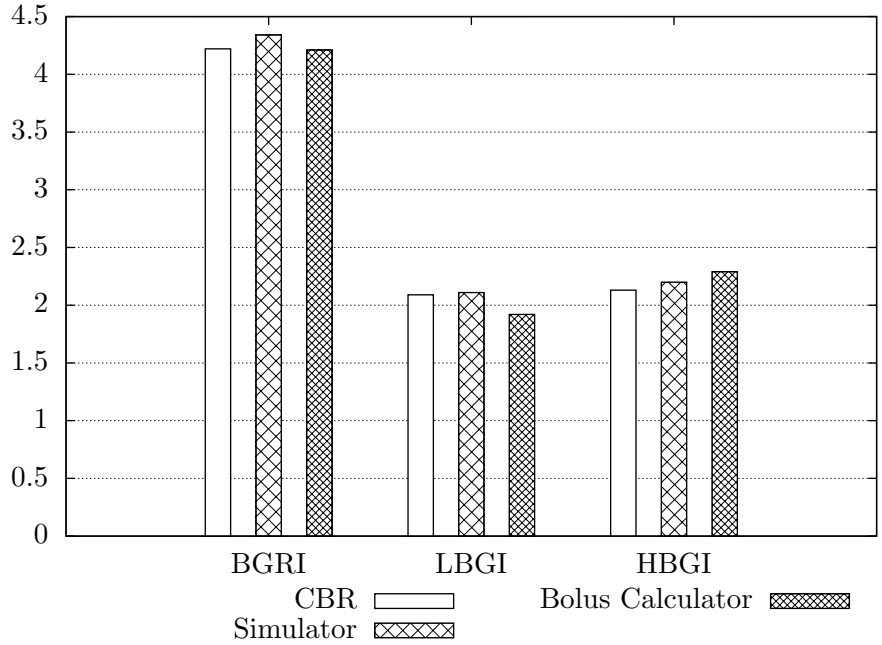


Figure 5.6: Comparison of CBR, simulated and bolus calculator results

BGRI	LBGI	HBGI	<TR %	>TR %	σ^2	σ	μ
4.14	1.87	2.26	0.00	0.00	0.74	0.86	6.48

Table 5.5: CBR retrieval result for case-bases produced by a bolus calculator

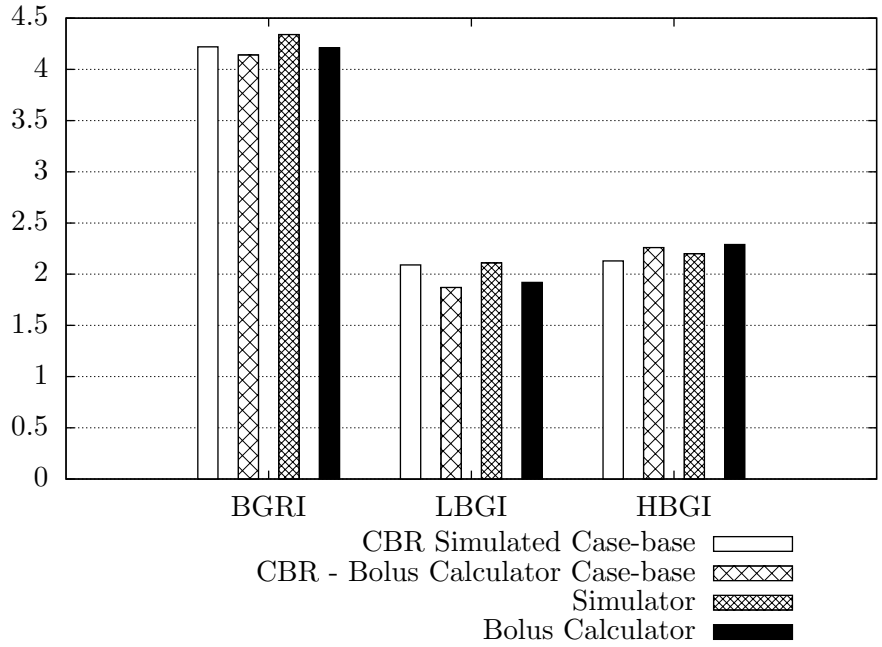


Figure 5.7: Comparison of CBR to other methods including CBR with bolus calculator case-base

The comparison of the CBR retrieval results to the other methods show evidence that the retrieval method discussed is capable of selecting reliable bolus advice when presented with a new problem. The results also show that the quality of the CBR suggestions is related to the quality of the cases retained in the case-base.

5.4 Insulin on board adaptation testing and results

Insulin on board adaptation was tested against a combination of five case-base sets, each representing 6 months of cases; and five sets of problems of to be solved, each containing 1 month of problems. Case retrieval is done using 5-NN weighted Euclidean distance on a temporal sequence length of 5, with feature weights defined using the One Rule algorithm. This configuration is selected as it demonstrated the best blood glucose control in the retrieval analysis and evaluation conducted in Section 5.3.1.

	No IOB	With IOB
BGRI	4.22 ±0.31	3.94 ±0.27
LBGI	2.09 ±0.23	1.96 ±0.23
HBGI	2.13 ±0.16	1.98 ±0.26
$<$ TR %	0.03 ±0.19	0.01 ±0.12
$>$ TR %	0.00 ±0.00	0.00 ±0.00
$\in$ TR %	99.97 ±0.19	99.99 ±0.12
σ^2	0.76 ±0.09	0.66 ±0.06
σ	0.87 ±0.05	0.81 ±0.04
μ mmol/L	6.34 ±0.13	6.30 ±0.21

Table 5.6: Continuous blood glucose statistical results

Table 5.6 shows that on average IOB adaptation results in small decreases to BGRI, LBGI and HBGI against the already good blood glucose control obtained without IOB adaptation. There is also a small increase in the percentage of time blood glucose levels are within target range, alongside a notable reduction in variance. The results for HBGI, LBGI and HBGI are displayed visually in Fig. 5.8.

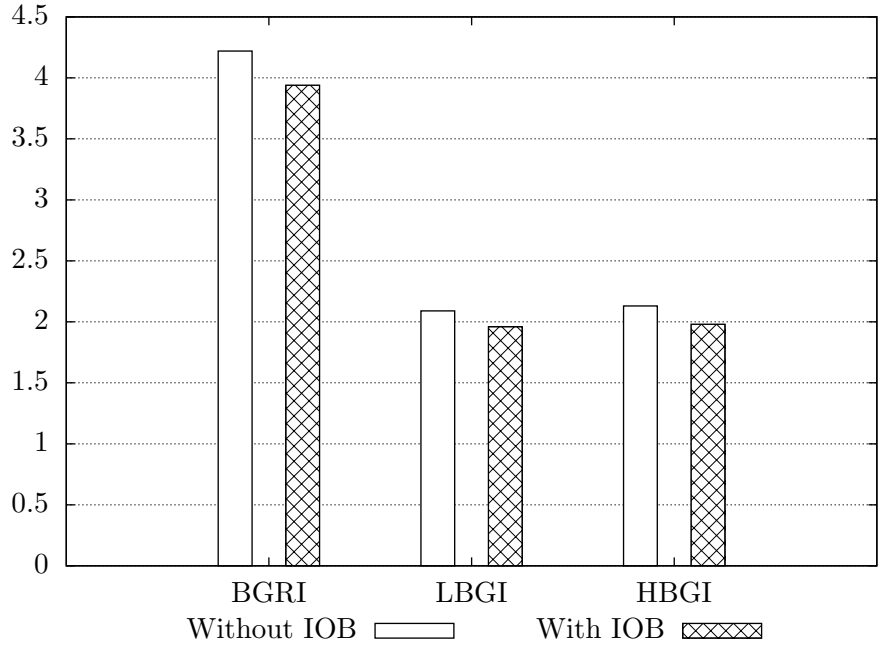


Figure 5.8: Insulin on board comparison

The results show some improvement in overall blood glucose control with IOB adaptation, with a 6.6% reduction in BGRI, in addition to a reduction in variance. The decrease in both LBGI and HBGI suggests that the adaptation rule is able to correctly decrease or increase the bolus insulin suggestion to account for differences between the IOB of the new problem and retrieved case(s). This provides evidence that including this adaptation rule in the reuse step of the CBR system does help to improve the bolus advice. As the results only show a slight improvement from the pre-adaptation solution, it provides an indication that temporal sequences help to reduce the effect of insulin stacking. Since the goal is to seek the optimal solution, any adjustment which can improve the suggestion is beneficial to the system.

5.5 Postprandial evaluation testing and results

To test the effectiveness of the revision rule described in Chapter 4, the bolus insulin solutions following IOB adaptation were subject to five cycles of postprandial evaluation. Three sets of results are recorded for postprandial blood glucose readings taken at 2, 3 and 4 hours after the bolus dose. The test cycles aim to determine the effect on the simulated continuous blood glucose level using same the statistical measures to evaluate the retrieval process. The target blood glucose for the subject is set as 6.66 mmol/L, and TDD used to estimate ISF is calculated using Eq. 4.27 over 4 preceding days, where each day has at least three meals recorded. A daily basal dose of 20 insulin units is added to each of the 4 days, as defined for the subject by the simulator. For instances where the proceeding bolus dose occurs before the postprandial evaluation offset time, the offset is adjusted to 15 minutes prior to the time of the proceeding bolus dose.

Results for five cycles of postprandial blood glucose evaluation and revision using readings 2 hours post meal bolus are displayed in Table 5.7. The results display gradual improvements in all statistical measures for each evaluation cycle with the exception of HBGI. HBGI shows a slight rise following the fourth cycle, which is still well below the minimal risk boundary of less than 5.0. From observing the results of increased offset times (discussed below), the most likely cause for this slight rise in HBGI is due to the short offset time. The mean blood glucose level statistic demonstrates how the revised bolus insulin doses on average result in a continuous blood glucose level closer to the target blood glucose level of 6.66 mmol/L.

Table 5.8 displays postprandial evaluation results for blood glucose readings taken 3 hours post meal bolus. Increasing the postprandial evaluation offset time to 3 hours shows improvements to all statistical measures where an improvement can occur in comparison to the 2-hour offset. Most notably, LBGI is reduced to the minimum risk category (≤ 1.1) after three cycles of evaluation. The increased offset time also removed the rise in HBGI observed in the 2-hour offset.

The final postprandial evaluation uses a 4-hour offset and is presented in Table. 5.9. The results show a slight degradation in blood glucose control in comparison to the 3-hour offset, this

	Original Run	Cycle 1	Cycle 2
BGRI	3.94 ±0.27	3.50 ±0.30	3.29 ±0.32
LBGI	1.96 ±0.23	1.59 ±0.24	1.40 ±0.25
HBGI	1.98 ±0.26	1.91 ±0.31	1.89 ±0.20
$<$ TR %	0.01 ±0.12	0.00 ±0.00	0.00 ±0.00
$>$ TR %	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
σ^2	0.66 ±0.06	0.55 ±0.06	0.51 ±0.06
σ	0.81 ±0.04	0.74 ±0.04	0.71 ±0.04
μ mmol/L	6.30 ±0.21	6.40 ±0.17	6.46 ±0.15

	Cycle 3	Cycle 4	Cycle 5
BGRI	3.21 ±0.32	3.18 ±0.34	3.17 ±0.34
LBGI	1.31 ±0.24	1.27 ±0.23	1.25 ±0.23
HBGI	1.89 ±0.29	1.91 ±0.19	1.92 ±0.20
$<$ TR %	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
$>$ TR %	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
σ^2	0.48 ±0.06	0.48 ±0.06	0.48 ±0.06
σ	0.69 ±0.04	0.69 ±0.04	0.69 ±0.04
μ mmol/L	6.50 ±0.15	6.53 ±0.14	6.54 ±0.14

Table 5.7: Postprandial evaluation 2 hours after meal bolus

	Original Run	Cycle 1	Cycle 2
BGRI	3.94 ±0.27	3.32 ±0.31	3.02 ±0.41
LBGI	1.96 ±0.23	1.41 ±0.17	1.14 ±0.14
HBGI	1.98 ±0.26	1.91 ±0.25	1.88 ±0.28
$<$ TR %	0.01 ±0.12	0.00 ±0.00	0.00 ±0.00
$>$ TR %	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
σ^2	0.66 ±0.06	0.52 ±0.05	0.45 ±0.06
σ	0.81 ±0.04	0.72 ±0.03	0.67 ±0.04
μ mmol/L	6.30 ±0.21	6.44 ±0.15	6.52 ±0.13

	Cycle 3	Cycle 4	Cycle 5
BGRI	2.87 ±0.43	2.82 ±0.44	2.81 ±0.41
LBGI	1.00 ±0.14	0.95 ±0.16	0.94 ±0.15
HBGI	1.87 ±0.29	1.87 ±0.28	1.87 ±0.28
$<$ TR %	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
$>$ TR %	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
σ^2	0.43 ±0.06	0.41 ±0.06	0.41 ±0.05
σ	0.65 ±0.04	0.64 ±0.04	0.64 ±0.04
μ mmol/L	6.56 ±0.12	6.58 ±0.12	6.58 ±0.12

Table 5.8: Postprandial evaluation 3 hours after meal bolus

is likely due to duration of active insulin in simulation.

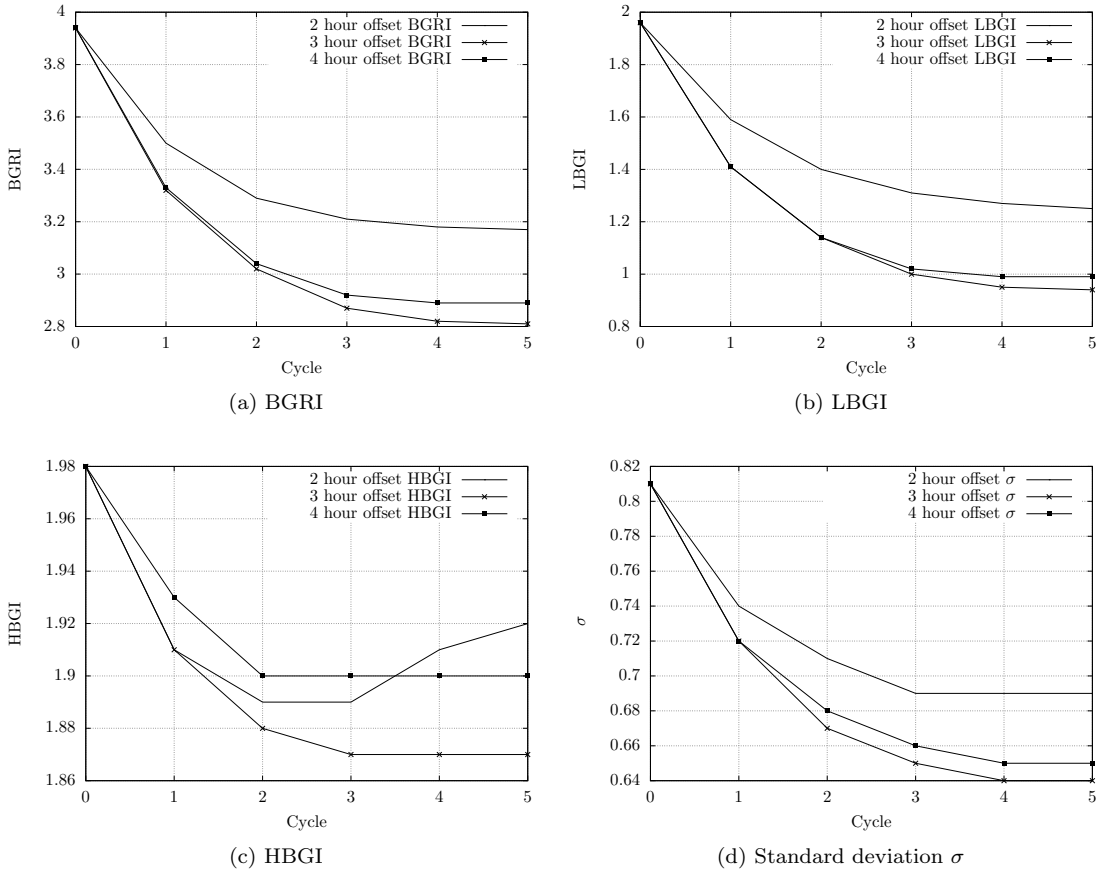
Visualisation of the postprandial evaluation revision results for BGRI, LBGI, HBGI and standard deviation are displayed in Fig. 5.9. The trend graphs illustrate the improvements in blood glucose control with increased learning cycles. Most notably, the 3-hour offset produces improved blood glucose control over both the 2-hour and 4-hour offsets. Cycle 0 in the trend graphs represent the original continuous blood glucose results prior to postprandial evaluation.

The postprandial evaluation results indicate that the postprandial blood glucose offset time is key to the quality of the bolus revision, with an offset time of 3 hours demonstrating the best

	Original Run	Cycle 1	Cycle 2
BGRI	3.94 ±0.27	3.33 ±0.26	3.04 ±0.37
LBGI	1.96 ±0.23	1.41 ±0.19	1.14 ±0.13
HBGI	1.98 ±0.26	1.93 ±0.25	1.90 ±0.27
$<$ TR %	0.01 ±0.12	0.00 ±0.00	0.00 ±0.00
$>$ TR %	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
σ^2	0.66 ±0.06	0.52 ±0.05	0.46 ±0.05
σ	0.81 ±0.04	0.72 ±0.03	0.68 ±0.04
μ mmol/L	6.30 ±0.21	6.43 ±0.16	6.50 ±0.14

	Cycle 3	Cycle 4	Cycle 5
BGRI	2.92 ±0.41	2.89 ±0.40	2.89 ±0.37
LBGI	1.02 ±0.14	0.99 ±0.14	0.99 ±0.14
HBGI	1.90 ±0.27	1.90 ±0.27	1.90 ±0.37
$<$ TR %	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
$>$ TR %	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
σ^2	0.43 ±0.06	0.43 ±0.06	0.43 ±0.05
σ	0.66 ±0.04	0.65 ±0.04	0.65 ±0.04
μ mmol/L	6.53 ±0.13	6.54 ±0.13	6.54 ±0.13

Table 5.9: Postprandial evaluation 4 hours after meal bolus



(a) BGRI (b) LBGI

(c) HBGI (d) Standard deviation σ

Figure 5.9: Postprandial evaluation revision cycle trends

revision results over 2-hour and 4-hour offsets. With a 3-hour offset, a decrease in BGRI of 28.7% is observed after five cycles of revision. This demonstrates the importance of revision to CBRs ability to learn, as shortcomings of the solution can only be observed through simulation or real-world use. The results demonstrate that an automated approach to bolus dose revision is possible

within this CBR system, allowing the user to quickly evaluate the solution and obtain improved bolus insulin suggestions in future reuse.

5.6 Summary

In this chapter the testing methodology of the proposed case-based reasoner for T1DM bolus decision support was discussed, and the subsequent results analysed and evaluated. Prior to devising the test plan, methods to analyse the positive and negative effects of bolus insulin suggestions were identified. This research identified statistical measures for evaluating the continuous blood glucose data, which is an accessible output of the UVa/Padova T1DM simulator. Statistical measures for evaluating the continuous blood glucose data include BGRI, time within a target range, mean, and variance. Blood glucose risk index is broken down into LBGI and HBGI, which can be translated into risk categories, providing a clear method to identify the risk of hypoglycaemic and hyperglycaemic events occurring.

With the identification of appropriate statistical measures to evaluate the retrieved cases, a test plan was devised. The test plan involved the generation of five test case-bases and five problem sets using the simulator. Retrieval results were obtained for each of the problems sets against all the sample case-bases using the different retrieval configurations identified. These retrieval configurations define the distance metric, k-NN, temporal sequence length, and feature selection algorithm for feature weighting. The resulting bolus suggestions from case retrieval were then simulated to obtain continuous blood glucose data.

Analysis of case retrieval found improvements in retrieval results with the use of temporal sequences. In all cases, a temporal sequence length of 2, which includes one preceding case resulted in improved continuous blood glucose results. The success of larger temporal sequences is dependent upon k-NN, with larger temporal sequences degrading the bolus suggestions in 1-NN retrieval, and improving the bolus suggestions in 5-NN retrieval.

Analysis of the feature selection algorithms used for feature weighting identified similar continuous blood glucose results overall for Chi-Squared, Information Gain, Gain Ratio, and Symmetrical Uncertainty. This result is expected due to the similarity of the weights obtained using these algorithms. One Rule was identified as the feature selection algorithm producing the best continuous blood glucose results on average, with RELIEF-F performing the worst. A test run without feature weighting produced inconsistent results, and overall produced poorer continuous blood glucose levels. This highlights the benefit of weighted distance metrics in general and that the feature weighting algorithms improve retrieval results. The two distance metrics evaluated (Euclidean and Manhattan) produced similar results, with Euclidean resulting in the best blood glucose control overall, especially for larger temporal sequence lengths.

The retrieval results were also compared to the continuous blood glucose results of closed-loop

simulation and an existing state-of-the-art bolus calculator formula on the same problem sets. The comparison showed similar results to those obtained by closed-loop simulation, providing evidence the retrieval strategy successfully identified the best case(s) to solve a new problem. The bolus calculator produced marginally superior results to those obtained using CBR retrieval and the simulator. Retrieval was then re-run on a case-base created using the bolus calculator to see if the results would match those produced by the calculator. Analysis of this re-run demonstrated that retrieval on this case-base again produced similar results to those obtained by the bolus calculator. This highlights the importance of the quality of the cases retained by the case-base for retrieving optimal solutions.

The inclusion of IOB adaptation in the reuse stage of the CBR system resulted in an improvement to the bolus insulin suggestions based on analysis of the continuous blood glucose data. Insulin on board is indirectly a feature of the cases when temporal sequences are used as the time between boluses are part of the retrieval algorithm. The adaptation results demonstrate that temporal sequences alone were not sufficient in the prevention of insulin stacking, but through adaptation can be corrected to further improve the bolus suggestion.

Analysis and evaluation of the automated revision algorithm demonstrated significant improvements to all the statistical measures. Observations highlighted that for optimal revision, a post-prandial blood glucose reading should be taken approximately 3 hours after the bolus dose. Using a 3-hour offset, the revision algorithm was able to reduce BGRI by 28.7% after only five revision cycles. These observations demonstrate that with long term use of the system, the automated revision method will allow the CBR system to improve bolus suggestions over time.

The CBR model introduced in Chapter 4 and tested in this chapter will now be implemented as a mobile app. The functional and non-functional requirements of this app refer back to the formal specification and design considerations identified in Chapter 2. The chapter also looks at methods to determine the number of cases required for CBR to useful, and how the CBR system performs on a mobile device.

Chapter 6

Mobile implementation

At this point we have analysed existing bolus calculators to understand the domain and described a CBR model to replace the formulas used by them. This work will now be used in this chapter to design and implement a prototype mobile app for bolus advice using CBR. The implementation is then subject to unit testing, and finally the performance of CBR on a mobile device analysed. The implementation will allow us to evaluate the practicality and usability of the CBR model on a mobile device.

The work carried out on developing the CBR model for bolus advice was conducted using a desktop system programmed in Java, which was chosen with the foresight of a simple transition to an Android device. The mobile prototype requires a method to build a case-base as no reasoning can be conducted without any cases to reason upon. The development and testing of the retrieval algorithm was primarily conducted using a case-base produced by the UVa/Padova T1DM simulator. Whilst this method produced satisfactory cases and would allow a large case-base to be produced quickly, it is not a viable option for the mobile implementation, since the simulator cannot be embedded into the app. Instead the formula derived from the state-of-the-art bolus advisers and formally specified in Chapter 2 will be adopted for this purpose. Evaluation of the retrieval algorithm found that this formula produced marginally superior advice over the simulator, providing confidence that this approach is viable. The drawback is that the user will have to use the app without the aid of CBR until a sufficient number of cases have been retained. Section 6.1 discusses an approach to estimating how many cases are required for CBR to be enabled for bolus advice.

The mobile prototype seeks to fulfil the range of functionality provided by the state-of-the-art bolus advisers discussed in Chapter 2 in addition to the CBR model for bolus advice discussed in Chapters 4 and 5. The implemented prototype must also conform to the safety constraints presented in the formal specification, and address usability issues. These functional non-functional requirements are described in Section 6.2.

The implementation of the requirements is presented as behavioural models in Section 6.3 using use case diagrams, use case descriptions, and activity diagrams from the Unified Modeling Language (UML) [Booch et al., 2004]. These behavioural models describe and visualise the interaction between the users and system.

The implemented prototype is subject to unit testing in Section 6.4 to ensure that operations performed by the system do not violate the imposed constraints, and to ensure the system operations produce the correct outputs. The implementation is also subjected to performance testing in Section 6.6 to assess if CBR is viable for mobile devices from a usability standpoint.

6.1 Cases required for case-based reasoning bolus advice

A method to create the initial case-base through the use of the bolus advice formula has been specified by the formal specification in Chapter 2. Since the goal is to provide advice using CBR, the app must know under which conditions CBR should be enabled. This section looks at an approach to estimating the number of cases required for successful retrieval to occur.

Leake and Wilson [2011] conducted research into case coverage and discuss a method to predict the case-base size required for accurate results to be returned. This research proposed a non-representativeness Monte Carlo method for estimating coverage which demonstrated similar estimations to the representativeness Leave-One-Out method. The Monte Carlo method uses a random uniform sample of the problem space against a case-base. The percentage of cases which solved with this method provide an estimate of the case-base coverage. A case is considered solved if the similarity is above a certain threshold. McSherry [2003] investigated similarity thresholds in CBR, with results demonstrating an almost 100% retrieval accuracy with a 90% threshold, and 80% accuracy with a 80% threshold. As the system is of a safety-critical nature, a provisional high similarity requirement of 90% is imposed.

Research by MacDonald et al. [2008] showed that the number of cases required increases as the number of case features or similarity threshold increases. As a temporal sequence is a single case with an increased number of features and the similarity threshold is high, the success rate of retrieval is likely to be low with a small case-base. To overcome this, the the system will retry the retrieval with the temporal sequence length reduced in length by 1 each time. If no cases are found with the minimum temporal sequence length of 1, the calculator will revert to the bolus advice formula and inform the user that the result is not obtained using CBR.

The Monte Carlo method proposed by Leake and Wilson [2011] was applied to sample case-bases from 10 to 200 in size, with a uniform problem space sample of 4,600. The results of the Monte Carlo coverage estimation (see Fig. 6.1) found a rapid improvement in successful retrievals up to a case-base size of 35. At 35 cases there is a 55% probability of retrieving at least one case with greater than or equal to 90% similarity. The probability of retrieving cases over 90%

similarity then begins to plateau with a gradual increase to 65% with a case-base of size of 200. These results are consistent with those presented by Leake and Wilson [2011], where there is a rapid increase in prediction accuracy that quickly plateaus into a gradual improvement.

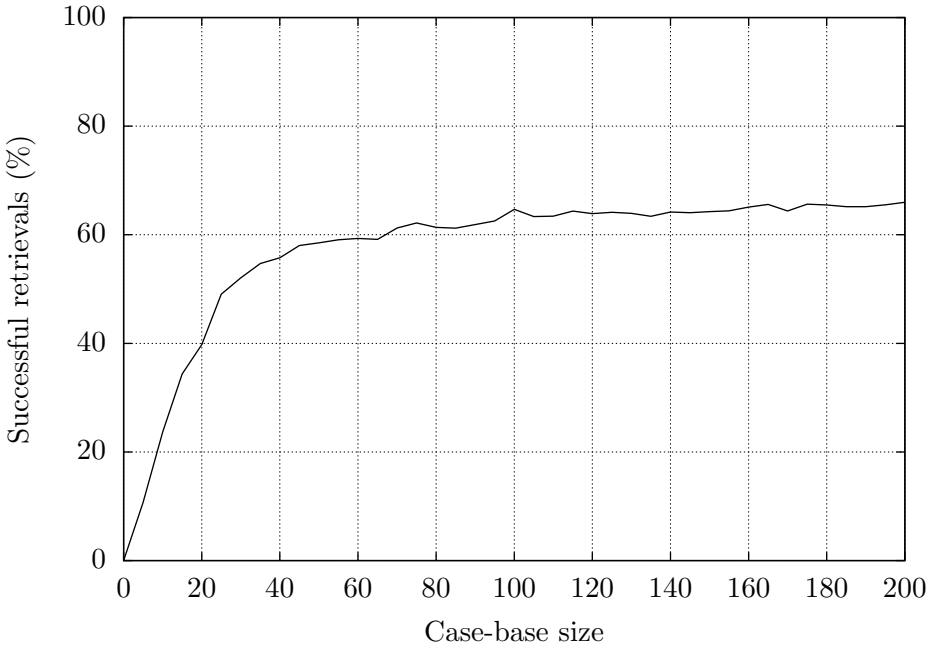


Figure 6.1: Monte Carlo successful retrievals for a 90% similarity threshold

In real-world use, the distribution of problems and the case-base is unlikely to cover the whole problem space, and instead may be limited to certain areas of feature space, especially if the user has a regular daily pattern. In these scenarios, the probability of retrieving cases above 90% similarity is likely greater than the estimate.

For a user recording four new cases a day, the app would require 9 days of continuous use to retain the 35 cases to reach the beginning of the plateau identified in the Monte Carlo coverage estimation. Unless the case-base has 100% coverage of the problem space there is still a possibility for cases below the 90% similarity threshold to be retrieved. To account for this, the app will revert to the bolus calculator formula should no cases above 90% similarity be retrieved. With this additional safety mechanism, bolus advice through CBR is only used when a suitable case is retrieved.

6.2 Design specification

Prior to the implementation of the mobile prototype, it is important to outline the functional and non-functional requirements of the system. The formal specification in Chapter 2 focused on the non-functional requirements of the system in regards to the constraints on system variables. The functional requirements instead provide a list of required functionality for the mobile prototype.

6.2.1 Functional requirements

The functional requirements are derived from the functionality presented by the state-of-the-art bolus calculators in addition to the system requirements to perform CBR for bolus insulin advice. In total, 13 functional requirements for the app were identified and are described below (FR-1 - FR-13). Each of the functional requirements describes the general functionality, inputs, outputs and storage required to fulfil the specified functionality.

FR-1	
Function	The user is able to define and save their subject specific settings.
Inputs	Blood glucose unit (mmol/L or mg/dL). Upper target blood glucose range (mmol/L or mg/dL). Lower target blood glucose range (mmol/L or mg/dL). High blood glucose warning level (mmol/L or mg/dL). Low blood glucose warning level (mmol/L or mg/dL). Hyperglycaemia blood glucose warning level (mmol/L or mg/dL). Hypoglycaemia blood glucose warning level (mmol/L or mg/dL). Blood glucose units (mmol/L or mg/dL) reduced by one insulin unit. Carbohydrates (grams) covered by one insulin unit. Active insulin time (minutes). Daily bolus insulin dose (insulin units). Maximum suggested bolus insulin dose (insulin units).
Outputs	Confirmation when the settings are saved.
Storage	Store all the user defined settings on the local device.

FR-2	
Function	The user is able to reset the app to the default settings.
Outputs	Confirmation that the settings are reset.
Storage	Delete the user's defined settings on the local device.

FR-3	
Function	The user is suggested bolus insulin advice based on previous events using CBR.
Inputs	Planned carbohydrate intake (grams) Preprandial blood glucose reading (mmol/L or mg/dL). Date and time. Active insulin time (minutes).
Outputs	Active insulin (insulin units). Suggested bolus insulin (insulin units), rounded to the nearest half unit.
Storage	Required inputs stored as settings are accessed from the local device.

FR-4	
Function	Incorporate a bolus calculator to suggest bolus insulin advice and create a sufficient number of cases for CBR to be used.
Inputs	Planned carbohydrate intake (grams). Preprandial blood glucose reading (mmol/L or mg/dL). Date and time. Upper target blood glucose range (mmol/L or mg/dL). Lower target blood glucose range (mmol/L or mg/dL). Blood glucose units (mmol/L or mg/dL) reduced by one insulin unit. Carbohydrates (grams) covered by one insulin unit. Active insulin time (minutes).
Outputs	Correction dose (insulin units). Meal dose (insulin units). Active insulin (insulin units). Suggested bolus insulin (insulin units), rounded to the nearest half unit.
Storage	Required inputs stored as settings are accessed from the local device.

FR-5	
Function	The user may override the suggested bolus insulin advice or cancel the process in the event they choose not to use the advice presented.
Inputs	Bolus insulin dose (insulin units).
Outputs	Updated suggested bolus insulin dose (insulin units).

FR-6	
Function	The user is prompted to perform a postprandial blood glucose reading through a notification. The time of the reminder is specified as an offset from the current time.
Inputs	Time offset (minutes) for the reminder to displayed.
Outputs	Confirmation that the reminder has been set. Display a reminder at the specified time.
Storage	Store the reminder on the local device.

FR-7	
Function	The user is able to record their postprandial blood glucose reading during the evaluation process.
Inputs	Postprandial blood glucose reading (mmol/L or mg/dL). Unevaluated case details.
Outputs	High or low blood glucose level warning if the postprandial blood glucose level is above or below the high or low blood glucose defined in the user settings. Hyperglycaemia or hypoglycaemia warning if the postprandial blood glucose level is above or below the high or low blood glucose defined threshold in the user's settings.
Storage	Access the unevaluated case from the local device. Update the unevaluated case on the local device to store the postprandial blood glucose reading.

FR-8	
Function	The user is able to perform manual revision of the bolus insulin dose for the unevaluated case.
Inputs	Revised bolus insulin dose (insulin units). Unevaluated case details.
Outputs	Revised bolus insulin dose (insulin units). Difference between the used bolus insulin dose and revised bolus insulin dose.
Storage	Access the unevaluated case from the local device.

FR-9	
Function	The user is able to perform automatic revision of the bolus insulin dose for the unevaluated case based upon the postprandial blood glucose reading.
Inputs	Unevaluated case details. Postprandial blood glucose reading (mmol/L or mg/dL). Previous cases retained in the case-base. Blood glucose units (mmol/L or mg/dL) reduced by one insulin unit. Daily bolus insulin dose (insulin units).
Outputs	Revised bolus insulin dose (insulin units). Difference between the used bolus insulin dose and revised bolus insulin dose.
Storage	Access the unevaluated case from the local device. Access the user's defined settings from the local device.

FR-10	
Function	Following successful evaluation of the proposed solution, the case is retained in the case-base for logging, visualisation and future CBR reuse.
Inputs	Unevaluated case details. Postprandial blood glucose reading (mmol/L or mg/dL). Revised bolus insulin dose (insulin units).
Outputs	Confirmation the case has been retained.
Storage	Access the unevaluated case from the local device. Update the case as evaluated and store the revised bolus insulin dose in the case-base on the local device.

FR-11	
Function	The user is able to view past cases which have been evaluated and retained in the case-base.
Outputs	A list of retained cases which have been evaluated by the user.
Storage	Access the case-base containing evaluated cases on the local device.

FR-12	
Function	The user is able to delete a past case from the case-base.
Outputs	Confirmation the case has been deleted.
Storage	Delete the selected case from the case-base on the local device.

FR-13	
Function	The user is able to visualise trend graphs for blood glucose readings, carbohydrate intake, and administered bolus insulin doses over a selected period of time.
Inputs	Trend graph to display: blood glucose readings, carbohydrate intake and administered bolus insulin doses. Number of preceding days to display.
Outputs	Trend graph of the selected data over the selected period of time.
Storage	Access the case-base containing evaluated cases on the local device.

6.2.2 Non-functional requirements

The non-functional requirements of the app describe the constraints imposed on inputs for safety, and also the usability and quality factors required of the prototype.

With a app for use in the medical domain, it is crucial that it is created with safety at the forefront. The safety constraints required of the system were derived from the Accu-Chek® Aviva Expert, an FDA-approved blood glucose meter which also provides bolus advice. The Accu-Chek®

Aviva Expert provides a detailed list of lower and upper boundaries for various setting variables and are described in Table. 6.1.

The formal specification in Chapter 2 validated that the guards required for actions which alter the systems state or perform bolus advice using the specified formula do not violate the invariants of the system. Inputs required for the CBR adviser will also be subjected to the same guards and outputs limited by the same constraints, such as the maximum allowed bolus dose. The safety and usability requirements of the system are described by non-functional requirements (NFRs) 1 through to 5.

NFR-1
All user inputs must be validated to ensure they are within the permitted range or listed allowed values in the case of string selection. Valid ranges and values are displayed in Table 6.1 and Table 6.2.

NFR-2
All blood glucose data will be stored and calculated in mmol/L units. Values will only be converted to mg/dL for users with this setting when display of the value is required.

NFR-3
The user will not be able to obtain bolus advice until the app settings are configured and saved.

NFR-4
Bolus advice through CBR is only displayed if the similarity is over 90% to the retrieved case(s). The bolus calculator formula will be used if no similar cases are retrieved.

NFR-5
All actions will be displayed within an acceptable response time. Nielsen [1993] describes 10,000 ms as a maximum tolerable response time for keeping the user's attention.

Data type	Min	Max
Acting time (minutes)	90	480
Active insulin offset time (minutes)	45	Acting time
Basal insulin (IU)	0	99
Maximum bolus dose (IU)	0	50
Calculated bolus dose (IU)	0	50
Blood glucose (mmol/L) reduced by 1 insulin unit	0.1	55.4
Carbohydrates (grams) covered by 1 insulin unit	1	240
Target blood glucose upper range value (mmol/L)	5.5	15
Target blood glucose lower range value (mmol/L)	3	8
High blood glucose threshold (mmol/L)	6.5	19.5
Low blood glucose threshold (mmol/L)	3	5.5
Hyperglycaemia limit (mmol/L)	10	19.5
Hypoglycaemia limit (mmol/L)	3	5

Table 6.1: Input and output inclusive range constraints

Data type	Allowed values
Blood glucose unit	mmol/L, mg/dL

Table 6.2: Input and output allowed value constraints

6.3 Behavioural modelling

Behavioural modelling is used to describe the dynamic interactions between objects [Booch et al., 2004]. In this section, the interaction between the actors and the system are described using UML use case diagrams, use case descriptions and activity diagrams.

The use case diagram presented in Fig. 6.2 describes the relationship between actors and seven use cases: edit setting, load default settings, save settings, obtain bolus advice, postprandial bolus advice evaluation, view past case, and view trend graph. Two types of actor are modelled in the use case: the patient, and the doctor or carer. The patient is able to perform all interactions with the system, whilst the doctor or carer may wish to view or visualise past events to help manage the patient's condition.

The use cases presented describe the behaviour required by the system to meet all the functional requirements (Table 6.3) of the system outlined in Section 6.2.1. Each use case diagram is accompanied by a use case description and activity diagram to capture and visualise the desired behaviour in an understandable way.

Use case	Functional requirements
Edit setting	FR-1
Load default settings	FR-2
Save settings	FR-1
Obtain bolus advice	FR-3, FR-4, FR-5, FR-6
Postprandial bolus advice evaluation	FR-7, FR-8, FR-9, FR-10
View past case	FR-11, FR-12
View trend graph	FR-13

Table 6.3: Use case relationship to the functional requirements

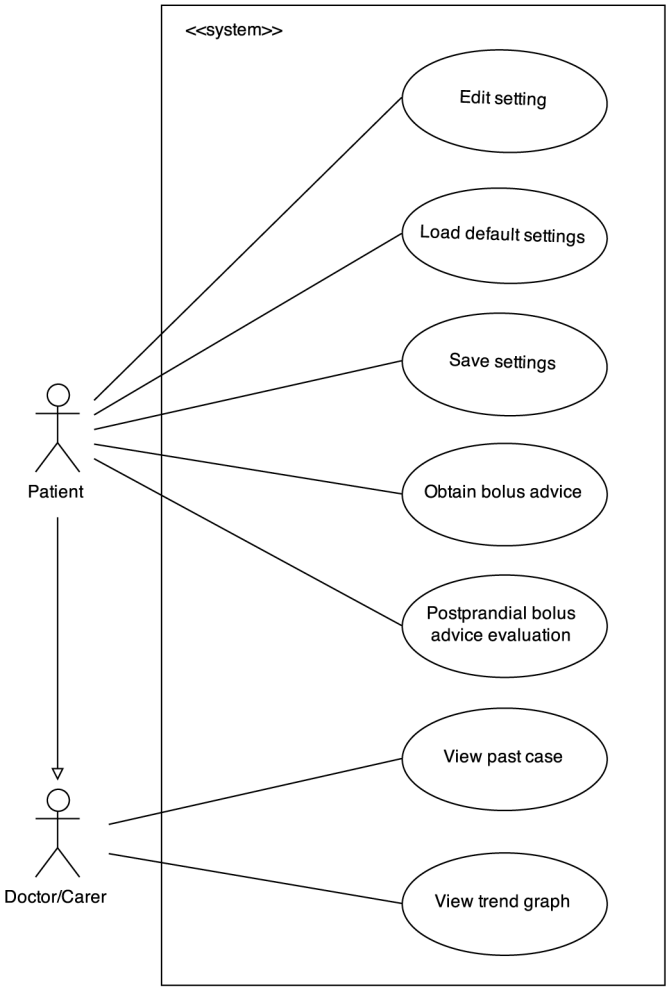


Figure 6.2: Use case diagram

6.3.1 Edit setting

Use case description

The *edit setting* use case describes the behaviour of the user changing an individual setting to satisfy FR-1. The user will select a field on the settings screen to change. Once the value is changed, the input will be validated by the system in accordance with NFR-1. If the input is valid then the value will be updated, otherwise the user will be informed the value is invalid and prompted to input the value again. The behaviour is visualised by the activity diagram Fig. 6.3a.

Actors

Patient.

Main success scenario

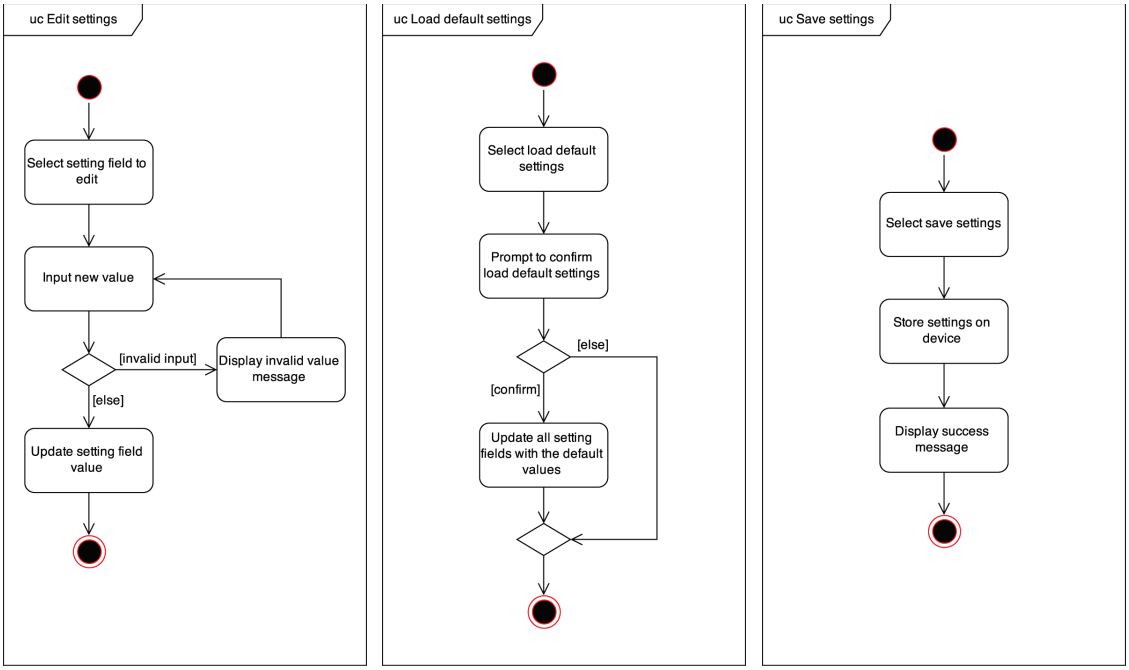
1. User selects a setting field to edit.
2. User inputs a new value.
3. System successfully validates the user's input.
4. System updates the setting field value.

5. System displays the settings screen.

Extensions

3a: The user's input is invalid.

.1: System displays an invalid value message, returns to main success scenario (MSS) step 2.



(a) Edit setting (b) Load default settings (c) Save settings

Figure 6.3: Edit setting, load default settings and save settings activity diagrams

6.3.2 Load default settings

Use case description

The *load default settings* use case describes the behaviour of the user loading the systems default settings to satisfy FR-2. The user will initiate the use case by selecting the option to load the default settings from the settings screen. The user will then be prompted to confirm if they want to load the default settings. If the user confirms the decision then the default settings are loaded and settings screen updated, otherwise no changes are made. The behaviour is visualised by the activity diagram Fig. 6.3b.

Actors

Patient.

Precondition

All values defined by the default settings are valid.

Main success scenario

1. User selects load default settings.
2. System prompts user to confirm loading of the default settings.
3. User confirms load default settings.
4. System updates all setting fields with the default values.
5. System displays the settings screen.

Extensions

3a: The user cancels the loading of the default settings when prompted.

.1: System returns to MSS step 5.

6.3.3 Save settings

Use case description

The *save settings* use case describes the behaviour of the user saving their settings to satisfy FR-1. Saving the settings is a precondition for the obtain bolus, postprandial blood glucose reading evaluation, view past cases, and view trend graph use cases. To initialise the save settings use case the user will select the save settings option on the settings screen. The settings will then be stored on the device by the system and inform the user that the settings have successfully been saved. The behaviour is visualised in activity diagram Fig. 6.3c.

Actors

Patient.

Main success scenario

1. User selects save settings.
2. System stores the settings on the device.
3. System displays save successful message.
4. System displays the settings screen.

6.3.4 Obtain bolus advice

Use case description

The *obtain bolus advice* use case describes the behaviour of the user inputting information about their current situation and the system returning bolus advice; covering FR-3, FR-4, FR-5, and FR-6. The use case is initiated when the user selects the bolus calculator option from the menu or main screen. The user will then input the required details for obtaining bolus advice. At this point, the user will choose between using CBR or the bolus formula for advice. If there are sufficient cases in the case-base, CBR is selected by default. If CBR is selected, the system will ensure that reused cases are above the similarity threshold. If this threshold is not met or if the

user selects to use the bolus formula, then the system will use the bolus formula to calculate the bolus advice. Once the system obtains the the bolus advice it is displayed to the user. The user may now discard, override or accept the advice. Overriding the advice will allow the user to input a manual bolus dose. Once the advice is accepted, the user will be prompted to input a time offset for displaying the postprandial blood glucose reminder. When the reminder is set, the system will store the unevaluated case and return to the main screen. The behaviour is visualised by the activity diagram in Fig. 6.4.

Actors

Patient.

Precondition

The user settings are defined and saved.

Main success scenario

1. User inputs carbohydrate, current blood glucose level, and if advice should be obtained CBR or using the bolus formula.
2. System uses CBR to obtain bolus advice.
3. System displays bolus advice.
4. User accepts bolus advice.
5. User sets postprandial blood glucose reading reminder offset time.
6. System stores the unevaluated case.
7. System returns to main screen.

Extensions

2a: Insufficient cases in the case-base or no similar cases retrieved by CBR.

.1: System uses bolus formula to obtain bolus advice instead of CBR, returns to MSS step 3.

4a: User selects override bolus advice.

.1: User manually overrides the bolus advice and returns to MSS step 3.

4b: User selects discard bolus advice.

.1: System cancels the process and returns to MSS step 7.

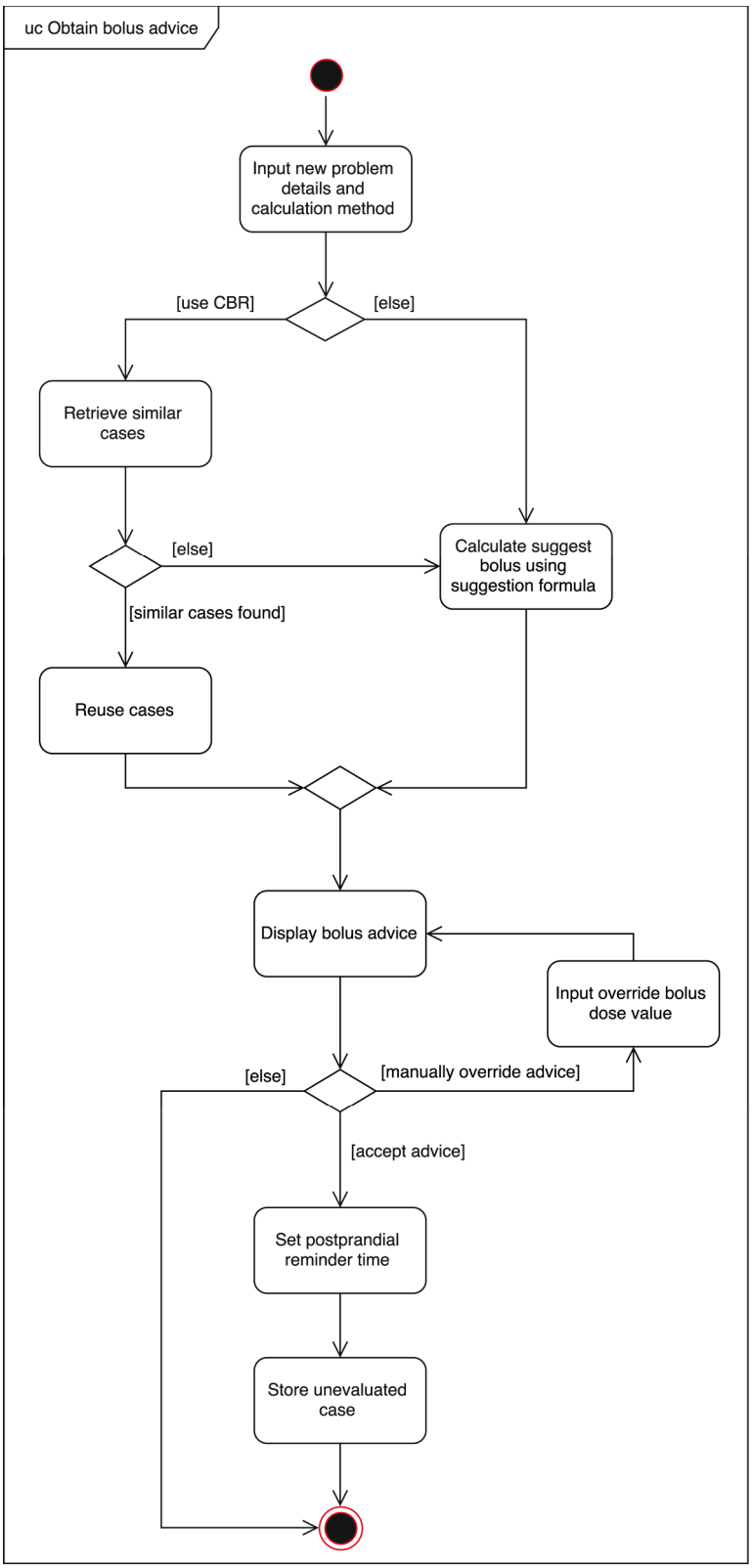


Figure 6.4: Obtain bolus advice activity diagram

6.3.5 Postprandial bolus advice evaluation

Use case description

The *postprandial bolus advice evaluation* use case describes the behaviour of the user recording their postprandial blood glucose level and evaluating the associated bolus advice; covering FR-7,

FR-8, FR-9, and FR-10. This use case occurs a period of time after the obtain bolus advice use case. The use case is initiated either through the reminder dialogue (set in the obtain bolus use case) or through the pending evaluation screen. Upon initialisation the user is requested to input their postprandial blood glucose reading. Once the reading is recorded, the user can select between automated or manual revision of the bolus advice. Automated revision is completed by the system using the user's settings and the postprandial blood glucose reading. Manual revision involves the user inputting amended bolus advice based upon their own evaluation of the advice. After revision is completed, the user is presented with the updated information. The user must then select whether to accept or decline the evaluated advice. If the user accepts the revision, the new case is retained in the case-base by the system. The process is visualised in activity diagram Fig. 6.5.

Actors

Patient.

Preconditions

1. The user settings are defined and saved.
2. Bolus advice has been obtained and stored in an unevaluated case.

Main success scenario

1. User loads an unevaluated case.
2. User inputs postprandial blood glucose reading.
3. User selects automated evaluation method.
4. System displays revised bolus advice.
5. User accepts bolus advice revision.
6. System retains the new case in the case-base.
7. System returns to the main screen.

Extensions

3a: User selects manual evaluation.

.1: User inputs revised bolus dose, returns to MSS step 4.

5a: User selects cancel revision.

.1: System cancels the process, returns to MSS step 7.

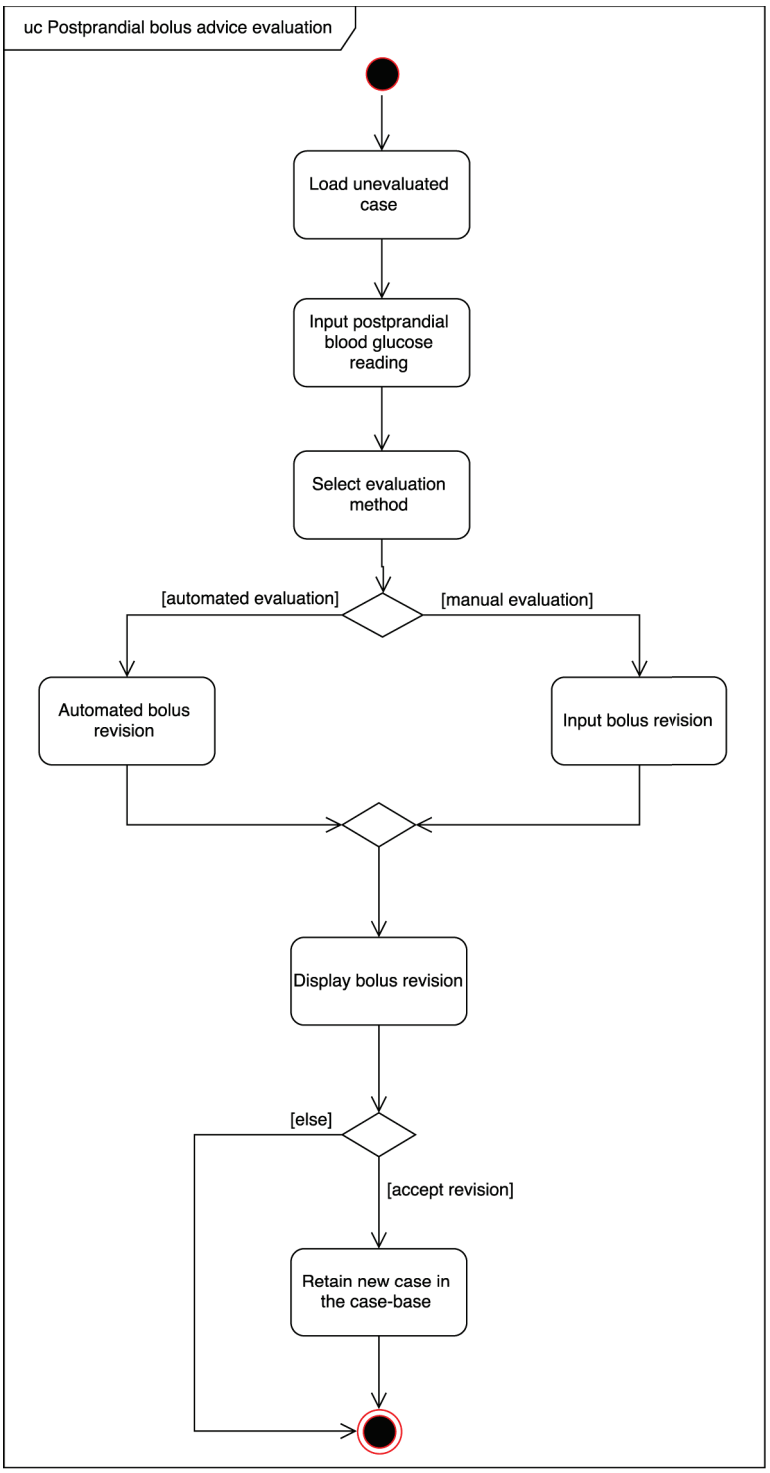


Figure 6.5: Postprandial bolus advice evaluation activity diagram

6.3.6 View past case

Use case description

The *view past case* use case describes the behaviour of the user viewing the details of a previously evaluated case retained in the case-base to satisfy FR-11 and FR-12. The use case is initiated from the menu or main screen. The user will then be presented with a list of all cases retained in the case-base. From this list of cases, the user is able to select a case to view. Viewing the case

will display all the case details to the user and provide an option to delete the case if desired. If the user chooses to delete the case, the system will prompt the user to confirm. Confirmation of deletion will remove the case from the case-base and the user is returned to the list of past cases. The process is visualised in activity diagram Fig. 6.6a.

Actors

Patient, Doctor/Carer.

Preconditions

1. The user settings are defined and saved.
2. At least one previous case exists in the case-base.

Main success scenario

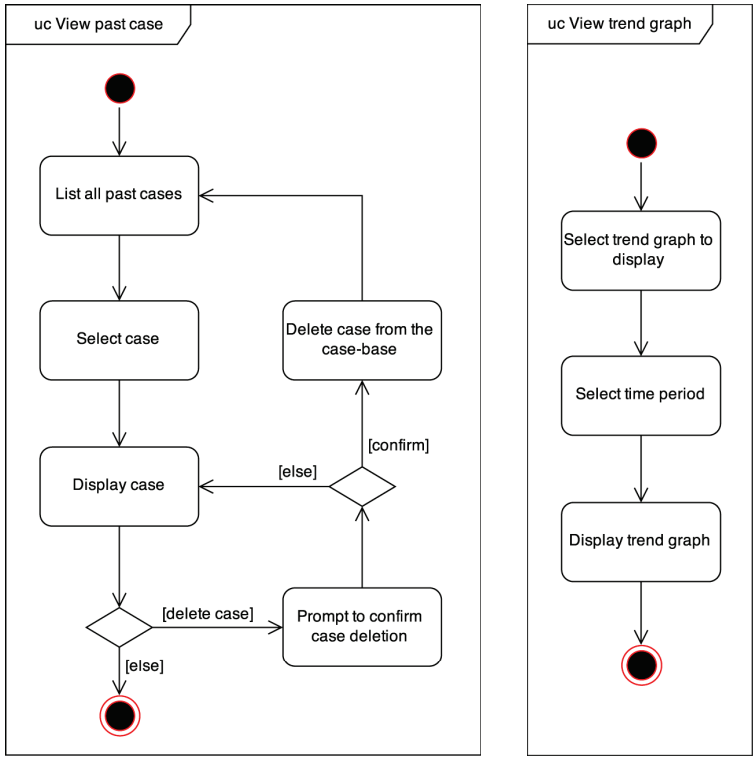
1. System lists all past cases.
2. User selects a case to view.
3. System displays the past case.
4. System returns to the main screen.

Extensions

3a: User selects delete past case.

.1: System prompts for confirmation to delete the past case.

.2: If the user confirms the deletion, the past case is removed from the case-base and returns to MSS 1, otherwise user does not confirm and returns to MSS 3.



(a) View past case (b) View trend graph

Figure 6.6: View past case and view trend graph activity diagrams

6.3.7 View trend graph

Use case description

The *view trend graph* use case describes the behaviour of the user viewing recorded data to visualise trends to satisfy FR-13. The use case is initialised from the menu or main screen and presents the user with different feature trends to show. Upon selecting the feature to visualise, the user is prompted to select the time period for which the trend graph will display records. Selection of the time period will result in the system displaying the trend graph to the user. The process is visualised in activity diagram Fig. 6.6b.

Actors

Patient, Doctor/Carer.

Preconditions

1. The user settings are defined and saved.
2. At least one previous case exists in the case-base.

Main success scenario

1. User selects trend graph to display (blood glucose, carbohydrate intake, or bolus taken).
2. User selects the trend graph time period.
3. System displays the trend graph.
4. System returns to the main screen.

Extensions

1a: No past cases exist.

.1: System displays notification that there are no past cases, returns to MSS 4.

6.4 Unit testing

Correct implementation is fundamental to the safety of the app. It is important that all inputs are successfully validated, and that all actions performed by the app produce the correct output.

The use of a formal specification provides confidence that the guards present in the system prevent invalid actions from occurring. It is always possible that a mistake can be made during implementation that would cause the system to either violate the constraints or produce incorrect results. In addition, certain aspects of the system were not formally specified, such as the CBR model. To identify possible errors in implementation, unit testing is used to test critical methods in the system. Each method is passed multiple valid and invalid inputs where possible to check if the output is as expected.

Unit testing of the methods which modify the system settings and user inputs are achieved using an automated approach. To perform the unit testing, random valid and invalid inputs are

passed into the method. The invalid inputs include boundary values and just outside boundary values (±1 in the case of integer inputs, and ±0.1 for double inputs). The expected output for the input is then compared to the actual output. If the expected output matches the actual output, the method passes the unit test.

This unit testing is performed on all methods required to adjust the app settings, calculation outputs, retrieval similarity, IOB adjustment, and postprandial evaluation and revision. All the unit tests conducted passed without the need for extensive debugging, which provides evidence that the implementation correctly reflects the constraints and guards defined by the formal specification. Appendix E displays the results of the unit tests for a combination of valid and invalid values.

6.5 System demonstration

In this section two brief case studies of the mobile implementation are demonstrated. The case studies selected for demonstration will follow the main success scenarios of the *obtain bolus advice* (Section 6.3.4) and *postprandial bolus advice evaluation* (Section 6.3.5) use cases. These use cases have been selected as they demonstrate the case-based reasoning system on the mobile advice, starting with the input of a new problem through to the retaining of a new case.

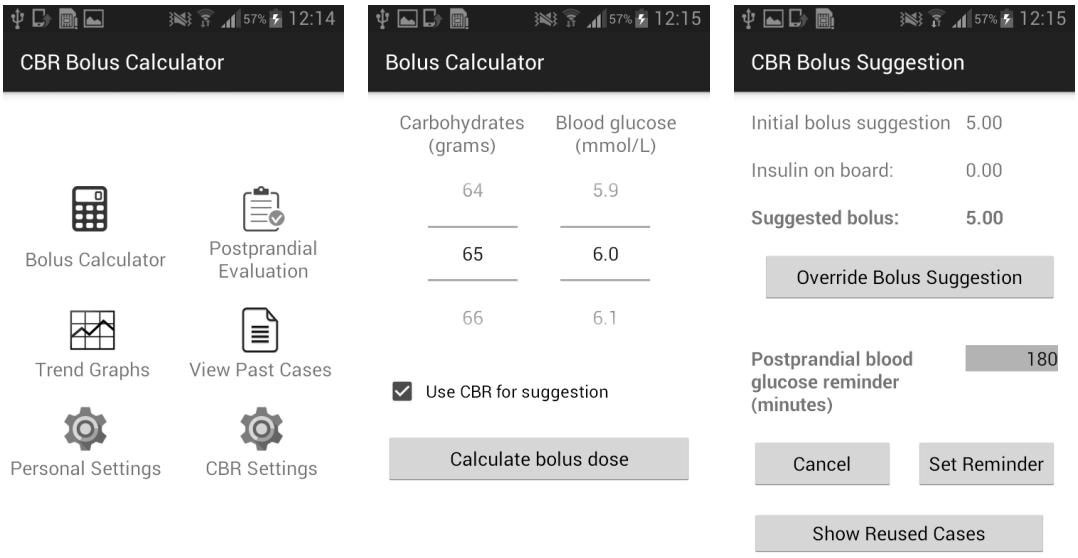
The system settings have been defined prior to this example, as required by the precondition of the *obtain bolus advice* use case. Additionally the application has been pre-populated with a set of pseudo cases, sufficient to allow CBR retrieval to be enabled.

Obtain bolus advice use case demonstration

The *obtain bolus advice* use case begins by tapping *Bolus Calculator* on the app's main screen (Fig. 6.7a). The user is then presented with a screen to input their new problem description (Fig. 6.7b), which requires selecting the quantity of carbohydrates in grams and current blood glucose level (in this example the mmol/L unit is used) from the number pickers or by typing in the values. In this example 65 grams of carbohydrates and a blood glucose reading of 6.0 mmol/L are input. The *Use CBR for suggestion* checkbox is selected by default, since the system has sufficient cases to use CBR. With the values input, the user can proceed by tapping the *Calculate bolus dose* button.

The user is then presented with the bolus insulin suggestion retrieved by the CBR algorithm (Fig. 6.7c). In this example the retrieved advice is 5.0 insulin units, and since no insulin on board adjustment is needed is also the final suggestion (highlighted in bold). The user can choose to override the advice manually by tapping *Override Bolus Suggestion* or tap *Cancel* to return to the main screen. Additionally, the user can also view the details of the cases reused to suggestion this solution by tapping the *Show Reused Cases* button. In this example the user wishes to accept the bolus insulin advice, and so must define the postprandial reminder time (in this case 180 minutes) and tap the *Set Reminder* to save a new case that is pending postprandial evaluation. This new

unevaluated case can be seen from the list of cases pending postprandial evaluation (Fig. 6.8b). After tapping *Set Reminder* the user is returned to the main screen (Fig. 6.7a).



(a) Main screen (b) New problem input (c) Retrieved solution

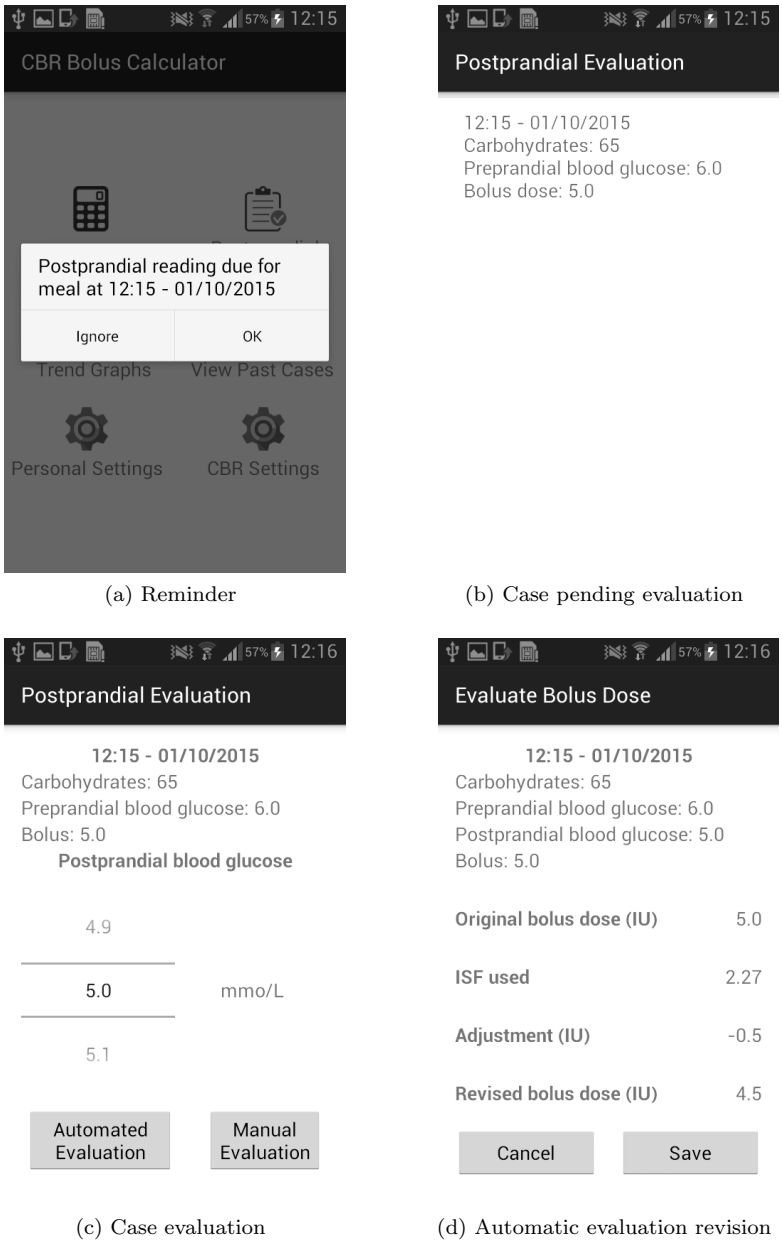
Figure 6.7: Obtain bolus advice use case demonstration

Postprandial bolus advice evaluation use case demonstration

The *postprandial bolus advice evaluation* use case can begin either through the reminder notification dialog box (Fig. 6.8a), which is automatically displayed after a certain duration defined by the user in the *obtain bolus advice* use case, or by viewing the cases pending evaluation by tapping *Postprandial Evaluation* on the main screen (Fig. 6.7a) and selecting the unevaluated case from the list (Fig. 6.8b).

After selecting either *OK* from the reminder dialog box or selecting a case from those listed as awaiting postprandial evaluation, the user is presented with the postprandial evaluation screen (Fig. 6.8c). On this screen the user is presented with details of the pending case, this includes the quantity of carbohydrates, the preprandial blood glucose reading, date and time, and the bolus dose. The user is then required to input their postprandial blood glucose reading using the number picker, in this example 5.0 mmol/L is input. The user may now tap *Automated Evaluation* to perform the automated evaluation described in Section 4.11, or perform manual evaluation by tapping the *Manual Evaluation* button. In this example automated evaluation will be used.

Following selection of automated evaluation, the user is presented with the bolus dose evaluation screen (Fig. 6.8d). This screen displays the original bolus dose and the revised bolus advice. Additionally, the insulin sensitivity factor (ISF) used and adjustment value is displayed to the user for reference. The evaluation screen provides the user the option of cancelling or saving the evaluated case. The user may wish to cancel if they have input the incorrect postprandial blood glucose value or if they wish to use manual evaluation instead. To complete the use case, the user



(a) Reminder

(b) Case pending evaluation

(c) Case evaluation

(d) Automatic evaluation revision

Figure 6.8: Postprandial bolus advice evaluation use case demonstration

taps the *Save* button, which will store the evaluated case in the case-base for future reuse and return the user to the main screen (Fig. 6.7a).

The demonstration of these two use cases illustrates the full CBR cycle on the implement mobile app. The CBR process in the app requires minimal input from the user, especially when the advice does not require manual adjustment and automated evaluation is used. Where input is required number pickers have been used where ever possible. This is to limit the user to valid inputs whilst still having the option of keyboard input that is limited to the required character sets. Both use cases demonstrated in this section can be completed in as few as nine actions (taps and spins) by the user, starting from the selection of the bolus calculator on the main screen through to the retaining of the evaluated case.

6.6 Case-based reasoning performance

A drawback of CBR is that its retrieval process can be resource intensive. This poses the question as to whether CBR is viable on a mobile device in order to meet Nielsen's usability requirements on acceptable response times [Nielsen, 1993]. Nielsen states that 10,000 ms is the limit for maintaining a user's attention. The prototype app is subjected to performance tests to determine the time taken in milliseconds to compute the resource intensive retrieval stage on an Android smart phone.[1]

The time taken to perform a retrieval is tested against sample case-bases containing an increasing number of cases. The retrieval process performs a full aggregate match across the entire case-base. The first case-base consists of 10 cases, the second 100, and the subsequent incrementing by 100 more cases, up to a total of 2,000 cases inclusive. The results obtained are from running the app on a single thread as multi-threading on the device failed to result in any performance improvements. This could be due to the overheads of multi-threading outweighing the benefits on the dual-core CPU of the device. This test is repeated five times to assess consistency between the results which are displayed in Table 6.4 and Fig. 6.9.

The results of the CBR retrieval performance on the device demonstrated acceptable computation time for smaller case-bases. The device on average processes retrieval in under 1,000 ms on a case-base of size 300, which based on an average of four cases a day would equate to 75 days of retained cases. The time taken to compute retrieval increases gradually as the case-base size grows, with the computation time increasing more rapidly with each additional case. In 10,000 ms, the device on average was able to compute retrieval on 1,100 cases, whilst retrieval on 2,000 cases requires around 32,500 ms.

Based on Nielsen [1993] 10,000 ms limit and the performance results, retrieval can be computed on 1,100 cases. This at an average of four cases recorded per day would allow 275 days of information to be used. Determining an optimal number of cases to perform a full aggregate match upon is difficult due to potentially varying user expectation, and differing performance between devices.[2] Through effectively utilising the multiple cores, increased frequency, and RAM available to the latest mobile devices, there is potential for retrieval to be performed on larger case-bases whilst still producing results in an acceptable time frame. Additionally, there is also much scope for improving the efficiency of the retrieval through dimensional matching.

[1] Samsung Galaxy S2 running Android 4.1.2. 1.2 GHz CPU and 1 GB of RAM.

[2] The latest equivalent model of the Samsung Galaxy S2 at the time of writing is the Samsung Galaxy S6, featuring a 2.1 GHz quad-core CPU and 3 GB RAM.

No. of cases	Time (ms)								
	Run 1	Run 2	Run 3	Run 4	Run 5	μ	$\pm$	Min	Max
10	32	22	8	9	9	16	16	8	32
100	222	250	126	125	127	170	80	125	250
200	495	504	408	477	399	456.6	47.4	399	504
300	1155	1021	969	871	850	973.2	181.8	850	1155
400	1685	1819	1404	1462	1400	1554	265	1400	1819
500	2448	2363	2150	2219	2261	2288.2	159.8	2150	2448
600	3249	3412	3034	2987	3023	3141	271	2987	3412
700	4467	4537	4203	4257	4073	4307.4	229.6	4073	4537
800	5883	5920	5208	5344	5488	5568.6	351.4	5208	5920
900	6830	6597	6689	6726	6794	6727.2	102.8	6597	6830
1000	8392	7847	8229	8029	8036	8106.6	285.4	7847	8392
1100	10485	9635	9610	9922	9635	9857.4	627.6	9610	10485
1200	11904	11564	11593	12981	11622	11932.8	1048.2	11564	12981
1300	14307	14131	13623	13513	13565	13827.8	479.2	13513	14307
1400	16902	15919	15779	15530	15763	15978.6	923.4	15530	16902
1500	19529	18146	17939	19395	18142	18630.2	898.8	17939	19529
1600	22608	20814	20486	20656	21155	21143.8	1464.2	20486	22608
1700	23826	22611	22838	22717	22899	22978.2	847.8	22611	23826
1800	26890	25507	25982	25667	26027	26014.6	875.4	25507	26890
1900	29279	28122	28260	28808	29997	28893.2	1103.8	28122	29997
2000	32821	32143	32856	31554	33139	32502.6	636.4	31554	33139

Table 6.4: Case-based reasoning retrieval performance results

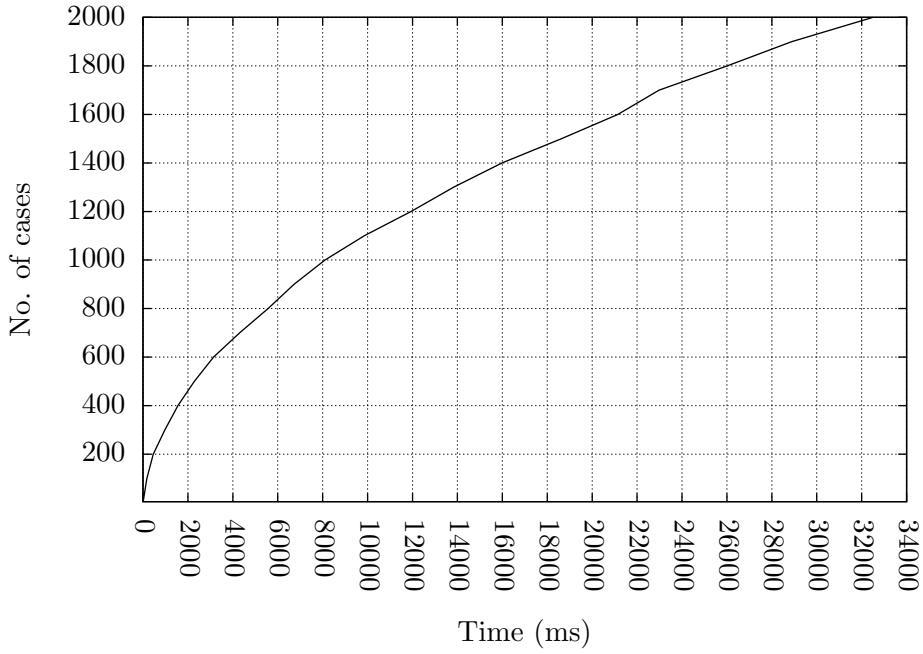


Figure 6.9: Mean time taken to perform CBR retrieval

6.7 Summary

In this chapter, the implementation of a prototype mobile app for bolus advice using CBR was discussed. The prototype allows the user to define their settings, build a case-base using a state-of-the-art bolus calculator formula, and use CBR for bolus advice. Trend graphs were also included in the prototype app to provide a means for data visualisation. The trend graphs implemented

provide information on blood glucose readings, carbohydrate intake and bolus insulin doses over a variable period of time.

The Monte Carlo method for case coverage estimation was used to gauge the approximate number of cases required to successfully retrieve cases of at least 90% similarity. The results found that retrieval success begins to plateau after 35 cases are acquired with a success rate of 55%. After 200 cases were acquired, the success rate increased to 65%. The Monte Carlo method adopts a uniform representation of the case-base, so it is likely the real-world success rate will be higher as a user's behaviour is likely to follow patterns with the distribution confined to clusters. As a fail safe mechanism, the prototype will revert to the bolus formula should no cases above 90% similarity be retrieved.

The implementation uses the formal specification described in Chapter 2 to ensure that the prototype successfully meets the constraints required of the system. The constraints include valid values for user definable settings and outputs from the system. The app was subjected to unit testing, where safety critical methods were tested to ensure the actual output matched the expected output for a selection of random valid and invalid inputs.

Performance tests of CBR on a mobile device demonstrated acceptable performance times of under 10,000 ms for case-bases up to 1,100 cases in size. With scope to improve efficiency of the retrieval algorithm, and with further advancements in mobile device performance, the prototype app demonstrates that CBR is viable on a mobile device.

Chapter 7

Conclusions and further research

This research resulted in the successful completion of the primary aim, to implement an intelligent and robust bolus adviser for T1DM subjects. The app's use of CBR resulted in improved simulated blood glucose control in comparison to the state-of-the-art bolus calculators assessed in this research. As a result, this research confirms the original hypothesis which stated that *CBR can be used to overcome the inability to learn and improve future advice.*

A notable contribution of this research is the resulting CBR system based on the R^4 model [Aamodt and Plaza, 1994]. The CBR system in this research enhanced the retrieval step through the introduction of temporal sequences and dynamic feature weighting, resulting in better case retrieval in comparison to a traditional approach. Temporal sequences allow factors from previous events to be considered when identifying the most similar case, which is an important consideration in temporal or sequential domains. Additionally, the research utilised domain specific rules to enable automatic adaptation and revision, allowing the system to both improve suggestions and optimise future advice. Results of these domain specific adaptation and revision rules showed significant improvement in simulated blood glucose control and highlighted the potential of CBR for bolus advice.

The research also adopted formal methods to assist the design of the resulting app. This helped to resolve safety issues prominent in mobile health apps currently available on the market. The use of formal methods helped to ensure that the domain was well understood and allowed system constraints to be validated prior to implementation.

This concluding chapter looks at the work undertaken in the previous chapters in relation to the aim and objectives outlined in Section 1.1. The chapter also looks at the impact of the research and makes recommendations for further research.

7.1 Reflection on the research aim and objectives

This section will reflect on the primary aim and objectives set out in Section 1.1 of the introductory chapter.

1. Identify the state-of-the-art approaches for T1DM bolus advice

A motivation for this research was the mobile POIRO MK4 expert system for T1DM self-management [Poerschke, 2004]. This particular system was created in the early 2000's during the era of palm PCs, when mobile technology was less powerful. As technology has advanced significantly since then, alongside the ever growing popularity of smartphones and tablets, it was important to see what state-of-the-art solutions for T1DM self-management currently exist.

The research into these state-of-the-art solutions were presented in Chapter 2, where the Accu-Chek™ Aviva Expert blood glucose meter and four mobile apps for iOS were assessed. Undertaking this task found that all the solutions employed a similar formula to calculate bolus insulin doses. These formulas can improve on past advice only through the optimisation of some constant values, which would either require clinician advice or a good understanding of the domain.

The Accu-Chek® Aviva Expert blood glucose meter served as a benchmark when comparing the mobile apps due to its FDA approved status, and the developers Roche being industry leaders. The mobile apps assessed found a large variation in terms of sophistication. RapidCalc was the only mobile app found for iOS that closely mirrored the functionality provided by the Accu-Chek® Aviva Expert blood glucose meter. This was not only terms of the features used in bolus calculations, but also additional features such as data recording, a comprehensive set of user definable settings, and the ability to visualise data. Other apps including Diabetic Dosage and Insulin Calc provided very little functionality in contrast, acting only as a simple bolus calculator with no ability to record data.

2. Discover how the state-of-the-art approaches predict bolus advice and evaluate what safety mechanisms are in place

The assessment of the state-of-the-art bolus calculators and associated literature in Chapter 2 highlighted many safety concerns. Some of the apps assessed did not impose constraints on user inputs, allowing the suggestion of potentially fatal advice. Additionally, some of the apps used suboptimal variations of the formula used by the Accu-Chek® Aviva Expert. These issues coupled with the lack of official assessment required, raised concerns about health apps available to major platforms. As a result, a formal approach was adopted during the design phase to best mirror the constraints of the FDA approved Accu-Chek®.

To ensure a firm understanding of the domain and constraints a bolus calculator was formally specified using the Event-B language. This step was adopted to ensure the perceived understanding

of how a bolus calculator works was correct. Additionally, it helped ensure that all data types and constraints were not violated by actions performed by the system. Performing this task aided the implementation in Chapter 6 greatly by ensuring the pre-conditions in the model were sufficient in preventing erroneous actions by the implemented system.

3. Discover how CBR provides intelligence and how it has been utilised in the T1DM domain previously

Before devising a CBR system for bolus advice it was important to first understand the mechanisms of CBR. In Chapter 3, we looked at the fundamental aspects of CBR. This research found the R^4 model to be the most understandable and widely adopted model in the field [Aamodt and Plaza, 1994]. The R^4 model draws upon earlier CBR work to produce a model for all purposes consisting of a four step cycle: retrieve, reuse, revise, and retain.

Although the R^4 model serves as a good guideline for creating a CBR system, the individual steps of the cycle are abstract, relying on the designers to identify suitable concrete methods for each stage of the cycle.

Starting with the retrieve step; the research identified that case retrieval can consist of two search methods, a dimensional match and an aggregate match. The dimensional match helps to reduce the number of cases considered for a more detailed aggregate match. Due to the rigid structure of the cases in the context of T1DM bolus advice, a dimensional match would be hard to achieve, since all features will be present in each case. For this reason, the research focused on aggregate matching, with a look at a few methods: distance functions, heuristic matching, and abstraction. Since the cases will work with continuous numeric values, distance functions were found to be the most appropriate for this research.

Following retrieval is the reuse of cases, where the retrieved solution can either be reused directly or adapted to better solve the problem. Adaptation allows differences to be reconciled between the problem and retrieved case, either through substitution or transformation. Substitution adaptation involves applying specialist rules to resolve differences, whilst transformation applies a heuristic approach to reconciling differences either by removing and adding features, or inferring feature values based on domain models. For the context of this research substitution adaptation was found to be the most suitable, with the ability to adapt the bolus advice through reconciling differences in active insulin.

After reuse it is assumed the adapted solution has been accepted. However, before retaining the case it is important that the solution is validated first in the revise stage. Revision of the case requires either real-world or simulated evaluation of the solution. This evaluation determines if there are any faults and how they can be resolved. In this research, postprandial evaluation was adopted to evaluate the solution, where a blood glucose reading is taken at a delayed time following the administration of the bolus insulin. This evaluation allowed the assessment of the

subject's postprandial reading against their target blood glucose level. The difference between the postprandial reading and target level is then used to revise the solution.

Finally, a case with its revised solution is retained in the case-base. The difficulty in the retain step is selecting appropriate indexes for a case, where appropriate levels of abstraction should be used to best index the problem for future retrievals. However, as the cases in the system are consistent in structure and all cases relate to the same feature value pairs, indexing the case-base did not seem beneficial.

Comparing CBR to rule-based and model-based reasoning discovered that although the approaches are different, there is often synergy between the methods. Many CBR systems adopt rule-based and model-based reasoning techniques in order to reconcile differences during adaptation or to assist the repair of solutions which failed. Looking at seminal CBR systems such as MEDIATOR [Simpson Jr, 1985], CHEF [Hammond, 1986], CASEY [Koton, 1988], and JULIA [Hinrichs and Kolodner, 1991], found that all these systems rely somewhat on rules or models to assist the CBR process. The same is true for this research, where adaptation and revision use specific rules to improve the solution.

By understanding the mechanisms of CBR, appropriate strategies to implement a CBR system based on the R^4 model were identified, which are consistent with techniques used by other CBR systems.

4. Develop a CBR system for predicting user tailored T1DM bolus advice

The strategies identified when exploring the CBR literature were then used to propose a CBR system for obtaining bolus suggestions. Chapter 4 discussed the CBR system in detail, and Chapter 5 analysed and evaluated the different steps in the CBR cycle. A key component of CBR is the retrieval of the most relevant previous cases. Research into retrieval investigated the effect of different distance metrics, temporal sequences, and feature weighting using feature selection algorithms. The retrieved solutions were analysed using statistical measures for continuous blood glucose readings.

Analysis of the case retrieval process when using temporal sequences to include previous events identified an improvement in blood glucose control. The result of larger temporal sequences was dependent upon other retrieval factors; in particular, the number of nearest neighbours retrieved, and the feature selection algorithm.

The results also yielded more stable blood glucose control with the use of feature weighting over retrieval with no feature weighting. The choice of feature selection algorithm made little difference to the results, but on average the One Rule feature selection algorithm produced the best blood glucose control and RELIEF-F feature selection algorithm the poorest. The observation that One Rule and RELIEF-F feature selection algorithms produced the best and worst retrieval results respectively is not unexpected due to their different approach to feature ranking. The

entropy based Information Gain, Gain Ratio and Symmetrical Uncertainty feature selection all resulted in similar weightings. It is expected that these algorithms would produce similar results as Gain Ratio and Symmetrical Uncertainty are evolutions of Information Gain. The results do suggest a minor improvement in blood glucose control from Symmetrical Uncertainty and Gain Ratio over Information Gain, which would reflect the improvements to the original algorithm. Although the feature weightings obtained from the Chi-Squared feature selection algorithm were similar to RELIEF-F in some instances, the results exhibited similar performance to the entropy based algorithms.

The last variant in the retrieval process is the distance metric used. This research tested two distance metrics Euclidean and Manhattan. Overall, little difference was observed between the two metrics, with a marginal benefit when using the Euclidean distance metric.

Adaptation and evaluation rules were then introduced to the CBR system. Adaptation was included to factor in the effects of insulin stacking before presenting a solution, reducing the risk of hypoglycaemic episodes. The inclusion of the adaptation rule to prevent insulin stacking produced notable improvements in blood glucose control, with a reduction in both the low blood glucose risk index and high blood glucose risk index. On average, this resulted in a 6.6% reduction in the blood glucose risk index compared to the continuous blood glucose levels prior to adaptation.

An automated evaluation rule to revise a bolus suggestion was then introduced to help improve the suggestions for future reuse. Automated evaluation revises the bolus suggestion based on the difference between a postprandial blood glucose reading and the subject's target blood glucose level. Repeated evaluation of the suggestions resulted in greatly improved blood glucose control. Five cycles of postprandial evaluation with a 3-hour meal offset resulted in a further reduction of 28.7% in the blood glucose risk index from the original solution with adaptation to account for insulin stacking.

5. Implement the system on a mobile device

A prototype mobile app was developed to assess the viability of CBR on a mobile device in Chapter 6. The app used the formal specification in Chapter 2 to help ensure the app was both safe and robust by conforming to the constraints of the Accu-Chek® Aviva Expert.

An initial problem to solve was how could CBR be used when there are no existing cases. Since the state-of-the-art applications utilise a formula, which despite not being intelligent is able to produce satisfactory results, it was decided that this approach could be used to build a case-base. This then lead to the question of how many cases are required for case-based reasoning in order to retrieve an appropriate solution. This question was addressed using the Monte Carlo method on a uniform distribution of the feature space. This method found that approximately 35 cases were required to reach the beginning of the plateau for reliably retrieving a case with 90% similarity. However, in real-world use the problem space is unlikely to follow a uniform distribution and

instead be isolated to clusters, so the number of cases required is likely to be less. As an extra safety measure, it was decided that the app should revert to the bolus formula if a defined similarity threshold is not met.

To guide the implementation, functional and non-functional requirements of the system were defined, and UML adopted in the form of use cases, use case descriptions and activity diagrams. Formally defining the behaviour of the system prior to implementation, alongside the formal specification helped to ensure the development of a reliable app which allowed the user to perform all the tasks available to the more sophisticated bolus calculators.

The resulting app provides a method to produce a subject specific case-base using a state-of-the-art bolus calculator formula, and the ability to evaluate the calculated results prior to case retention. Once a sufficient case-base is created, the user can obtain suggestions through CBR instead.

Implementation of the prototype app from the formal specification and design specification proved beneficial, with the unit tests of all safety critical methods passing without the need for debugging. This provides some evidence for the use of formal methods in aiding implementation, especially in ensuring preconditions successfully prevent the actions of the system from violating the system constraints.

6. Evaluate the implemented system for suitability on a mobile device

Performance analysis of the mobile prototype was conducted in Chapter 6 to assess the performance viability of CBR on a mobile device. The benchmark for CBR performance testing was set at 10,000ms using the upper acceptable response time described by Nielsen [1993]. Performance testing against this benchmark found that an Android device could successfully perform CBR on case-base size of up to 1,100 cases. This finding provides evidence that CBR is viable for T1DM bolus advice on mobile devices, and potentially other similar domains. Additionally, with continuing improvements in hardware and more efficient retrieval algorithms, it is feasible for CBR to be implemented on mobile devices in a range of contexts.

Primary aim: Develop an intelligent and robust mobile app using CBR to predict bolus insulin advice for T1DM subjects

The objectives outlined above all assisted in reaching the primary aim of this research; to develop an intelligent and robust mobile app using CBR to predict bolus insulin advice for T1DM subjects. The resulting mobile app uses CBR to fulfil the intelligence aspect of the tool, a process aided through a novel retrieval algorithm, and domain specific adaptation and revision rules. The CBR system proved to be just as effective as the approach used by state-of-the-art bolus calculators prior to any form of adaptation or learning through revision. With the added functionality of adaptation

to account for insulin on board, and the addition of revision through postprandial evaluation, the app is able to learn from suboptimal suggestions and improve future advice.

The issue of performance on a mobile device was a concern for using CBR. This research showed that CBR in this domain demonstrated acceptable response times on a mobile device. With the advancement of hardware and a more efficient retrieval algorithm, CBR appears to be a viable method to use on mobile devices.

The robustness of the resulting app was achieved through careful assessment of existing bolus calculators including the FDA approved Accu-Chek® Aviva Expert. To ensure the understanding of the domain was correct, a bolus calculator was initially modelled in Event-B. This process helped to define concrete system constraints which were successfully validated through unit testing after implementation.

In summary, this research successfully adapted and contributed to existing methods to fulfil the research aim, and confirms the hypothesis that CBR is a valid approach for improving bolus advice.

7.2 Impact

Successful management of T1DM is a difficult task for subjects due to the complexity of the condition. As a result, subjects often seek methods to aid their ability to successfully manage the condition through good blood glucose control. Successful management of blood glucose levels reduces the risk of long-term complications. Smart phones and tablets can provide a convenient and portable method to provide this assistance to a large number of T1DM subjects.

State-of-the-art mobile apps for T1DM bolus advice such as RapidCalc and Diabetes Personal Calculator for iOS provide a means for bolus advice, but are reliant on continuous medical advice from a doctor for optimal configuration. The research and development of an intelligent mobile app presented in this research provides a means to obtain bolus advice from past solutions. These solutions are not constrained to user defined constants and through evaluation and revision, future suggestions can be improved and tailored towards the subject.

The concepts discussed are not limited to mobile devices and could be utilised for insulin pumps, blood glucose monitors, personal computers, and as a web service accessible from any device with internet access. The potential versatility allows for a widely accessible intelligent solution to bolus advice.

7.3 Recommendations for further research

The research conducted was limited to a non-clinical perspective of the domain. Related literature and state-of-the-art approaches were used throughout the research to obtain a detailed understand-

ing of the domain, and to identify accepted measures for the analysis and evaluation of results. All the concepts presented were analysed and tested using the FDA-approved UVa/Padova T1DM simulator. However, the simulator constrained the number of features which could be accommodated. This resulted in the exclusion of features such as exercise and stress which are present in some existing bolus calculators. As a simulator is used, it is not possible to truly assess the real-world success of the concepts discussed in this research without clinical studies. The CBR system discussed in this research can be extended to include features which due to the limitations of the simulator could not be included. Some features omitted in this research which should explored include exercise, stress and alcohol. An expanded system with clinical trials would provide a real-world insight into the viability of CBR for T1DM self-management.

Evaluation of case retrieval illustrated how the quality of CBR suggestions are dependent upon the quality of the cases retained in the case-base. As a result, the effectiveness of CBR is reliant on a well-managed case-base produced by accurate user inputs and evaluation in addition to regular use. This is an area related to user behaviour, and the identification of methods to improve the user's interaction with the system would aid CBR in all domains that involve humans.

Efficiency of CBR retrieval was not an area of focus in this research. There is potential to increase the performance of the proposed retrieval process through the inclusion of dimensional matching to reduce the number of cases requiring an aggregate match. Improvements to the efficiency increases the range of devices for which CBR in this domain would be viable. As the majority of efficiency concerns reside in the retrieval process, a cloud service would be an approach to consider. Such an approach would have to be implemented carefully to prevent issues associated with network access, which is an area of concern for quality of experience.

A CBR service in the cloud also opens up the possibility of case sharing between subjects. This would reduce the need for a case-base to be created using a traditional bolus calculator or similar method, potentially allowing CBR to be used without the need to create a case-base. For case sharing to be reliable, a suitable method for identifying the similar subjects would be required. Additionally, steps would need to be taken to ensure the security and confidentiality of any stored user information.

Some elements of this research also have the potential for use in other domains. Most notably the use of temporal sequences to improve case retrieval in sequential domains. Research into the use of temporal sequences in other domains will help provide evidence to validate the method. Examples of such use may be other medical conditions which require self-management, cost-estimation for software-engineering, and the prediction of future events in a number of domains based on sequential events. There are certainly questions to be asked about how to determine optimal temporal sequence length, possibly through an hypothesis by domain experts which can then be validated through simulated or real-world testing. Additionally, the use of temporal sequences is highly dependant on feature weighting, so future research could explore the use of different

algorithms, or explore how the algorithms used in this research perform in other domains.

7.4 Concluding remarks

This research set out to see if CBR was a viable approach for predicting bolus insulin advice, and the analysis and evaluation conducted found that the approach is viable. Retrieval results prior to adaptation and revision produced suggestions similar to closed-loop simulation and state-of-the-art bolus calculators. The inclusion of adaptation and evaluation rules further improved the suggestions, with significant improvements in blood glucose control after repeated revisions. Performance of the CBR system on a mobile device resulted in acceptable performance of the app for up to 1,100 cases. With a large scope for improved efficiency of the resource intensive retrieval process and further improvements in mobile technology, it can be concluded that CBR is viable on mobile devices.

With continued research of intelligent systems for personalised T1DM advice, there is great potential for reliable, robust, cost effective, and accessible solutions to become available to an increasing number of individuals.

Bibliography

Aamodt, A. and Plaza, E. (1994). Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59.

Abrial, J.-R. (1996). *The B-Book: Assigning programs to meanings*. Cambridge University Press.

Abrial, J.-R. (2010). *Modeling in Event-B: system and software engineering*. Cambridge University Press.

Abrial, J.-R., Butler, M., Hallerstede, S., Hoang, T. S., Mehta, F., and Voisin, L. (2010). Rodin: an open toolset for modelling and reasoning in Event-B. *STTT*, 12(6):447–466.

Abrial, J.-R., Lee, M. K. O., Neilson, D. S., Scharbach, P. N., and Sørensen, I. H. (1991). The B-Method. In *VDM'91 Formal Software Development Methods*, pages 398–405, Noordwijkerhout, The Netherlands. Springer.

Alberti, K. G. M. M. and Zimmet, P. f. (1998). Definition, diagnosis and classification of diabetes mellitus and its complications. Part 1: diagnosis and classification of diabetes mellitus. Provisional report of a WHO consultation. *Diabetic medicine*, 15(7):539–553.

Allen, B. (1994). Case-based reasoning: Business applications. *Communications of the ACM*, 37(3):40–42.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Avramenko, Y. and Kraslawski, A. (2008). Case based design: Applications in process engineering. In *Studies in Computational Intelligence*, volume 87. Springer Science & Business Media.

Bain, W. (1986). A case-based reasoning system for subjective assessment. In *Proceedings of the 5th National Conference on Artificial Intelligence*, volume 86 of *AAAI-86*, pages 523–527, Philadelphia, PA, USA.

Barnard, K., Parkin, C., Young, A., and Ashraf, M. (2012). Use of an automated bolus calculator reduces fear of hypoglycemia and improves confidence in dosage accuracy in patients with type

1 diabetes mellitus treated with multiple daily insulin injections. *Journal of Diabetes Science and Technology*, 6(1):144–149.

Becton, Dickinson and Company (2005). *Staying on Target^{TM}: Your Insulin Adjustment Workbook. Yes, You Can Do It!* Becton, Dickinson and Company.

Bergmann, R. and Wilke, W. (1996). On the role of abstraction in case-based reasoning. In *Third European Workshop, EWCBR-96*, Advances in Case-Based Reasoning, pages 28–43, Lausanne, Switzerland. Springer.

Booch, G., Rumbaugh, J., and Jacobson, I. (2004). *Unified Modeling Language Reference Manual, The*. Pearson Higher Education.

Borus, J. S. and Laffel, L. (2010). Adherence challenges in the management of type 1 diabetes in adolescents: Prevention and intervention. *Current opinion in pediatrics*, 22(4):405.

Branting, L. K. and Hastings, J. D. (1994). An empirical evaluation of model-based case matching and adaptation. In *AAAI-94 Workshop: Case-Based Reasoning*, pages 72–78, Seattle, WA, USA.

Brown, D., Bayley, I., Harrison, R., and Martin, C. (2012). Formal specification of a mobile diabetes management application using the Rodin platform and Event-B. In *Rodin User and Development Workshop 2012*, Fontainebleau, France.

Brown, D., Bayley, I., Harrison, R., and Martin, C. (2013). Developing a mobile case-based reasoning application to assist type 1 diabetes management. In *2013 IEEE 15th International Conference on e-Health Networking, Applications & Services (Healthcom)*, Lisbon, Portugal. IEEE.

Campbell, R. and Abramovich, A. (2012). Calculating insulin on board for extended bolus being delivered by an insulin delivery device. US Patent 8,140,275.

Clarke, W. and Kovatchev, B. (2009). Statistical tools to analyse continuous glucose monitor data. *Diabetes Technology and Therapeutics*, 11:S–45–S–55.

Cunningham, P. and Delany, S. J. (2007). k-Nearest neighbour classifiers. *Multiple Classifier Systems*, pages 1–17.

Davidson, P. C., Hebblewhite, H. R., Bode, B. W., Steed, R. D., Welch, N. S., Greenlee, M. C., Richardson, P. L., and Johnson, J. (2003). Statistically based CSII parameters: correction factor, CF (1700 rule), carbohydrate-to-insulin ratio, CIR (2.8 rule), and basal-to-total ratio. *Diabetes Technology and Therapeutics*, 5(5):A237.

Davidson, P. C., Hebblewhite, H. R., Steed, R. D., and Bode, B. W. (2008). Analysis of guidelines for basal-bolus insulin dosing: basal insulin, correction factor, and carbohydrate-to-insulin ratio. *Endocrine Practice*, 14(9):1095–1101.

DCCT (1993). The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus. *New England Journal of Medicine*, 329(14):977–986. PMID: 8366922.

DCCT (2005). Intensive diabetes treatment and cardiovascular disease in patients with type 1 diabetes. *The New England journal of medicine*, 353(25):2643.

Diabetes.co.uk (2015). Basal bolus injection regimen. `http://www.diabetes.co.uk/insulin/basal-bolus.html`. Last accessed: 2015-06-11.

Diabetic Dosage (2015). Diabetic Dosage. `http://diabeticdosage.com/`. Last accessed: 2015-02-20.

El Emam, K., Benlarbi, S., Goel, N., and Rai, S. N. (2001). Comparing case-based reasoning classifiers for predicting high risk software components. *Journal of Systems and Software*, 55(3):301–320.

Farmer, T., Edgar, T., and Peppas, N. (2008). The future of open and closed-loop insulin delivery for diabetes mellitus. *The Journal of Pharmacy and Pharmacology*, 60:1–13.

Fayyad, U. M. and Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1027, Chambery, France. Morgan Kaufmann.

Finnie, G. and Sun, Z. (2003). R^5 model for case-based reasoning. *Knowledge-Based Systems*, 16(1):59–65.

Golding, A. R. and Rosenbloom, P. S. (1996). Improving accuracy by combining rule-based and case-based reasoning. *Artificial Intelligence*, 87(1):215–254.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. (2009). The Weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11:10–18.

Hall, M. A. (1999). *Correlation-based feature selection for machine learning*. PhD thesis, Department of Computer Science, The University of Waikato, Hamilton, Waikato, New Zealand.

Hall, M. A. and Smith, L. A. (1999). Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper. In *FLAIRS conference*, volume 1999, pages 235–239, Orlando, FL, USA.

Hammond, K. J. (1986). CHEF: A model of case-based planning. In *Proceedings of the 5th National Conference on Artificial Intelligence*, volume 1 of *AAAI-86*, pages 267–271, Philadelphia, PA, USA. AAAI Press/MIT Press.

Harvey, R. A., Wang, Y., Grosman, B., Percival, M. W., Bevier, W., Finan, D. A., Zisser, H., Seborg, D. E., Jovanovic, L., Doyle, F. J., and Dassau1, E. (2010). Quest for the artificial pancreas: Combining technology with treatment. *Engineering in Medicine and Biology Magazine, IEEE*, 29(2):53–62.

Heinemann, L. (2004). *Time-Action Profiles of Insulin Preparations*. Die Deutsche Bibliothek.

Herrero, P., Pesl, P., Reddy, M., Oliver, N., Georgiou, P., and Toumazou, C. (2014). Advanced insulin bolus advisor based on run-to-run control and case-based reasoning. *Biomedical and Health Informatics, IEEE Journal of*, 99.

Hinrichs, T. R. and Kolodner, J. L. (1991). The roles of adapation in case-based design. In *Proceedings of the 9th National Conference on Artificial Intelligence*, AAAI-91, pages 28–33, Anaheim, CA, USA. AAAI Press/MIT Press.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used dataset. *Machine Learning*, 11:63–91.

Hunt, J. (1995). Evolutionary case based design. In Waston, I., editor, *Progress in case-based reasoning*, pages 17–31. Springer, Berling.

Insulin Calc (2013). Insulin Calc. `http://www.maxwellapps.com/`. Last accessed: 2015-02-20.

iTenuto Soft (2015). Diabetes Personal Calculator. `http://iapps.itenutosoft.com/diabetes-personal-calculator/`. Last accessed: 2015-02-20.

Jaczynski, M. (1997). A framework for the management of past experiences with time-extended situations. In *Proceedings of the sixth international conference on Information and knowledge management*, pages 32–39, Las Vegas, NV, USA. ACM.

Jære, M. D., Aamodt, A., and Skalle, P. (2002). Representing temporal knowledge for case-based prediction. In *Advances in case-based reasoning*, pages 174–188. Springer.

Joint Formulary Committee (2010). *British national formulary*, volume 59. Pharmaceutical Press.

Kerber, R. (1992). ChiMerge: Discretization of numeric attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI-92, pages 123–128. AAAI Press.

Kerr, D. (2010). Poor numeracy: The elephant in the diabetes technology room. *Journal of Diabetes Science and Technology*, 4:1284–1287.

King, A. B. and Armstrong, D. U. (2007). A prospective evaluation of insulin dosing recommendations in patients with type 1 diabetes at near normal glucose control: Bolus dosing. *Journal of Diabetes Science and Technology*, 1(1):42–46.

Kira, K. and Rendell, L. (1992). A practical approach to feature selection. *Proceedings of the Ninth International Conference on Machine Learning*, pages 249–256.

Klonoff, D. C. (2012). The current status of bolus calculator decision-support software. *Journal of Diabetes Science and Technology*, 6(5):990–994.

Kolodner, J. (1983). Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7:243–280.

Kolodner, J. L. (1993). *Case-based Reasoning*. Morgan Kaufmann Publisher.

Kolodner, J. L. and Simpson, R. L. (1989). The MEDIATOR: Analysis of an early case-based problem solver. *Cognitive Science*, 13:507–549.

Kononenko, I. (1993). Inductive and bayesian learning in medical. *Applied Artificial Intelligence*, 7:317–337.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of relief. In *Procedings of the Seventh European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence, pages 171–182, Catania, Italy. Springer-Verlag Berlin Heidelberg.

Koton, P. (1988). Reasoning about evidence in casual explanations. In *Seventh National Conference on Artificial Intelligence*, AAAI-88, pages 256–261, St. Paul, MN, USA. AAAI Press.

Koton, P. (1989). *Using experience in learning and problem solving*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.

Kovatchev, B., Breton, M., Dalla Man, C., and Cobelli, C. (2009). In silico preclinical trials: A proof of concept in closed-loop control of type 1 diabetes. *Journal of Diabetes Science and Technology*, 3:44–55.

Kovatchev, B. P., Cox, D. J., Gonder-Frederick, L. A., Young-Hyman, D., Schlundt, D., and Clarke, W. (1998). Assessment of risk for severe hypoglycemia among adults with IDDM: Validation of the low blood glucose index. *Diabetes care*, 21(11):1870–1875.

Kovatchev, B. P., Otto, E., Cox, D., Gonder-Frederick, L., and Clarke, W. (2006). Evaluation of a new measure of blood glucose variability in diabetes. *Diabetes Care*, 29(11):2433–2438.

Kovatchev, B. P., Renard, E., Cobelli, C., Zisser, H. C., Keith-Hynes, P., Anderson, S. M., Brown, S. A., Chernavvsky, D. R., Breton, M. D., Mize, L. B., Farret, A., Place, J., Bruttomesso, D., Del Favero, S., Boscari, F., Galasso, S., Avogaro, A., Magni, L., Di Palma, F., Toffanin, C., Messori, M., Dassau, E., and Doyle, F. J. (2014). Safety of outpatient closed-loop control: First randomized crossover trials of a wearable artificial pancreas. *Diabetes Care*, 37(7):1789–1796.

Leake, D. and Wilson, M. (2011). How many cases do you need? Assessing and predicting case-base coverage. In *Case-Based Reasoning Research and Development*, pages 92–106. Springer.

Liu, H., Hussain, F., Tan, C. L., and Dash, M. (2002). Discretization: An enabling technique. *Data mining and knowledge discovery*, 6(4):393–423.

Liu, H. and Setiono, R. (1995). Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of the Seventh International Conference on Tools with Artificial Intelligence, 1995*, pages 388–391, Herndon, VA, USA. IEEE.

Long, W. J. and Naimi, S. (1987). The development and use of a causal model for reasoning about heart failure. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pages 30–36, Washington, DC, USA. American Medical Informatics Association.

MacDonald, C. M., Weber, R., and Richter, M. M. (2008). Case base properties: A first step. In *ECCBR workshops*, pages 159–170, Trier, Germany.

Marden, S., Thomas, P., Sheppard, Z., Knott, J., Lueddeke, J., and Kerr, D. (2012). Poor numeracy skills are associated with glycaemic control in type 1 diabetes. *Diabetic Medicine*, 29(5):662–669.

Marling, C., Shubrook, J., and Schwartz, F. (2007). Towards case-based reasoning for diabetes management. In *Workshop Proceedings of the Seventh International Conference on Case-Based Reasoning*, pages 305–314, Belfast, Northern Ireland.

Marling, C., Shubrook, J., and Schwartz, F. (2008). Case-based decision support for patients with type 1 diabetes on insulin pump therapy. *Advances in Case-Based Reasoning, Lecture Notes in Computer Science*, 5239:325–339.

Marling, C., Wiley, M., Cooper, T., Bunescu, R., Shubrook, J., and Schwartz, F. (2011). The 4 Diabetes Support System: A case study in CBR research and development. *Case-Based Reasoning Research and Development, Lecutre Notes in Computer Science*, 6880:137–150.

Martin, C., Flood, D., Sutton, D., Aldea, A., Harrison, R., and Waite, M. (2011). A systematic evaluation of mobile applications for diabetes management. In *INTERACT 2011, 13th IFIP TC 13 International Conference, Proceedings, Part IV*, Human-Computer Interaction, pages 466–469, Lisbon, Portugal. Springer.

McSherry, D. (2003). Increasing dialogue efficiency in case-based reasoning without loss of solution quality. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, IJCAI-03, pages 121–126, Acapulco, Mexico. Morgan Kaufmann Publisher.

Mendes, E., Watson, I., Triggs, C., Mosley, N., and Counsell, S. (2003). A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2):163–196.

Mingers, J. (1989). An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3:319–342.

Montani, S., Bellazzi, R., Portinale, L., d'Annunzio, G., Fiocchi, S., and Stefanelli, M. (2000). Diabetic patients management exploiting case-based reasoning techniques. *Computer Methods and Programs in Biomedicine*, 62:205–218.

Nevill-Manning, C. G., Holmes, G., and Witten, I. H. (1995). The development of Holte's 1R classifier. In *Proceedings of the Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, pages 239–242, Dunedin, New Zealand. IEEE.

NICE (2004). *Type 1 diabetes: Diagnosis and management of type 1 diabetes in children, young people and adults*. NICE.

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Nimri, R., Muller, I., Atlas, E., Miller, S., Kordonouri, O., Bratina, N., Tsioli, C., Stefanija, M. A., Danne, T., Battelino, T., and Phillip, M. (2014). Night glucose control with MD-Logic artificial pancreas in home setting: A single blind, randomized crossover trial—interim analysis. *Pediatric diabetes*, 15(2):91–99.

Poerschke, C. (2004). *Development and evaluation of an intelligent handheld insulin dose advisor for patients with Type 1 diabetes*. PhD thesis, Oxford Brookes University, Oxford, UK.

Poerschke, C., Lightfoot, D. E., and Nealon, J. L. (2003). A formal specification in B of a medical decision support system. In *Third International Conference of B and Z Users*, ZB 2003: Formal Specification and Development in Z and B, pages 497–512, Turku, Finland. Springer.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1988). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

RapidCalc (2015). RapidCalc. `http://www.gilport.com/rapidcalc/`. Last accessed: 2015-02-20.

Register, M. S. and Rewari, A. (1991). CANASTA: The crash analysis troubleshooting assistant. In *Proceedings of the Third Innovative Applications of Artificial Intelligence Conference on Artificial Intelligence*, IAAI-91, pages 195–212, Anaheim, CA, USA.

Riesbeck, C. K. and Schank, R. C. (2013). *Inside case-based reasoning*. Psychology Press.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5):465–471.

Roche (2012). *Accu-Chek® Aviva Expert Blood Glucose Meter - Standard Owner's Booklet*.

Roche (2013). *Accu-Chek® Aviva Expert Blood Glucose Meter - Advanced Owner's Booklet*.

Roche (2015a). Accu-Chek® - Global leaders of quality & innovation in diabetes care and management devices.
`https://www.accu-chek.co.uk/gb/heritage-statement.html`. Last accessed: 2015-02-20.

Roche (2015b). Blood glucose index.
`https://hcp.accu-chek.co.uk/gbconnect/blood-glucose-index.html`. Last accessed: 2015-02-20.

Rodbard, D. (2009). Interpretation of continuous glucose monitoring data: Glycemic variability and quality of glycemic control. *Diabetes Technology and Therapeutics*, 11:S–55–S–67.

Sánchez-Marré, M., Cortés, U., Martínez, M., Comas, J., and Rodríguez-Roda, I. (2005). An approach for temporal case-based reasoning: Episode-based reasoning. In *Case-Based Reasoning Research and Development*, pages 465–476. Springer.

Schank, R. (1983). *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, New York, NY, USA.

Schwartz, F. L., Schubrook, J. H., and Marling, C. R. (2008). Use of case-based reasoning to enhance intensive management of patients on insulin pump therapy. *Journal of Diabetes Science and Technology*, 2:603–611.

Simoudis, E. (1992). Using case-based retrieval for customer technical support. *IEEE Expert: Intelligent Systems and Their Applications*, 7:7–12.

Simpson Jr, R. L. (1985). A computer model of case-based reasoning in problem solving: An investigation in the domain of dispute mediation. Technical report, DTIC Document.

Skalak, D. B. and Rissland, E. L. (1990). Inductive learning in a mixed paradigm setting. In *Proceedings of the Eighth National Conference on Artificial Intelligence - Volume 2*, AAAI-90, pages 840–847. AAAI Press.

Snook, C. and Harrison, R. (2001). Practitioners' views on the use of formal methods: An industrial survey by structured interview. *Information and Software Technology*, 43(4):275–283.

Sussman, A., Taylor, E. J., Patel, M., Ward, J., Alva, S., Lawrence, A., and Ng, R. (2012). Performance of a glucose meter with a built-in automated bolus calculator versus manual bolus calculation in insulin-using subjects. *Journal of Diabetes Science and Technology*, 6(2):339–344.

Toffanin, C., Zisser, H., Doyle, F. J., and Eyal, D. (2013). Dynamic insulin on board: Incorporation of circadian insulin sensitivity variation. *Journal of Diabetes Science and Technology*, 7:928–940.

Tu, J. V. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of clinical epidemiology*, 49(11):1225–1231.

Tukey, J. W. (1977). *Exploratory data analysis*. Pearson.

University College London Hospitals (2013). *Correcting a high blood glucose level*. NHS.

Walsh, J. and Roberts, R. (2015). An accurate DIA prevents excessive insulin stacking. `http://www.diabetesnet.com/about-diabetes/insulin/insulin-action-time/duration-insulin-action`. Last accessed: 2015-02-20.

Walsh, J., Roberts, R., and Bailey, T. (2011). Guidelines for optimal bolus calculator settings in adults. *Journal of Diabetes Science and Technology*, 5:129–135.

Wess, S., Althoff, K.-d., and Derwand, G. (1993). Using k-d trees to improve the retrieval step in case-based reasoning. In *EWCBR '93 Selected papers from the First European Workshop on Topics in Case-Based Reasoning*, pages 167–181, Kaiserslautern, Germany. Springer-Verla.

WHO (2011). *Use of glycated haemoglobin (HbA1c) in diagnosis of diabetes mellitus: abbreviated report of a WHO consultation*. Geneva: World Health Organization.

WHO (2014a). Facts and figures about diabetes. `http://www.who.int/features/factfiles/diabetes/en/`. Last accessed: 2015-02-20.

WHO (2014b). *Global status report on noncommunicable diseases 2014*. World Health Organisation.

Wilson, D. R. and Martinez, T. R. (1997). Improved heterogeneous distance functions. *Journal of Artifical Intelligence Research*, 6:1–34.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Woodcock, J., Larsen, P. G., Bicarregui, J., and Fitzgerald, J. (2009). Formal methods: Practice and experience. *ACM Computing Surveys (CSUR)*, 41(4):19.

Woodworth, J. R., Howey, D. C., and Bowsher, R. (1994). Establishment of time-action profiles for regular and NPH insulin using pharmacodynamic modeling. *Diabetes Care*, 1:64–69.

Yu, L. and Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the Twentieth Internation Conference on Machine Learning*, volume 3, pages 856–863, Washington, DC, USA.

Appendices

Appendix A

Event-B specification: Abstract machine

MACHINE t1dm_m0

VARIABLES

`active` Active insulin time (minutes)

`basal` Basal insulin (insulin units)

`bgThresHigh` Upper blood glucose threshold (mmol/L)

`bgThresLow` Lower blood glucose threshold (mmol/L)

`carbRatioC` Carbohydrate-to-insulin ratio: carbohydrates per carbRatioI

`carbRatioI` Carbohydrate-to-insulin ratio: insulin units (insulin units)

`hyperThreshold` Hyperglycaemia threshold (mmol/L)

`hypoThreshold` Hypoglycaemia threshold (mmol/L)

`isfBG` Insulin sensitvity factor: reduction in blood glucose for isfI insulin units (mmol/L)

`isfI` Insulin sensitivity factor: insulin units

`maxBolus` Maximum bolus dose (insulin units)

`targetRangeUpper` Target blood glucose range upper value (mmol/L)

`targetRangeLower` Target blood glucose range lower value (mmol/L)

`bolusSuggestion` Suggested bolus dose (insulin units)

`settingsDefined` Boolean informing the system the user settings are defined

INVARIANTS

`inv1` : $active \in \mathbb{N}$

`inv2` : $active \geq 900 \wedge active \leq 4800$

Active insulin time must be in the range [90.0,480.0] minutes

`inv3` : $basal \in \mathbb{N}$

`inv4` : $basal \leq 990$

Basal insulin dose must be in the range [0.0,99.0] IU

`inv5` : $bgThresHigh \in \mathbb{N}$

`inv6` : $bgThresHigh \geq 65 \wedge bgThresHigh \leq 195$

High blood glucose threshold must be in the range [6.5,19.5] mmol/L

`inv7` : $bgThresLow \in \mathbb{N}$

`inv8` : $bgThresLow \geq 30 \wedge bgThresLow \leq 55$

Low blood glucose threshold must be in the range [3.0,5.5] mmol/L

`inv9` : $carbRatioC \in \mathbb{N}$

inv10 : $carbRatioC \geq 10 \wedge carbRatioC \leq 2400$
Carbohydrates correct by carb-to-insulin ratio per carbRatioI must be in the range [1.0,240.0] grams

inv11 : $carbRatioI \in \mathbb{N}$

inv12 : $carbRatioI \geq 1 \wedge carbRatioI \leq 500$
Insulin units for carb-to-insulin ratio pmust be in the range [1.0,50.0] mmol/L

inv13 : $hyperThreshold \in \mathbb{N}$

inv14 : $hyperThreshold \geq 100 \wedge hyperThreshold \leq 195$
Hyperglycaemia threshold must be in the range [10.0,19.5] mmol/L

inv15 : $hypoThreshold \in \mathbb{N}$

inv16 : $hypoThreshold \geq 30 \wedge hypoThreshold \leq 50$
Hypoglycaemia threshold must be in the range [3.0,5.0] mmol/L

inv17 : $isfBG \in \mathbb{N}$

inv18 : $isfBG \geq 10 \wedge isfBG \leq 554$
Blood glucose reduction for ISF must be in the range [1.0,55.4] mmol/L

inv19 : $isfI \in \mathbb{N}$

inv20 : $isfI \geq 1 \wedge isfI \leq 500$
Insulin units for ISF must be in the range [1.0,50.0] IU

inv21 : $maxBolus \in \mathbb{N}$

inv22 : $maxBolus \leq 500$
The maximum bolus dose must be in the range [0.0,50.0] IU

inv23 : $targetRangeUpper \in \mathbb{N}$

inv24 : $targetRangeUpper \geq 55 \wedge targetRangeUpper \leq 150$
Target blood glucose range upper value must be in the range [5.5,15.0]

inv25 : $targetRangeLower \in \mathbb{N}$

inv26 : $targetRangeLower \geq 30 \wedge targetRangeLower \leq 80$
Target blood glucose range lower value must be in the range [3.0,8.0]

inv27 : $bolusSuggestion \in \mathbb{N}$

inv28 : $bolusSuggestion \leq maxBolus$
The bolus suggestion must be < = the maximum allowed bolus dose

inv29 : $settingsDefined \in BOOL$

EVENTS

Initialisation

Initilialises the machine

begin

act1 : $active := 2400$
Default active insulin time 240.0 minutes

act2 : $basal := 0$
Default basal dose 0 IU

act3 : $bgThresHigh := 165$
Default high blood glucose threshold 16.5 mmol/L

act4 : $bgThresLow := 40$
Default low blood glucose threshold 4.0 mmol/L

act5 : $carbRatioC := 10$
Default carbohydrates per carbRatioI insulin unit for carbohydrate-to-insulin ratio

act6 : $carbRatioI := 10$
Default insulin unit for carbohydrate-to-insulin ratio

act7 : $hyperThreshold := 165$
Default hyperglycamia glucose threshold 16.5 mmol/L

act8 : $hypoThreshold := 40$
Default hypoglycamia glucose threshold 16.5 mmol/L

act9 : $isfBG := 10$
Default reduction in blood glucose per isfI insulin unit for insulin sensitivty factor

act10 : $isfI := 10$
Default insulin unit for insulin sensitvity factor

act11 : $maxBolus := 0$
Default maximum bolus suggestion

act12 : $targetRangeUpper := 80$
Target blood glucose range upper value
act13 : $targetRangeLower := 40$
Target blood glucose range lower value
act14 : $bolusSuggestion := 0$
Bolus suggestion initially 0
act15 : $settingsDefined := FALSE$
Settings are not defined

end

Event *bolusCalc* $\widehat{=}$
Bolus calculator event for instances where the result is ≥ 0 and $\leq$ maxBolus

any
s Bolus suggestion

where
grd1 : $s \in \mathbb{N}_1$
Bolus suggestion is greater than 0
grd2 : $s \leq maxBolus$
The bolus solution suggestion is less than or equal to maxBolus

then
act1 : $bolusSuggestion := s$
Sets the bolus suggestion value

end

Event *bolusCalcNeg* $\widehat{=}$
Bolus calculator event for instances where s < 0

any
s Bolus suggestion

where
grd1 : $s \in \mathbb{Z}$
Bolus suggestion is any integer
grd2 : $s \leq 0$
The bolus suggestion is less than 0

then
act1 : $bolusSuggestion := 0$
Sets the bolus suggestion to 0 as negative values are not permitted.

end

Event *bolusCalcMax* $\widehat{=}$
Bolus calculator event for instances where s $>$ maxBolus

any
s Bolus suggestion

where
grd1 : $s \in \mathbb{N}$
grd2 : $s > maxBolus$
The bolus solution suggestion is greater than maxBolus

then
act1 : $bolusSuggestion := maxBolus$
Sets the bolus suggestion equal to maxBolus as values greater than maxBolus are not permitted

end

Event *defineSettings* $\widehat{=}$
Modifies the users settings

any
ac Active insulin time
ba Daily basal dose
bgH High blood glucose warning threshold
bgL Low blood glucose warning threshold
crC Carbohydrate-to-insulin ratio: carbohydrates
crI Carbohydrate-to-insulin ratio: insulin units
heT Hyperglycaemia warning threshold
hoT Hypoglycaemia warning threshold
iBG Insulin sensitivity factor: blood glucose

iI Insulin sensitivity factor: insulin units
mB Maximum allowed bolus dose
tU Target blood glucose upper value
tL Target blood glucose lower value
b Revised bolus value, if the current state is invalidated by a change to maxBolus

where

grd1 : $ac \in \mathbb{N}$
grd2 : $ac \geq 900 \wedge ac \leq 4800$
grd3 : $ba \in \mathbb{N}$
grd4 : $ba \leq 990$
grd5 : $bgH \in \mathbb{N}$
grd6 : $bgH \geq 65 \wedge bgH \leq 195$
grd7 : $bgL \in \mathbb{N}$
grd8 : $bgL \geq 30 \wedge bgL \leq 55$
grd9 : $crC \in \mathbb{N}$
grd10 : $crC \geq 10 \wedge crC \leq 2400$
grd11 : $crI \in \mathbb{N}$
grd12 : $crI \geq 1 \wedge crI \leq 500$
grd13 : $heT \in \mathbb{N}$
grd14 : $heT \geq 100 \wedge heT \leq 195$
grd15 : $hoT \in \mathbb{N}$
grd16 : $hoT \geq 30 \wedge hoT \leq 50$
grd17 : $iBG \in \mathbb{N}$
grd18 : $iBG \geq 10 \wedge iBG \leq 554$
grd19 : $iI \in \mathbb{N}$
grd20 : $iI \geq 1 \wedge iI \leq 500$
grd21 : $mB \in \mathbb{N}$
grd22 : $mB \leq 500$
grd23 : $tU \in \mathbb{N}$
grd24 : $tU \geq 55 \wedge tU \leq 150$
grd25 : $tL \in \mathbb{N}$
grd26 : $tL \geq 30 \wedge tL \leq 80$
grd27 : $b \in \mathbb{N}$
bolusSuggestion, required incase the new maxBolus violates the current state
grd28 : $bolusSuggestion > mB \Rightarrow b = mB$
If bolusSuggestion in the current state would violate the new maxBolus value, then amend the bolusSuggestion to maxBolus
grd29 : $bolusSuggestion \leq mB \Rightarrow b = bolusSuggestion$
If bolusSuggestion in the current state does violate the new maxBolus value, then maintain the same value

then

act1 : $active := ac$
act2 : $basal := ba$
act3 : $bgThresHigh := bgH$
act4 : $bgThresLow := bgL$
act5 : $carbRatioC := crC$
act6 : $carbRatioI := crI$
act7 : $hyperThreshold := heT$
act8 : $hypoThreshold := hoT$
act9 : $isfBG := iBG$
act10 : $isfI := iI$
act11 : $maxBolus := mB$
act12 : $targetRangeUpper := tU$
act13 : $targetRangeLower := tL$
act14 : $bolusSuggestion := b$

end

END

Appendix B

Event-B specification: First refinement

MACHINE t1dm_m1
REFINES t1dm_m0
VARIABLES

`active` Active insulin time (minutes)
`basal` Basal insulin (insulin units)
`bgThresHigh` Upper blood glucose threshold (mmol/L)
`bgThresLow` Lower blood glucose threshold (mmol/L)
`carbRatioC` Carbohydrate-to-insulin ratio: carbohydrates per carbRatioI (grams)
`carbRatioI` Carbohydrate-to-insulin ratio: insulin units (insulin units)
`hyperThreshold` Hyperglycaemia threshold (mmol/L)
`hypoThreshold` Hypoglycaemia threshold (mmol/L)
`isfBG` Insulin sensitvity factor: reduction in blood glucose for isfI insulin units (mmol/L)

`isfI` Insulin sensitivity factor: insulin units (insulin units)
`maxBolus` Maximum bolus dose (insulin units)
`targetRangeUpper` Target blood glucose range upper value (mmol/L)
`targetRangeLower` Target blood glucose range lower value (mmol/L)
`bolusSuggestion` Suggested bolus dose (insulin units)
`carbs` User input for planned carbohydrate intake
`preBG` User input for preprandial blood glucose level (mmol/L)
`inputsDefined` Boolean informing the system the user inputs are defined
`activeInsulin` Active insulin (insulin units)
`settingsDefined` Boolean informing the system the user settings are defined

INVARIANTS

`inv1` : $carbs \in \mathbb{N}$
`inv2` : $carbs \leq 2400$
`inv3` : $preBG \in \mathbb{N}$
`inv4` : $preBG \leq 554$
`inv5` : $inputsDefined \in BOOL$
`inv6` : $activeInsulin \in \mathbb{N}$

EVENTS

Initialisation

extended

Initilialises the machine

begin

`act1` : `active` := `2400`
Default active insulin time 240.0 minutes

`act2` : `basal` := `0`
Default basal dose 0 IU

`act3` : `bgThresHigh` := `165`
Default high blood glucose threshold 16.5 mmol/L

`act4` : `bgThresLow` := `40`
Default low blood glucose threshold 4.0 mmol/L

`act5` : `carbRatioC` := `10`
Default carbohydrates per carbRatioI insulin unit for carbohydrate-to-insulin ratio

`act6` : `carbRatioI` := `10`
Default insulin unit for carbohydrate-to-insulin ratio

`act7` : `hyperThreshold` := `165`
Default hyperglycamia glucose threshold 16.5 mmol/L

`act8` : `hypoThreshold` := `40`
Default hypoglycamia glucose threshold 16.5 mmol/L

`act9` : `isfBG` := `10`
Default reduction in blood glucose per isfI insulin unit for insulin sensitivty factor

`act10` : `isfI` := `10`
Default insulin unit for insulin sensitvity factor

`act11` : `maxBolus` := `0`
Default maximum bolus suggestion

`act12` : `targetRangeUpper` := `80`
Target blood glucose range upper value

`act13` : `targetRangeLower` := `40`
Target blood glucose range lower value

`act14` : `bolusSuggestion` := `0`
Bolus suggestion initially 0

`act15` : `settingsDefined` := `FALSE`
Settings are not defined

`act16` : $carbs := 0$
Initial carbohydrate value is 0. Must be modified before use

`act17` : $preBG := 0$
Initial preprandial blood glucose value is 0. Must be modified before use

`act18` : $inputsDefined := FALSE$
User inputs are not defined

`act19` : $activeInsulin := 0$
Initial active insulin value is 0

end

Event $bolusCalc \mathrel{\widehat=}$

Bolus calculator event for instances where the result is ≥ 0 and $\leq$ maxBolus

extends *bolusCalc*

any

`s` Bolus suggestion

where

`grd1` : $\mathtt{s} \in \mathbb{N}_1$
Bolus suggestion is greater than 0

`grd2` : $\mathtt{s} \leq \mathtt{maxBolus}$
The bolus solution suggestion is less than or equal to the maximum bolus dose

`grd4` : $s = (preBG - (targetRangeLower + (targetRangeUpper - targetRangeLower)/2)) * (isfI/isfBG) + carbs * (carbRatioI/carbRatioC) - activeInsulin$
The bolus suggestion equals the bolus calculation result

`grd5` : $inputsDefined = TRUE$
The user inputs for the calculation must be done first

then

`act1` : `bolusSuggestion` := `s`
Sets the bolus suggestion value

end

Event *bolusCalcNeg* $\widehat{=}$
Bolus calculator event for instances where s < 0

extends *bolusCalcNeg*

any
`s` Bolus suggestion

where
`grd1` : `s` $\in \mathbb{Z}$
Bolus suggestion is any integer
`grd2` : `s` $\leq$ `0`
The bolus suggestion is less than 0
`grd3` : $s = (preBG - (targetRangeLower + (targetRangeUpper - targetRangeLower)/2)) * (isfI/isfBG) + carbs * (carbRatioI/carbRatioC) - activeInsulin$
The bolus suggestion equals the bolus calculation result
`grd4` : $inputsDefined = TRUE$
The user inputs for the calculation must be done first

then
`act1` : `bolusSuggestion` := `0`
Sets the bolus suggestion to 0 as negative values are not permitted.

end

Event *bolusCalcMax* $\widehat{=}$
Bolus calculator event for instances s $>$ maxBolus

extends *bolusCalcMax*

any
`s` Bolus suggestion

where
`grd1` : `s` $\in \mathbb{N}$
`grd2` : `s` $>$ `maxBolus`
The bolus solution suggestion is greater than the maximum bolus dose
`grd3` : $s = (preBG - (targetRangeLower + (targetRangeUpper - targetRangeLower)/2)) * (isfI/isfBG) + carbs * (carbRatioI/carbRatioC) - activeInsulin$
The bolus suggestion equals the bolus calculation result
`grd4` : $inputsDefined = TRUE$
The user inputs for the calculation must be done first

then
`act1` : `bolusSuggestion` := `maxBolus`
Sets the bolus suggestion to maxBolus as values greater than maxBolus are not permitted

end

Event *defineSettings* $\widehat{=}$
Modifies the users settings

extends *defineSettings*

any
`ac` Active insulin time
`ba` Daily basal dose
`bgH` High blood glucose warning threshold
`bgL` Low blood glucose warning threshold
`crC` Carbohydrate-to-insulin ratio: carbohydrates
`crI` Carbohydrate-to-insulin ratio: insulin units
`heT` Hyperglycaemia warning threshold
`hoT` Hypoglycaemia warning threshold
`iBG` Insulin sensitivity factor: blood glucose
`iI` Insulin sensitivity factor: insulin units
`mB` Maximum allowed bolus dose
`tU` Target blood glucose upper value
`tL` Target blood glucose lower value
`b` Revised bolus value, if the current state is invalidated by a change to maxBolus

where

grd1 : ac ∈ ℕ
grd2 : ac ≥ 900 ∧ ac ≤ 4800
grd3 : ba ∈ ℕ
grd4 : ba ≤ 990
grd5 : bgH ∈ ℕ
grd6 : bgH ≥ 65 ∧ bgH ≤ 195
grd7 : bgL ∈ ℕ
grd8 : bgL ≥ 30 ∧ bgL ≤ 55
grd9 : crC ∈ ℕ
grd10 : crC ≥ 10 ∧ crC ≤ 2400
grd11 : crI ∈ ℕ
grd12 : crI ≥ 1 ∧ crI ≤ 500
grd13 : heT ∈ ℕ
grd14 : heT ≥ 100 ∧ heT ≤ 195
grd15 : hoT ∈ ℕ
grd16 : hoT ≥ 30 ∧ hoT ≤ 50
grd17 : iBG ∈ ℕ
grd18 : iBG ≥ 10 ∧ iBG ≤ 554
grd19 : iI ∈ ℕ
grd20 : iI ≥ 1 ∧ iI ≤ 500
grd21 : mB ∈ ℕ
grd22 : mB ≤ 500
grd23 : tU ∈ ℕ
grd24 : tU ≥ 55 ∧ tU ≤ 150
grd25 : tL ∈ ℕ
grd26 : tL ≥ 30 ∧ tL ≤ 80
grd27 : b ∈ ℕ
bolusSuggestion, required incase the new maxBolus violates the current state
grd28 : bolusSuggestion > mB ⇒ b = mB
If bolusSuggestion in the current state would violate the new maxBolus value, then amend the bolusSuggestion to maxBolus
grd29 : bolusSuggestion ≤ mB ⇒ b = bolusSuggestion
If bolusSuggestion in the current state does violate the new maxBolus value, then maintain the same value

then

act1 : active := ac
act2 : basal := ba
act3 : bgThresHigh := bgH
act4 : bgThresLow := bgL
act5 : carbRatioC := crC
act6 : carbRatioI := crI
act7 : hyperThreshold := heT
act8 : hypoThreshold := hoT
act9 : isfBG := iBG
act10 : isfI := iI
act11 : maxBolus := mB
act12 : targetRangeUpper := tU
act13 : targetRangeLower := tL
act14 : bolusSuggestion := b

end

Event *defineInput* ≙

Modifies user inputs for the bolus calculation

any

c Planned carbohydrate intake
bg Preprandial blood glucose level (mmol/L)

where

grd1 : $c \in \mathbb{N}$
Carbohydrate value must be greater than or equal to 0
grd2 : $c \leq 2400$
Carbohydrate value must be less than or equal to 240 grams

grd3 : $bg \in \mathbb{N}$
Blood glucose value must be greater than or equal to 0
grd4 : $bg \leq 554$
Carbohydrate value must be less than or equal to 55.4 mmol/L

then

act1 : $carbs := c$
act2 : $preBG := bg$
act3 : $inputsDefined := TRUE$
The user inputs have been defined

end

Event $calcActiveInsulin \mathrel{\widehat{=}}$
Calculates active insulin

any

a Active insulin

where

grd1 : $a \in \mathbb{N}$
Active insulin must be greater than or equal to 0

then

act1 : $activeInsulin := a$

end

END

Appendix C

Event-B specification: Second refinement

MACHINE t1dm_m2
REFINES t1dm_m1
VARIABLES

- `active` Active insulin time
- `basal` Basal insulin
- `bgThresHigh` Upper blood glucose threshold
- `bgThresLow` Lower blood glucose threshold
- `carbRatioC` Carbohydrate-to-insulin ratio: carbohydrates per carbRatioI
- `carbRatioI` Carbohydrate-to-insulin ratio: insulin units
- `hyperThreshold` Hyperglycaemia threshold
- `hypoThreshold` Hypoglycaemia threshold
- `isfBG` Insulin sensitvity factor: reduction in blood glucose for isfI insulin units
- `isfI` Insulin sensitivity factor: insulin units
- `maxBolus` Maximum bolus dose
- `targetRangeUpper` Target blood glucose range upper value
- `targetRangeLower` Target blood glucose range lower value
- `bolusSuggestion` Suggested bolus dose
- `carbs` User input for planned carbohydrate intake
- `preBG` User input for preprandial blood glucose level (mmol/L)
- `inputsDefined` Boolean informing the system the user inputs are defined
- `activeInsulin` Active insulin (insulin units)
- `settingsDefined` Boolean informing the system the user settings are defined
- `caseBase` Sequence of case IDs
- `caseCarbs` Function mapping a case ID to the carbohydrate intake
- `casePreBG` Function mapping a case ID to the preprandial blood glucose reading
- `caseUsedBolus` Function mapping a case ID to the bolus suggestion
- `caseTime` Function mapping a case ID to time
- `time` Time of the calculation
- `activeInsulinCases` Set of applicable cases active insulin has been calculated for
- `solutionObtained` Boolean defining if a bolus solution has been obtained

INVARIANTS

inv1 : $caseBase \subseteq \mathbb{N}_1$
Case IDs are a subset of natural numbers

inv2 : $finite(caseBase)$
Case-base (case IDs) has cardinality

inv3 : $caseCarbs \in caseBase \rightarrow \mathbb{N}$
Total function mapping every case ID to carbohydrate intake

inv4 : $\forall c \cdot c \in ran(caseCarbs) \Rightarrow c \leq 2400$
All retain carbohydrate values must be less than or equal to 240 grams

inv5 : $casePreBG \in caseBase \rightarrow \mathbb{N}$
Total function mapping every case ID to preprandial blood glucose reading

inv6 : $\forall bg \cdot bg \in ran(casePreBG) \Rightarrow bg \leq 554$
All retained preprandial blood glucose readings must be less than or equal to 55.4 mmol/L

inv7 : $caseUsedBolus \in caseBase \rightarrow \mathbb{N}$
Total function mapping every case ID to a bolus suggestion

inv8 : $caseTime \in caseBase \rightarrow \mathbb{N}$
Total function mapping every case ID to case time

inv9 : $time \in \mathbb{N}$
Time is a number greater than or equal to 0

inv10 : $activeInsulinCases \subseteq caseBase$
Applicable active insulin cases are a member of caseBase

inv11 : $solutionObtained \in BOOL$

EVENTS

Initialisation

extended
Initilialises the machine

begin

act1 : `active` := `2400`
Default active insulin time 240.0 minutes

act2 : `basal` := `0`
Default basal dose 0 IU

act3 : `bgThresHigh` := `165`
Default high blood glucose threshold 16.5 mmol/L

act4 : `bgThresLow` := `40`
Default low blood glucose threshold 4.0 mmol/L

act5 : `carbRatioC` := `10`
Default carbohydrates per carbRatioI insulin unit for carbohydrate-to-insulin ratio

act6 : `carbRatioI` := `10`
Default insulin unit for carbohydrate-to-insulin ratio

act7 : `hyperThreshold` := `165`
Default hyperglycamia glucose threshold 16.5 mmol/L

act8 : `hypoThreshold` := `40`
Default hypoglycamia glucose threshold 16.5 mmol/L

act9 : `isfBG` := `10`
Default reduction in blood glucose per isfI insulin unit for insulin sensitivty factor

act10 : `isfI` := `10`
Default insulin unit for insulin sensitvity factor

act11 : `maxBolus` := `0`
Default maximum bolus suggestion

act12 : `targetRangeUpper` := `80`
Target blood glucose range upper value

act13 : `targetRangeLower` := `40`
Target blood glucose range lower value

act14 : `bolusSuggestion` := `0`
Bolus suggestion initially 0

act15 : `settingsDefined` := `FALSE`
Settings are not defined

act16 : carbs := 0
Initial carbohydrate value is 0. Must be modified before use
act17 : preBG := 0
Initial preprandial blood glucose value is 0. Must be modified before use
act18 : inputsDefined := FALSE
User inputs are not defined
act19 : activeInsulin := 0
Initial active insulin value is 0
act20 : *caseBase* := ∅
No cases initially
act21 : *caseCarbs* := ∅
No cases initially
act22 : *casePreBG* := ∅
No cases initially
act23 : *caseUsedBolus* := ∅
No cases initially
act24 : *caseTime* := ∅
No cases initially
act25 : *time* := *0*
Initial time value is 0 minutes
act26 : *activeInsulinCases* := ∅
No active insulin cases initially
act27 : *solutionObtained* := *FALSE*
Solution has not been obtained

end

Event *bolusCalc* ≙
Bolus calculator event for instances where the result is ≥ 0 and ≤ maxBolus

extends *bolusCalc*

any
s Bolus suggestion

where
grd1 : s ∈ $\mathbb{N}_1$
Bolus suggestion is greater than 0
grd2 : s ≤ maxBolus
The bolus solution suggestion is less than or equal to the maximum bolus dose
grd4 : s = (preBG − (targetRangeLower + ((targetRangeUpper − targetRangeLower)/2))) ∗ (isfI/isfBG) + carbs ∗ (carbRatioI/carbRatioC) − activeInsulin
The bolus suggestion equals the bolus calculation result
grd5 : inputsDefined = TRUE
The user inputs for the calculation must be done first
grd6 : *caseBase* ≠ ∅ ⇒ (∀*c*·*c* ∈ *caseBase* ∧ *time* − *caseTime*(*c*) > *0* ∧ *c* ≤ *active* ⇒ *c* ∈ *activeInsulinCases*)
All cases where active insulin is applicable must be a member of activeInsulinCases

then
act1 : bolusSuggestion := s
Sets the bolus suggestion value
act2 : *solutionObtained* := *TRUE*
A solution has been obtained

end

Event *bolusCalcNeg* ≙
Bolus calculator event for instances where s < 0

extends *bolusCalcNeg*

any
s Bolus suggestion

where
grd1 : s ∈ ℤ
Bolus suggestion is any integer

grd2 : $\texttt{s} \leq 0$
The bolus suggestion is less than 0
grd3 : $\texttt{s} = (\texttt{preBG} - (\texttt{targetRangeLower} + ((\texttt{targetRangeUpper} - \texttt{targetRangeLower})/2))) * (\texttt{isfI}/\texttt{isfBG}) + \texttt{carbs} * (\texttt{carbRatioI}/\texttt{carbRatioC}) - \texttt{activeInsulin}$
The bolus suggestion equals the bolus calculation result
grd4 : $\texttt{inputsDefined} = \texttt{TRUE}$
The user inputs for the calculation must be done first
grd5 : $caseBase \neq \varnothing \Rightarrow (\forall c \cdot c \in caseBase \land time - caseTime(c) > 0 \land c \leq active \Rightarrow c \in activeInsulinCases)$
All cases where active insulin is applicable must be a member of activeInsulinCases

then
act1 : $\texttt{bolusSuggestion} := 0$
Sets the bolus suggestion to 0 as negative values are not permitted.
act2 : $solutionObtained := TRUE$
A solution has been obtained
end

Event *bolusCalcMax* $\widehat{=}$
Bolus calculator event for instances s > maxBolus
extends *bolusCalcMax*
any
s Bolus suggestion
where
grd1 : $\texttt{s} \in \mathbb{N}$
grd2 : $\texttt{s} > \texttt{maxBolus}$
The bolus solution suggestion is greater than the maximum bolus dose
grd3 : $\texttt{s} = (\texttt{preBG} - (\texttt{targetRangeLower} + ((\texttt{targetRangeUpper} - \texttt{targetRangeLower})/2))) * (\texttt{isfI}/\texttt{isfBG}) + \texttt{carbs} * (\texttt{carbRatioI}/\texttt{carbRatioC}) - \texttt{activeInsulin}$
The bolus suggestion equals the bolus calculation result
grd4 : $\texttt{inputsDefined} = \texttt{TRUE}$
The user inputs for the calculation must be done first
grd5 : $caseBase \neq \varnothing \Rightarrow (\forall c \cdot c \in caseBase \land time - caseTime(c) > 0 \land c \leq active \Rightarrow c \in activeInsulinCases)$
All cases where active insulin is applicable must be a member of activeInsulinCases

then
act1 : $\texttt{bolusSuggestion} := \texttt{maxBolus}$
Sets the bolus suggestion to maxBolus as values greater than maxBolus are not permitted
act2 : $solutionObtained := TRUE$
A solution has been obtained
end

Event *defineInput* $\widehat{=}$
extends *defineInput*
any
c Planned carbohydrate intake
bg Preprandial blood glucose level (mmol/L)
t The time of the calculation
where
grd1 : $\texttt{c} \in \mathbb{N}$
Carbohydrate value must be greater than or equal to 0
grd2 : $\texttt{c} \leq 2400$
Carbohydrate value must be less than or equal to 240 grams
grd3 : $\texttt{bg} \in \mathbb{N}$
Blood glucose value must be greater than or equal to 0
grd4 : $\texttt{bg} \leq 554$
Carbohydrate value must be less than or equal to 55.4 mmol/L

grd5 : $t \in \mathbb{N}$
Time must be greater than or equal to 0 minutes

then

act1 : carbs := c
act2 : preBG := bg
act3 : inputsDefined := TRUE
The user inputs have been defined
act4 : $time := t$

end

Event *calcActiveInsulin* $\widehat{=}$

extends *calcActiveInsulin*

any

a Active insulin remaining from a previous case
c The caseID which active insulin is to be calculated for

where

grd1 : $a \in \mathbb{N}$
Active insulin must be greater than or equal to 0
grd2 : $c \in caseBase$
The caseID must exist in the case-base
grd3 : $c \notin activeInsulinCases$
The caseID must not already be accounted for in terms of active insulin
grd4 : $time - caseTime(c) > 0$
The current time must be after the previous case time
grd5 : $time - caseTime(c) \leq active$
The different between the current time and case time must be less than or equal to the active insulin duration time
grd6 : $a = activeInsulin + (caseUsedBolus(c) * (1 - ((time - caseTime(c))/active)))$
Active insulin calculation

then

act1 : $activeInsulin := a$
act2 : $activeInsulinCases := activeInsulinCases \cup \{c\}$
Adds the caseID into the cases where active insulin has already been accounted for

end

Event *retainCase* $\widehat{=}$

Retains a case in the case-base

any

c New case ID

where

grd1 : $solutionObtained = TRUE$
A solution must be obtained first
grd2 : $c \in \mathbb{N}_1$
The caseID must be greater than or equal to 0
grd3 : $c \notin caseBase$
The caseID must not already exist in the case-base
grd4 : $caseBase = \varnothing \Rightarrow c = 1$
If the case-base is empty, then the caseID is 0
grd5 : $caseBase \neq \varnothing \Rightarrow c = card(caseBase) + 1$
If the case-base is not empty then the caseID is the cardinality of the case-base + 1

then

act1 : $caseBase := caseBase \cup \{c\}$
Adds the caseID to the case-base
act2 : $caseCarbs := caseCarbs \cup \{c \mapsto carbs\}$
Maps the caseID to carbohydrates
act3 : $casePreBG := casePreBG \cup \{c \mapsto preBG\}$
Maps the caseID to the preprandial blood glucose reading
act4 : $caseUsedBolus := caseUsedBolus \cup \{c \mapsto bolusSuggestion\}$
Maps the caseID to the bolus suggestion
act5 : $caseTime := caseTime \cup \{c \mapsto time\}$
Maps the caseID to the time of the case

end

Event *removeCase* $\widehat{=}$

any

c The caseID to remove

where

grd1 : $c \in caseBase$

The caseID must be a member of case-base

then

act1 : $caseBase := caseBase \setminus \{c\}$

Removes the caseID from the case-base

act2 : $caseCarbs := \{c\} \lhd caseCarbs$

Removes the caseID to carbohydrate mapping

act3 : $casePreBG := \{c\} \lhd casePreBG$

Removes the caseID to preprandial blood glucose mapping

act4 : $caseUsedBolus := \{c\} \lhd caseUsedBolus$

Removes the caseID to bolus mapping

act5 : $caseTime := \{c\} \lhd caseTime$

Removes the caseID to case time mapping

act6 : $activeInsulinCases := activeInsulinCases \setminus \{c\}$

Removes the caseID from the cases active insulin cases if present

end

Event *defineSettings* $\widehat{=}$

extends *defineSettings*

any

ac Active insulin time

ba Daily basal dose

bgH High blood glucose warning threshold

bgL Low blood glucose warning threshold

crC Carbohydrate-to-insulin ratio: carbohydrates

crI Carbohydrate-to-insulin ratio: insulin units

heT Hyperglycaemia warning threshold

hoT Hypoglycaemia warning threshold

iBG Insulin sensitivity factor: blood glucose

iI Insulin sensitivity factor: insulin units

mB Maximum allowed bolus dose

tU Target blood glucose upper value

tL Target blood glucose lower value

b Revised bolus value, if the current state is invalidated by a change to maxBolus

where

grd1 : $ac \in \mathbb{N}$

grd2 : $ac \geq 900 \land ac \leq 4800$

grd3 : $ba \in \mathbb{N}$

grd4 : $ba \leq 990$

grd5 : $bgH \in \mathbb{N}$

grd6 : $bgH \geq 65 \land bgH \leq 195$

grd7 : $bgL \in \mathbb{N}$

grd8 : $bgL \geq 30 \land bgL \leq 55$

grd9 : $crC \in \mathbb{N}$

grd10 : $crC \geq 10 \land crC \leq 2400$

grd11 : $crI \in \mathbb{N}$

grd12 : $crI \geq 1 \land crI \leq 500$

grd13 : $heT \in \mathbb{N}$

grd14 : $heT \geq 100 \land heT \leq 195$

grd15 : $hoT \in \mathbb{N}$

grd16 : $hoT \geq 30 \land hoT \leq 50$

grd17 : $iBG \in \mathbb{N}$

grd18 : $iBG \geq 10 \land iBG \leq 554$

grd19 : $iI \in \mathbb{N}$

grd20 : $iI \geq 1 \land iI \leq 500$

grd21 : $mB \in \mathbb{N}$

grd22 : mB ≤ 500
grd23 : tU ∈ ℕ
grd24 : tU ≥ 55 ∧ tU ≤ 150
grd25 : tL ∈ ℕ
grd26 : tL ≥ 30 ∧ tL ≤ 80
grd27 : b ∈ ℕ
bolusSuggestion, required incase the new maxBolus violates the current state
grd28 : bolusSuggestion > mB ⇒ b = mB
If bolusSuggestion in the current state would violate the new maxBolus value, then amend the bolusSuggestion to maxBolus
grd29 : bolusSuggestion ≤ mB ⇒ b = bolusSuggestion
If bolusSuggestion in the current state does violate the new maxBolus value, then maintain the same value

then

act1 : active := ac
act2 : basal := ba
act3 : bgThresHigh := bgH
act4 : bgThresLow := bgL
act5 : carbRatioC := crC
act6 : carbRatioI := crI
act7 : hyperThreshold := heT
act8 : hypoThreshold := hoT
act9 : isfBG := iBG
act10 : isfI := iI
act11 : maxBolus := mB
act12 : targetRangeUpper := tU
act13 : targetRangeLower := tL
act14 : bolusSuggestion := b

end

END

Appendix D

Case-base reasoning retrieval statistical results

DM	k-NN	TS	BGRI	LBGI	HBGI	<TR (%)	>TR (%)	σ^2	σ	μ
Euc.	1	1	4.44	2.23	2.21	0.16	0.00	0.82	0.91	6.33
		2	4.40	2.22	2.17	0.15	0.00	0.81	0.90	6.33
		3	4.40	2.19	2.21	0.14	0.00	0.81	0.90	6.34
		4	4.43	2.21	2.22	0.16	0.00	0.82	0.90	6.34
		5	4.46	2.23	2.22	0.16	0.00	0.83	0.91	6.34
	2	1	4.44	2.22	2.22	0.14	0.00	0.82	0.90	6.34
		2	4.39	2.21	2.18	0.13	0.00	0.80	0.90	6.33
		3	4.40	2.19	2.21	0.14	0.00	0.81	0.90	6.34
		4	4.42	2.21	2.21	0.12	0.00	0.81	0.90	6.33
		5	4.44	2.23	2.21	0.11	0.00	0.82	0.91	6.33
	3	1	4.42	2.21	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.38	2.19	2.20	0.13	0.00	0.80	0.90	6.34
		3	4.40	2.20	2.20	0.14	0.00	0.81	0.90	6.34
		4	4.41	2.21	2.20	0.13	0.00	0.81	0.90	6.33
		5	4.42	2.21	2.21	0.11	0.00	0.81	0.90	6.34
	4	1	4.43	2.22	2.22	0.16	0.00	0.82	0.90	6.34
		2	4.38	2.19	2.19	0.13	0.00	0.80	0.89	6.33
		3	4.38	2.18	2.20	0.14	0.00	0.80	0.89	6.34
		4	4.41	2.21	2.20	0.12	0.00	0.81	0.90	6.33
		5	4.41	2.20	2.21	0.10	0.00	0.81	0.90	6.34
	5	1	4.44	2.22	2.22	0.17	0.00	0.82	0.90	6.34
		2	4.39	2.19	2.19	0.13	0.00	0.80	0.89	6.33
		3	4.38	2.19	2.19	0.14	0.00	0.80	0.89	6.33
		4	4.38	2.19	2.19	0.10	0.00	0.80	0.89	6.33
		5	4.39	2.20	2.19	0.11	0.00	0.80	0.90	6.34
Man.	1	1	4.43	2.21	2.22	0.16	0.00	0.81	0.90	6.34
		2	4.43	2.21	2.22	0.15	0.00	0.81	0.90	6.34
		3	4.43	2.21	2.22	0.15	0.00	0.81	0.90	6.34
		4	4.44	2.22	2.22	0.17	0.00	0.82	0.90	6.33
		5	4.45	2.24	2.21	0.18	0.00	0.82	0.91	6.33
	2	1	4.42	2.20	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.42	2.20	2.22	0.14	0.00	0.81	0.90	6.34
		3	4.42	2.20	2.22	0.15	0.00	0.81	0.90	6.34
		4	4.42	2.21	2.21	0.16	0.00	0.81	0.90	6.33
		5	4.44	2.23	2.21	0.15	0.00	0.81	0.90	6.33
	3	1	4.41	2.20	2.22	0.14	0.00	0.81	0.90	6.34
		2	4.42	2.21	2.21	0.14	0.00	0.81	0.90	6.34
		3	4.41	2.21	2.21	0.16	0.00	0.81	0.90	6.33
		4	4.42	2.22	2.21	0.16	0.00	0.81	0.90	6.33
		5	4.43	2.21	2.21	0.15	0.00	0.81	0.90	6.34
	4	1	4.41	2.20	2.21	0.15	0.00	0.81	0.90	6.34
		2	4.41	2.20	2.21	0.14	0.00	0.80	0.90	6.34
		3	4.41	2.21	2.21	0.16	0.00	0.81	0.90	6.33
		4	4.42	2.21	2.21	0.15	0.00	0.81	0.90	6.33
		5	4.42	2.20	2.21	0.15	0.00	0.81	0.90	6.33
	5	1	4.41	2.20	2.21	0.15	0.00	0.81	0.90	6.34
		2	4.41	2.20	2.21	0.14	0.00	0.81	0.90	6.34
		3	4.41	2.21	2.20	0.15	0.00	0.81	0.90	6.34
		4	4.42	2.21	2.21	0.15	0.00	0.81	0.90	6.34
		5	4.41	2.20	2.21	0.15	0.00	0.81	0.90	6.34

Table D.1: Chi-Squared retrieval statistics

DM	k-NN	TS	BGRI	LBGI	HBGI	<TR (%)	>TR (%)	σ^2	σ	μ
Euc.	1	1	4.44	2.22	2.21	0.16	0.00	0.82	0.90	6.33
		2	4.38	2.21	2.17	0.15	0.00	0.80	0.90	6.33
		3	4.40	2.20	2.20	0.14	0.00	0.81	0.90	6.34
		4	4.42	2.21	2.22	0.15	0.00	0.82	0.90	6.34
		5	4.46	2.25	2.21	0.15	0.00	0.83	0.91	6.33
	2	1	4.43	2.21	2.22	0.14	0.00	0.82	0.90	6.34
		2	4.38	2.20	2.18	0.13	0.00	0.80	0.89	6.33
		3	4.39	2.19	2.20	0.14	0.00	0.81	0.90	6.34
		4	4.40	2.19	2.21	0.11	0.00	0.81	0.90	6.34
		5	4.44	2.23	2.21	0.11	0.00	0.82	0.91	6.33
	3	1	4.43	2.21	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.37	2.19	2.19	0.13	0.00	0.80	0.89	6.33
		3	4.40	2.19	2.21	0.14	0.00	0.81	0.90	6.34
		4	4.41	2.21	2.20	0.13	0.00	0.81	0.90	6.33
		5	4.41	2.21	2.21	0.12	0.00	0.81	0.90	6.34
	4	1	4.43	2.21	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.38	2.19	2.18	0.13	0.00	0.80	0.89	6.33
		3	4.38	2.19	2.20	0.15	0.00	0.80	0.89	6.34
		4	4.39	2.19	2.20	0.12	0.00	0.80	0.90	6.33
		5	4.40	2.20	2.21	0.10	0.00	0.81	0.90	6.34
	5	1	4.44	2.21	2.22	0.17	0.00	0.81	0.90	6.34
		2	4.38	2.19	2.19	0.13	0.00	0.80	0.89	6.33
		3	4.37	2.18	2.19	0.13	0.00	0.80	0.89	6.34
		4	4.37	2.19	2.18	0.11	0.00	0.80	0.89	6.33
		5	4.39	2.20	2.19	0.11	0.00	0.80	0.90	6.33
Man.	1	1	4.43	2.21	2.22	0.16	0.00	0.81	0.90	6.34
		2	4.43	2.21	2.21	0.15	0.00	0.81	0.90	6.34
		3	4.43	2.21	2.22	0.15	0.00	0.81	0.90	6.34
		4	4.44	2.22	2.22	0.17	0.00	0.82	0.90	6.33
		5	4.45	2.23	2.21	0.16	0.00	0.82	0.91	6.33
	2	1	4.42	2.20	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.41	2.20	2.21	0.14	0.00	0.81	0.90	6.34
		3	4.42	2.20	2.22	0.15	0.00	0.81	0.90	6.34
		4	4.43	2.21	2.21	0.15	0.00	0.81	0.90	6.33
		5	4.44	2.23	2.21	0.14	0.00	0.82	0.90	6.33
	3	1	4.41	2.20	2.22	0.14	0.00	0.81	0.90	6.34
		2	4.42	2.21	2.21	0.13	0.00	0.81	0.90	6.34
		3	4.42	2.21	2.21	0.15	0.00	0.81	0.90	6.34
		4	4.43	2.22	2.21	0.14	0.00	0.81	0.90	6.33
		5	4.42	2.21	2.21	0.13	0.00	0.81	0.90	6.34
	4	1	4.41	2.20	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.41	2.20	2.21	0.14	0.00	0.81	0.90	6.34
		3	4.41	2.21	2.21	0.14	0.00	0.81	0.90	6.34
		4	4.41	2.21	2.21	0.14	0.00	0.81	0.90	6.33
		5	4.42	2.21	2.21	0.15	0.00	0.81	0.90	6.33
	5	1	4.41	2.20	2.21	0.15	0.00	0.81	0.90	6.34
		2	4.41	2.21	2.21	0.14	0.00	0.81	0.90	6.34
		3	4.41	2.21	2.21	0.14	0.00	0.81	0.90	6.34
		4	4.41	2.20	2.21	0.15	0.00	0.81	0.90	6.34
		5	4.42	2.21	2.21	0.15	0.00	0.81	0.90	6.33

Table D.2: Information Gain retrieval statistics

DM	k-NN	TS	BGRI	LBGI	HBGI	<TR (%)	>TR (%)	σ^2	σ	μ
Euc.	1	1	4.44	2.23	2.22	0.16	0.00	0.82	0.91	6.34
		2	4.39	2.21	2.18	0.14	0.00	0.80	0.90	6.33
		3	4.40	2.20	2.21	0.15	0.00	0.81	0.90	6.34
		4	4.43	2.21	2.22	0.15	0.00	0.82	0.90	6.34
		5	4.45	2.23	2.22	0.15	0.00	0.83	0.91	6.34
	2	1	4.44	2.21	2.23	0.15	0.00	0.82	0.91	6.34
		2	4.38	2.20	2.18	0.12	0.00	0.80	0.89	6.33
		3	4.40	2.19	2.20	0.14	0.00	0.81	0.90	6.34
		4	4.42	2.21	2.21	0.11	0.00	0.81	0.90	6.33
		5	4.43	2.21	2.21	0.12	0.00	0.82	0.90	6.33
	3	1	4.43	2.21	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.38	2.19	2.19	0.14	0.00	0.80	0.90	6.33
		3	4.39	2.18	2.21	0.14	0.00	0.80	0.90	6.34
		4	4.41	2.21	2.20	0.12	0.00	0.81	0.90	6.33
		5	4.40	2.19	2.21	0.11	0.00	0.81	0.90	6.34
	4	1	4.43	2.22	2.22	0.16	0.00	0.82	0.90	6.34
		2	4.38	2.19	2.19	0.12	0.00	0.80	0.89	6.33
		3	4.37	2.18	2.20	0.14	0.00	0.80	0.89	6.34
		4	4.39	2.19	2.20	0.12	0.00	0.80	0.90	6.33
		5	4.41	2.20	2.21	0.12	0.00	0.81	0.90	6.34
	5	1	4.44	2.22	2.22	0.17	0.00	0.82	0.90	6.34
		2	4.39	2.19	2.19	0.14	0.00	0.80	0.89	6.33
		3	4.38	2.18	2.19	0.13	0.00	0.80	0.89	6.34
		4	4.36	2.18	2.18	0.11	0.00	0.79	0.89	6.33
		5	4.39	2.20	2.19	0.12	0.00	0.80	0.90	6.34
Man.	1	1	4.43	2.20	2.23	0.16	0.00	0.81	0.90	6.34
		2	4.43	2.21	2.21	0.15	0.00	0.81	0.90	6.34
		3	4.43	2.21	2.22	0.16	0.00	0.81	0.90	6.34
		4	4.43	2.22	2.22	0.16	0.00	0.81	0.90	6.33
		5	4.44	2.22	2.22	0.16	0.00	0.82	0.90	6.33
	2	1	4.42	2.20	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.41	2.20	2.21	0.14	0.00	0.81	0.90	6.34
		3	4.42	2.21	2.21	0.15	0.00	0.81	0.90	6.34
		4	4.42	2.21	2.21	0.15	0.00	0.81	0.90	6.33
		5	4.43	2.22	2.21	0.14	0.00	0.81	0.90	6.33
	3	1	4.41	2.20	2.22	0.14	0.00	0.81	0.90	6.34
		2	4.42	2.21	2.21	0.13	0.00	0.81	0.90	6.34
		3	4.41	2.20	2.21	0.14	0.00	0.81	0.90	6.34
		4	4.43	2.22	2.21	0.14	0.00	0.81	0.90	6.33
		5	4.43	2.21	2.21	0.12	0.00	0.81	0.90	6.33
	4	1	4.41	2.20	2.22	0.14	0.00	0.81	0.90	6.34
		2	4.41	2.20	2.21	0.13	0.00	0.81	0.90	6.34
		3	4.41	2.21	2.21	0.15	0.00	0.81	0.90	6.34
		4	4.41	2.21	2.20	0.14	0.00	0.81	0.90	6.33
		5	4.42	2.21	2.21	0.14	0.00	0.81	0.90	6.33
	5	1	4.41	2.20	2.22	0.14	0.00	0.81	0.90	6.34
		2	4.41	2.21	2.21	0.14	0.00	0.81	0.90	6.34
		3	4.41	2.20	2.20	0.14	0.00	0.81	0.90	6.34
		4	4.42	2.21	2.20	0.14	0.00	0.81	0.90	6.34
		5	4.43	2.21	2.21	0.16	0.00	0.81	0.90	6.34

Table D.3: Gain Ratio retrieval statistics

DM	k-NN	TS	BGRI	LBGI	HBGI	<TR (%)	>TR (%)	σ^2	σ	μ
Euc.	1	1	4.45	2.25	2.20	0.15	0.00	0.82	0.91	6.33
		2	4.49	2.26	2.23	0.23	0.00	0.85	0.92	6.34
		3	4.59	2.30	2.28	0.23	0.00	0.89	0.94	6.35
		4	4.62	2.31	2.31	0.21	0.00	0.90	0.95	6.35
		5	4.67	2.34	2.33	0.21	0.00	0.92	0.96	6.36
	2	1	4.43	2.24	2.19	0.13	0.00	0.81	0.90	6.33
		2	4.39	2.21	2.18	0.16	0.00	0.81	0.90	6.33
		3	4.43	2.20	2.23	0.13	0.00	0.83	0.91	6.34
		4	4.40	2.19	2.21	0.10	0.00	0.82	0.90	6.34
		5	4.39	2.19	2.20	0.12	0.00	0.82	0.90	6.35
	3	1	4.40	2.23	2.18	0.12	0.00	0.80	0.90	6.33
		2	4.40	2.20	2.19	0.14	0.00	0.81	0.90	6.33
		3	4.34	2.14	2.19	0.09	0.00	0.79	0.89	6.33
		4	4.34	2.16	2.18	0.08	0.00	0.79	0.89	6.33
		5	4.28	2.13	2.15	0.04	0.00	0.78	0.88	6.35
	4	1	4.41	2.23	2.18	0.12	0.00	0.81	0.90	6.33
		2	4.35	2.19	2.16	0.14	0.00	0.79	0.89	6.33
		3	4.31	2.13	2.18	0.07	0.00	0.78	0.88	6.33
		4	4.29	2.13	2.15	0.11	0.00	0.77	0.88	6.33
		5	4.24	2.11	2.13	0.03	0.00	0.77	0.87	6.34
	5	1	4.40	2.22	2.17	0.13	0.00	0.80	0.90	6.33
		2	4.34	2.18	2.16	0.11	0.00	0.79	0.89	6.33
		3	4.29	2.13	2.16	0.07	0.00	0.78	0.88	6.33
		4	4.26	2.11	2.15	0.06	0.00	0.76	0.87	6.32
		5	4.22	2.09	2.13	0.03	0.00	0.76	0.87	6.34
Man.	1	1	4.46	2.23	2.23	0.17	0.00	0.82	0.91	6.34
		2	4.42	2.23	2.19	0.18	0.00	0.81	0.90	6.33
		3	4.46	2.24	2.22	0.17	0.00	0.83	0.91	6.33
		4	4.44	2.22	2.22	0.18	0.00	0.83	0.91	6.34
		5	4.50	2.27	2.23	0.17	0.00	0.85	0.92	6.35
	2	1	4.46	2.23	2.23	0.15	0.00	0.82	0.91	6.34
		2	4.40	2.21	2.19	0.15	0.00	0.81	0.90	6.33
		3	4.42	2.21	2.21	0.13	0.00	0.82	0.90	6.34
		4	4.38	2.19	2.19	0.13	0.00	0.80	0.90	6.33
		5	4.39	2.21	2.18	0.11	0.00	0.81	0.90	6.34
	3	1	4.45	2.23	2.22	0.16	0.00	0.82	0.91	6.34
		2	4.38	2.20	2.19	0.13	0.00	0.80	0.89	6.33
		3	4.39	2.18	2.20	0.11	0.00	0.80	0.90	6.34
		4	4.37	2.19	2.18	0.12	0.00	0.80	0.89	6.33
		5	4.33	2.16	2.17	0.08	0.00	0.79	0.89	6.34
	4	1	4.45	2.23	2.22	0.15	0.00	0.82	0.90	6.34
		2	4.38	2.19	2.18	0.12	0.00	0.80	0.89	6.33
		3	4.38	2.18	2.20	0.11	0.00	0.80	0.89	6.33
		4	4.34	2.18	2.17	0.12	0.00	0.79	0.89	6.32
		5	4.32	2.16	2.16	0.08	0.00	0.78	0.89	6.34
	5	1	4.43	2.22	2.21	0.15	0.00	0.81	0.90	6.34
		2	4.37	2.19	2.18	0.11	0.00	0.79	0.89	6.33
		3	4.37	2.18	2.19	0.11	0.00	0.80	0.89	6.33
		4	4.33	2.16	2.17	0.11	0.00	0.78	0.89	6.33
		5	4.30	2.15	2.15	0.06	0.00	0.78	0.88	6.34

Table D.4: One Rule retrieval statistics

DM	k-NN	TS	BGRI	LBGI	HBGI	<TR (%)	>TR (%)	σ^2	σ	μ
Euc.	1	1	4.44	2.22	2.22	0.19	0.00	0.82	0.90	6.34
		2	4.41	2.22	2.19	0.16	0.00	0.81	0.90	6.33
		3	4.45	2.25	2.20	0.17	0.00	0.82	0.91	6.33
		4	4.45	2.25	2.20	0.17	0.00	0.82	0.91	6.33
		5	4.45	2.23	2.22	0.16	0.00	0.82	0.91	6.34
	2	1	4.44	2.22	2.22	0.16	0.00	0.82	0.90	6.34
		2	4.41	2.22	2.19	0.13	0.00	0.81	0.90	6.33
		3	4.42	2.23	2.20	0.15	0.00	0.81	0.90	6.33
		4	4.44	2.23	2.21	0.14	0.00	0.82	0.90	6.33
		5	4.42	2.21	2.21	0.13	0.00	0.81	0.90	6.33
	3	1	4.43	2.21	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.40	2.20	2.20	0.14	0.00	0.81	0.90	6.33
		3	4.43	2.22	2.21	0.13	0.00	0.81	0.90	6.34
		4	4.40	2.20	2.21	0.13	0.00	0.81	0.90	6.34
		5	4.42	2.21	2.21	0.12	0.00	0.81	0.90	6.34
	4	1	4.43	2.21	2.22	0.17	0.00	0.81	0.90	6.34
		2	4.39	2.21	2.19	0.15	0.00	0.80	0.90	6.33
		3	4.42	2.22	2.20	0.15	0.00	0.81	0.90	6.34
		4	4.41	2.20	2.20	0.14	0.00	0.81	0.90	6.34
		5	4.41	2.20	2.21	0.12	0.00	0.81	0.90	6.33
	5	1	4.43	2.21	2.22	0.14	0.00	0.81	0.90	6.34
		2	4.39	2.20	2.19	0.14	0.00	0.80	0.90	6.33
		3	4.42	2.22	2.20	0.14	0.00	0.81	0.90	6.34
		4	4.40	2.20	2.20	0.12	0.00	0.81	0.90	6.33
		5	4.40	2.21	2.19	0.11	0.00	0.80	0.90	6.33
Man.	1	1	4.43	2.21	2.22	0.18	0.00	0.81	0.90	6.34
		2	4.43	2.21	2.22	0.20	0.00	0.81	0.90	6.34
		3	4.46	2.23	2.22	0.22	0.00	0.82	0.91	6.34
		4	4.45	2.23	2.22	0.20	0.00	0.82	0.91	6.34
		5	4.45	2.23	2.22	0.17	0.00	0.82	0.90	6.33
	2	1	4.43	2.21	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.43	2.21	2.22	0.14	0.00	0.81	0.90	6.34
		3	4.44	2.23	2.22	0.17	0.00	0.82	0.90	6.33
		4	4.43	2.22	2.21	0.16	0.00	0.81	0.90	6.33
		5	4.44	2.22	2.22	0.16	0.00	0.82	0.90	6.33
	3	1	4.43	2.21	2.22	0.17	0.00	0.81	0.90	6.34
		2	4.43	2.21	2.22	0.14	0.00	0.81	0.90	6.34
		3	4.44	2.22	2.21	0.16	0.00	0.81	0.90	6.33
		4	4.44	2.22	2.22	0.15	0.00	0.81	0.90	6.33
		5	4.44	2.22	2.21	0.14	0.00	0.81	0.90	6.33
	4	1	4.44	2.22	2.22	0.16	0.00	0.81	0.90	6.34
		2	4.43	2.21	2.21	0.14	0.00	0.81	0.90	6.33
		3	4.43	2.22	2.21	0.16	0.00	0.81	0.90	6.33
		4	4.43	2.22	2.21	0.15	0.00	0.81	0.90	6.33
		5	4.44	2.22	2.22	0.15	0.00	0.82	0.90	6.33
	5	1	4.44	2.22	2.22	0.16	0.00	0.81	0.90	6.34
		2	4.42	2.21	2.21	0.14	0.00	0.81	0.90	6.34
		3	4.43	2.22	2.21	0.15	0.00	0.81	0.90	6.33
		4	4.44	2.22	2.22	0.14	0.00	0.81	0.90	6.33
		5	4.44	2.22	2.22	0.15	0.00	0.82	0.90	6.34

Table D.5: RELIEF-F retrieval statistics

DM	k-NN	TS	BGRI	LBGI	HBGI	<TR (%)	>TR (%)	σ^2	σ	μ
Euc.	1	1	4.44	2.22	2.22	0.16	0.00	0.82	0.91	6.34
		2	4.39	2.21	2.18	0.14	0.00	0.80	0.90	6.33
		3	4.40	2.20	2.21	0.15	0.00	0.81	0.90	6.34
		4	4.42	2.20	2.22	0.14	0.00	0.82	0.90	6.34
		5	4.45	2.23	2.22	0.15	0.00	0.83	0.91	6.34
	2	1	4.44	2.21	2.22	0.14	0.00	0.82	0.90	6.34
		2	4.38	2.20	2.18	0.13	0.00	0.80	0.90	6.33
		3	4.39	2.19	2.20	0.15	0.00	0.81	0.90	6.34
		4	4.41	2.20	2.21	0.11	0.00	0.81	0.90	6.34
		5	4.43	2.22	2.21	0.11	0.00	0.82	0.90	6.33
	3	1	4.42	2.21	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.38	2.19	2.19	0.13	0.00	0.80	0.89	6.33
		3	4.39	2.19	2.20	0.14	0.00	0.81	0.90	6.34
		4	4.40	2.21	2.20	0.12	0.00	0.81	0.90	6.33
		5	4.41	2.20	2.21	0.11	0.00	0.81	0.90	6.34
	4	1	4.43	2.21	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.38	2.19	2.19	0.12	0.00	0.80	0.89	6.33
		3	4.38	2.18	2.20	0.14	0.00	0.80	0.90	6.34
		4	4.39	2.19	2.20	0.12	0.00	0.80	0.90	6.33
		5	4.40	2.20	2.21	0.11	0.00	0.81	0.90	6.34
	5	1	4.44	2.22	2.22	0.17	0.00	0.82	0.90	6.34
		2	4.38	2.19	2.19	0.14	0.00	0.80	0.89	6.33
		3	4.37	2.18	2.19	0.13	0.00	0.80	0.89	6.34
		4	4.37	2.18	2.18	0.10	0.00	0.79	0.89	6.33
		5	4.39	2.20	2.19	0.11	0.00	0.80	0.90	6.34
Man.	1	1	4.43	2.21	2.22	0.16	0.00	0.81	0.90	6.34
		2	4.42	2.21	2.22	0.15	0.00	0.81	0.90	6.34
		3	4.43	2.21	2.22	0.16	0.00	0.81	0.90	6.34
		4	4.43	2.22	2.22	0.17	0.00	0.82	0.90	6.33
		5	4.45	2.23	2.22	0.16	0.00	0.82	0.91	6.33
	2	1	4.42	2.20	2.22	0.15	0.00	0.81	0.90	6.34
		2	4.41	2.20	2.21	0.14	0.00	0.81	0.90	6.34
		3	4.42	2.20	2.22	0.15	0.00	0.81	0.90	6.34
		4	4.42	2.21	2.21	0.15	0.00	0.81	0.90	6.33
		5	4.44	2.22	2.22	0.14	0.00	0.82	0.90	6.33
	3	1	4.41	2.20	2.22	0.14	0.00	0.81	0.90	6.34
		2	4.42	2.21	2.21	0.13	0.00	0.81	0.90	6.34
		3	4.41	2.20	2.21	0.14	0.00	0.81	0.90	6.34
		4	4.43	2.22	2.21	0.14	0.00	0.81	0.90	6.33
		5	4.42	2.21	2.21	0.12	0.00	0.81	0.90	6.34
	4	1	4.41	2.20	2.22	0.14	0.00	0.81	0.90	6.34
		2	4.41	2.21	2.21	0.14	0.00	0.81	0.90	6.34
		3	4.41	2.21	2.21	0.14	0.00	0.81	0.90	6.34
		4	4.41	2.21	2.21	0.14	0.00	0.81	0.90	6.33
		5	4.42	2.21	2.21	0.14	0.00	0.81	0.90	6.33
	5	1	4.41	2.20	2.21	0.15	0.00	0.81	0.90	6.34
		2	4.41	2.21	2.21	0.14	0.00	0.81	0.90	6.34
		3	4.41	2.20	2.20	0.14	0.00	0.81	0.90	6.34
		4	4.41	2.21	2.20	0.15	0.00	0.81	0.90	6.34
		5	4.43	2.21	2.21	0.16	0.00	0.81	0.90	6.33

Table D.6: Symmetrical Uncertainty retrieval statistics

DM	k-NN	TS	BGRI	LBGI	HBGI	<TR (%)	>TR (%)	σ^2	σ	μ
Euc.	1	1	4.50	2.28	2.22	0.21	0.00	0.84	0.92	6.33
		2	4.50	2.29	2.20	0.21	0.00	0.84	0.92	6.32
		3	4.56	2.33	2.23	0.19	0.00	0.87	0.93	6.31
		4	4.82	2.47	2.35	0.35	0.03	0.97	0.98	6.34
		5	5.35	2.77	2.57	0.90	0.11	1.20	1.09	6.37
	2	1	4.54	2.37	2.17	0.21	0.00	0.84	0.92	6.26
		2	4.56	2.44	2.12	0.23	0.00	0.84	0.92	6.21
		3	4.57	2.46	2.11	0.24	0.00	0.85	0.92	6.20
		4	4.60	2.47	2.13	0.25	0.00	0.86	0.93	6.20
		5	4.73	2.52	2.22	0.58	0.01	0.94	0.97	6.26
	3	1	4.46	2.24	2.23	0.10	0.00	0.82	0.91	6.34
		2	4.41	2.24	2.18	0.11	0.00	0.81	0.90	6.32
		3	4.40	2.22	2.18	0.11	0.00	0.81	0.90	6.31
		4	4.33	2.17	2.16	0.10	0.00	0.79	0.89	6.31
		5	4.41	2.21	2.20	0.27	0.01	0.84	0.92	6.35
	4	1	4.48	2.30	2.18	0.13	0.00	0.83	0.91	6.29
		2	4.44	2.30	2.14	0.15	0.00	0.81	0.90	6.27
		3	4.39	2.28	2.12	0.17	0.00	0.80	0.89	6.26
		4	4.35	2.25	2.11	0.16	0.00	0.79	0.89	6.25
		5	4.33	2.20	2.13	0.14	0.00	0.81	0.90	6.30
	5	1	4.46	2.24	2.22	0.10	0.00	0.82	0.91	6.33
		2	4.37	2.20	2.17	0.10	0.00	0.80	0.89	6.32
		3	4.33	2.18	2.15	0.08	0.00	0.79	0.89	6.32
		4	4.25	2.12	2.13	0.06	0.00	0.77	0.88	6.31
		5	4.22	2.09	2.13	0.11	0.00	0.78	0.88	6.35
Man.	1	1	4.50	2.27	2.23	0.23	0.00	0.84	0.92	6.33
		2	4.44	2.23	2.21	0.17	0.00	0.83	0.91	6.33
		3	4.58	2.30	2.27	0.25	0.00	0.88	0.94	6.33
		4	4.83	2.50	2.33	0.50	0.00	0.97	0.98	6.32
		5	4.92	2.55	2.36	0.50	0.09	1.03	1.01	6.33
	2	1	4.53	2.36	2.17	0.24	0.00	0.84	0.92	6.27
		2	4.53	2.39	2.14	0.24	0.00	0.84	0.91	6.23
		3	4.54	2.44	2.11	0.25	0.00	0.84	0.92	6.21
		4	4.67	2.51	2.16	0.31	0.00	0.89	0.94	6.21
		5	4.58	2.43	2.15	0.31	0.00	0.87	0.93	6.23
	3	1	4.46	2.23	2.23	0.12	0.00	0.82	0.91	6.34
		2	4.38	2.19	2.19	0.12	0.00	0.80	0.90	6.33
		3	4.38	2.22	2.16	0.13	0.00	0.80	0.90	6.32
		4	4.37	2.20	2.17	0.15	0.00	0.81	0.90	6.32
		5	4.27	2.13	2.14	0.10	0.00	0.78	0.88	6.33
	4	1	4.49	2.29	2.20	0.16	0.00	0.83	0.91	6.30
		2	4.42	2.27	2.15	0.15	0.00	0.81	0.90	6.28
		3	4.38	2.27	2.11	0.20	0.00	0.80	0.89	6.27
		4	4.35	2.24	2.11	0.14	0.00	0.79	0.89	6.26
		5	4.25	2.16	2.09	0.11	0.00	0.77	0.88	6.28
	5	1	4.45	2.23	2.22	0.12	0.00	0.82	0.91	6.33
		2	4.34	2.16	2.17	0.10	0.00	0.79	0.89	6.33
		3	4.29	2.17	2.13	0.09	0.00	0.78	0.88	6.32
		4	4.25	2.14	2.11	0.09	0.00	0.76	0.87	6.30
		5	4.14	2.04	2.10	0.04	0.00	0.74	0.86	6.33

Table D.7: No feature weighting retrieval statistics

Appendix E

Unit testing

Class.Method
String Settings.setUnit(String value)

Description
Modifies the blood glucose unit setting.

Initial value
mmol/L

Valid outputs
mg/dL, mmol/L

input	expected output	actual output	result
mg/dL	mg/dL	mg/dL	pass
mmol/L	mmol/L	mmol/L	pass
obtko	mmol/L	mmol/L	pass
oib6y	mmol/L	mmol/L	pass
orhyi	mmol/L	mmol/L	pass
o5byd	mmol/L	mmol/L	pass
odiry	mmol/L	mmol/L	pass

Table E.1: Settings.setUnit() unit test results

Class.Method
double Settings.setBasal(double value)

Description
Modifies the normal daily basal insulin dose in insulin units.

Initial value
0.0

Valid outputs
$[0.0, 99.0]$

input	expected output	actual output	result
0.0	0.0	0.0	pass
99.0	99.0	99.0	pass
95.46	95.46	95.46	pass
66.71	66.71	66.71	pass
6.35	6.35	6.35	pass
68.12	68.12	68.12	pass
43.83	43.83	43.83	pass
-0.1	0.0	0.0	pass
99.1	0.0	0.0	pass
-77.23	0.0	0.0	pass
-105.05	0.0	0.0	pass
457.73	0.0	0.0	pass
-384.88	0.0	0.0	pass

Table E.2: Settings.setBasal() unit test results

Class.Method
double Settings.setMaxBolusLimit(double value)

Description
Modifies the maximum allowed bolus dose in insulin units.

Initial value
0.0

Valid outputs
$[0.0, 50.0]$

input	expected output	actual output	result
0.0	0.0	0.0	pass
50.0	50.0	50.0	pass
39.52	39.52	39.52	pass
45.16	45.16	45.16	pass
33.02	33.02	33.02	pass
49.22	49.22	49.22	pass
4.19	4.19	4.19	pass
-0.1	0.0	0.0	pass
50.1	0.0	0.0	pass
252.09	0.0	0.0	pass
-79.75	0.0	0.0	pass
-290.37	0.0	0.0	pass
187.03	0.0	0.0	pass

Table E.3: Settings.setMaxBolusLimit() unit test results

Class.Method
int Settings.setCarbsToInsulin(int value)

Description
Modifies the carbohydrates in grams corrected by one insulin unit.

Initial value
12

Valid outputs
$[1, 240]$

input	expected output	actual output	result
1	1	1	pass
240	240	240	pass
3	3	3	pass
233	233	233	pass
28	28	28	pass
140	140	140	pass
86	86	86	pass
0	12	12	pass
241	12	12	pass
666	12	12	pass
998	12	12	pass
-976	12	12	pass
-941	12	12	pass

Table E.4: Settings.setCarbsToInsulin() unit test results

Class.Method
double Settings.setInsulinSensitivity(double value)

Description
Modifies the decrease in blood glucose mmol/L per one insulin unit.

Initial value
2.0

Valid outputs
$[0.1, 55.4]$

input	expected output	actual output	result
0.1	0.1	0.1	pass
55.4	55.4	55.4	pass
13.62	13.62	13.62	pass
7.22	7.22	7.22	pass
30.59	30.59	30.59	pass
21.03	21.03	21.03	pass
18.09	18.09	18.09	pass
0.0	2.0	2.0	pass
55.5	2.0	2.0	pass
292.12	2.0	2.0	pass
71.86	2.0	2.0	pass
295.38	2.0	2.0	pass
173.38	2.0	2.0	pass

Table E.5: Settings.setInsulinSensitivity() unit test results

Class.Method
double Settings.setTargetRangeUpper(double value)

Description
Modifies the upper value of the target blood glucose range in mmol/L.

Initial value
8.0

Valid outputs
$[5.5, 15.0]$

input	expected output	actual output	result
5.5	5.5	5.5	pass
15.0	15.0	15.0	pass
9.34	9.34	9.34	pass
13.61	13.61	13.61	pass
8.35	8.35	8.35	pass
7.0	7.0	7.0	pass
11.01	11.01	11.01	pass
5.4	8.0	8.0	pass
15.1	8.0	8.0	pass
-3.11	8.0	8.0	pass
-128.68	8.0	8.0	pass
416.8	8.0	8.0	pass
-793.18	8.0	8.0	pass

Table E.6: Settings.setTargetRangeUpper() unit test results

Class.Method
double Settings.setTargetRangeLower(double value)

Description
Modifies the lower value of the target blood glucose range in mmol/L.

Initial value
4.0

Valid outputs
$[3.0, 8.0]$

input	expected output	actual output	result
3.0	3.0	3.0	pass
8.0	8.0	8.0	pass
4.02	4.02	4.02	pass
6.73	6.73	6.73	pass
3.82	3.82	3.82	pass
7.93	7.93	7.93	pass
3.91	3.91	3.91	pass
2.9	4.0	4.0	pass
8.1	4.0	4.0	pass
212.82	4.0	4.0	pass
107.85	4.0	4.0	pass
87.19	4.0	4.0	pass
-313.91	4.0	4.0	pass

Table E.7: Settings.setTargetRangeLower() unit test results

Class.Method
double Settings.setTargetBG(double targetRangeLower, double targetRangeHigher)

Description
Defines the target blood glucose level in mmol/L from the user's defined lower and upper target range values in mmol/L.

Initial value
6.0

Precondition
$targetRangeLower \leq targetRangeUpper$

Valid outputs
$[targetRangeLower, targetRangeUpper]$
$value = targetRangeLower + (targetRangeUpper - targetRangeLower \div 2)$

inputs		expected output	actual output	result
target range lower	target range upper			
5.4	5.6	5.5	5.5	pass
5	10	7.5	7.5	pass
4	7	5.5	5.5	pass
6	7.4	6.7	6.7	pass
3.5	10	6.75	6.75	pass
5	5	5	5	pass
5.4	5.3	6.0	6.0	pass
8	5.5	6.0	6.0	pass
8	7.9	6.0	6.0	pass

Table E.8: Settings.setTargetBG() unit test results

Class.Method
double Settings.setHighBGThreshold(double value)

Description
Modifies the high blood glucose warning level in mmol/L.

Initial value
16.5

Valid outputs
$[6.5, 19.5]$

input	expected output	actual output	result
6.5	6.5	6.5	pass
19.5	19.5	19.5	pass
13.64	13.64	13.64	pass
9.06	9.06	9.06	pass
14.47	14.47	14.47	pass
10.32	10.32	10.32	pass
17.58	17.58	17.58	pass
6.4	16.5	16.5	pass
19.6	16.5	16.5	pass
561.48	16.5	16.5	pass
-229.7	16.5	16.5	pass
-375.67	16.5	16.5	pass
5.64	16.5	16.5	pass

Table E.9: Settings.setHighBGThreshold() unit test results

Class.Method
double Settings.setLowBGThreshold(double value)

Description
Modifies the low blood glucose warning level in mmol/L.

Initial value
4.0

Valid outputs
$[3.0, 5.5]$

input	expected output	actual output	result
3.0	3.0	3.0	pass
5.5	5.5	5.5	pass
4.65	4.65	4.65	pass
3.77	3.77	3.77	pass
4.89	4.89	4.89	pass
3.44	3.44	3.44	pass
3.54	3.54	3.54	pass
2.9	4.0	4.0	pass
5.6	4.0	4.0	pass
-202.48	4.0	4.0	pass
259.68	4.0	4.0	pass
690.36	4.0	4.0	pass
362.2	4.0	4.0	pass

Table E.10: Settings.setLowBGThreshold() unit test results

Class.Method
double Settings.setHyperThreshold(double value)

Description
Modifies the hyperglycaemia warning level in mmol/L.

Initial value
16.5

Valid outputs
$[10.0, 19.5]$

input	expected output	actual output	result
10.0	10.0	10.0	pass
19.5	19.5	19.5	pass
14.61	14.61	14.61	pass
17.19	17.19	17.19	pass
13.08	13.08	13.08	pass
14.91	14.91	14.91	pass
12.92	12.92	12.92	pass
9.9	16.5	16.5	pass
19.6	16.5	16.5	pass
394.11	16.5	16.5	pass
229.76	16.5	16.5	pass
-114.18	16.5	16.5	pass
144.17	16.5	16.5	pass

Table E.11: Settings.setHyperThreshold() unit test results

Class.Method
double Settings.setHypoThreshold(double value)

Description
Modifies the hypoglycaemia warning level in mmol/L.

Initial value
4.0

Valid outputs
$[3.0, 5.0]$

input	expected output	actual output	result
3.0	3.0	3.0	pass
5.0	5.0	5.0	pass
3.37	3.37	3.37	pass
4.62	4.62	4.62	pass
3.98	3.98	3.98	pass
3.69	3.69	3.69	pass
4.57	4.57	4.57	pass
2.9	4.0	4.0	pass
5.1	4.0	4.0	pass
-12.0	4.0	4.0	pass
125.41	4.0	4.0	pass
-192.96	4.0	4.0	pass
166.54	4.0	4.0	pass

Table E.12: Settings.setHypoThreshold() unit test results

Class.Method
int Settings.isValidCarbInput(int value)

Description
Validates a carbohydrate input in grams. Returns -1 if out of range [0,240].

Valid outputs
$[0, 240]$

input	expected output	actual output	result
0	0	0	pass
240	240	240	pass
202	202	202	pass
145	145	145	pass
105	105	105	pass
226	226	226	pass
73	73	73	pass
-1	-1	-1	pass
241	-1	-1	pass
-428	-1	-1	pass
-164	-1	-1	pass
310	-1	-1	pass
-779	-1	-1	pass

Table E.13: Settings.isValidCarbInput() unit test results

Class.Method
double Settings.validateBolus(double value)

Description
Ensures a bolus dose suggestion is not negative and is less than or equal to the maximum allowed bolus dose in insulin units. Output is rounded to the nearest half (#.0 or #.5).

Valid outputs
$[0.0, 15.0]$

input	expected output	actual output	result
3.55	3.5	3.5	pass
12.5	12.5	12.5	pass
9.3	9.5	9.5	pass
1.23	1.0	1.0	pass
11.38	11.5	11.5	pass
-0.1	0.0	0.0	pass
15.1	15.0	15.0	pass
74.27	15.0	15.0	pass
83.28	15.0	15.0	pass
-60.0	0.0	0.0	pass
140.09	15.0	15.0	pass

Table E.14: Settings.validateBolus() unit test results

Class.Method
double Settings.validateIOB(double value)

Description
Ensures the insulin on board is not negative.

Valid outputs
$output \geq 0.0$

input	expected output	actual output	result
0.0	0.0	0.0	pass
5.78	5.78	5.78	pass
13.51	13.51	13.51	pass
7.22	7.22	7.22	pass
0.9	0.9	0.9	pass
6.65	6.65	6.65	pass
1.61	1.61	1.61	pass
-0.1	0.0	0.0	pass
-236.17	0.0	0.0	pass
-283.32	0.0	0.0	pass
284.54	284.54	284.54	pass
104.12	104.12	104.12	pass

Table E.15: Settings.validateIOB() unit test results

Class.Method

double Features.normalise(double value, double min, double max)

Description

Normalises a feature-value to [0,1] based on the min and max feature values.
Expected output calculated manually.
All outputs rounded to two decimal places in the results table.

Preconditions

$min < max$
$min \leq value \leq max$

Valid outputs

Preconditions met: [0.00, 1.00]
Preconditions violated: -1.00

inputs			expected output	actual output	result
value	min	max			
0.00	0.00	100.00	0.00	0.00	pass
100.00	0.00	100.00	1.00	1.00	pass
30.03	5.45	90.37	0.29	0.29	pass
-14.84	-50.87	27.54	0.46	0.46	pass
-50.36	-95.12	-12.38	0.54	0.54	pass
60.98	31.58	65.61	0.86	0.86	pass
45.57	27.43	69.43	0.43	0.43	pass
0.00	100.00	0.00	-1.00	-1.00	pass
0.00	0.00	0.00	-1.00	-1.00	pass
100.00	50.00	0.00	-1.00	-1.00	pass

Table E.16: Features.normalise() unit test results

Class.Method

double TSeq.sim(TSeq problem, ArrayList<Weights> weights)

Description

Returns the weighted Euclidean distance between the temporal sequence of retained cases (current instance of TSeq) and the temporal sequence of the new problem.

Tested manually using the examples from Chapter 4 without rounding.

Precondition

$|problem| = |case| = |weights|$

Valid outputs

Precondition met: Return the weighted Euclidean distance, $output \geq 0.00$

Precondition violated: -1.00

$p = (1.00, 0.00, 0.33)$

$c_1 = (0.67, 0.20, 0.00, 0.20)$ $\quad w_1 = (0.40, 0.20, 0.10, 0.50)$

$c_2 = (0.00, 1.00, 1.00, 0.00)$ $\quad w_2 = (1.00, 0.30, 0.20)$

$c_3 = (1.00, 0.6, 0.5, 1.00)$

inputs			expected output	actual output	result
problem	case (this)	weights			
$\langle p \rangle$	$\langle c_1 \rangle$	$\langle w_2 \rangle$	0.37773	0.37773	pass
$\langle p \rangle$	$\langle c_2 \rangle$	$\langle w_2 \rangle$	1.178889	1.178889	pass
$\langle p \rangle$	$\langle c_3 \rangle$	$\langle w_2 \rangle$	0.337313	0.337313	pass
$\langle c_3, p \rangle$	$\langle c_1, c_2 \rangle$	$\langle w_1, w_2 \rangle$	1.228959	1.228959	pass
$\langle c_3, p \rangle$	$\langle c_2, c_3 \rangle$	$\langle w_1, w_2 \rangle$	1.034785	1.034785	pass
$\langle c_3, p \rangle$	$\langle c_1 \rangle$	$\langle w_1 \rangle$	-1.00	-1.00	pass
$\langle c_3, p \rangle$	$\langle c_1 \rangle$	$\langle w_1, w_2 \rangle$	-1.00	-1.00	pass
$\langle p \rangle$	$\langle c_1, c_2 \rangle$	$\langle w_1 \rangle$	-1.00	-1.00	pass
$\langle p \rangle$	$\langle c_3 \rangle$	$\langle w_1, w_2 \rangle$	-1.00	-1.00	pass
$\langle p \rangle$	$\langle c_2, c_3 \rangle$	$\langle w_1, w_2 \rangle$	-1.00	-1.00	pass

Table E.17: TSeq.sim() unit test results

Class.Method

double Core.iobAdjust(Case problem, ArrayList<Case> retrieved, ArrayList<Case> caseBase, double, ait)

Description

Returns the adjusted bolus insulin suggestion after resolving differences between insulin on board in the new problem and the retrieved case(s).

Tested manually, includes the examples from Chapter 4 without rounding.

Precondition

$ait > 0$

Valid outputs

Precondition met: Return the adjusted bolus insulin advice, $output \geq 0.00$

Precondition violated: -1.00

Case-base #1 (CB_1)		
case	time	bolus insulin
c_1	180	5.0
c_2	360	4.0
c_3	540	5.0
c_4	720	6.0

Case-base #2 (CB_2)		
case	time	bolus insulin
c_1	200	7.0
c_2	320	3.0
c_3	520	5.0
c_4	700	3.0

Case-base #3 (CB_3)		
case	time	bolus insulin
c_1	200	7.0
c_2	420	5.0
c_3	700	7.0
c_4	750	5.0

Case-base #4 (CB_4)		
case	time	bolus insulin
c_1	200	7.0
c_2	320	3.0
c_3	520	5.0
c_4	700	2.5
c_5	840	3.0
c_6	1020	6.0
c_7	1180	4.0
c_8	1320	1.5
c_9	1500	6.0
c_{10}	1600	0.5

Problem #1 (p_1)	
time	bolus insulin
900	5.0

Problem #2 (p_2)	
time	bolus insulin
800	3.0

Problem #3 (p_3)	
time	bolus insulin
1720	5.5

inputs				expected output	actual output	result
problem	retrieved case(s)	case-base	ait			
p_1	$\langle c_3 \rangle$	CB_1	240	4.5	4.5	pass
p_2	$\langle c_2 \rangle$	CB_2	260	4.923076923	4.923076923076923	pass
p_2	$\langle c_2 \rangle$	CB_3	200	0.0	0.0	pass
p_3	$\langle c_3, c_6 \rangle$	CB_4	320	4.53125	4.53125	pass
p_1	$\langle c_1 \rangle$	CB_1	0	-1.0	-1.0	pass
p_1	$\langle c_1 \rangle$	CB_1	-1	-1.0	-1.0	pass

Table E.18: Core.iobAdjust() unit test results

Class.Method

double Core.evalBG(double postBG, double targetBG, double tdd)

Description

Returns a bolus insulin adjustment based on the postprandial blood glucose reading (mmol/L), target blood glucose level (mmol/L), and total daily insulin (insulin units).

Precondition

$tdd > 0$

Valid outputs

Precondition met: Return the bolus insulin adjustment.

If $postBG > targetBG$, the adjustment should be positive.

If $postBG < targetBG$, the adjustment should be negative.

If $postBG = targetBG$, the adjustment should be 0.0.

Precondition violated: exception thrown

inputs			expected output	actual output	result
postprandial blood glucose level	target blood glucose level	total daily insulin			
7.0	7.0	40.0	0.0	0.0	pass
3.5	5.5	41.5	-0.8797	-0.8797	pass
9.2	5.5	36.0	1.526232	1.526232	pass
5.0	7.0	38.5	-0.81611	-0.81611	pass
4.0	4.5	45.0	-0.23847	-0.23847	pass
6.0	6.0	0.0	exception	exception	pass

Table E.19: Core.evalBG() unit test results

Class.Method

double Core.calculator(double preBG, double targetBG, double isf, int ci, int carbs, double iob, double maxBolus)

Description

Returns a bolus insulin suggestion using the following formula:
bolus suggestion $= (preBG - targetBG) \times \frac{1}{isf} + carbs \times \frac{1}{ci} - iob$

Precondition

All inputs are validated prior to this method being called.

Valid outputs

$[0.0, maxBolus]$

preBG	targetBG	isf	ci (carbs-to-insulin)	carbs	iob	max bolus	expected output	actual output	result
6.6	5.5	2	13	60	2	10	3.165385	3.265385	pass
7	5.5	3	14	80	0	10	6.214286	7.214286	pass
5	6.5	2.5	10	140	5	10	8.4	9.6	pass
8	6	3	12	90	3	10	5.166667	6.166667	pass
10	6.75	2	15	65	0	10	5.958333	8.208333	pass
5	5	2	12	240	0	10	10	10	pass
5	5	2	12	0	5	10	0	0	pass

Table E.20: Core.calculator() unit test results

Appendix F

Rodin User and Developer Workshop 2012 conference paper

Appendix G

IEEE HealthCom 2013 conference paper