

TD

Belief Revision applied to Profile Dynamics

DOCTORAL THESIS

Yuri Aristides da Silva Godinho de Almeida

DOCTOR DEGREE IN INFORMATICS ENGINEERING



UNIVERSIDADE da MADEIRA

A Nossa Universidade

www.uma.pt

March | 2026

FCT

Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA EDUCAÇÃO E CIÊNCIA

Belief Revision applied to Profile Dynamics

DOCTORAL THESIS

Yuri Aristides da Silva Godinho de Almeida

DOCTOR DEGREE IN INFORMATICS ENGINEERING

SUPERVISION

Eduardo Leopoldo Fermé

CO-SUPERVISION

Arthur Henrique Casals do Nascimento
Sergi Bermúdez I Badia

Abstract

Belief revision is the process by which an intelligent agent modifies its beliefs when confronted with new information that contradicts its current knowledge state. Whilst the AGM framework provides rigorous normative principles for rational belief change, a persistent gap exists between these theoretical foundations and practical implementation. Concurrently, user profiling has become ubiquitous across recommender systems, adaptive interfaces, and cognitive rehabilitation platforms; however, the temporal dynamics of user profiles remain largely unaddressed by formal, principled methods. Most systems resort to ad hoc recreation of profiles from scratch, and data-driven approaches suffer from recency bias, leading to volatile and opaque profile evolution.

This thesis addresses three research questions. First, it investigates what insights can be extracted from the early Belief Revision implementations identified by Doyle and London to inform modern computational implementations of the AGM postulates. Second, it examines whether a unified, knowledge-based formalism can be formulated to govern the structure and dynamics of user profiles. Third, it explores whether the integration of belief revision algorithms can produce user profiles that are more stable, interpretable, and accurate in their evolution than traditional deep learning approaches.

The thesis delivers four principal contributions. A systematic literature review of pre-AGM belief revision implementations reveals that early work emphasised representational devices and procedural strategies—such as dependency structures and minimal-change search—that prioritised computational tractability and remain valuable for contemporary systems. The thesis presents the complete and operational implementation of the Knowledge-Driven Profile Dynamics (KDPD) formalism, a modular system that operationalises profiles as integer tuples, employs the Manhattan distance to instantiate the principle of minimal change, and implements all four axiomatically characterised revision and update operators. A comparative analysis against the NeuroAIReh@b cognitive rehabilitation framework identifies specific alignments and functional gaps. Finally, a concrete integration pathway maps NeuroAIReh@b workflow steps to KDPD components—encompassing performance-to-formula translation, session-by-session revision, and therapist-driven priority specification—thereby establishing a blueprint for deployment in clinical settings.

The findings affirm that pre-AGM implementation insights can be successfully integrated with the AGM theoretical framework to yield systems that are both formally grounded and computationally viable. The KDPD formalism provides a unified, domain-agnostic framework for profile dynamics that enforces stability through minimal change, enhances interpretability through explicit constraint-driven updates, and offers a principled alternative to opaque data-driven approaches. This work advances the practical application of belief revision theory to user profile management and demonstrates its viability within the domain of cognitive rehabilitation.

Keywords: Belief revision · User profile dynamics · AGM framework · Cognitive rehabilitation

Resumo

A revisão de crenças (BR) é o processo pelo qual um agente inteligente modifica as suas crenças quando confrontado com nova informação que contradiz o seu estado de conhecimento atual. Embora o modelo AGM forneça princípios normativos rigorosos para a mudança racional de crenças, persiste um fosso entre estes fundamentos teóricos e a sua implementação prática. Paralelamente, a criação de perfis de utilizador tornou-se onnipresente em sistemas de recomendação, interfaces adaptativas e plataformas de reabilitação cognitiva; no entanto, a dinâmica temporal dos perfis de utilizador permanece, em grande parte, sem resposta por parte de métodos formais e fundamentados. A maioria dos sistemas recorre à recriação ad hoc de perfis do zero, e as abordagens baseadas em dados sofrem de enviesamento de recência, conduzindo a uma evolução de perfis volátil e opaca.

Esta tese aborda três questões de investigação. Primeiro, investiga que perspetivas podem ser extraídas das primeiras implementações de revisão de crenças identificadas por Doyle e London para informar as implementações computacionais modernas dos postulados AGM. Segundo, examina se um formalismo unificado, baseado em conhecimento, pode ser formulado para governar a estrutura e a dinâmica dos perfis de utilizador. Terceiro, explora se a integração de algoritmos de revisão de crenças pode produzir perfis de utilizador que sejam mais estáveis, interpretáveis e precisos na sua evolução do que as abordagens tradicionais de aprendizagem profunda (deep learning).

A tese apresenta quatro contribuições principais. Uma revisão sistemática da literatura de implementações de revisão de crenças pré-AGM revela que os trabalhos iniciais enfatizavam dispositivos representacionais e estratégias procedimentais — tais como estruturas de dependência e pesquisa de mudança mínima — que priorizavam a viabilidade computacional e permanecem valiosos para os sistemas contemporâneos. A tese apresenta uma implementação completa e operacional do formalismo *Knowledge-Driven Profile Dynamics* (KDPD), um sistema modular que operacionaliza perfis como tuplos de inteiros, utiliza a distância de Manhattan para instanciar o princípio da mudança mínima, e implementa todos os quatro operadores de revisão e atualização caracterizados axiomáticamente. Uma análise comparativa com a estrutura de reabilitação cognitiva *NeuroAIreh@b* identifica alinhamentos específicos e lacunas funcionais. Finalmente, uma via de integração concreta mapeia as etapas do fluxo de trabalho do *NeuroAIreh@b* para os componentes do KDPD — abrangendo a tradução de desempenho-para-fórmula, a revisão sessão a sessão

e a especificação de prioridades orientada pelo terapeuta — estabelecendo assim um plano para a implementação em contextos clínicos.

Os resultados confirmam que as perspectivas das implementações pré-AGM podem ser integradas com sucesso na estrutura teórica AGM para produzir sistemas que são, simultaneamente, formalmente fundamentados e computacionalmente viáveis. O formalismo KDPD fornece uma estrutura unificada e agnóstica ao domínio para a dinâmica de perfis que reforça a estabilidade através da mudança mínima, melhora a interpretabilidade através de atualizações explícitas orientadas por restrições, e oferece uma alternativa fundamentada às abordagens opacas baseadas em dados. Este trabalho faz avançar a aplicação prática da teoria da revisão de crenças na gestão de perfis de utilizador e demonstra a sua viabilidade no domínio da reabilitação cognitiva.

Keywords: Revisão de crenças · Dinâmica de perfis de utilizador · Modelo AGM · Reabilitação cognitiva

Acknowledgments

Em primeiro lugar começo em português, pois nem todas as pessoas da minha família sabem ler inglês e quero começar por agradecer a eles. Nunca deixaram de acreditar em mim, nem mesmo quando eu cheguei a duvidar de mim! Existiram muitos momentos complicados nesta longa jornada e sem o vosso apoio eu não teria chegado aqui. Obrigado!

This work would not have been possible without the support, encouragement, and trust of many people who accompanied me throughout this long and demanding journey. Completing this project was far from easy, and it would not have been achievable without the help of those who believed in it and in me from the very beginning.

I would like to express my deepest gratitude to my supervisors, Professor Eduardo Leopoldo Fermé and Professor Sergi Bermúdez i Badia, from the Universidade da Madeira, and to Arthur Casals from oneg.ai. Their guidance, patience, and commitment to this work were invaluable. They generously dedicated their time, knowledge, and insight throughout the development of this project, continuously challenging me to improve and refine the ideas presented here. Without their support and mentorship, this work would certainly not have reached the same level of quality.

I would also like to thank the entire team at the NeuroRehabLab and Madeira NOVA LINCS for the collaboration, discussions, and shared efforts over these years. Being part of such a stimulating and supportive research environment was a privilege and played an important role in shaping both this work and my development as a researcher.

Finally, I extend my sincere thanks to everyone who, in one way or another, supported me during this process. Your encouragement and belief in this project made it possible to reach the end of this journey.

This work was supported by the Portuguese Foundation for Science and Technology (FCT) under the grant number SFRH-BD-138911-2018.

To you, my love, thank you for continuing to be my muse and my greatest source of inspiration.

Your presence in my life constantly reminds me to strive to be a better man and a better father.

Your support, patience, and belief in me have meant more than words can fully express.

Table of Contents

1	Introduction	1
1.1	Problem statement	3
1.1.1	Implementation	3
1.1.2	Profile Dynamics	5
1.2	Research Question	5
1.3	Proposal and Main Contributions	6
1.4	Publications	7
1.5	Structure of the Thesis	9
2	State of the Art	11
2.1	Belief Revision	11
2.2	The AGM Model	14
2.2.1	Safe and Kernel Contraction	16
2.2.2	Epistemic Entrenchment	16
2.2.3	Importance and Criticisms	17
2.3	Pré-AGM Algorithms Revisited	18
2.3.1	Truth Maintenance Systems: The Doyle Legacy	23
2.3.2	Assumption-Based Truth Maintenance Systems	24
2.3.3	Probabilistic Belief Networks	25
2.3.4	Default Reasoning Systems	26
2.3.5	Dependency-Based Approaches	27
2.3.6	Additional Systems from Doyle's Bibliography	28
2.3.7	Conclusions and Lessons Learned	29
2.4	Profile Dynamics Review	30
2.4.1	User Profile Dynamics	31
2.5	Cognitive Rehabilitation	41
3	NeuroAIR@b: A Cognitive Profiling and Training Framework	45
3.1	The Framework and its Challenges	45
3.1.1	The Role of Artificial Intelligence	50
3.2	The Profiling Challenge	51

3.2.1	The Structure of the Profile	51
3.2.2	Aggregation of Neuropsychological Assessments	54
3.2.3	The General Formula.....	55
3.2.4	Commencing in the Absence of Data	57
3.3	The Training Challenge.....	58
3.4	Profile Dynamics.....	60
3.5	Contribution to NeuroAIR@b	63
4	Implementing Knowledge-Driven Profile Dynamics	65
4.1	Theory	65
4.1.1	Profile Definition	65
4.1.2	Profile Language	66
4.1.3	Manhattan Distance	69
4.2	System Architecture and General Implementation	70
4.2.1	Mapping Formal Definitions to Implementation Artefacts	72
4.2.2	Abstract Syntax Tree Engine	73
4.2.3	Domain Management System	74
4.2.4	Tuple Finder Module.....	76
4.2.5	Profile Dynamics Module	81
4.3	Operationalisation of Profile Dynamics Models	82
4.3.1	Model 1: From One Profile to One Profile.....	83
4.3.2	Model 2: From One Profile to a Set of Profiles	85
4.3.3	From a Set of Profiles to a Set of Profiles	87
4.4	Performance Evaluation	91
4.4.1	Benchmarks	92
4.4.2	Empirical scaling with domain size	93
4.4.3	Limitations	99
4.5	Applying Knowledge-Driven Profile Dynamics to NeuroAIR@b	102
4.5.1	Overview and Advantages	102
4.5.2	Profile Creation.....	103
4.5.3	Translating Performance into Well-Formed Formulae	104
4.5.4	Session-by-Session Training Profile	105
4.5.5	Evolution Profile Across Sessions.....	105

4.5.6 The Predicted Profile and Its Role	106
4.5.7 Re-evaluation and Fine-tuning Profile Creation.....	107
4.5.8 Therapist-Driven Fine-tuning: Domain Priority	108
4.5.9 Summary and Integration Mapping	108
5 Conclusion and Future Work	111
5.1 Implementation Summary	111
5.2 Answers to the Research Questions.....	112
5.2.1 RQ1: Insights from Early Belief Revision Implementations	112
5.2.2 RQ2: A Unified Knowledge-Based Formalism for Profile Dynamics	112
5.2.3 RQ3: Stability, Interpretability, and Accuracy of Belief	
Revision-Based Profiles.....	113
5.3 Future Work.....	114
5.3.1 Empirical Validation Methodology	114
5.3.2 Algorithmic and Heuristic Enhancements	114
5.3.3 Performance and Scalability Optimisation.....	115
5.3.4 Framework and Language Extensions	115
6 Appendices	117
6.1 Proofs	117
6.1.1 Model 1 Postulates.....	117
6.1.2 Model 2 Postulates.....	120
6.1.3 Model 3 Postulates.....	127
6.1.4 Model 4 Postulates.....	135
6.2 Implementation Details.....	144
6.2.1 Sequence Diagram of the implemented modules.....	144
6.2.2 Abstract Syntax Tree Engine	145
6.2.3 Domain Management System	157
6.2.4 Tuple Finder Module.....	166
6.3 Benchmark setup	176
6.4 NeuroAIR@b Database.....	180
References	183

List of Figures

1	Distribution of Doyle and London’s taxonomy	21
2	NeuroAIReh@b rehabilitation workflow	46
3	Pearson correlations between ADNIMERGE variables	59
4	Architecture of the Knowledge-Driven Profile Dynamics (KDPD) implementation	71
5	Runtime scaling in the unrestricted regime	94
6	Runtime scaling in the restricted regime	95
7	Effect of predicate arity on runtime scaling	96
8	Runtime for consecutive-value constraints	100
9	Runtime for pairwise equality constraints	100
10	Runtime for mixed linear constraints	101
11	Sequence of operations in the KDPD implementation	144
12	Custom Abstract Syntax Tree (AST) for an example profile formula	149
13	NeuroAIReh@b relational database schema	181

List of Tables

1	AGM contraction and revision postulates	15
2	Global Issues.	22
3	Representations for Revising Beliefs.....	22
4	Procedures for Revising Beliefs.....	22
5	Non-Monotonic Inference Techniques.....	22
6	Inexact Inferential Techniques.....	22
7	Theoretical Issues.	22
8	Applications.	22
9	Related Issues.	22
10	Mapping of Neuropsychological Assessment (NPA) instruments to cognitive domains ...	53
11	Mapping from NeuroAIR@b to KDPD.....	109

List of Algorithms

1	End-to-end profile revision and update workflow	77
2	AST-based preparation and domain narrowing for one conjunctive clause	78
3	Distance-guided recursive search with pruning and backtracking	79
4	Overall control flow of <code>findTuplesWithConstraints</code>	80

List of Acronyms

ACP	Assessed Cognitive Profile
AD	Alzheimer's Disease
ADL	Activity of Daily Living
ADNI	Alzheimer's Disease Neuroimaging Initiative
AGM	Alchourrón, Gärdenfors, and Makinson
AI	Artificial Intelligence
API	Application Programming Interface
AST	Abstract Syntax Tree
ASTe	Abstract Syntax Tree engine
ATMS	Assumption-Based Truth Maintenance System
BDI	Belief-Desire-Intention
BDI-II	Beck Depression Inventory-II
BR	Belief Revision
BTS	Brain Training System
BURS	Batch Update Recommender Systems
CDR	Clinical Dementia Rating
co-NP	complement of NP (nondeterministic polynomial time)
CR	Cognitive Rehabilitation
CS	Cognitive Status
CT	Cognitive Training
CTT	Cognitive Training Task
DFS	Depth-First Search

DL	Deep Learning
DMS	Domain Management System
DNF	Disjunctive Normal Form
FCRey	Rey's Complex Figure Test
FCSRT	Free and Cued Selective Reminding Test
GA	Genetic Algorithm
GDS-30	Geriatric Depression Scale-30
GNPT	Guttman Neuropersonal Trainer
IADL	Instrumental Activity of Daily Living
IAFAI	Adults and Older Adults Functional Assessment Inventory
IURS	Incremental Update Recommender Systems
KDPD	Knowledge-Driven Profile Dynamics
KPI	Key Performance Indicator
MCI	Mild Cognitive Impairment
ML	Machine Learning
MMSE	Mini-Mental State Examination
MoCA	Montreal Cognitive Assessment
ND	Normative Data
NPA	Neuropsychological Assessment
PCA	Principal Component Analysis
PDM	Profile Dynamics Module
PRECiS	Patient-Reported Evaluation of Cognitive State
SDI	Socio-Demographic Information
SMC	Subjective Memory Complaints
STRIPS	Stanford Research Institute Problem Solver

TFM	Tuple Finder Module
TMS	Truth Maintenance System
VD	Vascular Dementia
VR	Virtual Reality
WAIS-III	Wechsler Adult Intelligence Scale, Third Edition
wff	well-formed formula
WMS-III	Wechsler Memory Scale, Third Edition

1 Introduction

Belief Revision (BR) is the process by which an intelligent agent modifies its beliefs when confronted with new information that contradicts its current knowledge state [1]. This fundamental cognitive and computational process involves determining which existing beliefs to abandon, which new beliefs to accept, and how to maintain consistency and coherence in the resulting belief system. Unlike simple belief addition, which merely incorporates new information, BR requires sophisticated mechanisms to handle conflicts, prioritise information sources, and preserve the most reliable or important beliefs while discarding those that have become untenable.

To illustrate the problem faced by BR, an example is adapted from [2, pp. 1-2]. It is supposed that the following statements are all held to be true:

- John was born in Portugal (α);
- Pedro was born in Massaguassú (β);
- John and Pedro are compatriots (γ).

From these statements, it is inferred that Massaguassú is in Portugal. It is now assumed that two new pieces of information have been received:

- Massaguassú is not a country (δ);
- Massaguassú is in Brazil (ϵ).

Since these five statements are inconsistent, it follows that it is not viable to accept all five as true. There are several different approaches to dealing with this problem. Should some previous beliefs be abandoned? If so, which ones are chosen for discard, and why? These are just a few of the questions that BR theories seek to answer.

The formal study of BR encompasses both normative theories, which specify how rational agents should revise their beliefs, and descriptive theories, which model how BR actually occurs in natural and artificial systems. At its core, BR addresses three fundamental operations: *expansion*, the addition of new beliefs without removing old ones; *contraction*, the removal of beliefs to restore consistency; and *revision*, the addition of new beliefs while removing conflicting old ones to maintain consistency [3].

BR is crucial for Artificial Intelligence (AI), as real-world AI systems must operate in dynamic, uncertain environments where information is incomplete, potentially contradictory, and subject to change [4]. Unlike static knowledge bases that assume perfect and unchanging information, intelligent agents must continuously update their understanding based on new observations, sensor data, user input, and changing environmental conditions.

It also plays a pivotal role in simulating human-like reasoning [5]. By modelling cognitive processes, AI systems can more effectively emulate human thought patterns; this is invaluable in developing intelligent assistants, educational tutors, and collaborative systems. Furthermore, BR supports the integration of learning and reasoning in AI [6]. As systems increasingly combine data-driven methods, such as Machine Learning (ML), with symbolic reasoning, BR facilitates the reconciliation of learnt patterns with logical rules. This is achieved through approximate operators [7] and anytime algorithms [8] that balance computational efficiency with theoretical soundness. Such integration fosters the development of robust hybrid AI systems capable of excelling in complex tasks [9, 10].

Finally, BR is indispensable for maintaining consistency in knowledge systems. Systems such as expert systems and knowledge graphs rely on consistent information to ensure accuracy and utility [11]. Updating these systems without introducing logical errors remains a fundamental challenge that BR effectively addresses. By enabling these diverse capabilities, BR functions as a key mechanism to advance the adaptability, reliability, and intelligence of AI systems [10].

In summary, the importance of BR in AI manifests in multiple dimensions:

Adaptive Reasoning: AI systems must adapt their reasoning as new evidence emerges; this requires mechanisms to identify which beliefs are no longer supported and how to propagate changes throughout interconnected knowledge structures [4].

Consistency Maintenance: As agents acquire new information, conflicts inevitably arise with existing beliefs. BR provides principled methods for resolving these conflicts while preserving the coherence of the overall knowledge base [3].

Learning and Knowledge Acquisition: ML systems inherently involve BR as they update models based on new training data, necessitating sophisticated approaches to balance new evidence against established patterns [6].

Multi-agent Systems: When multiple AI agents share information, BR becomes essential to incorporate the knowledge of other agents while maintaining individual epistemic autonomy [12].

Human–AI Interaction: AI systems interacting with humans must revise their beliefs based on user feedback, corrections, and changing preferences. This necessitates robust BR mechanisms capable of handling subjective and contextual information [13].

Ontology Repair and Knowledge Base Maintenance: Modern AI systems increasingly rely on formal ontologies to structure domain knowledge; however, these ontologies must evolve as understanding deepens or inconsistencies emerge [14–16]. Recent work has established formal connections between AGM BR and ontology repair, demonstrating that contraction operations can be adapted to description logic ontologies through optimal repair strategies that minimise information loss while restoring consistency [15, 17–20]. This extension of classical BR to expressive logical frameworks provides principled methods for maintaining large-scale knowledge bases in semantic web technologies, knowledge graphs, and enterprise systems, making BR essential for the reliability of contemporary knowledge-centric AI applications.

1.1 Problem statement

1.1.1 Implementation

The field of BR faces a fundamental challenge in bridging the gap between theoretical foundations and practical implementation [2, 21]. While the AGM framework provides rigorous normative principles for rational belief change, these operations are computationally intractable in the worst case (complement of NP (nondeterministic polynomial time) (co-NP)-complete) [22]. This makes exact implementation infeasible for large-scale applications and often necessitates approximations that may compromise theoretical guarantees [7, 23, 24]. In contrast, pre-AGM computational approaches demonstrated practical feasibility but lacked systematic theoretical foundations that could ensure rational behaviour or provide principled evaluation criteria.

This theory-to-practice gap manifests in several critical ways:

- **Computational Complexity:** AGM revision operations are generally co-NP-complete, making exact implementation infeasible for large-scale applications; however, the relationship between approximation strategies and theoretical soundness remains poorly understood [22, 25, 26].

- **Preference Specification:** AGM theory requires preference structures, such as epistemic entrenchment orderings or selection functions, but provides limited guidance on how to acquire these preferences from users, data, or domain expertise [27].
- **Implementation Strategies:** Whilst AGM provides clear normative constraints, it offers little guidance on the data structures, algorithms, or architectural decisions needed for practical systems [24, 28].

The motivation for this research stems from the recognition that pre-AGM computational approaches—developed before theoretical foundations were established—may contain valuable implementation insights that have been overlooked by the post-AGM focus on theoretical development. These early systems were designed with explicit attention to computational tractability, incremental processing, and practical deployment constraints; these are concerns that remain central to contemporary AI applications but are often secondary in theoretical research.

Additionally, the proliferation of AI systems in dynamic, uncertain environments, from autonomous vehicles to conversational agents, has renewed interest in practical BR implementations that can operate under real-time constraints with limited computational resources. Understanding how pre-AGM approaches addressed these challenges, and how their insights can be integrated with the AGM theoretical framework, is essential for advancing the development of next-generation intelligent systems.

To better understand progress in this field, it is essential to revisit the BR algorithms that preceded the influential AGM model, which has since become the dominant approach. By comparing the state of pre-AGM algorithms with post-AGM developments, it is possible to understand how these models can be evaluated in the context of AGM theory and, ultimately, how they can be adapted for the implementation of BR in intelligent agents. It is argued that the theory of belief change for belief bases has now matured to a stage where the next step—implementing BR in real-world agents—can be undertaken. This is due to advances within the research field itself and also to the increased computational power of small devices over the past decades. With this understanding, it is contended that the BR algorithms proposed by the AI community in the 1970s and 1980s likely contain valuable insights into the implementation of BR mechanisms.

1.1.2 Profile Dynamics

Over the last two decades, user profiling has become a cornerstone of various information technology domains. Prominent applications include recommender systems [29, 30], adaptive user interfaces [31], personalised content delivery [32], and cognitive or physical rehabilitation platforms [33].

Typically, systems store these profiles as a set of user preferences, predominantly represented through numerical attributes or categorical linguistic labels (e.g., "High/Medium/Low" or "Single/Married").

Although extensive research has focused on generating these profiles—often using ML techniques based on user navigation history—the temporal dynamics of these profiles are frequently neglected. Instead of incrementally updating an existing profile, most systems resort to a systematic, ad hoc recreation of the user model from scratch. Consequently, there remains a notable lack of formal, generalised research into profile evolution. Furthermore, the reliance on purely data-driven techniques limits system transparency. Existing efforts in explainable AI within personalised systems heavily favour explaining the final recommendations [34–36], leaving the underlying creation and evolution of the user profile entirely opaque.

1.2 Research Question

In 1980, Jon Doyle and Philip London surveyed the state-of-the-art implementations of BR, highlighting the core algorithmic challenges of the era [37]. Shortly afterwards, Allen Newell proposed the "knowledge level", advocating for models of intelligent agents based on rational reasoning rather than computational mechanisms [38]. In 1985, Alchourrón, Gärdenfors, and Makinson (AGM) answered this call with the AGM model [3], defining the standard operations of belief change: expansion, contraction, and revision. While the AGM model established the theoretical gold standard, its success shifted the field's focus almost entirely towards logical postulates, leaving earlier implementation work behind. With the notable exception of Katsuno and Mendelzon [39], little effort has been made to reconcile these theoretical advances with practical computation. However, revisiting these early, pre-AGM implementations may offer vital insights for constructing algorithms capable of supporting modern, post-AGM theories. Therefore, the first research question is:

RQ1: What insights can be extracted from the early BR implementations identified by Doyle and London to inform and enhance modern computational implementations of the AGM postulates?

The second objective bridges this theoretical framework with modern information technology, specifically User Profile Dynamics. The application of user profiles is ubiquitous, driving recommender systems [29, 30], adaptive interfaces [31], and rehabilitation systems [33]. While contemporary profiling heavily leverages Deep Learning (DL) to capture complex patterns, these models remain isolated within proprietary ecosystems (e.g., Facebook, Spotify, Google), preventing cross-domain interoperability. Crucially, existing research lacks a rigorous theoretical formalism capable of defining both the structure of a generic user profile and the dynamics of its evolution over time. Thus, the second research question naturally follows:

RQ2: Can a unified, Knowledge-Based formalism be formulated to govern the structure and dynamics of user profiles?

A significant limitation of current DL profiling models is their susceptibility to recency bias, which leads to abrupt and unstable profile shifts. For instance, an accidental click on an advertisement can disproportionately alter a user’s digital footprint. This over-responsiveness stems from the models’ limited temporal integration. Such a limitation can be elegantly addressed through BR theory. A core tenet of BR is the ‘principle of minimal change’, which posits that a system should make the smallest possible adjustment to its existing knowledge base to maintain consistency. The integration of this logic could temper the volatility of data-driven models. This leads to the final research question:

RQ3: Can the integration of BR algorithms produce user profiles that are more stable, interpretable, and accurate in their evolution than traditional DL approaches?

1.3 Proposal and Main Contributions

This thesis proposes and delivers four principal contributions.

Systematic Literature Review: The first contribution is a systematic review of pre-AGM BR implementations catalogued by Doyle and London [37]. This includes an analysis of how their computational strategies—representational devices such as dependency structures and pro-

cedural mechanisms such as minimal-change search—may be exploited within contemporary post-AGM frameworks.

Operational Implementation of KDPD: The second is a complete and operational implementation of the KDPD formalism [40]. This comprises a modular system that operationalises profiles as integer tuples, utilises the Manhattan distance to instantiate the principle of minimal change, and implements all four axiomatically characterised revision and update operators.

Comparative Framework Analysis: The third involves a comparative analysis of this implementation against the profile dynamics employed in the NeuroAIRh@b Cognitive Rehabilitation (CR) framework, identifying specific alignments and functional gaps.

Clinical Integration Pathway: The fourth is the provision of a concrete integration pathway that maps NeuroAIRh@b workflow steps to KDPD components—encompassing performance-to-formula translation, session-by-session revision, and therapist-driven priority specification—thereby establishing a blueprint for deployment in clinical settings.

1.4 Publications

The research developed in this thesis has given rise to the following publications, presented in chronological order. For each publication, we briefly summarise its scope and our specific contribution to the work.

AI-Rehab: A Framework for AI-Driven Neurorehabilitation Training – The Profiling Challenge

Y. Almeida, M. S. Sirsat, S. Bermúdez i Badia, and E. Fermé. 2020. In *Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies – Cognitive Health IT*, INSTICC.

This publication introduced the AI-Rehab framework for AI-driven neurorehabilitation, with particular emphasis on the profiling challenge that underlies the personalisation of rehabilitation. In this work, we contributed to the development and formal presentation of the framework, especially regarding the organisation of patient information and neuropsychological assessment data into a coherent profiling process. More specifically, the paper established the profiling component as the mechanism through which heterogeneous assessment results could be aggregated into a cognitive representation suitable for subsequent adaptive training and AI-supported rehabilitation planning.

Traditional vs. Low-Code Development: Comparing Needed Effort and System Complexity in the NexusBRaNT Experiment

D. Aveiro, V. Freitas, E. Cunha, F. Quintal, and **Y. Almeida**. 2023. In *IEEE 25th Conference on Business Informatics (CBI)*.

In this paper, we contributed to the articulation between the conceptual framework and its practical implementation. Our role focused on helping bridge the gap between the system requirements and their concrete realisation, thereby supporting the comparison between traditional and low-code development in the NexusBRaNT experiment [41]. We also contributed to the writing of the manuscript.

NeuroAIRh@b: an AI-Based Methodology for Personalised and Adaptive Neurorehabilitation

A. L. Faria, **Y. Almeida**, D. Branco, J. Câmara, M. Cameirão, L. Ferreira, A. Moreira, T. Paulino, P. Rodrigues, M. Spinola, M. Vilar, S. Bermúdez i Badia, M. Simões, and E. Fermé. 2023. *Frontiers in Neurology*, 14.

In this publication, we contributed to the operational infrastructure required by the NeuroAIRh@b framework. More precisely, we developed the server-side backend, designed and implemented the relational database (shown in appendix 13), created the dashboard for health professionals, and implemented the Application Programming Interface (API) required for interoperability with the rehabilitation software. These components collectively provided the data persistence, administrative control, and system integration mechanisms necessary to support cognitive profile creation, monitoring, and adaptation throughout the neurorehabilitation workflow. We also contributed to the writing of the manuscript.

Knowledge-Driven Profile Dynamics

E. Fermé, M. Garapa, M. D. Reis, **Y. Almeida**, T. Paulino, and M. Rodrigues. 2024. *Artificial Intelligence*, 331.

In this article, we contributed to the identification and formal specification of the requirements that a profile dynamics framework should satisfy, as well as to the definition of the underlying structure of the profile itself. In particular, this work helped establish the axiomatic basis for profile dynamics that later served as the theoretical foundation for the algorithmic development undertaken in the present thesis. We also contributed to the writing of the manuscript.

From Doyle to AGM: A Survey and an Implementation Roadmap for Belief Change

Y. Almeida and A. Casals. 2026. *The European Journal on Artificial Intelligence*.

This publication presented a survey of the computational landscape of belief change from the pre-AGM period to the AGM tradition, with particular attention to implementation-relevant insights. In this work, we were responsible for the survey research, including the identification and examination of the relevant literature, the extraction and organisation of the data, the analytical synthesis of the main findings, and the writing of the manuscript.

Implementing Knowledge-Driven Profile Dynamics

Y. Almeida, A. Casals, and E. Fermé. 2026. Submitted for review.

In this submitted work, we developed the full computational implementation of the KDPD formalism. This included the design and implementation of the system architecture, the operationalisation of the profile language and tuple-based representation, and the implementation of the corresponding revision and update operators. We also contributed to the writing of the manuscript and to the positioning of the implementation in relation to the theoretical framework and its application to profile dynamics.

1.5 Structure of the Thesis

The remainder of this thesis is organised as follows:

- **Chapter 2** surveys the state of the art in BR and profile dynamics. It examines the AGM framework, the pre-AGM computational approaches catalogued by Doyle and London, and the application domain of CR.
- **Chapter 3** presents the NeuroAIReh@b cognitive profiling and training framework, outlining its inherent challenges and the profile dynamics currently employed within the system.
- **Chapter 4** introduces the formal theory underlying the KDPD implementation. It describes the system architecture and details the operationalisation of all four profile change models, alongside the proposed integration pathway into NeuroAIReh@b.
- **Chapter 5** synthesises the principal contributions, provides comprehensive answers to the research questions, and outlines prospective directions for future work.

- **The Appendices (Chapter 6)** contain formal proofs of postulate satisfaction for each model and supplementary implementation details.

2 State of the Art

This chapter surveys the state of the art in BR and its subsequent application to user profile dynamics. It commences by examining the theoretical foundations of BR, spanning from its philosophical origins and Newell’s knowledge-level analysis to the seminal AGM framework, which established the normative principles governing rational belief change. The chapter then reviews pre-AGM computational approaches; these systems predate the formalisation of AGM theory and may harbour valuable implementation insights that have been overlooked in subsequent theoretical development. Finally, the exposition addresses the domain of profile dynamics and CR, where BR may be applied to model the evolution of user preferences in a principled manner. This survey establishes the theoretical and historical context necessary to address the central research questions of this thesis: how early BR implementations can inform modern computational approaches, and how BR can be integrated into user profiling systems to achieve more stable, interpretable, and accurate profile evolution.

2.1 Belief Revision

The philosophical roots of BR theory stem from epistemological inquiries into knowledge acquisition and belief change. The formal study of BR emerged from the intersection of AI, the philosophy of science, and cognitive psychology during the 1960s and 1970s. In his influential concept of the *knowledge level* [38], Allen Newell provided a crucial theoretical foundation by distinguishing between different levels of analysis in cognitive systems: the biological level, the symbol level, and the knowledge level.

The biological level, which is the lowest level of analysis, refers to the physical substrate or hardware where cognition actually takes place. In human beings, this is the brain, composed of neurons, synapses, and electrochemical signals. In computers, this is the electronic hardware, including logic gates, circuits, and transistors. The biological level defines the absolute limits of processing speed and memory capacity, providing the physical mechanisms that execute the higher-level processes of the mind [38, 42].

Sitting above the biological level, the symbol level (or syntactic level) is where information processing occurs. It is the realm of algorithms, data structures, and computer programs. At this level, a system represents information through symbols (such as words, numbers, or bits) and

manipulates them according to strict logical and syntactic rules [43]. Crucially, the symbol level is indifferent to the meaning of the symbols, focusing only on how they are stored, retrieved, and transformed. This level forms the bridge between the physical hardware below it and the intelligent, goal-directed behaviour (the knowledge level) above it [38].

At the knowledge level, Newell argued, intelligent behaviour can be understood in terms of the knowledge an agent possesses and the goals it pursues, independent of the specific symbolic representations or computational mechanisms used to implement that knowledge [38]. This perspective was revolutionary because it suggested that BR could be studied as a rational process governed by the principles of coherence, consistency, and goal-directed behaviour, rather than merely as a collection of ad hoc computational procedures.

Newell’s framework influenced early AI researchers to develop principled approaches to knowledge representation and reasoning that could support rational BR. This led to the development of Truth Maintenance System (TMS) [4], assumption-based reasoning frameworks [44], and other computational architectures designed to track the justifications for beliefs and manage belief dependencies systematically.

The knowledge level perspective also highlighted the importance of distinguishing between the *competence* level (what an ideal rational agent should do) and the *performance* level (what computationally bounded agents actually do). This distinction became central to BR research, leading to both normative theories that specify ideal BR behaviour and computational theories that address the practical constraints of implementing BR in real systems [38].

Early contributions from philosophers such as [45] and [46] emphasised the dynamic character of belief systems and the necessity of rational methods for updating them. These philosophical investigations established several key principles:

- **Minimal Change:** When incorporating new information, one should preserve as much of the existing belief system as possible while ensuring consistency;
- **Coherence:** The resulting belief system should preserve internal consistency and logical soundness;
- **Epistemic Order among beliefs:** Some beliefs are more fundamental or important than others, influencing which beliefs should be retained during revision.

Although Peirce and James are central to pragmatist ideas regarding belief and coherence, the formal, systematic foundations of minimal change and entrenchment in BR were developed later by decision-theoretic and AGM-style authors (Levi; Alchourrón, Gärdenfors, and Makinson) and by AI/logics researchers. Historical precursors such as [47] and [48] also shaped the pragmatists' views.

From a computational perspective, BR explores how intelligent systems (or agents) can update their beliefs in response to new information. This process is essential because, in real-world contexts, AI systems often need to handle dynamic, incomplete, or contradictory information. With a BR process in place, intelligent systems can maintain a consistent information base when presented with new input, thereby supporting more accurate decision-making [8, 22].

Although the formal study of BR can be traced to early work in AI and philosophical logic during the 1960s and 1970s, the field was revolutionised by the seminal work of AGM [3], which provided the first comprehensive formal framework for belief change operations.

Before the AGM framework, researchers developed various algorithmic approaches to BR, often tailored to specific applications. In particular, the problem of how to revise a database or knowledge base emerged within the AI community during the 1970s. The survey article Doyle and London [37] provided a selected bibliography of approximately 250 papers related to this area. In its introduction, it states that "BR concentrates on the issue of revising systems of beliefs to reflect perceived changes in the environment or acquisition of new information".

During the 1980s, several authors proposed different models for performing belief change [39, 49–55]. Yet, these models were developed at the syntactic level, i.e., with a commitment to the representation language. Therefore, as Nebel [56] pointed out, "the question remains how BR can be described on an abstract level independent of how beliefs are represented and manipulated inside a machine".

It remains unclear how to describe BR at the knowledge level, as introduced by Newell in his seminal article [38], but these ideas also inspired the formal development of the Belief-Desire-Intention (BDI) paradigm, in which agent behaviour is explicitly modelled in terms of beliefs, desires, and intentions [57, 58].

Although Newell did not propose a BDI architecture himself, his knowledge-level analysis provided a conceptual foundation that strongly influenced subsequent BDI models and agent-oriented reasoning frameworks. That paper exerted an enormous influence on the AI community, particularly on researchers in Knowledge Representation and Reasoning, including Brachman, Levesque, Moore, Halpern, Moses, Vardi, Fagin, Ullman, Shapiro, Borgida, and others.

2.2 The AGM Model

The AGM framework represents the most influential normative theory of BR. Developed by Alchourrón, Gärdenfors, and Makinson, and culminating in Gärdenfors’s comprehensive treatment in *Knowledge in Flux* [1], the framework, named after its creators’ initials, provides rationality postulates that specify how rational agents should revise their beliefs when confronted with new information [3]. It transformed the field by: (i) unifying expansion, contraction, and revision under a single theoretical framework; (ii) establishing rationality postulates for belief change operations; (iii) introducing representation theorems linking abstract postulates to constructive methods; and (iv) formalising relationships between different types of belief change [1, 3].

The AGM approach assumes a propositional language $\mathcal{L}$ closed under standard connectives, a consequence operator Cn that is monotonic, idempotent, and compact, and belief sets K such that $K = Cn(K)$. A belief set is thus closed under logical consequence: $K = \{\alpha \in \mathcal{L} : K \vdash \alpha\}$, embodying logical omniscience [1]. The framework defines three fundamental operations. **Expansion** ($K + \alpha = Cn(K \cup \{\alpha\})$) adds a new belief without regard for consistency. **Contraction** ($K - \alpha$) removes sufficient information from K so that α is no longer derivable, whilst preserving as much of the original information as possible. **Revision** ($K * \alpha$) incorporates a new belief whilst preserving consistency; the Levi Identity [59] formalises it as $K * \alpha = (K - \neg\alpha) + \alpha$ —contract by the negation of the new belief, then expand. The Harper Identity [60] provides the converse: $K - \alpha = K \cap (K * \neg\alpha)$.

AGM provides not only an axiomatic framework, but also a family of constructive models for belief change. In the standard construction of *partial meet contraction*, contracting K by a sentence α requires considering the family of maximal subsets of K that do not imply α , usually called the remainder sets of K with respect to α . A selection function then chooses the most plausible remainder sets, and the contraction result is obtained as their intersection [1, 3]. This

construction gives formal expression to the principle of minimal change: beliefs are removed only to the extent necessary to eliminate commitment to α . Important limiting cases include *full meet contraction*, in which all remainder sets are intersected, and *maxichoice contraction*, in which exactly one remainder set is selected. When the selection is determined by a transitive ordering over remainder sets, one obtains *transitively relational* partial meet contraction [1, 2].

The AGM model is governed by rationality postulates that ensure belief change is conducted logically and coherently. These postulates do not merely constrain admissible operators; they also yield exact representation results. The six basic contraction postulates characterise partial meet contraction, and the six basic revision postulates characterise partial meet revision. The supplementary postulates characterise the transitively relational variants of these operations [1, 3]. Table 1 summarises the standard AGM postulates.

Table 1: AGM rationality postulates for belief contraction (left) and BR (right). K denotes a deductively closed belief set; α and β denote formulae; $Cn(\cdot)$ denotes closure under logical consequence; and $+$, $-$, and $*$ denote expansion, contraction, and revision, respectively. The six basic contraction and revision postulates characterise partial-meet contraction and partial-meet revision; adding the two supplementary postulates characterises their transitively relational variants [1, 3].

Basic contraction postulates		Basic revision postulates	
$K - \alpha = Cn(K - \alpha)$	(closure)	$K * \alpha = Cn(K * \alpha)$	(closure)
$K - \alpha \subseteq K$	(inclusion)	$K * \alpha \subseteq K + \alpha$	(inclusion)
If $K \not\models \alpha$, then $K - \alpha = K$	(vacuity)	If $K \not\models \neg\alpha$, then $K + \alpha \subseteq K * \alpha$	(vacuity)
If $\not\models \alpha$, then $\alpha \notin K - \alpha$	(success)	$\alpha \in K * \alpha$	(success)
If $\vdash \alpha \leftrightarrow \beta$, then $K - \alpha = K - \beta$	(extensionality)	If $\vdash \alpha \leftrightarrow \beta$, then $K * \alpha = K * \beta$	(extensionality)
$K \subseteq (K - \alpha) + \alpha$	(recovery)	If $\alpha \not\models \perp$, then $K * \alpha \not\models \perp$	(consistency)
Supplementary contraction postulates		Supplementary revision postulates	
$(K - \alpha) \cap (K - \beta) \subseteq K - (\alpha \wedge \beta)$ (conjunctive overlap)		$K * (\alpha \wedge \beta) \subseteq (K * \alpha) + \beta$	(superexpansion)
If $\alpha \notin K - (\alpha \wedge \beta)$, then $K - (\alpha \wedge \beta) \subseteq K - \alpha$ (conjunctive inclusion)		If $K * \alpha \not\models \neg\beta$, then $(K * \alpha) + \beta \subseteq K * (\alpha \wedge \beta)$ (subexpansion)	

The original AGM framework was formulated around *belief sets*: sets closed under logical deduction [1, 3]. Deductive closure implies that belief sets are infinite for any non-trivial agent, rendering them computationally infeasible. The AGM tradition therefore expanded to include *belief bases*: finite, non-closed sets of explicit beliefs [22, 61]. The tension between belief sets (theoretical purity) and belief bases (computational feasibility) remains central; modern approaches develop algorithms on finite bases that approximate the rational behaviour specified by the AGM postulates [8, 25].

2.2.1 Safe and Kernel Contraction

Although partial meet contraction is the canonical AGM construction, subsequent work developed equivalent and more operational characterisations. One of these is *safe contraction*, introduced by Alchourrón and Makinson [62]. Safe contraction is based on a non-circular preference relation over the elements of the belief set. Intuitively, a sentence is safe with respect to α if every inclusion-minimal subset of K implying α either excludes that sentence or contains another sentence that is less preferred. The contraction outcome is then obtained by retaining the sentences that are safe with respect to α . All safe contractions are partial meet contractions, and the approach makes explicit that contraction depends not only on logical consequence but also on an underlying structure of epistemic priority [62, 63].

A more general approach is *kernel contraction*, introduced by Hansson [64]. Instead of focusing on maximal subsets of K that avoid implying α , kernel contraction considers the minimal subsets of K that are sufficient to imply α , called the α -*kernels* of K . An incision function then selects at least one sentence from each kernel for removal, thereby ensuring that the resulting set no longer implies α [2, 64]. Kernel contraction is particularly important from a computational perspective because it works directly with explicit minimal support sets and explicit deletion choices. Moreover, partial meet contractions coincide exactly with those kernel contractions that satisfy the additional smoothness condition, which shows that kernel-based methods generalise the standard AGM construction rather than competing with it [2, 64].

2.2.2 Epistemic Entrenchment

A further influential characterisation of AGM contraction is given in terms of *epistemic entrenchment*, introduced by Gärdenfors and Makinson [65–67]. The underlying idea is that not all beliefs have the same resistance to withdrawal: some are more central, more strongly supported, or more deeply embedded in the agent’s epistemic state than others. Contraction should therefore relinquish less entrenched beliefs before more entrenched ones. In this way, epistemic entrenchment provides an ordering-based formulation of minimal change [1, 66].

Entrenchment relations are typically required to satisfy conditions such as transitivity, dominance, conjunctiveness, minimality, and maximality [65–67]. These conditions ensure that the ordering is compatible with logical consequence and with the special epistemic status of tautologies and non-beliefs. A contraction operator is entrenchment-based if and only if it is a transi-

tively relational partial meet contraction [66, 67]. This result is conceptually important because it shows that AGM can be understood in two complementary ways: as a framework based on postulates and constructive set-theoretic operations, and as a framework based on preference orderings over beliefs. Subsequent work explored weaker or alternative entrenchment conditions and related ordering-based contraction schemes [68–71].

2.2.3 Importance and Criticisms

The AGM model provided the first widely accepted formalisation of rational BR, with the principle of minimal change at its core: an agent should alter its beliefs as little as possible to accommodate new information [1]. The framework has influenced knowledge representation, non-monotonic reasoning, and practical implementations by providing systematic evaluation criteria, design guidance, and a unified language for comparing approaches [1, 72, 73]. Its enduring importance also derives from the fact that it admits multiple equivalent reconstructions—postulate-based, remainder-set-based, kernel-based, and entrenchment-based—which together provide a rich conceptual and technical basis for the study of rational change [2, 3, 64, 66].

Despite its influence, the AGM framework has been subject to considerable critique. Logical omniscience and deductive closure create a mismatch with bounded agents. Belief sets are infinite and computationally infeasible, forcing approximations that weaken theoretical guarantees [7, 74, 75]. The *Recovery* postulate has been undermined by counterexamples, motivating alternative contraction schemes and a more cautious interpretation of contraction on finite representations [2, 76]. These criticisms motivate departures from ideal belief sets towards tractable, finite representations—a direction that the present thesis adopts. In Chapter 4, the implementation addresses logical omniscience by employing finite bounded integer profiles rather than closed belief sets; tractability is achieved through explicit tuple representations and a restricted profile language; and minimal change is operationalised via Manhattan distance, which induces orderings that determine which profiles are selected when multiple candidates satisfy the constraints.

A further distinction relevant to the thesis is that between *revision* and *update*. Katsuno and Mendelzon [39] distinguished revision (incorporating new information about a static world) from update (incorporating changes in the world itself). The thesis implements both: the KDPP framework [40] defines four axiomatically characterised operators—revision and update for single profiles (Models 1 and 2) and for sets of profiles (Models 3 and 4)—whose rationality conditions inherit

from AGM theory. The thesis does not directly implement classical AGM belief-set constructors; instead, it adopts the AGM normative principles and operationalises them in KDPD using finite tuple-based profiles, explicit logical constraints expressed as well-formed formulae, Manhattan-distance-driven minimal change, and operators whose postulate satisfaction is established in the appendix.

With the AGM theoretical framework established, the computational approaches that preceded and informed its development can now be examined. These pre-AGM systems, whilst lacking systematic theoretical foundations, embodied algorithmic insights and implementation strategies that would later find formal expression in AGM theory and continue to influence contemporary BR implementations.

2.3 Pré-AGM Algorithms Revisited

Central to understanding this pre-AGM landscape is the bibliography by Doyle and London [37], which provided the first systematic organisation of the emerging field. The authors’ survey undertook a task that, at the time, had not yet been formalised by a consensus logic of belief change: to make sense of a rapidly growing *practice* of belief maintenance and model updating across AI subfields by carving the space into an interpretable taxonomy [77]. Their method was deliberately empirical and cross-disciplinary. Rather than proposing a monolithic theory, they curated an indexed bibliography and attached to each item a bundle of *descriptors* that reflected what the artefact *does* (e.g. maintains dependencies), *how* it is *represented* (e.g. operator schemas; context layers), *what kind* of *inference* it licences (e.g. default rules), and *where* it is *used* (e.g. planning, explanation). The result is a pragmatic and multi-axis classification consisting of eight top-level families with fine-grained subcategories that span implementation and theory.

Doyle and London’s indexing process proceeds in three steps:

1. Identify recurring **tasks and pressures** that necessitate belief change (e.g., model updating after action, reconciling conflicting hypotheses, or absorbing new evidence);
2. Identify **representational devices** upon which systems rely to make updates feasible (e.g., operator add/delete lists, context layers, and dependency graphs);

3. Identify **inference regimes** and **meta-theoretical commitments** involved (e.g. default/non-monotonic rules versus purely monotonic reasoning; probabilistic/fuzzy calculi; and modal, temporal, or situation logics).

Each bibliographic entry is then *multi-labelled* along these axes. Crucially, the categories are *not mutually exclusive*: a single system might introduce a novel representation (context layers), rely on a specific procedure (dependency-directed backtracking), and presuppose an inference style (defaults), whilst being deployed in a particular application (explanation).

At the top level, the taxonomy comprises:

1. **Global Issues:** the frame problem, choosing between competing BRs, changes of belief or mind, and revising beliefs in response to new information;
2. **Representations for Revising Beliefs:** manual updates, operator descriptions, context-layered databases, change-triggered procedures, logical data dependencies, causal dependencies, and dependency-like representations;
3. **Procedures for Revising Beliefs:** chronological and non-chronological backtracking, operator application, and dependency-based revision;
4. **Non-monotonic Inference Techniques:** default reasoning, non-monotonic justifications, negation in databases, minimal model circumscription, and partial matching;
5. **Inexact Inferential Techniques:** probabilistic inference, decision theory, conclusion theory, multi-valued logics, and fuzzy logics;
6. **Theoretical Issues:** logics of rational belief, temporal logics, situation calculus, evolution theory, proof theory, non-monotonic logics, meta-theory, the theory of incomplete databases, modal logics, and reasoning about uncertainty;
7. **Applications:** explanation, modelling and execution, replanning following failures, transfer of expertise, learning, debugging computer programs, evolutionary computer systems, causal and perturbation analysis, and the control of reasoning;
8. **Related Issues:** human belief and memory, epistemology, inductive reference, hypothetical reasoning and counterfactual conditionals, and scientific theory change.

The prehistory of BR, prior to an agreed-upon logic of change, already identifies core tasks: *(i)* **updating** world models following actions (the frame problem); *(ii)* **reconciling** inconsistent hypotheses; and *(iii)* **absorbing** new information with minimal disruption. Doyle’s categories precisely mirror these pressures. For example, “Global Issues” enumerates the update/choice/assimilation triad; whilst “Representations” and “Procedures” identify the data structures and algorithms that rendered these tasks tractable at scale. In other words, the taxonomy represents a codification of engineering practice that responds to the motivational cases introduced historically. This classification system not only captured the state of the field circa 1980, but also provided a framework that would prove durable across subsequent decades of research.

To understand the relative emphasis and development within each category of the Doyle and London taxonomy, a quantitative analysis was conducted regarding the literature distribution across the eight classification areas. This analysis not only reveals which research directions received the most attention during the pre-AGM period, but also provides insight into the computational priorities and theoretical interests that shaped early BR research. The distribution patterns illuminate the field’s evolution from predominantly procedural and representational concerns towards the more theoretical foundations that would eventually culminate in the AGM framework. Tables 2 to 9 present the numerical breakdown of the articles categorised by Doyle and London within each category, whilst Figure 1 provides a detailed visual representation of these research concentration patterns.

For the purposes of this review, the focus is placed particularly on extracting and analysing the computationally representative systems that emerged from categories 2 and 3 of the Doyle and London taxonomy, which address representations and procedures for BR. These systems embody the algorithmic innovations that would later inform both the development of AGM theory and contemporary implementations in intelligent agents. The computational approaches examined include the TMSs by Doyle [4], with their sophisticated dependency tracking mechanisms; the Assumption-Based Truth Maintenance System (ATMS) by de Kleer [44], which enabled reasoning with multiple contexts simultaneously; the probabilistic belief networks by Pearl [78], which introduced uncertainty quantification into BR; and various dependency-based reasoning systems that pioneered incremental belief update strategies [79].

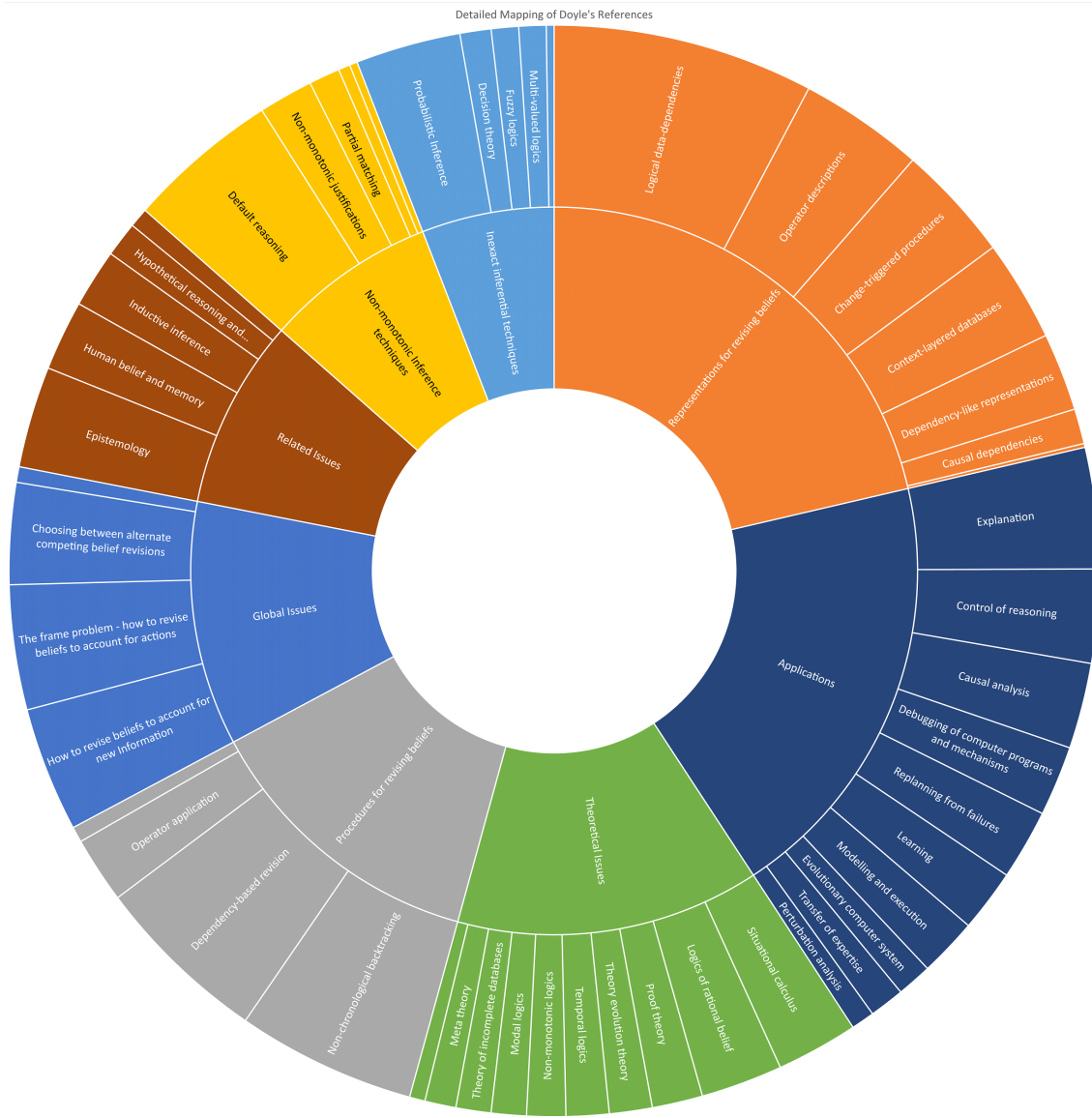


Figure. 1: Hierarchical distribution of the bibliography entries classified by Doyle and London across eight pre-AGM belief-revision research families. The inner ring represents the eight top-level categories, and the outer ring decomposes each category into its constituent subcategories; sector size is proportional to the corresponding number of bibliographic assignments. Because each entry may receive several labels, sector sizes represent classification assignments rather than mutually exclusive publications. The largest concentrations occur in Representations for Revising Beliefs and Applications, reflecting the practical and implementation-oriented emphasis of the pre-AGM literature

Table 2: Global Issues.

Subcategory	#Refs
1. The frame problem	32
2. Alternate competing BRs	26
3. Change of Belief or Mind	4
4. Revise beliefs accounting for new information	32
<i>Total</i>	<i>94</i>

Table 3: Representations for Revising Beliefs.

Subcategory	#Refs
1. Manual updates	1
2. Operator descriptions	32
3. Context-layered databases	26
4. Change-triggered procedures	30
5. Logical data-dependencies	67
6. Causal dependencies	9
7. Dependency-like representations	20
<i>Total</i>	<i>185</i>

Table 4: Procedures for Revising Beliefs.

Subcategory	#Refs
1. Chronological backtracking	4
2. Operator application	17
3. Dependency-based revision	45
4. Non-chronological backtracking	46
<i>Total</i>	<i>112</i>

Table 5: Non-Monotonic Inference Techniques.

Subcategory	#Refs
1. Default reasoning	39
2. Non-monotonic justifications	14
3. Negation in databases	3
4. Partial matching	8
5. Minimal model circumscription	2
<i>Total</i>	<i>66</i>

Table 6: Inexact Inferential Techniques.

Subcategory	#Refs
1. Probabilistic Inference	27
2. Decision theory	8
3. Conclusion theory	2
4. Multi-valued logics	7
5. Fuzzy logics	7
<i>Total</i>	<i>51</i>

Table 7: Theoretical Issues.

Subcategory	#Refs
1. Logics of rational belief	21
2. Temporal logics	11
3. Situational calculus	21
4. Theory evolution theory	11
5. Proof theory	13
6. Non-monotonic logics	10
7. Meta theory	8
8. Theory of incomplete databases	9
9. Modal logics	9
10. Reasoning about uncertainty	4
<i>Total</i>	<i>117</i>

Table 8: Applications.

Subcategory	#Refs
1. Explanation	31
2. modeling and execution	15
3. Replanning from failures	18
4. Transfer of expertise	9
5. Learning	16
6. Debugging	18
7. Evolutionary computer systems	9
8. Causal analysis	22
9. Perturbation analysis	6
10. Control of reasoning	24
<i>Total</i>	<i>168</i>

Table 9: Related Issues.

Subcategory	#Refs
1. Human belief and memory	18
2. Epistemology	26
3. Inductive inference	15
4. Hypothetical reasoning	9
5. Scientific theory change	5
<i>Total</i>	<i>73</i>

2.3.1 Truth Maintenance Systems: The Doyle Legacy

The development of TMS by Doyle [4] represents one of the most significant pre-AGM contributions to computational BR. TMS addresses the fundamental problem of maintaining consistency within knowledge bases by tracking the justifications for beliefs and automatically retracting them when their supporting assumptions are withdrawn.

2.3.1.1 Core Truth Maintenance System Architecture

The TMS architecture consists of several key components:

Belief Nodes: these represent individual beliefs or propositions, each associated with a status (IN or OUT) indicating whether the belief is currently accepted;

Justification Structures: these encode the inferential relationships between beliefs, specifying which combinations of beliefs support or contradict others;

Dependency Networks: these maintain explicit records of how beliefs depend upon one another, enabling the efficient propagation of belief changes;

Contradiction Detection: mechanisms for identifying inconsistencies and triggering BR processes.

The TMS operates through a cycle of assumption-making, inference, contradiction detection, and dependency-directed backtracking. When a contradiction is detected, the system identifies the minimal set of assumptions that must be retracted to restore consistency, utilising the recorded dependency structure to guide this process efficiently.

2.3.1.2 Algorithmic Innovations

Doyle's TMS introduced several algorithmic innovations that remain influential:

Dependency-Directed Backtracking: rather than chronological backtracking, which undoes the most recent assumptions first, TMS uses dependency information to identify and retract only those assumptions that are causally responsible for contradictions;

Incremental Consistency Maintenance: instead of rebuilding the entire belief set after each change, TMS propagates changes incrementally through the dependency network;

Assumption-Based Reasoning: TMS explicitly represents and manages assumptions, enabling the systematic exploration of alternative assumption sets;

Justification-Based Inference: all beliefs are associated with explicit justifications, providing transparency and enabling sophisticated BR strategies.

2.3.1.3 Implementation Details

Doyle provided detailed implementation descriptions and annotated code for TMS algorithms [80].

Key implementation features include:

Node Data Structures: each belief node maintains pointers to its justifications, consequences, and current status, enabling the efficient traversal of the dependency network;

Justification Records: these encode support relationships using lists of antecedent and subsequent nodes, with specific handling for conditional and default rules;

Propagation Algorithms: breadth-first and Depth-First Search (DFS) procedures for propagating belief changes through the network whilst maintaining consistency;

Conflict Resolution: algorithms for identifying minimal conflict sets and selecting which assumptions to retract when multiple resolution strategies are available.

2.3.2 Assumption-Based Truth Maintenance Systems

De Kleer’s ATMS [44] extends TMS by maintaining multiple consistent belief bases simultaneously, each corresponding to different assumption combinations. This approach avoids the computational overhead of repeated BRs by pre-computing the consequences of various assumption sets.

2.3.2.1 ATMS Architecture

The ATMS maintains the following:

Assumption Sets: collections of assumptions that can be held simultaneously without contradiction;

Labels: each belief node is associated with a label indicating the specific assumption sets that support that belief;

Nogood Sets: collections of assumptions that lead to contradictions and must therefore be avoided;

Justification Networks: structures similar to those in TMS but extended to manage multiple contexts simultaneously.

2.3.2.2 Computational Advantages

ATMS provides several computational advantages over single-context TMS:

Context Switching: the capacity for rapid switching between different assumption sets without the need for recomputation;

What-If Analysis: the efficient exploration of hypothetical scenarios by examining various assumption combinations;

Parallel Reasoning: the ability to maintain and explore multiple reasoning contexts simultaneously;

Assumption Minimality: the automatic identification of the minimal assumption sets sufficient to support particular conclusions.

2.3.3 Probabilistic Belief Networks

Pearl's work on probabilistic belief networks (subsequently known as Bayesian networks) provided a fundamentally different approach to BR, based upon probability theory rather than logical inference [78]. These networks represent probabilistic dependencies between variables and facilitate BR through Bayesian conditioning.

2.3.3.1 Network Structure

Probabilistic belief networks consist of:

Nodes: these represent random variables corresponding to propositions or events;

Directed Edges: these encode probabilistic dependencies between variables;

Conditional Probability Tables: these specify the probability distributions for each variable, given its parents within the network;

Evidence Nodes: these represent observed information that constrains the probability distributions.

2.3.3.2 Belief Revision Algorithms

Pearl developed several algorithms for BR within probabilistic networks:

Message Passing: for singly connected networks, BR can be performed efficiently using local message-passing algorithms that propagate probability updates through the network;

Clustering Methods: for multiply connected networks, clustering algorithms group variables to reduce computational complexity whilst maintaining exact inference;

Approximation Methods: when exact inference is intractable, various approximation methods provide efficient approximate BR.

2.3.3.3 Computational Complexity

The computational complexity of BR within probabilistic networks is dependent upon the network structure [78]:

Singly connected networks: linear time complexity relative to network size when utilising message-passing algorithms;

Multiply connected networks: exponential worst-case complexity, though often tractable for networks possessing an appropriate structure;

Approximation trade-offs: various approximation methods provide different trade-offs between accuracy and computational efficiency [7].

2.3.4 Default Reasoning Systems

Reiter's default logic and related non-monotonic reasoning systems provided a further pre-AGM approach to BR [81]. These systems manage incomplete information by making default assumptions that can be retracted upon the discovery of contradictory information.

2.3.4.1 Default Logic Framework

Default logic extends classical logic with default rules of the form:

$$\frac{\alpha : \beta_1, \dots, \beta_n}{\gamma}$$

This rule specifies that if α is believed and it remains consistent to believe $\beta_1, \dots, \beta_n$, then γ may be concluded by default.

2.3.4.2 Computational Approaches

Several computational approaches to default reasoning were developed:

Fixed-point algorithms: these compute default extensions by iteratively applying default rules until a fixed point is reached;

Proof-theoretic methods: these involve the development of proof procedures capable of handling default rules directly without the need to compute complete extensions;

Compilation approaches: these transform default theories into classical logic theories that can be processed using standard inference engines.

2.3.5 Dependency-Based Approaches

Several pre-AGM systems focused explicitly on dependency tracking and management.

2.3.5.1 Reason Maintenance Systems

In a reason maintenance system, McAllester [79] provides a distinct approach to dependency tracking that emphasises computational efficiency and incremental reasoning.

2.3.5.2 Causal Networks

Various researchers have developed causal network approaches that represent causal relationships explicitly and utilise them to guide BR processes [82–86].

Whilst many of these algorithms lacked the formalism later introduced by the AGM model, they represented significant efforts to address BR within the AI domain. These algorithms often adopted pragmatic approaches to updating belief bases, including:

- **Updating heuristics:** many pre-AGM algorithms were based on simple heuristics for adding or removing beliefs from the knowledge base, often without consideration of the logical consequences. Although these techniques were computationally efficient, they frequently resulted in mechanisms lacking rigour, which led systems into inconsistent states [37, 87, 88];
- **Probability-based models:** certain pre-AGM approaches utilised probabilistic methods for adjusting beliefs upon the receipt of new information. These methods often attempted to incorporate techniques such as Bayesian statistics; however, a lack of formal rigour also led to difficulties, particularly regarding scalability and applicability to systems based on pure logic [78, 82, 85];
- **Rule-based systems:** a further common approach in the 1970s involved rule-based systems, in which new beliefs were inserted into or removed from the knowledge base according to a set of predefined rules. Whilst such systems were effective in specific scenarios, they were unable

to manage contradictions or inconsistencies in the information received, as the rules lacked adaptability [87–89].

2.3.6 Additional Systems from Doyle’s Bibliography

Beyond the major systems detailed above, the Doyle and London bibliography documents several other influential pre-AGM approaches that contributed to the computational foundations of BR:

STRIPS and Operator-Based Planning: Fikes et al. [87] introduced operator descriptions with explicit add and delete lists, providing early mechanisms for representing state changes. Whilst primarily an action-planning system, the Stanford Research Institute Problem Solver (STRIPS) approach to minimal change (whereby only specified facts change) anticipated the AGM inclusion postulate and informed the subsequent distinction between BR and belief update [52].

CONNIVER and Context Mechanisms: McDermott et al. [90] introduced context layers and non-chronological backtracking, enabling the efficient exploration of alternative belief states. These mechanisms anticipated the multi-context reasoning of ATMS and informed theoretical work regarding modularity and independence in BR [91].

PLANNER and Pattern-Directed Invocation: Hewitt [88] provided early mechanisms for change-triggered procedures, where belief changes automatically invoke relevant inference rules. This approach influenced justification propagation within TMS and anticipated the active database triggers utilised in contemporary belief base implementations.

Limitations of Pre-AGM Implementations: most pre-AGM systems were demonstrated primarily on small-scale domains due to the computational constraints of the 1980s. Notable exceptions that achieved real-world deployment include the use of a TMS-style reasoning architecture in the DART expert system [89] and ATMS in the GDE diagnostic system [92]. Contemporary implementations benefit from significant improvements in processing power and memory, potentially rendering pre-AGM algorithmic insights more practically viable at scale.

Despite the lack of formalism, the work undertaken during this period contributed significantly to the initial development of BR within AI, establishing an important foundation for subsequent, more formal approaches.

2.3.7 Conclusions and Lessons Learned

The analysis of BR systems developed before the formulation of the AGM model reveals that early AI research addressed the problem of belief change primarily from an engineering and computational perspective, rather than from a logical or normative one. These systems emerged in concrete implementation contexts, where the main objective was to maintain consistent knowledge bases in dynamic environments with limited computational resources. When analysed retrospectively in light of AGM theory, these early approaches allow several lessons to be identified that remain relevant to the design of contemporary BR implementations.

First, pre-AGM systems highlight the importance of explicit representational structures for managing belief change. Many of the solutions developed at this stage used dependency graphs, justification structures, or context layers to record the relationships between beliefs. These representations made it possible to efficiently identify which parts of the knowledge base depended on particular assumptions, thereby facilitating the detection of inconsistencies and the selection of candidate beliefs for revision. This type of mechanism anticipates later concepts in AGM theory, such as the identification of minimal subsets responsible for an inconsistency in kernel-based or belief-base contraction methods.

A second important lesson concerns the incremental nature of belief maintenance. Early systems did not treat revision as an isolated logical operation, but rather as a continuous process integrated into the agent's reasoning. Through mechanisms such as dependency tracking or justification-driven backtracking, these systems propagated changes locally within the knowledge structure. This incremental approach contrasts with more abstract formulations of the AGM model, which typically consider belief sets closed under logical consequence, and suggests that practical implementations must operate on finite structures that are modified progressively.

Another relevant conclusion is the close relationship between representation and procedure in early revision systems. In practice, revision algorithms depended heavily on how knowledge was structured. Certain representations enabled specific revision strategies: for example, dependency graphs supported dependency-driven revision algorithms, while operator descriptions in planning systems facilitated model updates after actions were executed. This observation suggests that the

design of BR mechanisms cannot be separated from the design of the representational structures used by the system.

Likewise, the systems analysed demonstrate the importance of the locality principle in belief change. Instead of recomputing the entire state of knowledge after each update, many algorithms attempted to restrict revision to the portion of the knowledge base affected by the new information. This principle of local updating constitutes an operational way of implementing the idea of minimum change, later formalised within the AGM framework.

Finally, the historical analysis reveals that many of the ideas introduced by AGM theory can be interpreted as a later formalisation of intuitions already present in pre-AGM systems. Justificatory dependencies anticipate contraction methods based on minimal subsets; context-based representations are related to semantic interpretations based on possible worlds; and the preference mechanisms implicit in early algorithms prefigure concepts such as epistemic entrenchment ordering. This indicates that the transition to the AGM model did not completely replace previous approaches, but rather provided a formal framework capable of systematising many of their computational intuitions.

2.4 Profile Dynamics Review

The content presented in this section is discussed in substantially greater depth in the works of Purificato et al. [93] and Freitas et al. [94].

User profiles are composed of structured sets of attributes that characterise the ways in which individuals engage with a particular system. These profiles encapsulate the interests, requirements, preferences, demographic properties, and goals of users, providing a formalised representation of their interactional and contextual characteristics [95]. The construction and maintenance of a user profile comprise five principal steps:

1. **Information Retrieval.** To accurately infer user needs and interests, the system must acquire as much relevant information as possible; consequently, the information retrieval phase is crucial for effective personalisation. Information retrieval can be categorised as explicit, when information is collected through user feedback or by the manual editing of profiles [96–102]; implicit, when information is derived from the analysis of interactions and behaviour within

the system [97, 99, 101–108]; or hybrid, when both explicit and implicit sources of information are combined [97, 99, 107, 109–112];

2. **Feature Extraction.** Feature extraction constitutes a further critical phase in user modelling, involving the identification and retrieval of profile attributes across heterogeneous domains. Various methodological approaches are employed to derive these features, with the overarching objective of improving the predictive accuracy and overall effectiveness of the resulting models. Eke et al. [106] compiled a list of extracted features categorised by profile application;
3. **User Representation.** The representation of the user profile denotes the formal structuring, organisation, and presentation of information stored within a system or application. Its primary objective is to construct an accurate and comprehensive user model, enabling the system to facilitate informed inference, decision-making, and prediction regarding interactions, behaviours, and preferences. The survey by Purificato et al. discusses several representation variants: term-based, semantic-network, concept-based, graph-based, universal, and holistic [93];
4. **Preference Learning.** Following the collection of information and the determination of the profile structure, weights can be calculated and associated with user preferences. The values of these weights may be static or dynamic. The most common techniques utilised are k -Nearest Neighbours [110, 113], Support Vector Machines [101, 114], Convolutional Neural Networks [115], Naive Bayes [103, 110, 116], multi-agent systems [106], filtering techniques [110, 117], and knowledge-based techniques or knowledge graphs [118–124].

Finally, the concluding step consists of the maintenance and continuous updating of the user profile. The following section is devoted to an in-depth examination of the current state of the art regarding user profile dynamics.

2.4.1 User Profile Dynamics

At this stage, the profile may be regarded as complete. However, if it is intended to represent a human user, or any aspect thereof, it must be capable of accommodating and reflecting the recurrent changes that the user undergoes over time. This section examines several key challenges and discusses strategies utilised to address and mitigate them.

2.4.1.1 Long-Term vs Short-Term Interests

Multiple strategies have been proposed to address this problem. One such approach involves constructing weighted user profiles, commonly distinguishing between a short-term interest profile and a long-term interest profile for each user.

Li et al. [125] address shifting user preferences by maintaining two separate digital profiles: a stable "topic tree" for long-term interests and a small, rotating buffer for short-term activity. As a user clicks on new results, the system automatically strengthens relevant topics within the long-term model; conversely, if a topic is neglected for an extended period and falls out of the short-term buffer, its importance is gradually reduced and it is eventually deleted to ensure the profile remains current.

Bennett et al. [126] tackle the tension between recent and historical interests by proposing a unified framework that models search behaviour over three distinct temporal horizons: the current session, the long-term interaction history, and a global aggregate signal. Through empirical evaluation of large-scale search log data, the authors demonstrate that long-term history provides the greatest benefit for disambiguating intent at the onset of a search task, whereas short-term session behaviour becomes the dominant factor in predictive accuracy as the task progresses. The study ultimately concludes that the most effective approach to personalising search results is not to aggregate all temporal signals naïvely, but rather to employ a model that adaptively learns to weight recent versus historical information as a function of the user's current stage within the search process.

Marin et al. [102] proposed a strategy for interpreting implicit feedback by contrasting items selected by the user with those that are ignored. The system employs an algorithm inspired by physical laws of attraction and repulsion to refine its model of user preferences iteratively. Short-term preference shifts are incorporated immediately following each interaction, whereas long-term tendencies are captured by analysing the historical record of multiple selections, thereby enabling the system to preserve predictive accuracy even as interests evolve. Empirical evaluation demonstrated that this method substantially enhances recommendation quality, generally ranking a preferred item within the top results after approximately thirty interactions.

Ravi et al. [107] investigate the problem of evolving user interests by proposing a dynamic profiling framework that concurrently models long-term, relatively stable preferences and short-

term, transient intentions. To accommodate these temporal fluctuations, the approach employs a set of specialised digital agents that continuously observe ongoing activities in order to incorporate emerging interests, discard obsolete ones, and allow gradually diminishing preferences to decay over time. By updating the user profile automatically on the basis of real-time behavioural signals, the system can differentiate between enduring historical patterns and momentary, context-specific requirements, thereby maintaining the relevance of recommendations as tastes change.

Deng et al. [99] approach the long-term versus short-term user profile by explicitly decomposing user modelling into two complementary but interacting processes: a stable long-term preference model learnt from historical search sessions, and a rapidly adaptive short-term intention model derived from real-time interactions within the current session. Long-term change is captured through the hierarchical extraction of positive and negative feedback across past queries and sessions, producing a relatively persistent representation of general preferences and aversions over time. Conversely, short-term change is managed by a dedicated real-time dual-feedback module that prioritises recent sequential interactions to infer transient goals and information needs that may temporarily override long-term preferences. A gating mechanism then fuses these representations so that enduring interests provide a behavioural prior but are selectively modulated by immediate feedback when intent shifts, allowing the profile to remain stable over time whilst reacting sharply to session-level deviations.

Taken together, these works converge on a clear methodological lesson: effective personalisation under evolving user interests requires an explicit temporal decomposition of the user profile, rather than a monolithic or uniformly aggregated model. Across domains and techniques, the literature consistently shows that long-term profiles provide a stable semantic backbone that captures enduring preferences, whilst short-term models capture transient, context-dependent intent that dominates prediction as the interaction unfolds. The most successful approaches, therefore, do not merely separate these signals but dynamically regulate their relative influence through decay, weighting, or adaptive fusion mechanisms. This allows momentary behavioural evidence to override historical tendencies when appropriate without erasing them. Such temporal adaptivity emerges as a necessary condition for maintaining both robustness and responsiveness in personalised systems operating under continuous preference drift.

2.4.1.2 Recalculation vs Evolution

A representative illustration of the distinction between profile *recalculation* and profile *evolution* is provided in the survey on recommender systems by Zhang and Kim [127]. In their work, contemporary approaches are categorised into **Batch Update Recommender Systems (BURS)**, which rely on periodic batch-wise recomputation of user and item profiles, and **Incremental Update Recommender Systems (IURS)**, which continuously evolve these profiles through online or incremental updates.

BURS represent the traditional offline approach to recommendation, where models are trained on a large, static training set in a one-time process. In this framework, the recommender remains unchanged and continues to make the same suggestions until the next scheduled batch retraining occurs. During operation, new user interactions (such as clicks or views) are collected and added to the dataset, but they are not immediately reflected in the model's output. When the designated retraining period T is reached, the system must conventionally be retrained from scratch using the entire accumulated dataset D ; this process becomes increasingly computationally expensive and less scalable as the volume of data grows over time. Because BURS cannot immediately incorporate recent observations, they often struggle to capture real-time user interest drifts or shifts in item popularity, which can lead to diminished recommendation quality in dynamic environments.

In contrast, IURS are designed to manage real-time data streams by instantly updating the model as new observations arrive. Unlike batch systems, IURS do not wait for a periodic retraining window; instead, they capture user interest drifts and changes in item popularity dynamically by processing data incrementally. As a user interacts with the system, feedback is immediately utilised to update the model parameters, often through a specialised training function ($f_{\text{incre-train}}$), and the interaction is stored in a small, representative data memory or reservoir rather than requiring the entire historical dataset. This "test-then-learn" approach allows the system to deliver personalised suggestions that reflect the most recent activities whilst remaining computationally efficient and scalable, as it avoids the "train-from-scratch" paradigm.

Numerous additional examples may be found in the literature; however, the principal advantages and limitations remain common regardless of the specificities of the system.

In the **Profile Recalculation** approach, user profiles are updated through a periodic, comprehensive retraining process in which the system processes all available historical data as a single

static block. This method facilitates global optimisation, allowing the model to capture deep, long-term dependencies and maintain high stability by ignoring short-term behavioural noise [127]. As it relies on standard offline ML pipelines, it is relatively simple to implement. However, this stability is achieved at the cost of significant information lag; the system cannot capture "temporal dynamics" or sudden shifts in interest in real-time, meaning recommendations remain stagnant until the next scheduled update [128]. Furthermore, recalculation is computationally inefficient because it requires retraining the entire model from scratch, even when only a small fraction of the data has changed [127].

Conversely, **Profile Evolution** focuses on real-time responsiveness by updating user models sequentially as new data streams in. This approach effectively addresses the "consumer over-choice" problem by immediately reflecting the most recent actions of a user [127]. It is specifically designed for drift adaptation, employing mechanisms such as non-linear decay to manage "interest forgetting" and shifts in underlying data distributions [127–129]. Whilst it is more storage-efficient, often requiring only the current state rather than a full history, it faces the complex "stability–plasticity dilemma" [130]. Without sophisticated balancing mechanisms, these systems risk catastrophic forgetting, where valuable long-term preferences are prematurely overwritten by recent, noisy interactions, leading to a degradation in model quality known as recency bias [128].

In short, the recalculation–evolution distinction captures a fundamental trade-off in dynamic profiling: batch-based recalculation offers global stability and long-term optimisation at the expense of responsiveness and scalability, whereas incremental evolution enables real-time adaptation to preference drift. However, the latter must carefully manage the stability–plasticity tension to avoid recency bias and catastrophic forgetting. Consequently, the choice between the two is determined less by algorithmic fashion and more by the required balance between consistency, timeliness, and computational efficiency.

2.4.1.3 Evolutionary/Genetic Algorithms

Evolutionary Algorithm, or **Genetic Algorithm (GA)**, is a class of stochastic optimisation techniques inspired by the biological processes of natural selection and genetics. They maintain a "population" of potential solutions that undergo iterative refinement through operations such as crossover, mutation, and selection. By mimicking the principle of "survival of the fittest," these

algorithms explore complex, high-dimensional search spaces to find near-optimal solutions in instances where traditional mathematical methods might fail [131].

El Houda et al. proposed a system that treats specific search terms and interests as building blocks, analogous to genes, and combines them to create a variety of potential profiles [132]. By simulating a process of natural selection, the algorithm repeatedly tests, mixes, and slightly alters these combinations to identify the set of topics that most accurately reflects the current focus of the user. This enables the profile to evolve automatically over time by discovering new hobbies, highlighting frequent topics, and removing interests that are no longer relevant.

Rana et al. distinguish their approach from traditional GAs by proposing a dynamic recommender system based upon evolutionary clustering to manage shifting user interests over time [117]. Rather than utilising the typical biological mechanics of a GA, the system employs an optimisation procedure that balances two competing goals: ensuring the new user profile accurately reflects the most recent data (snapshot quality) whilst ensuring that it does not deviate so drastically from the past that it becomes inconsistent (history cost). By using a *variance score* to track changes in user ratings at different time intervals, the algorithm refines groups of similar users and incrementally updates their profiles. This allows the system to maintain high recommendation accuracy and manage large volumes of data significantly faster than complex genetic or static models.

In essence, evolutionary and genetic approaches frame user profile dynamics as a continuous optimisation problem in which preferences emerge from population-based search and incremental refinement, enabling robust adaptation to drifting interests at scale. However, profile change is driven by fitness heuristics and stability constraints rather than explicit semantic commitments, yielding effective but conceptually shallow models of preference evolution that offer limited interpretability or guarantees of rational consistency.

2.4.1.4 Adaptation Algorithms and Rules

Adaptation algorithms constitute the mechanisms, ranging from deterministic heuristics to deep-learning-driven update strategies, that map observed user behaviour and contextual changes to profile modifications. These rules dictate how profile attributes, numerical weights, or latent embeddings are refined over time. Modern approaches increasingly emphasise implicit data collection and real-time dynamic updates, allowing profiles to evolve from initial imprecise states into high-fidelity representations through continuous interaction analysis [93, 101, 102].

Marin et al. propose a system that automatically updates user profiles by observing specific choices and the alternatives that are ignored [102, 133]. This process is inspired by the physics of attraction and repulsion: when a user selects an item, the system "pulls" the profile closer to the characteristics of that choice, whilst simultaneously "pushing" the profile away from the traits of items that were ranked highly but ultimately rejected. To ensure that the profile remains stable and accurate, the system performs rapid updates after every interaction and more comprehensive adjustments once a more extensive history of decisions has been collected. The study conducted by Zhang et al. [101] may also be considered within this context.

López-Jaquero et al. address the necessity for software to adjust its appearance and behaviour automatically as user surroundings or personal expertise evolve over time [134]. Rather than authoring rigid, permanent code for every possible device or scenario, the authors propose a system that utilises "sensors" to monitor real-world changes. When these sensors detect a specific trigger, the system applies predefined visual transformation rules to update the interface, such as substituting large buttons for compact menus to conserve space. By employing a graphical design tool to map these changes, developers can create flexible interfaces that evolve in accordance with the user profile without the requirement to reprogram the application manually for every new situation.

Pukkhem describes the "LORecommendNet" [135] framework, which utilises semantic mapping and inference rules to bridge the gap between diverse course materials and individual learner profiles. The system employs an adaptation approach that analyses the unique learning style of a student, based upon standardised psychological models. These profiles are subsequently processed through a set of "if-then" reasoning rules that automatically match specific learner traits to the most appropriate educational resources.

Taken together, evolutionary and adaptation-based approaches demonstrate that user profiles can be rendered highly responsive by treating preference change as an optimisation or rule-driven update problem. This enables continuous, data-driven evolution that balances recency, stability, and scalability; yet, these methods prioritise empirical performance and heuristic convergence over semantic transparency. Consequently, they offer powerful but largely opaque mechanisms for profile change that lack explicit notions of belief, conflict resolution, or rational minimal change.

2.4.1.5 Context Awareness

A further approach to achieving dynamic user profiles relies upon context awareness, whereby information regarding the situational context (e.g. location, time, device, or activity) is continuously incorporated into the personalisation process [136, 137]. Context-aware systems have been widely shown to support adaptive and evolving user profiles by modulating preference relevance, activating contextual profile views, and triggering incremental profile updates as conditions change [93, 138].

Wu et al. define context-aware recommendation as a method that moves beyond static user–item interactions by integrating external factors, such as time, location, and past activity sequences, to better understand shifting user preferences [34]. Rather than treating a profile as a fixed set of interests, the model utilises context as a dynamic lens to adjust predictions based upon the specific environment of an interaction. For instance, the system can leverage recurrent neural networks to track the evolution of tastes over time or employ attention mechanisms to prioritise recent "short-term" actions over historical "long-term" habits.

Schaefer et al. describe a smart home system that utilises environmental sensors to monitor real-time data, such as a person's location or the current room temperature, to adjust and update digital user profiles automatically [139]. By combining these live observations with pre-existing personal preferences, the system creates temporary "customisation profiles" that dictate the behaviour of home appliances. Should a user be dissatisfied and manually override a setting, the system captures this change as new information. It then employs a set of automated rules to compare the manual adjustment against the original settings, evolving the permanent user profile over time to better predict and reflect evolving habits and requirements.

The SMARTMUSEUM system, introduced by Ruotsalo et al., utilises context awareness to create dynamic, personalised user profiles by capturing real-time data from the environment of a tourist and their direct interactions [140]. Using a probabilistic model, the system monitors situational variables such as physical location via GPS or RFID sensors, the current motivation for a visit, and whether the individual is travelling with companions. These contextual details are combined with explicit "I like" or "I dislike" feedback provided by the user regarding specific cultural objects to update interest preferences automatically. Because the system stores each piece of feedback alongside the specific context in which it was provided—for instance, an interest in historical buildings only whilst travelling within a specific region—it can refine the profile to provide more

accurate situational recommendations that adapt as the user moves between different locations and scenarios.

Golemati et al. establish a general framework for user profiles that records stable characteristics, such as education, expertise, and interests, to facilitate the adaptation of computer applications to specific requirements [141]. Whilst the core model focuses on permanent traits, it utilises context awareness to update and populate these profiles automatically by scanning the digital environment. For instance, the system can infer current interests by analysing web browsing history and bookmarks, identify social circles by scanning address books, and determine location or nationality through technical data such as IP addresses and language settings. By capturing these situational cues and digital habits, the system can refine the user profile without manual input, ensuring that personalised features—such as the optimal method for displaying complex information—remain relevant to the background and current goals of the user.

Sutterer et al. describe a system that employs context awareness to select and apply the most relevant user preferences from a profile automatically, based upon a real-time situation [142]. When a service platform detects a change in context, it triggers a profile selection service that evaluates current environmental data against predefined rules. By utilising ontology reasoning, the system can interpret complex relationships between locations, such as recognising that presence in a specific meeting room implies the user is in a "business place". This enables the platform to update the active settings for mobile applications and communication services intelligently without requiring manual intervention, ensuring the technology always behaves appropriately for the current environment.

Xu et al. utilise context awareness to learn and update user preferences by tracking the manner in which an individual interacts with different types of information throughout a daily routine [143]. Unlike traditional systems that rely upon pre-set rules, this approach logs every instance of a user accessing information alongside the current situation, including time, location, and the device in use. By analysing these logs to determine which types of content and levels of detail frequently coincide within specific situations, the system constructs an adaptive profile capable of automatically predicting and presenting the most relevant information for a given moment. The profile is updated both through explicit commands and through "implicit feedback", whereby the system assumes a preference has been correctly identified if the information provided is viewed without manual modification.

Gao et al. present a system that updates user profiles by gathering context from all connected smart devices, rather than exclusively examining web browsing history [144]. This context includes local information, such as the types of files saved (for example, emails or images) and the nature of device interaction—specifically tracking which application windows are open and the duration spent reading particular documents. A dedicated "client agent" on each device monitors these local behaviours, such as the frequency of window switching, to determine which topics are of primary importance at any given time. This information is subsequently combined with web search queries and analysed utilising a digital dictionary to identify related concepts, allowing the system to adjust and refine the user profile automatically to reflect personalised interests more accurately.

Yin et al. enhance user profiles by recognising that social media activity is driven by both long-term personal interests and the "temporal context", which refers to topics currently trending among the general public [145]. The system constructs more accurate profiles by distinguishing between these two influences, filtering out temporary spikes in behaviour caused by global news or events to identify individual tastes more clearly. To ensure these profiles remain relevant, the model treats user interest as a moving target that evolves over time rather than a fixed state. It addresses data gaps by assuming that current interests are likely to be similar to both the recent past of the individual and the contemporary interests of their social circle.

RecomMetz, introduced by Colombo-Mendoza et al., updates user profiles by automatically capturing real-time environmental data—specifically the current location and the exact time—whenever a recommendation is requested [146]. This contextual information is combined with explicit ratings and visiting history to create a dynamic digital portrait of preferences. By tracking these variables, the system can distinguish between static interests and immediate requirements, such as the distance an individual is willing to travel or the preferred time of day for visiting a theatre. This continuous feedback loop ensures that the profile evolves as the user interacts with the mobile application, allowing the system to refine suggestions based upon both past behaviour and current circumstances.

Context-aware approaches demonstrate that dynamic user profiling can be effectively achieved by continuously conditioning preferences on situational factors, enabling profiles to adapt in real time through contextual activation, relevance modulation, and incremental updates. However, these systems typically treat context as a pragmatic control signal or statistical correlate rather than

as information subject to principled consistency management. This results in adaptive but largely heuristic profile evolution that lacks the formal guarantees of rational change found in models based upon BR.

2.4.1.6 Belief Revision

The AGM BR model [2,3] is a formal framework used to represent the dynamics of the beliefs held by a rational agent. To the best of our knowledge, few works explicitly relate BR to user profiles. We proposed an AGM-based methodology for constructing and dynamically updating user profiles to represent and reassess information regarding user interests [40]. This approach additionally emphasises the formal specification and enforcement of the principle of minimal change, ensuring that each profile revision introduces only the least necessary modifications. This work is discussed in greater detail in Chapter 4.

Lau and Song employ BR to maintain up-to-date user profiles by treating preferences as a set of logical beliefs, each assigned a specific strength or "firmness" level [147]. When interests shift, the system automatically adjusts these strength levels within the profile. If a new preference contradicts an existing one, the model identifies and removes the older, weaker beliefs to ensure that the profile remains consistent and logically coherent. This adaptive process allows the recommendation agent to transition smoothly between different contexts, ensuring that the services suggested always reflect the most current requirements and priorities of the user.

Overall, this demonstrates that AGM BR provides a principled and underexploited foundation for modelling dynamic user profiles. Existing work shows that representing preferences as structured belief sets, revised under a minimal-change principle, yields profiles that evolve coherently, resolve conflicts rationally, and remain aligned with the current interests of the user. However, the literature remains sparse and fragmented, leaving substantial room for systematic, implementation-oriented frameworks that fully operationalise BR within profile dynamics.

2.5 Cognitive Rehabilitation

Given that the concepts developed in this thesis are applied to CR, a concise state-of-the-art review of this field is presented below to provide the necessary contextual framework, as described by Faria et al. [148].

According to the World Health Organization, dementia is amongst the leading causes of disability and dependency in individuals aged 65 or older. By 2050, the percentage of older people is expected to increase by 35%, which raises the number of individuals at risk of developing dementia from any etiology [149]. Up to 53.7% of all cases of dementia are assumed to be due to Alzheimer’s Disease (AD), whilst 15.8% are considered to be due to Vascular Dementia (VD) [150]. As no effective pharmacological treatment has been found to date, the increase in older adults with dementia necessitates the delivery of adapted and personalised neuropsychological interventions. This is particularly urgent for individuals with Mild Cognitive Impairment (MCI), a clinical condition that increases the risk of developing dementia by 38% [151], and those who have suffered a stroke, where the likelihood of developing VD is estimated to range from 36% to 67% [152].

NPA is essential to determine the neuropsychological profile of a patient. However, methodologies have remained reliant upon labour-intensive data collection procedures that provide relatively data-poor estimates of human behaviour, despite rapid technological advances in other healthcare fields [153]. The integration of technology, specifically through AI methodologies, into NPA practices possesses tremendous potential to advance the field more rapidly in numerous areas, such as the management of cognitive and functional deficits. Nevertheless, standard paper-and-pencil tasks remain the most frequently utilised methods [154].

Technology-based rehabilitation methods with high ecological validity, particularly those based upon the use of Virtual Reality (VR), have been shown to lead to improved outcomes in CR [155]. One reason for this may be that VR-based methods allow for the incorporation of cognitive tasks within the simulation of Activity of Daily Living (ADL) and the creation of well-controlled environments oriented towards the specific requirements of patients [156–158]. Reh@City, a VR-based CR tool, is an example in which tasks related to memory, attention, language, and executive functions are integrated into the performance of several ADLs [159]. A recent randomised controlled trial involving stroke participants who underwent rehabilitation with Reh@City revealed a significant impact on cognitive and functional domains compared to equivalent standard paper-and-pencil tasks [160].

Recently, a questionnaire was administered to healthcare institutions in Portugal to understand the current perspective of health professionals regarding the use of technologies for CR [161]. Data from 116 participants indicated that health professionals primarily utilise games, puzzles, and

paper-and-pencil tasks. Concerning the profile of patients undergoing CR, dementia and stroke were reported as the primary conditions being addressed, with the majority of patients aged over 60. Results indicated that technologies are not yet widely utilised by health professionals in CR sessions, with most participants (65.5%) reporting no experience with CR technologies. The most frequently mentioned barriers were the absence of technologies within the institution and a lack of qualified human resources to support their implementation.

The limited adoption of computer-based NPA and rehabilitation [162] may be attributed to the fact that, rather than incorporating many of the advances in cognitive neuroscience or computer science, most test developers merely redesign paper-and-pencil methods for computer administration [153]. Although digitising current tests certainly offers advantages over analogue technology, these could be leveraged far more effectively if development efforts also focused on capturing more behavioural data and increasing the ecological validity of the assessments. Ecological validity relates to the functional and predictive relationship between patient performance during NPA and behaviour in various real-world settings [163].

Over recent decades, AI capabilities have grown exponentially and have become ubiquitous. The technology is present in diverse sectors, from automotive and wearable technology to the operating theatres of advanced hospitals. However, as AI systems increase in complexity, it becomes more difficult to explain and understand the methodology behind their solutions. Martens et al. [164] identified that as performance increases, interpretability tends to decrease. They categorised AI systems in order of increasing performance: rule-based systems, intrinsically linear models, and Artificial Neural Networks or Support Vector Machines. The application of ML in healthcare is also evolving rapidly, with practical implications for the prevention, diagnosis, treatment, and prognosis of specific clinical conditions [165]. In the context of stroke, ML approaches are frequently utilised to support clinical diagnosis, yet few studies focus on applying ML to the rehabilitation of post-stroke impairments [166]. To the best of our knowledge, only three CR platforms utilise an ML approach to adapt and personalise training sessions: the Guttman Neuropersonal Trainer (GNPT) [167,168], Neuro-World [169], and the Brain Training System (BTS) [170,171].

The GNPT consists of a tele-CR platform for patients diagnosed with Acquired Brain Injuries, aiming to provide neuropsychological services beyond the clinical setting whilst increasing the personalisation, duration, and intensity of the rehabilitation process [168]. This platform encom-

passes telemedicine services and AI for knowledge extraction, such as data mining, collaborative environments, and automatic system adaptation based upon patient performance. The personalisation process begins with a baseline NPA; these results are stored within the GNPT system and utilised to define the cognitive profile of the patient. The system then proposes a Cognitive Training (CT) therapeutic plan based upon this profile. The adjustment of the plan—specifically the type of CT tasks and difficulty levels—is performed automatically by the system according to performance [172]. Regarding content, this platform comprises 95 computerised CT exercises targeting several cognitive functions and sub-functions, as designed by neuropsychologists [167].

The Neuro-World system is comprised of a set of six mobile games designed to challenge visuospatial short-term memory and selective attention [169]. This approach enables the self-administration of assessments and the remote monitoring of the Cognitive Status (CS) of the patient. This process is facilitated by analysing game performance and estimating Mini-Mental State Examination (MMSE) results through ML algorithms. A longitudinal study involving 12 stroke survivors with mild cognitive deficits demonstrated that Neuro-World could estimate MMSE scores with a low normalised root mean square error (5.75%). A significant contribution of this work is the integration of assessment and rehabilitation within a single tool.

Lastly, the BTS aims to maximise CR through the implementation of online supervision and an algorithm that automatically selects and schedules CT exercises [170, 171]. The difficulty level of the exercises is determined based upon the cognitive profile of the participant, which is updated as the individual progresses. The system utilises a scoring mechanism to compare performance across different exercises, which are merged according to the corresponding cognitive domain level. A supervision process was implemented based upon "red flags" triggered whenever the system detects issues regarding user engagement, compliance, or adherence. The BTS is currently being evaluated within an online personalised intervention designed to reduce cognitive decline in an older age cohort.

3 NeuroAIReh@b: A Cognitive Profiling and Training Framework

This section presents NeuroAIReh@b [148], a novel cognitive profiling methodology and training framework that utilises AI to maximise the personalisation and adaptation of rehabilitation prescriptions. NeuroAIReh@b constitutes a direct extension of our previous work, AI-Rehab [173]. As an initial proof of concept, the framework was implemented within an optimised version of Reh@City [159,160]. To create the initial patient profile, results from NPA instruments were first aggregated according to empirical input from neuropsychological assessment and rehabilitation experts; subsequently, these were (re)calibrated with the support of ML algorithms. To personalise and adapt the NeuroAIReh@b Cognitive Training Task (CTT)s, the system considers the patient profile, the training objectives, and the performance levels obtained in previous CTTs. The theoretical framework for profile dynamics developed in [40], based upon BR theory [2,3], was utilised to analyse the evolution of the patient profile.

3.1 The Framework and its Challenges

CR is considered the most effective approach to address cognitive deficits [174]. However, current CR tools present several limitations: a) they are challenging to adapt to individual patients as they require the application of an extensive battery of NPA instruments, the results of which are interpreted manually and are often prone to errors during the selection of CTTs; b) they involve high implementation costs, as they necessitate multiple sessions conducted in clinical environments by neuropsychologists; and c) session-to-session adaptation to patient performance is not consistently performed, which may limit rehabilitation potential and patient motivation.

To address these primary limitations, a framework was developed for the personalised and adaptive delivery of CR, which is divided into four distinct components as indicated in **Figure 2**. This section describes, at a high level, the processes to be performed in each framework component (represented by numbers 1–9 in the **Figure**) and the challenges involved in its construction.

Patient Profiling: this component aims to create a multidimensional patient profile that integrates several NPAs to determine a baseline CS.

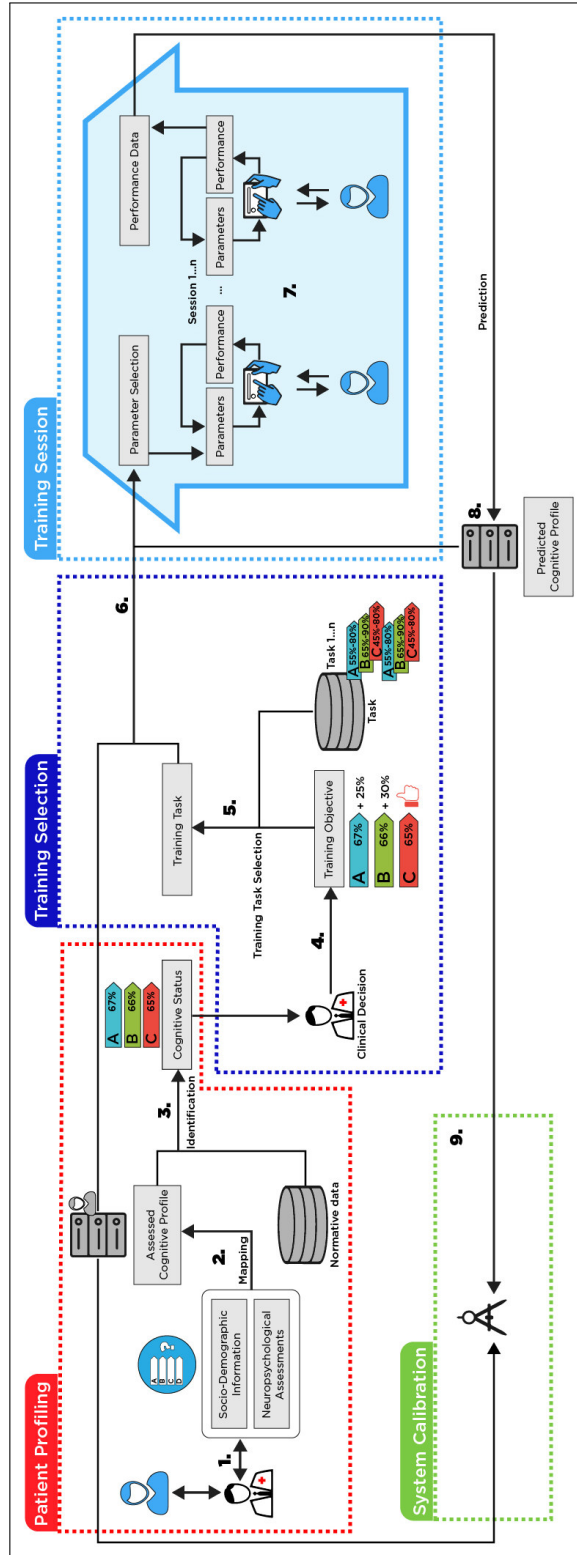


Figure 2: NeuroAIRh@b workflow for personalised and adaptive CR, organised into four components and nine numbered steps. Patient Profiling comprises NPA, aggregation of NPA results and Socio-Demographic Information (SDI) into an Assessed Cognitive Profile (ACP), and comparison with Normative Data (ND) to derive the patient's CS (Steps 1 to 3). Training Selection translates the CS into clinical objectives, selects suitable CTTs, and determines their initial parameters (Steps 4 to 6). Training Session adapts task difficulty to maintain approximately 50%–70% success and aggregates session performance into a predicted cognitive profile (Steps 7 to 8). System Calibration conducts a new assessment, compares the reassessed and predicted profiles, and recalibrates the task–domain relationships, adaptation parameters, or prediction process when discrepancies are detected (Step 9). Arrows indicate the longitudinal feedback cycle through which reassessment informs subsequent profiling and training.

1. Neuropsychologists utilise validated NPA instruments to assess the cognitive functioning of patients. An NPA instrument is a standard constituent of integrated medical care and is necessary to prescribe and evaluate the efficacy of rehabilitation procedures. A comprehensive NPA aims: (i) to measure and assess cognitive abilities, ADL performance, personality traits, and emotional and behavioural functioning in light of the premorbid functioning of the patient; (ii) to quantify the nature and severity of cognitive and functional deficits by analysing the symptoms and signs present within the context of the structural and functional integrity of brain functioning, thereby differentiating between normal and pathological cognitive decline; (iii) to define a baseline level of performance across cognitive, functional, and emotional domains, which can be examined in a longitudinal registry through repeated evaluations, thus enabling the monitoring of the clinical evolution of the patient regarding response to interventions (e.g. CT, psychotherapy, or pharmacological therapy) or disease progression; (iv) to identify personal resources and preserved functions that are useful for planning and implementing compensatory intervention procedures.

Challenges: to represent profiles effectively, it is necessary to define a formal language. This language is utilised for representing the profiles, expressing their properties, computing relevant metrics, and determining their dynamics. Following the definition of the general structure, the specific cognitive, functional, behavioural, and emotional domains to be considered during profile creation must be identified. Furthermore, it is necessary to specify the relevant NPAs for a comprehensive and multidimensional evaluation of these domains and to determine which SDI should be included for profiling purposes.

2. By aggregating the different NPAs with the SDI of the patient, the system constructs an ACP.

Challenges: to the best of our knowledge, a gap exists in the literature regarding the integration of data from multiple, heterogeneous NPAs and its consolidation into a consistent cognitive profile.

3. The profile ACP is not, in itself, sufficient to determine the CS of the patient; it must be compared against the ND available for each NPA tool.

Challenges: the ND for each NPA may be provided for different clinical conditions and are often categorised by socio-demographic groups. As with step 2, this information must be ag-

gregated and compared to the consolidated profile using an objective and quantifiable distance metric.

The Training Selection: After determining the patients' CS, which gives information about the preserved and impaired domains, neuropsychologists must define the most appropriate CTTs for neurorehabilitation.

4. Based on the patient's CS, neuropsychologists determine specific training objectives for each patient, i.e., in which cognitive domains the rehabilitation training must be focused on to regain or compensate for lost cognitive abilities and functional independence.

Challenges: The ultimate goal of CR is to help patients to regain independence and autonomy in their ADLs. It can be difficult to establish objectives on a system that will mostly accept numerical values when the objectives are usually set subjectively. The system should be able to perform this translation.

5. After establishing the training objectives and the set of available CTTs in the system, NeuroAIR@b will compute which tasks are most appropriate for the training. This is possible since each task has its own profile, i.e., one that details which cognitive domains are required to perform it and how the task's difficulty can be parameterized for the different cognitive domains.

Challenges: Each task must include information about which cognitive domain it trains. Additionally, it should include constraints regarding the minimal and maximal values for a particular cognitive domain of the suitable patient profile. The set of selected tasks must be optimally provided with the training objective (considering that the number of selected tasks is also limited given the amount of time and number of sessions assigned to the CT program).

6. By combining the initial profile and CTTs, the system establishes the initial parameters for the tasks. These initial parameters determine the task's difficulty according to the different cognitive domains.

Challenges: Combination is not trivial, and it is based on the tuple $\langle\langle$ task, the value of the cognitive domain in the profile, and associated difficulty $\rangle\rangle$ that will be adjusted in the system using ML techniques (see System Calibration).

The Training Session: this component describes the performance of a CT session within the NeuroAIR@b framework.

7. Prescribed training sessions may be conducted at a clinic or within a domestic setting. Each session consists of a set of predefined tasks executed on a tablet or a personal computer. The NeuroAIREh@b system calculates patient performance at each iteration and redefines task difficulty by adjusting parameters. This ensures the success rate remains between 50% and 70%, thereby avoiding boredom or frustration while ensuring the individual remains challenged and engaged [175].

Challenges: to maintain a task score between 50% and 70%, the system must establish a relationship between the scores across different tasks, the task parameters, and the resulting difficulty relative to each cognitive domain of the patient profile.

8. Following the completion of a training session, the system aggregates performance data across all tasks to estimate whether evolution or involution has occurred within the various cognitive domains, thereby defining a *predicted cognitive profile*.

Challenges: defining the predicted cognitive profile involves multiple complexities. Given a specific profile representation, it is necessary to define the profile dynamics when new information is provided. The system must perform minimal changes to the profile to accommodate this new information. Such minimal change requires the application of BR techniques adapted to the profile representation languages identified in Step 1.

System Calibration: this component describes the calibration of the system when comparing the predicted profile with newly acquired data.

9. A new NPA is conducted upon the completion of a training session, at which point the cycle restarts. The newly ACP is compared with the predicted cognitive profile generated by the system. Should these differ, the system analyses the possible causes of the divergence and (re)calibrates the parameters adequately.

Challenges: these divergences may originate from several sources: 1) an incorrect prediction in Step 8; 2) an inaccurate model of the relationship between tasks and cognitive domains in Step 5; or 3) a suboptimal adjustment of parameters in Step 7.

3.1.1 The Role of Artificial Intelligence

As previously elucidated, the implementation of the proposed framework entails several challenges. This subsection briefly describes the components and AI techniques utilised to address these requirements.

The first application of AI occurs during the creation of the profile ACP (see (2) in **Figure 2**), which is distinguished by two separate phases. To initiate the process in the absence of initial data, a focus group of six experts in neuropsychological assessment and rehabilitation conducted an empirical analysis of the NPAs to aggregate them by cognitive domain. This aggregation, based upon a weighted sum, was subsequently verified utilising belief merging and judgement aggregation procedures (for an overview, see [176,177]) and compared against correlations between NPAs established via ML techniques using available data for AD.

In the second phase, with a fully operational system and sufficient data collection, the weights assigned to each NPA regarding specific cognitive domains and subdomains will be calibrated using ML techniques. This process is explained in detail in **Section 3.2** (The Profiling Challenge). A similar approach is employed to aggregate the relationship between each NPA and the ND to obtain a consolidated CS.

The second application of AI methodology serves to optimise the process of CTT selection from the repository ((5) in **Figure 2**).

AI is further utilised to determine the difficulty level of a CTT during a training session ((7) in **Figure 2**). The CTT parameters must be adapted to ensure the performance of the patient remains between 50% and 70%.

The overall training performance recorded throughout the 12-session intervention provides the input for calculating the Predicted Cognitive Profile (P) ((8) in **Figure 2**). The process of relating CTT difficulty to cognitive domains is non-linear, primarily because a single CTT may train multiple domains. Consequently, it can be difficult to differentiate the extent to which a specific performance result relates to each individual domain.

To create P , the performance of the patient is transformed into a new entity expressed as a sentence in a formal language (see Subsection 3.2.1). This serves as the input for an update function that actualises the profile by applying a *minimal change* to incorporate the new information. The

algorithm for this update is based upon the theoretical framework for profile dynamics developed in [40].

Finally, the final AI challenge involves system calibration. As mentioned in Challenge 9, divergences may originate from various sources. In this context, data is collected to identify the origin of these divergences via ML approaches.

3.2 The Profiling Challenge

The Profiling Challenge corresponds to Steps 1, 2, and 3 of the framework illustrated in **Figure 2**.

3.2.1 The Structure of the Profile

This subsection identifies the aspects of cognitive, functional, and emotional domains relevant to the patient profile and the NPA instruments representative of those aspects. The process begins with the definition of a formal profile.

Definition 1. [40] *A profile P is a tuple $\ll label_1, \dots, label_n \gg$, where $label_i \in \mathbb{N}_0$.*

Informally, each element of the tuple within a profile is a characteristic that assumes a finite number of possible values. Whilst natural numbers have been utilised for the content of each $label_i$, the definition may be readily adapted to use linguistic labels; for instance, if $label_1$ represents marital status, values such as "single", "married", "separated", or "widowed" may be employed.

A simple example of a profile structure is $\ll \text{age, weight, height} \gg$, and a potential profile instance is $\text{John} = \ll 20, 80, 178 \gg$.

The subsequent stage involves defining a formal language to express profile properties, compute relevant metrics, and define profile dynamics.

Definition 2. [40] *A profile language is a finitary language $\mathcal{L}$, defined in the following manner:*

X is a term if and only if:

1. *X is a label.*
2. *If X is a term, then $-X$ is a term.*
3. *If X and Y are terms, then $X + Y$ is a term.*

An atom is an expression of one of the following forms:

1. $X = n$,
2. $X < n$,
3. $X > n$,

where X is a term and $n \in \mathbb{N}_0$. A well-formed formula (wff) is defined as:

1. *An atom is a wff.*
2. *If X is a wff, then $\neg X$ is a wff.*
3. *If X and Y are wffs, then $X \wedge Y$ is a wff.*

where $\neg$ (negation) and $\wedge$ (conjunction) are the classical negation and conjunction connectives.

The operators $\leq$ and $\geq$ are defined in the standard manner, as are the classical connectives $\vee$ (disjunction), $\rightarrow$ (implication), and $\leftrightarrow$ (equivalence). Returning to the previous example, “weight $\geq 90 \wedge$ height ≤ 180 ” constitutes a wff.

Following the definition of the profile structure and language, the subsequent stage involves defining the contents of a profile within the NeuroAIRh@b framework. Consequently, the integration of relevant NPA instruments, utilising specific weights for each domain and subdomain, is essential for a comprehensive evaluation of cognition (see **Table 10**). For instance, screening tests such as the Montreal Cognitive Assessment (MoCA) serve as brief, multidimensional instruments to identify patients at cognitive risk who require a more extensive evaluation. A domain and subdomain analysis identified the following NPA dimensions within the MoCA: general cognition; orientation; immediate verbal memory; executive functions (specifically working memory, processing speed, verbal fluency, inhibition, and visuoconstructive capacity); language (including comprehension and expression); and sustained attention.

Demographic variables, such as education and age, were considered during the analysis of NPA instrument results. The rationale for this selection recognises the impact of these variables in explaining result variance and defining normative benchmarks for test scores. From a rehabilitation perspective, it is essential to consider further variables, such as household composition and occupation.

Table 10: Expert-defined mapping of NPA instruments and sub-scores to the cognitive, functional, and affective domains and subdomains used to construct the NeuroAIREh@b patient profile. Percentages outside parentheses are the initial weights assigned to an instrument or sub-score, whereas percentages inside parentheses are the corresponding values after normalisation so that the contributions within each domain or subdomain sum to 100%. The initial mapping was elicited from a focus group of six neuropsychological assessment and rehabilitation experts and accounts for instruments that contribute to more than one domain. An asterisk identifies cases in which the instrument is selected according to the patient's age.

General cognition			MoCA total 100% (50%)		CDR cognitive cluster 100% (50%)	
Orientation			CDR orientation 33,33% (72,73%)		MoCA temporal and spatial orientation 12,5% (27,27%)	
Memory	Immediate	Verbal	FCSRT immediate memory 100% (68,57%)	CDR immediate memory 33,33% (22,86%)	MoCA delayed recall 12,5% (8,57%)	
		Visual	FCRey 3min 100% (50%)		WMS-III Visual Reproduction immediate recall 100% (50%)	
	Delayed	Verbal	FCSRT delayed recall 100% (75%)		CDR memory delayed recall 33,33% (25%)	
		Visual	WMS-III visual reproduction delayed recall 100% (50%)		WMS-III visual reproduction recognition 100% (50%)	
Executive functions	Working memory		WAIS-III Digit Symbol Coding 50% (68,54%)	MoCA digit in reverse 12,5% (17,13%)	MoCA calculus 6,25% (8,57%)	MoCA target detection 4,2% (5,76%)
	Processing speed		WAIS-III Symbol Search 50% (36,36%)	WAIS-III Digit Symbol Coding 50% (36,36%)	Phonemic and Semantic Verbal Fluency 33,33% (24,23%)	MoCA Phonemic Verbal Fluency 4,2% (3,05%)
	Verbal fluency		Phonemic and Semantic Verbal Fluency 33,33% (88,81%)		MoCA phonemic verbal fluency 4,2% (11,19%)	
	Inhibition		Phonemic and Semantic Verbal Fluency 33,33% (79,88%)		MoCA target detection 4,2% (10,06%)	MoCA phonemic verbal fluency 4,2% (10,06%)
	Visuoconstructive capacity		FCRey copy 100% (50%)		WMS-III Visual Reproduction total score 100% (50%)	
	Language	Comprehension	WAIS-III Vocabulary 50% (100%)			
Expression		MoCA naming and repetition 12,5% (100%)				
Attention	Divided	WAIS-III Symbol Search 50% (100%)				
	Sustained	Toulouse-Piéron test 100% (81,33%)	MoCA target detection 4,2% (3,42%)	MoCA calculus 6,25% (5,08%)	MoCA digit direct 12,5% (10,17%)	
Premorbid intelligence			WAIS-III Vocabulary 50% (100%)			
Functionality	Basic ADLs	IAFAI Basic ADLs 100% (50%)		CDR personal care 100% (50%)		
	Instrumental - Familiar ADLs	Instrumental - Familiar ADLs 100% (50%)		CDR home and hobbies 100% (50%)		
	Instrumental - Advanced ADLs	Instrumental Advanced ADLs 100% (33,33%)	CDR community affairs 100% (33,33%)	CDR judgment and problem solving 100% (33,33%)		
	Cognitive deficits perceived impact	PRECiS or SMC* 100%				
Depressive symptomatology			GDS-30 or BDI-II* 100%			

The results obtained from a comprehensive battery of NPA instruments provide a baseline of impaired cognitive functions, which facilitates the definition of the duration and type of CR required. For instance, deficits in executive functions—specifically regarding inhibition, planning, and monitoring—necessitate intervention programmes focused upon executive functioning. Furthermore, results obtained from memory tests contribute to the personalisation and adaptation of CR sessions; these may involve the use of calendars, notepads, and alerts, as well as the teaching and training of mnemonics, face-name associations, and the improvement of episodic, semantic, autobiographical, and visual memory.

In addition to the MoCA screening instrument, the following multidimensional battery of NPA instruments was established to construct a profile within NeuroAIReh@b: the Clinical Dementia Rating (CDR); Subjective Memory Complaints (SMC); the Free and Cued Selective Reminding Test (FCSRT); Visual Reproduction from the Wechsler Memory Scale, Third Edition (WMS-III); Semantic and Phonemic Verbal Fluency; the Toulouse-Piéron test; the Digit Symbol Coding, Symbol Search, and Vocabulary subtests from the Wechsler Adult Intelligence Scale, Third Edition (WAIS-III); Rey’s Complex Figure Test (FCRey); the Adults and Older Adults Functional Assessment Inventory (IAFAI) or the Patient-Reported Evaluation of Cognitive State (PRECiS); and the Geriatric Depression Scale-30 (GDS-30) or the Beck Depression Inventory-II (BDI-II). This selection was based upon the following criteria: 1) NPA instruments that are standard and widely utilised within Portuguese clinical and research contexts; 2) instruments with the specificity required to detect impairments in the cognitive, functional, and emotional dimensions of stroke, MCI, and dementia populations; and 3) instruments for which ND for the Portuguese population are available.

3.2.2 Aggregation of Neuropsychological Assessments

To construct the profile ACP, a bridge was established between the various NPAs and the SDI of the patient. To the best of our knowledge, a gap exists in the literature regarding the integration of data from multiple, heterogeneous NPA instruments to create a consolidated profile ACP.

As previously stated, the initial phase of the research lacked sufficient data concerning stroke, MCI, and dementia patients to utilise an ML approach for quantifying the contribution of each NPA to the various dimensions of the profile. Consequently, the following strategy was developed: (1) a focus group of six neuropsychological assessment and rehabilitation experts defined a general

formula for aggregating the NPAs, assigning weights to the relationship between NPA instruments and cognitive domains or subdomains; (2) the NPA instruments were empirically aggregated based upon expert experience, providing an initial set of weights; (3) this aggregation was subsequently pre-validated utilising correlations obtained from patients with dementia, and weights were readjusted accordingly; and (4) ML algorithms were defined for future system calibrations.

3.2.3 The General Formula

A methodology is proposed to map NPA instrument scores to a consolidated ACP within the interval $[0, 100]$; Equation 1 is utilised for this *Mapping* process. It should be noted that this formula was developed in the absence of initial empirical data, relying instead upon expert-derived heuristics.

$$ACP_k = \sum_{i=1}^n \sum_{j=1}^m Norm(NPA_i Dom_j) \cdot W_{kij} \quad (1)$$

In this context, m represents the frequency with which a cognitive domain (Dom) appears within an NPA instrument, and n denotes the total number of NPA instruments. A_k signifies the cognitive domain k for this profile ACP, while $NPA_i Dom_j$ refers to the domain or subdomain j of the NPA tool i . $Norm$ is a normalisation function, the interval of which is set between 0 and 100. Finally, W_{kij} represents a weight value within the interval $[0, 1]$, where $\sum w_{kij} = 1$. The profile ACP for a patient is therefore defined as:

$$ACP = \ll ACP_1, \dots, ACP_r \gg$$

where r is the total number of domains and subdomains considered.

For instance, utilising the weights for working memory (wm) provided in **Table 10**, the following calculation is obtained:

$$\text{Working_Memory} = \begin{cases} \text{WAIS-III_DSC}_{wm} \times 0.6854 \\ + \text{MoCA_digit}_{wm} \times 0.1713 \\ + \text{MoCA_calculus}_{wm} \times 0.0857 \\ + \text{MoCA_target}_{wm} \times 0.0576 \end{cases}$$

The ACP is not, in isolation, sufficient to determine the CS of a patient. To obtain this, a comparison with ND is required. The ND is organised according to the SDI of the patient. By interpreting the ACP using the ND from all NPAs whilst accounting for the specific SDI of the individual, a contextualised representation is achieved. The subsequent stage involves resolving how to contextualise this profile effectively. The following formula is proposed:

$$S_k = \sum_{i=1}^n Norm(ND_i, SDI) \cdot w_i \quad (2)$$

In this equation, n represents the number of NPA instruments. S_k denotes the normative value for the domain or subdomain k derived from the ND and SDI. $Norm$ is a normalisation function in which the interval ranges from 0 to 100. ND_i refers to the profile placement within the ND of k for each NPA tool i utilised to calculate the ACP, while SDI represents the SDI of the patient. Finally, w_i signifies the weight value of the ND_i for domain k within the interval $[0, 1]$, where $\sum w_i = 1$.

First, it is essential to note that access to normative values for individual NPA tasks or questions is often restricted; such data are generally only available for the total and, in certain instances, the sub-totals of each NPA instrument.

A weight must be assigned to the ND–SDI pair to facilitate the aggregation of results across all involved NPA instruments during summation. This pair provides the average performance on a specific NPA for an individual belonging to the same socio-demographic group as the patient (e.g. 65–70 years of age with 12 years of education). As the MoCA ND are available for the Portuguese population [178], these are utilised here as an example within the memory domain. The result of this formula represents the average performance expected for an individual with an SDI similar to that of the patient. By performing a simple cross-multiplication between the memory score in ACP and the output of this function, the relative value of the patient compared to the ND can be determined. In this example, this value constitutes the memory domain within the CS of the patient, where the 50th percentile represents an average performance relative to the socio-demographic group of the patient in the memory task.

The primary challenge in this context involves determining the value of weight W_{kij} in Equation 1 and the value of weight w_i in Equation 2.

3.2.4 Commencing in the Absence of Data

Table 10 depicts the weighting of each NPA instrument across the various cognitive domains and subdomains. As previously stated, this aggregation of NPA instruments resulted from a focus group session involving six experts in neuropsychological assessment and rehabilitation. Given that most selected NPA scores and sub-scores contribute to the evaluation of multiple cognitive domains, the weight of each instrument was divided by the number of sub-scores it contains and the number of cognitive domains and subdomains it targets.

For instance, the MoCA (100%) is a cognitive screening measure providing information regarding general cognitive functioning. Through eight of its sub-tests (12.5% each), this multidimensional tool contributes to the assessment of various domains: MoCA calculus (12.5%) contributes to the assessment of executive functions within the working memory subdomain (6.25%) and attention within the sustained attention subdomain (6.25%).

To ensure each domain sums to a total of 100%, these values were normalised. Consequently, if a subdomain contains only one NPA tool entry weighted at 50%, it is normalised to 100%. Should no score be provided for one or more subdomains within the NPA instruments, the NeuroAIReh@b system normalises the existing scores in accordance with the non-normalised values.

These empirical values were validated through two distinct methods. First, it was verified whether the weighted sum for the aggregation of NPAs adhered to basic aggregation rules (see [176, Chapter 6]). Second, the values were compared against data from external sources, specifically the Alzheimer’s Disease Neuroimaging Initiative (ADNI) database. The ADNI is a longitudinal, multi-site observational study of elderly individuals with normal cognition, MCI, and AD. As it includes a battery of NPAs, ML techniques were employed to identify correlations between the different instruments. These correlations were then compared with empirical correlations to establish a functional analogy. To obtain the aggregations for AD, the following procedure was adopted:

1. A file was prepared from the database in which all primary tables were merged. Consequently, this file contains the totals from all key tables and the comprehensive assessment results. It includes the diagnosis for each patient, filtered according to the NPAs utilised within the NeuroAIReh@b framework.

2. Records containing incomplete data were depurated and removed from the dataset, resulting in a refined dataset of approximately 2,200 entries. To clean the datasets prior to model generation, the Variance Threshold, Pearson Correlation, and Analysis of Variance were evaluated.
3. The correlation between tests was verified utilising the following ML algorithms: Linear Regression, Logistic Regression, Decision Tree, Support Vector Machine, Naïve Bayes, K-Nearest Neighbors, and Random Forest. **Figure 3** provides an example of the correlations obtained; a comprehensive description is available in [179]. These correlations identified which NPA instruments are capable of assessing specific domains (e.g. memory or depressive symptomatology).

Once sufficient data have been collected, the weights W_{kij} in Equation 1 and w_i in Equation 2 will be calibrated. To compute these weights, various statistical learning, ML, or DL techniques—such as PCA, Random Forest, or Neural Networks—will be applied to the NPA instrument data to obtain highly optimal values for W within the *Mapping* process.

Furthermore, should the number of domains (*Dom*) and NPA instruments increase over time, system performance may decline, and the data may suffer from the "curse of dimensionality". To mitigate such issues, feature selection and dimensionality reduction techniques, including Principal Component Analysis (PCA), LASSO, Ridge regression, or t-Distributed Stochastic Neighbour Embedding, may be utilised. These techniques identify highly influential parameters while preserving the integrity of the underlying information.

3.3 The Training Challenge

Although the training phase of this project was addressed in a separate publication [148], it is nevertheless important to summarise the adopted procedure here to provide the necessary context for the methodological modifications proposed later in this work¹.

The training challenge comprises the definition and development of CTTs, followed by their subsequent selection and personalisation according to a given cognitive objective (see (5) in **Figure 2**). All CTTs are integrated and orchestrated by the Reh@bSync software [180], which manages patient data, optimises task selection based upon cognitive profiles, adapts difficulty both within and across sessions, and provides the user interface for training interaction.

¹For an in-depth and comprehensive treatment of this topic, the reader is referred to Paulino's doctoral thesis [180].

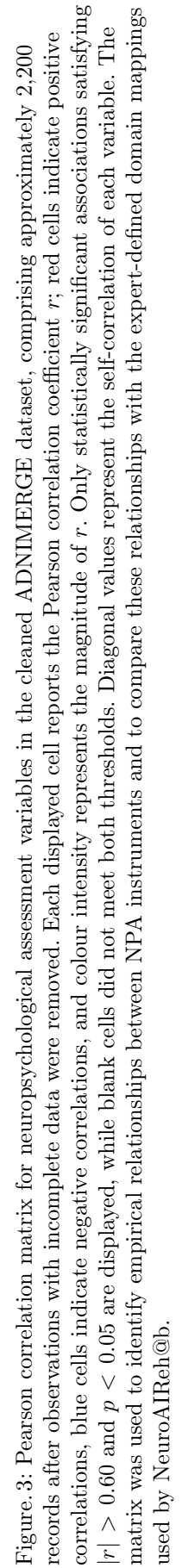


Figure 3: Pearson correlation matrix for neuropsychological assessment variables in the cleaned ADNIMERGE dataset, comprising approximately 2,200 records after observations with incomplete data were removed. Each displayed cell reports the Pearson correlation coefficient r ; red cells indicate positive correlations, blue cells indicate negative correlations, and colour intensity represents the magnitude of r . Only statistically significant associations satisfying $|r| > 0.60$ and $p < 0.05$ are displayed, while blank cells did not meet both thresholds. Diagonal values represent the self-correlation of each variable. The matrix was used to identify empirical relationships between NPA instruments and to compare these relationships with the expert-defined domain mappings used by NeuroAIR_{eh}@b.

The repository of CTTs was grounded upon empirical input from stroke patients and neuropsychologists, with particular emphasis on cognitively demanding Instrumental Activity of Daily Living (IADL) identified through functional assessment procedures [181, 182]. The resulting tasks prioritise ecological validity by simulating daily cognitive IADLs—understood as complex, real-world problem-solving activities [183]—which typically involve multitasking and the integration of multiple cognitive processes [184, 185]. Task design and operationalisation followed participatory and iterative methods, including structured brainstorming and braindrawing techniques [186], and were informed by previous serious game platforms [159, 160, 187]. Contextual interdependencies between activities were explicitly modelled to enhance ecological coherence and potential transfer effects [188, 189].

During execution, Reh@bSync matches the cognitive profile of each patient with the domain coverage of the CTTs utilising a distance-based mechanism, parameterises tasks accordingly, and supervises adaptive difficulty regulation. This adaptation extends previous work [187] by enabling task-specific difficulty adjustments within predefined challenge thresholds (50%–70%), aiming to maintain patients in an optimal engagement state consistent with flow theory [190].

Session performance data are synchronised with the NeuroAIRh@b server to update the cognitive profile iteratively, thereby closing the personalised loop.

For a comprehensive, implementation-oriented, and technically detailed description of all aspects pertaining to Reh@bSync, the reader is referred to [180].

3.4 Profile Dynamics

As previously noted, training tasks adapt parameters to maintain performance within a predefined range (50%–70%). Should a specific parameter of a CTT undergo an increment in value during training sessions, the patient manifests an improvement in their CS as a result of the rehabilitation process.

The performance levels recorded at the conclusion of a training session via the CTTs are utilised to estimate an intermediary virtual profile. This profile serves as the input for the subsequent session, thereby enabling the CTTs to be adapted to the specific requirements of the patient. However, the process of relating CTT difficulty to cognitive domains is non-linear, as a single CTT

usually addresses multiple cognitive domains. Consequently, it can be challenging to differentiate the extent to which a recorded performance relates to each specific domain or subdomain.

To facilitate the establishment of this relationship, NeuroAIReh@b creates and maintains a correlation for each CTT between its parameters and the relevant domains or subdomains. Utilising these correlations alongside the performance data from all CTTs within a session, the system summarises the outcome as a sentence in the formal language defined in Definition 2. This sentence acts as the input for the profile update. Hence, all combinations of different domains that could produce that specific performance level are considered.

By analysing these various combinations, the system selects the model that displays the minimal distance from the previous cognitive profile. The weights of the tasks (the CTT models) serve as a tiebreaker criterion when required (see [40] for the detailed theoretical methodology). The resulting profile serves as the input for further training sessions; it is utilised in the adaptation of CTTs to ensure they remain parameterised at the ideal level of difficulty throughout an iteration of multiple training sessions, referred to here as a training programme. At the conclusion of each CT programme, the patient is reassessed, and the cognitive user profile is compared against the estimated baseline profile.

The clinical validation of the NeuroAIReh@b prototype demonstrated promising results across multiple pilot studies involving stroke survivors. Initial findings from a one-month intervention showed significant improvements in general cognition and functional abilities, with evidence suggesting that training gains were generalised to activities of daily living due to the high ecological validity of the tablet-based tasks [191]. Subsequent feasibility trials for a blended neurorehabilitation program reported high levels of participant compliance and satisfaction, further confirming that the platform is a viable and acceptable tool for both in-person and remote CT [192]. These results served as a foundation for developing a difficulty progression algorithm designed to automate task personalization based on real-time participant performance.

Although AI serves as a powerful supplement to clinical judgement, it is intended to enhance rather than replace health professionals by reducing evaluation errors and optimising treatment decisions. The integration of ML allows for a dynamic update of a patient’s cognitive profile, ensuring that the training challenge remains aligned with their evolving capabilities. However, several limitations remain, including the current reliance on algorithms derived from AD data which

may not perfectly translate to stroke recovery. Additionally, the use of screening tools like the MoCA for domain-specific weighting and the "black box" nature of the underlying AI models necessitate further validation through ongoing large-scale randomised controlled trials. For a comprehensive account of the results, critical discussion, and limitations of NeuroAIReh@b, the reader is referred to Faria et al. [148].

3.5 Contribution to NeuroAIReh@b

The present thesis contributed to the development of the NeuroAIReh@b framework through the design and implementation of several foundational infrastructural components that support the operationalisation of the framework described in this section. These contributions encompass the data persistence layer, the administrative interface for clinical and research personnel, the programmatic interface for system interoperability, and the computational procedures for deriving cognitive profiles from neuropsychological assessment data.

Firstly, a relational database was designed and implemented to serve as the central repository for all information required by the NeuroAIReh@b framework (shown in appendix 13). This database stores the heterogeneous data necessary for the four principal components of the framework: Patient Profiling, Training Selection, Training Session, and System Calibration (as illustrated in **Figure 2**). The schema accommodates SDI, raw and normalised scores from the battery of NPA instruments, ND for contextualisation, cognitive profiles (both assessed and predicted), training objectives, CTT metadata and parameters, and session performance records. By consolidating these data within a structured relational model, the system ensures data integrity, supports the longitudinal tracking of patient evolution, and facilitates the aggregation and comparison operations required for profile creation and system calibration.

Secondly, a web-based dashboard was developed to enable clinicians and researchers to view and modify the information stored within the database. This dashboard provides a unified interface for managing the complete lifecycle of the NeuroAIReh@b workflow: entering and editing NPA results, socio-demographic data, and assessment parameters (corresponding to Step 1 of the framework); viewing and adjusting computed cognitive profiles and training objectives (Steps 4–6); and monitoring training sessions and performance metrics (Steps 7–9). The dashboard thus serves as the primary human–machine interface for the framework, ensuring that all information required for the NeuroAIReh@b pipeline is accessible and manageable without the necessity for direct database manipulation.

Thirdly, the algorithms for generating the user profile from NPA data were implemented and integrated into the system. These algorithms operationalise the formulae presented in Section 3.2: specifically, the aggregation of NPA instrument scores into an ACP via Equation 1, and the contextualisation of the ACP against ND to derive the CS via Equation 2 (Steps 2–3 of the framework).

The NPA data is stored within the database, and the profile creation algorithms execute upon these stored records to produce the baseline profile P_0 , which serves as the primary input for the training programme. The implementation incorporates the domain weights and mapping procedures established through expert consultation and validated against external datasets, as described in Section 3.2.4.

Fourthly, an API was implemented to enable Reh@bSync [180] —the software responsible for orchestrating CTTs and managing patient sessions—to retrieve and modify user profiles. This API is essential for the operational integrity of Reh@bSync, which must obtain the current cognitive profile of each patient to facilitate the selection and parameterisation of tasks (Steps 5–6). Furthermore, the system must submit updated profiles following each training session to reflect performance-derived changes (Step 8). The API exposes endpoints for profile retrieval, profile updates, and the synchronisation of session data, thereby closing the loop between the training component and the profile dynamics component of the framework.

Collectively, these contributions provide the data layer, administrative interface, interoperability layer, and profile creation logic that underpin the NeuroAIRh@b framework. They complement the theoretical and algorithmic contributions described in subsequent chapters—particularly the KDPD framework for session-by-session profile revision—by establishing the infrastructure upon which the complete rehabilitation cycle can be executed.

4 Implementing Knowledge-Driven Profile Dynamics

This Chapter introduces the conceptual and methodological foundations underpinning the study, outlining the key theoretical constructs, analytical approaches, and empirical considerations that guide the subsequent investigation.

4.1 Theory

This section introduces the formal framework that serves as the basis for the implementation. Building upon the theoretical foundations developed by us in the KDPD framework [40], the structures and logical apparatus employed are formally specified. In particular, the core definition of a profile, as presented in the KDPD work, is adopted. Subsequently, a specialised profile language, $\mathcal{L}_p$, is introduced; this extends the basic KDPD language, as suggested therein, by incorporating arithmetic and relational operators, whilst simultaneously restricting wffs to Disjunctive Normal Form (DNF) to align with specific algorithmic requirements. Finally, the Manhattan distance metric used to quantify profile similarity is defined, clarifying its relationship to the distance measure discussed in KDPD and specifying its role in the ordering mechanisms crucial for the profile change operators.

4.1.1 Profile Definition

As introduced in [40], a profile is defined as follows:

Definition 3. *Let $\mathbb{L} = \langle\langle L_1, L_2, \dots, L_n \rangle\rangle$ be a tuple of labels. For each $i \in \{1, \dots, n\}$, let D_i be a finite set associated with label L_i , designated as the domain of L_i . A profile, associated with $\mathbb{L}$ and denoted by $P_{\mathbb{L}}$ (or simply by P if the tuple of labels is clear from the context), is an element of $D_1 \times D_2 \times \dots \times D_n$. The set of all profiles associated with $\mathbb{L}$ is denoted by $\mathbb{P}_{\mathbb{L}}$ (or simply by $\mathbb{P}$).*

Within this framework, the operative set of labels is defined as the ordered sequence $\mathbb{L} = \langle\langle L_1, L_2, \dots, L_n \rangle\rangle$. Each label L_i , for $1 \leq i \leq n$, is associated with a specific domain, $D(L_i)$, which is strictly defined as a discrete, ordered set composed exclusively of integers. Furthermore, every domain is bounded, being delimited by a defined minimum ($\min_i$) and maximum ($\max_i$) integer

value. Formally, $D(L_i)$ is a subset of the integers within this closed interval; that is, $D(L_i) \subseteq \{k \in \mathbb{Z} \mid \min_i \leq k \leq \max_i\}$.

This specification represents a practical specialisation of the more general theoretical framework presented in the KDPD proposal, wherein label domains could encompass arbitrary alphanumeric values. For the purposes of this implementation, all profile attributes are constrained to integers within a defined, bounded range. This decision is a necessary and deliberate step to operationalise the core concepts of the framework.

By ensuring all domains are numerical, the use of well-defined distance metrics is enabled—specifically the Manhattan distance (detailed in Section 4.1.3)—which is fundamental to the revision and update operators. This numerical foundation also underpins the constraint satisfaction and domain-narrowing algorithms utilised by the system, which are designed to operate on integer ranges. Rather than a theoretical departure, this constraint should be understood as a concrete instantiation of the KDPD model, tailored to achieve computational tractability and to support the specific distance-guided search algorithms central to the implementation.

Example 1. Given the tuple of labels $\mathbb{L} = \langle L_1, L_2, L_3, L_4 \rangle$ with domains $D_0, D_1, D_2, D_3 \subseteq \{k \in \mathbb{Z} \mid 0 \leq k \leq 5\}$, the following are examples of profiles:

$$\langle\langle 3, 0, 1, 5 \rangle\rangle$$

$$\langle\langle 5, 5, 3, 2 \rangle\rangle$$

4.1.2 Profile Language

For the purpose of implementation and defining the dynamics of specific profiles, a formal language, denoted $\mathcal{L}_p$, is introduced. This language is based upon the one proposed in the foundational work on KDPD, which explicitly suggested the possibility of enriching its basic language (e.g. with comparisons such as $L_i > n$). The language $\mathcal{L}_p$ formalises such extensions by incorporating arithmetic and relational operators to express richer conditions on profile attributes, whilst also restricting the structure of wffs to DNF to facilitate processing. As previously stated, the domains D_i associated with each label L_i are restricted to subsets of integers ($\mathbb{Z}$).

Definition 4. *Two sets of operators are defined:*

- **Arithmetic Operators** (op_{arith}): This set includes addition (+), subtraction (−), multiplication (*), division (/), and modulo (%). These operators function according to their standard mathematical definitions. Division and modulo assume a non-zero divisor or modulus.
- **Relational Operators** (op_{rel}): This set includes equality (=), less than (<), greater than (>), less than or equal to ($\leq$), and greater than or equal to ($\geq$). These operators represent standard numerical comparisons.

Definition 5. (Alphabet) Given a tuple of labels $\mathbb{L} = \langle\langle L_1, L_2, \dots, L_n \rangle\rangle$, where each label L_i has an associated domain $D_i \subseteq \mathbb{Z}$, the alphabet of symbols for the language $\mathcal{L}_p$ consists of:

1. Labels: $L_1, L_2, \dots, L_n$, where $n \in \mathbb{N}$;
2. Constants: Any real number $(a, b, \dots \in \mathbb{R})$;
3. Arithmetic Operators: +, −, *, /, %;
4. Relational Operators: =, <, >, $\leq$, $\geq$;
5. Logical Connectives: $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction);
6. (,) (punctuation symbols).

Definition 6. (Terms) A Term (T) in $\mathcal{L}_p$ is defined recursively as follows:

1. A constant $a \in \mathbb{R}$ is a term;
2. A label L_i is a term;
3. If T_1 and T_2 are terms and $op \in op_{arith}$, then $(T_1 \text{ op } T_2)$ is a term. The type of the resulting term is determined by standard arithmetic rules.

Parentheses may be omitted where operator precedence rules ensure clarity (e.g. $L_1 + 5 + 3$ instead of $((L_1 + 5) + 3)$).

Definition 7. (Atomic Formulae) Let $\mathbb{L} = \langle\langle L_1, L_2, \dots, L_n \rangle\rangle$ be a tuple of labels. For each $i \in \{1, \dots, n\}$, let D_i be the domain associated with the label L_i . An atomic formula of $\mathcal{L}_p$ is defined as follows:

- $T_1 \text{ op } T_2$

where T_1 and T_2 are terms and $op \in op_{\text{rel}}$. This formula asserts a relationship between the values represented by the terms. The comparison is well-defined, as both terms evaluate to numerical values (integers or reals).

Definition 8. (*Literals*) A literal (l) is either an atomic formula A or its negation $\neg A$.

Definition 9. (*Well-Formed Formulae*) A wff of $\mathcal{L}_p$ is restricted to the DNF structure, defined as follows:

1. A conjunction of literals $C = (l_1 \wedge l_2 \wedge \dots \wedge l_k)$, where $k \geq 1$ and each l_k is a literal, is a **conjunctive clause**;
2. A disjunction of conjunctive clauses $\phi = (C_1 \vee C_2 \vee \dots \vee C_m)$, where $m \geq 1$ and each C_j is a conjunctive clause, is a **wff**.

A single literal or conjunctive clause is also considered a wff, corresponding to $m = 1$ or $m = 1, k = 1$, respectively.

The restriction of wffs to DNF is a deliberate design choice motivated by considerations of algorithmic efficiency and implementation clarity. It is a well-established principle of propositional logic that any Boolean expression can be systematically converted into an equivalent DNF representation; thus, this restriction does not limit the logical expressiveness of the language. This structure allows the system to treat a disjunction as a collection of independent subproblems, where each conjunctive clause can be processed separately by the core search algorithm.

The `findTuplesWithConstraints` function leverages this by splitting the wff string at the top-level disjunction operator (`"||"`), as detailed in Section 4.2.4. Each resulting conjunctive clause is then handled by a specialised procedure, `_findTuplesForConjunction`, which is optimised for constraint propagation and domain narrowing on a set of purely conjunctive constraints. This approach avoids the significant complexity of implementing a general-purpose solver for arbitrary Boolean formulae, which would necessitate more intricate parsing and a recursive search algorithm capable of handling nested logical structures. By enforcing DNF, a more tractable and performant search process is achieved, directly aligning with the focus of the implementation on efficient, minimal-change profile dynamics.

Example 2. Given the labels $\mathbb{L} = \ll L_1, L_2, L_3 \gg$ and $D_{1\dots 3} = \{1, 2, 3, 4, 5\}$, with integer domains $D_i \subseteq \mathbb{Z}$, the following are examples of formulae within the language:

- **Terms:** $L_1, 5.3, L_1 + L_2, L_3 \times 2.5$;
- **Atomic Formulae:** $L_1 > 5.3, L_1 + L_2 = L_3, L_3 \times 1.5 \leq 10$;
- **Literals:** $L_1 > 2, \neg(L_1 + L_2 = L_3)$;
- **Conjunctive Clause:** $L_1 > 2 \wedge \neg(L_1 + L_2 = L_3)$;
- **wff:** $(L_1 > 5 \wedge \neg(L_2 = 0)) \vee (L_3 \leq 10)$.

4.1.3 Manhattan Distance

To operationalise the profile change operators, which rely upon selecting profiles that are 'closest' to an initial profile or set of profiles according to specific criteria, a distance metric between profiles is required. The KDPD paper introduces a distance measure (Def. 4.11) termed "Hamming distance" [40]. However, for the numerical domains considered in KDPD and utilised exclusively in this implementation (where $D_i \subseteq \mathbb{Z}$ for all i), the formula provided corresponds precisely to the standard definition of Manhattan distance. Therefore, for clarity and consistency with established mathematical terminology, the term Manhattan distance is adopted for this measure.

Definition 10. (*Manhattan Distance*) Let $\mathbb{P} = D_1 \times \dots \times D_i$ be the set of all possible profiles, where each $D_i \subseteq \mathbb{Z}$. For any two profiles $p = \ll p_1, \dots, p_n \gg \in \mathbb{P}$ and $q = \ll q_1, \dots, q_n \gg \in \mathbb{P}$, the Manhattan distance $d_M(p, q)$ is defined as:

$$d_M(p, q) = \sum_{i=1}^n |p_i - q_i|$$

The profile change operators evaluate proximity to a specific reference profile (for single-profile dynamics) or potentially a set of profiles. Let $r \in \mathbb{P}$ be a designated reference profile. The distance $d_M(p, r)$ quantifies the extent to which profile p differs from the reference r . This distance induces specific orderings on the set of profiles $\mathbb{P}$.

Definition 11. (*Distance-Based Total Pre-order*) Let $r \in \mathbb{P}$ be a reference profile. The Manhattan distance to r induces a total pre-order $\preceq_r$ on $\mathbb{P}$, defined as follows:

For any $p, q \in \mathbb{P}$,

$$p \preceq_r q \iff d_M(p, r) \leq d_M(q, r)$$

This signifies that p is at least as close to r as q . To ensure a total strict ordering, which is necessary for certain profile change operators (e.g. those selecting a unique minimal profile), a tie-breaking mechanism is introduced for cases where two distinct profiles p and q are equidistant from the reference profile r . This mechanism relies upon a user-defined **label priority order**, represented by a permutation π of the label indices $1, \dots, n$. Here, $\pi(1)$ denotes the index of the highest priority label, $\pi(2)$ the second highest, and so forth, down to $\pi(n)$. The tie is broken by comparing the distances of the individual components $|p_i - r_i|$ according to this priority order.

Definition 12. (*Distance-Based Total Strict Order with Priority Tie-breaking*) Let $r \in \mathbb{P}$ be a reference profile and let π be a defined label priority order. The total strict order $\prec_r$ on $\mathbb{P}$ is defined as follows:

1. $d_M(p, r) < d_M(q, r)$

OR

2. $d_M(p, r) = d_M(q, r)$ and there exists an index $k \in \{1, \dots, n\}$ such that:

- $|p_{\pi(k)} - r_{\pi(k)}| < |q_{\pi(k)} - r_{\pi(k)}|$ **AND**

- For all $j \in \{1, \dots, k-1\}$, $|p_{\pi(j)} - r_{\pi(j)}| = |q_{\pi(j)} - r_{\pi(j)}|$

This means that p is strictly closer to r than q is, either because its total Manhattan distance is smaller or because the distances are equal but p is closer with respect to the highest priority label where their individual component distances differ. This definition ensures a total strict ordering over distinct profiles.

4.2 System Architecture and General Implementation

The system presented here is engineered to operationalise the KDPD process. Its central objective is the revision and updating of user profiles in accordance with specified constraints—articulated as wffs—and bounded value parameters. The architectural design is modular, comprising several principal components that interact synergistically to achieve the objectives of the system. These components are as follows:

- **Tuple Finder Module (TFM)**: Searches for profile tuples that satisfy a given wff and minimise the Manhattan distance to a target profile.
- **Abstract Syntax Tree engine (ASTe)**: Parses wff strings into a custom AST, validates them, and generates executable check predicates.
- **Domain Management System (DMS)**: Represents and iteratively narrows the permissible value ranges for each profile label to reduce the search space.
- **Profile Dynamics Module (PDM)**: Orchestrates the selection of a KDPD revision model and delegates tasks to other components.

Figure 4 provides a high-level overview of this architecture, illustrating the interconnections and primary data flows between these core modules.

System Architecture Overview

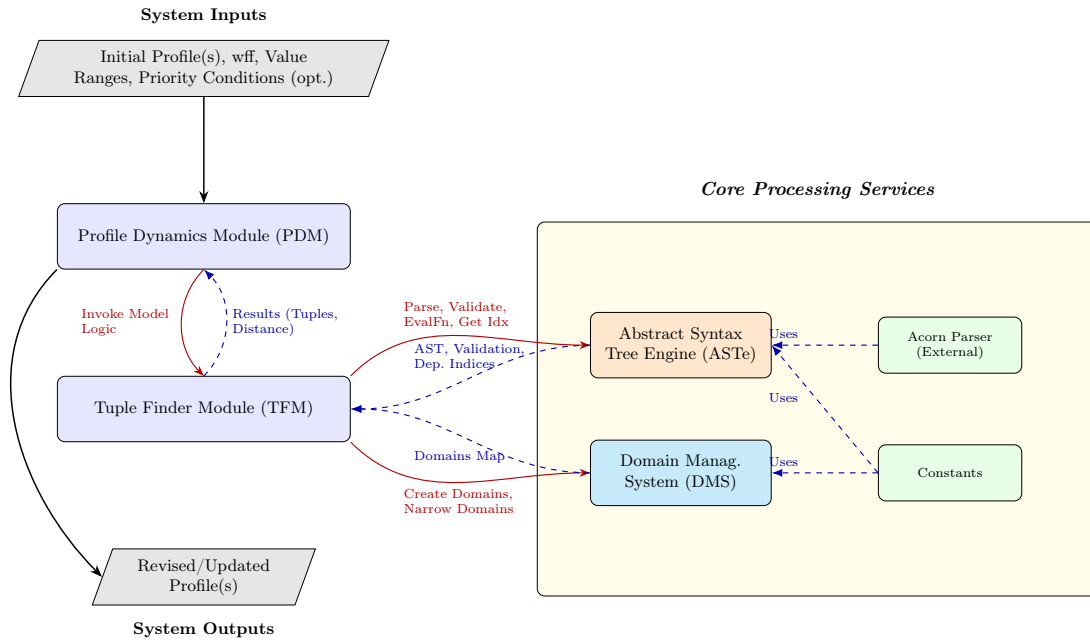


Figure 4: High-level architecture and principal data flows of the KDPD implementation. The system receives one or more initial profiles, a wff expressing the new evidence, bounded value ranges, and an optional priority condition. The PDM selects and orchestrates the applicable revision or update model. The ASTe, using the external Acorn parser, parses and validates the wff, identifies the dependent profile indices, and constructs an executable predicate. The DMS creates and iteratively narrows the permissible domains of the constrained indices. The TFM searches those domains for profiles that satisfy the wff while minimising Manhattan distance from the reference profile. The resulting minimum-distance tuple or tuples are returned to the PDM, which may apply priority-based tie-breaking before producing the revised or updated profile set.

Operationally, the system initiates its process upon receiving one or more initial profiles alongside a wff string and specified upper and lower bounds. This wff, as defined in Section 4.1, dictates the criteria for the revision or update of the input profile(s). Initially, the ASTe parses this wff into an AST and subsequently validates its structural and semantic integrity.

This validated AST then guides the TFM in its search for valid tuples. The TFM employs domain-narrowing techniques—a process informed by the AST and managed by the DMS—to reduce the combinatorial search space. Depending on the specific operational model selected within the PDM, the system returns all tuples that satisfy the wff at the minimum Manhattan distance or, in certain models, applies priority conditions to discern a single "best" tuple². This sequence of operations culminates in the generation of the revised or updated profile(s). The general flow of interaction is illustrated in the sequence diagram in **Appendix 11**.

4.2.1 Mapping Formal Definitions to Implementation Artefacts

The successful implementation of the KDPD framework requires the translation of its core theoretical constructs, as defined in Section 4.1, into concrete software artefacts. This subsection elucidates the mapping between these formal definitions—specifically, the profile structure, the specialised language $\mathcal{L}_p$, and the Manhattan distance metric—and their respective counterparts within the implemented codebase.

4.2.1.1 Profile Definition and Domain Representation

The formal definition of a profile as an n -tuple $P = \ll p_1, \dots, p_n \gg$, where each element p_i is an integer drawn from a bounded domain $D_i \subseteq \mathbb{Z}$ (see Section 4.1.1), finds its direct analogue in the internal representation of the system. Within the JavaScript implementation, a profile is manifested as an array of numbers. The conceptual designations L_i from the theoretical framework correspond to the zero-based indices of these arrays, wherein L_1 is mapped to index 0, L_2 to index 1, and so forth.

The global `minValue` and `maxValue` parameters, propagated through the revision functions, establish the overarching range of acceptable integers for all elements of the profile. The specific, potentially restricted, set of valid values for each constrained element L_i (its effective domain)

²The designation "best" refers to the tuple(s) demonstrating the lowest Manhattan distance relative to the target or initial profile(s) among those fulfilling the specified constraints. Within this particular model, "best" additionally incorporates a tie-breaking mechanism governed by a priority condition.

is managed by instances of the `Domain` class, particularly during the domain narrowing phase orchestrated by the DMS (detailed in Section 4.2.3).

4.2.1.2 Profile Language $\mathcal{L}_p$ and its Processing

The specialised profile language $\mathcal{L}_p$, which extends the foundational KDPD language with arithmetic and relational operators and restricts wffs to DNF (as detailed in Section 4.1.2), is central to expressing conditions for profile revision.

Users provide wffs as JavaScript strings. The construct `e1[i]` directly implements the theoretical label L_{i+1} . Constants, arithmetic operators (addition (+), subtraction (-), multiplication (*), division (/), and modulo (%)), relational operators (equality (==), inequalities (<, >, <=, and >=)), and logical connectives (logical AND (&&), logical OR (||), and negation (!)) map to their $\mathcal{L}_p$ counterparts. Consequently, the following represent the same logical expression:

```
"e1[0] < 3 && (e1[1] === e1[0] + 4)"
```

$$L_1 < 3 \wedge (L_2 = L_1 + 4)$$

The `parseCondition` function, assisted by the Acorn³ library, transforms the wff strings into a custom AST mirroring the structures of $\mathcal{L}_p$. A comprehensive elaboration on this topic is provided in Section 4.2.2.

4.2.1.3 Manhattan Distance Metric

The Manhattan distance, $d_M(p, q) = \sum_{i=1}^n |p_i - q_i|$ (Definition 10), serves as the primary metric for assessing profile dissimilarity. Its computation is performed incrementally throughout the search process and is utilised to facilitate search space pruning. In particular, for Model 1, the system enforces the total strict order $\prec_r$ (Definition 12) by evaluating component distances in accordance with specified label priorities.

4.2.2 Abstract Syntax Tree Engine

The ASTe is pivotal for interpreting the wffs provided by the user. Its architecture manages the lifecycle of a wff, transforming it from a raw string into a validated, evaluable, and analysable structure. The responsibilities of the ASTe are as follows:

³<https://github.com/acornjs/acorn>

Parsing and Structural Transformation of the wff: The ASTe initially receives the wff as an input string and utilises the Acorn parser to analyse its grammatical structure, generating a preliminary AST. Subsequently, this general tree is converted into a custom AST specifically tailored to represent and manipulate profile conditions, with node types optimised for the logic of profile element access and operations.

Comprehensive Syntactic and Semantic Validation: Once converted into the custom AST format of the system, the wff undergoes a rigorous validation process. Initially, syntactic validation verifies the correct use of operators, the validity of function calls, and adherence to the language of profile conditions. Following this, semantic validation ensures the wff is meaningful and correct within the context of the system, verifying that profile element references correspond to valid positions and confirming that all operations are supported by the logic of the system.

Identification of Constrained Profile Attributes: During analysis, the ASTe identifies the unique profile attributes referenced or constrained by the formula. This information is shared with other system modules, such as the DMS, allowing processes like domain narrowing to concentrate resources upon the relevant attributes.

Generation of an Executable Predicate: Following parsing and validation, the ASTe augments the structured wff with an executable JavaScript function. This function encapsulates the logic of the wff and acts as a predicate. It can be invoked with a profile to determine whether that profile satisfies the conditions of the wff, returning a Boolean `true` or `false`. The TFM utilises this predicate to evaluate candidate profiles.

The ASTe thus ensures that wffs are robustly processed, validated, and rendered in a form suitable for both direct evaluation and for guiding the domain narrowing process. In-depth details of the ASTe implementation and processes are provided in Appendix 6.2.2.

4.2.3 Domain Management System

The DMS plays a crucial role in enhancing the efficiency of the tuple identification process by effectively reducing the search space. Although algorithm optimisation was not the primary priority, efforts were made to achieve marginal improvements. The DMS aims to: first, demonstrate the significant influence of domain restriction on the search space, enabling more thorough tests and diverse scenarios; and second, highlight the future significance of these enhancements. Its

implementation focuses on specifying and fine-tuning the allowed value ranges for each profile element as specified by the wff. For unconstrained elements, domains are restricted to a single value corresponding to the original profile. The responsibilities of the DMS are as follows:

Representation and Management of Permissible Attribute Values: The DMS is responsible for defining and maintaining the set of permissible values that each individual attribute or element within a profile can assume. To achieve this, an internal structure is employed that is capable of representing these allowed values not merely as simple continuous numerical ranges (for instance, "any integer from 1 to 100") but also as collections of specific, discrete numerical values, or a combination of both formats. This representational structure is dynamic; for example, if two continuous ranges overlap or are adjacent, they are automatically consolidated into a single, more comprehensive range. Similarly, if a specific discrete value subsequently falls within a continuous range due to the addition or expansion of that range, it is seamlessly absorbed, ensuring an efficient and non-redundant representation of possibilities. The system also provides foundational operations, such as verifying whether a particular value is contained within the current domain of an attribute, determining if a domain has become empty, and calculating the intersection between the domains of two attributes when required by a constraint.

Initialisation and Iterative Refinement of Value Ranges: The process of determining valid profiles commences with the DMS establishing an initial, broad set of possibilities for all profile attributes pertinent to the given condition or wff. Each relevant attribute is initially presumed to be capable of assuming any value within a globally defined minimum and maximum limit (e.g. if profile attributes represent scores from 0 to 1000, this entire span constitutes the initial domain for each). Following this initialisation, the DMS embarks on an iterative refinement cycle. This cycle repeatedly processes the constraints imposed by the wff, progressively narrowing the domains of the involved attributes until a stable state is achieved, wherein no further reductions are possible, or until a contradiction is detected, manifesting as an empty domain for an attribute.

Constraint-Driven Deduction of Tighter Value Bounds: The core reasoning capability of the DMS resides in its ability to deduce tighter bounds for attribute values by systematically analysing each constraint within the wff. For any given constraint (e.g. "attribute *A* must be greater than attribute *B*", or "attribute *C* must be equal to the sum of attribute *A* and the constant 10"), the DMS first evaluates the current range of possible values for any expressions involved (such as

" $A + 10$ "), based upon the known, currently established domains of their constituent attributes. It then applies the logic of the specific relational operator in the constraint (e.g. "greater than", "equals to") to infer new, more restrictive limits for the expressions and, by extension, for the attributes themselves.

For instance, if it is determined that the expression " $A + 10$ " must equate to attribute C , and attribute C is currently known to lie within the range $[20, 30]$, the DMS can deduce that attribute A must consequently lie within the range $[10, 20]$. This newly inferred, tighter domain for attribute A is then recorded. As the domains of attributes are often interdependent through multiple constraints, a reduction in the permissible range for one attribute can precipitate further deductions and reductions for others. This constraint propagation continues iteratively across all components of the wff.

Enhancement of Profile Search Efficiency: The primary purpose of this meticulous domain-narrowing process is the significant enhancement of the efficiency of the overall system. By proactively and systematically eliminating values that cannot logically form part of any profile satisfying the entirety of the specified conditions, the DMS drastically reduces the magnitude of the search space that the TFM must subsequently explore. This substantial pruning of possibilities allows the TFM to identify valid profiles much more rapidly, which is particularly crucial when dealing with complex, multi-faceted conditions or profiles possessing a large number of attributes with extensive initial value ranges.

The details of the implementation are provided in Appendix 6.2.3. Awareness of the limitations of the current version exists, and these are discussed in further detail in Section 4.4.3. Despite these limitations, the existing domain-narrowing mechanism provides a substantial and crucial reduction in search complexity.

4.2.4 Tuple Finder Module

The TFM is entrusted with the intricate and computationally central task of identifying profile tuples that not only conform to the logical constraints articulated in a given wff, but also exhibit optimal proximity—quantified by Manhattan distance—to a designated target profile. The TFM orchestrates a multistage search process, the efficiency of which is critically dependent on the preparatory analyses conducted by the ASTe and the domain-reduction capabilities furnished by the DMS. **Algorithm 1** provides an overview of this process.

Algorithm 1 End-to-end workflow for revising or updating one or more initial profiles under a wff. The procedure handles trivial or already satisfied conditions, determines the target profile as either the initial profile or the element-wise average of a set of initial profiles, decomposes a disjunctive wff into conjunctive clauses, parses each clause, identifies its dependent profile attributes, narrows their admissible domains, and searches for satisfying profiles at minimum Manhattan distance from the target. The resulting revised or updated profile set is returned; when no satisfying solution is found, the original profile or profiles are preserved.

```

if wff is trivial then return startingProfile(s)
if startingProfile(s) satisfies wff then return startingProfile(s)
if multiple startingProfiles then
    targetProfile  $\leftarrow$  average of startingProfiles
Split wff into conjunctions
for all conjunction do
    Parse conjunction into ast
    Gather relevant profile elements
    Create and Narrow domains for those elements
    Search closest profiles to targetProfile
if no solution then return startingProfile(s)
return revisedProfile(s)

```

The operational tenets of the TFM encompass the following:

Initialisation, Pre-emptive Checks, and wff Decomposition: The operational sequence of the TFM commences with the handling of preliminary conditions and input normalisation. This includes the expedient processing of trivial wffs, such as empty strings or constant **true/false** values. For substantive wffs, a "quick check" is performed, whereby the wff is dynamically compiled to evaluate whether the provided initial profile(s) already satisfy the stipulated condition; an affirmative outcome allows for an immediate return, obviating the need for a more extensive search. Should the input comprise a set of profiles, an element-wise average profile may be computed to serve as a singular reference point for subsequent distance calculations, particularly for Model 3 (see Section 4.3.3.1). As mandated by the structural constraints of the language for reasons of computational tractability (see Definition 9), a crucial preparatory step involves the decomposition of wffs containing disjunctions (logical **OR** operators) into their constituent conjunctive clauses. Each such clause is thereafter subjected to independent processing by the core search algorithm; the globally optimal solutions—those achieving the overall minimum distance across all processed clauses—are ultimately retained.

Orchestration of the ASTe and DMS Services: For each distinct conjunctive component derived from the wff, the TFM assumes an orchestrating role, directing the ASTe to undertake a sequence of essential preparatory tasks. These include the parsing of the wff clause string into a

custom AST; the validation of this AST against the defined profile arity and the lexicon of supported operations and syntactic structures of the system; the extraction of a unique set of tuple indices (dependencies) referenced within the AST, thereby identifying which profile elements are constrained; and, finally, the generation of a dynamically executable predicate function from the AST. This predicate function serves as an efficient mechanism for verifying whether a candidate tuple satisfies the logical conditions of the wff clause. **Algorithm 2** illustrates this interaction. Subsequently, the TFM collaborates with the DMS to optimise the ensuing search process strategically. Initially, the TFM instructs the DMS to construct an initial domain map; thereafter, the TFM invokes the domain-narrowing functionality of the DMS.

Algorithm 2 Preparation of a single conjunctive clause of the wff by the TFM, using the ASTe and the DMS. The clause is parsed and validated against the profile length, the indices of its dependent profile attributes are extracted, and an initial domain map is created and narrowed through constraint propagation. Attributes whose domains have been reduced to a single value are pre-assigned and incorporated into the initial Manhattan distance, while the remaining dependent indices are ordered and passed to the recursive search procedure. Invalid syntax, unsupported expressions, or an empty narrowed domain terminate the processing of the clause without producing a solution.

```

1: ast ← ASTe.parseCondition(wff)
2: if not ASTe.validateAST(ast, targetProf.length) then return                                ▷ No solution
3: deps ← ASTe.getAllIndexes(ast)
4: domainMap ← DMS.createInitialDomainMap(deps, min, max, len)
5: domainMap ← DMS.narrowDomainMap(ast, domainMap, min, max)
6: if any domain in domainMap is empty then return                                        ▷ No solution
7: assignment ← []; initialDistance ← 0; toSearch ← []
8: for all idx ∈ deps do                                                                ▷ Pre-assign single-value domains
9:   if domainMap.get(idx) is single value v then
10:     assignment[idx] ← v
11:     initialDistance ← initialDistance + |v - targetProf[idx]|
12:   else
13:     Add idx to toSearch (sorted)
14: evalFn ← ASTe.buildEvalFunctionFromAST(ast)
15: EXPLORE(...)

```

Execution of Distance-Guided Constrained Combinatorial Search: Equipped with the executable predicate from the ASTe and the narrowed attribute domains from the DMS, the TFM initiates its primary combinatorial search procedure. This search—implemented as a recursive, depth-first exploration—seeks to discover valid assignments of values to the dependent profile attributes, operating exclusively within their refined domains.

The exploration is guided by the objective of minimising the Manhattan distance to the target profile. Throughout this search, the cumulative Manhattan distance of any current partial assignment is tracked and compared against the best-known minimum distance found thus far; search branches that demonstrably cannot lead to a solution superior to the current optimum are pruned. Upon the construction of a complete assignment for all dependent attributes, its validity is confirmed by an invocation of the executable predicate. This recursive search process is illustrated in

Algorithm 3.

Algorithm 3 Recursive distance-guided DFS used by the TFM to assign values to the constrained profile attributes. At each recursion level, candidate values from the narrowed domain are examined in order of proximity to the corresponding value in the target profile. The cumulative Manhattan distance is updated after each assignment, and any branch whose distance already exceeds the best global distance is pruned. Once all dependent attributes have been assigned, the executable predicate is used to test the complete candidate; satisfying assignments either establish a new global minimum or are added to the set of unique solutions tied at the current minimum. The procedure then backtracks to explore the remaining admissible values.

```

1: procedure EXPLORE(depIdx, curDist, assignment, toSearch, domainMap, targetProf,
  evalFn)
2:   if depIdx=length of toSearch then                                ▷ Base case: full assignment for dependencies
3:     if evalFn(assignment,targetProf) then
4:       assignmentKey←JSON.stringify(assignment)
5:       if curDist<globalMinDistance then
6:         globalMinDistance←curDist
7:         globalAssignments←[copy(assignment)]
8:         globalSeenSolutions←assignmentKey
9:       else if curDist=globalMinDistance AND assignmentKey ∉ globalSeenSolutions
10:      then
11:        Add copy(assignment) to globalAssignments;
12:        Add assignmentKey to globalSeenSolutions
13:      return
14:      idxToAssign←toSearch[depIdx]
15:      originalValueAtIndex←targetProf[idxToAssign]
16:      currentDomain←domainMap.get(idxToAssign)
17:      for all val in currentDomain.intervals and currentDomain.discreteValues (prioritising values closer to originalValueAtIndex) do
18:        newDist←curDist + |val - originalValueAtIndex|
19:        if newDist>globalMinDistance then continue                                ▷ Pruning
20:        assignment[idxToAssign]←val
21:        EXPLORE(depIdx+1,newDist,assignment,...)
22:      Delete assignment[idxToAssign]                                ▷ Backtrack

```

Consolidation of Results and Profile Reconstruction: Assignments that satisfy the wff and either establish a new global minimum distance or match an existing one are recorded, with mechanisms in place to ensure the uniqueness of stored solutions. Upon the completion of the search across all relevant conjunctive clauses of the wff, the TFM consolidates these findings. The

identified partial assignments, which specify values only for the dependent attributes, are then transformed into complete profile tuples.

This reconstruction process involves populating any non-dependent (unconstrained) indices with their original values from the target profile. Finally, the TFM returns an object encapsulating the overall minimum Manhattan distance achieved and an array containing the unique, complete profile tuples that correspond to this minimum distance. This output is subsequently passed to the PDM for any model-specific final processing.

Algorithm 4 illustrates the general flow of the TFM.

Algorithm 4 Complete control flow of the `findTuplesWithConstraints` operation implemented by the TFM. The procedure first handles trivial wffs and returns any initial profiles that already satisfy the supplied condition. It then determines the reference profile used for Manhattan-distance calculations—either the initial profile itself or the element-wise average of a set of initial profiles—and initialises the global minimum-distance and solution-tracking structures. Each disjunctive clause of the formula is processed independently by the conjunctive-clause search procedure, which updates the shared global optimum. Finally, the optimal partial assignments are reconstructed as complete profile tuples, and the procedure returns the minimum Manhattan distance together with every unique satisfying profile attaining that distance. When no satisfying assignment exists, the original profile or profiles are returned unchanged.

```

1: Handle trivial wff (empty, "true", "false"); if so, return  $P_{init}$  with distance 0.
2:  $checkFn \leftarrow ASTe.buildEvalFunctionFromAST(ASTe.parseCondition(wff))$ 
3: if  $P_{init}$  (or any  $P \in P_{init}$  if set) satisfies  $checkFn$  then
4:   return  $minDistance:0, tuplesWithMinDistance:satisfying P_i \in P_{init}$ 
5: if  $P_{init}$  is a set of profiles then
6:    $targetArray \leftarrow calculateTupleAverages(P_{init})$  ▷ Reference for distance
7: else
8:    $targetArray \leftarrow P_{init}$ 
9:  $globalMinDistance \leftarrow \infty; globalAssignments \leftarrow [];$ 
10:  $globalSeenSolutions \leftarrow Set();$ 
11:  $orParts \leftarrow split\ wff\ by\ "||"$ 
12: for all  $part \in orParts$  do
13:   Call FINDTUPLESFORCONJUNCTION( $part, targetArray, minVal, maxVal$ ) ▷ Updates globals
14: if  $globalMinDistance = \infty$  then
15:   return  $minDistance:0, tuplesWithMinDistance:P_{init}$  ▷ No solution found, return original
16:  $fullTuples \leftarrow reconstructFullTuple(globalAssignments, targetArray)$ 
17: return  $minDistance:globalMinDistance, tuplesWithMinDistance:fullTuples$ 

```

This architecture allows the TFM to find optimal profiles efficiently by initially reducing the complexity of the problem through AST analysis and domain narrowing, before performing a guided, pruned search. Detailed specifications of these implementations are provided in Appendix 6.2.4.

4.2.5 Profile Dynamics Module

The PDM functions as the primary high-level interface and central orchestrator for the diverse profile revision and update capabilities of the system. It provides a structured selection interface, exporting distinct functions—namely `model1`, `model2`, `model3`, and `model4`—each corresponding to one of the theoretical profile change models advanced by us [40]. The PDM is responsible for receiving the initial operational parameters, which include the initial profile(s), the wff stipulating the change criteria, global value range constraints, and any model-specific parameters such as priority conditions. A core function of the PDM is to prepare these inputs appropriately, managing distinctions such as whether a single profile or a set of profiles is being processed, before delegating the computationally intensive task of identifying candidate tuples that satisfy the wff and associated distance criteria to the TFM, as detailed in Section 4.2.4. Subsequent to the execution of the TFM, the PDM undertakes model-specific post-processing to produce the final output.

The operationalisation of Model 1 within the PDM addresses the revision of a single profile to yield a unique resultant profile. Upon receiving the set of candidate tuples from the TFM—all of which are minimally distant from the initial profile—the PDM ascertains whether a plurality of such tuples exists. Should this be the case, the PDM selects a single tuple based upon a `priorityCondition`. This process is explained in further detail in Section 4.3.1. This model is applicable, for instance, when revising a specific cognitive abilities profile for a patient based on new assessment data. In such scenarios, certain cognitive functions are deemed to have a higher priority for consideration in the revision, and a single definitive updated profile is required for subsequent therapeutic planning.

For Model 2, which facilitates the revision of a single profile into a set of profiles, the role of the PDM is more direct. After invoking the TFM with the initial profile and wff, the PDM receives the set of all tuples that satisfy the wff and are minimally distant from the original. Without applying any further selection or tie-breaking mechanisms, the PDM returns this entire set of tuples. Such an approach is suitable for scenarios where exploring all equally plausible alternative profiles is desired; for example, when a user expresses a new, potentially ambiguous interest and the system aims to present all minimally divergent profile interpretations that accommodate this new interest. Section 4.3.2 provides a more detailed explanation of this process.

Model 3 is designed for the revision of an initial set of profiles into a new set of profiles, conceptualising the change as affecting the set as a whole. The PDM supplies the initial set of profiles directly to the TFM. As explained in Section 4.2.4, the internal logic of the TFM for this scenario involves generating an average profile from the initial set; this average serves as the reference point for the TFM’s distance-minimisation search for profiles satisfying the wff. The PDM then returns the set of profiles produced by the TFM. A pertinent application of Model 3 is the updating of a profile representation for a specific demographic segment in response to newly observed general trends or characteristics relevant to that entire segment, resulting in a new set of profiles that collectively represent the updated demographic. This topic is explored in greater detail in Section 4.3.3.1.

Finally, Model 4 also addresses the transformation of an initial set of profiles into a resultant set of profiles, but it conceptualises this as an update process that preserves the individuality of each profile within the set. While Model 3 is effective when each profile in the set represents a group collectively, Model 4 is designed to maintain the uniqueness of individual profiles. The PDM processes each profile from the initial set independently and updates it based upon the wff. This yields a set of closest satisfying profiles for each original profile. The PDM then aggregates these individual result sets, ensuring uniqueness. For further information, please refer to Section 4.3.3.2.

In this manner, the PDM functions as a façade, providing a clear interface to the KDPD models whilst abstracting the complexities of the underlying tuple identification, constraint satisfaction, and domain management processes, and tailoring the final output in accordance with the specific semantics of the chosen KDPD model.

4.3 Operationalisation of Profile Dynamics Models

The preceding sections have delineated the comprehensive system architecture, the functional roles and interdependencies of its principal modules—namely the PDM, TFM, ASTe, and DMS—and the generalised algorithmic pipeline that underpins the processing of wffs through to the identification of candidate profiles. Furthermore, the systematic mapping of the foundational theoretical constructs, as articulated in Section 4.1, to their concrete manifestations within the implemented software artefacts has been established.

Building upon this established architectural and implementational groundwork, the present section provides a detailed exposition of the specific operationalisation of each of the four distinct

profile dynamics models proposed by [40]. These models provide distinct approaches to profile modification, catering for diverse scenarios of profile revision and update:

- **Model 1:** Revises a singular input profile to produce a unique resultant profile, employing a user-defined priority condition to resolve any ambiguities among equally optimal candidates.
- **Model 2:** Transforms a singular input profile into a set of resultant profiles, encompassing all candidates that satisfy the revision criteria at a minimum distance.
- **Model 3:** Revises an initial set of profiles into a new set of profiles, considering satisfaction within the initial set or referencing an average of the initial profiles for distance computations.
- **Model 4:** Updates an initial set of profiles to produce a new set, independently processing each initial profile against the stipulated condition and subsequently aggregating the unique, optimal results.

The ensuing subsections examine each model in turn, detailing its unique control flow as orchestrated by the PDM, its specific utilisation of the core system services (TFM, ASTe, DMS), and an analysis of its implemented conformity to the respective theoretical postulates defined within the KDPD framework.

4.3.1 Model 1: From One Profile to One Profile

Model 1 refers to the first profile revision scenario (see Section 4.2.5), in which a single initial profile is revised by a wff. This revision yields a singular, uniquely determined output profile. Model 1 is particularly suitable when a definitive outcome is necessitated, even if multiple profiles could potentially satisfy the input wff at a minimal distance from the original profile.

Within the implementation, Model 1 is realised by the `model1` function in the PDM, which orchestrates the revision process through a common core processing pipeline. This function accepts four parameters: (i) an *initial profile*, (ii) the *wff string*, (iii) the *global minimum and maximum value constraints* for the tuple elements, and (iv) a *priority condition string*. These parameters guide the following revision steps:

Core Tuple Finding: Initially, `model1` delegates candidate identification to the TFM by invoking the function `findTuplesWithConstraints` with the initial profile, the wff string, and the global constraints. This function processes the wff—utilising the ASTe and DMS—and returns all tuples that satisfy the wff at the minimum Manhattan distance from the initial profile.

Uniqueness and Tie-Breaking: If `findTuplesWithConstraints` returns exactly one tuple, that tuple is selected as the result. Otherwise, the *priority condition* parameter is utilised to select among the tied candidates.

Priority-Based Selection: The priority condition string (e.g. "`e1[2] === e1[1] > e1[0]`") is parsed by the auxiliary function `parsePriorityCondition` into ordered priority groups, where `>` denotes higher priority and `===` indicates equal priority within a group. For instance, the parsed groups `[[2, 1], [0]]` indicate that indices 2 and 1 share the highest priority, followed by index 0. Within the `findBestTuple` function, candidate tuples are compared with the initial profile group by group: for each priority group, the sum of absolute differences over the indices in that group is computed. The tuple with the smallest distance in the first group where differences occur is selected. Should ties remain, the comparison proceeds to the next group; if no distinction arises after all groups (at which point any remaining indices are treated with lower individual priority), the first such tuple encountered is selected to ensure deterministic behaviour.

Result: Finally, `model1` returns an object that encapsulates the minimum distance identified, along with an array containing the selected optimal tuple.

This implementation ensures that Model 1 consistently generates precisely one revised profile by employing a deterministic tie-breaking mechanism, which is contingent upon user-specified attribute priorities.

This section demonstrates how the implementation of `model1`—specifically through the functions `findTuplesWithConstraints` and `findBestTuple`—satisfies postulates **P1** through **P6** as defined in [40]. Detailed formal proofs for these claims are provided in Appendix 6.1.1.

(P1) If $\|\alpha\| \neq \emptyset$, then $P \odot \alpha \in \|\alpha\|$.

Whenever $\|\alpha\| \neq \emptyset$, the call to `findTuplesWithConstraints` first gathers $T = \{\tau \in \|\alpha\| \mid \min(d_M(P, \tau))\}$ and then `findBestTuple` picks some $\tau^* \in T$. Thus, $P \odot \alpha = \tau^* \in T \subseteq \|\alpha\|$.

(P2) If $\|\alpha\| = \emptyset$, then $P \odot \alpha = P$.

If $\|\alpha\| = \emptyset$, the function either returns P immediately (for trivial or ill-formed conditions) or, after domain narrowing detects a contradiction, executes a fallback that returns P at distance 0. In both cases $P \odot \alpha = P$.

(P3) If $P \models \alpha$, then $P \odot \alpha = P$.

If $P \models \alpha$, the dynamic predicate `checkFn(P)` is evaluated as true and the procedure immediately returns P with zero distance, thus $P \odot \alpha = P$.

(P4) If $\|\alpha\| = \|\beta\|$, then $P \odot \alpha = P \odot \beta$.

The underlying procedure of `findTuplesWithConstraints` is driven by the semantic content of the input formula. If α and β possess identical model sets, all semantically dependent steps within the code, including AST interpretation, domain narrowing, and distance-guided search, produce identical results for both, ensuring $P \odot \alpha = P \odot \beta$.

(P5) $P \odot (\alpha \vee \beta) = P \odot \alpha$ or $P \odot (\alpha \vee \beta) = P \odot \beta$.

The disjunction $\alpha \vee \beta$ is split at “|” into (α, β) ; the revisions $P \odot \alpha$ and $P \odot \beta$ are computed separately, and the overall minimum $\Delta = \min(\Delta_\alpha, \Delta_\beta)$ is used to select a profile from $T_\alpha \cup T_\beta$. That profile is in one of the two sets, so $P \odot (\alpha \vee \beta) = P \odot \alpha$ or $P \odot \beta$.

(P6) If $P \odot \alpha \models \beta$ and $P \odot \beta \models \alpha$, then $P \odot \alpha = P \odot \beta$.

If $P \odot \alpha \models \beta$ and $P \odot \beta \models \alpha$, then both $P_\alpha = P \odot \alpha$ and $P_\beta = P \odot \beta$ lie in $\|\alpha \wedge \beta\|$. The calculations of minimal distance enforce $\delta_\alpha = \delta_\beta$, and the determinism of `findBestTuple` guarantees $P_\alpha = P_\beta$.

4.3.2 Model 2: From One Profile to a Set of Profiles

Model 2, in accordance with the theoretical framework detailed in Section 4.2.5, addresses the revision of an initial profile when subjected to the constraints imposed by a wff. Unlike Model 1, the operational outcome of Model 2 is not a unique profile but rather the set of profiles that are equidistant from the initial profile. This approach is particularly pertinent in contexts where the revision process identifies multiple distinct profiles that are equally optimal with respect to the defined distance metric and the satisfaction of the wff. In such cases, the system design does not necessitate, nor provide for, a subsequent tie-breaking mechanism to elect a single profile from the resulting set.

The implementation of Model 2 is carried out within the PDM through the exported function designated `model2`. This function receives as input parameters: (i) an *initial profile*, represented as an integer tuple; (ii) the *wff string* that articulates the revision criteria; and (iii) the *global minimum and maximum value constraints* applicable to the elements of the profile tuples.

Upon invocation, the `model2` function passes its input parameters to the `findTuplesWithConstraints` function within the TFM. As detailed in Section 4.2.4, this procedure undertakes the comprehensive process of wff parsing and validation (via the ASTe), domain initialisation and narrowing (via the DMS), and the subsequent constrained search for profiles.

This search identifies all profiles that both satisfy the given wff and are minimally distant, in terms of Manhattan distance, from the initial profile. The `model2` function then directly returns the entire collection of such profiles as provided by the TFM. Consequently, if the revision process yields a plurality of profiles that are equally "best" according to these criteria, all are included in the resultant set, without recourse to any priority-based selection.

Presented here is a succinct articulation of the manner in which the implementation delineated in `model2`—facilitated by `findTuplesWithConstraints`—upholds the postulates **S1** to **S10** of [40]. The detailed formal proofs substantiating these assertions are elaborated in Appendix 6.1.2.

(S1) If $\|\alpha\| \neq \emptyset$, then $P \odot \alpha \subseteq \|\alpha\|$.

Adapted version of **P1**.

(S2) If $\|\alpha\| = \emptyset$, then $P \odot \alpha = \{P\}$.

Adapted version of **P2**.

(S3) $P \odot \alpha \neq \emptyset$.

In the implementation, it follows directly from **S1** and **S2**.

(S4) If $\|\alpha\| = \|\beta\|$, then $P \odot \alpha = P \odot \beta$.

Adapted version of **P4**.

(S5) If $P \models \alpha$, then $P \odot \alpha = \{P\}$.

Adapted version of **P3**.

(S6) $P \odot (\alpha \vee \beta) = P \odot \alpha$ or $P \odot (\alpha \vee \beta) = P \odot \beta$ or $P \odot (\alpha \vee \beta) = P \odot \alpha \cup P \odot \beta$.

The code treats $\alpha \vee \beta$ as a DNF, thus it computes $P \odot \alpha$ ($T_{P,\alpha}$ at distance $\delta_{P,\alpha}$) and $P \odot \beta$ ($T_{P,\beta}$ at distance $\delta_{P,\beta}$). The final result for $P \odot (\alpha \vee \beta)$ is then $T_{P,\alpha}$ if $\delta_{P,\alpha} < \delta_{P,\beta}$, or $T_{P,\beta}$ if $\delta_{P,\beta} < \delta_{P,\alpha}$, or $T_{P,\alpha} \cup T_{P,\beta}$ if $\delta_{P,\alpha} = \delta_{P,\beta}$.

(S7) $P \odot (\alpha \vee \beta) \subseteq P \odot \alpha \cup P \odot \beta$.

Weaker version of **S6**.

(S8) If $P \odot \alpha \models \beta$ and $P \odot \beta \models \alpha$, then $P \odot \alpha = P \odot \beta$.

Adapted version of **P6**.

(S9) $P \odot \alpha \cap P \odot \beta \subseteq P \odot (\alpha \vee \beta)$.

By **S6**, $P \odot (\alpha \vee \beta)$ is either $P \odot \alpha$, $P \odot \beta$, or $P \odot \alpha \cup P \odot \beta$. In all three cases, the set $P \odot \alpha \cap P \odot \beta$ is a subset of the result for $P \odot (\alpha \vee \beta)$ due to basic set theory.

(S10) $P \odot \alpha \cap \|\beta\| \subseteq P \odot (\alpha \wedge \beta)$.

If profile X is in $P \odot \alpha$ and also satisfies β , then X is a model of $\alpha \wedge \beta$ at distance $\delta_{P,\alpha}$. The code establishes that this distance $\delta_{P,\alpha}$ is also the minimum distance for $\alpha \wedge \beta$. Thus, X is included in $P \odot (\alpha \wedge \beta)$.

4.3.3 From a Set of Profiles to a Set of Profiles

The KDPD framework [40] further extends its formalisms to address scenarios in which dynamic operations are applied to an initial set of profiles, denoted Γ , to produce a resultant set of profiles. Within this category, two distinct models—Model 3 and Model 4—are proposed. These are primarily differentiated by their conceptual foundations (revision versus update) and, consequently, by the manner in which the initial set Γ informs the selection of the resultant profiles.

4.3.3.1 Model 3: Revising Sets of Profiles

Model 3, as conceptualised within the KDPD framework (see Section 4.2.5), addresses the scenario of revising an initial set of profiles, denoted Γ , upon the introduction of new information encapsulated by a wff α . The outcome of this revision operation is also a set of profiles.

The implementation of Model 3 is realised through the `model3` function. This function accepts as input parameters: (i) an *initial set of profiles* (`initialSetProfiles`), (ii) the *wff string* (α), and (iii) the *global minimum and maximum value constraints* that define the permissible range for tuple elements. The `model3` function orchestrates the revision process by directly invoking the `findTuplesWithConstraints` function of the TFM, which incorporates specialised logic for handling an input set of profiles.

The operational sequence initiated by `model3` via the TFM involves, firstly, an examination of the initial set of profiles (`initialSetProfiles`) to ascertain whether any constituent profile $P_i \in \Gamma$ already satisfies the wff α . Should such profiles exist, the set comprising these directly satisfying profiles (i.e. $\Gamma \cap \|\alpha\|$) is returned as the result, with an associated Manhattan distance of zero.

If, however, no profile within the initial set Γ satisfies α , the `findTuplesWithConstraints` function proceeds to compute an element-wise average profile, $P_{avg}(\Gamma)$, derived from all profiles in the `initialSetProfiles`. This average profile, $P_{avg}(\Gamma)$, subsequently serves as the reference target for the TFM’s distance-minimisation search. The TFM then employs its standard mechanisms—AST-driven wff processing, domain initialisation and narrowing via the DMS, and constrained combinatorial search—to identify all tuples that satisfy α and are minimally distant from this computed average profile $P_{avg}(\Gamma)$. The `model3` function then returns the set of profiles thus identified by the TFM.

It is crucial to clarify that this average profile, $P_{avg}(\Gamma)$, serves exclusively as a computational proxy; a reference point for the Manhattan distance metric. It provides a single concrete target against which the distance of candidate profiles can be measured, thus instantiating the abstract notion of proximity to the set Γ . The final output set returned by the TFM, and thus by Model 3, consists of profiles from the actual search space that are minimally distant from this average. The average profile $P_{avg}(\Gamma)$ itself is not included in the output, unless it coincidentally satisfies the wff and is found to be minimally distant during the search.

Presented herein is a succinct articulation of the manner in which the implementation delineated in `model3`—facilitated through `findTuplesWithConstraints`—upholds the postulates **SP1** to **SP10** of [40]. The detailed formal proofs substantiating these assertions are elaborated in Appendix 6.1.3.

(SP1) If $\|\alpha\| \neq \emptyset$, then $\Gamma \odot \alpha \subseteq \|\alpha\|$.

Adapted version of **S1**.

(SP2) If $\|\alpha\| = \emptyset$, then $\Gamma \odot \alpha = \Gamma$.

Adapted version of **S2**.

(SP3) $\Gamma \odot \alpha \neq \emptyset$.

In the implementation, it follows directly from **SP1** and **SP2**.

(SP4) If $\|\alpha\| = \|\beta\|$, then $\Gamma \odot \alpha = \Gamma \odot \beta$.

Adapted version of **S4**.

(SP5) If $\Gamma \cap \|\alpha\| \neq \emptyset$, then $\Gamma \odot \alpha = \Gamma \cap \|\alpha\|$.

The `model3` function directly implements this by first checking if the initial set Γ contains any profiles satisfying α . If it does, as per the postulate's premise, the function immediately returns exactly this subset ($\Gamma \cap \|\alpha\|$) with zero distance, bypassing any further search.

(SP6) $\Gamma \odot (\alpha \vee \beta) = \Gamma \odot \alpha$ or $\Gamma \odot (\alpha \vee \beta) = \Gamma \odot \beta$ or $\Gamma \odot (\alpha \vee \beta) = \Gamma \odot \alpha \cup \Gamma \odot \beta$.

Adapted version of **S6**.

(SP7) $\Gamma \odot (\alpha \vee \beta) \subseteq \Gamma \odot \alpha \cup \Gamma \odot \beta$.

Adapted version of **S7**.

(SP8) If $\Gamma \odot \alpha \models \beta$ and $\Gamma \odot \beta \models \alpha$, then $\Gamma \odot \alpha = \Gamma \odot \beta$.

Adapted version of **S8**.

(SP9) $\Gamma \odot \alpha \cap \Gamma \odot \beta \subseteq \Gamma \odot (\alpha \vee \beta)$.

Adapted version of **S9**.

(SP10) $\Gamma \odot \alpha \cap \|\beta\| \subseteq \Gamma \odot (\alpha \wedge \beta)$.

Adapted version of **S10**.

4.3.3.2 Model 4: Updating Sets of Profiles

Model 4, as conceptualised within the KDPD framework and distinct from the revision-based approach of Model 3 (see Section 4.2.5), addresses the *update* of an initial set of profiles, denoted Γ , in response to new information encapsulated by a wff, α . The outcome of this update operation is also a set of profiles.

The implementation of Model 4 is realised through the `model4` function within the PDM. This function accepts as input parameters: (i) an *initial set of profiles* (`initialSetProfiles`), (ii) the *wff string* (α) representing the new information regarding the state of the world, and (iii) the *global minimum and maximum value constraints* that define the permissible range for tuple elements.

The operational logic of the `model14` function is characterised by an iterative application of the core tuple-finding mechanism to each profile within the initial set. For every profile $P_i \in \Gamma$, the PDM invokes the `findTuplesWithConstraints` function of the TFM. In each such invocation, P_i serves as the target profile against which the wff is processed to find the set of closest satisfying tuples, T_{P_i} .

After processing all profiles in the initial set Γ , the PDM aggregates the resultant sets, utilising the `containsTuple` helper function to ensure that the final collection comprises only unique profiles.

Presented herein is a succinct articulation of the manner in which the implementation delineated in `model14`—facilitated through `findTuplesWithConstraints` and the subsequent aggregation function—upholds the postulates **U0** to **U11** of [40]. The detailed formal proofs substantiating these assertions are elaborated in Appendix 6.1.4.

(U0) If $\|\alpha\| = \emptyset$, then $\Gamma \diamond \alpha = \Gamma$.

Adapted version of **SP2**.

(U1) If $\|\alpha\| \neq \emptyset$, then $\Gamma \diamond \alpha \subseteq \|\alpha\|$.

Adapted version of **SP1**.

(U2) If $\Gamma \subseteq \|\alpha\|$, then $\Gamma \diamond \alpha = \Gamma$.

Adapted version of **P3**.

(U3) $\Gamma \diamond \alpha \neq \emptyset$.

In the implementation, it follows directly from **U0** and **U1**.

(U4) If $\|\alpha\| = \|\beta\|$, then $\Gamma \diamond \alpha = \Gamma \diamond \beta$.

Adapted version of **SP4**.

(U5) $\Gamma \diamond \alpha \cap \|\beta\| \subseteq \Gamma \diamond (\alpha \wedge \beta)$.

If a profile X is in $\Gamma \diamond \alpha$ and also satisfies β , then X satisfies $\alpha \wedge \beta$. The code ensures that X 's distance from P_k ($\delta_{P_k, \alpha}$) is also the minimal distance for $\alpha \wedge \beta$ from P_k ($\delta_{P_k, \alpha \wedge \beta}$). Thus, X is included in $T_{P_k, \alpha \wedge \beta}$ and consequently in $\Gamma \diamond (\alpha \wedge \beta)$.

(U6) If $\Gamma \diamond \alpha \models \beta$ and $\Gamma \diamond \beta \models \alpha$, then $\Gamma \diamond \alpha = \Gamma \diamond \beta$.

Adapted version of **SP8**.

(U7) If P is a profile, then $\{P\} \diamond \alpha \cap \{P\} \diamond \beta \subseteq \{P\} \diamond (\alpha \vee \beta)$.

For a single profile P , $\{P\} \diamond \phi$ reduces to finding $T_{P,\phi}$. If $X \in T_{P,\alpha} \cap T_{P,\beta}$, then X is the model closest to P for both α and β , showing the same minimal distances. When processing $\alpha \vee \beta$, the procedure merges the results for α and β sharing this minimal distance, thus including X .

(U8) $(\Gamma_1 \cup \Gamma_2) \diamond \alpha = (\Gamma_1 \diamond \alpha) \cup (\Gamma_2 \diamond \alpha)$.

The `model4` function computes $\Gamma_S \diamond \alpha$ by forming a union of results $(T_{P,\alpha})$ obtained from applying the function `findTuplesWithConstraints` to each profile $P \in \Gamma_S$. The equality then follows directly from the distributive property of set union: $\bigcup_{P \in \Gamma_1 \cup \Gamma_2} T_{P,\alpha} = (\bigcup_{P \in \Gamma_1} T_{P,\alpha}) \cup (\bigcup_{P \in \Gamma_2} T_{P,\alpha})$.

(U9) If $\|\alpha \wedge \beta\| \neq \emptyset$, P is a profile and $(\{P\} \diamond \alpha) \cap \|\beta\| \neq \emptyset$, then $\{P\} \diamond (\alpha \wedge \beta) \subseteq (\{P\} \diamond \alpha) \cap \|\beta\|$.

For a single profile P , if its closest models for α ($T_{P,\alpha}$) include a profile X that also satisfies β , then X satisfies $\alpha \wedge \beta$ at the same minimal distance $\delta_{P,\alpha}$. The code establishes $\delta_{P,\alpha \wedge \beta} = \delta_{P,\alpha}$. Thus, $\{P\} \diamond (\alpha \wedge \beta)$ (which is $T_{P,\alpha \wedge \beta}$) becomes identical to $T_{P,\alpha} \cap \|\beta\|$, satisfying the subset relation.

(U10) If $\Gamma_1 \subseteq \Gamma_2$, then $\Gamma_1 \diamond \alpha \subseteq \Gamma_2 \diamond \alpha$.

The `model4` function computes $\Gamma_S \diamond \alpha$ as the union of results $(T_{P,\alpha})$ from each $P \in \Gamma_S$. If $\Gamma_1 \subseteq \Gamma_2$, every $P \in \Gamma_1$ is also in Γ_2 . Therefore, all $T_{P,\alpha}$ contributing to $\Gamma_1 \diamond \alpha$ also contribute to $\Gamma_2 \diamond \alpha$, making the former union a subset of the latter.

(U11) $\Gamma \cap \|\alpha\| \subseteq \Gamma \diamond \alpha$.

If a profile X is in the initial set Γ and also satisfies α , then when `model4` processes X , X itself will be identified as the closest model. Since `model4` unions all such results, X is included in $\Gamma \diamond \alpha$.

4.4 Performance Evaluation

The performance evaluation of the implementation aims to empirically characterise the computational behaviour of the principal update and revision operations, with particular emphasis on scalability and practical feasibility within real-world parameter regimes. In this section, the experimental methodology is described, highlighting the benchmarking strategy and the rationale underlying the choice of parameters. Subsequently, the analysis focuses on the observed runtime characteristics across systematically varied domain sizes and tuple lengths, elucidating the factors that most strongly influence algorithmic performance. Attention is directed towards how the structural properties of profile constraints—as opposed to merely the nominal dimensionality—govern

the tractability of the underlying search procedures. The results thereby provide quantitative support for the suitability of the approach within bounded domains typical of user modelling applications, and offer constructive insights for deployment and optimisation in applied contexts.

4.4.1 Benchmarks

Analysis of the system architecture reveals that the majority of the runtime is concentrated within the function `findTuplesWithConstraints`, which is utilised by all four revision models. For this reason, the empirical behaviour of this function is interpreted under two benchmark regimes—*unrestricted* and *restricted*—and the observed timings are related to the search dynamics of the algorithm. The evaluation setup (tuple length N , domain size D , and the condition set) is first recapitulated, followed by a quantification of how median runtime scales with D and an explanation of why it remains largely invariant with respect to N . It is demonstrated that performance is governed by the *structural arity* k of the predicate (the number of indices actually constrained), rather than the nominal tuple length; this is illustrated through per-condition decade ratios and worst-case examples. Finally, simple power-law models are fitted to the medians to extrapolate to larger domains, and practical guidance is distilled—covering variable ordering, stronger narrowing, and neighbourhood tuning—to reduce the effective search exponent in real-world deployments.

4.4.1.1 Benchmark Design and Code Paths

The function `findTuplesWithConstraints` was benchmarked using two distinct regimes:

- (i) **Unrestricted:** Starting tuples are generated completely at random, with values drawn from the full domain D .
- (ii) **Restricted:** Starting tuples are generated within a *close* margin of the solution, specifically at most 5% Manhattan distance. This approach is intended to simulate real-world scenarios more accurately, where revision models are necessitated upon the arrival of new information. Consequently, in most cases, the resulting variations are minimal.

For the unrestricted regime, tests were conducted with $N = \{10, 100\}$ and $D = \{10, 100\}$. For the restricted regime, the parameters were expanded to $N = \{10, 100, 1000, 2500, 5000, 10000\}$ and $D = \{10, 100, 1000\}$. All conditions utilised for these tests are provided in Appendix 6.3. Each condition was subjected to 500 trials. For each trial, a new starting tuple was generated and the execution of `findTuplesWithConstraints` was timed in isolation. Median values are analysed

to mitigate the impact of skew. Furthermore, certain conditions are examined independently to illustrate specific limitations of the current implementation.

4.4.1.2 Runtime Growth

Across both regimes:

- The runtime is dominated by D and the *structural arity* k of the predicate (the number of distinct indices referenced).
- At a fixed D and predicate arity/propagation, the *search* cost is governed by the number of dependent indices actually explored ($m \leq k$), rather than by the nominal length N . The DFS ranges only over the set `toSearch`, ordering values by proximity and pruning by the distance bound; indices never referenced by the predicate do not enlarge the search tree.
- *Implementation note:* After the search, the current code materialises each best-distance solution as a full N -tuple (reconstructing from sparse assignments), which is $O(N)$ per reported solution. Consequently, wall time may increase with N when there are numerous equal-cost solutions, even though the branching work remains unchanged.

4.4.2 Empirical scaling with domain size

The following figures quantify the manner in which median runtime scales with domain size D under each benchmark regime. Figure 5 presents the results for the unrestricted regime, wherein starting tuples are drawn uniformly at random from the full domain. The observed curves exhibit pronounced sensitivity to D : a tenfold increase in domain size yields approximately a fifteen- to sixteenfold increase in median runtime, consistent with the power-law behaviour described above. Notably, the curves for $N = 10$ and $N = 100$ are nearly indistinguishable, confirming that nominal tuple length exerts negligible influence when the search is governed by the constrained indices alone. Figure 6 illustrates the corresponding behaviour under the restricted regime, where starting tuples lie within 5% Manhattan distance of a satisfying solution. At small domain sizes ($D = 10$), median runtimes collapse to sub-millisecond values across all tested N , reflecting the efficiency of proximity-guided search when the initial tuple is already close to a witness. As D increases, the D -sensitivity reasserts itself, whilst the dependence on N remains modest—curves for N ranging from 10 to 10,000 remain tightly clustered, with growth in N contributing far less to runtime than growth in D . Together, these figures substantiate the claim that the tractability of the implementation is

dominated by domain size and structural arity, rather than by the nominal dimensionality of the profile space.

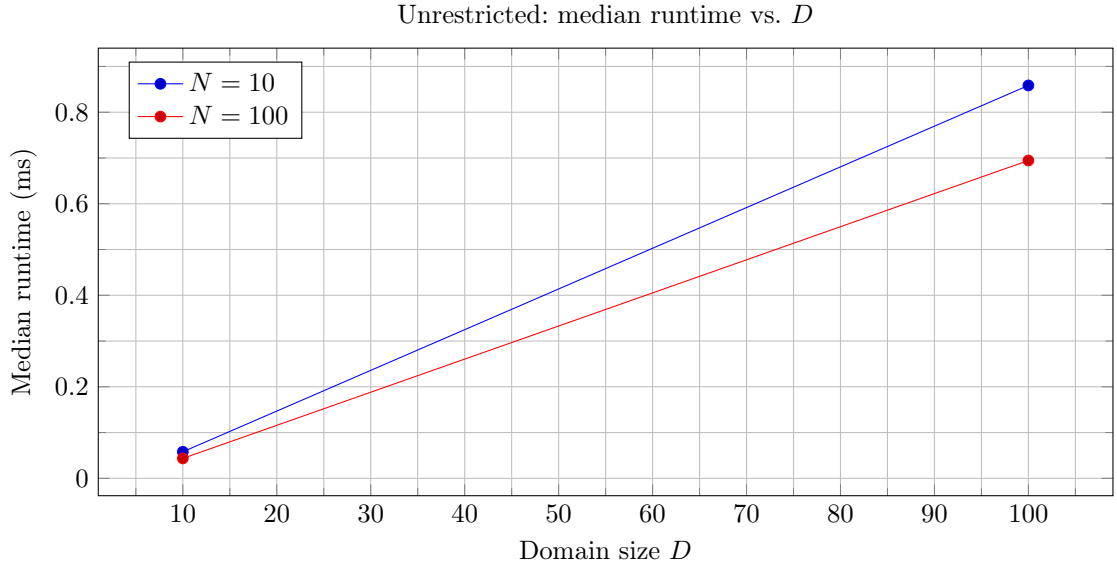


Figure 5: Median execution time, in milliseconds, of `findTuplesWithConstraints` as a function of domain size D in the unrestricted benchmark regime, in which each starting tuple is sampled from the full bounded domain. Separate curves represent tuple lengths $N = 10$ and $N = 100$, and each plotted median is based on 500 independently generated starting tuples for the corresponding benchmark condition. Runtime increases strongly with D : the fitted scaling implies an approximately 15–16-fold increase for each tenfold increase in domain size. The almost coincident curves show that nominal tuple length has little effect because the search branches primarily over indices referenced by the predicate.

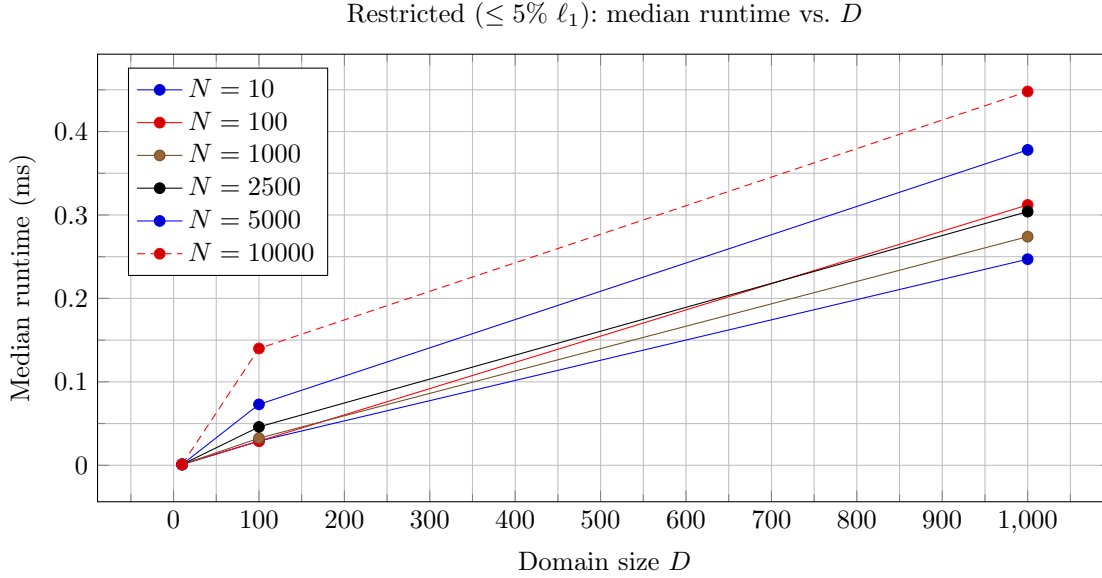


Figure 6: Median execution time, in milliseconds, of `findTuplesWithConstraints` as a function of domain size D in the restricted benchmark regime. Starting tuples are generated no more than 5% Manhattan distance from a satisfying tuple, modelling revisions in which the required change is local. Curves represent tuple lengths from $N = 10$ to $N = 10,000$, and each median is calculated from 500 trials for the corresponding condition. Near-witness initialisation produces sub-millisecond runtimes at small D , but runtime still increases substantially as the domain grows. The clustering of the curves demonstrates that D has a substantially stronger effect than nominal tuple length N .

4.4.2.1 Arity k Controls the Slope

Predicates are grouped by the number of distinct indices they reference (k), and the ratio $\text{median}(D_2)/\text{median}(D_1)$ per group is plotted (see Figure 7). The ratios climb steeply with k : domain growth modestly multiplies runtime when $k \leq 2$, but once $k \geq 3$, the coordination of several indices causes near-exponential inflation.

Minor non-monotonicities across adjacent k values reflect predicate semantics—such as narrowing strength or the presence of disjunctions—rather than a change in the overall trend.

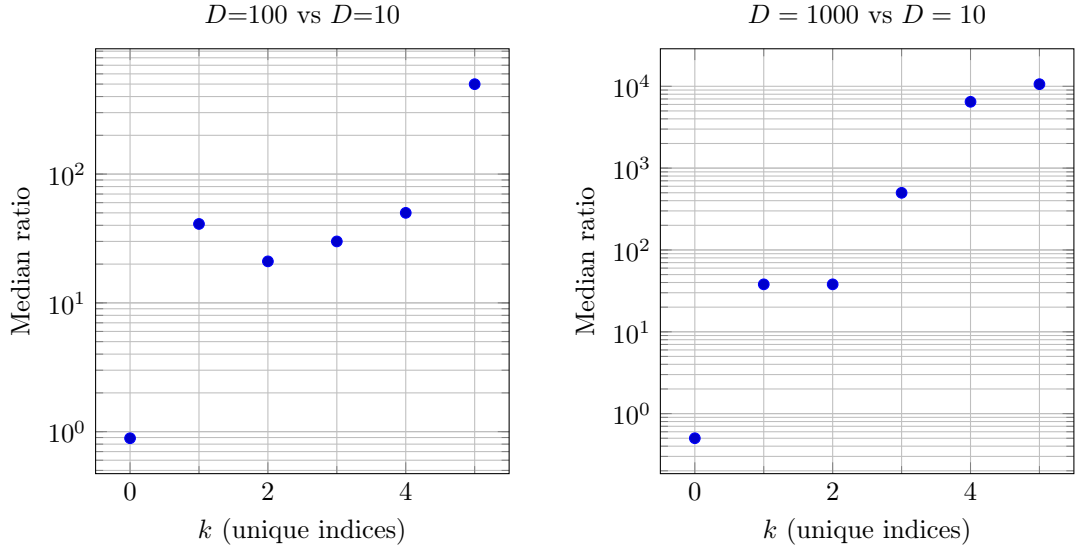


Figure. 7: Effect of predicate structural arity k , the number of distinct tuple indices referenced by a predicate, on runtime growth under increasing domain size. The left panel reports the ratio between median runtimes at $D = 100$ and $D = 10$; the right panel reports the corresponding ratio between $D = 1000$ and $D = 10$. Predicates are grouped by k , and the vertical logarithmic scale expresses the multiplicative increase in median runtime. Domain growth produces relatively modest multipliers for $k \leq 2$, whereas predicates involving $k \geq 3$ indices exhibit substantially stronger, super-linear runtime inflation. Small non-monotonic differences between adjacent values of k arise from predicate-specific semantics and differences in domain-narrowing effectiveness.

4.4.2.2 Impact of N in restricted mode

Median growth from $N = 10$ to $N = 10,000$ at fixed D : $1.00\times$ ($D = 10$), $4.83\times$ ($D = 100$), $1.81\times$ ($D = 1000$). This is minor compared to the $\sim 16\text{--}18\times$ per decade in D .

4.4.2.3 Search-Space Complexity and Empirical Runtime Extrapolation

Let k denote the number of profile indices referenced by the constraint. After domain narrowing, some of these indices may have singleton domains and can therefore be assigned without branching. Let $m \leq k$ denote the number of indices whose narrowed domains still contain more than one admissible value. For each such index i , let w_i denote the cardinality of its narrowed domain. If D is the maximum cardinality of an original, non-narrowed domain, then $w_i \leq D$.

The maximum number of complete assignments formed from the remaining branching variables is

$$C_{\max} = \prod_{i=1}^m w_i \leq D^m.$$

Therefore, when domain narrowing and distance-based pruning are ineffective, and when the cost of evaluating the predicate for one complete assignment is treated as bounded, the worst-case search cost is

$$T_{\max} \in O\left(\prod_{i=1}^m w_i\right) \subseteq O(D^m).$$

This expression describes an upper bound on the combinatorial search space. It does not directly predict the measured execution time, because the actual runtime also depends on the amount of domain reduction, the order in which values are explored, the effectiveness of distance-based pruning, the density and position of satisfying assignments, and the computational cost of the predicate.

To characterise the observed dependence of runtime on domain size, an empirical power-law model was fitted separately to each benchmark regime:

$$\hat{T}(D) = cD^p,$$

or, equivalently, on logarithmic axes,

$$\log \hat{T}(D) = \log c + p \log D.$$

Here, D is the domain size used in the benchmark, $\hat{T}(D)$ is the predicted median execution time of `findTuplesWithConstraints`, c is the fitted scaling coefficient, and p is the fitted scaling exponent. The exponent p corresponds to the slope of the regression line on log-log axes.

The fitted exponents were approximately

$$\text{Unrestricted: } p \approx 1.17\text{--}1.20, \quad \text{Restricted: } p \approx 1.20\text{--}1.25.$$

Under this empirical model, the predicted runtime ratio associated with a tenfold increase in domain size is

$$\frac{\hat{T}(10D)}{\hat{T}(D)} = \frac{c(10D)^p}{cD^p} = 10^p.$$

The observed values of p therefore correspond to an estimated runtime multiplication of approximately $10^p \approx 15\text{--}18$ for each tenfold increase in D .

For a target domain size D^* , the extrapolated median runtime can be calculated from a fitted reference point D_0 as

$$\hat{T}(D^*) = \hat{T}(D_0) \left(\frac{D^*}{D_0} \right)^p.$$

The estimates for $D^* \in \{10^4, 10^5\}$ were obtained by extending the fitted linear relationship on the log-log scale beyond the measured domain range. These values should therefore be interpreted as model-based extrapolations rather than observed runtimes or implementation-independent complexity bounds. Their validity depends on the assumption that the fitted exponent remains approximately constant outside the measured range. Changes in predicate structure, solution density, domain-narrowing effectiveness, pruning behaviour, hardware, or implementation strategy may substantially alter both c and p .

4.4.2.4 Worst cases by condition (restricted)

To illustrate the upper bound of the implementation's computational cost, three conditions with the most severe scaling behaviour under the restricted regime are examined individually.

Each of the following figures presents a dual-panel view:

- **Left panel:** Median runtime versus domain size D , with separate curves for each tuple length N .
- **Right panel:** Median runtime versus N , with separate curves for each D .

Figure 8 corresponds to a predicate imposing a chain of consecutive-value constraints:

```
e1[1]==e1[0]+1 && e1[2]==e1[1]+1 && e1[3]==e1[2]+1
```

This couples four indices and yields structural arity $k = 4$. The left panel reveals extreme D -sensitivity: median runtime grows from sub-millisecond values at $D = 10$ to over 1.1 seconds at $D = 1000$, an increase of approximately three orders of magnitude. The right panel confirms that runtime remains essentially flat across N — all curves for a given D cluster tightly, regardless of whether $N = 10$ or $N = 10,000$.

Figure 9 depicts a predicate with pairwise equalities and a disequality:

```
e1[0]===e1[1] && e1[2]===e1[3] && e1[0]!==e1[2]
```

Again, four indices are constrained. The scaling pattern is qualitatively similar: there is pronounced growth with D (~ 0.7 ms at $D = 100$ to ~ 880 ms at $D = 1000$), and near-invariance with respect to N .

Figure 10 illustrates a predicate combining mixed linear atoms:

```
e1[0]+e1[1]===5 && e1[2]>2 && e1[3]+e1[4]<15 && e1[4]>2
```

This constrains five indices. Here, the D -scaling is less severe than in the preceding cases— median runtime at $D = 1000$ reaches approximately 11 ms—yet the same qualitative pattern holds: D dominates, and N contributes negligibly.

Collectively, these worst-case conditions reinforce the conclusion that:

- Domain size (D) and structural arity govern tractability.
- Predicates coupling many indices through arithmetic or equality constraints pose the greatest challenge to the current implementation.

4.4.3 Limitations

The implemented domain-narrowing algorithm—whilst effective for a range of linear constraints—exhibits several fundamental limitations that curtail its universal applicability and efficiency. These limitations may be categorised into two principal groups: domains for which narrowing is entirely precluded and domains for which the narrowing process is substantively restricted, yielding suboptimal results.

A primary constraint of the algorithm is its inability to process expressions involving function calls, represented as `CallExpression` nodes in the AST. In such instances, the algorithm defaults to a conservative stance, assuming that the function may return any value within the global minimum and maximum bounds. Consequently, no domain reduction is achieved for variables involved in constraints that include function calls. This represents a significant restriction, as it precludes the analysis of any system where behaviour is encapsulated within functions.

Similarly, the algorithm fails to narrow domains in the context of division operations where the domain of the divisor includes zero. To avert the indeterminate outcome of division by zero, the

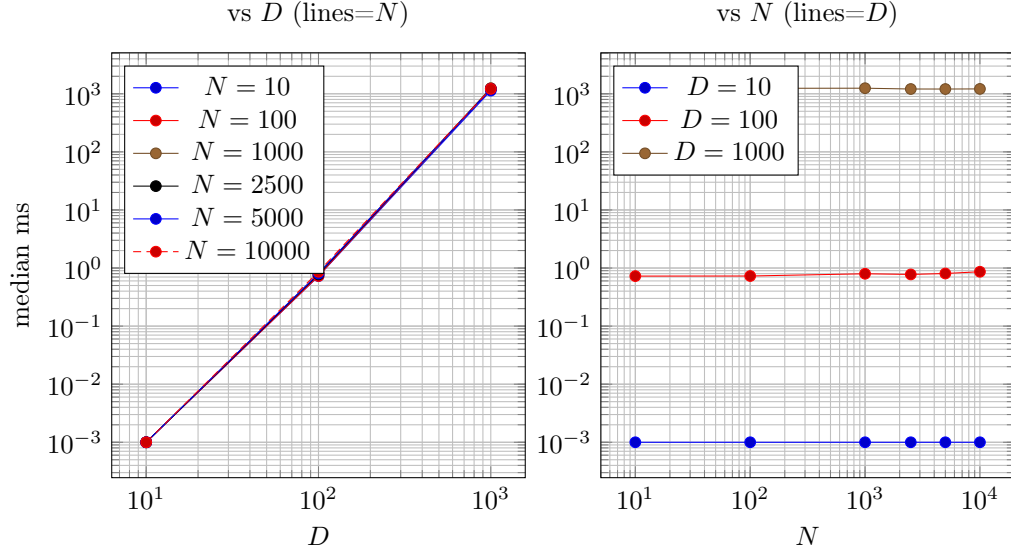


Figure 8: Restricted-regime runtime for the four-index predicate `e1[1]==e1[0]+1 && e1[2]==e1[1]+1 && e1[3]==e1[2]+1`, which requires four tuple elements to form a sequence of consecutive values and therefore has structural arity $k = 4$. The left panel plots median execution time against domain size D , with one curve for each tuple length N ; the right panel plots median execution time against N , with one curve for each D . Both axes are logarithmic, and medians are derived from 500 trials per condition. Runtime rises from sub-millisecond values at $D = 10$ to more than 1.1 seconds at $D = 1000$, while remaining approximately invariant across $N = 10$ to $N = 10,000$. This condition demonstrates that tightly coupled constraints and domain size, rather than total tuple length, dominate search cost.

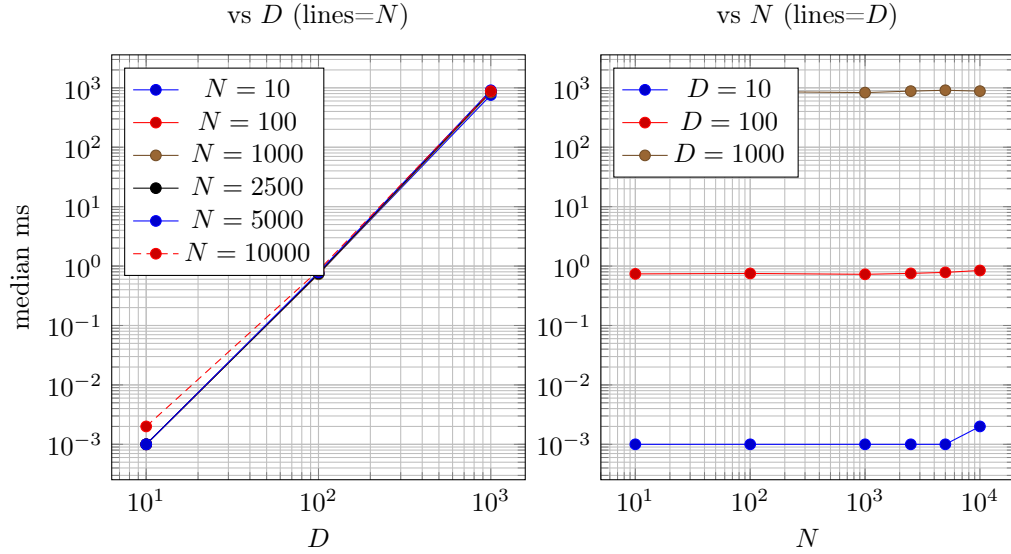


Figure 9: Restricted-regime runtime for the four-index predicate `e1[0]==e1[1] && e1[2]==e1[3] && e1[0]!=e1[2]`, which requires two internally equal pairs to take different values and has structural arity $k = 4$. The left panel plots median execution time against domain size D , with curves corresponding to different tuple lengths N ; the right panel plots median execution time against N , with curves corresponding to different values of D . Both axes are logarithmic, and each median is based on 500 trials per condition. Runtime increases from approximately 0.7 ms at $D = 100$ to approximately 880 ms at $D = 1000$, but changes little as N increases, showing that pairwise equality constraints scale primarily with domain size.

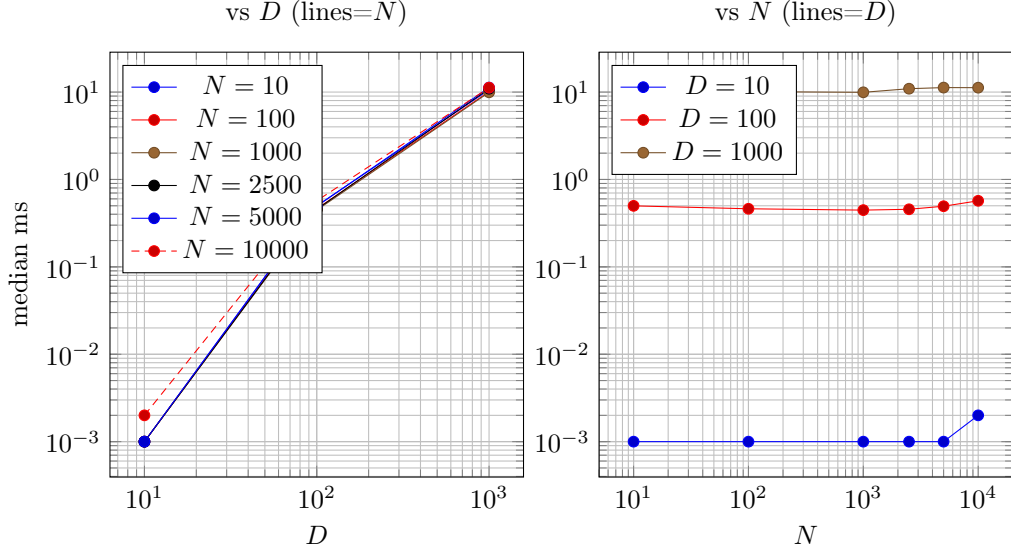


Figure 10: Restricted-regime runtime for the five-index predicate `e1[0]+e1[1]==5 && e1[2]>2 && e1[3]+e1[4]<15 && e1[4]>2`, which combines equality, inequality, and arithmetic constraints and has structural arity $k = 5$. The left panel plots median execution time against domain size D , with separate curves for tuple length N ; the right panel plots median execution time against N , with separate curves for D . Both axes are logarithmic, and medians are calculated from 500 trials per condition. Although its domain-size scaling is less severe than that of Figures 8 and 9, median runtime still reaches approximately 11 ms at $D = 1000$, while variation with nominal tuple length remains negligible.

algorithm returns the global value range, thereby foregoing any potential for domain restriction. This conservative approach, while ensuring logical consistency, is a notable source of imprecision in the constraint-propagation process. Furthermore, the algorithm does not handle inequality constraints (`!=`, `!==`) effectively when both operands are ranges, which limits its pruning capabilities in such scenarios.

In addition to these absolute limitations, the algorithm provides only partial or limited domain narrowing for certain arithmetic operations—most notably multiplication and division. The backward propagation mechanism, implemented in the `_refineArithExpressionRange` function, relies on interval arithmetic to estimate the domains of operands. It is a well-established principle that interval arithmetic for these operations can yield bounds that are not maximally constrained, a phenomenon known as the dependency problem [193].

The algorithm calculates the new interval by considering the endpoints of the operand intervals, which can result in an overestimation of the true domain. This effect is particularly pronounced when the operand domains are wide, leading to less effective pruning of the search space. Conse-

quently, while some narrowing occurs, it is not guaranteed to be the tightest possible, potentially leaving a larger-than-necessary search space for the tuple-finding phase of the algorithm.

4.5 Applying Knowledge-Driven Profile Dynamics to NeuroAIReh@b

This section elucidates how the implementation of the KDPD framework, as described above, can be integrated into the NeuroAIReh@b CR system [148]. This integration addresses the profile dynamics challenge identified in Section 3.4: the requirement for a principled, minimal-change mechanism to update patient cognitive profiles when novel information emerges from training performance or re-evaluation. The present exposition describes the proposed integration pathway, the advantages it affords, and the role of therapist-driven fine-tuning in resolving ambiguities during profile revision.

4.5.1 Overview and Advantages

The NeuroAIReh@b framework comprises four principal components—Patient Profiling, Training Selection, Training Session, and System Calibration—each of which presents specific challenges for profile representation and evolution, as detailed in Section 3.1. The KDPD implementation provides a formal, axiomatically characterised set of operators for profile revision and update, grounded in BR theory [3, 40]. Integrating KDPD into the NeuroAIReh@b ecosystem yields several distinct advantages:

Adherence to Minimal Change: The principle of *minimal change* ensures that profile updates reflect only the information conveyed by new evidence, thereby precluding arbitrary or excessive modifications. This is particularly pertinent in CR, where a single CTT commonly engages multiple cognitive domains. The non-linear mapping from performance metrics to domain-level alterations necessitates a disciplined approach to avoid the over-interpretation of noisy or ambiguous data.

Explainability and Auditability: The KDPD framework offers enhanced *explainability* and *auditability*. Unlike opaque ML models, the revision process is governed by explicit wffs that encode the constraints derived from performance data. Consequently, clinicians and researchers can trace the rationale for each profile modification to the underlying logical conditions and the specific distance metric employed.

Operational Versatility: The availability of *multiple operational models* (Models 1 through 4, as detailed in Section 4.3) enables the system to accommodate diverse clinical scenarios. These include the maintenance of a single definitive profile for session planning (Model 1), the exploration of plausible alternatives (Model 2), or the aggregation across multiple candidate profiles in the presence of uncertainty (Models 3 and 4).

Integration of Clinical Priority: The *priority-based tie-breaking* mechanism facilitates therapist input to resolve ambiguities when several profiles are equally consistent with the observed performance. This ensures that clinical priorities—such as the relative importance of working memory versus processing speed in a specific rehabilitation phase—are rigorously respected during the revision process.

4.5.2 Profile Creation

The initial patient profile within NeuroAIRh@b is constructed through the aggregation of NPA instruments, as detailed in Section 3.2. The CS is computed via a weighted sum of normalised NPA scores across cognitive domains and subdomains (Equation 1), whilst the CS is obtained by contextualising the ACP against normative data (Equation 2). The resulting profile is represented as a tuple $\ll ACP_1, \dots, ACP_r \gg$, where each ACP_k corresponds to a specific cognitive domain or subdomain and assumes values within a bounded interval (e.g. $[0, 100]$).

To align with the KDPD implementation, this profile is mapped to the formal structure specified in Definition 4.1.1: the tuple of labels $\mathbb{L} = \ll L_1, \dots, L_n \gg$ corresponds to the cognitive domains (e.g. working memory, sustained attention, verbal fluency), and each domain L_i is associated with a discrete, bounded integer domain $D_i \subseteq \mathbb{Z}$. This mapping procedure may involve rounding or discretisation if the original ACP values are continuous; the level of granularity is selected to preserve clinically significant distinctions whilst remaining computationally tractable.

The profile generated through this process serves as the baseline P_0 for the training programme. No KDPD revision is applied at this juncture, as the creation process is governed exclusively by the NPA aggregation formulae. The KDPD framework is integrated into the workflow from the commencement of the first training session, when performance-derived data become available for processing.

4.5.3 Translating Performance into Well-Formed Formulae

A critical stage in the integration process involves the translation of CTT performance data into the wffs that drive profile revision. As established in Section 3.4, each CTT maintains a specific correlation between its operational parameters and the cognitive domains it engages. Upon the completion of a session’s task set, the system records performance metrics—such as success rates, response latencies, and the difficulty parameters at which the 50%–70% target threshold was attained. The primary challenge lies in expressing these empirical observations as formal logical constraints within the profile.

The translation procedure follows a structured methodology. For each CTT executed during a session, the system retains: (i) the task parameters (e.g. difficulty levels per domain), (ii) the observed performance (e.g. success rate), and (iii) the domain–parameter correlation models. If performance surpasses a predefined threshold—for instance, a success rate exceeding 65% at a specific difficulty level—the system infers that the patient’s capacity within the relevant domains is at least sufficient to sustain that level. This inference is subsequently encoded as a constraint of the form $L_i \geq \theta_i$ or $L_i > \theta_i$, where θ_i is derived from the task difficulty and the corresponding correlation model. Conversely, should performance fall below a threshold, a constraint such as $L_i \leq \theta_i$ may be inferred.

Given that a single CTT typically necessitates the involvement of multiple cognitive domains, the constraints for a given session are synthesised into a conjunctive clause. In instances where multiple CTTs produce constraints on the same domain, the system may construct a DNF to encapsulate various plausible interpretations. Alternatively, it may aggregate these into a single conjunctive clause utilising the most conservative or informative bounds. The resultant wff α is expressed within the language $\mathcal{L}_p$ (Definition 9) and serves as the primary input for the KDPD revision operator.

For example, should a patient achieve a 68% success rate on a task engaging working memory (L_1) and sustained attention (L_2) at difficulty levels 60 and 55 respectively, the system might generate a wff such as $\alpha = (L_1 \geq 55 \wedge L_2 \geq 50)$. The precise mapping from performance metrics to thresholds remains a matter of system calibration, informed by the ML models described in Section 3.2 and the CTT–domain correlations established within the Reh@bSync system [180].

4.5.4 Session-by-Session Training Profile

At the conclusion of each training session (Step 8 in the NeuroAIRh@b framework), the system is required to produce an updated profile to inform the subsequent session’s task selection and parameterisation. KDPD Model 1 is the most adequate⁸ to this scenario: it revises a single profile—either the profile at the commencement of the session or that produced by the preceding session—utilising a wff α derived from the session’s performance to yield a unique revised profile $P' = P \odot \alpha$.

The workflow proceeds as follows. Let P_t denote the profile at the beginning of session t . Upon completion of the prescribed CTTs, the system aggregates the performance data and generates the wff α_t in accordance with the aforementioned procedure. The PDM subsequently invokes Model 1, providing the initial profile P_t , the wff α_t , the global value bounds, and a priority condition. This priority condition is either specified explicitly by the therapist or inferred by the system based on the specific domains targeted during the session. The resulting output, $P_{t+1} = P_t \odot \alpha_t$, is established as the profile for session $t + 1$.

The priority condition is of paramount importance in instances where multiple profiles satisfy α_t at an identical minimal Manhattan distance from P_t . As specified in Section 4.3.1, the therapist may define a label priority order—for instance, $\mathbf{el}[2] === \mathbf{el}[1] > \mathbf{el}[0]$ —indicating that domains 2 and 1 (e.g. working memory and executive function) possess an equal highest priority, followed by domain 0 (e.g. general cognition).

The tie-breaking mechanism, defined in Definition 12, subsequently selects the profile that minimises the distance relative to the highest-priority domains first. This enables the clinician to emphasise specific cognitive domains in cases where the performance data underdetermine a unique profile, thereby ensuring that the revision aligns with clinical objectives.

4.5.5 Evolution Profile Across Sessions

Over the duration of a training programme comprising multiple sessions, the profile evolves as a sequential chain:

$$P_0 \xrightarrow{\odot \alpha_1} P_1 \xrightarrow{\odot \alpha_2} P_2 \xrightarrow{\odot \alpha_3} \dots \xrightarrow{\odot \alpha_T} P_T$$

In this sequence, P_0 represents the initial profile derived from the NPA-based ACP/CS aggregation, whilst P_T denotes the *predicted cognitive profile* established at the conclusion of the programme.

This evolution profile captures the trajectory of the patient’s estimated cognitive status throughout the intervention. It may be analysed to evaluate both the rate and pattern of change: for instance, whether improvements are concentrated within the initial sessions or distributed across the entire duration, or whether specific domains exhibit more rapid progression than others. The KDPD framework ensures that each transition $P_t \rightarrow P_{t+1}$ remains minimal with respect to the Manhattan distance; consequently, the evolution reflects incremental adjustments rather than arbitrary fluctuations.

In scenarios where the system maintains multiple candidate profiles—frequently due to epistemic uncertainty—Models 2, 3, or 4 can be used. Model 2 produces a set of equally plausible profiles per session; the system may subsequently retain and propagate this set, or apply a secondary selection process (such as a therapist review) prior to the beginning of the next session.

Models 3 and 4 become relevant when the initial state is represented as a set of profiles—for example, derived from multiple baseline assessments or a population prior—and the update must preserve either the collective or individual structure of that set.

4.5.6 The Predicted Profile and Its Role

The predicted cognitive profile P_T represents the system’s estimate of the patient’s cognitive status following the completion of the training programme, derived exclusively from the performance data captured during the CTTs. It constitutes the output of the final revision within the aforementioned evolution chain.

The predicted profile serves two primary objectives. Firstly, it guides immediate post-programme planning: should further intervention be recommended, P_T is utilised as the foundational baseline for the subsequent phase. Secondly, it is compared against the newly assessed profile derived from a secondary battery of NPAs administered at the conclusion of the programme (Step 9 in the framework). This comparison is fundamental to the calibration of the system.

4.5.7 Re-evaluation and Fine-tuning Profile Creation

Upon the completion of a training programme, the patient undergoes a secondary battery of NPAs. The results are aggregated into a novel ACP' , and a corresponding CS' , utilising the same formulae employed for the initial baseline profile. The system subsequently compares ACP' with the predicted profile P_T .

Divergences between P_T and ACP' may originate from several distinct sources [148]: an erroneous prediction within the profile dynamics (Step 8); an inaccurate model of the correlation between CTTs and cognitive domains (Step 5); or a suboptimal adjustment of task parameters during the sessions (Step 7). The KDPD framework does not directly resolve these divergences; rather, it provides a rigorous, structured basis for their analysis.

The fundamental insight for fine-tuning resides in the fact that the comparison between P_T and ACP' can inform the calibration of the *profile creation* process. The following refinement pathways are identified:

- **Weight Calibration:** Should P_T systematically over- or underestimate specific domains relative to ACP' , the weights W_{kij} in Equation 1 and w_i in Equation 2 may be adjusted. These weights govern the aggregation of NPA scores into the ACP; recalibrating these parameters enhances the accuracy of future profile creation for new patients.
- **Translation Refinement:** The performance-to-woff translation, as described in Section 4.5.3, may also be refined. If the predicted profile consistently diverges from the re-assessed profile in specific domains, the mapping from CTT performance to logical constraints on those domains may require modification. For instance, the thresholds θ_i utilised in constraints such as $L_i \geq \theta_i$ could be scaled or shifted to improve alignment.
- **ML Integration:** Accumulated data from a cohort of patients—specifically the comparison between predicted and re-assessed profiles—can be utilised to train or tune ML models. These models may then suggest optimal weights and translation parameters, as envisaged in Section 3.2.4.

Consequently, the re-evaluation cycle serves to close the feedback loop: the KDPD-updated predicted profile is compared against the gold-standard NPA-derived profile, and the resulting

feedback facilitates the fine-tuning of both the initial profile creation process for future patients and the performance-to-wff translation for subsequent sessions.

4.5.8 Therapist-Driven Fine-tuning: Domain Priority

The therapist performs an essential role in resolving ambiguities during the profile revision process. As previously noted, in instances where multiple profiles satisfy the performance-derived wff at an identical minimal distance, a tie-breaking mechanism becomes necessary. The KDPD implementation incorporates a *priority condition* that specifies which profile attributes—namely, specific cognitive domains—should be prioritised during the resolution of ties.

The therapist may configure this priority based on the unique rehabilitation objectives of the individual patient. For example, should the primary objective be the improvement of working memory, the clinician may assign the highest priority to the working memory domain index. In such a scenario, when several candidate profiles are equidistant, the system selects the profile that most accurately preserves or enhances the working memory estimate. Conversely, if the clinical focus is directed towards attention, that specific domain may be prioritised accordingly.

The priority condition is expressed as an ordered partition of domain indices. For instance, the expression `e1[2] == e1[1] > e1[0] > e1[3]` indicates that domains 2 and 1 share the highest priority, followed by domain 0, and subsequently domain 3. The `parsePriorityCondition` logic, as detailed in Section 4.3.1, interprets this expression and applies the tie-breaking rule established in Definition 12: among tied candidates, the profile exhibiting the minimal distance within the highest-priority group is selected; if ties persist, the subsequent priority group is evaluated, and so forth.

This mechanism ensures that the profile dynamics rigorously respect clinical judgement. The therapist’s selection of domain priorities effectively encodes which cognitive functions are most critical for the current rehabilitation phase. The system utilises this prioritisation to disambiguate the profile in instances where performance data alone prove insufficient.

4.5.9 Summary and Integration Mapping

Table 11 summarises the mapping between the NeuroAIRh@b framework steps and the KDPD components employed at each stage of the process.

Table 11: Mapping between the NeuroAIReh@b rehabilitation workflow and the KDPD components involved at each stage. In Steps 1 to 3, NPA results are aggregated into an ACP and CS, which are mapped to the KDPD tuple representation. Steps 5 to 7 perform task selection, initial parameterisation, and within-session difficulty adaptation without applying KDPD revision. At Step 8, session evidence is translated into a wff α_t , and Model 1 revises the current profile P_t according to $P_{t+1} = P_t \odot \alpha_t$, using therapist-defined priorities to break ties between equidistant candidates. Across several sessions, successive revisions produce the predicted profile $P_T = P_0 \odot \alpha_1 \odot \dots \odot \alpha_T$. At Step 9, P_T is compared with the reassessed profile ACP' to calibrate the aggregation weights and the performance-to-formula translation. Step 4, the clinical definition of training objectives, is not represented as a separate KDPD operation in this table.

NeuroAIReh@b Step	KDPD Component	Description
1–3	Profile Structure	NPA aggregation yields ACP/CS; mapped to KDPD tuple format.
5–6	—	Task selection and initial parameterisation; no KDPD revision.
7	—	In-session difficulty adaptation; no KDPD revision.
8	Model 1, wff α_t	Performance $\rightarrow$ wff $\rightarrow P_{t+1} = P_t \odot \alpha_t$; priority for tie-breaking.
8 (multi-session)	Evolution Chain	$P_0 \odot \alpha_1 \odot \dots \odot \alpha_T = P_T$ (predicted profile).
9	Comparison & Calibration	P_T vs. CS'; feedback to weights and wff translation.

The integration of the KDPD framework into the NeuroAIReh@b system thus provides a principled, explainable, and clinician-guided mechanism for profile dynamics throughout the rehabilitation cycle. Future research may encompass empirical validation studies comparing KDPD-based profile updates with alternative methodologies, alongside the extension of the performance-to-wff translation process. This extension could incorporate learned or adaptive mappings derived from more extensive longitudinal datasets, further enhancing the system's predictive accuracy and clinical utility.

5 Conclusion and Future Work

This chapter synthesises the principal contributions of this thesis and addresses the research questions posed in the introduction. It begins with a summary of the implementation achievements, proceeds to provide explicit answers to each research question, and concludes with directions for future work.

5.1 Implementation Summary

This thesis has presented a complete and operational implementation of the KDPD framework, directly addressing the persistent gap between the extensive theoretical body of work in AGM-style BR and its practical application to user profile management. The primary contribution is a robust modular system that renders the principles of rational, logic-based profile dynamics computationally tractable. By operationalising profiles as integer tuples and employing the Manhattan distance, the system provides a concrete instantiation of the principle of minimal change. The system architecture—centred on the synergistic interaction of a custom Abstract Syntax Tree Engine, a DMS, and a TFM—successfully manages a rich constraint language encompassing arithmetic and relational conditions, establishing it as a versatile tool for profile manipulation.

The efficacy of the system and its adherence to the underlying theory have been demonstrated through the implementation of all four distinct KDPD models for revising and updating both individual profiles and sets of profiles. Benchmark results confirm that performance is primarily governed by the domain size and the structural arity of the constraining formula, rather than the nominal length of the profile tuple, thereby validating the architectural design choices. This work serves not only as a crucial proof-of-concept for the viability of logic-based profile dynamics but also provides a foundational platform upon which further research and more complex applications may be developed.

Furthermore, the thesis has detailed the integration pathway for applying the KDPD implementation to the NeuroAIReh@b CR framework. The mapping from NeuroAIReh@b workflow steps to KDPD components—including the translation of performance data into wffs, the session-by-session revision mechanism, and the therapist-driven priority specification—establishes a concrete blueprint for deployment within real-world clinical settings.

5.2 Answers to the Research Questions

The introduction formulated three research questions. This subsection provides explicit answers based upon the contributions of this thesis.

5.2.1 RQ1: Insights from Early Belief Revision Implementations

What insights can be extracted from the early BR implementations identified by Doyle and London to inform and enhance modern computational implementations of the AGM postulates?

The thesis affirms that valuable insights can indeed be extracted from pre-AGM implementations. Chapter 2 undertook a systematic review of the systems catalogued in Doyle and London’s bibliography [37], revealing that early work emphasised representational devices (e.g. logical data dependencies, context-layered databases) and procedural strategies (e.g. dependency-directed backtracking, non-chronological backtracking) that prioritised computational tractability. These concerns were largely sidelined during the post-AGM focus on logical postulates, yet they remain essential for practical deployment.

The KDPD implementation incorporates several such insights. The use of explicit dependency structures—manifest in the AS_Te’s representation of constraint relationships and the DMS’s domain-narrowing procedures—echoes the dependency-based revision approaches identified in the pre-AGM literature. The distance-guided DFS with pruning reflects the principle that revision should proceed by exploring minimal alternatives, analogous to the preference for minimal-change strategies in early TMS. The decomposition of formulae into independent conjunctive clauses for parallel processing aligns with the modular, incremental processing patterns observed in pre-AGM systems. Thus, the thesis demonstrates that the integration of these historical implementation strategies with the AGM theoretical framework yields a system that is both formally grounded and computationally viable.

5.2.2 RQ2: A Unified Knowledge-Based Formalism for Profile Dynamics

Can a unified, knowledge-based formalism be formulated to govern the structure and dynamics of user profiles?

The thesis answers this question affirmatively. The KDPD framework [40], as formalised by Fermé et al. and implemented herein, provides a unified formalism that governs both the structure

and the dynamics of user profiles. The structure is defined by a tuple of labels $\mathbb{L}$ with associated domains D_i , yielding profiles as elements of $D_1 \times \dots \times D_n$. The profile language $\mathcal{L}_p$ extends this structure with arithmetic and relational operators, enabling the expression of constraints over profile attributes in wffs. The dynamics are governed by four axiomatically characterised operators—revision and update for single profiles (Models 1 and 2) and for sets of profiles (Models 3 and 4)—each satisfying the rationality postulates derived from AGM theory. The Manhattan distance metric operationalises the principle of minimal change, inducing orderings that determine which profiles are selected when multiple candidates satisfy a given constraint.

This formalism is domain-agnostic: it applies to cognitive profiles in rehabilitation (as in NeuroAIR@b), to recommender system user models, and to any setting where profiles are represented as tuples of attributes subject to logical constraints. The implementation presented in Chapter 4 demonstrates that this formalism is not merely theoretical but can be rendered computationally tractable for practical applications.

5.2.3 RQ3: Stability, Interpretability, and Accuracy of Belief Revision-Based

Profiles

Can the integration of BR algorithms produce user profiles that are more stable, interpretable, and accurate in their evolution than traditional DL approaches?

The thesis provides a qualified affirmative answer. With respect to *stability*, the KDPD framework enforces the principle of minimal change: each revision selects the profile that minimises the Manhattan distance from the prior profile whilst satisfying the new constraint. This design directly addresses the recency bias and volatility characteristic of data-driven models, wherein transient user actions can induce disproportionate updates. The KDPD operators ensure that profile evolution reflects only the information conveyed by the evidence, precluding arbitrary or excessive modifications.

With respect to *interpretability*, the framework offers a substantial advantage over opaque DL models. Each profile update is driven by an explicit wff α that encodes the logical constraints derived from the observed data. Clinicians and system operators can trace the rationale for each modification to the underlying constraint and the distance metric employed. The therapist-driven priority mechanism further enhances interpretability by allowing domain experts to specify which attributes should be preserved when multiple profiles are equally consistent with the evidence.

With respect to *accuracy*, the thesis has established the theoretical and computational foundations for KDPD-based profile dynamics and has demonstrated its integration into NeuroAIR@b. However, a rigorous empirical comparison between KDPD-updated profiles and those produced by traditional DL approaches has not yet been conducted. Such a comparative study would require longitudinal data from CR programmes, with predicted profiles compared against gold-standard neuropsychological reassessments under both methodologies. The thesis therefore identifies this as a critical direction for future work. The framework’s adherence to minimal change and its explicit constraint structure provide strong theoretical grounds to expect improved accuracy in settings where data are sparse or noisy; empirical validation remains to be undertaken.

5.3 Future Work

Despite these achievements, the current implementation presents several opportunities for enhancement and further investigation. The most pertinent directions for future work are outlined below.

5.3.1 Empirical Validation Methodology

A rigorous empirical comparison between KDPD-updated profiles and alternative approaches would require several components. First, longitudinal data from CR programmes would be necessary to capture the evolution of patient profiles across multiple sessions. Second, the predicted profiles generated by KDPD would need to be compared against a baseline—such as simple averaging or a ML model—with both methodologies evaluated against gold-standard neuropsychological reassessments. Third, the evaluation would require metrics addressing each dimension of interest: stability (e.g. variance over sessions), interpretability (e.g. traceability of updates to their underlying constraints), and accuracy (e.g. agreement with reassessments). A simulation study on synthetic data could provide preliminary evidence of the framework’s behaviour prior to undertaking full clinical trials.

5.3.2 Algorithmic and Heuristic Enhancements

The current system employs a straightforward domain-narrowing algorithm and a distance-guided DFS with pruning. Significant performance gains could be achieved by incorporating more advanced techniques from the field of Constraint Satisfaction Problems. Future work should focus on implementing more sophisticated constraint-propagation algorithms—such as those enforcing arc

consistency—to achieve tighter domain bounds, particularly for challenging non-linear constraints involving multiplication and division, as identified in Section 4.4.3.

Furthermore, the search strategy could be substantially improved by moving beyond the current DFS. An exploration of heuristic search algorithms, such as A^* , utilising the Manhattan distance as the heuristic function $h(n)$, could guide the search more efficiently towards optimal solutions. This could be augmented with dynamic variable-ordering heuristics, such as the *minimum-remaining-values* approach, to prune the search space more aggressively.

5.3.3 Performance and Scalability Optimisation

The current single-threaded JavaScript implementation, whilst sufficient for demonstrating feasibility, presents a bottleneck for large-scale applications. The inherent structure of the problem—particularly the decomposition of DNF formulae—lends itself naturally to parallelisation.

A future iteration of the system should explore parallel processing, distributing the search for solutions to independent conjunctive clauses across multiple cores or nodes. For production-grade performance, porting the core computational modules (TFM and DMS) to a lower-level, compiled language such as C++ or Rust (or to WebAssembly for web environments) would offer substantial speed improvements through finer control over memory management and execution. This would effectively create a high-performance library for KDPD operations.

5.3.4 Framework and Language Extensions

The foundational framework itself offers several avenues for expansion. The profile language $\mathcal{L}_p$ could be enriched to support a wider range of data types beyond integers—such as categorical or floating-point values—which would necessitate the definition and integration of appropriate distance metrics.

Investigating the impact of alternative distance functions (e.g. Chebyshev distance) on the satisfaction of the rationality postulates represents another fertile area of theoretical and practical research. Finally, a particularly promising direction lies in the development of hybrid systems. ML models could be employed to learn or suggest the wffs (α) from user data, which are then applied via the transparent and rationally grounded KDPD framework.

Such an approach would synergise the pattern-recognition strengths of data-driven methods with the explainability and predictability of the knowledge-driven system, representing a significant step towards more trustworthy and robust personalisation engines.

6 Appendices

6.1 Proofs

6.1.1 Model 1 Postulates

The revision operation $P \odot \alpha$ is realised by the `model1`($P, \alpha, \text{minValue}, \text{maxValue}, \text{priorityCondition}$) function. This function begins by invoking the procedure:

$$(T_{P,\alpha}, \delta_{P,\alpha}) = \text{findTuplesWithConstraints}(P, \alpha, \text{minValue}, \text{maxValue})$$

Here, $T_{P,\alpha}$ is the set of all profiles that satisfy the formula α and are at the minimum Manhattan distance $\delta_{P,\alpha}$ from the initial profile P . If the set $T_{P,\alpha}$ contains more than one profile, the `model1` function then employs a selection mechanism—`findBestTuple`($T_{P,\alpha}, P, \text{priorityCondition}$)—to select a single, unique profile, denoted X^* , from $T_{P,\alpha}$ based upon the specified *priorityCondition*. The final result of the revision, $P \odot \alpha$, is thus the singleton set $\{X^*\}$, or it is $T_{P,\alpha}$ if the set initially contained at most one profile.

(P1) If $\|\alpha\| \neq \emptyset$, then $P \odot \alpha \in \|\alpha\|$

This postulate asserts that if the wff α is satisfiable (i.e. has models), the revised profile $P \odot \alpha$ must be one of those models.

Fulfilment:

Assume $\|\alpha\| \neq \emptyset$. The `findTuplesWithConstraints` procedure, by definition, computes the set $T_{P,\alpha}$ containing exclusively models of α at a minimum Manhattan distance from P . Therefore, $T_{P,\alpha} \subseteq \|\alpha\|$. The subsequent `findBestTuple` function selects a single element $\tau^* \in T_{P,\alpha}$. The revision is thus defined as $P \odot \alpha = \tau^*$. It follows directly that $P \odot \alpha \in \|\alpha\|$, thereby satisfying the postulate.

(P2) If $\|\alpha\| = \emptyset$, then $P \odot \alpha = P$

This postulate states that if α is unsatisfiable (possesses no models), the revision operation should leave the original profile P unchanged.

Fulfilment:

Assume $\|\alpha\| = \emptyset$. The implementation addresses two conditions under which a model of α cannot be found. Firstly, if the wff string is syntactically invalid, the `parseCondition` function

fails and the original profile P is returned. Secondly, for a syntactically valid but semantically unsatisfiable wff, the `narrowDomainMap` procedure identifies a contradiction, resulting in an empty search space.

Consequently, the `findTuplesWithConstraints` procedure does not identify any satisfying tuples, causing its internal minimum distance parameter to remain at its initial state of infinity. The procedure is designed to return the original profile P in this eventuality.

In all cases where no model of α can be found, the operation yields $P \odot \alpha = P$, thereby satisfying the postulate.

(P3) If $P \models \alpha$, then $P \odot \alpha = P$

This postulate requires that if the original profile P already satisfies the formula α , then the revision by α leaves P unchanged.

Fulfilment:

Assume $P \models \alpha$. The `findTuplesWithConstraints` procedure first evaluates the satisfaction of the input formula for the initial profile. Since P is a model of α , the result of this evaluation is true, and its Manhattan distance to itself is the minimum achievable distance (zero). The procedure therefore identifies P as the unique optimal solution and returns the set $\{P\}$ with a distance of zero. The `model11` function propagates this result, yielding $P \odot \alpha = P$ as required.

(P4) If $\|\alpha\| = \|\beta\|$, then $P \odot \alpha = P \odot \beta$

This postulate requires that the revision operator depends only on the semantic content of the input formula—namely its set of models—and not on its particular syntactic form.

Fulfilment:

Assume $\|\alpha\| = \|\beta\|$. The revision operator is defined by procedures that depend exclusively on the semantic properties of the input formula, rather than its syntactic representation. The `parseCondition`, `narrowDomainMap`, and distance-guided search functions operate on the logical constraints imposed by the formula, which are identical for any two formulae sharing the same model set. Given the assumption, any tuple τ satisfies $\tau \models \alpha$ if and only if $\tau \models \beta$; thus, the sets of minimal-distance models $T_{P,\alpha}$ and $T_{P,\beta}$ identified by `findTuplesWithConstraints` will be identical. The deterministic `findBestTuple` function subsequently selects the same unique profile from this identical set. Therefore, $P \odot \alpha = P \odot \beta$.

(P5) $P \odot (\alpha \vee \beta) = P \odot \alpha$ or $P \odot (\alpha \vee \beta) = P \odot \beta$

This postulate requires that when revising by a disjunction $\alpha \vee \beta$, the resulting profile must coincide with the revision by one of the individual disjuncts.

Fulfilment:

The implementation processes the disjunction $\alpha \vee \beta$ by independently determining the set of closest models for each disjunct. Let $T_{P,\alpha}$ be the set of models of α at the minimum distance $\delta_{P,\alpha}$, and let $T_{P,\beta}$ be the set for β at distance $\delta_{P,\beta}$. The procedure then identifies the overall minimum distance, $\delta = \min(\delta_{P,\alpha}, \delta_{P,\beta})$. The final set of candidates for selection, $T_{P,\alpha \vee \beta}$, comprises all profiles from $T_{P,\alpha} \cup T_{P,\beta}$ whose distance from P is equal to δ . The final result, $P \odot (\alpha \vee \beta)$, is a unique profile selected from $T_{P,\alpha \vee \beta}$.

If $\delta_{P,\alpha} < \delta_{P,\beta}$, then $T_{P,\alpha \vee \beta} = T_{P,\alpha}$, and thus $P \odot (\alpha \vee \beta) = P \odot \alpha$.

If $\delta_{P,\beta} < \delta_{P,\alpha}$, then $T_{P,\alpha \vee \beta} = T_{P,\beta}$, and thus $P \odot (\alpha \vee \beta) = P \odot \beta$.

If $\delta_{P,\beta} = \delta_{P,\alpha}$, the selected profile will be drawn from $T_{P,\alpha} \cup T_{P,\beta}$ and will be identical to the profile that would be selected from either $T_{P,\alpha}$ or $T_{P,\beta}$ based on the deterministic priority tie-breaking.

Thus, the result is always equivalent to either $P \odot \alpha$ or $P \odot \beta$.

(P6) If $P \odot \alpha \models \beta$ and $P \odot \beta \models \alpha$, then $P \odot \alpha = P \odot \beta$.

This postulate requires that whenever revising by α yields a profile satisfying β , and revising by β yields a profile satisfying α , both revision results must coincide.

Fulfilment:

Let $P_\alpha = P \odot \alpha$ and $P_\beta = P \odot \beta$. The premises state that $P_\alpha \models \beta$ and $P_\beta \models \alpha$. From Postulate **P1**, it is also established that $P_\alpha \models \alpha$ and $P_\beta \models \beta$. It follows that both P_α and P_β are models of the conjunction; that is, $\{P_\alpha, P_\beta\} \subseteq \|\alpha \wedge \beta\|$.

By definition of the operator, $d_M(P, P_\alpha) = \delta_\alpha$ and $d_M(P, P_\beta) = \delta_\beta$. Since $P_\alpha \in \|\alpha \wedge \beta\|$, it must be the case that $\delta_{P,\alpha \wedge \beta} \leq \delta_{P,\alpha}$. Similarly, $\delta_{P,\alpha \wedge \beta} \leq \delta_{P,\beta}$. If $\delta_{P,\alpha} < \delta_{P,\beta}$, then P_α would be a model of β closer to P than P_β , which contradicts the definition of P_β as a minimally distant model. A symmetric argument excludes the possibility that $\delta_{P,\beta} < \delta_{P,\alpha}$.

Therefore, it must be that $\delta_{P,\alpha} = \delta_{P,\beta}$. As both P_α and P_β belong to the same minimal-distance set relative to P , and the `findBestTuple` selection function is deterministic, it must select the same unique profile in both instances. Thus, $P_\alpha = P_\beta$, thereby satisfying the postulate.

6.1.2 Model 2 Postulates

The one-to-many revision operation $P \odot \alpha$ is realised by the `model12`($P, \alpha, \text{minValue}, \text{maxValue}$) function. The primary step of this function is to invoke the procedure:

$$(T_{P,\alpha}, \delta_{P,\alpha}) = \text{findTuplesWithConstraints}(P, \alpha, \text{minValue}, \text{maxValue})$$

The set $T_{P,\alpha}$ obtained from this procedure consists of all profiles τ that satisfy the formula α and are at the minimum Manhattan distance $\delta_{P,\alpha}$ from the initial profile P . The `model12` function then directly returns this set $T_{P,\alpha}$ (along with the distance $\delta_{P,\alpha}$) as the outcome of the revision $P \odot \alpha$. In summary, `model12` is operationally identical to `model11`, but excludes the `findBestTuple` tie-breaking procedure.

(S1) If $\|\alpha\| \neq \emptyset$, then $P \odot \alpha \subseteq \|\alpha\|$

This postulate asserts that whenever the wff α is satisfiable, every profile produced by the revision $P \odot \alpha$ must itself be a model of α .

Fulfilment:

The operator $P \odot \alpha$ is realised by the `findTuplesWithConstraints` procedure. A core step in this procedure is the construction of an executable predicate that is extensionally equivalent to the satisfaction relation for α ($\tau \models \alpha$). The search mechanism considers for inclusion in the final result set only those tuples for which this predicate evaluates to true. Consequently, any tuple τ in the returned set $P \odot \alpha$ must be a model of α . Therefore, $P \odot \alpha \subseteq \|\alpha\|$.

(S2) If $\|\alpha\| = \emptyset$, then $P \odot \alpha = \{P\}$

This postulate stipulates that when the wff α is unsatisfiable, such that it possesses no models, the revision operator should return exactly the singleton set containing the original profile P .

Fulfilment:

Assume $\|\alpha\| = \emptyset$. The implementation addresses this scenario in two ways. Firstly, for syntactically invalid or trivial formulae, the procedure returns $\{P\}$ immediately. Secondly, for semantically

unsatisfiable but valid formulae, the domain-narrowing algorithm detects a contradiction, resulting in an empty search space.

The search mechanism therefore fails to identify models of α . This condition is detected by the procedure, which is designed to return $\{P\}$ as a fallback. In all cases where $\|\alpha\| = \emptyset$, the operator yields $P \odot \alpha = \{P\}$, thereby satisfying the postulate.

(S3) $P \odot \alpha \neq \emptyset$

This postulate demands that for every wff α , the result of revising P by α must yield at least one profile.

Fulfilment:

Let α be an arbitrary wff. Two cases arise:

Case 1: Assume $\|\alpha\| \neq \emptyset$. According to **S1**, it holds that $P \odot \alpha \subseteq \|\alpha\|$. The revision mechanism—specifically the distance-guided exhaustive search implemented within `findTuplesWithConstraints`—is designed to always produce at least one tuple $\tau \in \|\alpha\|$ when $\|\alpha\|$ is non-empty. Thus, $P \odot \alpha \neq \emptyset$.

Case 2: Assume $\|\alpha\| = \emptyset$. According to **S2**, the revision is given by $P \odot \alpha = \{P\}$. Given that P is the initial profile and is inherently well-defined, the set $\{P\}$ is non-empty. Thus, $P \odot \alpha \neq \emptyset$.

In both considered cases, the revision $P \odot \alpha$ results in a non-empty set, thereby satisfying the postulate.

(S4) If $\|\alpha\| = \|\beta\|$, then $P \odot \alpha = P \odot \beta$

This postulate requires that revising by two logically equivalent formulae—that is, those with identical model sets—yields the same set of revised profiles.

Fulfilment:

The revision operator procedures are determined by the semantic content of the input formula, rather than its syntactic structure. The premise $\|\alpha\| = \|\beta\|$ implies that for any tuple τ , $\tau \models \alpha$ if and only if $\tau \models \beta$.

Consequently, the AST-derived constraints, the narrowed domain maps, and the executable satisfaction predicates will be extensionally identical for both α and β . Since all subsequent distance calculations and search procedures are based upon these identical semantic foundations, the final

set of minimally distant tuples returned must also be identical. Thus, $P \odot \alpha = P \odot \beta$, thereby satisfying the postulate.

(S5) If $P \models \alpha$, then $P \odot \alpha = \{P\}$

This postulate requires that if the original profile P already satisfies the formula α , then the revision by α yields no change, resulting in the singleton set $\{P\}$.

Fulfilment:

Assume $P \models \alpha$. The `findTuplesWithConstraints` procedure first evaluates the satisfaction of α for the initial profile P . Given the premise, this evaluation yields true. The distance from P to itself is $d_M(P, P) = 0$.

As no profile τ can possess a distance to P less than zero, P is identified as an optimal solution. Any other model $\tau' \in \|\alpha\|$, where $\tau' \neq P$, would necessarily result in $d_M(P, \tau') > 0$. Therefore, P is the unique minimally distant model. The procedure returns $\{P\}$ with a distance of 0, thereby satisfying the postulate.

(S6) $P \odot (\alpha \vee \beta) = P \odot \alpha$ or $P \odot (\alpha \vee \beta) = P \odot \beta$ or $P \odot (\alpha \vee \beta) = P \odot \alpha \cup P \odot \beta$

This postulate requires that when revising by a disjunction $\alpha \vee \beta$, the resulting set of profiles must coincide either with the revision by α alone, the revision by β alone, or the union of both revisions.

Fulfilment:

Let $T_{P,\phi}$ be the set of tuples and $\delta_{P,\phi}$ be the minimum Manhattan distance associated with $P \odot \phi$, such that $T_{P,\phi} = \{\tau \mid \tau \models \phi \text{ and } d_M(P, \tau) = \delta_{P,\phi}\}$, as returned by `findTuplesWithConstraints`.

Consider the computation of $P \odot (\alpha \vee \beta)$. When provided with the condition $\alpha \vee \beta$, the `findTuplesWithConstraints` function processes the input by splitting the condition string at the top-level "||" operators. This effectively separates the disjunction into its constituent parts: $orParts = [\alpha, \beta]$. The function then iterates through these components.

Internally, `findTuplesWithConstraints` maintains the variables *globalMinDistance* (initialised as infinity) and *globalAssignments* (an empty list). When the first component, α , is processed by the internal procedure `_findTuplesForConjunction`, it yields the set $T_{P,\alpha}$ at distance $\delta_{P,\alpha}$. The state is updated accordingly: $globalMinDistance \leftarrow \delta_{P,\alpha}$ and $globalAssignments \leftarrow T_{P,\alpha}$.

Subsequently, when the second component, β , is processed, it yields $T_{P,\beta}$ at distance $\delta_{P,\beta}$. The aggregation logic within `findTuplesWithConstraints` then compares $\delta_{P,\beta}$ with the current *globalMinDistance* (which is $\delta_{P,\alpha}$).

- If $\delta_{P,\beta} < \delta_{P,\alpha}$, then *globalMinDistance* is updated to $\delta_{P,\beta}$ and *globalAssignments* is replaced by $T_{P,\beta}$. In this instance, the final result $P \odot (\alpha \vee \beta)$ is equal to $T_{P,\beta}$, which is $P \odot \beta$.
- If $\delta_{P,\beta} > \delta_{P,\alpha}$, then *globalMinDistance* and *globalAssignments* remain unchanged from their state after processing α . Thus, $P \odot (\alpha \vee \beta) = T_{P,\alpha} = P \odot \alpha$.
- If $\delta_{P,\beta} = \delta_{P,\alpha}$, the tuples from $T_{P,\beta}$ are added to *globalAssignments* (ensuring uniqueness). In this case, *globalAssignments* becomes $T_{P,\alpha} \cup T_{P,\beta}$. Thus, $P \odot (\alpha \vee \beta) = (P \odot \alpha) \cup (P \odot \beta)$.

In all scenarios, the implementation adheres to the requirements of the postulate.

This operational behaviour of `findTuplesWithConstraints`—the result of which is directly returned by `model2`—ensures that $P \odot (\alpha \vee \beta)$ is identical to $P \odot \alpha$, to $P \odot \beta$, or to their union, thereby satisfying Postulate **S6**.

$$(S7) \quad P \odot (\alpha \vee \beta) \subseteq P \odot \alpha \cup P \odot \beta$$

This postulate requires that every profile returned as a minimally distant model of the disjunction $\alpha \vee \beta$ must also appear as a minimally distant model of at least one of the individual disjuncts.

Fulfilment:

The revision $P \odot (\alpha \vee \beta)$ is computed by `model2(P,  $\alpha \vee \beta$ )`, which invokes `findTuplesWithConstraints(P,  $\alpha \vee \beta$ )`. The `findTuplesWithConstraints` function processes the condition string by splitting it based upon the top-level logical OR operator ("||"). This yields a set of disjunctive components, denoted as *orParts*. For the expression $\alpha \vee \beta$, this results in *orParts* = $[\alpha, \beta]$.

The algorithm iterates through each component $\gamma \in \textit{orParts}$. For each such γ , the internal procedure (`_findTuplesForConjunction`) determines the set T_γ of tuples that satisfy γ and are closest to P . Let this minimum distance be $\Delta_\gamma = \min_{\tau \models \gamma} d_M(P, \tau)$, such that $T_\gamma = \{\tau \in \|\gamma\| \mid d_M(P, \tau) = \Delta_\gamma\}$.

Throughout this iterative process, **findTuplesWithConstraints** maintains a record of the minimum distance identified thus far, *globalMinDistance*, and the set of assignments (tuples) achieving this distance, *globalAssignments*. When a disjunct γ is processed:

- If $\Delta_\gamma < \text{globalMinDistance}_{\text{old}}$ (where $\text{globalMinDistance}_{\text{old}}$ is the minimum distance prior to processing γ), then *globalMinDistance* is updated to Δ_γ , and *globalAssignments* is reset to T_γ .
- If $\Delta_\gamma = \text{globalMinDistance}_{\text{old}}$, then the tuples in T_γ are added to *globalAssignments*, whilst maintaining uniqueness.
- If $\Delta_\gamma > \text{globalMinDistance}_{\text{old}}$, *globalAssignments* remains unaltered by T_γ .

After all disjuncts in *orParts* (namely α and β) have been processed, the final set of tuples returned by the **findTuplesWithConstraints** function is *globalAssignments*. Thus, $P \odot (\alpha \vee \beta) = \text{globalAssignments}$.

Crucially, any tuple $\tau \in \text{globalAssignments}$ must have originated from some T_γ where $\Delta_\gamma = \text{globalMinDistance}$. Specifically, τ must belong to T_α if $\Delta_\alpha \leq \Delta_\beta$, or to T_β if $\Delta_\beta \leq \Delta_\alpha$. If $\Delta_\alpha = \Delta_\beta$, it may belong to both, and is certainly contained within their union. Given that $P \odot \alpha$ represents the set of closest models to P for α (i.e. T_α), and $P \odot \beta$ represents T_β , it follows that any $\tau \in \text{globalAssignments}$ must be an element of $T_\alpha \cup T_\beta$. Therefore,

$$P \odot (\alpha \vee \beta) \subseteq T_\alpha \cup T_\beta = (P \odot \alpha) \cup (P \odot \beta),$$

which satisfies postulate **S7**.

(S8) If $P \odot \alpha \models \beta$ and $P \odot \beta \models \alpha$, then $P \odot \alpha = P \odot \beta$.

This postulate requires that whenever every profile produced by revising P with α also satisfies β , and likewise every profile produced by revising P with β also satisfies α , the two revision results must coincide.

Fulfilment: Let $T_{P,\phi}$ denote the set of tuples and $\delta_{P,\phi}$ the minimum Manhattan distance for this operation, such that $T_{P,\phi} = \{\tau \mid \tau \models \phi \text{ and } d_M(P, \tau) = \delta_{P,\phi}\}$. Thus, $P \odot \phi = T_{P,\phi}$.

Assume (1) $P \odot \alpha \models \beta$ and (2) $P \odot \beta \models \alpha$. Premise (1) implies that every tuple $\tau \in T_{P,\alpha}$ also satisfies β . Since, by the definition of $T_{P,\alpha}$, every $\tau \in T_{P,\alpha}$ satisfies α , it follows that all tuples in $T_{P,\alpha}$ satisfy $\alpha \wedge \beta$. Therefore, $T_{P,\alpha} \subseteq \|\alpha \wedge \beta\|$.

As all tuples in $T_{P,\alpha}$ satisfy $\alpha \wedge \beta$ and are at distance $\delta_{P,\alpha}$ from P , the minimum distance from P to any model of $\alpha \wedge \beta$, denoted $\delta_{P,\alpha \wedge \beta}$, must be less than or equal to $\delta_{P,\alpha}$. Conversely, any tuple $\tau' \in T_{P,\alpha \wedge \beta}$ (the set of closest models of $\alpha \wedge \beta$ to P) must satisfy α . Since $\delta_{P,\alpha}$ is the minimum distance from P to models of α , it must be the case that $\delta_{P,\alpha} \leq d_M(P, \tau') = \delta_{P,\alpha \wedge \beta}$.

Combining these observations, it is concluded that $\delta_{P,\alpha} = \delta_{P,\alpha \wedge \beta}$. This implies that $T_{P,\alpha}$ is precisely the set of models of $\alpha \wedge \beta$ that are at distance $\delta_{P,\alpha}$ from P ; thus, $T_{P,\alpha} = T_{P,\alpha \wedge \beta}$.

By a symmetrical argument utilising premise (2), it is deduced that $T_{P,\beta} \subseteq \|\alpha \wedge \beta\|$. This leads to the conclusion that $\delta_{P,\beta} = \delta_{P,\alpha \wedge \beta}$, meaning $T_{P,\beta}$ is the set of models of $\alpha \wedge \beta$ at distance $\delta_{P,\beta}$ from P ; thus, $T_{P,\beta} = T_{P,\alpha \wedge \beta}$.

From these deductions, $\delta_{P,\alpha} = \delta_{P,\alpha \wedge \beta}$ and $\delta_{P,\beta} = \delta_{P,\alpha \wedge \beta}$, which implies $\delta_{P,\alpha} = \delta_{P,\beta}$. Let this common minimum distance be δ^* . Then:

$$T_{P,\alpha} = \{\tau \mid \tau \models (\alpha \wedge \beta) \text{ and } d_M(P, \tau) = \delta^*\}$$

and

$$T_{P,\beta} = \{\tau \mid \tau \models (\alpha \wedge \beta) \text{ and } d_M(P, \tau) = \delta^*\}$$

Since both $T_{P,\alpha}$ and $T_{P,\beta}$ are defined as the set of all models of $\alpha \wedge \beta$ at the identical minimum distance δ^* from P , they must constitute the same set. Therefore, $T_{P,\alpha} = T_{P,\beta}$, which signifies that $P \odot \alpha = P \odot \beta$, thereby satisfying the postulate.

$$(S9) \quad P \odot \alpha \cap P \odot \beta \subseteq P \odot (\alpha \vee \beta)$$

This postulate asserts that any profile common to both the revision of P by α and the revision of P by β must also be present in the set of profiles resulting from revising P by the disjunction $\alpha \vee \beta$.

Fulfilment:

Let $T_{P,\phi}$ denote the set of tuples and $\delta_{P,\phi}$ the minimum Manhattan distance for this operation, such that $T_{P,\phi} = \{\tau \mid \tau \models \phi \text{ and } d_M(P, \tau) = \delta_{P,\phi}\}$. Thus, $P \odot \phi = T_{P,\phi}$.

Assume $X \in (P \odot \alpha \cap P \odot \beta)$. This implies that $X \in T_{P,\alpha}$ and $X \in T_{P,\beta}$. Consequently, $X \models \alpha$ with its distance from P being $d_M(P, X) = \delta_{P,\alpha}$, and $X \models \beta$ with its distance from P also being

$d_M(P, X) = \delta_{P, \beta}$. It follows directly that $\delta_{P, \alpha} = \delta_{P, \beta} = d_M(P, X)$, as X achieves the minimum distance from P for both individual revisions.

Now, consider the computation of $P \odot (\alpha \vee \beta)$, which is determined by `findTuplesWithConstraints`($P, \alpha \vee \beta, \dots$). This procedure internally processes the condition $\alpha \vee \beta$ by splitting it at the "||" operator into $orParts = [\alpha, \beta]$. It then iterates through these parts to identify tuples satisfying either disjunct that are closest to P .

When processing the first component, α , the procedure identifies $T_{P, \alpha}$ as the set of closest models to P for α , with the minimum distance being $\Delta_\alpha = \delta_{P, \alpha}$. The internal state variable *globalMinDistance* within `findTuplesWithConstraints` is set to Δ_α , and *globalAssignments* (the collection of resulting tuples) is initialised with $T_{P, \alpha}$. Since $X \in T_{P, \alpha}$, X is included in *globalAssignments*.

Subsequently, when processing the second component, β , the procedure identifies $T_{P, \beta}$ as the set of closest models to P for β , with the minimum distance being $\Delta_\beta = \delta_{P, \beta}$. Given that $\delta_{P, \alpha} = \delta_{P, \beta} = d_M(P, X)$, it holds that $\Delta_\beta = d_M(P, X)$. The current *globalMinDistance* (established from processing α) is $\delta_{P, \alpha}$. Since $\Delta_\beta = d_M(P, X) = \delta_{P, \alpha}$, the condition $\Delta_\beta = \text{globalMinDistance}$ is met.

Consequently, the update rule for *globalAssignments* dictates $\text{globalAssignments} \leftarrow \text{globalAssignments} \cup T_{P, \beta}$ (whilst ensuring uniqueness). Because X was already present in *globalAssignments*—due to its inclusion in $T_{P, \alpha}$ —and $X \in T_{P, \beta}$, X is guaranteed to remain in the result set *globalAssignments*.

The function `findTuplesWithConstraints` ultimately returns *globalAssignments* as the set of tuples for $P \odot (\alpha \vee \beta)$. Since $X \in \text{globalAssignments}$, it follows that $X \in P \odot (\alpha \vee \beta)$. As X was an arbitrary element of $P \odot \alpha \cap P \odot \beta$, it is concluded that

$$P \odot \alpha \cap P \odot \beta \subseteq P \odot (\alpha \vee \beta),$$

thereby satisfying the postulate.

$$\textbf{(S10)} \quad P \odot \alpha \cap \|\beta\| \subseteq P \odot (\alpha \wedge \beta)$$

This postulate requires that any profile which is both a minimally distant model of α (according to $P \odot \alpha$) and simultaneously a model of β , must also be identified among the minimally distant models of the conjunction $\alpha \wedge \beta$ (according to $P \odot (\alpha \wedge \beta)$).

Fulfilment:

Let $T_{P,\phi}$ denote the set of tuples and $\delta_{P,\phi}$ the minimum Manhattan distance for this operation, such that $T_{P,\phi} = \{\tau \mid \tau \models \phi \text{ and } d_M(P, \tau) = \delta_{P,\phi}\}$. Thus, $P \odot \phi = T_{P,\phi}$.

Assume $X \in (P \odot \alpha \cap \|\beta\|)$. This implies two conditions: (1) $X \in T_{P,\alpha}$, and (2) $X \models \beta$. From (1), $X \models \alpha$ and its distance from P is $d_M(P, X) = \delta_{P,\alpha}$ (the minimum distance from P to any model of α). From (2), $X \models \beta$. Combining these, it follows that X satisfies $\alpha \wedge \beta$.

Now, consider the operation $P \odot (\alpha \wedge \beta)$, which yields the set $T_{P,\alpha \wedge \beta}$ at a minimum distance $\delta_{P,\alpha \wedge \beta} = \min_{\tau' \models (\alpha \wedge \beta)} d_M(P, \tau')$. Since X is a model of $\alpha \wedge \beta$ and its distance from P is $d_M(P, X) = \delta_{P,\alpha}$, it must be the case that $\delta_{P,\alpha \wedge \beta} \leq d_M(P, X) = \delta_{P,\alpha}$.

Conversely, let τ' be any tuple in $T_{P,\alpha \wedge \beta}$. By definition, $\tau' \models (\alpha \wedge \beta)$ and $d_M(P, \tau') = \delta_{P,\alpha \wedge \beta}$. Since $\tau' \models (\alpha \wedge \beta)$ implies $\tau' \models \alpha$, and $\delta_{P,\alpha}$ is the minimum distance from P to any model of α , it must hold that $\delta_{P,\alpha} \leq d_M(P, \tau')$. As $d_M(P, \tau') = \delta_{P,\alpha \wedge \beta}$, it follows that $\delta_{P,\alpha} \leq \delta_{P,\alpha \wedge \beta}$.

The two inequalities, $\delta_{P,\alpha \wedge \beta} \leq \delta_{P,\alpha}$ and $\delta_{P,\alpha \wedge \beta} \geq \delta_{P,\alpha}$, together imply that $\delta_{P,\alpha \wedge \beta} = \delta_{P,\alpha}$. Let this common minimum distance be δ^* . It has been established that $X \models (\alpha \wedge \beta)$ and its distance from P is $d_M(P, X) = \delta_{P,\alpha} = \delta^*$. This signifies that X is a model of $\alpha \wedge \beta$ and its distance from P is the minimum possible (δ^*) for models of $\alpha \wedge \beta$. Therefore, by the definition of $T_{P,\alpha \wedge \beta}$, it holds that $X \in T_{P,\alpha \wedge \beta}$.

As X was an arbitrary element of $P \odot \alpha \cap \|\beta\|$, it follows that $P \odot \alpha \cap \|\beta\| \subseteq P \odot (\alpha \wedge \beta)$, thereby satisfying the postulate.

6.1.3 Model 3 Postulates

The many-to-many revision operation $\Gamma \odot \alpha$ is realised by the `model3`($\Gamma, \alpha, \text{minValue}, \text{maxValue}$) function. The operation of this function proceeds in two stages. Firstly, the initial set of profiles Γ is examined to determine whether any of its constituent profiles $P_i \in \Gamma$ already satisfy the formula α . If such profiles exist, the result of the revision $\Gamma \odot \alpha$ is precisely this subset $\{P_i \in \Gamma \mid P_i \models \alpha\}$, with an associated minimum distance of 0.

If, however, no profile within Γ initially satisfies α , `model3` then invokes the procedure:

`findTuplesWithConstraints`($\Gamma, \alpha, \text{minValue}, \text{maxValue}$)

This procedure, when provided with a set of profiles Γ , first calculates an average profile, P_{avg} , from the set Γ .

It then identifies the set of profiles $T_{P_{\text{avg}},\alpha}$, which satisfy α and are at the minimum Manhattan distance, $\delta_{P_{\text{avg}},\alpha}$, from P_{avg} . In this scenario, the set $T_{P_{\text{avg}},\alpha}$ constitutes the result $\Gamma \odot \alpha$.

(SP1) If $\|\alpha\| \neq \emptyset$, then $\Gamma \odot \alpha \subseteq \|\alpha\|$.

If α is satisfiable, every profile in the resultant set $\Gamma \odot \alpha$ must satisfy α .

Fulfilment:

Initially, `model3` constructs an executable predicate, $\text{checkFn}(\tau)$, which evaluates to true if and only if $\tau \models \alpha$. This predicate is applied to each profile $\tau \in \Gamma$. If the set of profiles within Γ that satisfy α is non-empty, `model3` immediately returns this set as the result of the revision, with an associated *minDistance* of 0. Since all profiles in this returned set are elements of Γ that also satisfy α , this outcome is manifestly a subset of $\|\alpha\|$.

Alternatively, if no profile $\tau \in \Gamma$ initially satisfies α , the `model3` function proceeds by invoking the procedure `findTuplesWithConstraints`($\Gamma, \alpha, \text{minValue}, \text{maxValue}$). The latter procedure first computes an average profile, P_{avg} , from the input set Γ to serve as a reference point.

The procedure then conducts a search for profiles satisfying α that are minimally distant from P_{avg} . A core component of this search is the consistent application of the aforementioned $\text{checkFn}(\tau)$. Candidate profiles τ are only considered for inclusion in the final set of results if they satisfy this predicate, meaning $\tau \models \alpha$. Consequently, this set is also guaranteed to be a subset of $\|\alpha\|$. In both operational branches, every profile returned as part of $\Gamma \odot \alpha$ satisfies α , thereby ensuring $\Gamma \odot \alpha \subseteq \|\alpha\|$.

(SP2) If $\|\alpha\| = \emptyset$, then $\Gamma \odot \alpha = \Gamma$.

If α is unsatisfiable, the revision should return the original set of profiles Γ .

Fulfilment:

Within `findTuplesWithConstraints`, if the formula α is unsatisfiable, the core search mechanism will fail to identify any satisfying tuples.

Consequently, an internal state variable, *globalMinDistance*—which tracks the minimum distance to a satisfying tuple—will retain its initial value of ∞ . Upon completion of the search phase

without identifying any models of α , `findTuplesWithConstraints` executes a specific fallback conditional logic that returns the original set Γ . The `model3` function then directly propagates this output. Therefore, under the condition $\|\alpha\| = \emptyset$, the revision yields $\Gamma \odot \alpha = \Gamma$, thereby satisfying the postulate.

(SP3) $\Gamma \odot \alpha \neq \emptyset$.

This postulate requires that for every wff α , the revision of a non-empty profile set Γ yields a non-empty result.

Fulfilment:

The result follows directly from the operational logic established in **SP1** and **SP2**.

(SP4) If $\|\alpha\| = \|\beta\|$, then $\Gamma \odot \alpha = \Gamma \odot \beta$.

Revising by two logically equivalent wff must yield the same set of profiles.

Fulfilment: Initially, `model3` determines whether any profiles in Γ directly satisfy the input formula ϕ . This involves verifying if the intersection $\Gamma \cap \|\phi\|$ is non-empty. Given the premise $\|\alpha\| = \|\beta\|$, it follows that $\Gamma \cap \|\alpha\| = \Gamma \cap \|\beta\|$. If this common intersection is non-empty, `model3` returns this set as the result for both $\Gamma \odot \alpha$ and $\Gamma \odot \beta$. Consequently, in this scenario, $\Gamma \odot \alpha = \Gamma \odot \beta$ as they both equal $\Gamma \cap \|\alpha\|$ (or $\Gamma \cap \|\beta\|$).

Alternatively, if $\Gamma \cap \|\alpha\| = \emptyset$, the procedure first computes an average profile, P_{avg} , to serve as the reference point for distance calculations. The subsequent operations within `findTuplesWithConstraints`, parsing the formula ϕ into an AST, performing domain narrowing based upon the AST-derived constraints, and conducting a distance, guided search for tuples τ that satisfy an executable predicate $\text{checkFn}(\tau) \equiv (\tau \models \phi)$ whilst minimising distance from P_{avg} —are fundamentally driven by semantic content.

Since $\|\alpha\| = \|\beta\|$, the semantic information guiding these steps is identical for both α and β . Although the ASTs for α and β might differ structurally, their semantic equivalence ensures that they impose equivalent constraints. This leads to identical domain maps after narrowing. Furthermore, the `checkFn` predicates derived for α and β will be extensionally identical. As all subsequent search and distance computations are founded upon these identical semantic interpretations relative to the same P_{avg} , the `findTuplesWithConstraints` procedure explores the same search space and

identifies the same set of minimally distant tuples for both α and β . Let this common set of tuples be T^* and the common minimum distance be δ^* .

In this fallback case, $\Gamma \odot \alpha = T^*$ and $\Gamma \odot \beta = T^*$. Thus, in both operational pathways of `model3`, it is established that if $\|\alpha\| = \|\beta\|$, then $\Gamma \odot \alpha = \Gamma \odot \beta$, thereby satisfying the postulate.

(SP5) If $\Gamma \cap \|\alpha\| \neq \emptyset$, then $\Gamma \odot \alpha = \Gamma \cap \|\alpha\|$.

This postulate requires that whenever the initial set Γ already contains profiles satisfying α , the revision returns exactly that subset.

Fulfilment:

The computation of $\Gamma \odot \alpha$ by the `model3` function begins with an initial verification corresponding to this postulate. Firstly, a predicate $checkFn(\tau)$, representing the evaluation of $\tau \models \alpha$, is constructed. This predicate is then applied to each profile $\tau \in \Gamma$. Within the implementation, an empty collection, *solution*, is initialised.

Assuming Γ is a set of profiles, the algorithm iterates through each $\tau \in \Gamma$; if $checkFn(\tau)$ evaluates to true, τ is added to the *solution* collection. If, upon examining all profiles in Γ , the *solution* collection is non-empty—which corresponds to the premise $\Gamma \cap \|\alpha\| \neq \emptyset$ —the function immediately returns that set without proceeding to further search or distance-minimisation steps. In this instance, *solution* is precisely the set $\{\tau \in \Gamma \mid \tau \models \alpha\}$, which is extensionally equivalent to $\Gamma \cap \|\alpha\|$. Therefore, the revision operator yields $\Gamma \odot \alpha = \Gamma \cap \|\alpha\|$, thereby satisfying the postulate.

(SP6) $\Gamma \odot (\alpha \vee \beta) = \Gamma \odot \alpha$ or $\Gamma \odot (\alpha \vee \beta) = \Gamma \odot \beta$ or $\Gamma \odot (\alpha \vee \beta) = \Gamma \odot \alpha \cup \Gamma \odot \beta$.

When revising by a disjunction, the result must coincide with revising by one disjunct, the other, or their union.

Fulfilment:

Firstly, `model3` considers the case of direct satisfaction, where profiles within the initial set Γ already satisfy the disjunction $\alpha \vee \beta$. An executable predicate corresponding to $\tau \models (\alpha \vee \beta)$ is evaluated for each $\tau \in \Gamma$. If the set $\Gamma \cap \|\alpha \vee \beta\|$ is found to be non-empty, `model3` returns this set as $\Gamma \odot (\alpha \vee \beta)$. This returned set, $\Gamma \cap \|\alpha \vee \beta\|$, can be equivalently expressed as $(\Gamma \cap \|\alpha\|) \cup (\Gamma \cap \|\beta\|)$.

Depending on whether $\Gamma \cap \|\alpha\|$ or $\Gamma \cap \|\beta\|$ (or both) are non-empty, and how $\Gamma \odot \alpha$ and $\Gamma \odot \beta$ would be formed under **SP5**, this result aligns with one of the three forms stipulated by **SP6**. For example, if $\Gamma \cap \|\alpha\| \neq \emptyset$ and $\Gamma \cap \|\beta\| = \emptyset$, the result is $\Gamma \cap \|\alpha\|$, which corresponds to $\Gamma \odot \alpha$.

Secondly, if no profile $\tau \in \Gamma$ satisfies $\alpha \vee \beta$ directly, **model3** invokes **findTuplesWithConstraints**($\Gamma, \alpha \vee \beta, \dots$). This procedure then calculates an average profile $P_{\text{avg}} = \text{calculateTupleAverages}(\Gamma)$ to serve as the reference for distance computations.

The disjunctive condition $\alpha \vee \beta$ is split into its constituent parts, namely $\{\alpha, \beta\}$. For each such disjunct γ , **findTuplesWithConstraints**—via its internal **_findTuplesForConjunction** mechanism—determines the set T_γ of all profiles τ such that $\tau \models \gamma$ and $d_M(P_{\text{avg}}, \tau)$ is minimised at Δ_γ . The overall minimum distance $\Delta = \min(\Delta_\alpha, \Delta_\beta)$ across these disjuncts is then identified.

The final set of returned profiles, *globalAssignments*, is constructed as follows:

- If $\Delta_\alpha < \Delta_\beta$, then *globalAssignments* = T_α (corresponding to $\Gamma \odot \alpha$ computed via P_{avg}).
- If $\Delta_\beta < \Delta_\alpha$, then *globalAssignments* = T_β (corresponding to $\Gamma \odot \beta$ via P_{avg}).
- If $\Delta_\alpha = \Delta_\beta$, then *globalAssignments* = $T_\alpha \cup T_\beta$ (corresponding to $(\Gamma \odot \alpha) \cup (\Gamma \odot \beta)$ via P_{avg}).

In both operational branches, the **model3** function returns a set of profiles that is equivalent to $\Gamma \odot \alpha$, $\Gamma \odot \beta$, or their union, thereby satisfying Postulate **SP6**.

(SP7) $\Gamma \odot (\alpha \vee \beta) \subseteq \Gamma \odot \alpha \cup \Gamma \odot \beta$.

When revising by a disjunction, every profile returned must also appear in the union of the revisions by the individual disjuncts.

Fulfilment:

Initially, the **model3** function assesses whether any profile $\tau \in \Gamma$ directly satisfies the disjunction $\alpha \vee \beta$. This is achieved by constructing an executable predicate *checkFn*(τ) for $\tau \models (\alpha \vee \beta)$ and applying it to all profiles in Γ . If the resulting set of satisfying profiles, *solution* = $\{\tau \in \Gamma \mid \text{checkFn}(\tau)\}$, is non-empty, **model3** returns this set *solution* as $\Gamma \odot (\alpha \vee \beta)$. Any profile $X \in \text{solution}$ satisfies $\alpha \vee \beta$, signifying that $X \models \alpha$ or $X \models \beta$.

If $X \models \alpha$ and $X \in \Gamma$, then by the operational definition of **model3**, $X \in \Gamma \odot \alpha$. Similarly, if $X \models \beta$ and $X \in \Gamma$, then $X \in \Gamma \odot \beta$. Therefore, any such X from *solution* is necessarily contained within $\Gamma \odot \alpha \cup \Gamma \odot \beta$. Consequently, in this instance, $\Gamma \odot (\alpha \vee \beta) \subseteq \Gamma \odot \alpha \cup \Gamma \odot \beta$.

Alternatively, if no profile $\tau \in \Gamma$ initially satisfies $\alpha \vee \beta$, **model3** invokes **findTuplesWithConstraints**($\Gamma, \alpha \vee \beta, \dots$). This procedure then calculates an average profile P_{avg}

from Γ and processes the disjunction $\alpha \vee \beta$ by splitting it into its constituent parts, namely $[\alpha, \beta]$. For each disjunct $\gamma \in \{\alpha, \beta\}$, an internal mechanism determines $\Delta_\gamma = \min_{\tau \models \gamma} d_M(P_{\text{avg}}, \tau)$ and the set $T_\gamma = \{\tau \mid \tau \models \gamma \text{ and } d_M(P_{\text{avg}}, \tau) = \Delta_\gamma\}$ of profiles satisfying γ at this minimal distance from P_{avg} .

The `findTuplesWithConstraints` function then constructs its final output, *globalAssignments*, based upon these values: *globalAssignments* = T_α if $\Delta_\alpha < \Delta_\beta$; *globalAssignments* = T_β if $\Delta_\beta < \Delta_\alpha$; or *globalAssignments* = $T_\alpha \cup T_\beta$ if $\Delta_\alpha = \Delta_\beta$. In this fallback scenario, T_α represents the outcome of $\Gamma \odot \alpha$ (computed via P_{avg} because $\Gamma \cap \|\alpha\| = \emptyset$), and T_β represents $\Gamma \odot \beta$ (similarly via P_{avg}). Thus, the returned set $\Gamma \odot (\alpha \vee \beta) = \text{globalAssignments}$ is, by its construction, equivalent to $\Gamma \odot \alpha$, $\Gamma \odot \beta$, or $(\Gamma \odot \alpha) \cup (\Gamma \odot \beta)$, each of which is inherently a subset of $(\Gamma \odot \alpha) \cup (\Gamma \odot \beta)$.

In both distinct operational pathways, the implementation ensures that $\Gamma \odot (\alpha \vee \beta) \subseteq \Gamma \odot \alpha \cup \Gamma \odot \beta$, thereby satisfying Postulate **SP7**.

(SP8) If $\Gamma \odot \alpha \models \beta$ and $\Gamma \odot \beta \models \alpha$, then $\Gamma \odot \alpha = \Gamma \odot \beta$.

This postulate states that if revising a profile set Γ by a formula α implies β , and revising Γ by β implies α , the resulting profile sets from both revisions must be identical.

Fulfilment:

Let the revision operation $\Gamma \odot \gamma$ be implemented by the `model3`($\Gamma, \gamma, \text{minValue}, \text{maxValue}$) function. This function produces a set of tuples, denoted here as T_γ , which are at a minimum Manhattan distance δ_γ from a reference point—either a profile in Γ or an average profile P_{avg} . By construction of the underlying `findTuplesWithConstraints` procedure, $T_\alpha \subseteq \|\alpha\|$ and $T_\beta \subseteq \|\beta\|$. The premises of the postulate, $\Gamma \odot \alpha \models \beta$ and $\Gamma \odot \beta \models \alpha$, signify that all tuples in T_α also satisfy β , and all tuples in T_β also satisfy α . Consequently, it follows that both T_α and T_β are subsets of $\|\alpha \wedge \beta\|$.

The `model3` function operates via two principal pathways. Firstly, an initial satisfaction test is performed. If $\Gamma \cap \|\alpha\| \neq \emptyset$, then $\Gamma \odot \alpha$ is defined as $\Gamma \cap \|\alpha\|$. Given the premise $T_\alpha = (\Gamma \cap \|\alpha\|) \subseteq \|\beta\|$, it is implied that $T_\alpha = \Gamma \cap \|\alpha \wedge \beta\|$. By a symmetrical argument for β , if $\Gamma \cap \|\beta\| \neq \emptyset$, then $\Gamma \odot \beta$ is defined as $\Gamma \cap \|\beta\|$. Given $T_\beta \subseteq \|\alpha\|$, this implies $T_\beta = \Gamma \cap \|\alpha \wedge \beta\|$. Thus, if this initial satisfaction branch is applicable to both revisions under the postulate's premises—which occurs if $\Gamma \cap \|\alpha \wedge \beta\| \neq \emptyset$ —then $T_\alpha = T_\beta = \Gamma \cap \|\alpha \wedge \beta\|$, ensuring $\Gamma \odot \alpha = \Gamma \odot \beta$.

Secondly, if no profile $\tau \in \Gamma$ initially satisfies the respective condition (α for the first revision, or β for the second), **model3** invokes **findTuplesWithConstraints**. This procedure computes an average profile P_{avg} from Γ and subsequently conducts an AST-driven domain narrowing and a recursive search (**explore**) for tuples satisfying the given formula, whilst minimising distance from P_{avg} . For each $\gamma \in \{\alpha, \beta\}$, this search identifies $\delta_\gamma = \min_{\tau \models \gamma} d_M(P_{\text{avg}}, \tau)$ and $T_\gamma = \{\tau \mid \tau \models \gamma \text{ and } d_M(P_{\text{avg}}, \tau) = \delta_\gamma\}$.

The premise $T_\alpha \subseteq \|\beta\|$ signifies that all tuples in T_α (the closest models of α to P_{avg}) also satisfy β , and thus satisfy $\alpha \wedge \beta$. This implies that δ_α is the minimum distance from P_{avg} to models of $\alpha \wedge \beta$ when originating from the α -revision context. Symmetrically, $\delta_\beta = \delta(\alpha \wedge \beta)$. Therefore, $\delta_\alpha = \delta_\beta$. Since the underlying search mechanism of **findTuplesWithConstraints** is deterministic and, due to the premises, both revisions effectively seek the closest models of $\alpha \wedge \beta$ to P_{avg} at this common minimum distance, the resulting sets T_α and T_β must be identical.

In both operational branches of the **model3** function, the conditions imposed by the postulate lead to $\Gamma \odot \alpha = T_\alpha = T_\beta = \Gamma \odot \beta$, thereby fulfilling the postulate.

(SP9) $\Gamma \odot \alpha \cap \Gamma \odot \beta \subseteq \Gamma \odot (\alpha \vee \beta)$.

This postulate asserts that any profile common to both the revision of Γ by α and the revision of Γ by β must also be present in the set of profiles resulting from revising Γ by the disjunction $\alpha \vee \beta$.

Fulfilment:

The satisfaction of this postulate by the **model3** function follows directly from the properties established by Postulate **SP6**. According to **SP6**, the result of the revision $\Gamma \odot (\alpha \vee \beta)$ is equivalent to one of three specific forms: $\Gamma \odot \alpha$, $\Gamma \odot \beta$, or the union $\Gamma \odot \alpha \cup \Gamma \odot \beta$.

Demonstrating that $\Gamma \odot \alpha \cap \Gamma \odot \beta \subseteq \Gamma \odot (\alpha \vee \beta)$ requires consideration of these three cases:

- If $\Gamma \odot (\alpha \vee \beta) = \Gamma \odot \alpha$, the required inclusion becomes $\Gamma \odot \alpha \cap \Gamma \odot \beta \subseteq \Gamma \odot \alpha$, which is a fundamental property of set intersection.
- If $\Gamma \odot (\alpha \vee \beta) = \Gamma \odot \beta$, the required inclusion becomes $\Gamma \odot \alpha \cap \Gamma \odot \beta \subseteq \Gamma \odot \beta$, which is likewise a fundamental property of set intersection.

– Finally, if $\Gamma \odot (\alpha \vee \beta) = \Gamma \odot \alpha \cup \Gamma \odot \beta$, the required inclusion becomes $\Gamma \odot \alpha \cap \Gamma \odot \beta \subseteq \Gamma \odot \alpha \cup \Gamma \odot \beta$.

This is also a fundamental property of set theory, as the intersection of two sets is inherently a subset of their union.

Since in all possible outcomes for $\Gamma \odot (\alpha \vee \beta)$ as dictated by **SP6**, the set $\Gamma \odot \alpha \cap \Gamma \odot \beta$ is indeed a subset of $\Gamma \odot (\alpha \vee \beta)$, Postulate **SP9** is satisfied.

(SP10) $\Gamma \odot \alpha \cap \|\beta\| \subseteq \Gamma \odot (\alpha \wedge \beta)$.

This postulate requires that any profile which is both a result of revising Γ by α (that is, an element of $\Gamma \odot \alpha$) and also a model of β , must appear among the results of revising Γ by the conjunction $\alpha \wedge \beta$.

Fulfilment:

Let $\Gamma \odot \phi$ denote the outcome of the `model3`($\Gamma, \phi, \text{minValue}, \text{maxValue}$) function. Assume $X \in (\Gamma \odot \alpha \cap \|\beta\|)$. This implies $X \in \Gamma \odot \alpha$ and $X \models \beta$. Since $X \in \Gamma \odot \alpha$, it follows that $X \models \alpha$. Consequently, $X \models (\alpha \wedge \beta)$.

The `model3` function determines $\Gamma \odot \phi$ via one of two pathways. Firstly, if $\Gamma \cap \|\phi\| \neq \emptyset$, then $\Gamma \odot \phi = \Gamma \cap \|\phi\|$ (by Postulate **SP5**). Secondly, if $\Gamma \cap \|\phi\| = \emptyset$, `model3` invokes `findTuplesWithConstraints`($\Gamma, \phi, \dots$). This procedure computes an average profile P_{avg} from Γ and subsequently identifies $T_{P_{\text{avg}}, \phi}$, the set of profiles satisfying ϕ at a minimum Manhattan distance $\delta_{P_{\text{avg}}, \phi}$ from P_{avg} . In this instance, $\Gamma \odot \phi = T_{P_{\text{avg}}, \phi}$.

Consider the profile X . Since $X \in \Gamma \odot \alpha$: If $\Gamma \odot \alpha = \Gamma \cap \|\alpha\|$ (because $\Gamma \cap \|\alpha\| \neq \emptyset$), then $X \in \Gamma$ and $X \models \alpha$. Given that $X \models \beta$ by the initial assumption, it follows that $X \in (\Gamma \cap \|\alpha \wedge \beta\|)$. When computing $\Gamma \odot (\alpha \wedge \beta)$, the `model3` function first verifies if $\Gamma \cap \|\alpha \wedge \beta\|$ is non-empty. Since X is a member of this set, this condition is satisfied; thus, $\Gamma \odot (\alpha \wedge \beta) = \Gamma \cap \|\alpha \wedge \beta\|$, which necessarily includes X .

Alternatively, if $\Gamma \odot \alpha = T_{P_{\text{avg}}, \alpha}$ (because $\Gamma \cap \|\alpha\| = \emptyset$), then $X \models \alpha$ and $d_M(P_{\text{avg}}, X) = \delta_{P_{\text{avg}}, \alpha}$. As $X \models \beta$, X is a model of $\alpha \wedge \beta$ at distance $\delta_{P_{\text{avg}}, \alpha}$ from P_{avg} .

Now, consider the computation of $\Gamma \odot (\alpha \wedge \beta)$. If $\Gamma \cap \|\alpha \wedge \beta\| \neq \emptyset$ (and $X \in \Gamma$), then, as demonstrated above, $X \in \Gamma \odot (\alpha \wedge \beta)$. If $\Gamma \cap \|\alpha \wedge \beta\| = \emptyset$, then $\Gamma \odot (\alpha \wedge \beta) = T_{P_{\text{avg}}, \alpha \wedge \beta}$ with distance $\delta_{P_{\text{avg}}, \alpha \wedge \beta}$. Since X is a model of $\alpha \wedge \beta$, it must be that $\delta_{P_{\text{avg}}, \alpha \wedge \beta} \leq d_M(P_{\text{avg}}, X) = \delta_{P_{\text{avg}}, \alpha}$.

If $\delta_{P_{\text{avg}}, \alpha \wedge \beta} < \delta_{P_{\text{avg}}, \alpha}$, it would imply the existence of a $\tau' \in \|\alpha \wedge \beta\|$ (and thus $\tau' \in \|\alpha\|$) such that $d_M(P_{\text{avg}}, \tau') < \delta_{P_{\text{avg}}, \alpha}$. This contradicts the definition of $\delta_{P_{\text{avg}}, \alpha}$ as the minimal distance from P_{avg} to models of α .

Therefore, it must be the case that $\delta_{P_{\text{avg}}, \alpha \wedge \beta} = \delta_{P_{\text{avg}}, \alpha}$. The procedure `findTuplesWithConstraints` for $\alpha \wedge \beta$ (operating via P_{avg}) deterministically identifies all models of $\alpha \wedge \beta$ at this established minimal distance. Since X is such a model—satisfying $\alpha \wedge \beta$ at distance $\delta_{P_{\text{avg}}, \alpha}$ from P_{avg} —it follows that $X \in T_{P_{\text{avg}}, \alpha \wedge \beta}$, and thus $X \in \Gamma \odot (\alpha \wedge \beta)$.

In all valid operational pathways, if $X \in \Gamma \odot \alpha \cap \|\beta\|$, then $X \in \Gamma \odot (\alpha \wedge \beta)$. As X was an arbitrary element, it is concluded that

$$\Gamma \odot \alpha \cap \|\beta\| \subseteq \Gamma \odot (\alpha \wedge \beta),$$

thereby satisfying Postulate **SP10**.

6.1.4 Model 4 Postulates

The update operation $\Gamma \diamond \alpha$ is realised by the `model14`($\Gamma, \alpha, \text{minValue}, \text{maxValue}$) function. The first step in this function involves iterating through each profile $P_i \in \Gamma$ and invoking the procedure $(T_{P_i}, \delta_{P_i}) = \text{findTuplesWithConstraints}(P_i, \alpha, \text{minValue}, \text{maxValue})$ for each such P_i , where T_{P_i} is the set of profiles that satisfy α and are at the minimum Manhattan distance δ_{P_i} from P_i . Finally, `model14` performs a set union of all T_{P_i} , resulting in $\Gamma \diamond \alpha = \bigcup_{P \in \Gamma} T_{P, \alpha}$.

(U0) If $\|\alpha\| = \emptyset$, then $\Gamma \diamond \alpha = \Gamma$.

This postulate asserts that if the wff α is unsatisfiable, updating a set of profiles Γ with α should result in the original set Γ , indicating no change.

Fulfilment:

Assume $\|\alpha\| = \emptyset$.

When α is unsatisfiable, the `findTuplesWithConstraints` procedure—when called with any $P_i \in \Gamma$ as its target—will fail to identify any tuple τ such that $\tau \models \alpha$. This occurs either because α is trivially unsatisfiable (for instance, syntactically invalid or a literal false) or because the domain-narrowing process results in an empty search space.

In such cases, an internal state variable, *globalMinDistance*, remains at its initial value of ∞ . Consequently, a fallback branch is executed and P_i is returned. The `model14` function then constructs its final output by taking the union of all returned tuples. In this scenario, this union precisely reconstructs the original set of profiles Γ . Thus, if $\|\alpha\| = \emptyset$, then $\Gamma \diamond \alpha = \Gamma$, thereby satisfying the postulate.

(U1) If $\|\alpha\| \neq \emptyset$, then $\Gamma \diamond \alpha \subseteq \|\alpha\|$.

This postulate states that if α is satisfiable, the result of updating a set of profiles Γ with α must be a subset of $\|\alpha\|$.

Fulfilment:

The `findTuplesWithConstraints` function parses α into an AST, from which an executable Boolean predicate, *checkFn*(τ), is constructed. This predicate evaluates to `true` if and only if $\tau \models \alpha$. Only those tuples that satisfy this predicate are included in the resulting set T_{P_i} ; thus, $T_{P_i} \subseteq \|\alpha\|$ for every $P_i \in \Gamma$.

The `model14` function subsequently aggregates all sets T_{P_i} and returns their union as the final result. Since every constituent set in the union is a subset of $\|\alpha\|$, it follows that the union itself satisfies the inclusion:

$$\Gamma \diamond \alpha = \bigcup_{P_i \in \Gamma} T_{P_i} \subseteq \|\alpha\|$$

Consequently, the implementation satisfies the postulate.

(U2) If $\Gamma \subseteq \|\alpha\|$, then $\Gamma \diamond \alpha = \Gamma$.

This postulate states that if the set of profiles Γ is already a subset of the models of α , then updating Γ with α should result in no change to Γ .

Fulfilment:

The premise of the postulate is $\Gamma \subseteq \|\alpha\|$. This implies that every individual profile $P_i \in \Gamma$ is a model of α .

When `findTuplesWithConstraints` is invoked for each input profile P_i , its internal logic evaluates *checkFn*(P_i). Since $P_i \models \alpha$, this evaluation returns true. P_i is recognised as a valid solution with a Manhattan distance of $d_M(P_i, P_i) = 0$.

As no other tuple can achieve a distance less than zero, the procedure immediately returns the singleton set $\{P_i\}$. Consequently, the `model4` function aggregates all returned tuples. The union of these singleton sets is precisely the original set Γ . Thus, if $\Gamma \subseteq \|\alpha\|$, the `model4` function ensures that $\Gamma \diamond \alpha = \Gamma$, thereby satisfying the postulate.

(U3) $\Gamma \diamond \alpha \neq \emptyset$.

This postulate mandates that the update of a profile set Γ by any wff α must always produce a non-empty result.

Fulfilment:

Two cases are considered:

- If α is satisfiable, then according to **U1**, $\Gamma \diamond \alpha$ will consist of at least one model of α for each $P_i \in \Gamma$, assuming Γ itself is non-empty.
- Alternatively, if α is unsatisfiable, then according to **U0**, the operation $\Gamma \diamond \alpha$ results in the original set Γ .

In both scenarios, `model4` produces a non-empty set of profiles, thereby satisfying the postulate.

(U4) If $\|\alpha\| = \|\beta\|$, then $\Gamma \diamond \alpha = \Gamma \diamond \beta$.

This postulate stipulates that if α and β are logically equivalent, updating a set of profiles Γ with α must yield the same result as updating Γ with β .

Fulfilment:

The premise of the postulate is $\|\alpha\| = \|\beta\|$. This semantic equivalence implies that for any tuple τ , $\tau \models \alpha$ if and only if $\tau \models \beta$. The `findTuplesWithConstraints` procedure is fundamentally driven by the semantic properties of the input wff ϕ .

Its operations include: parsing ϕ into an AST which reflects its logical structure; performing domain narrowing for tuple attributes based upon constraints derived from the AST; and conducting a distance-guided search that identifies tuples τ satisfying an executable predicate $checkFn(\tau) \equiv (\tau \models \phi)$, whilst minimising their Manhattan distance from the specific P_i .

Given that $\|\alpha\| = \|\beta\|$, the *checkFn* generated for α will be extensionally identical to that for β . Although the ASTs for α and β might differ structurally, their semantic equivalence ensures that the domain-narrowing phase results in identical domain maps for each element of any given P_i .

Consequently, the subsequent search process—guided by these identical domains, the same effective *checkFn*, and identical distances relative to P_i —traverses the same search space and identifies the same set of minimally distant tuples.

Therefore, for each $P_i \in \Gamma$, the call to `findTuplesWithConstraints`($P_i, \alpha, \dots$) produces the same set of tuples $T_{P_i, \alpha}$ and the same minimum distance $\delta_{P_i, \alpha}$ as the call to `findTuplesWithConstraints`($P_i, \beta, \dots$); that is, $T_{P_i, \alpha} = T_{P_i, \beta}$ and $\delta_{P_i, \alpha} = \delta_{P_i, \beta}$.

Since these individual results are identical for α and β for every $P_i \in \Gamma$, the aggregation mechanism within `model14` inevitably produces an identical final outcome for both $\Gamma \diamond \alpha$ and $\Gamma \diamond \beta$. Thus, the postulate is satisfied.

(U5) $\Gamma \diamond \alpha \cap \|\beta\| \subseteq \Gamma \diamond (\alpha \wedge \beta)$.

This postulate requires that any profile resulting from the update of Γ by α which also happens to be a model of β , must also be a result of updating Γ by the conjunction $\alpha \wedge \beta$.

Fulfilment:

Let $\Gamma \diamond \phi$ denote the `model14`($\Gamma, \phi, \text{minValue}, \text{maxValue}$) function. The result of $\Gamma \diamond \phi$ is the union $\bigcup_{P \in \Gamma} T_{P, \phi}$.

Assume $X \in (\Gamma \diamond \alpha \cap \|\beta\|)$. This implies:

1. $X \in \Gamma \diamond \alpha$
2. $X \models \beta$

From (1), since $X \in \bigcup_{P \in \Gamma} T_{P, \alpha}$, there must exist at least one profile $P_k \in \Gamma$ such that $X \in T_{P_k, \alpha}$. Its distance $d_M(P_k, X)$ is equal to $\delta_{P_k, \alpha}$, the minimum distance from P_k to any model of α . From (2), $X \models \beta$. Combining these, it follows that X satisfies $\alpha \wedge \beta$.

Now, consider the computation of $\Gamma \diamond (\alpha \wedge \beta)$, which is $\bigcup_{P \in \Gamma} T_{P, \alpha \wedge \beta}$. To demonstrate that X is an element of this union, it suffices to show that $X \in T_{P_k, \alpha \wedge \beta}$ for the same P_k . The set $T_{P_k, \alpha \wedge \beta}$ consists of all tuples τ such that $\tau \models (\alpha \wedge \beta)$ and $d_M(P_k, \tau) = \delta_{P_k, \alpha \wedge \beta}$, where $\delta_{P_k, \alpha \wedge \beta} = \min_{\tau' \models (\alpha \wedge \beta)} d_M(P_k, \tau')$.

Since $X \models (\alpha \wedge \beta)$, X is a candidate when determining $\delta_{P_k, \alpha \wedge \beta}$. Thus, $\delta_{P_k, \alpha \wedge \beta} \leq d_M(P_k, X)$. Given that $d_M(P_k, X) = \delta_{P_k, \alpha}$, it holds that $\delta_{P_k, \alpha \wedge \beta} \leq \delta_{P_k, \alpha}$.

Conversely, any tuple $\tau' \in T_{P_k, \alpha \wedge \beta}$ satisfies $\tau' \models \alpha$. By the definition of $\delta_{P_k, \alpha}$ as the minimum distance from P_k to models of α , it follows that $\delta_{P_k, \alpha} \leq d_M(P_k, \tau')$. Since $d_M(P_k, \tau') = \delta_{P_k, \alpha \wedge \beta}$, we have $\delta_{P_k, \alpha} \leq \delta_{P_k, \alpha \wedge \beta}$.

The two inequalities imply that $\delta_{P_k, \alpha \wedge \beta} = \delta_{P_k, \alpha}$. Therefore, $d_M(P_k, X) = \delta_{P_k, \alpha} = \delta_{P_k, \alpha \wedge \beta}$. As $X \models (\alpha \wedge \beta)$ and its distance from P_k is the minimum possible for models of $\alpha \wedge \beta$, it follows that $X \in T_{P_k, \alpha \wedge \beta}$.

Since $X \in T_{P_k, \alpha \wedge \beta}$ for some $P_k \in \Gamma$, it is concluded that $X \in \Gamma \diamond (\alpha \wedge \beta)$. As X was an arbitrary element of $\Gamma \diamond \alpha \cap \|\beta\|$, the postulate is satisfied.

(U6) If $\Gamma \diamond \alpha \models \beta$ and $\Gamma \diamond \beta \models \alpha$, then $\Gamma \diamond \alpha = \Gamma \diamond \beta$.

This postulate states that if the set of profiles resulting from updating Γ by α entirely satisfies β , and symmetrically, the set of profiles from updating Γ by β entirely satisfies α , the two resulting update sets must be identical.

Fulfilment:

Let $\Gamma \diamond \phi$ denote the result of the `model14`($\Gamma, \phi, \text{min Value}, \text{max Value}$) function. This result is the union $\bigcup_{P \in \Gamma} T_{P, \phi}$. Let $\delta_{P, \phi}$ be the minimum distance $d_M(P, \tau)$ for $\tau \in T_{P, \phi}$.

The premises are (1) $\Gamma \diamond \alpha \models \beta$ and (2) $\Gamma \diamond \beta \models \alpha$. Premise (1) implies that for any $P \in \Gamma$, every tuple $\tau \in T_{P, \alpha}$ also satisfies β . Since $\tau \models \alpha$ by the definition of $T_{P, \alpha}$, it follows that every $\tau \in T_{P, \alpha}$ satisfies $\alpha \wedge \beta$. Thus, $T_{P, \alpha} \subseteq \|\alpha \wedge \beta\|$ for all $P \in \Gamma$.

Since $T_{P, \alpha}$ contains models of $\alpha \wedge \beta$ that are at distance $\delta_{P, \alpha}$ from P , the minimum distance from P to models of $\alpha \wedge \beta$ (denoted $\delta_{P, \alpha \wedge \beta}$) must be less than or equal to $\delta_{P, \alpha}$. Conversely, any tuple $\tau' \in T_{P, \alpha \wedge \beta}$ satisfies α . Therefore, $\delta_{P, \alpha}$ must be less than or equal to $\delta_{P, \alpha \wedge \beta}$. These two conditions imply that $\delta_{P, \alpha} = \delta_{P, \alpha \wedge \beta}$. Consequently, $T_{P, \alpha}$ is precisely the set of models of $\alpha \wedge \beta$ that are closest to P ; hence, $T_{P, \alpha} = T_{P, \alpha \wedge \beta}$ for each $P \in \Gamma$.

By a symmetrical argument utilising premise (2), $\Gamma \diamond \beta \models \alpha$, it is concluded that for each $P \in \Gamma$, $T_{P, \beta} \subseteq \|\alpha \wedge \beta\|$. Following similar reasoning, this leads to $\delta_{P, \beta} = \delta_{P, \alpha \wedge \beta}$, and therefore $T_{P, \beta} = T_{P, \alpha \wedge \beta}$ for each $P \in \Gamma$.

Combining these findings, for every $P \in \Gamma$, it holds that $T_{P, \alpha} = T_{P, \alpha \wedge \beta}$ and $T_{P, \beta} = T_{P, \alpha \wedge \beta}$. This implies $T_{P, \alpha} = T_{P, \beta}$ for all $P \in \Gamma$. Since $\Gamma \diamond \alpha = \bigcup_{P \in \Gamma} T_{P, \alpha}$ and $\Gamma \diamond \beta = \bigcup_{P \in \Gamma} T_{P, \beta}$, and

the constituent sets $T_{P,\alpha}$ and $T_{P,\beta}$ are identical for each respective P , their unions must also be identical. Thus, $\Gamma \diamond \alpha = \Gamma \diamond \beta$, thereby satisfying the postulate.

(U7) If P is a profile, then $\{P\} \diamond \alpha \cap \{P\} \diamond \beta \subseteq \{P\} \diamond (\alpha \vee \beta)$.

This postulate asserts that any profile X resulting from updating the singleton set $\{P\}$ by α , which is also a result of updating $\{P\}$ by β , must also be a result of updating $\{P\}$ by the disjunction $\alpha \vee \beta$.

Fulfilment: Let $\Gamma = \{P\}$. Given that Γ is a singleton set containing only P , the `model4` function effectively executes only for the profile P . Consequently, the union of the results $\bigcup_{P' \in \{P\}} T_{P',\phi}$ simplifies to $T_{P,\phi}$, where $(T_{P,\phi}, \delta_{P,\phi}) = \text{findTuplesWithConstraints}(P, \phi, \text{minValue}, \text{maxValue})$. Thus, for a single profile P , $\{P\} \diamond \gamma$ is equivalent to the set of tuples $T_{P,\gamma}$ identified by `findTuplesWithConstraints` at the minimum Manhattan distance $\delta_{P,\gamma} = \min_{\tau \models \gamma} d_M(P, \tau)$.

Assume $X \in (\{P\} \diamond \alpha \cap \{P\} \diamond \beta)$. This implies $X \in T_{P,\alpha}$ and $X \in T_{P,\beta}$. Therefore, $X \models \alpha$ with its distance from P being $d_M(P, X) = \delta_{P,\alpha}$, and $X \models \beta$ with its distance from P also being $d_M(P, X) = \delta_{P,\beta}$. It follows directly that $\delta_{P,\alpha} = \delta_{P,\beta} = d_M(P, X)$, as X achieves the minimum distance from P for both individual revisions.

Now, consider the computation of $\{P\} \diamond (\alpha \vee \beta)$, which is determined by `findTuplesWithConstraints` $(P, \alpha \vee \beta, \dots)$. This procedure internally processes the condition $\alpha \vee \beta$ by splitting it at the `"||"` operator into `orParts` $= [\alpha, \beta]$. It then iterates through these components to identify tuples satisfying either disjunct that are closest to P .

When processing the first component, α , the procedure identifies $T_{P,\alpha}$ as the set of closest models to P for α , with the minimum distance being $\Delta_\alpha = \delta_{P,\alpha}$. The internal state variable `globalMinDistance` is set to Δ_α , and `globalAssignments` is initialised with $T_{P,\alpha}$. Since $X \in T_{P,\alpha}$, X is included in `globalAssignments`.

Subsequently, when processing the second component, β , the procedure identifies $T_{P,\beta}$ as the set of closest models to P for β , with the minimum distance being $\Delta_\beta = \delta_{P,\beta}$. Given that $\delta_{P,\alpha} = \delta_{P,\beta} = d_M(P, X)$, it holds that $\Delta_\beta = d_M(P, X)$. Since $\Delta_\beta = d_M(P, X) = \delta_{P,\alpha}$, the condition $\Delta_\beta = \text{globalMinDistance}$ is met.

Consequently, the update rule for *globalAssignments* dictates $\text{globalAssignments} \leftarrow \text{globalAssignments} \cup T_{P,\beta}$. Because X was already present in *globalAssignments*—due to its inclusion in $T_{P,\alpha}$ —and $X \in T_{P,\beta}$, X is guaranteed to be retained in the resulting *globalAssignments* set.

The function `findTuplesWithConstraints` ultimately returns *globalAssignments* as the set of tuples for $\{P\} \diamond (\alpha \vee \beta)$. Since $X \in \text{globalAssignments}$, it follows that $X \in \{P\} \diamond (\alpha \vee \beta)$. As X was an arbitrary element of $\{P\} \diamond \alpha \cap \{P\} \diamond \beta$, it is concluded that:

$$\{P\} \diamond \alpha \cap \{P\} \diamond \beta \subseteq \{P\} \diamond (\alpha \vee \beta)$$

which satisfies the postulate.

$$\textbf{(U8)} \quad (\Gamma_1 \cup \Gamma_2) \diamond \alpha = (\Gamma_1 \diamond \alpha) \cup (\Gamma_2 \diamond \alpha).$$

This postulate requires that the update operator $\diamond$ be distributive over the union of profile sets. Updating the union of Γ_1 and Γ_2 by α must yield the same set of profiles as updating each set individually by α and then taking the union of the results.

Fulfilment:

Let L_T denote the set of profiles resulting from $(\Gamma_1 \cup \Gamma_2) \diamond \alpha$. Based upon the operation of `model14`, $L_T = \bigcup_{P \in (\Gamma_1 \cup \Gamma_2)} T_{P,\alpha}$.

Let R_{1T} denote the set of profiles from $\Gamma_1 \diamond \alpha$. According to `model14`, $R_{1T} = \bigcup_{P \in \Gamma_1} T_{P,\alpha}$. Let R_{2T} denote the set of profiles from $\Gamma_2 \diamond \alpha$. According to `model14`, $R_{2T} = \bigcup_{P \in \Gamma_2} T_{P,\alpha}$. The set of profiles for $(\Gamma_1 \diamond \alpha) \cup (\Gamma_2 \diamond \alpha)$ is therefore $R_{1T} \cup R_{2T}$.

By the distributive property of set union over a collection of sets, it holds that:

$$\bigcup_{P \in (\Gamma_1 \cup \Gamma_2)} T_{P,\alpha} = \left(\bigcup_{P \in \Gamma_1} T_{P,\alpha} \right) \cup \left(\bigcup_{P \in \Gamma_2} T_{P,\alpha} \right).$$

Consequently, $L_T = R_{1T} \cup R_{2T}$, and thus Postulate **U8** is satisfied.

$$\textbf{(U9)} \quad \text{If } \|\alpha \wedge \beta\| \neq \emptyset, P \text{ is a profile and } (\{P\} \diamond \alpha) \cap \|\beta\| \neq \emptyset, \text{ then } \{P\} \diamond (\alpha \wedge \beta) \subseteq (\{P\} \diamond \alpha) \cap \|\beta\|.$$

This postulate states that if the conjunction $\alpha \wedge \beta$ is satisfiable, and if there are profiles resulting

from updating $\{P\}$ by α that also satisfy β , then the result of updating $\{P\}$ by $\alpha \wedge \beta$ must be a subset of those profiles from $\{P\} \diamond \alpha$ that satisfy β .

Fulfilment:

Let $\Gamma = \{P\}$. As Γ is a singleton set, this effectively reduces to the execution of the procedure `findTuplesWithConstraints`(P, ϕ). Let $T_{P,\phi}$ denote the set of tuples returned by this procedure, such that $\{P\} \diamond \phi = T_{P,\phi}$. Let $\delta_{P,\phi} = \min_{\tau \models \phi} d_M(P, \tau)$ be the associated minimum Manhattan distance, where all $\tau \in T_{P,\phi}$ have $d_M(P, \tau) = \delta_{P,\phi}$.

The premises of the postulate are: (1) $\|\alpha \wedge \beta\| \neq \emptyset$, ensuring that $\delta_{P,\alpha \wedge \beta}$ is well-defined and finite; and (2) $(T_{P,\alpha} \cap \|\beta\|) \neq \emptyset$. Let $S = T_{P,\alpha} \cap \|\beta\|$. The non-emptiness of S implies the existence of at least one profile $X \in S$ such that $X \in T_{P,\alpha}$ (thereby $X \models \alpha$ and $d_M(P, X) = \delta_{P,\alpha}$) and $X \models \beta$. Consequently, $X \models (\alpha \wedge \beta)$.

Since X is a model of $\alpha \wedge \beta$ and its distance from P is $\delta_{P,\alpha}$, the minimum distance to any model of $\alpha \wedge \beta$, denoted $\delta_{P,\alpha \wedge \beta}$, must satisfy $\delta_{P,\alpha \wedge \beta} \leq d_M(P, X) = \delta_{P,\alpha}$.

Now, consider any arbitrary tuple $Y \in T_{P,\alpha \wedge \beta}$. By definition, $Y \models (\alpha \wedge \beta)$ and $d_M(P, Y) = \delta_{P,\alpha \wedge \beta}$. Since $Y \models (\alpha \wedge \beta)$ implies $Y \models \alpha$, and $\delta_{P,\alpha}$ is the minimum distance from P to any model of α , it must be that $\delta_{P,\alpha} \leq d_M(P, Y) = \delta_{P,\alpha \wedge \beta}$.

Combining these two inequalities, $\delta_{P,\alpha \wedge \beta} \leq \delta_{P,\alpha}$ and $\delta_{P,\alpha} \leq \delta_{P,\alpha \wedge \beta}$, it is concluded that $\delta_{P,\alpha \wedge \beta} = \delta_{P,\alpha}$. Let this common minimal distance be denoted δ^* .

The set $T_{P,\alpha \wedge \beta}$ thus consists of all tuples τ such that $\tau \models (\alpha \wedge \beta)$ and $d_M(P, \tau) = \delta^*$. The set $S = T_{P,\alpha} \cap \|\beta\|$ consists of all tuples τ such that $\tau \in T_{P,\alpha}$ and $\tau \models \beta$. This implies that tuples in S satisfy $\tau \models (\alpha \wedge \beta)$ and have $d_M(P, \tau) = \delta^*$.

Therefore, the set $T_{P,\alpha \wedge \beta}$ is precisely identical to the set $S = T_{P,\alpha} \cap \|\beta\|$. From this equality, $\{P\} \diamond (\alpha \wedge \beta) = (\{P\} \diamond \alpha) \cap \|\beta\|$, the subset relation specified by the postulate, $\{P\} \diamond (\alpha \wedge \beta) \subseteq (\{P\} \diamond \alpha) \cap \|\beta\|$, follows directly. This satisfies Postulate **U9**.

(U10) If $\Gamma_1 \subseteq \Gamma_2$, then $\Gamma_1 \diamond \alpha \subseteq \Gamma_2 \diamond \alpha$.

This postulate states that if a set of profiles Γ_1 is a subset of another set Γ_2 , then updating Γ_1 by

α should result in a set of profiles that is a subset of (or equal to) the set resulting from updating I_2 by α .

Fulfilment:

Let $L = I_1 \diamond \alpha$ and $R = I_2 \diamond \alpha$ denote the sets of profiles resulting from the update operation performed by `model4`. Assume the premise $I_1 \subseteq I_2$. The set of tuples in L is $L_T = \bigcup_{P \in I_1} T_{P,\alpha}$ and in R is $R_T = \bigcup_{P \in I_2} T_{P,\alpha}$.

Since $I_1 \subseteq I_2$, every profile P that is an element of I_1 is also necessarily an element of I_2 . Consequently, every set $T_{P,\alpha}$ that is a constituent of the union forming L_T is also a constituent set of the union forming R_T .

By the fundamental properties of set union, if all sets contributing to one union are also included among the sets contributing to another union, the first union must be a subset of (or equal to) the second. Therefore, it holds that:

$$\bigcup_{P \in I_1} T_{P,\alpha} \subseteq \bigcup_{P \in I_2} T_{P,\alpha}$$

This demonstrates that the set of profiles $I_1 \diamond \alpha$ is a subset of $I_2 \diamond \alpha$, thereby satisfying Postulate **U10**.

(U11) $I \cap \|\alpha\| \subseteq I \diamond \alpha$.

This postulate states that any profile initially present in the set I that already satisfies the formula α must also be included in the result of updating I by α .

Fulfilment:

Let X be an arbitrary profile such that $X \in (I \cap \|\alpha\|)$. This implies two conditions: (1) $X \in I$ and (2) $X \models \alpha$.

Since $X \in I$, the computation of $I \diamond \alpha$ involves a specific invocation with X as the target profile, namely `findTuplesWithConstraints`($X, \alpha, \text{minValue}, \text{maxValue}$). Because $X \models \alpha$, the procedure identifies X as a model of α . The Manhattan distance from X to itself is $d_M(X, X) = 0$.

As no other tuple can achieve a distance less than zero from X , the set of closest models of α to X , denoted $T_{X,\alpha}$, necessarily includes X . In this specific instance, $T_{X,\alpha} = \{X\}$.

Given that $X \in T_{X,\alpha}$, and noting that $\Gamma \diamond \alpha = \bigcup_{P \in \Gamma} T_{P,\alpha}$, it follows directly that $T_{X,\alpha}$ is one of the constituent sets contributing to this union because $X \in \Gamma$. Therefore, X must be an element of the resulting union.

Consequently, $X \in \Gamma \diamond \alpha$. As X was an arbitrary element of $\Gamma \cap \|\alpha\|$, it is concluded that $\Gamma \cap \|\alpha\| \subseteq \Gamma \diamond \alpha$, thereby satisfying Postulate **U11**.

6.2 Implementation Details

This subsection presents the comprehensive implementation details governing profile dynamics.

6.2.1 Sequence Diagram of the implemented modules.

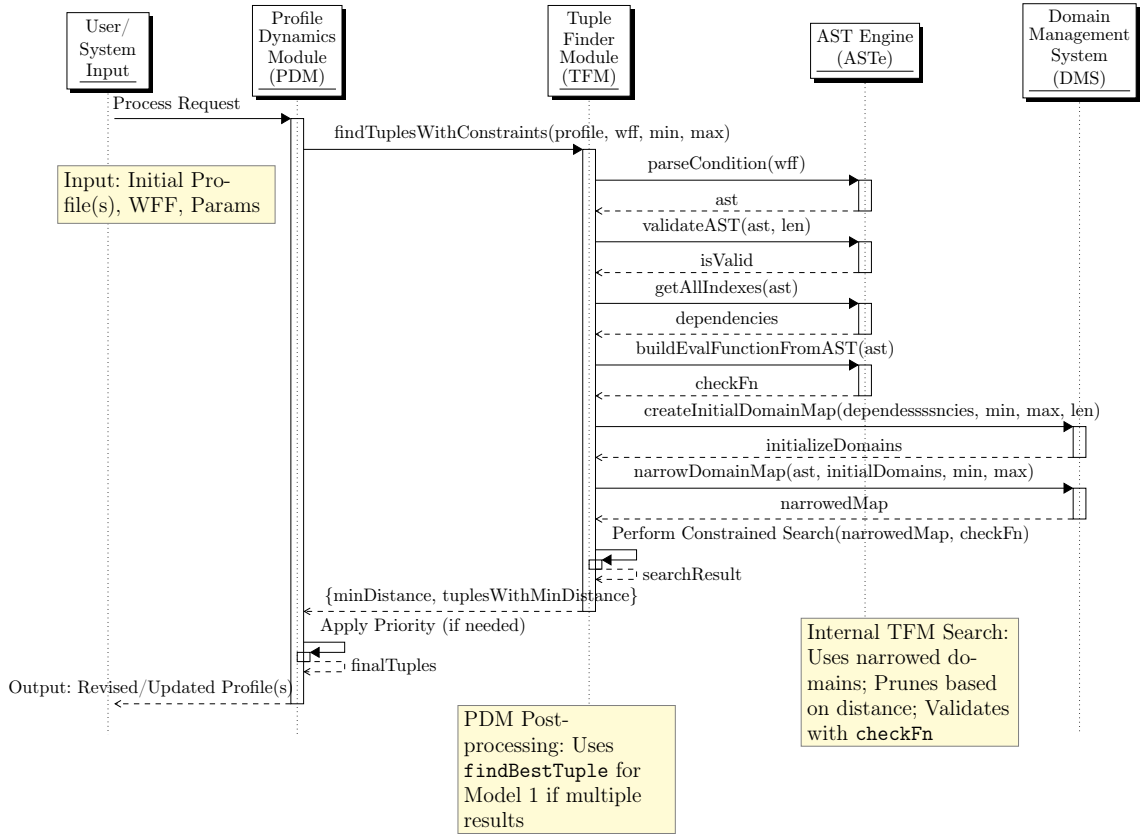


Figure. 11: Sequence of calls and data exchanges used to process a KDPD profile-revision or update request. The PDM receives the initial profile or profile set, the wff, domain bounds, and model parameters, and delegates constrained tuple search to the TFM. The TFM requests the ASTe to parse and validate the wff, identify the referenced indices, and construct an executable predicate. It then requests the DMS to initialise and narrow the corresponding value domains. Using the narrowed domains, the TFM performs distance-bounded search, validates candidates using the generated predicate, and returns the minimum Manhattan distance and all tuples attaining it. The PDM subsequently applies priority-based tie-breaking when required by Model 1 and returns the final revised or updated profile or profile set.

6.2.2 Abstract Syntax Tree Engine

6.2.2.1 Parsing and Structural Transformation of the wff

The conversion of a wff, initially provided as a string, into a system-specific AST is a foundational process orchestrated by the ASTe. This transformation is pivotal, as it translates a human-readable condition into a structured hierarchical representation amenable to programmatic validation, analysis, and the subsequent generation of executable code. The process unfolds in two primary phases: initial syntactic parsing by an external library and a subsequent recursive transformation into the system's custom AST format, a task managed chiefly by the `astBuilder.js` module.

Phase 1: Initial Syntactic Parsing via Acorn

The entry point for wff processing within the ASTe is the function `parseCondition`, defined in `astBuilder.js`. Upon receiving the wff string, the first step involves the invocation of the Acorn parser. Acorn is a robust JavaScript parser capable of transforming JavaScript-like expression strings into a standard AST format that complies with the ESTree specification. Because the wff processed by this system are designed to adhere to a subset of JavaScript expression syntax, Acorn is an appropriate choice for this initial, low-level syntactic analysis.

The invocation of Acorn is encapsulated within a `try...catch` block to manage potential syntax errors in the input wff string:

```
// Snippet from astBuilder.js - parseCondition function
// String received (\ac{wff}):
// conditionStr = "el[0] > 10 \&\& el[1] < 20"
let acornAST;
try {
  // Wrap in parentheses to ensure expression parsing
  acornAST = acorn.parse(`(${conditionStr})`,
    { ecmaVersion: 'latest' });
} catch (e) {
  throw new Error(
    'Invalid condition syntax: ${e.message}');
}
// The Acorn \ac{ast} is in acornAST.body[0].expression
```

In the snippet above, the system (via Acorn) receives a wff string, `conditionStr`, for processing. The system specifically expects the string to be a single expression. If the string violates JavaScript expression syntax—for instance, through mismatched parentheses or invalid operators at this stage—a system-specific error is raised to indicate the nature of the problem. Otherwise, the procedure produces a standard AST representing the parsed wff expression.

Phase 2: Recursive Transformation to Custom AST via `buildAST`

Once the standard Acorn AST is obtained, the `parseCondition` function delegates the core task of converting this generic AST into the system's custom AST structure to an internal recursive helper function named `buildAST`. This function traverses the Acorn AST node by node; for each node, it determines the appropriate custom AST node type (defined in `astNodes.js`) to instantiate. The logic within `buildAST` is primarily driven by a switch statement operating on the `type` property of the current Acorn AST node.

The transformation process handles various Acorn node types, mapping them to their corresponding custom AST nodes as follows:

- **Acorn Literal nodes:** Primitive data values, such as numbers, strings, or booleans, are directly mapped to instances of the custom `LiteralNode` class. A `LiteralNode` encapsulates this raw value, with its primary function being the generation of the JavaScript string representation via its `buildCode()` method.
- **Acorn Identifier nodes:** These represent names or variables. The `buildAST` function maps Acorn `Identifier` nodes to instances of the custom `IdentifierNode`. This class contains specific logic: if the identifier's name is `"e1"` (the designated identifier for accessing profile elements), it is treated as a valid reference. Other identifiers encountered lead to `buildCode()` generating `"undefined"`, reflecting their unsupported status as general-purpose variables within wff. The primary purpose of `IdentifierNode` for `"e1"` is to serve as the object component of a member expression, such as `e1[0]`.
- **Acorn MemberExpression nodes:** These represent property access, such as `object.property` or `object[property]`. The `buildAST` function imposes strict constraints on `MemberExpression` nodes:

- If the expression follows the form `el[index]`, where `el` is an `Identifier` and `index` is a numerical `Literal`, it is transformed into an instance of the custom `ArrayAccessNode`. This class represents an access to a specific element of the profile tuple and stores the numerical `index`. Its `buildCode()` method generates JavaScript code such as: `"((assignment[index] !== undefined) ? assignment[index] : target[index])"` This allows evaluation against a target profile whilst providing a potential override from an `assignment` object, directly supporting the formal language construct L_i .
 - The procedure also verifies if the member expression constitutes a call to a `Math` object property.
 - Any other form of `MemberExpression` is considered unsupported, causing `buildAST` to throw an error to ensure that only valid profile element accesses are permitted.
- **Acorn BinaryExpression nodes:** These represent operations with two operands and a binary operator (e.g. `a + b`, `x > y`). The `buildAST` function maps these to instances of the custom `BinaryExpressionNode`. This class stores the `operator` and recursively processes custom AST nodes for its `left` and `right` operands. Its `buildCode()` method generates JavaScript code by combining the code generated by its children with the operator, typically enclosed in parentheses to ensure correct precedence. This node type handles both arithmetic ($T_1 \text{ op}_{\text{arith}} T_2$) and relational ($T_1 \text{ op}_{\text{rel}} T_2$) operations from the formal language $\mathcal{L}_p$.
 - **Acorn LogicalExpression nodes:** These represent logical operations with two operands, specifically `&&` (logical AND) and `||` (logical OR). The `buildAST` function maps these to instances of the custom `LogicalExpressionNode`. Similar to the `BinaryExpressionNode`, it stores the `operator` and recursively processes the `left` and `right` operands.
 - **Acorn CallExpression nodes:** These represent function calls. The `buildAST` function restricts these to a specific pattern: only calls to methods of the global `Math` object are permitted (e.g. `Math.sqrt(el[0])`).
 - It verifies that the `callee` is a `MemberExpression` whose `object` is an `Identifier` named `Math` and whose `property` is a supported math method.

- If these conditions are met, a custom `MathCallNode` is instantiated, storing the `method` name and an array of `args`.
 - Any other `CallExpression` results in an error, as general function calls are not supported.
- **Acorn ParenthesizedExpression nodes:** These are explicitly handled to ensure that parentheses from the original expression's logic are preserved for the correct evaluation order. The `buildAST` function recursively processes the inner `expression` and wraps the result in a `ParenthesizedNode`, whose `buildCode()` method wraps the resulting inner code in parentheses.
 - **Unsupported Node Types:** If `buildAST` encounters an Acorn node type for which no mapping is defined (e.g. assignment expressions, function declarations, or loop statements), it throws an error. This acts as a filter, ensuring that only the subset of JavaScript expressions relevant to $\mathcal{L}_p$ and deemed safe for evaluation are transformed.

All custom AST node classes inherit from a `BaseNode`, which establishes their `type` property and mandates the implementation of the `buildCode()` method.

For example, consider the wff string: `"el[0] > 10 && Math.sqrt(el[1]) < 5"`.

1. Acorn parses this string into a standard ESTree-compliant AST.
2. `buildAST` then recursively transforms this into the system's custom structure. In this instance, the root node becomes a `LogicalExpressionNode` (operator: `&&`).
 - Its left child is a `BinaryExpressionNode` (operator: `>`).
 - The left child of the `>` node is an `ArrayAccessNode` (index: 0).
 - The right child of the `>` node is a `LiteralNode` (value: 10).
 - Its right child is a `BinaryExpressionNode` (operator: `<`).
 - The left child of the `<` node is a `MathCallNode` (method: `sqrt`).
 - * The argument to `sqrt` is an `ArrayAccessNode` (index: 1).
 - The right child of the `<` node is a `LiteralNode` (value: 5).

A visual representation of this custom AST structure is provided in Figure 12.

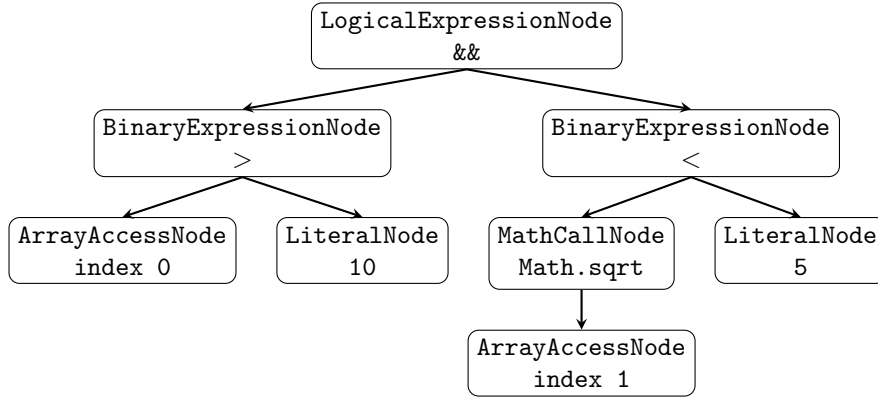


Figure. 12: Custom abstract syntax tree generated by the AST engine for the wff `e1[0] > 10 && Math.sqrt(e1[1]) < 5`. The root `LogicalExpressionNode` represents conjunction (`&&`). Its left subtree is a `BinaryExpressionNode` comparing tuple element `e1[0]`, represented by an `ArrayAccessNode` with index 0, against the literal 10. Its right subtree is a `BinaryExpressionNode` comparing `Math.sqrt(e1[1])`, represented by a `MathCallNode` applied to an `ArrayAccessNode` with index 1, against the literal 5. The figure illustrates how JavaScript syntax is converted into the restricted, system-specific node structure used for validation, dependency extraction, domain narrowing, and predicate generation.

6.2.2.2 Syntactic and Semantic Validation of the Custom AST

Subsequent to the transformation of the input wff string into a custom AST, the ASTe proceeds to the validation phase. Orchestrated primarily by the `validateAST` function within the `astValidator.js` module, this phase is designed to ensure that the custom AST is both structurally sound according to its formation rules and semantically meaningful within the system's constraints. This validation step is essential to prevent run-time errors during wff evaluation and to ensure the formula adheres to the expressive limitations of the profile language $\mathcal{L}_p$.

The `validateAST` function serves as the foundation of this process. It accepts two primary arguments: the root node of the custom AST (or any subtree thereof during recursive calls) and an integer `tupleLength`, representing the arity of the profiles against which the wff will be evaluated. The function returns a result object in the form: `{valid: boolean, error?: string}`. A `valid` status of `true` signifies that the AST segment conforms to all validation rules, whereas `false` indicates a violation; in the latter case, the optional `error` property provides a descriptive message of the issue encountered. This function operates through a recursive traversal of the custom AST. For each node, it applies specific validation checks based on the node's type, as defined by its `ast._type` property. The principal validation routines are as follows:

Validation of Tuple Element Access (`ArrayAccessNode`)

When an `ArrayAccessNode` is encountered, a crucial bounds check is performed. The `index` property stored within the `ArrayAccessNode` is verified to ensure that it falls within the permitted range $[0, \text{tupleLength} - 1]$.

```
// Conceptual check for ArrayAccessNode (el[index])
if (ast._type === NODE_TYPES.MemberExpression) {
  const idx = ast.index;
  if (idx < 0 || idx >= tupleLength) {
    return { valid: false,
            error: 'Index ${idx} out of bounds!' };
  }
}
```

A failure of this check results in the AST being marked as invalid, accompanied by an error message such as: “Index 3 out of bounds!”. This mechanism ensures that wff do not attempt to reference non-existent profile elements.

Validation of Binary Operations (`BinaryExpressionNode`)

For nodes of type `BinaryExpressionNode`, which represent both arithmetic (e.g. `el[0] + 5`) and relational (e.g. `el[1] > 10`) operations, the validation proceeds in two stages. Firstly, `validateAST` is invoked recursively on the left and right child nodes. If either sub-expression is invalid, the parent `BinaryExpressionNode` is also deemed invalid, and the child’s error is propagated up the tree.

Secondly, if both operands are valid, the `operator` property of the `BinaryExpressionNode` itself is validated. The system verifies whether this operator is a recognised arithmetic operator—matching a key in `arithOpsMap` within `constants.js`—or a supported boolean/relational operator, matching a key in `boolOpsMap`.

```
// Conceptual check for BinaryExpressionNode
if (ast._type === NODE_TYPES.BinaryExpression) {
  // ... recursive validation of ast.left and ast.right ...
  if (!boolOpsMap[ast.operator] &&
      !arithOpsMap[ast.operator]) {
```

```

        return { valid: false,
                error: 'Unsupported operator:
                        ${ast.operator}' };
    }
}

```

An operator not found within these predefined maps results in an invalid AST status, accompanied by an error such as: “Unsupported operator: **”.

Validation of Logical Operations (LogicalExpressionNode)

Nodes representing logical operations (`&&`, `||`), termed `LogicalExpressionNode`, undergo a similar validation process. The left and right child nodes are recursively validated; subsequently, the operator is verified against the `logicalOpsMap`. Unsupported symbols produce an error message such as: “Invalid logical operator: &”.

```

// Conceptual check for LogicalExpressionNode
if (ast._type === NODE_TYPES.LogicalExpression) {
    // ... recursive validation of ast.left and ast.right ...
    if (!logicalOpsMap[ast.operator]) {
        return { valid: false,
                error: 'Invalid logical operator:
                        ${ast.operator}' };
    }
}

```

Validation of Parenthesised Expressions (ParenthesizedNode)

For `ParenthesizedNode` instances, which preserve explicit grouping within the expression, `validateAST` recurses on the inner expression. The validity of the parenthesised expression is determined solely by the validity of its content.

Validation of Function Calls (MathCallNode)

Custom nodes of type `MathCallNode`, representing invocations of global `Math` methods, are validated by recursively checking each argument within the `ast.args` array. If any argument is found to be invalid, the entire `MathCallNode` is rendered invalid. The validity of the method name

itself is enforced during the transformation phase by restricting selection to those methods defined within the `mathOpsMap`.

If every node within the custom AST passes its respective check, `validateAST` returns `{valid: true}`. This result indicates that the wff, as represented by the custom AST, is structurally sound, employs only supported operations, and maintains profile element references consistent with the specified `tupleLength`. At this stage, the formula is deemed ready for subsequent processing.

6.2.2.3 Identification of Constrained Profile Attributes

Following the successful parsing and structural transformation of a wff string into the system's custom AST, the ASTe identifies all unique profile attributes that are actively referenced or restricted within that wff. These attributes, represented by their numerical indices within a profile tuple, are collectively termed *dependencies*.

Specifically, this identification enables the DMS to focus its domain-narrowing procedures exclusively on these pertinent attributes, rather than on the entire profile. Similarly, it allows the TFM to structure its search space more effectively around these constrained elements. This targeted approach is fundamental to maintaining computational feasibility, particularly when dealing with profiles of high arity or complex wff.

This dependency extraction task is performed primarily by the `getAllIndexes` function, a utility located in the `astUtils.js` module. The `getAllIndexes` function itself is a specialised application of a more generic AST traversal mechanism, `walkAST`, also provided by `astUtils.js`.

The `walkAST` procedure implements a depth-first traversal of the custom AST. Given any node and a user-supplied callback, it first tests for the base case (null or undefined) and returns immediately if this condition is met. Otherwise, it invokes the callback on the current node and subsequently dispatches according to the node's type:

- For **binary** or **logical** expressions, it recurses into both left and right children.
- For **parenthesised** expressions, it descends into the single enclosed sub-expression.
- For **call expressions** (such as `Math` calls), it iterates through and recurses on each argument.
- For all **leaf nodes** (literals, identifiers, array accesses), it terminates without further recursion.

In this manner, `walkAST` guarantees that every reachable node in the AST is visited exactly once, allowing arbitrary analyses or transformations to be applied uniformly across the tree.

In order to determine which profile elements are constrained by a given wff, the `getAllIndexes` function first instantiates an empty `Set`—designated as `indexSet`—to guarantee the uniqueness of the captured indices. It subsequently invokes the generic traversal routine `walkAST` on the AST root, supplying a callback that inspects each visited node. Whenever a node is identified as a `MemberExpression` (specifically, an `ArrayAccessNode` of the form `e1[index]`), the callback extracts the numerical `index` and adds it to the `indexSet`. Once the traversal is complete, `getAllIndexes` returns the `indexSet`, which contains every distinct profile attribute index referenced within the formula.

Consider the same wff as before, "`e1[0] > 10 && Math.sqrt(e1[1]) < 5`", as illustrated in Figure 12. When `getAllIndexes` traverses this AST:

1. The traversal encounters the `ArrayAccessNode` representing `e1[0]` (serving as the left operand of the `>` operator). The `callback` function extracts the index 0 and adds it to the `indexSet`.
2. Subsequently, during the traversal of the arguments for the `MathCallNode`, the routine encounters the `ArrayAccessNode` for `e1[1]`. The `callback` extracts the index 1 and adds it to the `indexSet`.

The `getAllIndexes` function then returns the set `{0,1}`. This set of constrained attributes is passed to the DMS to initiate the domain-narrowing process specifically for these elements, and to the TFM to guide its search space configuration.

6.2.2.4 Generation of an Executable Predicate

Following the successful parsing of a wff string into a custom AST and its subsequent validation, the ASTe transforms this structured, validated AST into an executable predicate. This predicate, a dynamically generated JavaScript function, encapsulates the logic of the original wff. Its purpose is to provide a mechanism for the TFM to determine whether a given candidate profile (or a partial assignment of profile attributes) satisfies the conditions of the wff. This process is orchestrated by `astBuilder.js`, leveraging the `buildCode()` method implemented by each custom AST node class defined in `astNodes.js`.

The `buildCode()` Method: Recursive Code String Generation

Each node class in the custom AST implements `buildCode()`, which returns the JavaScript code fragment representing that specific node. A recursive traversal originating from the AST root concatenates these fragments into a complete predicate string.

– **LiteralNode:**

Returns the JSON-stringified value of the literal.

```
// Conceptual buildCode() for LiteralNode
return JSON.stringify(this.value);
```

– **IdentifierNode:**

Translates the primary identifier "el" to the internal variable "target"; all other identifiers are mapped to "undefined".

```
// Conceptual buildCode() for IdentifierNode
return this.name === 'el' ? "target" : "undefined";
```

– **ArrayAccessNode:**

Generates a secure element access that prioritises a partial `assignment` over the full `target` tuple.

```
// Conceptual buildCode() for ArrayAccessNode
return (
  (assignment[this.index] !== undefined)
  ? assignment[this.index]
  : target[this.index]
);
```

– **BinaryExpressionNode:**

Recursively constructs its left and right operands and inserts the operator between them.

```
// Conceptual buildCode() for BinaryExpressionNode
return (
  this.left.buildCode() + " " +
  this.operator + " " +
```

```

    this.right.buildCode()

    );

```

– LogicalExpressionNode:

Follows a procedure analogous to `BinaryExpressionNode`, specifically for `&&` and `||` operators.

```

// Conceptual buildCode() for LogicalExpressionNode
return (
    this.left.buildCode() + " " +
    this.operator + " " +
    this.right.buildCode()
);

```

– MathCallNode:

Constructs each argument string, joins them with commas, and encapsulates them within a `Math.method(...)` call.

```

// Conceptual buildCode() for MathCallNode
const argsCode = this.args.map(a => a.buildCode()).join(', ');
return Math.${this.method}(${argsCode});

```

– ParenthesizedNode:

Ensures explicit grouping is preserved by wrapping the inner expression in parentheses.

```

// Conceptual buildCode() for ParenthesizedNode
return "(" + this.expression.buildCode() + " ";

```

Invoking `buildCode()` on the AST root yields a single JavaScript expression string representing the entirety of the wff.

Dynamic Function Creation

The `buildEvalFunctionFromAST` function within `astBuilder.js` receives the code string produced by `ast.buildCode()` and constructs a new function at runtime:

```

// Snippet from astBuilder.js

```

```

export function buildEvalFunctionFromAST(ast) {

  return new Function(

    'assignment', 'target',

    'return ${ast.buildCode()};'

  );

}

```

In this implementation, 'assignment' and 'target' designate the function parameters. The resulting function evaluates the wff when invoked with a partial assignment and a complete profile tuple.

Illustrative Example of Predicate Generation

Given the wff:

```
"el[0] > 10 && Math.sqrt(el[1]) < 5"
```

The recursive buildCode() invocations generate the following string:

```

"(((assignment[0] !== undefined) ?
  assignment[0] : target[0]) > 10)
&& (Math.sqrt(((
  assignment[1] !== undefined) ?
  assignment[1] : target[1])) < 5)"

```

Subsequently, buildEvalFunctionFromAST constructs the executable function as follows:

```

new Function(

  'assignment', 'target',

  'return (((assignment[0] !== undefined) ?
    assignment[0] : target[0]) > 10)
    && (Math.sqrt(((
      assignment[1] !== undefined) ?
      assignment[1] : target[1])) < 5);'

)

```

This process yields a validated predicate function, which is then utilised by the TFM for candidate profile evaluation.

6.2.3 Domain Management System

6.2.3.1 Representation and Management of Permissible Attribute Values

The efficacy of the DMS in reducing the combinatorial search space for the TFM is intrinsically linked to its capacity to represent and manipulate the sets of permissible values attributable to each element within a profile. This responsibility is encapsulated in the `Domain` class (located in `Domain.js`), whose design supports a flexible, integer-only representation of attribute domains. The following analysis examines its internal structure and core methods.

Internal Structure of the Domain Class

An instance of `Domain` represents the allowable integer values of a single profile attribute via two primary fields:

1. **intervals**: an `Array` of objects in the form `{min: INT, max: INT}`, each denoting a continuous, inclusive integer interval `[min, max]`.
2. **discreteValues**: a `Set` of integers that lie outside all defined intervals, ensuring each discrete value is stored uniquely.

The constructor `new Domain()` initialises both structures as empty, yielding the most restrictive (empty) domain.

Core Methods for Domain Manipulation

The `Domain` class exposes the following functions:

addInterval(minValue, maxValue) This method first verifies that $\text{minValue} \leq \text{maxValue}$, with both endpoints subsequently converted to integers. A new interval object containing these values is appended to **intervals**. Following this, all intervals are sorted in ascending order of their **min** property; any overlapping or adjacent intervals are then merged into a single interval whose bounds span the union of the constituents. Finally, any values in **discreteValues** that now reside within one of the merged intervals are removed. This ensures that discrete values represent only those points not covered by a continuous interval.

addDiscreteValues(...values) This method accepts a variadic list of numeric arguments. Each value is floored to an integer and subsequently tested against the existing intervals: if the value does not fall within any `[min, max]` interval, it is inserted into the **discreteValues** `Set`. The

use of a `Set` guarantees that duplicates are automatically discarded and that no discrete value replicates a point already covered by an interval.

`isEmpty()` Returns `true` if and only if `intervals` is empty and `discreteValues` contains no elements. Otherwise, it returns `false`.

`contains(value)` Floors the input `value` to an integer and subsequently verifies membership: it returns `true` if the integer is present in `discreteValues` or if it lies between the `min` and `max` (inclusive) of any interval within `intervals`; otherwise, it returns `false`.

`intersect(otherDomain)` Constructs and returns a new `Domain` instance representing the set-theoretic intersection of the current domain and `otherDomain`. This process is executed in three stages:

1. **Interval Intersection:** For every interval $[a, b]$ in `intervals` and every interval $[c, d]$ in `otherDomain.intervals`, the procedure computes the potential overlap as $[\max(a, c), \min(b, d)]$. If this overlap is non-empty, it is added to the new domain's intervals.
2. **Discrete Value Intersection:** Each discrete value from both domains is evaluated. Any value x in `discreteValues` for which `otherDomain.contains(x)` holds, and any value x in `otherDomain.discreteValues` for which `this.contains(x)` holds, is retained within the new domain's `discreteValues`.
3. **Normalisation:** Finally, the resulting domain is normalised by merging overlapping intervals and removing any discrete values that have become redundant due to interval coverage.

This operation serves as the core mechanism by which multiple constraints on the same attribute are combined during the narrowing process.

Example: Let Domain A comprise the interval $[0, 10]$ and the discrete value $\{15\}$; let Domain B comprise the interval $[5, 12]$ and the discrete values $\{15, 20\}$. Their intersection yields the interval $[5, 10]$ and the discrete value $\{15\}$.

Additional utilities, such as `equals(other)` (determining the logical equivalence of domains) and `toString()` (generating a human-readable listing of intervals and discrete points), support debugging and comparative analysis. Through this structural design and method suite, the DMS accurately and efficiently narrows attribute domains under successive constraints.

6.2.3.2 Initialisation and Iterative Refinement of Value Ranges

The DMS operates in two primary phases. Firstly, it constructs an initial map of allowable value ranges for each constrained attribute. Secondly, it iteratively refines those ranges by propagating the logical and arithmetic constraints encoded within the AST until no further narrowing is possible or a domain becomes empty.

The refinement process itself is not the primary focus of this study. The procedures employed are straightforward and efficient, serving primarily to demonstrate their effect on execution time. It is acknowledged that considerable further investigation is warranted to optimise performance fully.

Phase 1: Initial Domain Map Construction

Following the identification of the dependent attribute indices by the ASTe (Section 6.2.2.3), the TFM invokes the function `createInitialDomainMap`. This invocation includes a `Set` of indices, the global minimum `minValue`, the global maximum `maxValue`, and the tuple length `len`.

An empty `Map`, designated `domainMap`, is created. For each index i from 0 to `len` – 1, if i is present within the dependencies set, a new `Domain` instance is initialised. This domain is populated using `addInterval(minValue, maxValue)` to represent the full permissible range, and the pair (i, Domain) is subsequently added to the `domainMap`. Indices not contained within the set are omitted. Consequently, the map assigns the broadest possible domain to each constrained attribute prior to any specific narrowing operations.

Example. If `dependencies = {0, 2}`, `minValue = 1`, `maxValue = 10`, and `len = 3`, then:

$$\text{domainMap} = \{ 0 \mapsto [1, 10], 2 \mapsto [1, 10] \},$$

and index 1 contains no entry.

Phase 2: Iterative Domain Refinement via Constraint Propagation

The function `narrowDomainMap` accepts the `ast`, the initial `domainMap`, and the global bounds `minValue` and `maxValue`. The procedure then repeats the following sequence until either no further domain changes occur or a domain becomes empty:

1. **State Persistence:** A deep clone of the current `domainMap` is created to facilitate subsequent comparison.

2. **Constraint Application:** The AST is traversed using `walkAST`, which supplies the following callback process: `ProcessASTNodeForNarrowing(node, domainMap, minValue, maxValue)`. This callback inspects each boolean-valued node (represented by a `BinaryExpressionNode`) and invokes: `_refineBooleanConstraint(leftAST, rightAST, op, domainMap, minValue, maxValue)`.
3. **Convergence Check:** The updated `domainMap` is compared to its previously stored clone. If the maps are identical, indicating that no further narrowing has occurred, the loop is exited.
4. **Unsatisfiability Check:** If any domain within the `domainMap` is found to be empty (via `isEmpty()`), the process is terminated immediately, as the clause is deemed unsatisfiable.

The `_refineBooleanConstraint` Function

Given a relational or equality operator `op` and two sub-expressions `L` and `R`, this function proceeds in two distinct stages:

Stage 1: Forward Range Computation

The function computes the current numeric range of each side by invoking `_computeArithRange(node, domainMap, minValue, maxValue)`:

- For `LiteralNode`, the range is defined as `[value, value]`.
- For `ArrayAccessNode` (profile access `el[index]`), the range is derived from the union of the domain's intervals and discrete values. If the domain is empty or missing, the result is `null`.
- For arithmetic `BinaryExpressionNode` (`+`, `-`, `*`, `/`), the operand ranges `L` and `R` are combined as follows:
 - `+`: $[\min L + \min R, \max L + \max R]$
 - `-`: $[\min L - \max R, \max L - \min R]$
 - `*`: The minimum and maximum are derived from all possible products of the interval endpoints.
 - `/`: If $0 \in R$'s range, the conservative range `[minValue, maxValue]` is returned; otherwise, the minimum and maximum are derived from all possible quotients of the endpoints.
 - `%`: No refinement is performed for the modulo operator; the full global range is maintained.

- For `MathCallNode` or other unhandled nodes, the conservative global range `[minValue, maxValue]` is returned.
- For boolean expressions (`&&`, `||`, or relational `BinaryExpressionNode`), a range of `[0, 1]` is returned.

If the range on either side evaluates to `null`, no refinement is possible and the function returns immediately.

Stage 2: Backward Propagation of Constraints

Based on the operator `op` and the computed ranges `[Lmin, Lmax]` and `[Rmin, Rmax]`, the function derives tighter target ranges for `L` and `R` that would satisfy the constraint. It then invokes `_refineArithExpressionRange` on each operand:

- **=== or ==:** If the ranges do not overlap ($L_{\max} < R_{\min}$ or $R_{\max} < L_{\min}$), the clause is deemed unsatisfiable; consequently, domains in both subtrees are cleared via `_wipeDomainsInSubtree`. Otherwise, the procedure computes the intersection `[max(Lmin, Rmin), min(Lmax, Rmax)]` and refines both `L` and `R` to this common range.
- **>:** The range for `L` is refined to `[Rmin + 1, Lmax]` and the range for `R` is refined to `[Rmin, Lmax - 1]`. Analogous logic is applied to the operators `<`, `>=`, and `<=`.
- **!= or !=:** If both ranges consist of the same single point, the domains are cleared. If `L` represents a single value `v`, then `v` is excluded from `R` by refining `R` to the union of intervals `[minValue, v - 1]` and `[v + 1, maxValue]`.

The `_refineArithExpressionRange` Function

This routine propagates a required numeric interval `[tMin, tMax]` back to an expression node within the AST:

- **LiteralNode:** No change is applied, as the value is fixed by definition.
- **ArrayAccessNode (el[index]):** The current domain `D` is retrieved. A new temporary domain `Dnew` is constructed and initialised to the interval:

$$[[\max(tMin, minValue)], [\min(tMax, maxValue)]]$$

Any discrete values of D falling within this interval are subsequently added. The existing domain is then updated via the intersection $D \leftarrow D \cap D_{\text{new}}$.

- **BinaryExpressionNode ($A \text{ op } B$):** The current ranges of A and B are computed. For each operator, the procedure derives the implied target range for A by "inverting" the operation. For instance, in the case of $A + B$, the target range for A is calculated as $[tMin - \max(B), tMax - \min(B)]$, with a similar logic applied to B . The function then recurses on each operand. Specific rules are enforced for multiplication and division, particularly concerning zero-inclusion tests.
- **MathCallNode or complex expressions:** Presently, no dedicated backward rules are implemented; only argument domains are refined in cases where they represent simple variables.

This forward-and-backward propagation loop continues until the `domainMap` stabilises. The final narrowed domains dramatically reduce the combinatorial search space that the TFM must explore.

6.2.3.3 Domain Representation

The `Domain` class provides a structure to represent the permissible integer values for a single profile attribute, accommodating both:

- **Continuous Intervals:** These are inclusive integer ranges (e.g. $[0, 100]$). The class automatically merges overlapping or adjacent intervals; for instance, adding the interval $[4, 8]$ to a domain already containing $[1, 5]$ results in the single interval $[1, 8]$. It is crucial to acknowledge that these intervals are defined solely by their endpoints, rather than by an array encompassing every possible value. This approach was a deliberate design decision aimed at minimising memory consumption. Consequently, the domain $[1, 4]$ is delineated by the object `{min: 1, max: 4}` instead of the explicit array `[1, 2, 3, 4]`.
- **Discrete Values:** These are specific, individual integers (e.g. 105, 110). They are integrated with the interval structure; if an interval expands to encompass a discrete value, that value is absorbed into the interval.

6.2.3.4 Domain Narrowing

The domain narrowing component of the DMS executes the iterative refinement of attribute domains, predicated on the relationships and restrictions delineated in the wff's AST, as constructed by the ASTe (Section 4.2.2). This process is crucial for reducing the search space explored by the TFM (Section 4.2.4). The procedure commences with the establishment of an initial `domainMap` by

the `createInitialDomainMap` function. For each profile index identified as a dependency in the wff (via `getAllIndexes`), this function instantiates a `Domain` object, typically initialised to span the global `minValue` and `maxValue` parameters. Consequently, this map associates each relevant attribute index with its broadest possible domain.

Subsequent refinement is orchestrated by the `narrowDomainMap` function, which iteratively applies constraints until a fixed-point is achieved; that is, until a full pass over all constraints induces no further changes to any domain, or until a domain becomes empty, signalling that the wff clause is unsatisfiable. This function systematically traverses the AST of a conjunctive wff clause, primarily targeting `BinaryExpression` nodes that represent boolean-valued constraints. These include relational comparisons (e.g. `"e1[0] > 5"`) or equalities involving arithmetic expressions (e.g. `"e1[1] + e1[2] === 20"`).

For each identified constraint, the internal `refineBooleanConstraint` function is invoked. This function begins by computing the potential arithmetic range (minimum and maximum bounds) for the left-hand and right-hand sub-expressions of the constraint. This task is performed by the `computeArithRange` helper, which recursively evaluates expressions by considering the present domains of variables (retrieved from the `domainMap`), the values of literals (whose range is the value itself), and the semantics of standard arithmetic operators (+, -, *, /, %). These operators typically combine the interval endpoints of their operands; for instance, for an expression $L + R$, the resulting range is:

$$[\min_L + \min_R, \max_L + \max_R]$$

Once these ranges, denoted L_{range} and R_{range} , are determined, `refineBooleanConstraint` applies logic specific to the relational operator within the constraint. For an equality constraint, such as $L === R$, if L_{range} and R_{range} do not overlap, the constraint is deemed unsatisfiable for the current domains. Consequently, the domains of all variables within the sub-trees of L and R are "cleared" (i.e. set to empty) by the `wipeDomainsInSubtree` procedure.

If an overlap exists, representing a common satisfiable range $[\text{common}_{\text{min}}, \text{common}_{\text{max}}]$, this range is subsequently propagated backwards to the constituent variables of both L and R using the `refineArithExpressionRange` function. For relational operators such as $L > R$, the system adjusts the bounds by, for example, setting the new target lower bound for L to $\min(R_{\text{range}}) +$

1 and the new target upper bound for R to $\max(L_{\text{range}}) - 1$. These tighter target ranges are then similarly propagated back to the variables in L and R via `refineArithExpressionRange`. Inequality constraints (`!=`) are handled by attempting to exclude specific discrete values if one side of the inequality reduces to a constant.

The function `refineArithExpressionRange` receives a target interval for an arithmetic expression and subsequently tightens the domains of its constituent variables. When the expression consists of a single variable X with a desired range $[t_{\min}, t_{\max}]$, its current domain $\text{dom}(X)$ is replaced by the intersection:

$$\text{dom}(X) \leftarrow \text{dom}(X) \cap [t_{\min}, t_{\max}].$$

If the expression is a sum $A + B$ required to reside within $[C_{\min}, C_{\max}]$, an implied range for A is computed by subtracting the current bounds of B :

$$R_A = [C_{\min} - B_{\max}, C_{\max} - B_{\min}].$$

Subsequently:

$$\text{dom}(A) \leftarrow \text{dom}(A) \cap R_A,$$

and an analogous step refines B by utilising the updated domain of A :

$$R_B = [C_{\min} - A_{\max}, C_{\max} - A_{\min}], \quad \text{dom}(B) \leftarrow \text{dom}(B) \cap R_B.$$

The repetition of these backward steps until no further domain reduction occurs produces the tightest intervals compatible with the constraint $A + B = C$. A forward-propagation step may then refine the domain of C via:

$$A + B \in [A_{\min} + B_{\min}, A_{\max} + B_{\max}].$$

To illustrate this mechanism, consider the following initial intervals for A , B , and C :

$$(i) A \in [1, 10]; \quad B \in [1, 10]; \quad C \in [1, 10].$$

A single backward pass, using the target C , yields the following refinement:

$$(ii) R_A = [1 - B_{\max}, 10 - B_{\min}] = [1 - 10, 10 - 1] = [-9, 9],$$

consequently,

$$A \leftarrow [1, 10] \cap [-9, 9] = [1, 9].$$

and $[-9, 9]$

Utilising the refined domain $A \in [1, 9]$ then produces:

$$R_B = [1 - A_{\max}, 10 - A_{\min}] = [1 - 9, 10 - 1] = [-8, 9],$$

and hence:

$$B \leftarrow [1, 10] \cap [-8, 9] = [1, 9].$$

Following this backward iteration, the domains are established as:

$$A \in [1, 9], \quad B \in [1, 9], \quad C \in [1, 10].$$

A subsequent forward step refines C as follows:

$$A + B \in [1 + 1, 9 + 9] = [2, 18], \quad C \leftarrow [1, 10] \cap [2, 18] = [2, 10].$$

Repeating the backward pass with the updated range $C \in [2, 10]$ yields no further modifications.

The final, converged domains are:

$$A \in [1, 9], \quad B \in [1, 9], \quad C \in [2, 10].$$

It is observed that these tightest intervals now satisfy the constraint $A + B = C$.

This iterative refinement primarily achieves a form of bounds consistency, ensuring that the interval endpoints for each variable's domain remain consistent with the applied constraints. While this mechanism significantly prunes the domains prior to the TFM's combinatorial search, the current implementation does not achieve more stringent forms of consistency, such as full arc consistency. For example, consider the variables `e1[0]` and `e1[1]`, both with the domain $[2, 5]$,

and the constraint "`e1[0] * e1[1] === 10`". The propagation of current bounds would refine both domains to $[2, 5]$, as $10/5 = 2$ and $10/2 = 5$. However, specific values such as `e1[0] = 3` or `e1[0] = 4` possess no corresponding integer support within the domain of `e1[1]` to satisfy the product, yet they persist within the established bounds.

6.2.4 Tuple Finder Module

6.2.4.1 Initialisation, Pre-emptive Checks, and wff Decomposition

The primary entry point for the TFM is the function `findTuplesWithConstraints`. This function accepts the following inputs:

- An initial profile P_{init} , or a set of profiles Γ_{init} .
- A wff string α specifying the revision condition.
- Global integer bounds `minValue` and `maxValue` for each profile attribute.

Before the full search machinery is invoked, the TFM performs several preliminary steps to address trivial cases, normalise inputs, and decompose the wff into its constituent conjunctive components.

Handling of Trivial wff

If the wff is `null`, an empty string, or the literal strings "`true`" or "`false`", the function immediately returns the original profile (or set of profiles) with a distance of zero. Conceptually:

```
// If \ac{wff} is trivial, return P_init with minDistance=0
if (!\ac{wff} || \ac{wff} === "true" || \ac{wff} === "false") {
  return {
    minDistance: 0,
    tuplesWithMinDistance:
      Array.isArray(P_init) ? P_init :
      [P_init]
  };
}
```

This approach adheres to the semantic postulates for tautologies and contradictions, thereby precluding the need for further computation.

Early Syntactic Validation and Predicate Generation

For non-trivial wff, the TFM attempts a lightweight compilation utilising the JavaScript `Function` constructor:

```
// Attempt to construct a rapid predicate checkFn(el) { return \ac{wff}; }
let checkFn;
try {
  checkFn = new Function("el", 'return ${\ac{wff}};');
} catch (e) {
  // Malformed \ac{wff}: treated as unsatisfiable
  return {
    minDistance: 0,
    tuplesWithMinDistance: Array.isArray(P_init) ?
      P_init : [P_init]
  };
}
```

Should this process fail due to a gross syntax error, the wff is deemed unsatisfiable, and the original profiles are returned.

Pre-emptive Satisfaction Check

Utilising the `checkFn`, the TFM verifies whether any of the input profiles already satisfy the wff:

- If P_{init} is a single profile, the system evaluates `checkFn( $P_{\text{init}}$ )`.
- If I_{init} is an array of profiles, each $P_i \in I_{\text{init}}$ is tested, and those for which `checkFn( $P_i$ )` returns `true` are collected.

If any initial profile satisfies the wff, the TFM returns the corresponding profiles with a `minDistance` of 0.

Normalisation of the Target Profile

In the event that no initial solution is identified, the TFM subsequently constructs a single `targetArray` to facilitate distance calculations:

- For a single P_{init} , the `targetArray` is set to P_{init} .

- For a set Γ_{init} , the element-wise arithmetic mean (rounded to the nearest integer) is computed via the `calculateTupleAverages` function. The resulting tuple is designated as the `targetArray`.

Decomposition into Conjunctive Clauses

The profile language $\mathcal{L}_p$ requires the wff to be in DNF, as specified in Definition 9. To facilitate this, the TFM partitions the wff using the logical OR operator "||" to obtain an array of conjunctive clauses:

```
// Decompose \ac{wff} = C1 V C2 V ... V Cm
const orParts = \ac{wff}
  .split("||")
  .map(part => part.trim())
  .filter(Boolean);
```

Each clause C_j within `orParts` is subsequently processed independently by the internal `findTuplesForConjunction` routine, utilising the `targetArray` for Manhattan distance computations. Global variables, specifically `globalMinDistance` and `globalAssignments`, are employed to track the optimal solutions identified across all clauses.

Having completed these initialisation, validation, and decomposition stages, the TFM is prepared to execute the full AST-based domain narrowing and combinatorial search for each conjunctive component of the original wff.

6.2.4.2 Orchestration of ASTe and DMS Services, and wff Decomposition

Following the completion of preliminary checks and input normalisation, the TFM proceeds to the primary processing phase of the wff. This stage integrates services from both the ASTe and the DMS, while decomposing the wff into its constituent conjunctive clauses to mitigate search complexity.

Decomposition of DNF wff

The profile language L_p requires the wff to be expressed in DNF:

$$\alpha = C_1 \vee C_2 \vee \dots \vee C_m,$$

where each conjunctive clause is defined as $C_j = (\ell_1 \wedge \ell_2 \wedge \dots \wedge \ell_k)$.

The TFM partitions the input string α at each occurrence of the OR operator "`||`", subsequently trimming whitespace and filtering out empty substrings to generate an array of clause strings designated as `orParts`. Each clause is then processed independently through the `_findTuplesForConjunction` helper routine. Finally, the system aggregates the solutions that achieve the global minimum Manhattan distance across all evaluated clauses.

Processing a Single Conjunctive Clause

The `_findTuplesForConjunction` function is invoked for each clause string C_j . The procedure encompasses the following primary stages:

1. **AST Construction.** The function invokes `customAST = ASTe.parseCondition( $C_j$ )`. Should the parsing process fail, the clause is deemed unsatisfiable.
2. **AST Validation.** The routine invokes `res = ASTe.validateAST(customAST, targetArray.length)`. If the result is `false`, the processing of the clause is aborted.
3. **Dependency Identification.** The function invokes `deps = ASTe.getAllIndexes(customAST)`, which yields the set of profile indices explicitly referenced by C_j .
4. **Predicate Generation.** The procedure invokes `evalFn = ASTe.buildEvalFunctionFromAST(customAST)`, resulting in an executable function `evalFn(assignment, target)` designed to test the satisfaction of C_j .
5. **Initial Domain Map.** The routine initialises the domain map:

$$\text{domainMap} = \text{DMS.createInitialDomainMap}(\text{deps}, \text{minVal}, \text{maxVal}, n)$$

where n represents `targetArray.length`. Each dependent index is assigned a `Domain` instance spanning the interval $[\text{minVal}, \text{maxVal}]$.

6. **Domain Narrowing.** The function invokes:

$$\text{DMS.narrowDomainMap}(\text{customAST}, \text{domainMap}, \text{minVal}, \text{maxVal})$$

This stage iteratively propagates constraints until either a fixed point is achieved or an empty domain is encountered, indicating unsatisfiability.

7. Pre-assignment and Search Preparation. The following variables are initialised:

`assignment` as an empty object, `dist` as 0, and `toSearch` as an empty array. For each index $i \in \text{deps}$:

- If `domainMap.get(i)` is a singleton set $\{v\}$, the value v is added to `assignment[i]`, and the corresponding Manhattan distance $|v - \text{targetArray}[i]|$ is added to `dist`.
- In all other cases, i is appended to the `toSearch` array.

Finally, the recursive search is initiated:

`TFM.Explore(0, dist, assignment, toSearch, domainMap, targetArray, evalFn)`.

6.2.4.3 Execution of Distance-Guided Constrained Combinatorial Search

Subsequent to the preparatory phases involving wff decomposition, AST generation and validation by the ASTe, and the substantial reduction of attribute domains by the DMS, the TFM initiates its core computational task: a distance-guided, constrained combinatorial search for satisfying profile tuples. This search is encapsulated within a recursive procedure, internally designated as **Explore**, which systematically investigates potential value assignments for the profile attributes identified as requiring further exploration.

The **Explore** procedure is invoked by `_findTuplesForConjunction` with several key parameters: an index (`depIdx`) indicating the current attribute within the ordered list `attributesToSearch` (comprising those attributes whose domains were not reduced to a singleton value by the DMS); the cumulative Manhattan distance (`curDist`) of the assignments made thus far from the target profile P_{target} ; the current partial `assignment` object; the `attributesToSearch` list; the `domainMap` containing the narrowed domains for all dependent attributes; the reference P_{target} ; and the dynamically generated predicate function `EvalFnCj` corresponding to the conjunctive wff clause C_j being processed.

Recursive Structure and Base Case

The **Explore** procedure operates on a DFS paradigm. The recursion deepens as values are assigned to successive attributes within the `attributesToSearch` list. The base case for the recursion is reached when `depIdx` equals `attributesToSearch.length`. This condition signifies that all attributes requiring explicit search have been assigned a value within the current recursive path, thereby forming a complete candidate assignment for the dependent attributes.

At this juncture, the `currentAssignment`—which incorporates any pre-assigned values from attributes with singleton domains—is evaluated against the wff clause C_j by invoking:

$$\text{EvalFn}_{C_j}(\text{currentAssignment}, P_{\text{target}}).$$

Should EvalFn_{C_j} return `true`, indicating that the `currentAssignment` satisfies the clause, the TFM proceeds to assess this solution within the context of the globally optimal solutions identified thus far. The `curDist`, representing the Manhattan distance of this complete dependent assignment from the corresponding elements of P_{target} , is compared against a globally maintained `globalMinDistance` variable.

- **Optimal Solution Discovery:** If `curDist < globalMinDistance`, the current assignment constitutes a strictly superior solution. Consequently, `globalMinDistance` is updated to the value of `curDist`, and the `globalAssignments` list is replaced with a deep copy of the current assignment. Furthermore, a stringified representation of this assignment is stored within a global set, `globalSeenSolutions`, to facilitate the tracking of unique solutions.
- **Equally Optimal Solution Discovery:** If `curDist = globalMinDistance`, the current assignment represents an equally optimal solution. To preclude redundancy, its stringified representation is verified against `globalSeenSolutions`. Should it represent a novel assignment not previously encountered at this minimal distance, a deep copy of the current assignment is appended to `globalAssignments`, and its stringified form is added to `globalSeenSolutions`.

Following this evaluation, the recursive procedure terminates for the current path and backtracks.

Recursive Step: Value Selection and Pruning

If the base case is not satisfied (i.e. where `depIdx < attributesToSearch.length`), the `Explore` procedure identifies the next attribute to be assigned:

$$\text{idxToAssign} = \text{attributesToSearch}[\text{depIdx}].$$

The procedure retrieves the narrowed domain $D_{\text{idxToAssign}}$ for this attribute from the `domainMap` and subsequently iterates through all permissible values $v \in D_{\text{idxToAssign}}$. This iteration encom-

passes values from both the continuous intervals and the discrete value sets defined within the domain.

A heuristic is employed when traversing values within continuous intervals. Rather than a simple linear scan from the minimum value to the maximum, the search commences with the integer closest to $P_{\text{target}}[\text{idxToAssign}]$ that resides within the interval. The procedure then explores values by expanding outwards from this initial point, alternating between decrementing and incrementing the offset.

This strategy is designed to prioritise the exploration of values that are, *prima facie*, more likely to yield a minimal Manhattan distance. This potentially facilitates the earlier discovery of optimal solutions, thereby enhancing the efficacy of distance-based pruning (discussed further in Section 4.4.3).

For each candidate value v considered for idxToAssign :

1. **Distance Calculation:** The Manhattan distance contribution for this specific assignment is calculated as:

$$|v - P_{\text{target}}[\text{idxToAssign}]|$$

This value is added to `curDist` to obtain `newDist`, representing the projected distance of the partial assignment should v be selected.

2. **Distance-Based Pruning:** If `newDist > globalMinDistance`, it is implied that any complete assignment derived from this partial path will necessarily possess a Manhattan distance strictly greater than the optimal solution identified thus far. Consequently, this search branch is pruned: the iteration for the current value v is abandoned, and the procedure backtracks. Due to the outward-expanding heuristic, subsequent values within the same interval exploration that are further from the target are likewise bypassed.
3. **Tentative Assignment:** If `newDist ≤ globalMinDistance`, the value v is tentatively assigned to `currentAssignment[idxToAssign]`.
4. **Recursive Propagation:** A recursive call to `Explore` is subsequently performed for the next attribute in the `attributesToSearch` list (i.e. with `depIdx + 1`), passing the updated `newDist` and the modified `currentAssignment`.

After the recursive call returns—signifying that all possibilities for subsequent attributes, given the current assignment to `idxToAssign`, have been exhausted—the assignment `currentAssignment[idxToAssign]` is typically implicitly "undone" as the loop proceeds to the next candidate value. This systematic exploration, enhanced by the synergy of DMS domain narrowing and TFM distance-based pruning, enables the TFM to navigate the solution space with high efficiency.

The `ValuesFromDomain` abstraction, as delineated in Algorithm 3, represents the iteration logic that integrates interval-based exploration (commencing in the proximity of `originalValueAtIndex` and expanding outwards) with the iteration over discrete values. The pruning condition (`newDist > globalMinDistance`) is applied rigorously throughout this process. Upon the completion of this recursive search across all conjunctive clauses, the TFM will have populated `globalAssignments` with all unique partial assignments that satisfy at least one clause and achieve the overall `globalMinDistance`. These assignments are subsequently utilised for the final reconstruction of complete profiles, as detailed in Section 4.2.4.

6.2.4.4 Consolidation of Results and Profile Reconstruction

Following the completion of the preparatory phases involving wff analysis by the ASTe and domain narrowing by the DMS, and subsequent to the execution of the distance-guided constrained combinatorial search via the `Explore` procedure for each constituent conjunctive clause, the TFM performs the final operation of consolidating the gathered results and reconstructing complete profile tuples. This phase ensures that the TFM returns a coherent and comprehensive set of optimal solutions to the invoking PDM.

A crucial stage in this final phase involves addressing scenarios in which no solution is identified. If, following the processing of all conjunctive clauses, the `globalMinDistance` remains at its initialised value of ∞ , it signifies that no assignment satisfying any component of the wff could be identified within the specified constraints and value ranges. In this particular implementation, rather than signalling an error or returning an empty set, the TFM returns the original input profile(s) ($P_{\text{initOriginal}}$) with an associated `minDistance` of 0.

Conversely, if solutions are identified (that is, `globalMinDistance` is finite), the `globalAssignments` array contains one or more partial assignments. Each such partial assignment is an object mapping only the dependent attribute indices—those explicitly appearing in the

wff—to their assigned values. To generate complete profile tuples suitable for output, each partial assignment must be transformed. This transformation is executed by the internal helper function `reconstructFullTuple(assignment, target)`.

The `reconstructFullTuple` function operates as follows:

1. **Input Parameters:** The function accepts a `partialAssignment` object (e.g. `{0: val0, 2: val2}`) and the reference `target` profile. This reference is the `targetArray` established previously, consisting of either the original single input profile or the mean of an input set.
2. **Array Initialisation:** A new array, designated `full`, is created with the same length as the `target` profile, typically utilising an `Int32Array` for numerical profiles.
3. **Base Population:** The `full` array is initially populated by creating a deep copy of the `target` profile. This ensures that all attribute positions are initialised with the corresponding values from the reference profile.
4. **Attribute Overwriting:** The function iterates through each key-value pair in the `partialAssignment` object. For each pair, denoted `index: assignedValue`, it updates the element at the specified `index` in the `full` array to the `assignedValue`.
5. **Resultant Tuple:** The final `full` array is returned. This represents a complete profile tuple where dependent attributes possess their optimally assigned values, while non-dependent attributes retain their values from the reference `target` profile.

For example, if $P_{\text{targetArray}} = [5, 5, 5]$ and a `partialAssignment` is given as `{0: 1, 1: 3}`, the `reconstructFullTuple` function produces the complete profile `[1, 3, 5]`. The attribute at index 2, being absent from the assignment, retains its original value from $P_{\text{targetArray}}$.

The `findTuplesWithConstraints` function applies `reconstructFullTuple` to every partial assignment stored within `globalAssignments`, thereby generating a list of complete, optimal profile tuples. Finally, it returns an object containing the determined `globalMinDistance` and this list of `fullTuples` (designated as `tuplesWithMinDistance`) to the PDM. This structured output provides the PDM with all necessary information to proceed with model-specific post-processing or to directly present the results of the profile dynamics operation.

The core logic for processing each conjunctive component of the wff is situated within the internal helper function `findTuplesForConjunction`, as detailed in Algorithm 2.

The **Explore** procedure (Algorithm 3) subsequently performs a recursive DFS:

The operational stages of the TFM are as follows:

1. **Initialisation and Pre-emptive Checks (Alg. 4, lines 1–8):** Trivial wff are handled immediately. For non-trivial expressions, the module utilises an ASTe-generated `checkFn` to verify whether any initial profiles already satisfy the wff. If satisfaction is confirmed, these profiles are returned with a distance of zero. In cases involving multiple initial profiles (specifically relevant for Model 3), an average target profile, designated as `targetArray`, is computed.
2. **Disjunction Handling (Alg. 4, lines 11–13):** The wff is partitioned by the `"||"` operator into constituent conjunctive clauses. Each clause is subsequently processed by the `findTuplesForConjunction` routine (Alg. 2).
3. **AST-Driven Preparation (Alg. 2, lines 1–3):** For each conjunctive clause, the ASTe performs parsing, validates the resulting AST, and extracts dependencies (the indices of the profile elements involved in the constraint).
4. **Domain Initialisation and Narrowing (Alg. 2, lines 4–7):** The DMS generates an initial `domainMap` for the identified dependencies and subsequently narrows these domains using the AST. Should any domain become empty, the clause is deemed unsatisfiable.
5. **Constrained Combinatorial Search (Alg. 2, lines 8–15, invoking Alg. 3):**
 - Indices with domains narrowed to a singleton value are pre-assigned to the partial solution.
 - The **Explore** function (Alg. 3) recursively assigns values to the remaining indices from their respective narrowed domains.
 - The Manhattan distance (`curDist`) is calculated incrementally. If the value of `curDist` exceeds the `globalMinDistance`, the search branch is pruned (Alg. 3, line 10).
 - Completed assignments are validated using the `evalFn` (provided by the ASTe). Valid assignments that achieve or match the `globalMinDistance` are stored, with uniqueness ensured via stringified set tracking.

6. **Result Aggregation and Reconstruction (Alg. 4, lines 15–20):** Following the processing of all clauses, any identified partial assignments (affecting only dependent indices) are converted into complete profiles using `reconstructFullTuple`. The overall `globalMinDistance` and the set of unique `fullTuples` are returned. If no solutions are identified, the original input profile(s) is returned.

6.3 Benchmark setup

Setup. Let $\mathbf{el} = (el_0, \dots, el_{n-1})$ be a profile. Given a condition C_k with feasible set $F_k = \{\mathbf{el} : \varphi_k(\mathbf{el})\}$ and a starting profile $\mathbf{x}$, the target is $\mathbf{y}^* \in F_k$ minimising the Manhattan distance $d_1(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|$. Arithmetic is over $\mathbb{Z}$ unless division appears; in equalities with “/” interpret over $\mathbb{R}$ with non-zero denominators.

C1: `true` $\top$. *Test:* Baseline unconstrained search. *Analogue:* No constraints.

C2: `e1[1]==5 || e1[2]==5 || e1[2]>8` $(el_1 = 5) \vee (el_2 = 5) \vee (el_2 > 8)$. *Test:* Disjunction and thresholds. *Analogue:* Any one of several compliance targets suffices.

C3: `e1[0]>e1[7]` $el_0 > el_7$. *Test:* Pairwise ordering across distant indices. *Analogue:* A lead metric must exceed a benchmark.

C4: `e1[0]>e1[1] && e1[1]>e1[2]` $el_0 > el_1 > el_2$. *Test:* Strict monotone chain. *Analogue:* Ranked priorities.

C5: `e1[0]!=e1[1]` $el_0 \neq el_1$. *Test:* Disequality. *Analogue:* Distinct IDs.

C6: `e1[0]+e1[1]==e1[2]` $el_0 + el_1 = el_2$. *Test:* Linear conservation. *Analogue:* Inputs sum to output.

C7: `e1[0]==3 || e1[0]==7` $el_0 \in \{3, 7\}$. *Test:* Categorical membership. *Analogue:* Allowed codes.

C8: `e1[0]*2==e1[1]+e1[2]` $2el_0 = el_1 + el_2$. *Test:* Balanced linear relation. *Analogue:* Weighted budget split.

C9: `e1[0]<e1[1] && e1[1]<e1[2] && e1[2]<e1[3]` $el_0 < el_1 < el_2 < el_3$. *Test:* Strict increasing sequence. *Analogue:* Growing Key Performance Indicator (KPI) series.

C10: `e1[0]==e1[1] && e1[1]==e1[2]` $el_0 = el_1 = el_2$. *Test:* Replicate consistency. *Analogue:* Redundant copies match.

- C11:** $e1[0]+1===e1[1] \ \&\& \ e1[1]*2===e1[2]$ $el_1 = el_0 + 1, \ el_2 = 2el_1$. *Test:* Mixed affine–multiplicative chain. *Analogue:* Increment then scale.
- C12:** $e1[0]-1===e1[1]$ $el_1 = el_0 - 1$. *Test:* Successor (down). *Analogue:* Countdown step.
- C13:** $e1[0]+1===e1[1]$ $el_1 = el_0 + 1$. *Test:* Successor (up). *Analogue:* Next period value.
- C14:** $(e1[0]+e1[1])*e1[2]===20$ $(el_0 + el_1)el_2 = 20$. *Test:* Multiplicative coupling to constant. *Analogue:* Rate×capacity equals fixed quota.
- C15:** $(e1[0]>e1[1] \ \&\& \ e1[1]>e1[2]) \ || \ (e1[0]<e1[1] \ \&\& \ e1[1]<e1[2])$ strict trend up or down. *Test:* Monotonicity with unknown sign. *Analogue:* Detect trend.
- C16:** $e1[0]+e1[1]===8$ $el_0 + el_1 = 8$. *Test:* Targeted sum. *Analogue:* Simple knapsack.
- C17:** $e1[0]+e1[1]===e1[2]*e1[3]$ $el_0 + el_1 = el_2el_3$. *Test:* Bilinear equality. *Analogue:* Demand equals capacity product.
- C18:** $(e1[0]===4 \ \&\& \ e1[1]===4) \ || \ (e1[1]===6 \ \&\& \ e1[2]===6)$. *Test:* Disjunctive exact patterns. *Analogue:* Alternative fallback configurations.
- C19:** $e1[0]+e1[1]===e1[2] \ \&\& \ e1[2]+e1[3]===10$ chain sums. *Test:* Coupled linear constraints. *Analogue:* Composition with cap.
- C20:** $e1[0]*2===e1[1]+1 \ \&\& \ e1[1]-1===e1[2]*2$ $2el_0 = el_1 + 1, \ el_1 - 1 = 2el_2$. *Test:* Linked affine constraints. *Analogue:* Offset scaling across stages.
- C21:** $e1[0]>7 \ \&\& \ e1[3]===6$ $el_0 > 7, \ el_3 = 6$. *Test:* Threshold + fixed anchor. *Analogue:* Eligibility plus fixed setting.
- C22:** $e1[0]*e1[0]===e1[1]$ $el_1 = el_0^2$. *Test:* Quadratic relation. *Analogue:* Area from side.
- C23:** $e1[1]-e1[0]===e1[2]-e1[1]$ equal first differences. *Test:* 3-term arithmetic progression. *Analogue:* Constant step trend.
- C24:** $e1[1]/e1[0]===e1[2]/e1[1]$ equal ratios. *Test:* 3-term geometric progression. *Analogue:* Constant growth factor.
- C25:** $e1[0]+e1[1]===e1[0]*e1[1]$ $(el_0-1)(el_1-1) = 1$. *Test:* Nonlinear curve. *Analogue:* Break-even locus.

- C26:** $e_1[0] + e_1[1] > e_1[2] \ \&\& \ e_1[1] + e_1[2] > e_1[0] \ \&\& \ e_1[0] + e_1[2] > e_1[1]$ triangle inequalities. *Test:* Feasible triangle sides. *Analogue:* Metric consistency.
- C27:** $e_1[0] < e_1[1] \ \&\& \ e_1[1] > e_1[2] \ \&\& \ e_1[2] < e_1[3]$ up-peak-up. *Test:* Local maximum pattern. *Analogue:* Spike detection.
- C28:** $e_1[0] * e_1[1] * e_1[2] \leq 24$ bounded product. *Test:* Multiplicative capacity. *Analogue:* Volume or throughput cap.
- C29:** $(e_1[0] + e_1[1] + e_1[2]) / 3 \geq 4$ average ≥ 4 . *Test:* Mean threshold. *Analogue:* KPI minimum.
- C30:** $e_1[0] \neq e_1[1] \ \&\& \ e_1[1] \neq e_1[2] \ \&\& \ e_1[2] \neq e_1[3]$ all-different on 4. *Test:* Pairwise distinctness. *Analogue:* Unique assignments.
- C31:** $e_1[0] \leq e_1[1] \ \&\& \ e_1[1] \leq e_1[2] \ \&\& \ e_1[2] > e_1[0]$ non-decreasing with strict spread. *Test:* Sortedness with variability. *Analogue:* Ties allowed but not all equal.
- C32:** $(e_1[0] + e_1[1]) * e_1[2] == e_1[0] * (e_1[1] + e_1[2])$ simplifies to $el_1(el_2 - el_0) = 0$. *Test:* Hidden disjunction ($el_1 = 0$ or $el_2 = el_0$). *Analogue:* Either a null weight or equality of endpoints.
- C33:** $2 \leq e_1[0] \leq 4 \ \&\& \ 2 \leq e_1[1] \leq 4$. *Test:* Box constraints. *Analogue:* Allowed operating ranges.
- C34:** $e_1[1] == e_1[0] * 2$ $el_1 = 2el_0$. *Test:* Linear scaling. *Analogue:* Doubling rule.
- C35:** $e_1[1] == (e_1[0] + e_1[2]) / 2$ midpoint. *Test:* Centring. *Analogue:* Rolling mean.
- C36:** $e_1[1] == e_1[0] + 1 \ \&\& \ e_1[2] == e_1[1] + 1 \ \&\& \ e_1[3] == e_1[2] + 1$ four consecutive integers. *Test:* Fixed-step AP on 4. *Analogue:* Sequential slots.
- C37:** $e_1[0] * e_1[0] + e_1[1] * e_1[1] == e_1[2] * e_1[2]$ $el_0^2 + el_1^2 = el_2^2$. *Test:* Pythagorean relation. *Analogue:* Right-triangle sides.
- C38:** $e_1[0] + e_1[1] == 10$ sum to target. *Test:* Knapsack-like. *Analogue:* Budget split to fixed total.
- C39:** $e_1[0] + e_1[1] == e_1[2]$ duplicate of C6/C42 form. *Test:* Additive composition. *Analogue:* Aggregation.
- C40:** $(e_1[0] == 3 \ \&\& \ e_1[1] == 4) \ || \ (e_1[1] == 5 \ \&\& \ e_1[2] == 6) \ || \ (e_1[0] == 7 \ \&\& \ e_1[2] == 8)$. *Test:* Multi-branch exact matches. *Analogue:* Alternative policy bundles.
- C41:** $(e_1[0] + e_1[1] + e_1[2] + e_1[3]) / 4 == 5$ average = 5 (sum = 20). *Test:* Exact mean. *Analogue:* Team score target.

- C42:** $e1[0]+e1[1]==e1[2]$ as C39. *Test:* Duplicate for caching checks. *Analogue:* Same as C39.
- C43:** $e1[1]==e1[0]*e1[0] \ \&\& \ e1[2]==e1[1]*e1[1]$ $el_1 = el_0^2$, $el_2 = el_0^4$. *Test:* Iterated squaring. *Analogue:* Compounding growth.
- C44:** $e1[0]/e1[1]==e1[2]$ ratio. *Test:* Division equality (domain guard). *Analogue:* Efficiency metric.
- C45:** $e1[1]==e1[0]*2 \ \&\& \ e1[2]==e1[1]*2$ $el_1 = 2el_0$, $el_2 = 4el_0$. *Test:* Geometric scaling chain. *Analogue:* Doubling per stage.
- C46:** $e1[0]<4 \ \&\& \ e1[1]==5 \ \&\& \ e1[2]>6$ mixed bounds with fixed middle. *Test:* Conjunction of heterogenous constraints. *Analogue:* Gate conditions.
- C47:** $(e1[0]+e1[1])*2==e1[2]$ $el_2 = 2(el_0+el_1)$. *Test:* Aggregation then scale. *Analogue:* Total cost to bill.
- C48:** $e1[0]-e1[1]==e1[1]-e1[2]$ constant second difference = 0. *Test:* Linear trend over 3. *Analogue:* Uniform acceleration = 0.
- C49:** $e1[0]==e1[1] \ \&\& \ e1[2]==e1[3] \ \&\& \ e1[0]!=e1[2]$ two equal pairs, pairs differ. *Test:* Partition into two classes. *Analogue:* Two cohorts.
- C50:** $6 \leq e1[0]+e1[1]+e1[2] \leq 12$ bounded sum. *Test:* Band constraint. *Analogue:* Capacity window.
- C51:** $e1[0]/e1[1]==e1[2]$ duplicate of C44. *Test:* Repeat division case. *Analogue:* Same as C44.
- C52:** $e1[1]-e1[0]==e1[2]-e1[1] \ \&\& \ e1[2]-e1[1]==e1[3]-e1[2]$ 4-term AP. *Test:* Constant step over 4. *Analogue:* Linear trajectory.
- C53:** $e1[1]==4 \ \&\& \ e1[2]-e1[1]==e1[3]-e1[2]*50000000$ $el_1 = 4$, $el_2 - el_1 = el_3 - 50,000,000 el_2$. *Test:* Extreme coefficients and precedence. *Analogue:* Outlier stress.
- C54:** $e1[13234234]==4 \ \&\& \ e1[2]-e1[1]==e1[3]-e1[2]$. *Test:* Out-of-range index plus valid AP. *Analogue:* Schema mismatch detection.
- C55:** bananas. *Test:* Parser rejection of non-boolean token. *Analogue:* Input validation.
- C56:** $e1[0]+e1[1]>e1[2]+e1[3]$ compare two sums. *Test:* Linear inequality. *Analogue:* Team A beats Team B.

C57: $e_1[0] + e_1[1] == 5 \ \&\& \ e_1[2] > 2 \ \&\& \ e_1[3] + e_1[4] < 15 \ \&\& \ e_1[4] > 2$. *Test:* Multi-constraint across five vars. *Analogue:* Composite policy with min/max and target.

C58: $e_1[0] - e_1[1] == 0$ $el_0 = el_1$. *Test:* Equality by difference. *Analogue:* Tie constraint.

6.4 NeuroAIReh@b Database

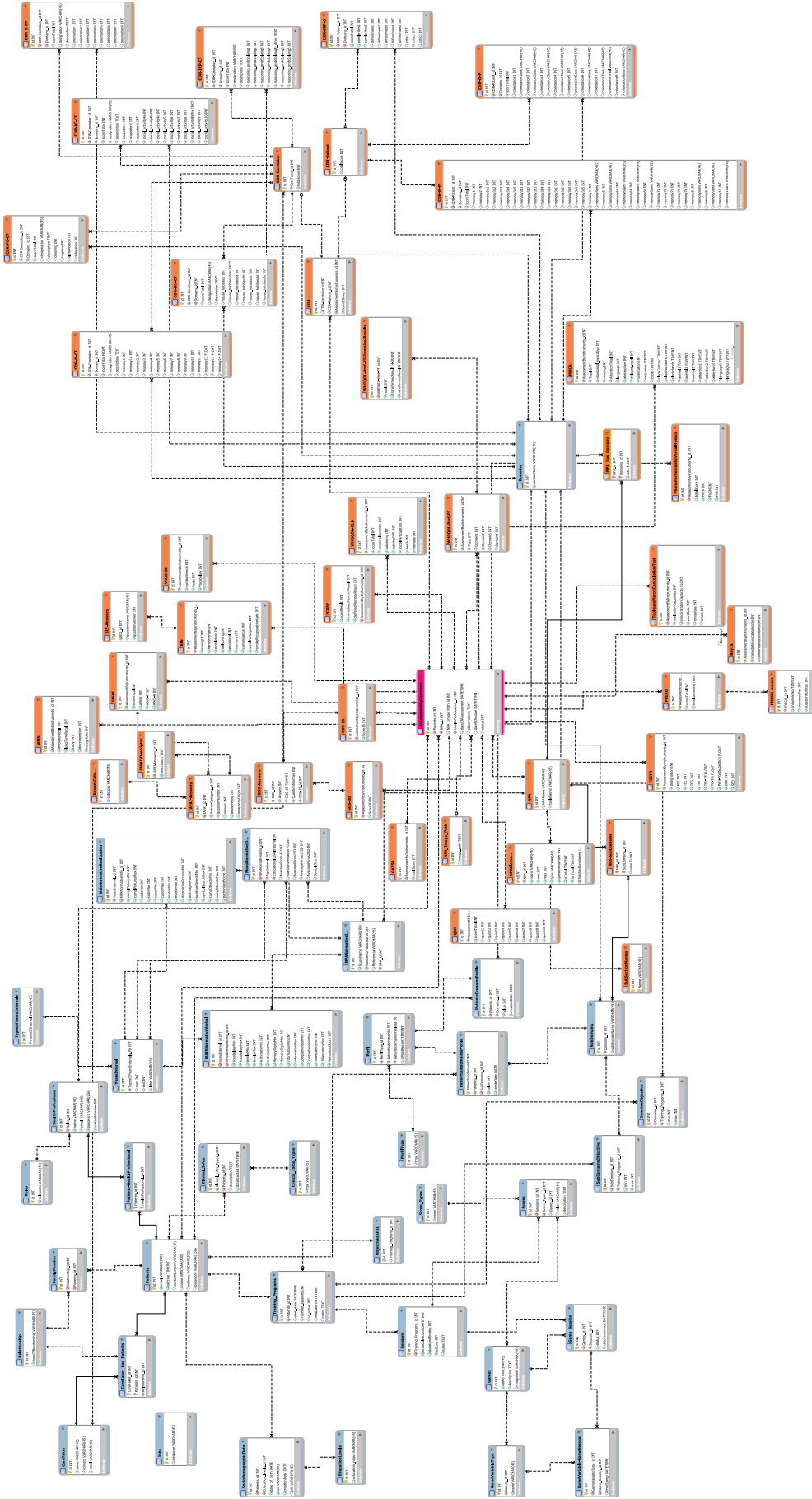


Figure 13: Entity-relationship schema of the NeuroAIR@b relational database. Rectangles represent database tables and connecting lines represent primary-key/foreign-key relationships between them. The schema integrates the data required by the four principal framework components—Patient Profiling, Training Selection, Training Session, and System Calibration—including SDI, raw and normalised neuropsychological assessment results, normative data, assessed and predicted cognitive profiles, training objectives, CTT metadata and parameters, and session-level performance records. This relational structure supports data integrity, longitudinal tracking of patient evolution, profile construction, task parameterisation, and comparison between predicted and reassessed profiles.

References

- [1] P. Gärdenfors, *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. Cambridge: The MIT Press, 1988.
- [2] E. Fermé and S. O. Hansson, *Belief Change: Introduction and Overview*, 1st ed., ser. Springer-Briefs in Intelligent Systems. Springer Cham, 2018, eBook ISBN: 978-3-319-60535-7.
- [3] C. E. Alchourrón, P. Gärdenfors, and D. Makinson, “On the logic of theory change: Partial meet contraction and revision functions,” *Journal of Symbolic Logic*, vol. 50, no. 2, p. 510–530, 1985, <https://doi.org/10.2307/2274239>.
- [4] J. Doyle, “A truth maintenance system,” in *Readings in Artificial Intelligence*, B. L. Webber and N. J. Nilsson, Eds. Morgan Kaufmann, 1979, vol. 12, pp. 496–516. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780934613033500398>
- [5] G. Harman, “Change in view: Principles of reasoning,” *Cognitive Science*, vol. 10, no. 1, pp. 63–81, 1986.
- [6] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
- [7] S. Chopra, R. Parikh, and R. Wassermann, “Approximate belief revision,” *Logic Journal of the IGPL*, vol. 9, no. 6, pp. 755–768, 11 2001. [Online]. Available: <https://doi.org/10.1093/jigpal/9.6.755>
- [8] R. Wassermann, “Resource bounded belief revision,” *Erkenntnis (1975-)*, vol. 50, no. 2/3, pp. 429–446, 1999. [Online]. Available: <http://www.jstor.org/stable/20012927>
- [9] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [10] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2010.
- [11] R. Davis and D. Lenat, *Knowledge-Based Systems in Artificial Intelligence: 2 Case Studies*. McGraw-Hill, Inc., 1982.

- [12] P. Stone and M. Veloso, “Multiagent systems: A survey from a machine learning perspective,” *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.
- [13] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, “Power to the people: The role of humans in interactive machine learning,” *AI Magazine*, vol. 35, no. 4, pp. 105–120, 2014.
- [14] G. Flouris and D. Plexousakis, “Handling ontology change: Survey and proposal for a future research direction,” *Institute of Computer Science, Forth. Greece, Technical Report TR-362 FORTH-ICS*, 2005.
- [15] —, “Bridging ontology evolution and belief change,” in *Advances in Artificial Intelligence*, G. Antoniou, G. Potamias, C. Spyropoulos, and D. Plexousakis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 486–489.
- [16] G. Konstantinidis, G. Flouris, G. Antoniou, and V. Christophides, “On rdf/s ontology evolution,” in *Semantic Web, Ontologies and Databases*, V. Christophides, M. Collard, and C. Gutierrez, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21–42.
- [17] G. Flouris, Z. Huang, J. Pan, D. Plexousakis, and H. Wache, “Inconsistencies, negations and changes in ontologies,” in *Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference, AAAI-06/IAAI-06*, vol. 2, Nov. 2006, pp. 1295–1300, 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference, AAAI-06/IAAI-06 ; Conference date: 16-07-2006 Through 20-07-2006.
- [18] G. Flouris, G. Konstantinidis, G. Antoniou, and V. Christophides, “Formal foundations for rdf/s kb evolution,” *Knowledge and Information Systems*, vol. 35, no. 1, pp. 153–191, Apr 2013. [Online]. Available: <https://doi.org/10.1007/s10115-012-0500-2>
- [19] F. Baader and R. Wassermann, “Contractions Based on Optimal Repairs,” in *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning*, 8 2024, pp. 94–105. [Online]. Available: <https://doi.org/10.24963/kr.2024/9>
- [20] D. A. d. Souza, “Bridging belief revision and ontology repair: moving closer to optimal repairs,” Ph.D. dissertation, Universidade de São Paulo, 2024.

- [21] E. Fermé, M. Garapa, and M. D. L. Reis., *Non-prioritized belief change*, ser. Artificial Intelligence: Foundations, Theory, and Algorithms Series. Springer Nature, 2025.
- [22] B. Nebel, “How hard is it to revise a belief base?” in *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, D. Dubois and H. Prade, Eds. Kluwer Academic Publishers, 1998, vol. 3, ch. 3, pp. 77–145.
- [23] M.-A. Williams, “Anytime belief revision,” *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pp. 74–79, 1997.
- [24] J. Delgrande, T. Schaub, H. Tompits, and S. Woltran, “A model-theoretic approach to belief change in answer set programming,” *ACM Trans. Comput. Logic*, vol. 14, no. 2, June 2013. [Online]. Available: <https://doi.org/10.1145/2480759.2480766>
- [25] P. Liberatore, M. Schaerf *et al.*, “The complexity of model checking for belief revision and update,” in *AAAI/IAAI, Vol. 1*, 1996, pp. 556–561.
- [26] P. Liberatore, “Mixed iterated revisions: rationale, algorithms, and complexity,” *ACM Transactions on Computational Logic*, vol. 24, no. 3, pp. 1–49, 2023.
- [27] M.-A. Williams, “Transmutations of knowledge systems,” in *Principles of Knowledge Representation and Reasoning*. Elsevier, 1994, pp. 619–629.
- [28] —, “Towards a practical approach to belief revision: Reason-based change,” *KR*, vol. 96, pp. 412–420, 1996.
- [29] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, “Recommender systems survey,” *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013, <https://doi.org/10.1016/j.knosys.2013.03.012>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0950705113001044>
- [30] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” *ACM Comput. Surv.*, vol. 52, no. 1, Feb. 2019. [Online]. Available: <https://doi.org/10.1145/3285029>
- [31] D. Browne, P. Totterdell, and M. Norman, Eds., *Adaptive User Interfaces*. San Diego, CA: Academic Press (Elsevier), 2016.

- [32] P. Lops, C. Musto, F. Narducci, and G. Semeraro, *Semantics in Adaptive and Personalised Systems: Methods, Tools and Applications*, 1st ed. Cham, Switzerland: Springer Cham, 2019.
- [33] W. Deng, I. Papavasileiou, Z. Qiao, W. Zhang, K.-Y. Lam, and S. Han, “Advances in automation technologies for lower extremity neurorehabilitation: A review and future challenges,” *IEEE Reviews in Biomedical Engineering*, vol. 11, pp. 289–305, 2018.
- [34] L. Wu, X. He, X. Wang, K. Zhang, and M. Wang, “A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 5, pp. 4425–4445, May 2023.
- [35] Z. Yang, S. Dong, and J. Hu, “Gfe: General knowledge enhanced framework for explainable sequential recommendation,” *Knowledge-Based Systems*, vol. 230, p. 107375, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0950705121006377>
- [36] Y. Zhang and X. Chen, “Explainable recommendation: A survey and new perspectives,” *Foundations and Trends® in Information Retrieval*, vol. 14, no. 1, pp. 1–101, 2020, <https://doi.org/10.1561/15000000066>. [Online]. Available: <http://dx.doi.org/10.1561/15000000066>
- [37] J. Doyle and P. London, “A selected descriptor-indexed bibliography to the literature on belief revision,” *ACM SIGART Bulletin*, vol. 71, no. 71, p. 7–22, Apr. 1980. [Online]. Available: <https://doi.org/10.1145/1056441.1056442>
- [38] A. Newell, “The knowledge level,” *Artificial intelligence*, vol. 18, no. 1, pp. 87–127, 1982.
- [39] H. Katsuno and A. O. Mendelzon, “On the difference between updating a knowledge base and revising it,” in *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, ser. KR’91. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991, p. 387–394.
- [40] E. Fermé, M. Garapa, M. D. Reis, Y. Almeida, T. Paulino, and M. Rodrigues, “Knowledge-driven profile dynamics,” *Artificial Intelligence*, vol. 331, p. 104117, 2024, <https://doi.org/10.1016/j.artint.2024.104117>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0004370224000535>

- [41] É. F. D. Cunha, “Nexus brant backoffice para brant,” Master’s thesis, Universidade da Madeira, Funchal, Portugal, 2023, master’s thesis. [Online]. Available: <https://digituma.uma.pt/handle/10400.13/5102>
- [42] A. Newell, *Unified theories of cognition*. Harvard University Press, 1994.
- [43] A. Newell and H. A. Simon, *Computer science as empirical inquiry: symbols and search*. New York, NY, USA: Association for Computing Machinery, 2007, p. 1975. [Online]. Available: <https://doi.org/10.1145/1283920.1283930>
- [44] J. De Kleer, “An assumption-based tms,” *Artificial intelligence*, vol. 28, no. 2, pp. 127–162, 1986.
- [45] W. James, “The will to believe,” *The New World*, vol. 5, no. 1, pp. 327–347, 1896, early articulation of ideas later expanded in the book.
- [46] C. S. Peirce, “The fixation of belief,” *Popular Science Monthly*, vol. 12, no. 1, pp. 1–15, 1877.
- [47] A. Zimmerman, “Pragmatism in the philosophy of mind,” in *The Routledge Companion to Pragmatism*. Routledge eBooks, 2022, pp. 226–238.
- [48] A. Hollingsworth, “Towards a doctrinal pragmatism: Charles s. peirce and the nature of doctrine,” *Neue Zeitschrift für Systematische Theologie und Religionsphilosophie*, vol. 64, pp. 207–228, 2022.
- [49] A. Borgida, “Language features for flexible handling of exceptions in information systems,” *ACM Trans. Database Syst.*, vol. 10, no. 4, pp. 565–603, 1985.
- [50] M. Dalal, “Investigations into a theory of knowledge base revision: Preliminary report,” in *Seventh National Conference on Artificial Intelligence, (AAAI-88)*, St. Paul, 1988, pp. 475–479.
- [51] R. Fagin, J. Ullman, and M. Vardi, “On the semantics of updates in databases,” in *Proceedings of Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, 1983, pp. 352–365.
- [52] H. Katsuno and A. O. Mendelzon, “Propositional knowledge base revision and minimal change,” *Artificial Intelligence*, vol. 52, no. 3, pp. 263–294, 1991.

- [53] J. Martins and S. Shapiro, "A model for belief revision," *Artificial Intelligence*, vol. 35, pp. 25–79, 1988.
- [54] K. Satoh, "Nonmonotonic reasoning by minimal belief revision," in *Proceedings of the International Conference on Fifth Generation Computer Systems (FGCS'88)*, 1988, pp. 455–462.
- [55] M. Winslett, "Reasoning about action using a possible models approach," in *AAAI*, 1988, pp. 89–93.
- [56] B. Nebel, "A knowledge level analysis of belief revision," in *Proceedings of the 1st International Conference of Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, 1989, pp. 301–311.
- [57] A. S. Rao, M. P. Georgeff *et al.*, "Bdi agents: from theory to practice." in *ICMAS*, vol. 95. Menlo Park, CA: AAAI Press, 1995, pp. 312–319.
- [58] P. R. Cohen and H. J. Levesque, "Intention is choice with commitment," *Artif. Intell.*, vol. 42, no. 2-3, pp. 213–261, 1990. [Online]. Available: [https://doi.org/10.1016/0004-3702\(90\)90055-5](https://doi.org/10.1016/0004-3702(90)90055-5)
- [59] I. Levi, "Subjunctives, dispositions, and chances," *Synthese*, vol. 34, pp. 423–455, 1977.
- [60] W. L. Harper, "Rational conceptual change," *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*, vol. 1976, pp. 462–494, 1976.
- [61] S. O. Hansson, *A Textbook of Belief Dynamics. Theory Change and Database Updating.*, ser. Applied Logic Series. Dordrecht: Kluwer Academic Publishers, 1999.
- [62] C. E. Alchourrón and D. Makinson, "On the logic of theory change: Safe contraction," *Studia Logica*, vol. 44, no. 4, pp. 405–422, Dec 1985. [Online]. Available: <https://doi.org/10.1007/BF00370430>
- [63] H. Rott, "On the logic of theory change: More maps between different kinds of contraction functions," *Belief Revision*, vol. 29, pp. 122–141, 1992.
- [64] S. O. Hansson, "Kernel contraction," *Journal of Symbolic Logic*, vol. 59, no. 3, pp. 845–859, 1994.

- [65] P. Gärdenfors, “Epistemic importance and minimal changes of belief,” *Australasian Journal of Philosophy*, vol. 62, no. 2, pp. 136–157, 1984.
- [66] P. Gärdenfors and D. Makinson, “Revisions of knowledge systems using epistemic entrenchment,” in *Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge*, M. Y. Vardi, Ed. Los Altos: Morgan Kaufmann, 1988, pp. 83–95.
- [67] P. Gärdenfors, “Knowledge in flux. modeling the dynamics of epistemic states,” *Studia Logica*, vol. 49, no. 3, pp. 421–424, 1988.
- [68] H. Rott, “Two dogmas of belief revision,” *The Journal of Philosophy*, vol. 97, no. 9, pp. 503–522, 2000.
- [69] —, “Basic entrenchment,” *Studia Logica*, vol. 73, no. 2, pp. 257–280, Mar 2003. [Online]. Available: <https://doi.org/10.1023/A:1022988014704>
- [70] A. C. Nayak, P. Nelson, and H. Polansky, “Belief change as change in epistemic entrenchment,” *Synthese*, vol. 109, no. 2, pp. 143–174, Nov 1996. [Online]. Available: <https://doi.org/10.1007/BF00413766>
- [71] E. Fermé, “A little note about maxichoice and epistemic entrenchment,” in *Proceedings of the Workshop on Logic, Language, Information and Computation (WoLLIC’99)*, R. de Queiroz and W. Carnielli, Eds., Itatiaia, Brazil, 1999, pp. 111–114.
- [72] S. O. Hansson, “A test battery for rational database updating,” *Artificial Intelligence*, vol. 82, pp. 341–352, 1996.
- [73] D. Makinson, “Ways of doing logic: what was different about agm 1985?” *Journal of Logic and Computation*, vol. 13, pp. 3–13, 2002.
- [74] H. Rott, ““Just Because”: Taking Belief Bases Seriously,” Department of Philosophy, Lund University, Lund, Sweden, Technical Report LP-1998-13, 1998.
- [75] S. O. Hansson, “Semi-revision,” *Journal of Applied Non-Classical Logic*, vol. 7, no. 1-2, pp. 151–175, 1997.
- [76] —, “Belief contraction without recovery,” *Studia logica*, vol. 50, no. 2, pp. 251–260, 1991. [Online]. Available: <http://www.jstor.org/stable/20015577>

- [77] J. Doyle, "A model for deliberation, action, and introspection," Massachusetts Institute of Technology, USA, Tech. Rep., 1980.
- [78] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artificial Intelligence*, vol. 29, no. 3, pp. 241–288, 1986.
- [79] D. A. McAllester, "Reasoning utility package user's manual," *Version One, Artificial Intelligence Laboratory, AIM-667, MIT, Cambridge, MA*, 1982.
- [80] J. Doyle, "Truth maintenance systems for problem solving," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, August 1978, department of Electrical Engineering and Computer Science.
- [81] R. Reiter, "A logic for default reasoning," *Artificial Intelligence*, vol. 13, no. 1–2, pp. 81–132, 1980.
- [82] T. Verma and J. Pearl, "Causal networks: semantics and expressiveness," in *Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence*, ser. UAI '88. NLD: North-Holland Publishing Co., 1990, p. 69–78.
- [83] J. Pearl, *Causality: Models, Reasoning and Inference*, 2nd ed. Cambridge: Cambridge University Press, 2009.
- [84] P. Spirtes, C. Glymour, and R. Scheines, "Causation, prediction, and search," *Lecture Notes in Statistics*, vol. 81, 1993.
- [85] T. Dean and K. Kanazawa, "A model for reasoning about persistence and causation," *Computational Intelligence*, vol. 5, no. 3, pp. 142–150, 1989.
- [86] J. Y. Halpern and J. Pearl, "Causes and explanations: A structural-model approach. part i: Causes," *The British Journal for the Philosophy of Science*, vol. 56, no. 4, pp. 843–887, 2005.
- [87] R. E. Fikes and N. J. Nilsson, "STRIPS: a new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3–4, pp. 189–208, 1971.
- [88] C. Hewitt, "Description and theoretical analysis (using schemata) of PLANNER: a language for proving theorems and manipulating models in a robot," Massachusetts Institute of Technology (MIT), Artificial Intelligence Lab, Cambridge, Mass., Tech. Rep. AI-TR-258, April 1972, ph.D. Thesis.

- [89] M. Genesereth, J. S. Bennett, and C. R. Hollander, “Dart: Expert systems for automated computer fault diagnosis,” in *Proceedings of the ACM '81 Conference*, ser. ACM '81. New York, NY, USA: Association for Computing Machinery, 1981, p. 246. [Online]. Available: <https://doi.org/10.1145/800175.809887>
- [90] D. McDermott and G. J. Sussman, “The CONNIVER reference manual,” Massachusetts Institute of Technology, AI Lab, Cambridge, MA, Tech. Rep. Memo No. 259, 1972.
- [91] R. Parikh, “Beliefs, belief revision, and splitting languages,” in *Logic, Language, and Computation*, ser. CSLI Lecture Notes. Springer-Verlag, 1999, vol. 96-2, pp. 266–268.
- [92] J. De Kleer and B. C. Williams, “Diagnosing multiple faults,” *Artificial Intelligence*, vol. 32, no. 1, pp. 97–130, 1987.
- [93] E. Purificato, L. Boratto, and E. W. De Luca, “User Modeling and User Profiling: A Comprehensive Survey,” *arXiv e-prints*, p. arXiv:2402.09660, Feb. 2024, <https://doi.org/10.48550/arXiv.2402.09660>.
- [94] D. N. Freitas, K. Varela, and E. Fermé, “User profiling and its dynamics: A narrative review,” *The European Journal on Artificial Intelligence*, vol. 0, no. 0, p. 30504554251407092, 0. [Online]. Available: <https://doi.org/10.1177/30504554251407092>
- [95] G. Amato and U. Straccia, “User profile modeling and applications to digital libraries,” in *Research and Advanced Technology for Digital Libraries*, S. Abiteboul and A.-M. Vercoustre, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 184–197.
- [96] B. Krulwich, “Lifestyle finder: Intelligent user profiling using large-scale demographic data,” *AI Magazine*, vol. 18, no. 2, p. 37, Jun. 1997. [Online]. Available: <https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1292>
- [97] R. Xie, C. Ling, Y. Wang, R. Wang, F. Xia, and L. Lin, “Deep feedback network for recommendation,” in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 2519–2525, main track. [Online]. Available: <https://doi.org/10.24963/ijcai.2020/349>

- [98] A. Da’u, N. Salim, I. Rabiou, and A. Osman, “Weighted aspect-based opinion mining using deep learning for recommender system,” *Expert Systems with Applications*, vol. 140, p. 112871, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417419305810>
- [99] C. Deng, Y. Zhou, and Z. Dou, “Improving personalized search with dual-feedback network,” in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, ser. WSDM ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 210–218. [Online]. Available: <https://doi.org/10.1145/3488560.3498447>
- [100] S. Gauch, M. Speretta, A. Chandramouli, and A. Micarelli, *User Profiles for Personalized Information Access*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 54–89. [Online]. Available: <https://doi.org/10.1007/978-3-540-72079-9%5F2>
- [101] Z. Zhang, Y. Liu, G. Xu, and H. Chen, “A weighted adaptation method on learning user preference profile,” *Knowledge-Based Systems*, vol. 112, pp. 114–126, 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0950705116303100>
- [102] L. Marin, D. Isern, and A. Moreno, “A generic user profile adaptation framework,” in *Artificial Intelligence Research and Development*, ser. Frontiers in Artificial Intelligence and Applications. Amsterdam, Netherlands: IOS Press, 2010, vol. 220, pp. 143–152.
- [103] P. Brusilovsky and E. Millán, *User Models for Adaptive Hypermedia and Adaptive Educational Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 3–53. [Online]. Available: <https://doi.org/10.1007/978-3-540-72079-9%5F1>
- [104] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, “The query-flow graph: model and applications,” in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, ser. CIKM ’08. New York, NY, USA: Association for Computing Machinery, 2008, p. 609–618. [Online]. Available: <https://doi.org/10.1145/1458082.1458163>
- [105] P. Lops, C. Musto, F. Narducci, and G. Semeraro, *Semantics in Adaptive and Personalised Systems: Methods, Tools and Applications*, ser. Studies in Computational Intelligence. Cham, Switzerland: Springer, 2019, vol. 814.

- [106] C. I. Eke, A. A. Norman, L. Shuib, and H. F. Nweke, “A survey of user profiling: State-of-the-art, challenges, and solutions,” *IEEE Access*, vol. 7, pp. 144 907–144 924, 2019.
- [107] L. Ravi, M. Devarajan, V. V, A. K. Sangaiah, L. Wang, S. A, and V. Subramaniaswamy, “An intelligent location recommender system utilising multi-agent induced cognitive behavioural model,” *Enterprise Information Systems*, vol. 15, no. 10, pp. 1376–1394, 2021. [Online]. Available: <https://doi.org/10.1080/17517575.2020.1812003>
- [108] C. He, D. Parra, and K. Verbert, “Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities,” *Expert Systems with Applications*, vol. 56, pp. 9–27, 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417416300367>
- [109] C. Rana and S. K. Jain, “A study of the dynamic features of recommender systems,” *Artificial Intelligence Review*, vol. 43, no. 1, pp. 141–153, Jan 2015. [Online]. Available: <https://doi.org/10.1007/s10462-012-9359-6>
- [110] F. Isinkaye, Y. Folajimi, and B. Ojokoh, “Recommendation systems: Principles, methods and evaluation,” *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 261–273, 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1110866515000341>
- [111] S. Zhou, X. Dai, H. Chen, W. Zhang, K. Ren, R. Tang, X. He, and Y. Yu, “Interactive recommender system via knowledge graph-enhanced reinforcement learning,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 179–188. [Online]. Available: <https://doi.org/10.1145/3397271.3401174>
- [112] C. Gao, W. Lei, X. He, M. de Rijke, and T.-S. Chua, “Advances and challenges in conversational recommender systems: A survey,” *AI Open*, vol. 2, pp. 100–126, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2666651021000164>
- [113] A. Ben Hassen, S. Ben Ticha, and A. H. Chaibi, “Deep learning for visual-features extraction based personalized user modeling,” *SN Computer Science*, vol. 3, no. 4, p. 261, Apr 2022. [Online]. Available: <https://doi.org/10.1007/s42979-022-01131-y>

- [114] J. Tang, L. Yao, D. Zhang, and J. Zhang, “A combination approach to web user profiling,” *ACM Trans. Knowl. Discov. Data*, vol. 5, no. 1, Dec. 2010. [Online]. Available: <https://doi.org/10.1145/1870096.1870098>
- [115] T. Qi, F. Wu, C. Wu, and Y. Huang, “News recommendation with candidate-aware user modeling,” in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1917–1921. [Online]. Available: <https://doi.org/10.1145/3477495.3531778>
- [116] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, June 2005.
- [117] C. Rana and S. K. Jain, “An extended evolutionary clustering algorithm for an adaptive recommender system,” *Social Network Analysis and Mining*, vol. 4, no. 1, p. 164, Feb 2014. [Online]. Available: <https://doi.org/10.1007/s13278-014-0164-x>
- [118] T. Di Noia, C. Magarelli, A. Maurino, M. Palmonari, and A. Rula, “Using ontology-based data summarization to develop semantics-aware recommender systems,” in *The Semantic Web*, A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, and M. Alam, Eds. Cham: Springer International Publishing, 2018, pp. 128–144.
- [119] J.-C. Zhang, A. M. Zain, K.-Q. Zhou, X. Chen, and R.-M. Zhang, “A review of recommender systems based on knowledge graph embedding,” *Expert Systems with Applications*, vol. 250, p. 123876, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417424007425>
- [120] Y. Cao, X. Wang, X. He, Z. Hu, and T.-S. Chua, “Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences,” in *The World Wide Web Conference*, ser. WWW ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 151–161. [Online]. Available: <https://doi.org/10.1145/3308558.3313705>
- [121] Y. Liu, Z. Zhou, Y. Li, and D. Jin, “Urban knowledge graph aided mobile user profiling,” *ACM Trans. Knowl. Discov. Data*, vol. 18, no. 1, Oct. 2023. [Online]. Available:

<https://doi.org/10.1145/3596604>

- [122] P. Wang, K. Liu, L. Jiang, X. Li, and Y. Fu, “Incremental mobile user profiling: Reinforcement learning with spatial knowledge graph for modeling event streams,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 853–861. [Online]. Available: <https://doi.org/10.1145/3394486.3403128>
- [123] L.-Y. Zhou and Y.-Y. Wang, “Simulation of personalized english learning path recommendation system based on knowledge graph and deep reinforcement learning,” *Scientific Reports*, vol. 15, no. 1, p. 34554, Oct 2025. [Online]. Available: <https://doi.org/10.1038/s41598-025-17918-x>
- [124] H. Cui, J. Lu, R. Xu, S. Wang, W. Ma, Y. Yu, S. Yu, X. Kan, C. Ling, L. Zhao, Z. S. Qin, J. C. Ho, T. Fu, J. Ma, M. Huai, F. Wang, and C. Yang, “A review on knowledge graphs for healthcare: Resources, applications, and promises,” *Journal of Biomedical Informatics*, vol. 169, p. 104861, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1532046425000905>
- [125] L. Li, Z. Yang, B. Wang, and M. Kitsuregawa, “Dynamic adaptation strategies for long-term and short-term user profile to personalize search,” in *Advances in Data and Web Management*, G. Dong, X. Lin, W. Wang, Y. Yang, and J. X. Yu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 228–240.
- [126] P. N. Bennett, R. W. White, W. Chu, S. T. Dumais, P. Bailey, F. Borisyuk, and X. Cui, “Modeling the impact of short- and long-term behavior on search personalization,” in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’12. New York, NY, USA: Association for Computing Machinery, 2012, p. 185–194. [Online]. Available: <https://doi.org/10.1145/2348283.2348312>
- [127] P. Zhang and S. Kim, “A survey on incremental update for neural recommender systems,” 2023. [Online]. Available: <https://arxiv.org/abs/2303.02851>
- [128] H. Ding, W. Zhu, G. Hu, and Z. Bu, “Capturing dynamic user preferences: A recommendation system model with non-linear forgetting and evolving topics,” *Systems*, vol. 13, no. 11, 2025. [Online]. Available: <https://www.mdpi.com/2079-8954/13/11/1034>

- [129] K. Malialis, J. Li, C. G. Panayiotou, and M. M. Polycarpou, “Incremental learning with concept drift detection and prototype-based embeddings for graph stream classification,” in *2024 International Joint Conference on Neural Networks (IJCNN)*, June 2024, pp. 1–7.
- [130] H. Yoo, S. Kang, and H. Tong, “Continual recommender systems,” in *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, ser. CIKM ’25. New York, NY, USA: Association for Computing Machinery, 2025, p. 6857–6860. [Online]. Available: <https://doi.org/10.1145/3746252.3761452>
- [131] M. Mitchell, *An introduction to genetic algorithms*, ser. Complex Adaptive Systems. Cambridge, MA: MIT press, 1998.
- [132] B. N. El Houda, B. Nadjia, and M. Abdelkrim, “Queries-based profile evolution using genetic algorithm,” in *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, Nov 2019, pp. 1–6.
- [133] L. Marin, D. Isern, and A. Moreno, “Dynamic adaptation of numerical attributes in a user profile,” *Applied Intelligence*, vol. 39, no. 2, pp. 421–437, Sep 2013. [Online]. Available: <https://doi.org/10.1007/s10489-012-0421-5>
- [134] V. López-Jaquero, F. Montero, and F. Real, “Designing user interface adaptation rules with t: Xml,” in *Proceedings of the 14th International Conference on Intelligent User Interfaces*, ser. IUI ’09. New York, NY, USA: Association for Computing Machinery, 2009, p. 383–388. [Online]. Available: <https://doi.org/10.1145/1502650.1502705>
- [135] N. Pukkhem, “Lorecommendnet: An ontology-based representation of learning object recommendation,” in *Recent Advances in Information and Communication Technology*, S. Boonkrong, H. Unger, and P. Meesad, Eds. Cham: Springer International Publishing, 2014, pp. 293–303.
- [136] G. Adomavicius and A. Tuzhilin, *Context-Aware Recommender Systems*. Boston, MA: Springer US, 2011, pp. 217–253. [Online]. Available: <https://doi.org/10.1007/978-0-387-85820-3%5F7>
- [137] K.-L. Skillen, L. Chen, C. D. Nugent, M. P. Donnelly, W. Burns, and I. Solheim, “Ontological user profile modeling for context-aware application personalization,” in *Ubiquitous Computing*

- and Ambient Intelligence*, J. Bravo, D. López-de Ipiña, and F. Moya, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 261–268.
- [138] J. Thomsen, Y. Vanrompay, and Y. Berbers, “Evolution of context-aware user profiles,” in *2009 International Conference on Ultra Modern Telecommunications & Workshops*, Oct 2009, pp. 1–6.
- [139] R. Schaefer, W. Mueller, and J. Groppe, “Profile processing and evolution for smart environments,” in *Ubiquitous Intelligence and Computing*, J. Ma, H. Jin, L. T. Yang, and J. J.-P. Tsai, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 746–755.
- [140] T. Ruotsalo, K. Haav, A. Stoyanov, S. Roche, E. Fani, R. Deliai, E. Mäkelä, T. Kauppinen, and E. Hyvönen, “Smartmuseum: A mobile recommender system for the web of data,” *Journal of Web Semantics*, vol. 20, pp. 50–67, 2013. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1570826813000061>
- [141] M. Golemati, A. Katifori, C. Vassilakis, G. Lepouras, and C. Halatsis, “Creating an ontology for the user profile: Method and applications,” in *Proceedings of the First International Conference on Research Challenges in Information Science, RCIS 2007, Ouarzazate, Morocco, April 23-26, 2007*, C. Rolland, O. Pastor, and J. Cavarero, Eds., 2007, pp. 407–412.
- [142] M. Sutterer, O. Droegehorn, and K. David, “User profile selection by means of ontology reasoning,” in *2008 Fourth Advanced International Conference on Telecommunications*, June 2008, pp. 299–304.
- [143] K. Xu, M. Zhu, D. Zhang, and T. Gu, “Context-aware content filtering & presentation for pervasive & mobile information systems,” in *Proceedings of the 1st International Conference on Ambient Media and Systems*, ser. Ambi-Sys ’08. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.
- [144] Q. Gao, S. M. Xi, Y. I. Cho, and E. T. Matson, “A multi-agent context-based personalized user preference profile construction approach,” in *Soft Computing in Advanced Robotics*, Y.-T. Kim, I. Kobayashi, and E. Kim, Eds. Cham: Springer International Publishing, 2014, pp. 55–69. [Online]. Available: <https://doi.org/10.1007/978-3-319-05573-2%5F6>
- [145] H. Yin, B. Cui, L. Chen, Z. Hu, and X. Zhou, “Dynamic user modeling in social

- media systems,” *ACM Trans. Inf. Syst.*, vol. 33, no. 3, Mar. 2015. [Online]. Available: <https://doi.org/10.1145/2699670>
- [146] L. O. Colombo-Mendoza, R. Valencia-García, A. Rodríguez-González, G. Alor-Hernández, and J. J. Samper-Zapater, “Recommetz: A context-aware knowledge-based mobile recommender system for movie showtimes,” *Expert Systems with Applications*, vol. 42, no. 3, pp. 1202–1222, 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417414005557>
- [147] R. Y. K. Lau and L. Song, *Belief Revision for Intelligent Web Service Recommendation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–66. [Online]. Available: <https://doi.org/10.1007/978-3-642-30454-5%5F4>
- [148] A. L. Faria, Y. Almeida, D. Branco, J. Câmara, M. Cameirão, L. Ferreira, A. Moreira, T. Paulino, P. Rodrigues, M. Spinola, M. Vilar, S. Bermúdez i Badia, M. Simões, and E. Fermé, “Neuroaireh@b: an artificial intelligence-based methodology for personalized and adaptive neurorehabilitation,” *Frontiers in Neurology*, vol. 14, 2023. [Online]. Available: <https://www.frontiersin.org/journals/neurology/articles/10.3389/fneur.2023.1258323>
- [149] W. H. Organization *et al.*, *Global action plan on the public health response to dementia 2017–2025*. World Health Organization, 2017.
- [150] F. Mayer, A. Di Pucchio, E. Lacorte, I. Bacigalupo, F. Marzolini, G. Ferrante, V. Minardi, M. Masocco, M. Canevelli, T. Di Fiandra, and N. Vanacore, “An estimate of attributable cases of alzheimer disease and vascular dementia due to modifiable risk factors: The impact of primary prevention in europe and in italy,” *Dementia and Geriatric Cognitive Disorders Extra*, vol. 8, no. 1, pp. 60–71, 02 2018. [Online]. Available: <https://doi.org/10.1159/000487079>
- [151] G. Livingston, A. Sommerlad, V. Orgeta, S. G. Costafreda, J. Huntley, D. Ames, C. Ballard, S. Banerjee, A. Burns, J. Cohen-Mansfield *et al.*, “Dementia prevention, intervention, and care,” *The Lancet*, vol. 390, no. 10113, pp. 2673–2734, 2017.
- [152] M. D. Mijajlović, A. Pavlović, M. Brainin, W.-D. Heiss, T. J. Quinn, H. B. Ihle-Hansen, D. M. Hermann, E. B. Assayag, E. Richard, A. Thiel *et al.*, “Post-stroke dementia—a comprehensive review,” *BMC medicine*, vol. 15, no. 1, pp. 1–12, 2017.

- [153] J. B. Miller and W. B. Barr, “The technology crisis in neuropsychology,” *Archives of Clinical Neuropsychology*, vol. 32, no. 5, pp. 541–554, 05 2017. [Online]. Available: <https://doi.org/10.1093/arclin/acx050>
- [154] L. A. Rabin, E. Paolillo, and W. B. Barr, “Stability in test-usage practices of clinical neuropsychologists in the united states and canada over a 10-year period: A follow-up survey of ins and nan members,” *Archives of Clinical Neuropsychology*, vol. 31, no. 3, pp. 206–230, 03 2016. [Online]. Available: <https://doi.org/10.1093/arclin/acw007>
- [155] T. D. Parsons, “Virtual reality for enhanced ecological validity and experimental control in the clinical, affective and social neurosciences,” *Frontiers in Human Neuroscience*, vol. Volume 9 - 2015, 2015. [Online]. Available: <https://www.frontiersin.org/journals/human-neuroscience/articles/10.3389/fnhum.2015.00660>
- [156] A. Aminov, J. M. Rogers, S. Middleton, K. Caeyenberghs, and P. H. Wilson, “What do randomized controlled trials say about virtual rehabilitation in stroke? a systematic literature review and meta-analysis of upper-limb and cognitive outcomes,” *Journal of neuroengineering and rehabilitation*, vol. 15, no. 1, pp. 1–24, 2018.
- [157] M. C. Howard, “A meta-analysis and systematic literature review of virtual reality rehabilitation programs,” *Computers in Human Behavior*, vol. 70, pp. 317–327, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0747563217300134>
- [158] M. G. Maggio, D. Latella, G. Maresca, F. Sciarrone, A. Manuli, A. Naro, R. De Luca, and R. S. Calabrò, “Virtual reality and cognitive rehabilitation in people with stroke: an overview,” *Journal of Neuroscience Nursing*, vol. 51, no. 2, pp. 101–105, 2019.
- [159] A. L. Faria, A. Andrade, L. Soares, and S. B. i Badia, “Benefits of virtual reality based cognitive rehabilitation through simulated activities of daily living: a randomized controlled trial with stroke patients,” *Journal of neuroengineering and rehabilitation*, vol. 13, no. 1, pp. 1–12, 2016.
- [160] A. L. Faria, M. S. Pinho, and S. Bermúdez i Badia, “A comparison of two personalization and adaptive cognitive rehabilitation approaches: a randomized controlled trial with chronic stroke patients,” *Journal of NeuroEngineering and Rehabilitation*, vol. 17, no. 1, p. 78, Jun 2020. [Online]. Available: <https://doi.org/10.1186/s12984-020-00691-5>

- [161] T. Paulino and M. S. Cameirão, “Characterizing the use of interactive technologies for cognitive rehabilitation in portuguese healthcare institutions,” in *Proc. 13th Intl Conf. on Disability, Virtual Reality and Assoc. Technologies, P Gamito, D Brown, and S Koenig (Eds), pp. n-m, Serpa, Portugal, 8-10 Sept. 2021*. Universidade Lusófona de Humanidades e Tecnologias, Portugal, 2021, pp. 104–111. [Online]. Available: <http://hdl.handle.net/10400.13/5731>
- [162] S. Vermeent, G. Van Elswijk, B. Schmand, L. Klaming, J. Miller, and R. Dotsch, “A-62 the philips intellispace cognition platform: Ability to classify a mixed clinical sample,” *Archives of Clinical Neuropsychology*, vol. 34, no. 6, pp. 922–922, 07 2019. [Online]. Available: <https://doi.org/10.1093/arclin/acz034.62>
- [163] R. J. Sbordone, “Ecological validity: Some critical issues for the neuropsychologist,” in *Ecological validity of neuropsychological testing*, R. J. Sbordone and C. J. Long, Eds. Gr Press/St Lucie Press, Inc, 1996, p. 15–41.
- [164] D. Martens, J. Vanthienen, W. Verbeke, and B. Baesens, “Performance of classification models from a user perspective,” *Decision Support Systems*, vol. 51, no. 4, pp. 782–793, 2011, recent Advances in Data, Text, and Media Mining Information Issues in Supply Chain and in Service System Design. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S016792361100042X>
- [165] P.-H. C. Chen, Y. Liu, and L. Peng, “How to develop machine learning models for healthcare,” *Nature materials*, vol. 18, no. 5, pp. 410–414, 2019.
- [166] M. S. Sirsat, E. Fermé, and J. Câmara, “Machine learning for brain stroke: A review,” *Journal of Stroke and Cerebrovascular Diseases*, vol. 29, no. 10, p. 105162, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1052305720305802>
- [167] J. Solana, C. Cáceres, A. García-Molina, P. Chausa, E. Opisso, T. Roig-Rovira, E. Menasalvas, J. M. Tormos-Muñoz, and E. J. Gómez, “Intelligent therapy assistant (ita) for cognitive rehabilitation in patients with acquired brain injury,” *BMC medical informatics and decision making*, vol. 14, no. 1, pp. 1–13, 2014.
- [168] J. Solana, C. Cáceres, A. García-Molina, E. Opisso, T. Roig, J. M. Tormos, and E. J.

- Gómez, “Improving brain injury cognitive rehabilitation by personalized telerehabilitation services: Guttmann neuropsychological trainer,” *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 1, pp. 124–131, 2015.
- [169] H.-T. Jung, H. Lee, K. Kim, B. Kim, S. Park, T. Ryu, Y. Kim, J.-F. Daneault, and S. I. Lee, “Predicting cognitive impairment level after a serious game-based therapy in chronic stroke survivors,” in *2019 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*, 2019, pp. 1–4.
- [170] M. Heffernan, G. Andrews, M. A. F. Singh, M. Valenzuela, K. J. Anstey, A. J. Maeder, J. McNeil, L. Jorm, N. T. Lautenschlager, P. S. Sachdev, J. A. Ginige, M. J. Hobbs, C. Boulamatsis, T. Chau, L. Cobiack, K. L. Cox, K. Daniel, V. M. Flood, Y. Guerrero, J. Gunn, N. Jain, N. A. Kochan, A. Lampit, Y. Mavros, J. Meiklejohn, Y. Noble, F. O’Leary, S. Radd-Vagenas, C. C. Walton, , and H. Brodaty, “Maintain your brain: Protocol of a 3-year randomized controlled trial of a personalized multi-modal digital health intervention to prevent cognitive decline among community dwelling 55 to 77 year olds,” *Journal of Alzheimer’s Disease*, vol. 70, no. s1, pp. S221–S237, 2019, pMID: 30475762. [Online]. Available: <https://doi.org/10.3233/JAD-180572>
- [171] C. C. Walton, A. Lampit, C. Boulamatsis, H. Hallock, P. Barr, J. A. Ginige, H. Brodaty, T. Chau, M. Heffernan, P. S. Sachdev, M. A. Fiatarone Singh, and M. Valenzuela, “Design and development of the brain training system for the digital “maintain your brain” dementia prevention trial,” *JMIR Aging*, vol. 2, no. 1, p. e13135, Feb 2019. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/31518277>
- [172] M. Gil-Pagés, J. Solana, R. Sánchez-Carrión, J. M. Tormos, A. Enseñat-Cantalops, and A. García-Molina, “A customized home-based computerized cognitive rehabilitation platform for patients with chronic-stage stroke: study protocol for a randomized controlled trial,” *Trials*, vol. 19, no. 1, pp. 1–10, 2018.
- [173] Y. Almeida, M. S. Sirsat, S. Bermúdez i Badia, and E. Fermé, “Ai-rehab: A framework for ai driven neurorehabilitation training - the profiling challenge,” in *Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies - Cognitive Health IT*, INSTICC. SciTePress, 2020, pp. 845–853.

- [174] C. van Heugten, G. W. Gregório, and D. Wade, “Evidence-based cognitive rehabilitation after acquired brain injury: A systematic review of content of treatment,” *Neuropsychological Rehabilitation*, vol. 22, no. 5, pp. 653–673, 2012, pMID: 22537117. [Online]. Available: <https://doi.org/10.1080/09602011.2012.680891>
- [175] D. Sharek and E. Wiebe, “Using flow theory to design video games as experimental stimuli,” *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 55, no. 1, pp. 1520–1524, 2011. [Online]. Available: <https://journals.sagepub.com/doi/abs/10.1177/1071181311551316>
- [176] D. Grossi and G. Pigozzi, *Judgment Aggregation: A Primer*. Morgan & Claypool Publishers, 2014. [Online]. Available: <https://hal.science/hal-01497079>
- [177] G. Pigozzi, “Belief Merging and Judgment Aggregation,” in *The Stanford Encyclopedia of Philosophy*, Spring 2021 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2021.
- [178] S. Freitas, S. Batista, A. C. Afonso, M. R. Simões, L. de Sousa, L. Cunha, and I. Santana, “The montreal cognitive assessment (moca) as a screening test for cognitive dysfunction in multiple sclerosis,” *Applied Neuropsychology: Adult*, vol. 25, no. 1, pp. 57–70, 2018, pMID: 27791389. [Online]. Available: <https://doi.org/10.1080/23279095.2016.1243108>
- [179] P. A. G. Rodrigues, “A framework for ai-driven neurorehabilitation training: the profiling challenge,” Master’s thesis, Universidade da Madeira (Portugal), 2022.
- [180] T. d. C. a. M. Paulino, “From design to evaluation: a human-centered approach to ai-powered remote cognitive training for stroke patients,” Doctoral Thesis, Universidade da Madeira, Funchal, Portugal, January 2025, digitUMa Institutional Repository; Open Access. [Online]. Available: <http://hdl.handle.net/10400.13/7187>
- [181] L. B. Sousa, G. Prieto, M. Vilar, H. Firmino, and M. R. Simões, “The adults and older adults functional assessment inventory: A rasch model analysis,” *Research on Aging*, vol. 37, no. 8, pp. 787–814, 2015, pMID: 25651593. [Online]. Available: <https://doi.org/10.1177/0164027514564469>

- [182] J. Câmara, R. Ferreira, L. Teixeira, J. Nóbrega, C. Romeira, S. B. i. Badia, and A. L. Faria, “Efficacy of adaptive cognitive training through desktop virtual reality and paper-and-pencil in the treatment of mental and behavioral disorders,” *Virtual Reality*, vol. 27, no. 1, pp. 291–306, Mar 2023. [Online]. Available: <https://doi.org/10.1007/s10055-021-00559-6>
- [183] D. Romero-Ayuso, Álvaro Castellero-Perea, P. González, E. Navarro, J. P. Molina-Massó, M. J. Funes, P. Ariza-Vega, A. Toledano-González, and J. M. Triviño-Juárez, “Assessment of cognitive instrumental activities of daily living: a systematic review,” *Disability and Rehabilitation*, vol. 43, no. 10, pp. 1342–1358, 2021, pMID: 31549907. [Online]. Available: <https://doi.org/10.1080/09638288.2019.1665720>
- [184] P. W. Burgess, E. Veitch, A. de Lacy Costello, and T. Shallice, “The cognitive and neuroanatomical correlates of multitasking,” *Neuropsychologia*, vol. 38, no. 6, pp. 848–863, 2000. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0028393299001347>
- [185] R. H. Logie, S. Trawley, and A. Law, “Multitasking: multiple, domain-specific cognitive functions in a virtual environment,” *Memory & Cognition*, vol. 39, no. 8, pp. 1561–1574, Nov 2011. [Online]. Available: <https://doi.org/10.3758/s13421-011-0120-1>
- [186] M. J. Muller, J. H. Haslwanter, and T. Dayton, “Chapter 11 - participatory practices in the software lifecycle,” in *Handbook of Human-Computer Interaction (Second Edition)*, second edition ed., M. G. Helander, T. K. Landauer, and P. V. Prabhu, Eds. Amsterdam: North-Holland, 1997, pp. 255–297. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780444818621500777>
- [187] A. L. Faria, T. Paulino, and S. B. i. Badia, “Comparing adaptive cognitive training in virtual reality and paper-pencil in a sample of stroke patients,” in *2019 International Conference on Virtual Rehabilitation (ICVR)*, 2019, pp. 1–7.
- [188] A. L. Faria, M. S. Pinho, and S. Bermúdez i Badia, “Capturing expert knowledge for the personalization of cognitive rehabilitation: Study combining computational modeling and a participatory design strategy,” *JMIR Rehabil Assist Technol*, vol. 5, no. 2, p. e10714, Dec 2018. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/30522994>

- [189] G. C. dos Santos Palma, U. C. Corrêa, and C. Torriani-Pasin, “The differentiated effect of the task complexity on retention and transfer of stroke survivors,” *Human Movement Science*, vol. 69, p. 102545, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167945719303793>
- [190] C. R. Snyder, S. J. Lopez, L. M. Edwards, and S. C. Marques, *The Oxford Handbook of Positive Psychology*. Oxford University Press, 01 2021. [Online]. Available: <https://doi.org/10.1093/oxfordhb/9780199396511.001.0001>
- [191] J. Câmara, T. Paulino, M. Spínola, D. Branco, M. Cameirão, A. L. Faria, L. Ferreira, A. Moreira, A. R. Silva, and M. Vilar, “Treino cognitivo no pós-avc: Um estudo-piloto com a plataforma neuroaireh@b,” *Psicologia, Saúde & Doenças*, vol. 23, no. 2, pp. 343–354, 2022. [Online]. Available: <http://hdl.handle.net/10400.13/4984>
- [192] J. Câmara, L. Ferreira, A. L. Faria, M. Vilar, and S. Bermúdez i Badia, “Feasibility, acceptability, and preliminary impact of full-body interaction on computerized cognitive training based on instrumental activities of daily living: A pilot randomized controlled trial with chronic psychiatric inpatients,” *Games for Health Journal*, vol. 11, no. 6, pp. 435–446, 2022, pMID: 36251861. [Online]. Available: <https://doi.org/10.1089/g4h.2021.0228>
- [193] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, 2009, <https://doi.org/10.1137/1.9780898717716>. [Online]. Available: <https://epubs.siam.org/doi/abs/10.1137/1.9780898717716>

This work is licensed under a Creative Commons “Attribution 4.0 International” license.

