

Von Fällen zu Entscheidungen mittels ASP

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Logic and Artificial Intelligence

eingereicht von

Adrian Chroust, B.Sc.

Matrikelnummer 12023146

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Univ.Prof.in Dr.in Agata Ciabattoni

Wien, 17. November 2025

Adrian Chroust

 Agata Ciabattoni

From Cases to Decisions through ASP

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Logic and Artificial Intelligence

by

Adrian Chroust, B.Sc.

Registration Number 12023146

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof.in Dr.in Agata Ciabattoni

Vienna, November 17, 2025

Adrian Chroust


Agata Ciabattoni

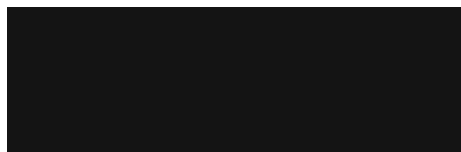
Erklärung zur Verfassung der Arbeit

Adrian Chroust, B.Sc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit Produktnamen und Versionsnummer/Datum angegeben.

Wien, 17. November 2025



Adrian Chroust

Danksagung

Zuerst möchte ich meiner Betreuerin, Univ.Prof.in Dr.in Agata Ciabattoni, danken. Ich danke ihr für ihre schnelle und direkte Kommunikation, ihre wertvollen Einblicke, und ihre konstruktiven Inputs, die die Qualität meiner Arbeit deutlich verbessert haben. Ihr Rat und ihre Anleitung haben mich dazu inspiriert, mich voll und ganz mit meiner Forschung zu befassen und während der Erstellung meiner Arbeit nach Exzellenz zu streben.

Ich möchte Dipl.-Ing. Henri Thölke und Yoann Morello, MA für das motivierende Feedback und die durchdachten Vorschläge danken, die die Struktur meiner Argumente gestärkt und mir geholfen haben, das Beste aus meiner Arbeit herauszuholen.

Ich möchte auch Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Christian Fermüller und Dipl.-Ing. Dr.techn. Sebastian Skritek für ihre großzügige Unterstützung und Ermutigung während meines Studiums danken und dafür, dass sie meine Neugier für die Forschung geweckt haben.

Abschließend möchte ich meiner Familie und meinen Freund:innen danken, die mich während meines Studiums und darüber hinaus unterstützt haben. Ich danke meinen Eltern, Andrea und Martin Chroust, für ihre bedingungslose Unterstützung und ihr Mitgefühl, und ich danke meinen Geschwistern, Niklas und Ella Chroust, für ihre Vertrautheit und Gutmütigkeit. Ich danke meiner Freundin, Clara Becker, für ihre stetige Zuneigung und ihr Vertrauen in mich. Außerdem danke ich meinen Freund:innen und all den wunderbaren Menschen, die ich während meines Studiums kennenlernen durfte, für die schönen Erinnerungen, inspirierenden Gespräche, und die Solidarität in stressigen Zeiten.

Acknowledgements

First, I would like to thank my advisor, Univ.Prof.in Dr.in Agata Ciabattoni. I thank her for her quick and straightforward communication, valuable insights, and constructive inputs, which greatly enhanced the quality of my work. Her advice and guidance inspired me to fully engage with my research and to pursue excellence throughout the development of my thesis.

I would like to thank Dipl.-Ing. Henri Thölke and Yoann Morello, MA for the motivating feedback and thoughtful suggestions, which strengthened the structure of my arguments and helped me bring out the best in my thesis.

I also would like to thank Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Christian Fermüller and Dipl.-Ing. Dr.techn. Sebastian Skritek for their generous support and encouragement throughout my studies, and for inspiring my curiosity for research.

Finally, I would like to thank my family and friends who supported me throughout my studies and beyond. I thank my parents, Andrea and Martin Chroust, for their unconditional support and affection, and I thank my siblings, Niklas and Ella Chroust, for their companionship and kindness. I thank my girlfriend, Clara Becker, for her boundless care and faith in me. And, I thank my friends and all the amazing people I got to know during my studies for the wonderful memories, inspiring conversations, and solidarity in stressful times.

Kurzfassung

Automatisierte Entscheidungsfindung beinhaltet die Klassifikation von Informationen, um in einem gegebenen Kontext geeignete Aktionen auszuführen. Dieses Problem ist bereits ausführlich im Bereich *künstliche Intelligenz* (KI) und Recht untersucht worden. Dort haben sich zwei Herangehensweisen etabliert, jede mit ihren eigenen Stärken und Schwächen. *Maschinelles Lernen* verwendet große Datensätze um Klassifikationen durchzuführen und unterstützt breite Anwendungsgebiete. Allerdings ist der interne Entscheidungsprozess oft verschleiert. Diese Arbeit fokussiert sich auf die zweite Herangehensweise, nämlich *symbolische KI*, welche formale Logik nutzt, um Entscheidungen transparent und interpretierbar zu machen.

Wir untersuchen *fallbasiertes normatives Schließen*, wobei die Entscheidungsfindung von vergangenen Fällen gelenkt und eingeschränkt wird. Wir untersuchen die Stärken und Schwächen von John Horty's *Dimensional Reason Model*, welches ursprünglich für Rechtsfälle konzipiert worden ist, aber in diversen Bereichen Anwendung findet. Mit dieser Theorie als Grundlage verwenden wir *Answer-Set-Programmierung* (ASP), eine Sprache zur Spezifikation von Logikprogrammen, um ein System für automatisierte Entscheidungsfindung zu bauen.

Unsere Analyse zeigt mehrere Einschränkungen des Dimensional Reason Modells auf, wodurch die Entwicklung eines neuen Modells motiviert wird. Wir präsentieren das *Dimensional Reduction Model*, welches die Entscheidungsfindung vereinfacht, indem es sich auf eine einfachere Beschränkungslogik stützt. Um die praktische Anwendbarkeit des Modells zu testen, erstellen wir eine Fallstudie von 20 Fällen und vergleichen die erlaubten Entscheidungen des Dimensional Reason Modells mit denen des Dimensional Reduction Modells. Weiters werden beide Modelle in ASP implementiert. Die Fallstudie dient hier der Verifikation, um sicherzustellen dass die Programmergebnisse mit denen der manuell ausgewerteten Fälle übereinstimmen.

Diese Arbeit trägt zum Gebiet der Maschinenethik bei, indem ein neues Modell zur Beschreibung von Präzedenzfällen präsentiert wird, das den State-of-the-Art erweitert. Weiters stellen wir eine Implementierung bereit, die automatisierte Entscheidungsfindung ermöglicht.

Der Code, der für diese Arbeit geschrieben worden ist, kann unter <https://git.logic.at/cbr-asp/cbr-asp> gefunden werden.

Abstract

Automated decision-making involves classifying information to select appropriate actions in a given context. This problem has been widely studied in the field of *artificial intelligence* (AI) and law, where two primary approaches have emerged, each with its own strengths and limitations. *Machine learning* relies on large datasets to perform classification and supports a broad range of applications; however, it often masks the reasoning behind decisions. This thesis focuses on the second approach—*symbolic AI*, which employs formal logic to guide decision-making in a transparent and interpretable manner.

We investigate *case-based normative reasoning*, a framework in which decision-making is guided and constrained by past cases that serve as precedent. Specifically, we examine the capabilities and limitations of John Horty’s *dimensional reason model*, originally developed for legal reasoning but applicable across a variety of domains. Building on this theory, we employ *answer set programming* (ASP), a logic programming paradigm, to construct an automated framework for decision-making.

Our analysis reveals several limitations of the dimensional reason model, motivating the development of a new approach that addresses these issues. We introduce the *dimensional reduction model*, which simplifies the assessment of permissible decisions by relying on a simpler notion of constraint. To evaluate its practical applicability, we design a case study consisting of 20 cases and compare the permitted decisions of the dimensional reason model and the dimensional reduction model. Both models are then implemented using ASP, and the case study is used to verify that the computational results align with the manually assessed cases.

This thesis contributes to the broader field of machine ethics by presenting a model of precedential constraint that extends the state of the art, along with an implementation that enables automated and explainable decision-making.

The code written for this thesis can be found at <https://git.logic.at/cbr-asp/cbr-asp>.

Contents

Kurzfassung	xi
Abstract	xiii
Contents	xv
1 Introduction	1
2 Background: Modelling Precedential Constraint	7
2.1 The Standard Setting	8
2.1.1 The Result Model	10
2.1.2 The Reason Model	14
2.1.3 Result Model vs Reason Model	24
2.2 The Dimensional Setting	26
2.2.1 The Result Model	30
2.2.2 The Reason Model	35
2.2.3 Result Model vs Reason Model: Collapse	39
2.2.4 The Revised Reason Model	41
2.2.5 Result Model vs Revised Reason Model: No Collapse	46
3 Reducing the Models	51
3.1 Strengthening Reasons	51
3.2 Motivation: The Standard Reduction Model	54
3.2.1 Relation to the Reason Model	57
3.3 The Dimensional Reduction Model	58
3.3.1 Benchmarks from the Literature	60
3.3.2 Relation to the Reason Model	65
3.3.3 Relation to the Reason Model with Complete Rules	68
3.3.4 Strengthening Reasons	69
4 Implementation	71
4.1 Answer Set Programming	72
4.2 Input	74
4.2.1 Dimensions	74
	xv

4.2.2	Solved Cases	77
4.2.3	Unsolved Cases	78
4.3	Shared Rules	79
4.3.1	Sides and Opposite Sides	79
4.3.2	Dimension Value Relation	80
4.4	Approach Outline	83
4.5	The Dimensional Reason Model	85
4.5.1	Case Base Consistency	86
4.5.2	Decisions	93
4.5.3	Rule Selection	95
4.6	The Dimensional Reduction Model	95
4.6.1	Translation	95
4.6.2	Consistency	96
4.6.3	Decisions	99
4.6.4	Rule Selection	100
5	Case Study	101
5.1	Generating a Dataset	101
5.2	Model Comparison on the Dataset	103
5.3	Discussion	107
6	Related Work	109
6.1	Modelling Precedential Constraint	109
6.2	Automating Case-based Reasoning	111
7	Conclusion	113
7.1	Future Work	114
	Overview of Generative AI Tools Used	117
	Übersicht verwendeter Hilfsmittel	119
	List of Figures	121
	List of Tables	123
	List of Algorithms	125
Reason Model Implementation		125
Reduction Model Implementation		128
Synthetic Dataset Generation		131
	Appendix: Background	135
	Bibliography	139

Introduction

The automated classification of information and decision-making has long been a central challenge in the field of *artificial intelligence* (AI). This task is inherently difficult for humans, as it requires defining rules that account for every possible situation. This challenge stems from the fact that much of human decision-making depends on context and subjective interpretation, rather than on strictly defined logical principles. Consider, for instance, a parent deciding whether a child’s behaviour is “good” or “bad”. While it is easy to identify specific factors that define acceptable or unacceptable behaviour in a single case, it is far more difficult to generalise these factors into universal rules. Moreover, different parents may weigh these factors differently, further complicating the creation of a consistent classification framework.

This difficulty points to a fundamental challenge in artificial intelligence: constructing systems capable of reasoning consistently in complex and ambiguous situations. Researchers have generally approached this problem from two directions: *machine learning* and *symbolic AI*.

The first, machine learning, employs statistical algorithms designed to perform classification by generalizing from large datasets. This approach is particularly useful when explicit rule sets are unavailable. However, a major drawback of machine learning systems is their opacity; the internal reasoning behind their outputs is difficult to interpret or verify (if possible at all)—posing a serious concern in domains that demand high levels of trust, such as the legal domain.

In contrast, symbolic AI is based on explicitly defined rules and constraints to govern decision-making. These rules are usually expressed in logic-based frameworks. While this approach may not be suitable for all tasks, it offers a crucial advantage: explainability. Because each decision follows a transparent reasoning path grounded in formal logic, humans can understand and evaluate the process behind the agent’s conclusions.

This thesis focuses on symbolic AI due to its inherent strengths in supporting explainability and accountability—both essential qualities for governing decision-making. Inspired by legal reasoning, we aim to develop an AI system that classifies situations to guide automated decision-making for a variety of applications.

To achieve this, let us first distinguish the type of normative structure within which the agent will operate. To reason with norms, we typically distinguish between rule-based and case-based systems. In rule-based systems, a legislative or authoritative body defines explicit norms that must be followed. Decision-making, in this case, involves applying relevant norms to specific situations to facilitate decisions.

Conversely, case-based systems operate based on past cases used as precedents. Here, decisions are guided not by formalised rules but by consistency with earlier decisions on similar factors. The agent must examine past cases and make judgments that align with established patterns. When a novel situation arises—one not addressed by existing precedents—the agent is expected to propose a reasoned decision, which can then serve as a new precedent. A central challenge in this paradigm is ensuring the coherence and consistency of decisions with the decisions made for past cases.

In this thesis, we focus on case-based reasoning and aim to develop a symbolic AI system that resolves cases in a manner consistent with prior decisions. By automating the comparison between new fact situations and existing precedents, this work contributes to the broader exploration of machine ethics. The symbolic nature of our system ensures that its reasoning process is both transparent and interpretable.

To motivate the development of a system capable of norm-compliant decision-making, we use the “Nanny Robot” example, proposed by Canavotto and Horty [CH22a]. In this hypothetical scenario, a family owns a nanny robot tasked with deciding whether a child is allowed to stay up past its bedtime. If the child behaves, it is allowed to stay up longer, if not, it is sent to bed on time. To make such decisions appropriately, the robot must understand the reasoning behind the parents’ past decisions based on several factors (e.g. “finishing dinner”, or “behaving well”). For this, it is provided with a database of past bedtime decisions. This database could be compiled via a questionnaire for the parents or through behavioural observation over several days. With enough examples, the nanny robot should be able to infer the parents’ preferences and apply them to new situations.

Foundational work in the area of case-based reasoning includes the influential *result model* of precedential constraint, first introduced by Alexander [Ale89]. In this model, case information is represented as a set of Boolean factors—that is, each factor is either present or absent in a given case, and an outcome. A decision is reached by comparing the factual strength of a new situation to that of prior cases. A more elaborate model is the *reason model* of precedential constraint, first introduced by Bench-Capon and Horty [HBC12]. This model also employs Boolean factors to model information. Unlike the result model, which assesses precedential constraint by comparing only factual information, the reason model examines how outcomes of past cases are *justified*. This is achieved by determining which factors in a case are important to the decision. The model thus provides a

structured and transparent way to trace how specific elements of a case contribute to its outcome.

Both of these models were originally designed in the context of case-based legal reasoning for *common law*. In common law, which is also referred to as precedent law or case law, decisions for new fact situations have to be made in accordance with past cases. When a past case constrains the decision of a new fact situation, we refer to the case as a precedent. In this thesis, we will not focus on the legal aspects, but on the general principles of case-based reasoning using these models, beyond legal reasoning.

Result Model and Reason Model We briefly present the result model as described by Horty [Hor04], and the reason model as described by Bench-Capon and Horty [HBC12]. A case is represented by a fact situation—a set of Boolean factors that are either present or absent. The goal is to resolve a new fact situation by comparing it to a collection of previously decided cases within the same domain. A decision must be made in favour of one of the two adversarial parties. Since the models were originally designed for legal reasoning, we call the two parties the *plaintiff* and the *defendant*. Each factor present in the case supports exactly one of the two sides. Where the result model solely compares previous fact situations with new ones to reach a decision, in the reason model the decisions must be justified by specifying which Boolean factors influenced the outcome. We consider an example within our case study of the nanny robot to illustrate how a case decision for a presented fact situation looks like.

Example 1 (Case decision). Consider a case involving a child’s behaviour throughout the day, characterised by three Boolean factors:

- (a) the child hit its sibling,
- (b) the child screamed at its parents, and
- (c) the child did its homework independently.

Factors (a) and (b) could be cited as supporting a decision in favour of the plaintiff—which in our scenario represents the parents wanting to enforce the bedtime—, as they indicate reckless behaviour by the defendant—which in our scenario represents the child wanting to have the bedtime extended. On the other hand, factor (c) could justify a decision in favour of the defendant, i.e., the child, as it provides a mitigating context for its actions.

In this thesis, we focus on an extension of the two models of precedential constraint by replacing Boolean factors with intermediate values (i.e., dimensions). The dimensional models were first proposed by Horty [Hor17]. Where the standard setting represents case information as Boolean factors (i.e., strictly present or absent), the dimensional setting introduces dimensional value assignments to capture the degree or strength of certain information. This allows for a more nuanced and expressive representation of fact

situations. While Boolean factors are limited to binary distinctions, dimensional value assignments allow a continuum or an ordered set of values, enabling a richer understanding of how specific elements influence a decision. The following example illustrates how a dimension can be used in place of a Boolean factor.

Example 2 (Dimension). While a Boolean factor can only indicate the presence or absence of some information—for example, whether a child protested its parents—a dimension provides a more detailed description. Instead of simply stating that the child protested or not, a protest dimension might encode a range of escalating values:

- (1) the child did not protest,
- (2) the child complained but ultimately complied,
- (3) the child complained and cried,
- (4) the child screamed at the parents, and
- (5) the child was kicking and screaming.

This graded structure enables the reasoning process to take into account the severity of an action, rather than treating all instances as equally significant.

As illustrated in the example above, higher values along a dimension can indicate a greater severity of the situation, increasing the likelihood of making a decision against having the bedtime of the child extended. However, a dimension does not inherently favour one side over the other; rather, it enables context-sensitive evaluation based on threshold values. This allows to assess the degree of some information when making a decision.

In the dimensional reason model, justifications for decisions are based on the strength of values within these dimensions. The difference between the result model and the reason model is the presence of reasons in the latter. Reasons serve as the core justification mechanism for a decision. While in the standard reason model a decision is supported by identifying relevant Boolean factors, the dimensional reason model justifies decisions by requiring that certain dimensional value assignments meet or exceed specified thresholds. This allows for more nuanced and explainable reasoning, grounded in the relative strength of the information presented.

Example 3 (Reason). Given the dimension introduced in Example 2, suppose (4) applies in a particular case—that is, the child screamed at the parents. The parents now seek to justify a decision to have the bedtime enforced. To do so, they must base their ruling on a valid reason, which specifies a set of thresholds for a subset of the present dimensions. For instance, the parents might argue that any case meeting or exceeding level (3)—i.e., the child complained and cried—warrants enforcing the bedtime. In that case, even if the child did not scream at the parents, the bedtime may be enforced under

similar circumstances. Alternatively, the parents may use (4) itself as the threshold for their decision, arguing that enforcing the bedtime is justified specifically when the child screamed at the parents. However, the parents cannot base their decision on (5)—that is, the child was kicking and screaming—because this level is not present in the current fact situation. A reason must apply to the case at hand, and (5) exceeds the actual severity observed. The parents may, however, use (5) to justify ruling in favour of the child, arguing that as long as the child was at most kicking and screaming, the bedtime should not be enforced. This illustrates how dimensions can be used to argue in favour of both sides, and how reasons in the dimensional model must align with the factual content of the case and cannot rely on information that is not present.

A key requirement in any case-based reasoning system is the consistency of decisions. On a high level, consistency requires that different decisions are not made in a conflicting manner. Consistency is the mechanism that allows to restrict which decisions are permissible for a new fact situation.

Central to the models in the standard setting is the fact that the reason model has a stronger notion of constraint than the result model. That is, any decision that is restricted in the result model must also be restricted in the reason model, while the converse does not hold. However, a central point of contention arises in how new decisions are resolved in the dimensional setting: In his original formulation of the dimensional reason model [Hor17], Horty observes that the model collapses with the dimensional result model. This collapse implies that reasons become irrelevant in determining new outcomes under certain specifications of the model, since the same constraint can be enforced with the result model which does not require the specification of reasons.

To address this issue, Horty proposes a revision to the definition of consistency in a later paper [Hor21], introducing additional constraints on how decisions can be justified within the dimensional reason model. These modifications prevent a collapse with the dimensional result model to preserve the explanatory role of reasons, ensuring that decisions remain grounded in explicit justificatory structures, rather than being derived solely from the outcomes of similar situations found in past cases. However, this new notion of consistency is complicated, and gives rise to a new problem. This problem is related to how existing reasons can be changed while preserving consistency.

To solve this problem, we refine the model giving rise to what we call the reduction model, and argue for its meaningfulness for case-based reasoning. This new model performs a reduction of case information to the result model, and then uses the simple notions of constraint of the result model for reasoning. As a motivation for its definition, we first introduce the reduction model of constraint in the standard setting and show that it is equivalent to the standard reason model. We then adapt this definition to the dimensional setting, and then show that the dimensional reduction model has a notion of constraint that is stronger than that of the revised dimensional reason model. Furthermore, in the dimensional reduction model it no longer happens that strengthening a reason can cause an inconsistency to occur, an anomaly that occurs in the reason model. We argue

why certain cases should be constrained like they are in the reduction model, where the revised reason model does not constrain a decision.

We implement both the dimensional reason model and the dimensional reduction model using answer set programming (ASP). ASP is a logic programming language that allows to model constraints in a similar fashion as in first order logic. Our implementation enables to check both the consistency of cases in the models as well as which decisions are permitted and constrained for new fact situations.

To further analyse how the dimensional reason model compares to the reduction model, we design a case study set in the scenario of the nanny robot. The dataset uses five hand-picked dimensions and contains 20 autogenerated cases. The consistency of the case base and applicable decision constraints are evaluated manually as well as automatically using our model implementation. This enables verifying the correctness of the implementation. The case study provides a set of examples that are closer to real-world scenarios in scope. We discuss the differing results between the models.

While this thesis addresses the general principles of case-based reasoning, it is heavily influenced by papers written for reasoning within the legal domain. Several systems have been developed to model and predict the outcomes of legal cases. Notable among these are HYPO [RA87] and CATO [Ale03], which were among the earliest efforts to formalise legal reasoning within AI systems. Building on these foundations, this thesis aims to contribute to the development of symbolic AI systems for case-based reasoning, with a particular emphasis on logical consistency and explainability—core requirements for deploying such systems in normative domains where justification and transparency are essential.

Thesis Structure This thesis is organised as follows: In Chapter 2 we explore in-depth the two models of precedential constraint, the result model and the reason model. We first describe both of these models in the standard setting using Boolean factors, and show that the reason model has a stronger notion of constraint than the result model. We then adjust these models to the dimensional setting, show that the models collapse, and describe the revision of the reason model proposed by Horty [Hor21] to prevent the collapse of the models. Chapter 3 introduces the reduction model in both the standard and the dimensional setting. We show how it compares to the existing models, and how it improves upon the revised dimensional reason model. An implementation of the dimensional reason model and of the dimensional reduction model using ASP is provided in Chapter 4, and a case study to compare these models in the scenario of the nanny robot is presented and discussed in Chapter 5. In Chapter 6 we discuss related work, and in Chapter 7 we conclude the thesis and mention potential future work.

Background: Modelling Precedential Constraint

In this chapter, we will explore different approaches to modelling precedential constraint. First, we will explore different ways of encoding information. A simple approach is to think of factual information as Boolean pieces of information where a single factor is either present, or not. For example, a child either read a book, or it did not. We will refer to this approach of encoding information as the *standard setting*. But, we could also think of factual information them as continuous or discrete values. Instead of claiming that a child did or did not read a book, we could state how many pages a child read in a book to encode information about a child's reading activity. We will refer to this approach as the *dimensional setting*.

Apart from encoding information, we also have to consider how decisions are reached and enforced. Given a fact situation, the *result model* simply compares the strength of information present for a side with the information present in previous cases [Ale89, Hor04, Hor17]. In this model, all pieces of information present in a case are considered equally important for the decision. However, the result model cannot distinguish between the factual information that is important to the decision and the factual information that did not influence the decision at all. To single out the important pieces of information, a decision in the the *reason model* requires a reason that justifies it [HBC12, CH22b, Hor04].

We will describe and compare the result model and the reason model of constraint. First, we will describe these models in the standard setting using Boolean factors. Following the standard setting, we describe the dimensional setting in a similar fashion. Since many definitions are shared in both the standard and dimensional setting, we will explicitly refer to some definitions as *standard* or *dimensional*. All other definitions apply to both models. Since this chapter contains many propositions from the literature, to increase

readability only relevant proofs are shown. All omitted proofs can be found in the appendix. We will use a running example to better illustrate the models in examples.

Running Example: The Nanny Robot

As mentioned earlier, this is a hypothetical scenario first introduced by Canavotto and Horty [CH22b]. In this scenario, a family owns a nanny robot tasked with deciding whether a child is allowed to stay up past its bedtime. To make such decisions appropriately, the robot must understand the reasoning behind the parents' past decisions. For this, it is provided with a database of past bedtime decisions. The nanny robot should be able to infer the parents' preferences and apply them to new, unseen situations. We will refer to the parents as the plaintiff, and to the child as the defendant. The plaintiff aims for a decision that enforces the bedtime where the defendant aims for a decision that does not enforce the bedtime.

In the following chapter, we will base all of our examples on this scenario. There will be differences in factual representation between the standard setting and the dimensional setting. The Boolean factors and dimensions to be used for the examples will be introduced in the respective section.

2.1 The Standard Setting

In this section, we introduce both the result model and the reason model in the standard setting—i.e., using Boolean factors as the basis for reasoning—as they are described in works by Horty [Hor04, HBC12]. We will begin by presenting the definitions common to both models, followed by model-specific definitions. The definitions that are not explicitly marked as *standard*, will be reused for the dimensional setting.

We start by presenting a formal definition of the two opposing parties involved, the plaintiff and the defendant. In the models, decisions are always made in favour of one of the two adversarial parties. A win for one of them means a loss for the other one. That is, in every case presented, a decision must be rendered in favour of one of the two sides.

Definition 1 (Sides). There are precisely two adversarial parties: the plaintiff, denoted by π , and the defendant, denoted by δ . For each side, the respective other side is defined as its opposite. For side $s \in \{\pi, \delta\}$, we write \bar{s} to denote its opposing party. That is, $\bar{\pi} = \delta$ and $\bar{\delta} = \pi$.

In the standard setting, information is represented as a collection of Boolean factors. To describe strength for a side, a factor either strengthens the plaintiff π or the defendant δ in a case. If we add a factor supporting side s to a fact situation, then the fact situation becomes stronger for that side. Conversely, if we remove a factor for side s from a fact situation, it becomes weaker for side s .

Definition 2 (Factor: standard). A standard factor, or Boolean factor, supporting side s is denoted by f^s .

Definition 3 (Fact situation: standard). A standard fact situation X is a set of standard factors. For a given side $s \in \{\pi, \delta\}$, we define

$$X^s = \{f^{s'} \in X \mid s = s'\}$$

as the subset of factors in X that support side s . That is, for the opposing parties plaintiff π and defendant δ we can assert that

$$X^\pi \cup X^\delta = X \text{ and } X^\pi \cap X^\delta = \emptyset.$$

Example 4. Before we continue with further definitions, we introduce 10 standard factors to motivate the scenario of a nanny robot that is tasked to decide a child's bedtime each day, according to the parents' preferences. A decision is made either in support of the plaintiff or the defendant. To reiterate, the plaintiff π represents the parents and the defendant δ represent the child. If the plaintiff wins, then the child has to go to bed on time. If the defendant wins, then the child can stay up past its bedtime. In this scenario, we define 10 standard factors, 5 supporting the plaintiff and 5 supporting the defendant.

- Factors supporting the plaintiff:
 1. f_{noise}^π : The child was very loud.
 2. f_{protest}^π : The child protested going to bed.
 3. f_{school}^π : The child has to go to school the following day.
 4. f_{screen}^π : The child had a considerable amount of screen time.
 5. f_{sugar}^π : The child had a high sugar intake.
- Factors supporting the defendant:
 1. $f_{\text{activity}}^\delta$: The child was exposed to physical activity.
 2. f_{clean}^δ : The child cleaned up after itself.
 3. f_{coop}^δ : The child cooperated or played with its sibling or friends.
 4. $f_{\text{homework}}^\delta$: The child did its homework reliably and independently.
 5. f_{read}^δ : The child spent some time reading.

Example 5. Let

$$X_1 = \{f_{\text{protest}}^\pi, f_{\text{screen}}^\pi, f_{\text{sugar}}^\pi, f_{\text{activity}}^\delta, f_{\text{homework}}^\delta\}$$

be a fact situation. X_1 expresses that the child protested going to bed (f_{protest}^π), had a considerable amount of screen time (f_{screen}^π), and had a high sugar intake (f_{sugar}^π). However, the child was also exposed to physical activity ($f_{\text{activity}}^\delta$), and did its homework

reliably and independently ($f_{\text{homework}}^\delta$). The first three constitute the factors supporting the plaintiff—the parents—and are expressed by

$$X_1^\pi = \{f_{\text{protest}}^\pi, f_{\text{screen}}^\pi, f_{\text{sugar}}^\pi\}.$$

Any factor added to X_1^π will strengthen the case for the parents. Conversely, removing factors from X_1^π will weaken the case for the parents. Likewise, the last two factors in X constitute the factors supporting the defendant—the child. They are expressed by

$$X_1^\delta = \{f_{\text{activity}}^\delta, f_{\text{homework}}^\delta\}.$$

Again, adding factors to X_1^δ strengthens the case for the child and removing factors weakens it for the child.

The described notions are shared between both the result model and the reason model. We will now continue with definitions specific to each model, starting with the result model, and then moving on to the reason model.

2.1.1 The Result Model

The result model enforces constraint by *a fortiori* information. A fortiori reasoning uses solely the strength of an argument to reach a decision. Thus, if a decision is to be made on a new fact situation that is stronger for a side than a previous fact situation decided for that side, then the new fact situation also has to be decided for that side.

Knowledge Representation We will now introduce definitions required to represent knowledge and reason within the result model. We start with the notion of a *case* that captures a decision in a past situation. Since for a decision only factual information present in a case is considered, a case in the result model holds information regarding (1) the fact situation, and (2) the winning side chosen.

Definition 4 (Result model case). Let X be a fact situation and s be a side. We say that

$$c = \langle X, s \rangle$$

is a case in the result model. We also define the following functions

$$\text{Facts}(c) = X \text{ and } \text{Outcome}(c) = s.$$

Example 6. Let

$$c_1 = \langle \{f_{\text{noise}}^\pi, f_{\text{protest}}^\pi, f_{\text{clean}}^\delta\}, \delta \rangle$$

be a case. Here, the factors $f_{\text{noise}}^\pi, f_{\text{protest}}^\pi, f_{\text{clean}}^\delta$ are present, and the defendant δ —i.e., child—was chosen as the winning side. Semantically, this means that in a situation where the child was noisy (f_{noise}^π), the child protested (f_{protest}^π), but the child also cleaned up after itself (f_{clean}^δ), the parents decided to extend the bedtime of the child.

We refer to a collection of cases as the *case base*. The case base will act as our knowledge base that enables us to make and justify decisions. When defining a case base, we are interested in two tasks:

1. The notion of *consistency* refers to the fact that no two cases in the case base have conflicting decisions. While the exact definition depends on the setting and model, as a general idea, if case c is stronger for side s than case c' , but case c was decided for side \bar{s} although case c' was decided for side s , a case base could be considered inconsistent. For any case base, we want to be able to verify if it is consistent or not.
2. Assume we have a consistent case base. If we are presented a new fact situation, we want to check what *decisions* are possible. Precisely, we want to check if a decision in favour of a side is forced, or if it is possible to decide in favour of both sides, without causing any inconsistency. If a decision is forced for one side, we refer to it as being *constrained*.

Definition 5 (Result model case base). We say that

$$\Gamma = \{c_1, c_2, \dots, c_n\}$$

is a case base in the result model, where c_1, c_2, \dots, c_n are result model cases and $n \geq 0$.

Consistency To allow enforcing consistency in a standard result model case base, we need to enable comparing the strength for a side of two standard fact situations. Since every standard fact situation contains factors supporting side s and factors supporting side \bar{s} , we can make it stronger for side s either by adding factors supporting side s or by removing factors supporting side \bar{s} . Thus, we can say that standard fact situation X is stronger for side s than standard fact situation Y , if X contains all factors supporting s that Y also contains, and it contains no factor supporting \bar{s} that Y does not contain.

Definition 6 (Strength for a side in a fact situation: standard). Given standard fact situations X and Y , we say that X is at least as strong for side s as Y —written $Y \leq^s X$ —if and only if

$$Y^s \subseteq X^s \text{ and } X^{\bar{s}} \subseteq Y^{\bar{s}}.$$

This notion is crucial to a fortiori reasoning, as it enables the direct comparison of arguments. $Y \leq^s X$ states that X is at least as strong as Y , because it contains at least as many factors supporting s as Y ($Y^s \subseteq X^s$) and at most as many factors supporting \bar{s} as Y ($X^{\bar{s}} \subseteq Y^{\bar{s}}$). We can also easily see that strength for a side works in reverse too: If a standard fact situation X is stronger for side s than a standard fact situation Y , then Y is stronger for side \bar{s} than X . More precisely, $Y \leq^s X$ if and only if $X \leq^{\bar{s}} Y$.

Example 7. Let

$$X_2 = \{f_{\text{school}}^\pi, f_{\text{sugar}}^\pi, f_{\text{clean}}^\delta\} \text{ and } X_3 = \{f_{\text{sugar}}^\pi, f_{\text{clean}}^\delta, f_{\text{read}}^\delta\}$$

be fact situations. Clearly $X_3 \leq^\pi X_2$, because $X_3^\pi \subseteq X_2^\pi$ and $X_2^\delta \subseteq X_3^\delta$.

Finally, we want to decide whether a case base is consistent or not. When we say that a case base is inconsistent, we say that a decision made for two opposing cases $c = \langle X, s \rangle$ and $c' = \langle Y, \bar{s} \rangle$ should actually have been made the other way around—that is, c should have been decided in favour of \bar{s} and c' should have been decided in favour of s . This happens when $X \leq^s Y$, because according to the decision in c , the fact situation Y would be at least as good a fit for a decision in favour of side s as the fact situation X . But conversely, according to the decision in c' , the fact situation X would be at least as good a fit for a decision in favour of side \bar{s} as the fact situation Y . We use this idea to model consistency.

Definition 7 (Result model consistency). Let Γ be a case base in the result model. We say that Γ is inconsistent if and only if there are two opposing cases $c, c' \in \Gamma$ —i.e., $c = \langle X, s \rangle$ and $c' = \langle Y, \bar{s} \rangle$ —such that $X \leq^s Y$. Equivalently, a case base is consistent if and only if for all opposing cases $c, c' \in \Gamma$, where $\langle X, s \rangle$ and $c' = \langle Y, \bar{s} \rangle$, we have $X \not\leq^s Y$.

Example 8. Let

$$\Gamma_1 = \{c_4, c_5\}$$

be a case base with cases

$$\begin{aligned} c_4 &= \langle X_4, \pi \rangle \text{ where } X_4 = \{f_{\text{school}}^\pi, f_{\text{activity}}^\delta, f_{\text{read}}^\delta\}, \text{ and} \\ c_5 &= \langle X_5, \delta \rangle \text{ where } X_5 = \{f_{\text{school}}^\pi, f_{\text{screen}}^\pi, f_{\text{activity}}^\delta\}. \end{aligned}$$

This means that in case c_4 the child needs to go to school the next day (f_{school}^π), had some physical activity ($f_{\text{activity}}^\delta$), and read throughout the day (f_{read}^δ). In contrast, for c_5 the child needs to go school the next day (f_{school}^π), had a considerable amount of screen time (f_{screen}^π), and also some physical activity ($f_{\text{activity}}^\delta$). c_4 was decided in favour of the parents (π), while c_5 was decided in favour of the child (δ). This is an example of an inconsistent case base, because

$$X_4 \leq^\pi X_5.$$

This expresses that although X_5 is stronger than X_4 for π , and X_4 is stronger than X_5 for δ , case c_5 that X_5 was decided in favour of δ and case c_4 that X_4 was decided in favour of π .

Constraint The notion of consistency is easily translated to new fact situations. If a new fact situation X is at least as strong as some case $\langle Y, s \rangle \in \Gamma$ —that is, $Y \leq^s X$ —deciding the fact situation in favour of \bar{s} and adding it to the case base, would render the case base inconsistent. Therefore, if $Y \leq^s X$ holds for some case $\langle Y, s \rangle \in \Gamma$, we are forced to decide X in favour of s too. We refer to this as the *decision constraint*. If there is no case that constrains the decision of a new fact situation, we are allowed to freely pick a side as the winning side.

Definition 8 (Result model decision constraint and permission). Let Γ be a case base and X be a fact situation. A decision in favour of side s is required if and only if there exists a case $c \in \Gamma$ such that

$$\text{Outcome}(c) = s \text{ and } \text{Facts}(c) \leq^s X.$$

Likewise, a decision in favour of side s is permitted if and only if there is no case such that

$$\text{Outcome}(c) = \bar{s} \text{ and } \text{Facts}(c) \leq^{\bar{s}} X.$$

Example 9. Let

$$\Gamma_2 = \{c_6, c_7\}$$

be a case base with opposing cases

$$\begin{aligned} c_6 &= \langle X_6, \pi \rangle \text{ where } X_6 = \{f_{\text{noise}}^\pi\}, \text{ and} \\ c_7 &= \langle X_7, \delta \rangle \text{ where } X_7 = \{f_{\text{activity}}^\delta, f_{\text{read}}^\delta\}. \end{aligned}$$

We can easily see that Γ_2 is consistent, since $X_7 \leq^\pi X_6$ and $X_6 \leq^\delta X_7$. In the scenario of the nanny robot, the robot has knowledge of the two past cases c_6 and c_7 that are stored in a dataset representing case base Γ_2 . It now is asked to make a decision for a new fact situation

$$X_8 = \{f_{\text{noise}}^\pi, f_{\text{protest}}^\pi\}.$$

In this scenario, the child was noisy (f_{noise}^π), and protested (f_{protest}^π). Is the robot required to send the child to bed early by making a decision in favour of π ? Yes, because in case c_6 a child was sent to bed early on the basis of just being noisy. Since this new fact situation X_8 is stronger than X_6 , that is,

$$X_6 \leq^\pi X_8,$$

the child now has to be sent to bed early as well.

Example 10. We reuse Γ_2 from the previous example, but this time we consider fact situation

$$X_9 = \{f_{\text{activity}}^\delta\}.$$

In this scenario, the child had physical activity ($f_{\text{activity}}^\delta$). We can see that it is not required to send the child to bed early by deciding X_9 in favour of π , because for c_6 we have

$$X_6 \not\leq^\pi X_9.$$

But it could still be that it is required to extend the bedtime of the child on the basis of c_7 . So, is required to decide X_9 in favour of δ ? Also no, because for c_7 we have

$$X_7 \not\leq^\delta X_9$$

as well. Since we are not required to decide in favour of either side, no constraint is applicable to the situation. This means that the case base Γ_2 does not impose any restriction on how X_9 should be decided. Both decisions are permitted in such a scenario.

To simplify matters, we can tie the notions of constraint and permission to the notion of consistency.

Proposition 1 (Result model decision constraint and permission through consistency). Let Γ be a consistent case base and X be a fact situation. A decision is constrained to side s if and only if $\Gamma \cup \{\langle X, \bar{s} \rangle\}$ is inconsistent. Equivalently, a decision in favour of side s is permitted if and only if $\Gamma \cup \{\langle X, s \rangle\}$ is consistent.

For completeness, we will also assert that a decision in favour of at least one of the two sides is always possible. This merely is to show that when a fact situation is presented to a consistent case base it cannot occur that no decision can be made at all.

Proposition 2 (Result model decision possible). Let Γ be a consistent case base and X be a fact situation. A decision in favour of at least one of the two sides π and δ is always possible.

The result model in the standard setting is concluded with these definitions and propositions. This model enables a framework for reasoning about factual strength to enable checking for consistency between cases as well as checking for decision constraints on new fact situations. Using this framework, if a nanny robot is provided a large enough case base, it can make a fortiori decisions justified by specific cases in its case base. What decisions it will be able to make depends on the fact situations present. It may occur that merely comparing strength of factual information is too weak for a system of constraint we would like to employ. For a more constraining model, in the next section we will now look at the reason model in the standard setting. It builds on top of the result model of constraint and enables to add further restrictions to the decision making.

2.1.2 The Reason Model

The reason model builds on top of the result model. Instead of relying purely on factual strength in the defined cases, each decision now requires a justification in this model. A justification describing what influenced the decision enables an agent to infer a *rule* that can be used more flexibly for new fact situations. In the reason model, we describe these rules as consisting of two parts:

1. A *side* that is chosen as the winning side for the fact situation.
2. A *reason* consisting of a subset of factors that are chosen as the relevant pieces of information.

Knowledge Representation A reason should separate the relevant from the irrelevant factors. In real-life situations, not every piece of information is important in a case. For example, sometimes there are factors so important that they make all the other factors irrelevant. To model this notion, if a factor is not used in the reason, we think of it as

not influencing the decision in that specific case despite its presence. We formalise this idea as follows.

Definition 9 (Reason: standard). We say that

$$Q = \{f_1^s, f_2^s, \dots, f_n^s\}$$

is a standard reason supporting side s , where $n \geq 0$. That is, Q only contains factors supporting side s .

Definition 10 (Rule). We say that $r = Q \rightarrow s$ is a rule supporting side s , where Q is a reason supporting side s . A rule expresses a decision for a side along with a reason for why that decision was reached. We define

$$\text{Premise}(r) = Q \text{ and } \text{Conclusion}(r) = s.$$

By our definition, only factors that support side s —i.e., f^s —can be used in reasons supporting side s . Using a factor that supports the opposite side \bar{s} —i.e., $f^{\bar{s}}$ —for a rule that supports side s does not make sense from a semantical perspective, as the presence of such a factor only ever strengthens the opposing side.

Example 11. An example of a reason supporting π is

$$Q_1 = \{f_{\text{protest}}^\pi, f_{\text{school}}^\pi\}.$$

Further, a rule using that reason looks like

$$Q_1 \rightarrow \pi.$$

Semantically, the rule states that a decision in favour of π is reached, because factors f_{protest}^π and f_{school}^π are present in a fact situation. That is, the child protested and the child has to attend school tomorrow. In contrast,

$$Q_1 \rightarrow \delta$$

is not a rule, since Q_1 is not a reason supporting δ . Also,

$$Q_2 = \{f_{\text{school}}^\pi, f_{\text{activity}}^\delta\}$$

is not a reason, since a reason can only contain factors supporting one side.

Instead of comparing case strength between fact situations, we now use reasons and rules as our central notion for consistency and decisions. In particular, we want to check if a reason is satisfied by a fact situation. In the standard setting, this notion is straightforward.

Definition 11 (Reason satisfaction: standard). We say that a standard reason Q is satisfied by some standard fact situation X —written $X \models Q$ —if and only if every factor contained in Q is present in X , i.e.,

$$Q \subseteq X.$$

Intuitively, a reason Q is satisfied by a fact situation X if and only if every factor in Q is also present in X . A reason Q supporting side s can only be considered for fact situation X , if it satisfies the reason, since the reason should justify the decision in favour of s .

Like we did with fact situations in the result model, we also want to enable comparing the strength of reasons. Strength, or rather *entailment*, of reasons can be modelled by comparing the fact situations that satisfy it. The stronger a reason—that is, the more factors used in a reason—the fewer fact situations will satisfy it. Thus, a reason W entails another reason Z , if Z satisfies a superset of fact situations that W satisfies. This means that if W satisfies a fact situation, Z satisfies it too.

Definition 12 (Reason entailment). Let W and Z be two reasons. We say that W entails Z —written $W \Vdash Z$ —if and only if $X \models Z$ whenever $X \models W$ for any fact situation X .

In the standard setting, $W \Vdash Z$ for reasons W and Z is equivalent to $Z \subseteq W$. That is, a reason W that contains at least as many factors as Z will satisfy at most as many fact situations as Z , and is therefore considered at least as strong for the side the reasons support.

We now continue with defining cases and case bases in the reason model. As previously described, a case in this model has to include a rule for the decision.

Definition 13 (Reason model case). Let X be a fact situation, s be a side and $r = Q \rightarrow s$ be a rule supporting side s with some reason Q such that $X \models Q$. We say that

$$c = \langle X, r, s \rangle$$

is a case in the reason model. In addition to $\text{Facts}(c) = X$ and $\text{Outcome}(c) = s$, we also define

$$\text{Rule}(c) = r.$$

Definition 14 (Reason model case base). We say that

$$\Gamma = \{c_1, c_2, \dots, c_n\}$$

is a case base in the reason model, where c_1, c_2, \dots, c_n are reason model cases and $n \geq 0$.

Example 12. Let $c_{10} = \langle X_{10}, r_{10}, s \rangle$ be a case where

$$\begin{aligned} X_{10} &= \{f_{\text{school}}^\pi, f_{\text{screen}}^\pi, f_{\text{activity}}^\delta, f_{\text{coop}}^\delta, f_{\text{homework}}^\delta\}, \\ r_{10} &= Q_{10} \rightarrow \delta, \text{ and} \\ Q_{10} &= \{f_{\text{coop}}^\delta, f_{\text{homework}}^\delta\}. \end{aligned}$$

In this scenario, the child has to attend school the next day (f_{school}^π), had a significant amount of screen time (f_{screen}^π), spent some time doing physical activity ($f_{\text{activity}}^\delta$), cooperated with its sibling throughout the day (f_{coop}^δ), and did its homework in a timely manner ($f_{\text{homework}}^\delta$). The decision is reached in favour of the child (δ). The reason for the decision is said to be the factors describing that the child cooperated with its sibling throughout the day (f_{coop}^δ), and that it did its homework in a timely manner ($f_{\text{homework}}^\delta$). This means that the parents' decision to extend the bedtime was not influenced by the factor describing that the child did some physical activity throughout the day ($f_{\text{activity}}^\delta$).

Consistency Like in the result model, we require a measure of consistency and decisions constraints. But instead of comparing factual strength, we compare reasons. We will start by adjusting the notion of consistency. For this, we construct a priority ordering of reasons for every case that models which reason, if any, is prioritised according to the decision at hand.

Definition 15 (Priority ordering derived from case). Let $c = \langle X, r, s \rangle$ be a case in the reason model and W and Z be two reasons supporting opposite sides where W supports side \bar{s} and Z supports side s . The relation $<_c$ represents the priority ordering of reasons for case c . We say that $W <_c Z$ if and only if

$$X \models W \text{ and } Z \Vdash \text{Premise}(r).$$

Let us further explore this definition by again considering some case $c = \langle X, r, s \rangle$, reason W for side \bar{s} , and reason Z for side s such that $W <_c Z$ holds. From the first condition $X \models W$ we know that fact situation X satisfies W , thus $c' = \langle X, W \rightarrow \bar{s}, \bar{s} \rangle$ would be a valid case for the given fact situation X . From the second condition $Z \Vdash \text{Premise}(r)$ we know that reason Z is at least as strong as reason $\text{Premise}(r)$ for side s . Let us consider $W <_c \text{Premise}(r)$ first: Trivially $X \models \text{Premise}(r)$ and $\text{Premise}(r) \Vdash \text{Premise}(r)$, so $W <_c \text{Premise}(r)$ holds. Since case c was decided in favour of side s using $\text{Premise}(r)$, reason $\text{Premise}(r)$ evidently must take precedence over W , otherwise the case would have been decided as given in c' . We now consider any Z such that $X \models Z$ and $Z \Vdash \text{Premise}(r)$. Since $X \models Z$, we know that Z is a reason that can be used to decide X in favour of s , i.e., $c'' = \langle X, Z \rightarrow s, s \rangle$ is a valid case. Since Z is at least as strong for side s as $\text{Premise}(r)$ by condition $Z \Vdash \text{Premise}(r)$, and we know that $W <_c \text{Premise}(r)$ holds, it becomes evident that any reason at least as strong as $\text{Premise}(r)$ should also take priority over W on case c . Thus, as per our definition, $W <_c Z$ holds for any Z such that $Z \Vdash \text{Premise}(r)$. More concisely, any potential reason at least as strong as $\text{Premise}(r)$ is prioritised over every reason for the opposing side that X satisfies.

Example 13. Let $c_{11} = \langle X_{11}, r_{11}, s \rangle$ be a case where

$$\begin{aligned} X_{11} &= \{f_{\text{noise}}^\pi, f_{\text{protest}}^\pi, f_{\text{sugar}}^\pi, f_{\text{clean}}^\delta, f_{\text{coop}}^\delta, f_{\text{homework}}^\delta\}, \\ r_{11} &= Q_{11} \rightarrow \pi, \text{ and} \\ Q_{11} &= \{f_{\text{noise}}^\pi, f_{\text{protest}}^\pi\}. \end{aligned}$$

That is, the children were sent to bed on time with the justification that they were loud (f_{noise}^π) and protested going to bed (f_{protest}^π). Both of these factors have to be present in X_{11} for $X_{11} \models Q_{11}$ to hold. Let us now consider two opposing reasons

$$\begin{aligned} Q_{12} &= \{f_{\text{sugar}}^\pi, f_{\text{noise}}^\pi, f_{\text{protest}}^\pi\}, \text{ and} \\ Q_{13} &= \{f_{\text{clean}}^\delta, f_{\text{coop}}^\delta\}. \end{aligned}$$

We can see that Q_{12} supports π and Q_{13} supports δ , since Q_{12} contains factors supporting π and Q_{13} contains factors supporting δ . Because $X_{11} \models Q_{13}$ and $Q_{12} \not\models Q_{11}$, we know that

$$Q_{12} <_{c_{11}} Q_{13}.$$

That means reason Q_{13} is assigned a higher priority than Q_{12} . We can easily see that, in fact, any reason that entails Q_{11} will be assigned a higher priority than Q_{12} . This, of course, also includes Q_{11} itself. Since $Q_{11} \models Q_{11}$, we also know that $Q_{12} <_{c_{11}} Q_{11}$.

With a priority ordering defined for every case, we now propagate the priority of reasons to the whole case base. We do this since a priority in a case in the case base should also reflect a priority for the whole case base.

Definition 16 (Priority ordering derived from case base). Let Γ be a case base in the reason model and W and Z be two reasons supporting opposite sides where W supports side \bar{s} and Z supports side s . The relation $<_\Gamma$ represents the priority ordering of reasons for the case base Γ . We say that $W <_\Gamma Z$ if and only if $W <_c Z$ for some case $c \in \Gamma$.

The more cases the case base contains, the more refined the priority ordering becomes. We consider the case base as a dataset of decisions performed by one entity—in our scenario, the parents. Since the parents should have a consistent idea of how decisions are to be made, we will model the consistency of the case base using the priority ordering. A case base is consistent if and only if the priority ordering of reasons imposed by the case base is consistent. Formally, we define this as follows.

Definition 17 (Reason model consistency). Let Γ be a case base in the reason model. We say that Γ is inconsistent if and only if there are two reasons W and Z supporting opposite sides such that $W <_\Gamma Z$ and $Z <_\Gamma W$. Equivalently, we say that Γ is consistent if and only if for all reasons W and Z supporting opposite sides we have $W \not<_\Gamma Z$ or $Z \not<_\Gamma W$.

If a case base Γ is inconsistent, then there are opposing reasons W and Z such that reason Z has higher priority than reason W for some case $c \in \Gamma$ while at the same time reason W has higher priority than reason Z for some other case $c' \in \Gamma$. That is, in case c reason Z is preferred over W whereas in case c' reason W is preferred over Z .

Example 14. Let $\Gamma_3 = \{c_{14}, c_{15}\}$ be a case base where

$$\begin{aligned} c_{14} &= \langle \{f_{\text{noise}}^\pi, f_{\text{sugar}}^\pi, f_{\text{activity}}^\delta, f_{\text{read}}^\delta\}, r_{14}, \pi \rangle && \text{where } r_{14} = \{f_{\text{noise}}^\pi\} \rightarrow \pi, \text{ and} \\ c_{15} &= \langle \{f_{\text{noise}}^\pi, f_{\text{screen}}^\pi, f_{\text{sugar}}^\pi, f_{\text{activity}}^\delta\}, r_{15}, \delta \rangle && \text{where } r_{15} = \{f_{\text{activity}}^\delta\} \rightarrow \delta. \end{aligned}$$

We can already see that rules r_{14} and r_{15} have very weak reasons. That is, they are weaker than most other reasons for their respective side. Let us now consider the two opposing reasons

$$\begin{aligned} Q_{16} &= \{f_{\text{noise}}^\pi, f_{\text{sugar}}^\pi\}, \text{ and} \\ Q_{17} &= \{f_{\text{activity}}^\delta, f_{\text{read}}^\delta\}. \end{aligned}$$

We can see that $\text{Facts}(c_{14})$ satisfies Q_{17} while Q_{16} is stronger for π than $\text{Premise}(r_{14})$. That is, $\text{Facts}(c_{14}) \models Q_{17}$ and $Q_{16} \Vdash \text{Premise}(r_{14})$. Therefore, we know that

$$Q_{17} <_{c_{14}} Q_{16}.$$

Conversely, since $\text{Facts}(c_{15})$ satisfies Q_{16} while r_{14} is stronger for side δ than r_{15} , that is, $\text{Facts}(c_{15}) \models Q_{16}$ and $Q_{17} \Vdash \text{Premise}(r_{15})$, we know that

$$Q_{16} <_{c_{15}} Q_{17}.$$

$Q_{17} <_{c_{14}} Q_{16}$ propagates to $Q_{17} <_{\Gamma_3} Q_{16}$, and $Q_{16} <_{c_{15}} Q_{17}$ propagates to $Q_{16} <_{\Gamma_3} Q_{17}$. Thus, from

$$Q_{17} <_{\Gamma_3} Q_{16} \text{ and } Q_{16} <_{\Gamma_3} Q_{17},$$

we deduce that Γ_3 is inconsistent.

To check whether a case base Γ is consistent by Definition 17 that was just introduced, we would need to check an infeasible amount of reasons. Therefore, Horty proposes a more efficient condition for consistency that is equivalent to Definition 17 [Hor17].

Proposition 3 (Efficient reason model consistency check). Let Γ be a case base. Then, Γ is inconsistent if and only if there are two cases $c, c' \in \Gamma$ where

$$c = \langle X, r, s \rangle \text{ and } c' = \langle Y, r', \bar{s} \rangle$$

such that

$$\text{Premise}(r') <_c \text{Premise}(r) \text{ and } \text{Premise}(r) <_{c'} \text{Premise}(r').$$

Because $\text{Premise}(r) \Vdash \text{Premise}(r)$ and $\text{Premise}(r') \Vdash \text{Premise}(r')$ always hold trivially, the statements are respectively equivalent to

$$X \models \text{Premise}(r') \text{ and } Y \models \text{Premise}(r).$$

Because Definition 17 and Proposition 3 are equivalent conditions for consistency, we can use Proposition 3 to efficiently check consistency of a case base in a program, as it only requires to check the priority ordering of reasons on all combinations of two cases. Instead of checking the priority ordering of exponentially many reasons, we only have to consider polynomially many.

Constraint We continue with defining decision constraints and permissions in the reason model. Decision constraints are defined slightly differently compared to the result model since a decision requires that a new fact situation is also assigned a rule in addition to a winning side. For this, we first define how rules can be selected to reach a decision.

Definition 18 (Reason model rule selection). Let Γ be a case base and X be a fact situation. Rule $r = Q \rightarrow s$ with reason Q such that $X \models Q$ can be selected to decide in favour of side s if and only if

$$\Gamma \cup \{\langle X, r, s \rangle\}$$

is a consistent case base.

In the result model the decision to be performed was always binary. Either π or δ were chosen for a winning side. In the reason model, however, many different rules may support a decision in favour of π , and just as many rules may support a decision in favour of δ , as long as fact situation X satisfies the reason in the proposed rules. Nonetheless, the reason model also defines the decision constraint that tests which rules can be selected. We define it as follows.

Definition 19 (Reason model decision constraint and permission). Let Γ be a consistent case base and X be a fact situation. A decision is constrained to side s if and only if all rules r that can be selected support side s —i.e., $\text{Outcome}(r) = s$. Equivalently, a decision in favour of side s is permitted if and only if there is some rule r that can be selected to support side s .

So, a decision in the reason model is constrained to side s if and only if all rules from the set of selectable rules for fact situation X support the side s . All rules that support the opposite side \bar{s} cause the case base to become inconsistent. That is, for any rule $Q \rightarrow \bar{s}$ the case base $\Gamma \cup \{\langle X, Q \rightarrow \bar{s}, s \rangle\}$ is inconsistent. Equivalently, if there exists at least one selectable rule, the decision is permitted.

Example 15. Let $\Gamma_4 = \{c_{18}, c_{19}\}$ be a consistent case base containing the two opposing cases

$$\begin{aligned} c_{18} &= \langle \{f_{\text{noise}}^\pi, f_{\text{screen}}^\pi, f_{\text{sugar}}^\pi, f_{\text{read}}^\delta\}, r_{18}, \pi \rangle & \text{where } r_{18} &= \{f_{\text{noise}}^\pi, f_{\text{screen}}^\pi\} \rightarrow \pi, \text{ and} \\ c_{19} &= \langle \{f_{\text{sugar}}^\pi, f_{\text{activity}}^\delta, f_{\text{homework}}^\delta\}, r_{19}, \delta \rangle & \text{where } r_{19} &= \{f_{\text{activity}}^\delta\} \rightarrow \delta. \end{aligned}$$

Let us now consider a new fact situation

$$X_{20} = \{f_{\text{activity}}^\delta\}$$

that has to be decided. Is it required to decide X_{20} in favour of π because of Γ_4 ? No, to be able to decide X_{20} in favour of π , we require a rule

$$r_{20} = Q_{20} \rightarrow \pi$$

with some reason Q_{20} such that $X_{20} \models Q_{20}$. Either we pick an empty reason $Q_{20} = \emptyset$ which X_{20} naturally satisfies, or we pick $Q_{20} = \{f_{\text{activity}}^\delta\}$ since $f_{\text{activity}}^\delta$ is the only factor present in X_{20} . But in both of these instances case base $\Gamma_5 = \Gamma_4 \cup \{c_{20}\}$ is inconsistent with $c_{20} = \langle X_{20}, r_{20}, \pi \rangle$, because

$$\text{Premise}(r_{19}) <_{\Gamma_5} Q_{20} \text{ and } Q_{20} <_{\Gamma_5} \text{Premise}(r_{19}).$$

Particularly, the first condition comes from $\text{Premise}(r_{19}) <_{c_{20}} Q_{20}$ which follows from $X_{20} \models \text{Premise}(r_{19})$ and $Q_{20} \Vdash Q_{20}$, and the second condition comes from $Q_{20} <_{c_{19}} \text{Premise}(r_{19})$ which follows from $\text{Facts}(c_{19}) \models Q_{20}$ and $\text{Premise}(r_{19}) \Vdash \text{Premise}(r_{19})$. Thus, a decision in favour of π is not possible. We will now pick a rule to decide in favour of δ . For example, to decide in favour of δ , we could pick rule

$$r_{21} = \{f_{\text{activity}}^\delta\} \rightarrow \delta,$$

since case base $\Gamma_4 \cup \{\langle X_{20}, r_{21}, \delta \rangle\}$ is consistent due to

$$\text{Facts}(c_{18}) \not\models \{f_{\text{activity}}^\delta\}.$$

Particularly, this follows from Proposition 3, which states that a case base Γ is inconsistent if and only if there are two cases $\langle X, r, s \rangle, \langle Y, r', \bar{s} \rangle \in \Gamma$ such that $X \models \text{Premise}(r')$ and $Y \models \text{Premise}(r)$. Because the rule r_{21} in favour of δ can be selected a decision in favour of δ is permitted, and because no rule in favour of π can be selected, a decision in favour of δ is required as well.

Example 16. Continuing on case base Γ_4 from the previous example, we now consider fact situation

$$X_{22} = \{f_{\text{protest}}^\pi, f_{\text{activity}}^\delta\}.$$

In this scenario, is it possible that the reason model decides in favour of the π and the δ alike? Like before, we could use rule

$$r_{21} = \{f_{\text{activity}}^\delta\} \rightarrow \delta$$

to decide in favour of δ , since $\text{Facts}(c_{18}) \not\models \{f_{\text{activity}}^\delta\}$, thus ensuring the consistency of case base $\Gamma_4 \cup \{\langle X_{11}, r_{21}, \delta \rangle\}$. That is, a decision in favour of δ is again permitted. However, for this fact situation X_{22} it is also possible to decide in favour of π by using rule

$$r_{22} = \{f_{\text{protest}}^\pi, f_{\text{activity}}^\delta\} \rightarrow \pi$$

for example. We can select rule r_{22} because

$$\text{Facts}(c_{19}) \not\models \text{Premise}(r_{22})$$

ensures the consistency of case base $\Gamma_4 \cup \{\langle X_{22}, r_{22}, \pi \rangle\}$.

To reiterate, Definition 19 states that a decision in favour of side s can be made if and only if a rule r supporting side s can be selected such that the case base is consistent. To claim that a decision can be made in favour of a side, we simply need to find one specific rule that verifies this claim. Given n factors in a fact situation where $k \leq n$ factors support s , there may be 2^k potential reasons for a rule in favour of s . Since checking exponentially many reasons is infeasible, we are interested in a smarter method to verify that a decision in favour of a side can be made.

The stronger a reason is, the fewer cases it should constrain. Our intuition here is that choosing the strongest reason for a fact situation X is sufficient to check if a decision is permissible. In the standard setting, the strongest reason for side s and fact situation X is X^s , since no further factor can be added to make reason X^s even stronger such that $X \models X^s$. We will now assert this claim.

Proposition 4 (Efficient reason model decision constraint and permission: standard). Let Γ be a consistent standard case base and X be a standard fact situation. A decision is constrained to side s if and only if

$$\Gamma \cup \{\langle X, X^{\bar{s}} \rightarrow \bar{s}, \bar{s} \rangle\}$$

is inconsistent. Equivalently, a decision in favour of s is permitted if and only if

$$\Gamma \cup \{\langle X, X^s \rightarrow s, s \rangle\}$$

is consistent.

Proof. Let Γ be a consistent standard case base and X be a standard fact situation.

Let us first show how decision constraints can be efficiently tested.

“ \implies ” Assume that every rule r that can be selected to decide X favours side s . Then, $X^{\bar{s}} \rightarrow \bar{s}$ is not a selectable rule, i.e., $\Gamma \cup \{\langle X, X^{\bar{s}} \rightarrow \bar{s}, \bar{s} \rangle\}$ is inconsistent.

“ \impliedby ” Assume that there is some rule $r = Q \rightarrow \bar{s}$ that can be selected to decide X in favour of \bar{s} . Then, $\Gamma \cup \{\langle X, r, \bar{s} \rangle\}$ is consistent. That is, for every $\langle Y, r', s \rangle \in \Gamma$ we have $X \not\models \text{Premise}(r')$ or $Y \not\models Q$. But if $Y \not\models Q$, then $Y \not\models X^{\bar{s}}$, since $X^{\bar{s}} \Vdash Q$. Thus, we know that $X \not\models \text{Premise}(r')$ or $Y \not\models X^{\bar{s}}$, from which follows that $\Gamma \cup \{\langle X, X^{\bar{s}} \rightarrow \bar{s}, \bar{s} \rangle\}$ is consistent.

Let us now show how decision permissions can be efficiently tested.

“ \implies ” Assume that there is a rule $r = Q \rightarrow s$ that can be selected to decide in favour of s on fact situation X . That is, $\Gamma \cup \{\langle X, r, s \rangle\}$ is consistent. This means that for all $\langle Y, r', \bar{s} \rangle \in \Gamma$ we have $Y \not\models Q$ or $X \not\models \text{Premise}(r')$. But if $Y \not\models Q$, then $Y \not\models X^{\bar{s}}$, since $X^{\bar{s}} \Vdash Q$. Thus, we know that $X \not\models \text{Premise}(r')$ or $Y \not\models X^{\bar{s}}$, from which follows that $\Gamma \cup \{\langle X, X^s \rightarrow s, s \rangle\}$ is consistent.

“ \impliedby ” Assume that there is no rule that can be selected to decide fact situation X in favour of s . Then, $X^s \rightarrow s$ is not a selectable rule either. \square

Proposition 4, which we just introduced, is useful since it enables the efficient verification of whether a decision in favour of a side can be made at all. Instead of testing all rules, testing rule $X^s \rightarrow s$ suffices for fact situation X , since every other reason Q for side s is entailed by X^s , i.e., $X^s \Vdash Q$, or equivalently $Q \subseteq X^s$. For completeness, we also propose that a decision for at least one of the two sides is always possible.

Proposition 5 (Reason model decision possible). Let Γ be a consistent case base and X be a fact situation. By the reason model, a decision in favour of at least one of the two sides π and δ is always possible.

These provide simple mechanisms for verifying whether a decision can or has to be made when comparing a fact situation X against a case base Γ . If a new fact situation X is presented, and we want to know if X can be decided in favour of s , simply checking whether rules $X^s \rightarrow s$ and $X^{\bar{s}} \rightarrow \bar{s}$ can be selected is sufficient deducing the decision permissions and constraint. If rule $X^s \rightarrow s$ can be selected such that the case base stays consistent, a decision in favour of s is permitted. If not, a decision in favour of \bar{s} is required.

Example 17. Let $\Gamma_6 = \{c_{23}\}$ be a case base containing one case

$$\begin{aligned} c_{23} &= \langle \{f_{\text{noise}}^\pi, f_{\text{sugar}}^\pi, f_{\text{screen}}^\pi, f_{\text{activity}}^\delta, f_{\text{homework}}^\delta\}, r_{23}, \pi \rangle, \text{ and} \\ r_{23} &= \{f_{\text{noise}}^\pi, f_{\text{sugar}}^\pi\} \rightarrow \pi. \end{aligned}$$

We now want to evaluate what decision permissions and constraints apply to fact situation

$$X_{24} = \{f_{\text{noise}}^\pi, f_{\text{protest}}^\pi, f_{\text{sugar}}^\pi, f_{\text{homework}}^\delta\}.$$

Can X_{24} now be decided in favour of π ? To check, we use reason X_{24}^π to generate rule

$$r_{24} = \{f_{\text{noise}}^\pi, f_{\text{protest}}^\pi, f_{\text{sugar}}^\pi\} \rightarrow \pi.$$

Evidently, $\Gamma_6 \cup \{\langle X_{24}, r_{24}, \pi \rangle\}$ is consistent, because there is no case in favour of δ . Thus, a decision in favour π is permitted for X_{24} . Let us now also check, if a decision in favour of δ is possible. For this, we use the reason X_{24}^δ to generate rule

$$r_{25} = \{f_{\text{homework}}^\delta\} \rightarrow \delta.$$

However, $\Gamma_6 \cup \{\langle X_{24}, r_{25}, \delta \rangle\}$ is inconsistent, because

$$\text{Facts}(c_{23}) \models \text{Premise}(r_{25}) \text{ and } X_{24} \not\models \text{Premise}(r_{23}).$$

This means that a decision in favour of π is not only permitted, but required as well, since a decision in favour of δ is not permitted.

2.1.3 Result Model vs Reason Model

The reason model was designed with the requirement that it provides a stronger notion of constraint compared to the result model. A stronger notion of constraint means that there are decisions which are constrained by the reason model, but not by the result model. Of course, conversely, any decision that the result model constrains should also be constrained by the reason model. But we have yet to assert these properties. We start by showing that the condition for inconsistency in the result model also applies in the reason model. This condition is given by Definition 7, which states that case base Γ is inconsistent if and only if there are two cases $\langle X, r, s \rangle, \langle Y, r', \bar{s} \rangle \in \Gamma$ such that $X \leq^s Y$.

Proposition 6 (Case inconsistency: standard). Let Γ be a standard case base containing two cases $\langle X, r, s \rangle, \langle Y, r', \bar{s} \rangle \in \Gamma$ such that $X \leq^s Y$. Then, Γ is inconsistent.

Proof. Let Γ be a standard case base containing two cases $\langle X, r, s \rangle, \langle Y, r', \bar{s} \rangle \in \Gamma$ where $X \leq^s Y$. We know that $X^s \Vdash \text{Premise}(r)$ —i.e., using X^s as a reason. Since $X \leq^s Y$, we have $Y^{\bar{s}} \subseteq X^{\bar{s}}$, and since $X^{\bar{s}} \subseteq X$, we get $X \Vdash Y^{\bar{s}}$. Together, $X^s \Vdash \text{Premise}(r)$ and $X \Vdash Y^{\bar{s}}$ yield $Y^{\bar{s}} <_c X^s$. Conversely, we have $Y^{\bar{s}} \Vdash \text{Premise}(r')$. Since $X \leq^s Y$, we have $X^s \subseteq Y^s$, and since $Y^{\bar{s}} \subseteq Y$, we get $Y \Vdash X^s$. Together, $Y \Vdash \text{Premise}(r')$ and $Y \Vdash X^s$ yield $X^s <_{c'} Y^{\bar{s}}$. Thus, $Y^{\bar{s}} <_{\Gamma} X^s$ and $X^s <_{\Gamma} Y^{\bar{s}}$ making Γ inconsistent. \square

To be as formal as possible when comparing case bases in the reason model and the result model, we define a model translation from the reason model to the result model. This translation simply turns every case $\langle X, r, s \rangle$ in the reason model into a case $\langle X, s \rangle$ in the result model by removing the reason of every case.

Definition 20 (Model translation). Let Γ be a consistent case base in the reason model. Let Γ^* be a translated case base of Γ in the result model such that

$$\Gamma^* = \{\langle X, s \rangle \mid \langle X, r, s \rangle \in \Gamma\}.$$

Now we will show that the reason model is at least as strong as the result model regarding decision constraints. In particular, we will show that if a decision constraint is present in the result model, then it is present in the reason model as well.

Proposition 7 (Result model constrains reason model: standard). Let Γ be a consistent case base in the standard reason model. If a standard fact situation X can only be decided in favour of side s for case base Γ^* , then X can also only be decided in favour of s for case base Γ as well.

Proof. Let Γ be a consistent case base in the standard reason model, let Γ^* be its translation to the standard result model and X be a standard fact situation. Assume that X has to be decided in favour of s for Γ^* . That means that there is some $\langle Y, s \rangle \in \Gamma^*$ such that $Y \leq^s X$. Assume for contradiction that the reason model does not require a decision in favour of s . This means that there is some rule r supporting side \bar{s}

such that $\Gamma \cup \{\langle X, r, \bar{s} \rangle\}$ is consistent. Since $\langle Y, s \rangle \in \Gamma^*$, there has to be some case $\langle Y, r', s \rangle \in \Gamma$. Proposition 6 states that if for two opposing cases $c, c' \in \Gamma$ we have $\text{Facts}(c) \leq^{\text{Outcome}(c)} \text{Facts}(c')$, then Γ is inconsistent. Since $Y \leq^s X$, we can conclude that $\Gamma \cup \{\langle X, r, \bar{s} \rangle\}$ is inconsistent. Therefore, the reason model requires a decision in favour of s as well. \square

Proposition 7, which we just introduced, asserts that the reason model is at least as strong regarding the decision constraints as the result model. However, this does not prove that the reason model is actually more restrictive than the result model. To show this, we will provide an example for a fact situation where the reason model requires a decision for a side, but where the result model can make a decision in favour of either side.

Proposition 8 (Reason model does not constrain result model: standard). Let Γ be a consistent case base in the standard reason model. If a standard fact situation X can only be decided in favour of side s for case base Γ , then X is not necessarily required to be decided in favour of s for case base Γ as well.

Proof. We show this by providing an example, where the reason model requires a decision in favour of π , but the result model does not. Let

$$\Gamma_7 = \{\{\{f_{\text{noise}}^\pi\}, \{f_{\text{noise}}^\pi\} \rightarrow \pi, \pi\}\}$$

be a case base in the standard reason model and

$$X_{25} = \{f_{\text{noise}}^\pi, f_{\text{activity}}^\delta\}$$

be a standard fact situation. The reason model could make decisions in favour of π through rule

$$\{f_{\text{noise}}^\pi\} \rightarrow \pi.$$

However, it is not possible to make a decision in favour of δ , since both rules and

$$\emptyset \rightarrow \delta \text{ and } \{f_{\text{activity}}^\delta\} \rightarrow \delta$$

are inconsistent. The standard result model, on the other hand, does not require a decision in favour of either side on Γ_7^* , since for π , we have

$$\{f_{\text{noise}}^\pi\} \not\leq^\pi \{f_{\text{noise}}^\pi, f_{\text{activity}}^\delta\}.$$

and we do not have a case in favour of δ . \square

Proposition 7 shows that the reason model is at least as strong in constraint as the result model, and Proposition 8 shows that there are fact situations that are constrained in the reason model, but not in the result model. Therefore, we now know that the reason model has a stronger notion of constraint than the result model. We now have an idea of what different models of precedential constraint look like. In the following section the introduced models will build on top of the models defined in this section. The representation of information will change by replacing the Boolean factors with a more refined approach.

2.2 The Dimensional Setting

In this section, we adjust the previously introduced result model and reason model to the dimensional setting, as proposed in [Hor17, Hor21]. In this setting, Boolean factors will be replaced by *dimensions*. Rather than modelling that a piece of information is present or absent like in the standard setting, in the dimensional setting every piece of information is assigned a certain strength for a side by picking from a range of values which allows for a more refined representation of information. Before we continue, do note that all definitions from the standard setting that are not explicitly marked as *standard* definitions carry over to the dimensional setting. Those which are marked as standard definitions will be replaced by *dimensional* equivalents, marked as such. The running example of the nanny robot will also be used in this setting, but the Boolean factors, such as $f_{\text{protest}}^\pi, f_{\text{screen}}^\pi, f_{\text{activity}}^\delta, \dots$ will be replaced by dimensions once they are defined. Let us first motivate the change from the standard to the dimensional setting with an example.

Example 18. We consider the Boolean factor f_{screen}^π which denotes that a child had screen time. However, what is considered screen time is not clearly defined. One parent might argue that one hour or more should be considered. With this definition, this means that in a fact situation

$$X_{26} = \{f_{\text{screen}}^\pi\}$$

where f_{screen}^π is present a child is said to have at least one hour of screen time. Thus, a rule

$$r_{26} = \{f_{\text{screen}}^\pi\} \rightarrow \pi$$

states that bedtime should be enforced for one hour of screen time or more. However, as children age, their screen time may increase naturally, for example, through smartphones, homework on the computer, or video games, and the weight that screen time has to enforce the bedtime diminishes for parents. For a 5-year-old child one hour of screen time might be considered a lot, whereas for a 15-year-old teenager five hours of screen time is the threshold. However, we cannot model such values in a single Boolean factor such as f_{screen}^π . We could consider introducing two different Boolean factors $f_{\text{screen}:1}^\pi$ and $f_{\text{screen}:5}^\pi$ to distinguish between the two notions, however, $f_{\text{screen}:5}^\pi$ implies $f_{\text{screen}:1}^\pi$. Further, a reason containing $f_{\text{screen}:1}^\pi$ should also apply to fact situations containing $f_{\text{screen}:5}^\pi$. Since this greatly complicates how Boolean factors can be chosen, it gives rise to the idea that a new notion of factual information is needed.

The replacement to standard factors will be dimensions. Let us now consider these formally.

Definition 21 (Dimension). Every dimension d admits a totally-ordered set $[d]$ of values ranging from favouring side \bar{s} to favouring side s . The total order of values is described with relations \leq_d^s and $\leq_d^{\bar{s}}$. For any $p, q, r \in [d]$, the following properties apply:

- Reflexivity: $p \leq_d^s p$.

- Transitivity: $p \leq_d^s q$ and $q \leq_d^s r$ implies $p \leq_d^s r$.
- Antisymmetry: $p \leq_d^s q$ and $q \leq_d^s p$ implies $p = q$.
- Duality: $p \leq_d^s q$ if and only if $q \leq_d^{\bar{s}} p$.

Furthermore, we define the strict version of the order $<_d^s$ for every $p, q \in [d]$ such that

$$p <_d^s q \text{ if and only if } p \leq_d^s q \text{ and } p \neq q.$$

A dimension d describes a range of values. Each value on that range can be put into relation with all other values. Say that dimension d has values $p, q \in [d]$. We put the values into relation through the total orders \leq_d^π and \leq_d^δ . If $p \leq_d^\pi q$ applies, then value q is equally strong or stronger for the plaintiff π compared to p . Conversely, if $q \leq_d^\delta p$ applies, then value p is equally strong or stronger for the defendant δ compared to q . By duality, if q is as strong as or stronger than p for side π , then p is as strong as or stronger than q for side δ . Let us consider an example for a dimension.

Example 19. To build on top of Example 18, which outlined the limits of Boolean factors, we revisit Boolean factor f_{screen}^π , which we now want to replace with dimension d_{screen} . Dimension d_{screen} denotes the minutes of screen time in a day. Thus, the relation $\leq_{d_{\text{screen}}}^\pi$ is given as:

$$0 \leq_{d_{\text{screen}}}^\pi 1 \leq_{d_{\text{screen}}}^\pi 2 \leq_{d_{\text{screen}}}^\pi 3 \leq_{d_{\text{screen}}}^\pi \dots \leq_{d_{\text{screen}}}^\pi 10 \leq_{d_{\text{screen}}}^\pi \dots \leq_{d_{\text{screen}}}^\pi 20 \leq_{d_{\text{screen}}}^\pi \dots$$

Conversely, the relation $\leq_{d_{\text{screen}}}^\delta$ is given as:

$$\dots \leq_{d_{\text{screen}}}^\pi 20 \leq_{d_{\text{screen}}}^\pi \dots \leq_{d_{\text{screen}}}^\pi 10 \leq_{d_{\text{screen}}}^\pi \dots \leq_{d_{\text{screen}}}^\pi 3 \leq_{d_{\text{screen}}}^\pi 2 \leq_{d_{\text{screen}}}^\pi 1 \leq_{d_{\text{screen}}}^\pi 0$$

While the range of a dimension is not restricted to any kind of value, as long as we define a total ordering, we will often use integer values throughout this thesis. If a dimension d has integer values, where the bigger the value gets, the more it favours side s (and, conversely, the smaller the value gets, the more it favours side \bar{s}), then we say that dimension d *strengthens* side s . In short, the *larger* the value for d , the *stronger* the argument for side s .

Definition 22 (Dimension set). We call

$$D = \{d_1, d_2, d_3, \dots, d_n\}$$

a dimension set where $n \geq 1$. It contains dimensions $d_1, d_2, d_3, \dots, d_n$ that are relevant in a domain.

Example 20. Before we continue further adjusting the model, we introduce 5 dimensions that we will reuse throughout following examples. These dimensions also motivate the scenario of the nanny robot that is tasked to make bedtime decisions for a child. They replace the 10 Boolean factors specified in the previous section and will be used throughout the rest of the chapter as well as in following Chapters 3, 4, and 5.

- Dimensions strengthening the plaintiff:
 1. d_{protest} : The level of protest by the child, i.e.,
 - (5) “kicked”: the child screamed and kicked around,
 - (4) “screamed”: the child screamed,
 - (3) “cried” the child cried,
 - (2) “complained”: the child complained but behaved, or
 - (1) “none” the child did not protest.
 2. d_{screen} : The minutes of screen time the child had.
- Dimensions strengthening the defendant:
 1. d_{activity} : The minutes of physical activity the child had.
 2. d_{coop} : The level of cooperation by the child, i.e.,
 - (4) “played”: the child played and cooperated with its sibling,
 - (3) “ignored”: the child ignored its sibling,
 - (2) “verbal”: the child got into a verbal fight with its sibling, or
 - (1) “physical”: the child got into a physical fight with its sibling.
 3. d_{read} : The number of pages the child read.

To clarify matters, the dimensions d_{protest} and d_{screen} are called dimensions strengthening the plaintiff, because higher values strengthen the case for the parents. The dimension d_{protest} contains five constant values “kicked”, “screamed”, “cried”, “complained”, and “none”. Their strength for the plaintiff is represented by their index values, that is,

$$\text{none} \leq_{d_{\text{protest}}}^{\pi} \text{complained} \leq_{d_{\text{protest}}}^{\pi} \text{cried} \leq_{d_{\text{protest}}}^{\pi} \text{screamed} \leq_{d_{\text{protest}}}^{\pi} \text{kicked}.$$

The dimension d_{screen} contains non-negative integers where the higher the integer, the stronger the case for the plaintiff. That is, the ordering is equivalent to the natural ordering of the integers: $\leq_{d_{\text{screen}}}^{\pi} \equiv \leq$. Conversely, the dimensions d_{activity} , d_{coop} , and d_{read} are called dimensions strengthening the defendant, because higher values strengthen the case for the child. The dimension d_{coop} contains four constant values “played”, “ignored”, “verbal”, and “physical”. Their strength for the defendant is represented by their index values, that is,

$$\text{physical} \leq_{d_{\text{coop}}}^{\delta} \text{verbal} \leq_{d_{\text{coop}}}^{\delta} \text{ignored} \leq_{d_{\text{coop}}}^{\delta} \text{played}.$$

The dimensions d_{activity} and d_{read} only contain non-negative integers. Therefore, the ordering is equivalent to the natural ordering of integers: $\leq_{d_{\text{activity}}}^{\delta} \equiv \leq_{d_{\text{read}}}^{\delta} \equiv \leq$. Do note, saying that a dimension “strengthening the plaintiff” or “strengthening the defendant” is purely notational meant to simplify matters when working with dimensions throughout this paper. Dimensions themselves never strengthen either side, it is merely the value relations that enable the comparison of strength.

We will use these dimensions in the following examples that further explore the usage of the models for child nursing. In these examples, we will assume that the chosen dimension set D is picked such that $D \subseteq D_{\text{nanny}}$ where

$$D_{\text{nanny}} = \{d_{\text{protest}}, d_{\text{screen}}, d_{\text{activity}}, d_{\text{coop}}, d_{\text{read}}\}.$$

In many examples the dimension set D will not be explicitly defined. Rather, all the dimensions present in an example will implicitly define D . Dimensions that are not present in an example can be assumed not to be part of D .

With dimensions defined, we will now redefine factors and fact situations in the dimensional setting. These will replace standard (Boolean) factors and standard fact situations defined in the previous section.

Definition 23 (Value assignment: dimensional). We call

$$\langle d, p \rangle$$

a dimensional value assignment where d is a dimension and $p \in [d]$ is a value in the range of the dimension.

Definition 24 (Fact situation: dimensional). We say that

$$X = \{\langle d, p \rangle \mid d \in D, \text{pick } p \in [d]\}$$

is a dimensional fact situation on dimension set D . Every dimension $d \in D$ is assigned exactly one value $p \in [d]$, that is, for all

$$\langle d, p \rangle, \langle d, q \rangle \in X \text{ implies } p = q.$$

Furthermore, we define

$$X(d) = p \text{ if and only if } \langle d, p \rangle \in X.$$

Note that in a dimensional fact situation, every dimension defined in D is present and assigned a value. This is different from standard fact situations where facts are either present or absent. In the dimensional setting, absence of information can be described through a value on the dimension. Where a missing standard factor f_{read}^δ expresses that a child did not read throughout the day, the same information can be expressed with dimensional value assignment $\langle d_{\text{read}}, 0 \rangle$.

Example 21. Let

$$X_{27} = \{\langle d_{\text{screen}}, 30 \rangle, \langle d_{\text{protest}}, \text{complained} \rangle, \langle d_{\text{activity}}, 60 \rangle, \langle d_{\text{coop}}, \text{ignored} \rangle, \langle d_{\text{read}}, 10 \rangle\}$$

be a dimensional fact situation on dimension set D_{nanny} . From the present dimensional value assignments we deduce the following information:

1. $\langle d_{\text{protest}}, \text{complained} \rangle$: The child complained but behaved.
2. $\langle d_{\text{screen}}, 30 \rangle$: The child had a screen time of 30 minutes.
3. $\langle d_{\text{activity}}, 60 \rangle$: The child had 60 minutes of physical activity.
4. $\langle d_{\text{coop}}, \text{ignored} \rangle$: The child ignored its sibling.
5. $\langle d_{\text{read}}, 10 \rangle$: The child read 10 pages in a book.

In the dimensional reason model, we will always assume that any objects compared, e.g., fact situations, cases, case bases, etc., use the same dimension set D . This is important to keep in mind when turning and returning to definitions to be used in the dimensional setting.

The dimension is the central notion of this setting. In the following sections, we will continue with model-specific definitions of the result model and the reason model. In particular, we will redefine and reassert properties that do not directly carry over from the standard setting. Relevant definitions that carry over from the standard setting will still be mentioned and described again to improve readability.

2.2.1 The Result Model

Knowledge Representation The dimensional result model looks and acts very similar to its twin in the standard setting. Like the standard result model the dimensional result model defines cases like $\langle X, s \rangle$ where X is a fact situation and s is a winning side. But where a standard case has a standard fact situation, a dimensional case has a dimensional fact situation that is defined on a dimension set D .

For a case in the dimensional result model, Definition 4 carries over from the standard setting: Let X be a fact situation and s be a side. We say that

$$c = \langle X, s \rangle$$

is a case in the result model. We also define the following functions

$$\text{Facts}(c) = X \text{ and } \text{Outcome}(c) = s.$$

Example 22. Let

$$c_{24} = \langle \{ \langle d_{\text{screen}}, 10 \rangle, \langle d_{\text{activity}}, 180 \rangle \}, \delta \rangle$$

be a case on dimension set

$$D_1 = \{ d_{\text{screen}}, d_{\text{activity}} \}.$$

Here, the value assignments $\langle d_{\text{screen}}, 10 \rangle$ and $\langle d_{\text{activity}}, 180 \rangle$ are present. They express that the child had 10 minutes of screen time and 180 minutes of physical activity. The defendant δ —i.e., the child—was chosen as the winning side.

Also Definition 5 for case bases carries over: We say that

$$\Gamma = \{c_1, c_2, \dots, c_n\}$$

is a case base in the result model, where c_1, c_2, \dots, c_n are result model cases and $n \geq 0$.

Example 23. Let

$$\Gamma_8 = \{\langle\langle d_{\text{protest}}, \text{cried}\rangle, \langle d_{\text{read}}, 0\rangle\rangle, \pi, \langle\langle d_{\text{protest}}, \text{none}\rangle, \langle d_{\text{read}}, 50\rangle\rangle, \delta\}$$

be a case base on dimension set

$$D_2 = \{d_{\text{protest}}, d_{\text{read}}\}.$$

To match dimension set D_2 , both cases in Γ_8 have to assign values to all dimensions in D_2 in their respective fact situation.

Example 24. Let

$$D_3 = \{d_{\text{protest}}, d_{\text{coop}}\}$$

be a dimension set. Consider a case base Γ_9 matching dimension set D_3 . For example, case

$$\langle\langle d_{\text{protest}}, \text{kicked}\rangle, \langle d_{\text{coop}}, \text{physical}\rangle\rangle, \pi$$

is matching with D_3 , and thus could be added to Γ_9 , since it uses the same dimensions. On the other hand cases

$$\begin{aligned} &\langle\langle d_{\text{protest}}, \text{cried}\rangle\rangle, \pi, \text{ and} \\ &\langle\langle d_{\text{protest}}, \text{kicked}\rangle, \langle d_{\text{coop}}, \text{physical}\rangle, \langle d_{\text{read}}, 4\rangle\rangle, \delta \end{aligned}$$

are not matching with D_3 and thus could not be added to Γ_9 , since the first case is missing dimension $d_{\text{coop}} \in D_3$ and the second case has defined a superfluous dimension $d_{\text{read}} \notin D_3$.

Note that we need to know the dimension set D when specifying cases. Because we require objects to be matching with a dimension set D , all cases in a case base have to use the same dimension set D . For simplicity, in following examples D will not be defined explicitly, but implied depending on the dimensions used.

Consistency The result model of constraint compares a fortiori information. That is, it compares the strength for a side of the value assignments present in fact situations. This comparison of strength is expressed by relation \leq^s for side s . In the standard setting, we considered presence and absence of Boolean factors. But this is not sufficient for the dimensional setting, since all dimensions in D are always present with some value assigned. Instead, we compare the assigned value of each dimension $d \in D$ using the value relation \leq_d^s .

Definition 25 (Strength for a side in a fact situation: dimensional). Given dimensional fact situations X and Y , we say that X is at least as strong for side s as Y —written $Y \leq^s X$ —if and only if for every dimension $d \in D$ we have

$$Y(d) \leq_d^s X(d).$$

With this new definition of strength for a side, we can reuse Definition 7 for defining consistency: Let Γ be a case base in the result model. We say that Γ is inconsistent if and only if there are two opposing cases $c, c' \in \Gamma$ —i.e., $c = \langle X, s \rangle$ and $c' = \langle Y, \bar{s} \rangle$ —such that $X \leq^s Y$. Equivalently, a case base is consistent if and only if for all opposing cases $c, c' \in \Gamma$, where $\langle X, s \rangle$ and $c' = \langle Y, \bar{s} \rangle$, we have $X \not\leq^s Y$.

Example 25. Let

$$\begin{aligned} X_{28} &= \{ \langle d_{\text{screen}}, 70 \rangle, \langle d_{\text{activity}}, 30 \rangle \}, \text{ and} \\ X_{29} &= \{ \langle d_{\text{screen}}, 60 \rangle, \langle d_{\text{activity}}, 45 \rangle \} \end{aligned}$$

be two fact situations. For the assertion that

$$X_{29} \leq^\pi X_{28}$$

holds, we have to ensure that $X_{29}(d_{\text{screen}}) \leq_{d_{\text{screen}}}^\pi X_{28}(d_{\text{screen}})$ and $X_{29}(d_{\text{activity}}) \leq_{d_{\text{activity}}}^\pi X_{28}(d_{\text{activity}})$ hold. Inferring the values, the conditions become $60 \leq_{d_{\text{screen}}}^\pi 70$ and $45 \leq_{d_{\text{activity}}}^\pi 30$. Since larger values of d_{screen} are stronger for the plaintiff, $60 \leq_{d_{\text{screen}}}^\pi 70$ holds. Since larger values of d_{activity} are stronger for the defendant, $45 \leq_{d_{\text{activity}}}^\pi 30$ holds. Conversely,

$$X_{28} \leq^\delta X_{29}$$

also holds because of the duality of the value relations.

Constraint For constraint and permission of decisions, Definition 8 carries over: Let Γ be a case base and X be a fact situation. A decision in favour of side s is required if and only if there exists a case $c \in \Gamma$ such that

$$\text{Outcome}(c) = s \text{ and } \text{Facts}(c) \leq^s X.$$

Likewise, a decision in favour of side s is permitted if and only if there is no case such that

$$\text{Outcome}(c) = \bar{s} \text{ and } \text{Facts}(c) \leq^{\bar{s}} X.$$

Of course, like in the standard setting, we can define constraint and permission through consistency, as shown in Proposition 1: Let Γ be a consistent case base and X be a fact situation. A decision is constrained to side s if and only if $\Gamma \cup \{ \langle X, \bar{s} \rangle \}$ is inconsistent. Equivalently, a decision in favour of side s is permitted if and only if $\Gamma \cup \{ \langle X, s \rangle \}$ is consistent.

Furthermore, also Proposition 2 continues to hold: Let Γ be a consistent case base and X be a fact situation. A decision in favour of at least one of the two sides π and δ is always possible.

These definitions conclude the result model of constraint in the dimensional setting. We will now look at a few more examples regarding consistency and decisions in the dimensional result model. We will also outline how dimension sets D that are not given explicitly in the examples are implied based on the dimensions used in the fact situations.

Example 26. Let $\Gamma_{10} = \{c_{30}, c_{31}\}$ be a case base with two opposing cases

$$c_{30} = \langle X_{30}, \pi \rangle \text{ where } X_{30} = \{\langle d_{\text{screen}}, 30 \rangle, \langle d_{\text{protest}}, \text{cried} \rangle, \langle d_{\text{activity}}, 90 \rangle\}, \text{ and}$$

$$c_{31} = \langle X_{31}, \delta \rangle \text{ where } X_{31} = \{\langle d_{\text{screen}}, 120 \rangle, \langle d_{\text{protest}}, \text{kicked} \rangle, \langle d_{\text{activity}}, 30 \rangle\}.$$

Because only dimensions d_{screen} , d_{protest} , and d_{activity} occur in fact situations X_{30} and X_{31} , the dimension set

$$D_4 = \{d_{\text{screen}}, d_{\text{protest}}, d_{\text{activity}}\}$$

is implied for Γ_{10} . The example states that in c_{30} the children had 30 minutes of screen time ($\langle d_{\text{screen}}, 30 \rangle$), cried in protest ($\langle d_{\text{protest}}, \text{cried} \rangle$), and had 90 minutes of physical activity ($\langle d_{\text{activity}}, 90 \rangle$). In contrast, for c_{31} the children had 120 minutes of screen time ($\langle d_{\text{screen}}, 120 \rangle$), screamed and got physical in protest ($\langle d_{\text{protest}}, \text{kicked} \rangle$), and had 30 minutes of physical activity ($\langle d_{\text{activity}}, 30 \rangle$). c_{30} was decided in favour of the parents, while c_{31} was decided in favour of the children. This is an example of an inconsistent case base, because

$$X_{30} \leq^{\pi} X_{31} \text{ or equivalently } X_{31} \leq^{\delta} X_{30}.$$

This expresses that although X_{31} is stronger than X_{30} for π , and equivalently X_{30} is stronger than X_{31} for δ , case c_{31} , where X_{31} belongs to, was decided in favour of π and case c_{30} , where X_{30} belongs to, was decided in favour of δ .

Example 27. Let $\Gamma_{11} = \{c_{32}, c_{33}\}$ be a consistent case base with two opposing cases

$$c_{32} = \langle X_{32}, \pi \rangle \text{ where } X_{32} = \{\langle d_{\text{screen}}, 60 \rangle, \langle d_{\text{read}}, 10 \rangle\}, \text{ and}$$

$$c_{33} = \langle X_{33}, \delta \rangle \text{ where } X_{33} = \{\langle d_{\text{screen}}, 30 \rangle, \langle d_{\text{read}}, 20 \rangle\}.$$

We now consider the fact situation

$$X_{34} = \{\langle d_{\text{screen}}, 80 \rangle, \langle d_{\text{read}}, 5 \rangle\}.$$

Is it required to make a decision in favour of either side? We first consider c_{32} as a potential candidate for constraining the decision of X_{34} in favour of π . Since c_{32} has a fact situation X_{32} weaker than X_{34} for π , that is

$$X_{32} \leq^{\pi} X_{34},$$

we know that a decision in favour π is required for X_{34} . Since c_{32} requires a decision in favour π , c_{33} cannot require a decision in favour of c_{33} , given that the case base is consistent.

Example 28. We reuse Γ_{11} from the previous example, but this time we consider the fact situation

$$X_{35} = \{\langle d_{\text{screen}}, 45 \rangle, \langle d_{\text{read}}, 15 \rangle\}.$$

It is not required to decide X_{35} in favour of π , because for c_{32} we have

$$X_{32} \not\leq^{\pi} X_{35}.$$

But is it now required to decide X_{35} in favour of δ ? Also no, because for c_{33} we have

$$X_{33} \not\leq^{\delta} X_{35}$$

as well. In this case, both decisions are permissible, because no existing case enforces precedential constraint.

Another benefit of the result model using dimensions is that we can visualise these using a coordinate system that runs along the dimensions. We will provide two examples to illustrate how decision constraints and consistency can be imagined.

Example 29. Let $\Gamma_{12} = \{c_{36}, c_{37}\}$ be a consistent case base with opposing cases

$$c_{36} = \langle \{\langle d_{\text{activity}}, 30 \rangle, \langle d_{\text{read}}, 30 \rangle\}, \pi \rangle, \text{ and}$$

$$c_{37} = \langle \{\langle d_{\text{activity}}, 50 \rangle, \langle d_{\text{read}}, 25 \rangle\}, \delta \rangle.$$

Suppose we now want to check the decision constraints of the following three fact situations

$$X_{38} = \{\langle d_{\text{activity}}, 25 \rangle, \langle d_{\text{read}}, 16 \rangle\},$$

$$X_{39} = \{\langle d_{\text{activity}}, 60 \rangle, \langle d_{\text{read}}, 30 \rangle\},$$

$$X_{40} = \{\langle d_{\text{activity}}, 45 \rangle, \langle d_{\text{read}}, 10 \rangle\}.$$

By case base Γ_{12} , fact situation X_{38} is constrained by case c_{36} , and fact situation X_{39} is constrained by case c_{37} . Fact situation X_{40} is not constrained. We can visualise these results in a two-dimensional coordinate system along the dimensions. This is shown in Figure 2.1. Any fact situation that falls into the grey square bounded by c_{36} is constrained to be decided in favour of π because it is at least as strong for π as $\text{Facts}(c_{36})$. Any fact situation that falls into the grey square bounded by c_{37} is constrained to be decided in favour of δ because it is at least as strong for δ as $\text{Facts}(c_{37})$.

Example 30. Let $\Gamma_{13} = \{c_{41}, c_{42}\}$ be a case base with opposing cases

$$c_{41} = \langle \{\langle d_{\text{activity}}, 60 \rangle, \langle d_{\text{read}}, 30 \rangle\}, \pi \rangle, \text{ and}$$

$$c_{42} = \langle \{\langle d_{\text{activity}}, 50 \rangle, \langle d_{\text{read}}, 25 \rangle\}, \delta \rangle.$$

We can see that Γ_{13} is inconsistent, since $\text{Facts}(c_{42}) \leq^{\delta} \text{Facts}(c_{41})$, i.e., case c_{41} is stronger for the defendant than case c_{42} . We illustrate this in Figure 2.2. Again, the grey squares bounded by the cases c_{41} and c_{42} require decisions for their respective side. The red square is the intersection of the squares for c_{41} and c_{42} . If a fact situation were to fall into that red square, a decision in favour of both sides would be required which is why we consider these cases inconsistent.

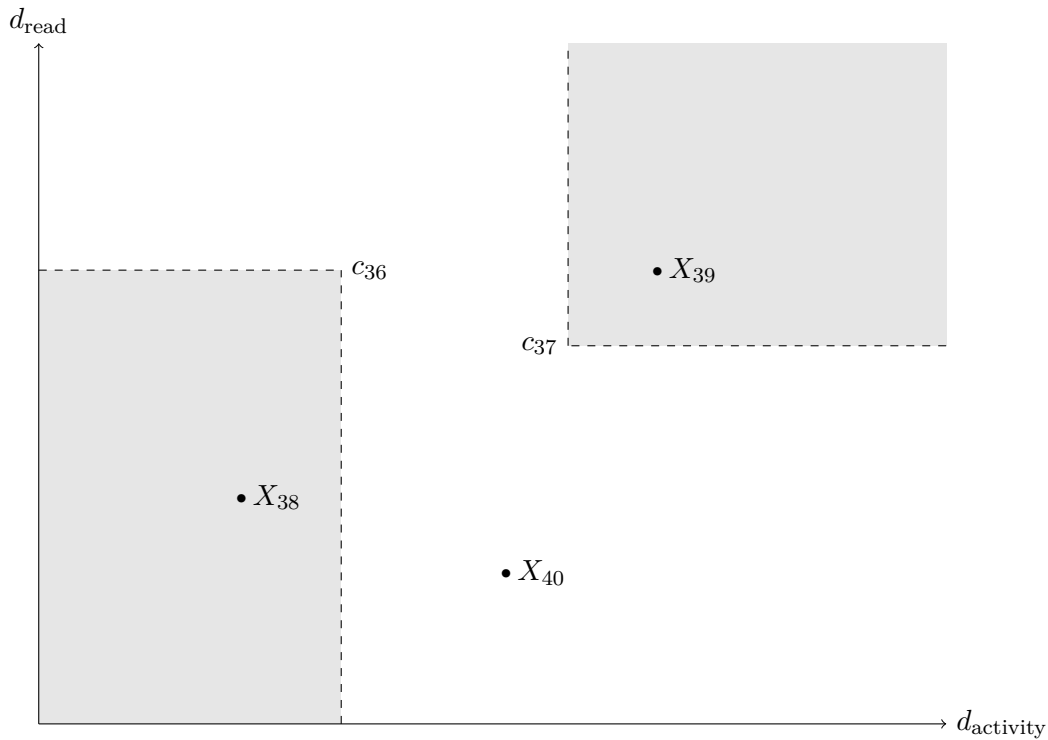


Figure 2.1: Decision constraints in the dimensional result model

2.2.2 The Reason Model

Adjusting the reason model for dimensions is not as straightforward as for the result model. First of all, it yields the challenge of specifying reasons. We will first take a look at Horty's initial approach of introducing dimensions to the reason model, and why this initial approach is problematic [Hor17].

Knowledge Representation Where a reason in the standard setting simply contains a subset of Boolean factors supporting the winning side, a reason in the dimensional setting must take into account the value ordering specified for each dimension. For this, we introduce a new notion called a *magnitude factor*.

Definition 26 (Magnitude factor). Let s be a side, d be a dimension, and $p \in [d]$ be a value in dimension d . We call $M_{d,p}^s$ a magnitude factor for dimension d supporting side s . Usually, we call p the *threshold value* of the magnitude factor.

Like value assignments, a magnitude factor $M_{d,p}^s$ consists of a dimension d , and a value p . Further, every magnitude supports a side s . Magnitude factors make up the atoms of reasons in dimensional setting.

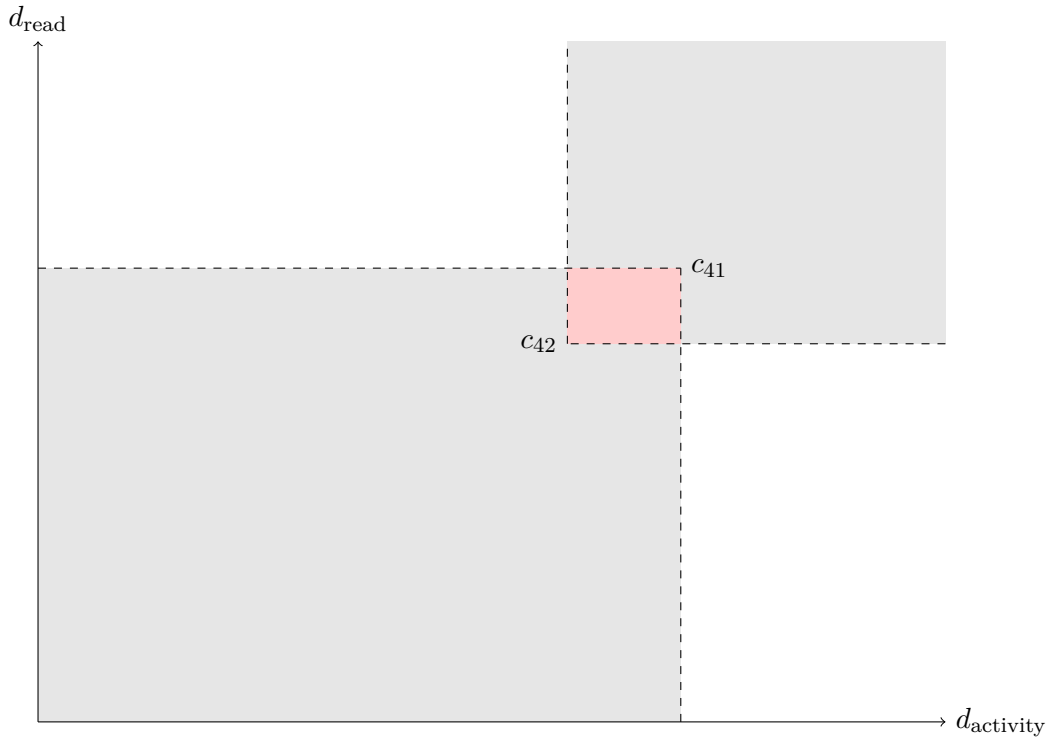


Figure 2.2: Inconsistency in the dimensional result model

Definition 27 (Reason: dimensional). Given a dimension set D , we say that Q is a dimensional reason supporting side s if and only if

$$Q \subseteq \{M_{d,p}^s \mid d \in D, p \in [d]\}$$

where

$$\text{for every } M_{d,p}^s, M_{d,q}^s \in Q \text{ we have } p = q.$$

That is, for every dimension $d \in D$ there is at most one magnitude factor present in a reason Q .

Now we can reuse Definition 10 to specify rules: We say that $r = Q \rightarrow s$ is a rule supporting side s , where Q is a reason supporting side s . A rule expresses a decision for a side along with a reason for why that decision was reached. We define

$$\text{Premise}(r) = Q \text{ and Conclusion}(r) = s.$$

In contrast to value assignments that contain all dimensions, rules and their reasons potentially contain magnitude factors that cover only a subset of the given dimensions. Each dimension $d \in D$ can have at most one magnitude factor defined. For each dimension d a magnitude factor $M_{d,p}^s$ acts as a requirement that the value assignment in X is at least as strong for side s as p . We formally define this idea as follows.

Definition 28 (Reason satisfaction: dimensional). We say that a dimensional reason Q supporting side s is satisfied by some dimensional fact situation X —written $X \models Q$ —if and only if every magnitude factor $M_{d,p}^s \in Q$ is satisfied in X , that is,

$$p \leq_d^s X(d).$$

Formally, for every $M_{d,p}^s$, p acts as a threshold value that requires a case to have a value assigned for d that is at least as strong for side s as p . That is, $M_{d,p}^s$ is a magnitude factor that requires $X(d)$ to be at least as strong as p for side s for some fact situation X . In contrast, $M_{d,q}^{\bar{s}}$ requires $X(d)$ to be at least as strong as q for side \bar{s} .

Like in the standard setting, reason entailment is given by Definition 12: Let W and Z be two reasons. We say that W entails Z —written $W \Vdash Z$ —if and only if $X \models Z$ whenever $X \models W$ for any fact situation X .

Using our adjustments as well as the definitions that carry over, we can apply Definition 13 to define a case: Let X be a fact situation, s be a side and $r = Q \rightarrow s$ be a rule supporting side s with some reason Q such that $X \Vdash Q$. We say that

$$c = \langle X, r, s \rangle$$

is a case in the reason model. In addition to $\text{Facts}(c) = X$ and $\text{Outcome}(c) = s$, we also define

$$\text{Rule}(c) = r.$$

Finally, a case base is given by Definition 14: We say that

$$\Gamma = \{c_1, c_2, \dots, c_n\}$$

is a case base in the reason model, where c_1, c_2, \dots, c_n are reason model cases and $n \geq 0$.

Given adjustments conclude an initial draft of the reason model in the dimensional setting. To represent factual information we continue to use value assignments, and to represent justifications of decisions we define magnitude factors that require a certain strength for a side for the present value assignments. We will soon see why this initial translation from the standard setting yields problems. But let us first look at an example of how cases are represented in the dimensional reason model.

Example 31. Let $c_{43} = \langle X_{43}, r_{43}, s \rangle$ be a case where

$$\begin{aligned} X_{43} &= \{ \langle d_{\text{protest}}, \text{none} \rangle, \langle d_{\text{screen}}, 40 \rangle, \langle d_{\text{activity}}, 100 \rangle, \langle d_{\text{coop}}, \text{played} \rangle, \langle d_{\text{read}}, 30 \rangle \}, \\ r_{43} &= Q_{43} \rightarrow \delta, \text{ and} \\ Q_{43} &= \{ M_{d_{\text{screen}}, 60}^\delta, M_{d_{\text{coop}}, \text{ignored}}^\delta \}. \end{aligned}$$

In this scenario, the children did not protest the parents ($\langle d_{\text{protest}}, \text{none} \rangle$), had 40 minutes of screen time ($\langle d_{\text{screen}}, 40 \rangle$), had 100 minutes of physical activity ($\langle d_{\text{activity}}, 100 \rangle$), played

and cooperated with each other throughout the day ($\langle d_{\text{coop}}, \text{played} \rangle$), and read have 30 pages in a book ($\langle d_{\text{read}}, 30 \rangle$). The decision is reached in favour of the child (δ). The reason for the decision is said to be the facts that the child had less than or equal to 60 minutes of screen time ($M_{d_{\text{screen}}, 60}^{\delta}$), and that they at least ignored each other or even played cooperated ($M_{d_{\text{coop}}, \text{ignored}}^{\delta}$). This means that the parents' decision to extend the bedtime was not influenced by the protests from the child, their amount physical activity, or the number of pages they read.

Consistency The definition for consistency carries over fully from the standard setting. Like before, Definition 15 specifies the priority ordering: Let $c = \langle X, r, s \rangle$ be a case in the reason model and W and Z be two reasons supporting opposite sides where W supports side \bar{s} and Z supports side s . The relation $<_c$ represents the priority ordering of reasons for case c . We say that $W <_c Z$ if and only if

$$X \models W \text{ and } Z \Vdash \text{Premise}(r).$$

Naturally, the priority ordering derived from a case base given by Definition 16 also applies: Let Γ be a case base in the reason model and W and Z be two reasons supporting opposite sides where W supports side \bar{s} and Z supports side s . The relation $<_{\Gamma}$ represents the priority ordering of reasons for the case base Γ . We say that $W <_{\Gamma} Z$ if and only if $W <_c Z$ for some case $c \in \Gamma$.

Finally, consistency is given by Definition 17: Let Γ be a case base in the reason model. We say that Γ is inconsistent if and only if there are two reasons W and Z supporting opposite sides such that $W <_{\Gamma} Z$ and $Z <_{\Gamma} W$. Equivalently, we say that Γ is consistent if and only if for all reasons W and Z supporting opposite sides we have $W \not<_{\Gamma} Z$ or $Z \not<_{\Gamma} W$.

We continue with an example outlining consistency in this initial translation to the dimensional setting.

Example 32. Let $\Gamma_{14} = \{c_{44}, c_{45}\}$ be a case base with opposing cases

$$\begin{aligned} c_{44} &= \langle \langle d_{\text{screen}}, 30 \rangle, \langle d_{\text{activity}}, 30 \rangle \rangle, r_{44}, \pi \quad \text{where } r_{44} = \{M_{d_{\text{activity}}, 60}^{\pi}\} \rightarrow \pi, \text{ and} \\ c_{45} &= \langle \langle d_{\text{screen}}, 10 \rangle, \langle d_{\text{activity}}, 45 \rangle \rangle, r_{45}, \delta \quad \text{where } r_{45} = \{M_{d_{\text{screen}}, 60}^{\delta}, M_{d_{\text{activity}}, 20}^{\delta}\} \rightarrow \delta. \end{aligned}$$

Rule r_{44} states that a decision in favour of the parents is made because the child had at most 60 minutes of physical activity ($M_{d_{\text{activity}}, 60}^{\pi}$), whereas rule r_{45} states that a decision is made in favour of the child because it had at most 60 minutes of screen time ($M_{d_{\text{screen}}, 60}^{\delta}$), and at least 20 minutes of physical activity ($M_{d_{\text{activity}}, 20}^{\delta}$). Since

$$\text{Facts}(c_{44}) \models \text{Premise}(r_{45}) \text{ and } \text{Facts}(c_{45}) \models \text{Premise}(r_{44}),$$

we can conclude that case base Γ_{14} is inconsistent.

Constraint Constraint and permission of decisions are again based on the notion of rule selection. Rule selection is given by Definition 18: Let Γ be a case base and X be a fact situation. Rule $r = Q \rightarrow s$ with reason Q such that $X \models Q$ can be selected to decide in favour of side s if and only if

$$\Gamma \cup \{\langle X, r, s \rangle\}$$

is a consistent case base.

Whether a decision is permitted or required depends on the rules that can be selected. This is given by Definition 19: Let Γ be a consistent case base and X be a fact situation. A decision is constrained to side s if and only if all rules r that can be selected support side s —i.e., $\text{Outcome}(r) = s$. Equivalently, a decision in favour of side s is permitted if and only if there is some rule r that can be selected to support side s .

2.2.3 Result Model vs Reason Model: Collapse

We will now see that this notion of constraint—in particular, because of the priority ordering—is not sufficient in the dimensional setting. This was observed by Horty in [Hor17] where he first introduced the dimensional reason model.

In the standard setting we could show that the reason model has a stronger notion of constraint than the result model. Unfortunately, this does not hold for the dimensional setting. We will first assert that the dimensional reason model has a notion of constraint that is at least as strong as the notion of constraint in the dimensional result model.

Proposition 9 (Result model constrains reason model: dimensional). Let Γ be a consistent case base in the dimensional reason model. If a dimensional fact situation X can only be decided in favour of side s for case base Γ^* by the dimensional result model, then X can only be decided in favour of s for case base Γ as well by the dimensional reason model.

Proof. Let Γ be a consistent case base in the dimensional reason model, let Γ^* be its translation to the dimensional result model and X be a dimensional fact situation. Assume that X has to be decided in favour of s for Γ^* . That means that there is some $\langle Y, r, s \rangle \in \Gamma$ such that $Y \leq^s X$. Assume for contradiction that the reason model does not require a decision in favour of s . This means that there is some rule r' supporting side \bar{s} such that $\Gamma \cup \{\langle X, r', \bar{s} \rangle\}$ is consistent.

Because of $Y \leq^s X$, we know that $Y(d) \leq_d^s X(d)$ for every d , and by duality $X(d) \leq_d^{\bar{s}} Y(d)$. Let $M_{d,p}^{\bar{s}} \in \text{Premise}(r')$. Since $p \leq_d^{\bar{s}} X(d)$, and since $X(d) \leq_d^{\bar{s}} Y(d)$ we have $p \leq_d^{\bar{s}} Y(d)$ by transitivity. Thus, $Y \models \text{Premise}(r')$. We can also show the converse. Let $M_{d,p}^s \in \text{Premise}(r)$. Since $p \leq_d^s Y(d)$, and since $Y(d) \leq_d^s X(d)$, we have $p \leq_d^s X(d)$ by transitivity. Thus, $X \models \text{Premise}(r)$.

Proposition 3 states that for any case base Γ is inconsistent if and only if there are two cases $\langle Y, r, s \rangle, \langle X, r', \bar{s} \rangle \in \Gamma$ such that $Y \models \text{Premise}(r')$ and $X \models \text{Premise}(r)$. Since

$Y \models \text{Premise}(r')$ and $X \models \text{Premise}(r)$ hold, we know that Γ is inconsistent. Since that is a contradiction with our assumption, we know that X has to be decided in favour of s for Γ . \square

Proposition 9 is the dimensional equivalent of Proposition 7. In the standard setting, we could also prove that the notion of constraint for the reason model is stronger than for the result model by finding a case that is constrained in the reason model but not in the result model. This result is given by Proposition 8. However, we are unable to find an equivalent to Proposition 8 in the dimensional setting. Instead, we notice that the opposite to Proposition 9 holds true as well, that is, decisions in the dimensional result model are constrained by decisions in the dimensional reason model as well. This was first observed by Horty when he introduced the dimensional reason model [Hor17].

Proposition 10 (Reason model constrains result model: dimensional). Let Γ be a consistent case base in the dimensional reason model. If a dimensional fact situation X can only be decided in favour of side s for case base Γ , then X can only be decided in favour of s for case base Γ^* as well.

Proof. Let Γ be a consistent case base in the dimensional reason model, let Γ^* be its translation to the dimensional result model and X be a dimensional fact situation. We argue by contraposition. Assume that Γ^* does not require a decision in favour of s . Then there is no case $c \in \Gamma$ such that $\text{Outcome}(c) = s$ and $\text{Facts}(c) \leq^s X$. That is, for every case $c \in \Gamma$ where $\text{Outcome}(c) = s$ we have some dimension d such that $\text{Facts}(c)(d) \not\leq_d^s X(d)$. To show that Γ cannot require a decision in favour of s either, we construct a rule r for side \bar{s} such that $\Gamma \cup \{ \langle X, r, \bar{s} \rangle \}$ is consistent.

Case 1: There is no case $c \in \Gamma$ such that $\text{Outcome}(c) = s$. Then, decision cannot be required in favour of s and we are done.

Case 2: There is some case $c \in \Gamma$ such that $\text{Outcome}(c) = s$. For every such case c , we define d_c as the dimension—or, one of them—where $\text{Facts}(c)(d_c) \not\leq_{d_c}^s X(d_c)$, which we know to exist in every case. Let us now consider magnitude factor

$$M_{d_c, X(d_c)}^{\bar{s}},$$

which states that assignment on dimension d_c must be at least as strong for side \bar{s} as $X(d_c)$. We specify rule

$$r = \{ M_{d_c, X(d_c)}^{\bar{s}} \mid c \in \Gamma, \text{Outcome}(c) = s \} \rightarrow \bar{s}.$$

We can assert that $c^* = \langle X, r, \bar{s} \rangle$ is a case, since $X \models \text{Premise}(r)$ because for every $M_{d_c, X(d_c)}^{\bar{s}}$ we know that $X(d_c) \leq_{d_c}^{\bar{s}} X(d_c)$ trivially holds.

Now we need to establish that $\Gamma \cup \{c^*\}$ is consistent. Assume for contradiction that it is inconsistent. Since Γ is consistent, by Proposition 3 there is some case $c' \in \Gamma$ such that $c' = \langle Y, r', s \rangle$ where $X \models \text{Premise}(r')$ and $Y \models \text{Premise}(r)$. However, since

$c' \in \Gamma$ and $\text{Outcome}(c') = s$, we know that there is some $M_{d_{c'}, X(d_{c'})}^{\bar{s}} \in \text{Premise}(r')$. Since $Y \models \text{Premise}(r')$, we know that $X(d_{c'}) \leq_{d_{c'}}^{\bar{s}} Y(d_{c'})$, which by duality is equivalent to $Y(d_{c'}) \leq_{d_{c'}}^s X(d_{c'})$. This is equivalent to $\text{Facts}(c')(d_{c'}) \leq_{d_{c'}}^s X(d_{c'})$. However, we asserted earlier that this statement is false. Therefore, $\Gamma \cup \{c^*\}$ is consistent, which shows that Γ does not require a reason in favour of s either. \square

Proposition 9 states that constraint in the dimensional reason model is at least as strong as constraint in the dimensional result model. Proposition 10 states that constraint in the dimensional result model is at least as strong as constraint in the dimensional reason model. Thus, the two models have equivalent notions of constraint in the dimensional setting. This conclusion is problematic as it effectively renders all reasons obsolete.

2.2.4 The Revised Reason Model

If a decision in favour of a side is only ever required in the dimensional reason model, whenever the dimensional result model requires the same decision, specifying a reason becomes superfluous. This result gives rise to the idea that the notion of consistency is ill-defined and needs to be replaced by a stronger version. In his paper that introduces the dimensional reason model, Horty is indecisive at first, how a new notion of consistency could look like [Hor17]. In a later paper, he settles for a solution that fortifies the notion of consistency by adjusting the priority ordering [Hor21]. While this proposed change merely redefines the priority ordering, it directly affects the consistency as well as decision constraints and permissions.

Knowledge Representation Representation of cases and factual information stays the same as in the original definition of the dimensional reason model, which is given in Subsection 2.2.2.

Consistency We will now look at a revised notion of consistency proposed by Horty in [Hor21]. For this, we first introduce negated reasons.

Definition 29 (Negated reason: dimensional). Let Q be a reason favouring side s . Then,

$$\bar{Q} = \{M_{d,p}^{\bar{s}} \mid M_{d,p}^s \in Q\}$$

is the negated reason.

A negated reason uses the same thresholds but reverses the side it argues in favour of. A reason $Q = \{M_{d,p}^s\}$ states that in a fact situation $X = \{\langle d, q \rangle, \dots\}$ the condition $p \leq_d^s q$ needs to hold for $X \models Q$. However, in the negated reason $\bar{Q} = \{M_{d,p}^{\bar{s}}\}$ the condition $p \leq_d^s q$ is essentially replaced with $p \leq_d^{\bar{s}} q$ for $X \models \bar{Q}$ to hold. We will now use this definition to define a modification to the priority ordering.

Definition 30 (Priority ordering derived from case, revised: dimensional). Let $c = \langle X, r, s \rangle$ be a case in the dimensional reason model and W and Z be two reasons supporting opposite sides where W supports side \bar{s} and Z supports side s . The relation $<_c$ represents the priority ordering of reasons for case c . We say that $W <_c Z$ if and only if

1. $Z \Vdash \text{Premise}(r)$, and
2. $X \models W$ or $\overline{\text{Premise}(r)} \Vdash W$.

This new definition for the priority ordering derived from a case again propagates to the whole case base by Definition 16: Let Γ be a case base in the reason model and W and Z be two reasons supporting opposite sides where W supports side \bar{s} and Z supports side s . The relation $<_\Gamma$ represents the priority ordering of reasons for the case base Γ . We say that $W <_\Gamma Z$ if and only if $W <_c Z$ for some case $c \in \Gamma$.

Now, consistency in the case base, given by Definition 17, uses the new priority ordering: Let Γ be a case base in the reason model. We say that Γ is inconsistent if and only if there are two reasons W and Z supporting opposite sides such that $W <_\Gamma Z$ and $Z <_\Gamma W$.

Let us further explore this new notion of consistency by again considering a case $c = \langle X, r, s \rangle$ with reasons W and Z such that W supports side \bar{s} , Z supports side s , and $W <_c Z$ according to this new definition. Like in the original definition, if $Z \Vdash \text{Premise}(r)$ —i.e., Z is stronger than $\text{Premise}(r)$ —and $X \models W$ —i.e., X satisfies W —then reason Z must take priority over W , since it is at least as strong as $\text{Premise}(r)$ which was chosen over W in case c . But this new priority ordering does not necessarily require that $X \models W$. Instead, it is also sufficient that $\overline{\text{Premise}(r)} \Vdash W$ —i.e., the negated reason of $\text{Premise}(r)$ is stronger than W —for Z to be prioritised. The intuition here is that $\overline{\text{Premise}(r)}$ for side s clearly has a higher priority than $\overline{\text{Premise}(r)}$ for side \bar{s} , otherwise $\overline{\text{Premise}(r)}$ would have been chosen. Since $\overline{\text{Premise}(r)}$ is stronger for side \bar{s} than W —because of $\overline{\text{Premise}(r)} \Vdash W$ —, we know that $\overline{\text{Premise}(r)}$ takes priority over W . Since $Z \Vdash \text{Premise}(r)$ —that is, Z is stronger for side s than $\text{Premise}(r)$ —, Z also takes priority over W .

Using this new definition of consistency, we need to find a new way of efficiently checking the consistency of a case base. In the standard setting, we used Proposition 3, which states that a case base Γ is inconsistent if and only if we have to cases $\langle X, r, s \rangle, \langle Y, r', \bar{s} \rangle \in \Gamma$ such that $X \models \text{Premise}(r')$ and $Y \models \text{Premise}(r)$. However, this proposition does not apply when the revised priority ordering is used. Therefore, we need to assert that there is an efficient measure of consistency for the revised priority ordering as well.

Proposition 11 (Efficient reason model consistency check, revised: dimensional). Let Γ be a dimensional case base. Then, Γ is inconsistent if and only if there are two cases $c, c' \in \Gamma$ where

$$c = \langle X, r, s \rangle \text{ and } c' = \langle Y, r', \bar{s} \rangle$$

such that

$$\text{Premise}(r') <_c \text{Premise}(r) \text{ and } \text{Premise}(r) <_{c'} \text{Premise}(r').$$

Because $\text{Premise}(r) \Vdash \text{Premise}(r)$ and $\text{Premise}(r') \Vdash \text{Premise}(r')$ always hold trivially, the statements are equivalent to

1. $X \models \text{Premise}(r')$ or $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$, and
2. $Y \models \text{Premise}(r)$ or $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$

respectively.

Since inconsistency per Proposition 3 (which states that a case base Γ is inconsistent if and only if we have to cases $\langle X, r, s \rangle, \langle Y, r', \bar{s} \rangle \in \Gamma$ such that $X \models \text{Premise}(r')$ and $Y \models \text{Premise}(r)$) implies inconsistency per Proposition 11, which we just introduced, through contraposition we know that consistency per Proposition 11 implies consistency per Proposition 3. So, if a case base is consistent for the revised priority ordering, it is consistent for the original priority ordering as well. This also means that the revised notion of consistency is at least as restrictive as the initial definition.

Constraint The notions of constraint and permission are directly affected by this new priority ordering. Like before, we speak of rule selection, given by Definition 18: Let Γ be a case base and X be a fact situation. Rule $r = Q \rightarrow s$ with reason Q such that $X \models Q$ can be selected to decide in favour of side s if and only if

$$\Gamma \cup \{\langle X, r, s \rangle\}$$

is a consistent case base.

Depending on the selectable rules, we speak of decision constraint or permission as given by Definition 19: Let Γ be a consistent case base and X be a fact situation. A decision is constrained to side s if and only if all rules r that can be selected support side s —i.e., $\text{Outcome}(r) = s$. Equivalently, a decision in favour of side s is permitted if and only if there is some rule r that can be selected to support side s .

We will now make further assertions to show which properties shown for the standard setting continue to hold for this new notion of constraint. In Proposition 9 we proved that if a decision is constrained in the result model, it is constrained in the reason model as well. We can easily see that Proposition 9 continues to hold even for the revised notion of consistency, since our revised priority ordering is at least as restrictive as our initial priority ordering.

However, we need to find an alternative to Proposition 4 from the standard setting, which provides an efficient measure for decision permissions and constraints: Let Γ be a consistent standard case base and X be a standard fact situation. A decision is constrained to side s if and only if

$$\Gamma \cup \{\langle X, X^{\bar{s}} \rightarrow \bar{s}, \bar{s} \rangle\}$$

is inconsistent. Equivalently, a decision in favour of s is permitted if and only if

$$\Gamma \cup \{\langle X, X^s \rightarrow s, s \rangle\}$$

is consistent.

We propose the following check for decision constraints and permissions.

Proposition 12 (Efficient reason model decision constraint and permission: dimensional).

Let Γ be a consistent dimensional case base and X be a dimensional fact situation. A decision is constrained to side s if and only if

$$\Gamma \cup \{\langle X, U_X^{\bar{s}} \rightarrow \bar{s}, \bar{s} \rangle\}$$

is inconsistent where

$$U_X^{\bar{s}} = \{M_{d,p}^{\bar{s}} \mid \langle d, p \rangle \in X\}.$$

Since Γ is consistent, a decision in favour of side s is constrained if and only if there is a case $\langle Y, r', s \rangle \in \Gamma$ such that

1. $X \models \text{Premise}(r')$, and
2. $X \leq^{\bar{s}} Y$ or $\overline{\text{Premise}(r')} \Vdash U_X^{\bar{s}}$.

Equivalently, a decision in favour of side s is permitted if and only if

$$\Gamma \cup \{\langle X, U_X^s \rightarrow s, s \rangle\}$$

is consistent where

$$U_X^s = \{M_{d,p}^s \mid \langle d, p \rangle \in X\}.$$

Since Γ is consistent, a decision in favour of side s is permitted if and only if there is no case $\langle Y, r', \bar{s} \rangle \in \Gamma$ such that

1. $X \models \text{Premise}(r')$, and
2. $X \leq^s Y$ or $\overline{\text{Premise}(r')} \Vdash U_X^s$.

Proof. We will only perform the proof for efficient decision permissions, as the dual proof for efficient decision constraints follows a mostly identical argumentation.

Let Γ be a consistent dimensional case base. We will first argue for the improved consistency check. Then will show the equivalence of the consistency permission.

When we propose rule $r = U_X^s \rightarrow s$, the inconsistency condition with some case $\langle Y, r', \bar{s} \rangle \in \Gamma$ transforms into

1. $X \models \text{Premise}(r')$ or $U_X^{\bar{s}} \Vdash \text{Premise}(r')$, and

2. $Y \models U_X^s$ or $\overline{\text{Premise}(r')} \Vdash U_X^s$.

$X \models \text{Premise}(r')$ requires that every value assignment in X is at least as strong as every threshold value in every magnitude factor $\text{Premise}(r')$ for side \bar{s} , and $U_X^{\bar{s}} \Vdash \text{Premise}(r')$ requires that every threshold value in every magnitude factor in $U_X^{\bar{s}}$ is at least as strong as every threshold value in matching every magnitude factor in $\text{Premise}(r')$ for side \bar{s} . But since X and $U_X^{\bar{s}}$ contain exactly the same values per definition, we know that

$$X \models \text{Premise}(r') \text{ if and only if } U_X^{\bar{s}} \Vdash \text{Premise}(r').$$

Furthermore, for the same reason we can rewrite

$$Y \models U_X^s \text{ as } X \leq^s Y.$$

Thus, to check whether X would make case base Γ inconsistent, we merely need to check if

1. $X \models \text{Premise}(r')$, and
2. $X \leq^s Y$ or $\overline{\text{Premise}(r')} \Vdash U_X^s$.

Thus, a decision for side s is consistent with a case $\langle Y, r', \bar{s} \rangle$ if and only if

1. $X \not\models \text{Premise}(r')$, or
2. $X \not\leq^s Y$ and $\overline{\text{Premise}(r')} \not\Vdash U_X^s$.

“ \implies ” Assume that there is a rule r that can be selected to make a decision in favour of s on fact situation X . That is, $\Gamma \cup \{\langle X, r, s \rangle\}$ is consistent. We know that for every case $\langle Y, r', \bar{s} \rangle \in \Gamma$ we have either

1. $X \not\models \text{Premise}(r')$ and $\overline{\text{Premise}(r)} \not\Vdash \text{Premise}(r')$, or
2. $Y \not\models \text{Premise}(r)$ and $\overline{\text{Premise}(r')} \not\Vdash \text{Premise}(r)$.

We make a case distinction for these two conditions:

Case 1: $X \not\models \text{Premise}(r')$ and $\overline{\text{Premise}(r)} \not\Vdash \text{Premise}(r')$. From $X \not\models \text{Premise}(r')$ we know that $U_X^s \rightarrow s$ can be selected as a rule.

Case 2: $Y \not\models \text{Premise}(r)$ and $\overline{\text{Premise}(r')} \not\Vdash \text{Premise}(r)$. Because $Y \not\models \text{Premise}(r)$, and since we know that $X \models \text{Premise}(r)$, we can derive that $X \not\leq^s Y$. Because $\overline{\text{Premise}(r')} \not\Vdash \text{Premise}(r)$, and since we know that $U_X^s \Vdash \text{Premise}(r)$, we can derive that $\overline{\text{Premise}(r')} \not\Vdash U_X^s$. From $X \not\leq^s Y$ and $\overline{\text{Premise}(r')} \not\Vdash U_X^s$ we now that $U_X^s \rightarrow s$ can be selected as a rule.

Thus, $U_X^s \rightarrow s$ can be selected as a rule if there is a selectable rule r supporting a decision in favour of s .

“ \impliedby ” Assume that there is no rule that can be selected to decide fact situation X in favour of s . Then, $U_X^s \rightarrow s$ is not a selectable rule either. \square

This shows that we can still efficiently check whether a decision is permitted by only selecting a single rule for a fact situation. Further, given a consistent case base, we know that it is always possible to make a decision in favour of at least one of the two sides in the revised dimensional reason model. This is the dimensional equivalent to Proposition 5 in the standard setting.

Proposition 13 (Reason model decision possible, revised: dimensional). Let Γ be a consistent dimensional case base and X be a dimensional fact situation. By the revised dimensional reason model, a decision in favour of at least one of the two sides π and δ is always possible.

2.2.5 Result Model vs Revised Reason Model: No Collapse

We have not yet shown that this new notion of consistency for the dimensional reason model given by the revised priority ordering in Definition 30 yields a stronger notion of constraint than the dimensional result model. To reiterate, Definition 30 defines the revised priority ordering as follows: Let $c = \langle X, r, s \rangle$ be a case in the dimensional reason model and W and Z be two reasons supporting opposite sides where W supports side \bar{s} and Z supports side s . The relation $<_c$ represents the priority ordering of reasons for case c . We say that $W <_c Z$ if and only if

1. $Z \Vdash \text{Premise}(r)$, and
2. $X \models W$ or $\overline{\text{Premise}(r)} \Vdash W$.

We will again show the stronger notion of constraint through an example where the revised reason model requires a decision whereas the result model does not.

Proposition 14 (Reason model does not constrain result model, revised: dimensional). Let Γ be a consistent case base in the dimensional reason model. If a dimensional fact situation X can only be decided in favour of side s for case base Γ by the revised dimensional reason model, then X is not necessarily required to be decided in favour of s for case base Γ^* as well by the dimensional result model.

Proof. We show this by providing an example where the reason model requires a decision in favour of π , but the result model does not. Let $\Gamma_{15} = \{c_{46}, c_{47}\}$ be a consistent case base with

$$\begin{aligned} c_{46} &= \langle \{ \langle d_{\text{protest}}, \text{kicked} \rangle \}, r_{46}, \pi \rangle && \text{where } r_{46} = \{ M_{d_{\text{protest}}, \text{screamed}}^\pi \} \rightarrow \pi, \text{ and} \\ c_{47} &= \langle \{ \langle d_{\text{protest}}, \text{none} \rangle \}, r_{47}, \delta \rangle && \text{where } r_{47} = \{ M_{d_{\text{protest}}, \text{cried}}^\delta \} \rightarrow \delta. \end{aligned}$$

We have defined translation Γ_{15}^* of Γ_{15} such that $\Gamma_{15}^* = \{ \langle c_{46}, \pi \rangle, \langle c_{47}, \delta \rangle \}$. Now let us look at fact situation

$$X_{48} = \{ \langle d_{\text{protest}}, \text{complained} \rangle \}.$$

A decision in favour of π or δ is not required according to Γ_{15}^* , since

$$\text{Facts}(c_{46}) \not\leq^{\pi} X_{48} \text{ and } \text{Facts}(c_{47}) \not\leq^{\delta} X_{48}.$$

As a reminder, we know that a decision in favour of s is required for Γ_{15} if and only if a decision in favour of \bar{s} is impossible. That is, a decision in favour of δ is required if and only if

$$\Gamma_{15} \cup \{\langle X_{48}, U_{X_{48}}^{\pi} \rightarrow \pi, \pi \rangle\}$$

is inconsistent, where $U_{X_{48}}^{\pi}$ is the strongest reason for side π . Further, remember that our new notion of inconsistency states that a case base Γ is inconsistent if and only if there are two cases $\langle X, r, s \rangle, \langle Y, r', \bar{s} \rangle \in \Gamma$ where

1. $X \models \text{Premise}(r')$ or $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$, and
2. $Y \models \text{Premise}(r)$ or $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$.

Regarding X_{48} , is a decision in favour of π permitted according to Γ_{15} ? No, because for opposing case c_{47} we have

1. $X_{48} \models \text{Premise}(r_{47})$ which is $\{\langle d_{\text{protest}}, \text{complained} \rangle\} \models \{M_{d_{\text{protest}}, \text{cried}}^{\delta}\}$, and
2. $\overline{\text{Premise}(r_{47})} \Vdash U_{X_{48}}^{\pi}$ which is $\{M_{d_{\text{protest}}, \text{cried}}^{\pi}\} \Vdash \{M_{d_{\text{protest}}, \text{complained}}^{\pi}\}$.

Therefore, a decision in favour of π is not permitted which is why it is required in favour of δ . \square

This shows that the dimensional reason model and the dimensional result model do not collapse using this definition. But, of course, it is still possible that the decision is not restricted in both models.

Example 33. Let $\Gamma_{16} = \{c_{48}, c_{49}\}$ be a consistent case base with

$$\begin{aligned} c_{48} &= \{\langle \langle d_{\text{protest}}, \text{kicked} \rangle, r_{48}, \pi \rangle\} & \text{where } r_{48} &= \{M_{d_{\text{protest}}, \text{screamed}}^{\pi}\} \rightarrow \pi, \text{ and} \\ c_{49} &= \{\langle \langle d_{\text{protest}}, \text{none} \rangle, r_{49}, \delta \rangle\} & \text{where } r_{49} &= \{M_{d_{\text{protest}}, \text{complained}}^{\delta}\} \rightarrow \delta. \end{aligned}$$

Now let us look at fact situation

$$X_{50} = \{\langle d_{\text{protest}}, \text{cried} \rangle\}.$$

A decision in favour of π or δ is not required according to Γ_{16}^* , since

$$\text{Facts}(c_{48}) \not\leq^{\pi} X_{50} \text{ and } \text{Facts}(c_{49}) \not\leq^{\delta} X_{50}.$$

When deciding in favour of π , for opposing case c_{49} we have

1. $X_{50} \not\models \text{Premise}(r_{49})$ and $\overline{U_{X_{50}}^\pi} \not\models \text{Premise}(r_{49})$.

That is,

1. $\{\langle d_{\text{protest}}, \text{cried} \rangle\} \not\models \{M_{d_{\text{protest}}, \text{complained}}^\delta\}$ and $\{M_{d_{\text{protest}}, \text{cried}}^\delta\} \not\models \{M_{d_{\text{protest}}, \text{complained}}^\delta\}$.

Therefore, a decision in favour of π is possible. When deciding in favour of δ , for opposing case c_{48} we have

1. $X_{50} \not\models \text{Premise}(r_{48})$ and $\overline{U_{X_{50}}^\delta} \not\models \text{Premise}(r_{48})$.

That is,

1. $\{\langle d_{\text{protest}}, \text{cried} \rangle\} \not\models \{M_{d_{\text{protest}}, \text{screamed}}^\pi\}$ and $\{M_{d_{\text{protest}}, \text{cried}}^\pi\} \not\models \{M_{d_{\text{protest}}, \text{screamed}}^\pi\}$.

Therefore, a decision in favour of δ is possible as well. Thus, we have asserted that neither for Γ_{16}^* nor for Γ_{16} a decision in favour of a side is required.

Let us conclude with a slightly more complicated example where a decision is required in the dimensional reason model but not in the dimensional result model.

Example 34. Let $\Gamma_{17} = \{c_{50}, c_{51}\}$ be a consistent case base with

$$c_{50} = \{\langle d_{\text{screen}}, 120 \rangle, \langle d_{\text{activity}}, 30 \rangle\}, r_{50}, \pi \quad \text{where } r_{50} = \{M_{d_{\text{screen}}, 100}^\pi, M_{d_{\text{activity}}, 100}^\pi\} \rightarrow \pi, \text{ and}$$

$$c_{51} = \{\langle d_{\text{screen}}, 30 \rangle, \langle d_{\text{activity}}, 90 \rangle\}, r_{51}, \delta \quad \text{where } r_{51} = \{M_{d_{\text{screen}}, 40}^\delta, M_{d_{\text{activity}}, 70}^\delta\} \rightarrow \delta.$$

Now let us look at fact situation

$$X_{52} = \{\langle d_{\text{screen}}, 20 \rangle, \langle d_{\text{activity}}, 80 \rangle\}.$$

A decision in favour of π or δ is not required according to Γ_{17}^* , since

$$\text{Facts}(c_{50}) \not\leq^\pi X_{52} \text{ and } \text{Facts}(c_{51}) \not\leq^\delta X_{52}.$$

Regarding X_{52} , is a decision in favour of π possible according to Γ_{17} ? No, because for opposing case c_{51} we have

1. $X_{52} \models \text{Premise}(r_{51})$, and
2. $\overline{U_{X_{52}}^\pi} \not\models \text{Premise}(r_{51})$

That is,

1. $\{\langle d_{\text{screen}}, 20 \rangle, \langle d_{\text{activity}}, 80 \rangle\} \models \{M_{d_{\text{screen}}, 40}^\delta, M_{d_{\text{activity}}, 70}^\delta\}$, and

$$2. \{M_{d_{\text{screen}},40}^\pi, M_{d_{\text{activity}},70}^\pi\} \Vdash \{M_{d_{\text{screen}},20}^\pi, M_{d_{\text{activity}},80}^\pi\}.$$

Therefore, a decision in favour of π is not possible and required in favour of δ . When deciding in favour of δ , for opposing case c_{50} we have

$$1. X_{52} \not\models \text{Premise}(r_{50}) \text{ and } \overline{U_{X_{52}}^\delta} \not\models \text{Premise}(r_{50}).$$

That is,

$$1. \{\langle d_{\text{screen}}, 20 \rangle, \langle d_{\text{activity}}, 80 \rangle\} \not\models \{M_{d_{\text{screen}},100}^\pi, M_{d_{\text{activity}},100}^\pi\} \text{ and} \\ \{M_{d_{\text{screen}},20}^\pi, M_{d_{\text{activity}},80}^\pi\} \not\models \{M_{d_{\text{screen}},100}^\pi, M_{d_{\text{activity}},100}^\pi\}.$$

Rule $U_{X_{52}}^\delta \rightarrow \delta$ exemplifies how a decision in favour of δ could be made.

Summary and Outlook

Translating the reason model from the standard to the dimensional setting introduced the problem of the model collapsing with the result model. To recover from this, a stronger notion of consistency was introduced by revising the priority ordering. The downside of this new definition is its added complexity.

In the next chapter, we will describe a property of the standard reason model that the dimensional reason model does not provide, but we consider important: In the standard setting, replacing a reason in a consistent case base with a stronger one cannot cause the case base to become inconsistent. However, this is not the case in the dimensional setting where replacing a reason with a stronger one may cause an inconsistency in certain case bases. We believe that this should not occur, since a stronger reason should apply to fewer cases instead of more, and thus constrain fewer cases as well. A lack of this property in the dimensional setting motivates the necessity to introduce another model of precedential constraint.

Reducing the Models

In Section 3.1 we start by describing a property of the standard reason model that is lacking in the dimensional reason model but we consider very important for a model of precedential constraint. This property expresses that it should always be possible to strengthen a reason in a case base without causing an inconsistency to occur. In Section 3.2 we motivate the reduction model by finding a notion of constraint in the standard setting that is equivalent to the reason model. The main result lies in Section 3.3 where we generalise our findings to the dimensional reduction model which produces a notion of constraint that is even stronger than the reason model while also supporting the feature described in Section 3.1.

3.1 Strengthening Reasons

In this section, we make and analyse the observation that in the standard reason model it is always possible to make a reason stronger. This means that for any reason in the case base, we can add more factors to it without causing an inconsistency. Let us first motivate this idea with an example.

Example 35. Let $\Gamma_{18} = \{c_{52}, c_{53}\}$ be a case base containing two opposing cases

$$\begin{aligned} c_{52} &= \langle \{f_{\text{protest}}^{\pi}, f_{\text{activity}}^{\delta}\}, r_{52}, \pi \rangle & \text{where } r_{52} &= \{f_{\text{protest}}^{\pi}\} \rightarrow \pi, \text{ and} \\ c_{53} &= \langle \{f_{\text{protest}}^{\pi}, f_{\text{activity}}^{\delta}, f_{\text{clean}}^{\delta}, f_{\text{read}}^{\delta}\}, r_{53}, \delta \rangle & \text{where } r_{53} &= \{f_{\text{activity}}^{\delta}, f_{\text{clean}}^{\delta}\} \rightarrow \delta. \end{aligned}$$

We know that Γ_{18} is consistent, because

$$\text{Facts}(c_{52}) \not\models \text{Premise}(r_{53}).$$

Imagine we now want to strengthen the reason in rule r_{53} such that

$$\text{Premise}(r_{53}) := \{f_{\text{activity}}^{\delta}, f_{\text{clean}}^{\delta}, f_{\text{read}}^{\delta}\},$$

that is, we add factor f_{read}^δ to the reason. Thus, the reason has become stronger because it has stronger conditions to be satisfied. The case base stays consistent because adding factors to $\text{Premise}(r_{53})$ cannot change the assertion that $\text{Facts}(c_{52}) \not\models \text{Premise}(r_{53})$.

Let us now further discuss the idea that we can always strengthen a present reason without causing an inconsistency. By Proposition 3, an inconsistency is introduced to a case base Γ if and only if there are two cases $\langle X, r, s \rangle, \langle Y, r', \bar{s} \rangle$ such that

$$X \models \text{Premise}(r') \text{ and } Y \models \text{Premise}(r).$$

Let us consider two reasons Q and Q' for the same side s . If $Q' \Vdash Q$ this means that Q' is stronger for side s than Q . That is, fewer fact situations will be satisfied by Q' as it requires stronger factual strength for side s . So if a reason Q in some case in the case base is replaced by Q' , our intuition is that the case base should not become inconsistent, because the conditions $X \models \text{Premise}(r')$ and $Y \models \text{Premise}(r)$ will apply to at most as many or fewer fact situations. Let us now assert this claim.

Theorem 1 (Consistency of strengthening a reason: standard). Let

$$\Gamma \cup \{\langle X, W \rightarrow s, s \rangle\}$$

be a consistent standard case base and Z be a standard reason for side s such that

$$X \models Z \text{ and } Z \Vdash W.$$

Then, $\Gamma \cup \{\langle X, Z \rightarrow s, s \rangle\}$ is also consistent.

Proof. Let $\Gamma \cup \{c_{\text{weak}}\}$ be a consistent standard case base where $c_{\text{weak}} = \langle V, l, s \rangle$ and u be a rule such that $X \models \text{Premise}(u)$ and $\text{Premise}(u) \Vdash \text{Premise}(l)$. We now want to verify that $\Gamma \cup \{c_{\text{strong}}\}$ is also consistent where $c_{\text{strong}} = \langle V, u, s \rangle$. Per Proposition 3 we know that case base $\Gamma \cup \{c_{\text{strong}}\}$ is consistent if and only if for all cases $c, c' \in \Gamma \cup \{c_{\text{strong}}\}$ such that $c = \langle X, r, s \rangle$ and $c' = \langle Y, r', \bar{s} \rangle$ either $X \not\models \text{Premise}(r')$ or $Y \not\models \text{Premise}(r)$. Let $c, c' \in \Gamma \cup \{c_{\text{strong}}\}$ where $c = \langle X, r, s \rangle$ and $c' = \langle Y, r', \bar{s} \rangle$. We distinguish two cases.

Case 1: $c, c' \in \Gamma$. Since $\Gamma \cup \{c_{\text{weak}}\}$ is consistent, Γ is consistent as well. Therefore, $X \not\models \text{Premise}(r')$ or $Y \not\models \text{Premise}(r)$.

Case 2: $c = c_{\text{strong}}$ or $c' = c_{\text{strong}}$. W.l.o.g. we say that $c = c_{\text{strong}}$ and $c' \in \Gamma$. Since $\Gamma \cup \{c_{\text{weak}}\}$ is consistent, and since $V = X$ we know that $X \not\models \text{Premise}(r')$ or $Y \not\models \text{Premise}(l)$.

Case 2.1: $X \not\models \text{Premise}(r')$.

Case 2.2.: $Y \not\models \text{Premise}(l)$. Since $\text{Premise}(u) \Vdash \text{Premise}(l)$, we know that $Y \not\models \text{Premise}(u)$ ¹. And because $u = r$, we know that $Y \not\models \text{Premise}(r)$.

We find that $X \not\models \text{Premise}(r')$ or $Y \not\models \text{Premise}(r)$ for all $c, c' \in \Gamma \cup \{c_{\text{strong}}\}$. Thus, $\Gamma \cup \{c_{\text{strong}}\}$ is consistent. \square

¹We argue that if $X \not\models Z$ and $W \Vdash Z$, then $X \not\models W$, which follows directly from Definition 12.

We believe that this is an important property of the standard reason model. If a reason is made stronger, it should apply to at most as many or to fewer cases. Therefore, strengthening a reason should never cause an inconsistency. However, the dimensional reason model does not satisfy this property. We can prove this by providing a counterexample where strengthening a reason causes two cases to become inconsistent. The proof is given as follows.

Theorem 2 (Potential inconsistency of strengthening a reason, revised: dimensional).
Let

$$\Gamma \cup \{\langle X, W \rightarrow s, s \rangle\}$$

be a consistent case base and Z be a stronger reason than W for side s , i.e.,

$$X \models Z \text{ and } Z \Vdash W.$$

Then, $\Gamma \cup \{\langle X, Z \rightarrow s, s \rangle\}$ is not necessarily consistent.

Proof. To show that strengthening a reason is not necessarily consistent, we construct an example. Let $\Gamma_{19} = \{c_{54}, c_{55}\}$ be a case base with opposing cases

$$\begin{aligned} c_{54} &= \langle \langle d_{\text{activity}}, 90 \rangle, \langle d_{\text{read}}, 30 \rangle \rangle, r_{54}, \pi \rangle & \text{ where } r_{54} &= \{M_{d_{\text{activity}},100}^{\pi}, M_{d_{\text{read}},40}^{\pi}\} \rightarrow \pi, \text{ and} \\ c_{55} &= \langle \langle d_{\text{activity}}, 120 \rangle, \langle d_{\text{read}}, 20 \rangle \rangle, r_{55}, \delta \rangle & \text{ where } r_{55} &= \{M_{d_{\text{activity}},60}^{\delta}\} \rightarrow \delta. \end{aligned}$$

The case base is consistent since $\text{Facts}(c_{55}) \not\models \text{Premise}(r_{54})$, because of dimension d_{activity} , and $\text{Premise}(r_{55}) \not\models \text{Premise}(r_{54})$, because of dimension d_{read} . We can now replace rule r_{55} with the following rule

$$r_{56} = \{M_{d_{\text{activity}},60}^{\delta}, M_{d_{\text{read}},20}^{\delta}\} \rightarrow \delta.$$

This rule merely adds the magnitude factor $M_{d_{\text{read}},20}^{\delta}$ to $\text{Premise}(r_{55})$. Clearly, $\text{Facts}(c_{55}) \models \text{Premise}(r_{56})$ and $\text{Premise}(r_{56}) \Vdash \text{Premise}(r_{55})$, thus it is a strengthened rule. However, now we have $\text{Premise}(r_{54}) \Vdash \text{Premise}(r_{56})$ and $\text{Premise}(r_{56}) \Vdash \text{Premise}(r_{54})$ which makes the case base inconsistent. \square

This might be problematic for the dimensional reason model. A stronger argument should not stand in conflict with more cases than a weaker argument, because fewer cases meet the requirements of the stronger argument. In particular, the problem lies within how consistency is defined in the dimensional reason model: Two opposing cases $\langle X, r, s \rangle$ and $\langle Y, r', \bar{s} \rangle$ are consistent if and only if (1) $X \not\models \text{Premise}(r')$ and $\text{Premise}(r) \not\models \text{Premise}(r')$, or (2) $Y \not\models \text{Premise}(r)$ and $\text{Premise}(r') \not\models \text{Premise}(r)$. W.l.o.g. we can just consider (1). Assume that $X \not\models \text{Premise}(r')$ and $\text{Premise}(r) \not\models \text{Premise}(r')$ holds. As we have just seen in Theorem 2, merely the absence of a dimension d in $\text{Premise}(r)$ that is present in $\text{Premise}(r')$ is sufficient for $\text{Premise}(r) \not\models \text{Premise}(r')$ to hold. But when that dimension is added to $\text{Premise}(r)$, thus strengthening the reason, $\text{Premise}(r) \Vdash \text{Premise}(r')$ may

become satisfied, causing an inconsistency. We believe that the absence of a dimension is not a sufficient criterium for justifying the consistency of two cases.

In Section 3.3 we will design a new extension of the dimensional reason model—the reduction model—that supports strengthening reasons like it is possible in the standard reason model. But we will first motivate this new model using the standard setting in Section 3.2.

3.2 Motivation: The Standard Reduction Model

We introduce the standard reduction model. It presents a notion of constraint equivalent to the standard reason model. The key element of the model is that it performs a reduction of the cases from the reason model to the result model, and then uses the rules of the result model to reason about decision constraints. We will start by defining the model, and then assert that it presents an equivalent notion of constraint. The model acts as a motivation for the dimensional reduction model introduced in Section 3.3.

Model Reduction Let us first look at the result model of precedential constraint in the standard setting. An essential feature of the result model can be found in Definition 6 which enables to measure and compare the strength of fact situations: Let X and Y be fact situations. We say that Y is at least as strong for side s as X —written $X \leq^s Y$ —if and only if

$$X^s \subseteq Y^s \text{ and } Y^{\bar{s}} \subseteq X^{\bar{s}}.$$

The notion of constraint in the result model is given by Definition 8. This definition states that, given a case base Γ and a fact situation X , a decision is required for side s if and only if there is some case

$$\langle Y, s \rangle \in \Gamma \text{ such that } Y \leq^s X.$$

That is, if fact situation X is “at least as good” for a side as some fact situation Y from a case in the case base, then X has to be decided for the same side as Y . We can think of the decision constraint in the result model as a form of *monotonicity* requirement: If X is at least as strong for side s as Y , and Y was decided in favour of s , then X must also be decided in favour of s .

Let us now take a look at the reason model in the standard setting. For this, let us first revisit the notion of consistency by Proposition 3: A case base Γ is inconsistent if and only if there are two cases $\langle X, r, s \rangle, \langle Y, r', \bar{s} \rangle \in \Gamma$ such that

$$X \models \text{Premise}(r') \text{ and } Y \models \text{Premise}(r).$$

In the standard setting, this is equivalent to

$$\text{Premise}(r') \subseteq X^{\bar{s}} \text{ and } \text{Premise}(r) \subseteq Y^s.$$

We notice that this definition is similar to the notion of consistency for the result model by Definition 7: A case base Γ is inconsistent if and only if there are two cases $\langle X, s \rangle, \langle Y, \bar{s} \rangle \in \Gamma$ such that $Y \leq^{\bar{s}} X$ which is equivalent to

$$Y^{\bar{s}} \subseteq X^{\bar{s}} \text{ and } X^s \subseteq Y^s.$$

Let us think of the fact situations X and Y as $X = X^s \cup X^{\bar{s}}$ and $Y = Y^{\bar{s}} \cup Y^s$. If we now replace the factors supporting the winning side, i.e., s for X and \bar{s} for Y , with the given reason, we get

$$X' = \text{Premise}(r) \cup X^{\bar{s}} \text{ and } Y' = \text{Premise}(r') \cup Y^s.$$

If we were to model the inconsistency between these generated cases $\langle X', s \rangle$ and $\langle Y', \bar{s} \rangle$, the requirement becomes $Y' \leq^{\bar{s}} X'$ which is equivalent to $\text{Premise}(r') \subseteq X^{\bar{s}}$ and $\text{Premise}(r) \subseteq Y^s$. Thus, we can reduce the requirement for inconsistency in the reason model to the requirement for inconsistency in the result model by translating the cases from the reason model to the result model. Instead of defining a new model, we reconstruct the reason model using the definitions of the result model. We formalise this idea with the following definition.

Definition 31 (Reason model case to result model case reduction: standard). Let $c = \langle X, r, s \rangle$ be a case in the standard reason model. We define $c^\times = \langle X^r, s \rangle$ as a case in the standard result model where

$$X^r = \text{Premise}(r) \cup X^{\bar{s}}.$$

Definition 32 (Reason model case base to result model case base reduction). Let Γ be a case base in the reason model. We define the translated case base Γ^\times in the result model such that

$$\Gamma^\times = \{c^\times \mid c \in \Gamma\}.$$

Thus, given a case in the standard reason model $\langle X, r, s \rangle$, we turn it into a case in the standard result model by only keeping

1. factors that are used for the reason in the rule r , and
2. factors in X that favour the opposite side \bar{s} .

Example 36. Let us consider case

$$\begin{aligned} c_{57} &= \langle X_{57}, r_{57}, \delta \rangle \text{ where} \\ X_{57} &= \{f_{\text{protest}}^\pi, f_{\text{school}}^\pi, f_{\text{activity}}^\delta, f_{\text{coop}}^\delta\} \text{ and} \\ r_{57} &= \{f_{\text{coop}}^\delta\} \rightarrow \pi. \end{aligned}$$

The case states that the child protested going to bed (f_{protest}^π), had to attend school the next day (f_{school}^π), was involved in physical activity ($f_{\text{activity}}^\delta$), and cooperated with its

sibling (f_{coop}^δ). The case was decided in favour of the child—i.e., the defendant δ —with the justification that the child cooperated with its sibling (f_{coop}^δ). We now want to translate case c_{57} in the reason model into a case $c_{57}^\times = \langle \{X_{57}^{r_{57}}\}, \delta \rangle$ in the reduction model. We define

$$X_{57}^{r_{57}} = \{f_{\text{protest}}^\pi, f_{\text{school}}^\pi, f_{\text{coop}}^\delta\}.$$

That is, all factors supporting the opposing side π are kept—i.e., X_{57}^π —whereas only the factors found in the justification are kept for side δ —i.e., $\text{Premise}(r_{57})$.

Consistency Consistency in the standard reduction model is achieved by using the consistency definition of the standard result model after performing a translation of the case base. It is given by Definition 7: Let Γ be a case base in the result model. We say that Γ is inconsistent if and only if there are two opposing cases $c, c' \in \Gamma$ —i.e., $c = \langle X, s \rangle$ and $c' = \langle Y, \bar{s} \rangle$ —such that $X \leq^s Y$. Equivalently, we say that Γ is consistent if and only if for all any two reasons W and Z supporting opposite sides we have $W \not\prec_\Gamma Z$ or $Z \not\prec_\Gamma W$.

Constraint We will now add definitions regarding rule selection, decision constraint as well as decision permission, and will then show that the notion of constraint for the reduction model is equivalent to that of the reason model in the standard setting.

Definition 33 (Reduction model rule selection). Let Γ be a case base in the reason model and X be a fact situation. We say that in the reduction model rule r in favour of s can be selected if and only if X^r does not require a decision in favour of \bar{s} for Γ^\times by the result model. This is equivalent to the fact that $\Gamma^\times \cup \{\langle X^r, s \rangle\}$ is consistent.

Definition 34 (Reduction model decision constraint). Let Γ be a case base in the reason model and X be a fact situation. We say that in the reduction model a decision in favour of s is required if and only if all rules r that can be selected are in favour of s .

As in the reason model, it is sufficient to use rule $X^s \rightarrow s$ to test whether X can be decided in favour of s . In the reduction, we retrieve

$$X^{X^s \rightarrow s} = X^s \cup X^{\bar{s}} = X.$$

We assert this claim in the following proposition.

Proposition 15 (Efficient reduction model decision permission: standard). Let Γ be a standard case base in the reason model and X be a standard fact situation. We say that a decision for X in favour of s is possible on Γ^\times if and only if there is a rule r that can be selected to decide X^r in favour of s on Γ^\times as well.

Proof. Let Γ be a standard case base in the reason model and X be a standard fact situation.

“ \implies ” Assume a decision in favour of s is possible for Γ^\times . This means that there is a selectable rule r that favours s , where we know for every $\langle Y, \bar{s} \rangle \in \Gamma^\times$ that $Y \not\leq^{\bar{s}} X^r$. Since $X^r \leq^s X$, we know that $Y \not\leq^{\bar{s}} X$. Thus, a decision in favour of s is possible for X on Γ^\times .

“ \impliedby ” Assume that a decision in favour of s is possible for X on Γ^\times . Therefore, $r = X^s \rightarrow s$ is a selectable rule, since $X^r = X$. \square

3.2.1 Relation to the Reason Model

We will now show that the standard reason model is equivalent to the standard reduction model, and that the standard reduction model is merely an alternative representation of the standard reason model. To illustrate that the models are equivalent, we show that a decision constraint always applies to case bases Γ and Γ^\times alike.

Proposition 16 (Reason model and reduction model equivalence: standard). Let Γ be a standard case base and X be a standard fact situation. A decision for X in favour of side s is required on Γ by the standard reason model if and only if a decision for X in favour of side s is required on Γ by the standard reduction model.

Proof. Let Γ be a standard case base and X be a standard fact situation.

“ \implies ” We argue by contraposition. Assume that Γ^\times does not require a decision in favour of either side. Proposition 4 states that a decision in favour of side s can be made if and only if $\Gamma \cup \{\langle X, X^s \rightarrow s, s \rangle\}$ is consistent. Thus, we can always select rule $r = X^s \rightarrow s$ for fact situation X , in case a decision in favour of s is possible. Since Γ^\times does not require a decision, for every case $\langle Y, \bar{s} \rangle \in \Gamma^\times$ we have $Y \not\leq^{\bar{s}} X$ which is equivalent to $Y^{\bar{s}} \not\subseteq X^{\bar{s}}$ or $X^s \not\subseteq Y^s$. Since $Y = \text{Premise}(r') \cup Y'^s$ for some $\langle Y', r', \bar{s} \rangle \in \Gamma$, we know that $\text{Premise}(r') \not\subseteq X^{\bar{s}}$ or $X^s \not\subseteq Y'^s$.

Case 1: $\text{Premise}(r') \not\subseteq X^{\bar{s}}$. Therefore, we have $\text{Premise}(r') \not\subseteq X$ which is equivalent to $X \not\models \text{Premise}(r')$.

Case 2: $X^s \not\subseteq Y'^s$. Since $X^s = \text{Premise}(r)$, we can assert $\text{Premise}(r) \not\subseteq Y'$ which is equivalent to $Y' \not\models \text{Premise}(r)$.

Proposition 3 states that a case base Γ is inconsistent if and only if there are two cases $\langle X, r, s \rangle, \langle Y', r', \bar{s} \rangle \in \Gamma$ such that $X \models \text{Premise}(r')$ and $Y' \models \text{Premise}(r)$. Since for any opposing case $\langle Y', r', \bar{s} \rangle \in \Gamma$ we have $X \not\models \text{Premise}(r')$ or $Y' \not\models \text{Premise}(r)$, we know that case base $\Gamma \cup \{\langle X, r, s \rangle\}$ is consistent. Since it allows a decision for any side s , it does not require a decision in favour of either side.

“ \impliedby ” Assume that X has to be decided in favour of s for Γ^\times . That means that there is some case $\langle Y, s \rangle \in \Gamma^\times$ such that $Y \leq^s X$. Assume for contradiction that Γ does not require a decision in favour of s . This means that there is some rule r supporting side \bar{s} such that $\Gamma \cup \{\langle X, r, \bar{s} \rangle\}$ is consistent. Since $\langle Y, s \rangle \in \Gamma^\times$, there has to be some case $\langle Y', r', s \rangle \in \Gamma$ such that $Y = \text{Premise}(r') \cup Y'^s$. Since $Y \leq^s X$ and $\text{Premise}(r') \subseteq Y$,

we know that $\text{Premise}(r') \subseteq X$, or equivalently $X \models \text{Premise}(r')$. Conversely, since $Y \leq^s X$ is equivalent to $X \leq^{\bar{s}} Y$, and $\text{Premise}(r) \subseteq X$, we know that $\text{Premise}(r) \subseteq Y$. Since $\text{Premise}(r)$ only contains factors favouring \bar{s} —i.e., $\text{Premise}(r) \subseteq X^{\bar{s}}$ —we know that $\text{Premise}(r) \subseteq Y^{\bar{s}}$, and because $Y^{\bar{s}} = Y'^{\bar{s}}$, we can assert that $\text{Premise}(r) \subseteq Y'^{\bar{s}}$. Thus $\text{Premise}(r) \subseteq Y'$, or equivalently $Y' \models \text{Premise}(r)$. Because $X \models \text{Premise}(r')$ and $Y' \models \text{Premise}(r)$, by Proposition 3 we know that $\Gamma \cup \{\langle X, r, \bar{s} \rangle\}$ is inconsistent which contradicts our assumption. Therefore, a decision in favour of s is required for Γ as well. \square

This proof shows that we can reduce the decision constraint in the reason model to the decision constraint in the result model. This reduction simplifies the model description as we now only rely on the straightforward definitions of the result model without taking away from the expressiveness of the reason model. We will build the dimensional reduction model on top of this idea.

3.3 The Dimensional Reduction Model

We introduce the dimensional reduction model. Where the standard reduction model simply represents an alternative representation of the standard reason model, the dimensional reduction model represents a new model that has a stronger notion of constraint than the dimensional reason model. We argue in favour of this new notion of constraint on the basis that it enables the strengthening of reasons, an important property that, as shown in Section 3.1, the dimensional reason model does not support.

Model Reduction We would like to be able to define a reduction for the dimensional setting in a similar manner to the standard reduction model. In particular, we would like to replace complex notion of consistency for the dimensional reason model introduced in the previous chapter. This notion of consistency states that, by Proposition 11, a case base Γ is inconsistent if and only if

1. $X \models \text{Premise}(r')$ or $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$, and
2. $Y \models \text{Premise}(r)$ or $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$

But, instead of trying to reformulate Proposition 11 such that it can be described by the dimensional result model, we try to generalise the model reduction, given by Definition 31, to the dimensional setting. Recall that in the standard setting, for case $\langle X, r, s \rangle \in \Gamma$, the reduction is defined as

$$X^r = \text{Premise}(r) \cup X^{\bar{s}}.$$

We cannot directly apply this definition in the dimensional setting, since for dimensions we do not have a notion of a factor favouring a side s , as each dimension admits a range of values. What we can do instead is to use every present magnitude factor

$M_{d,p}^s \in \text{Premise}(r)$ as a value assignment $\langle d, p \rangle$, which is similar to $\text{Premise}(r)$ in the standard setting, and all value assignments that do not have a magnitude factor we leave unchanged, which is similar to X^s . This idea is similar to Definition 31, since the fact situation is weakened for the winning side s such that it still meets the requirements to satisfy $\text{Premise}(r)$, but it cannot be weakened any further for side s such that $\text{Premise}(r)$ continues to hold. Let us now formalise this idea.

Definition 35 (Reason model case to result model case reduction: dimensional). Let $c = \langle X, r, s \rangle$ be a case in the dimensional reason model. We define $c^\times = \langle X^r, s \rangle$ as a case in the dimensional result model where

$$X^r = \{\langle d, p \rangle \mid M_{d,p}^s \in \text{Premise}(r)\} \cup \{\langle d, q \rangle \in X \mid \forall M_{d',p}^s \in \text{Premise}(r) : d' \neq d\}.$$

Thus, given a case $\langle X, r, s \rangle$ in the dimensional reason model, to turn it into a case in the dimensional result model $\langle X^r, s \rangle$, we assign X^r such that

1. we use the values of the magnitude factors for dimensions found in the reason, and
2. we keep the values of the dimensions not found in the reason.

It is exactly the reduction we just proposed in Definition 35 that changes the notion of consistency for the dimensional reduction model. Where in the standard setting we found a reduction with a notion of consistency that is equivalent to the reason model, for the dimensional setting the reduction generates a new notion of consistency that is different from the reason model, and enables the strengthening of reasons. Thus, the dimensional reduction model represents a new model of precedential constraint that is distinct from the dimensional reason model. We will soon analyse formally how the decision constraints between the models differ in particular and why. But let us first consider an example of how the reduction is performed.

Example 37. Let us consider the following case

$$\begin{aligned} c_{58} &= \langle X_{58}, r_{58}, \pi \rangle \text{ where} \\ X_{58} &= \{\langle d_{\text{protest}}, \text{screamed} \rangle, \langle d_{\text{activity}}, 45 \rangle\} \text{ and} \\ r_{58} &= \{M_{d_{\text{activity}}, 60}^\pi\} \rightarrow \pi. \end{aligned}$$

The case states that the child protested by screaming ($\langle d_{\text{protest}}, \text{screamed} \rangle$), and had 45 minutes of physical activity ($\langle d_{\text{activity}}, 45 \rangle$). The decision was in favour of the parents—i.e., the plaintiff π —with the justification that the child had at most 60 minutes of physical activity ($M_{d_{\text{activity}}, 60}^\pi$). We now want to translate c_{58} to a case $c_{58}^\times = \langle X_{58}^{r_{58}}, \pi \rangle$ in the reduction model. We define

$$X_{58}^{r_{58}} = \{\langle d_{\text{protest}}, \text{screamed} \rangle, \langle d_{\text{activity}}, 60 \rangle\}.$$

Since a magnitude factor for dimension d_{activity} is present, we replace $\langle d_{\text{activity}}, 45 \rangle$ with $\langle d_{\text{activity}}, 60 \rangle$. The value assignment $\langle d_{\text{protest}}, \text{screamed} \rangle$ is left untouched.

Consistency Like in the standard setting, consistency in the dimensional reduction model is achieved by using the consistency definition of the dimensional result model after performing a translation of the case base. It is also given by Definition 7: Let Γ be a case base in the result model. We say that Γ is inconsistent if and only if there are two opposing cases $c, c' \in \Gamma$ —i.e., $c = \langle X, s \rangle$ and $c' = \langle Y, \bar{s} \rangle$ —such that $X \leq^s Y$.

Constraint Given a fact situation X , we will now assert that we can efficiently check for a decision permission in favour of s by selecting rule $U_X^s \rightarrow s$. Since the translation yields

$$X^{U_X^s \rightarrow s} = X,$$

we will simply compare X against the translated case base Γ^\times .

Proposition 17 (Efficient reduction model decision permission: dimensional). Let Γ be a case base in the dimensional reason model and X be a dimensional fact situation. By the reduction model, we say that a decision for X in favour of s is possible on Γ^\times if and only if there is a rule r that can be selected to decide X^r in favour of s on Γ^\times as well.

Proof. Let Γ be a dimensional case base in the reason model and X be a dimensional fact situation.

“ \implies ” Assume a decision for X^r in favour of s is possible for Γ^\times . This means that for selectable rule r favouring s we know for every $\langle Y, \bar{s} \rangle \in \Gamma^\times$ that $Y \not\leq^{\bar{s}} X^r$. Since $X^r \leq^s X$ which is equivalent to $X \leq^{\bar{s}} X^r$, we know that $Y \not\leq^{\bar{s}} X$. Thus, a decision in favour of s is possible for X on Γ^\times as well.

“ \impliedby ” Assume that a decision in favour of s is possible for X on Γ^\times . Therefore, $r = U_X^s \rightarrow s$ is a selectable rule, since $X^r = X$. \square

3.3.1 Benchmarks from the Literature

To make a strong argument in favour of the reduction model, we will now motivate it with examples from the literature that describe the desired behaviour of a model of precedential constraint. We will then show that the reduction model satisfies our required properties from the given examples, before showing formally that these properties hold in the following sections. We will look at three examples, which are central for justifying why models should constrain certain decisions:

1. “Model collapse” from [Hor17]: This example illustrates the observed collapse between the dimensional result model and the original dimensional reason model.
2. “Revised reason model” from [Hor21]: This example is the justification for the revised dimensional reason model.
3. “Defence rule” from [Thö24]: This example is the justification why the notion of constraint in the revised dimensional reason model is not strong enough.

We will now see that the dimensional reduction model addresses all of the concerns discussed in the examples.

Example 38 (Model collapse). When Horty first introduced the dimensional reason model, he observed a collapse of the model with the dimensional result model. He illustrated this observation through an example where both the result model and the reason model fail to constrain the decision for a certain fact situation [Hor17]. The intention behind this example is to show that the dimensional reduction model does not collapse with the result model, since it constrains the decision for the fact situation.

The example is from the legal setting where a court evaluates a proposed change of fiscal domicile. A win for the defendant means that the change is accepted whereas a win for the plaintiff means that the change is rejected. For this, we define two dimensions d_{abroad} and $d_{\text{inc.abroad}}$ where the former represents the months spent abroad, and the latter represents the percentage of income that the individual earned abroad. A higher value for both of these strengthens the case for the defendant.

In our example, we have a case base $\Gamma_{20} = \{c_{59}\}$ with a single case

$$\begin{aligned} c_{59} &= \langle X_{59}, r_{59}, \delta \rangle \text{ where} \\ X_{59} &= \{ \langle d_{\text{abroad}}, 30 \rangle, \langle d_{\text{inc.abroad}}, 60\% \rangle \} \text{ and} \\ r_{59} &= \{ M_{d_{\text{abroad}}, 12}^\delta \} \rightarrow \delta. \end{aligned}$$

The case describes a scenario where an individual had lived abroad for 30 months ($\langle d_{\text{abroad}}, 30 \rangle$), and earned 60% of their income abroad ($\langle d_{\text{inc.abroad}}, 60\% \rangle$). The court decided to change fiscal domicile (a decision in favour of the defendant) on the basis that the individual had spent at least 12 months abroad ($M_{d_{\text{abroad}}, 12}^\delta$). The court is now presented with a new fact situation

$$X_{60} = \{ \langle d_{\text{abroad}}, 29 \rangle, \langle d_{\text{inc.abroad}}, 75\% \rangle \}$$

where an individual lived abroad for 29 months ($\langle d_{\text{abroad}}, 29 \rangle$) and earned 75% of their income abroad ($\langle d_{\text{inc.abroad}}, 75\% \rangle$). Compared to the fact situation X_{59} in c_{59} , this new fact situation is considerably stronger for the defendant along the dimension $d_{\text{inc.abroad}}$, and only slightly weaker along the dimension d_{abroad} . But since there is some dimension that is weaker for the defendant, i.e.,

$$X_{59} \not\leq^\delta X_{60},$$

the dimensional result model cannot constrain the decision in favour of the defendant. So the *original* dimensional reason model does not require a decision in favour of the defendant either, since we could propose a rule in favour the plaintiff, e.g.,

$$\{ \langle d_{\text{abroad}}, 29 \rangle \} \rightarrow \pi,$$

which requires that the individual spent at most 29 months abroad to deny a change of fiscal domicile. Like in the result model, any dimension that is weaker in the reason

model—however slight or minimal it may be—will prevent a decision constraint, causing the collapse of the dimensional result model and the dimensional reason model.

The reduction model eliminates this collapse. When we translate c_{59} to $c_{59}^{\times} = \langle X_{59}^{r_{59}}, \delta \rangle$ we get

$$X_{59}^{r_{59}} = \{ \langle d_{\text{abroad}}, 12 \rangle, \langle d_{\text{inc.abroad}}, 60\% \rangle \}.$$

Since X_{60} is stronger for the defendant than $X_{59}^{r_{59}}$, i.e.,

$$X_{59}^{r_{59}} \leq^{\delta} X_{60},$$

a decision in favour of δ is required by the reduction model.

The previous example illustrated that the original dimensional reason model collapses with the dimensional result model. Horty proposed a number of modifications to the dimensional reason model in [Hor17], however, all of these failed to reject a certain rule given in an example in [Hor21].

Example 39 (Revised reason model). This example is the motivation for Horty’s revised dimensional reason model, which was proposed in [Hor21] and described in Subsection 2.2.4. In this example, a certain rule that is allowed in the original dimensional reason model from [Hor17] is no longer allowed in its revision. The intention behind this example is to show that, like the revised reason model, the dimensional reduction model also rejects this rule.

For this example, we reuse the case base Γ_{20} from the previous example, containing the same case c_{59} . This time, the court is asked to decide on fact situation

$$X_{61} = \{ \langle d_{\text{abroad}}, 15 \rangle, \langle d_{\text{inc.abroad}}, 65\% \rangle \}.$$

In this scenario, the individual spent 15 months abroad ($\langle d_{\text{abroad}}, 15 \rangle$), and earned 65% of their income abroad ($\langle d_{\text{inc.abroad}}, 65 \rangle$). So, compared to fact situation $X_{59} = \{ \langle d_{\text{abroad}}, 30 \rangle, \langle d_{\text{inc.abroad}}, 60\% \rangle \}$ in the case base, the individual spent less time abroad, but earned a slightly bigger percentage of their income abroad. To recall, for case c_{59} the court ruled that for a change of fiscal domicile the individual must spend at least 12 months abroad, given by $r_{59} = \{ M_{d_{\text{abroad}}, 12}^{\delta} \} \rightarrow \delta$. Horty’s proposed modifications in [Hor17] admit a decision $c_{61} = \langle X_{61}, r_{61}, \pi \rangle$ for X_{61} in favour of the plaintiff, for example through rule

$$r_{61} = \{ M_{d_{\text{abroad}}, 24}^{\pi} \} \rightarrow \pi,$$

since X_{59} does not satisfy $\text{Premise}(r_{61})$. As discussed in Chapter 2, Horty argues in his revision of the reason model in [Hor21] that $\text{Premise}(r_{61})$ should be prioritised over $\text{Premise}(r_{59})$ for case c_{61} , since $\text{Premise}(r_{61})$ is a stronger reason for π than $\text{Premise}(r_{61})$ is for δ —otherwise we would have selected $\overline{\text{Premise}(r_{61})} \rightarrow \delta$ as a rule—, and since $\overline{\text{Premise}(r_{61})}$ is a stronger reason for δ than $\text{Premise}(r_{59})$. Since case c_{59} prioritises $\text{Premise}(r_{59})$ over $\text{Premise}(r_{61})$, an inconsistency is caused, thus rule r_{61} cannot be selected by the revised dimensional reason model.

To test whether rule r_{61} can be selected in the dimensional reduction model, we translate cases c_{59} and c_{61} to $c_{59} = \langle X_{59}^{r_{59}}, \delta \rangle$ and $c_{61} = \langle X_{61}^{r_{61}}, \pi \rangle$ where

$$\begin{aligned} X_{59}^{r_{59}} &= \{ \langle d_{\text{abroad}}, 12 \rangle, \langle d_{\text{inc.abroad}}, 60\% \rangle \} \text{ and} \\ X_{61}^{r_{61}} &= \{ \langle d_{\text{abroad}}, 24 \rangle, \langle d_{\text{inc.abroad}}, 65\% \rangle \}. \end{aligned}$$

Since $X_{61}^{r_{61}}$ is stronger for the defendant than $X_{59}^{r_{59}}$, i.e.,

$$X_{59}^{r_{59}} \leq^{\delta} X_{61}^{r_{61}},$$

the rule r_{61} cannot be selected by the dimensional reduction model either.

Another example that reinforces the necessity for a stronger notion of constraint was brought up by Thölke in his Master thesis [Thö24]. The intention behind this example is to show why further improvements beyond the revised dimensional reason model are necessary.

Example 40 (Defence rule). In this example from [Thö24], a certain rule for a fact situation is presented against a case base. Thölke argues that this rule should be rejected. While the revised dimensional reason model admits the rule, the dimensional reduction model rejects it, like Thölke argues it should.

Thölke uses the running example of dog adoption: In this scenario, an animal shelter processes dog adoption applications, and needs to decide, if an applicant should be allowed to adopt a dog. The two sides here are the decisions in favour of or against the applicant. If the plaintiff wins, the applicant is rejected, and if the defendant wins, the applicant is allowed to adopt a dog.

In his example, Thölke presents the scenario of a case base $\Gamma_{21} = \{c_{62}\}$ with a single case

$$\begin{aligned} c_{62} &= \langle X_{62}, r_{62}, \delta \rangle \text{ where} \\ X_{62} &= \{ \langle d_{\text{income}}, 1200\$ \rangle, \langle d_{\text{pet}}, \text{none} \rangle \} \text{ and} \\ r_{62} &= \{ M_{d_{\text{income}}, 1000\$}^{\delta} \} \rightarrow \delta. \end{aligned}$$

Two dimensions d_{income} and d_{pet} are defined here. The former denotes the monthly income of the applicant in dollars where a larger value improves the case for the defendant. The latter denotes the pet history of the applicant. For our example, it is sufficient to know that the relation looks as follows:

$$\text{none} \leq_{d_{\text{pet}}}^{\delta} \text{fish} \leq_{d_{\text{pet}}}^{\delta} \text{cat}.$$

So, if the applicant has owned a fish before they have a stronger case as compared to not having owned a pet before, and having owned a cat presents an even stronger case for the applicant. In case c_{62} the applicant has an income of 1200\$ ($\langle d_{\text{income}}, 1200\$ \rangle$), but has never owned a pet before ($\langle d_{\text{pet}}, \text{none} \rangle$). The animal shelter decides in favour

of the applicant because they earn at least 1000\$ monthly ($M_{d_{\text{income}},1000\$}^{\delta}$). Now a new applicant wants to adopt a dog. Their situation is given by

$$X_{63} = \{\langle d_{\text{income}}, 1100\$ \rangle, \langle d_{\text{pet}}, \text{fish} \rangle\}.$$

This applicant earns 1100\$ monthly, and previously owned a fish. We now make a decision using rule

$$r_{63} = \{M_{d_{\text{income}},1150\$}^{\pi}, M_{d_{\text{pet}},\text{cat}}^{\pi}\}.$$

This rule supports a decision in favour of π on the basis that the applicant earns at most 1150\$ and owned “at most” a cat. We know that $X_{62} \not\models \text{Premise}(r_{63})$, since $1200\$ \not\leq_{d_{\text{income}}^{\pi}} 1150$, and $\overline{\text{Premise}(r_{62})} \not\models \text{Premise}(r_{63})$, since dimension d_{pet} is present in $\text{Premise}(r_{63})$, but not in $\overline{\text{Premise}(r_{62})}$. Therefore, per the revised dimensional reason model presented in [Hor21] we can consistently decide fact situation X_{62} in favour of π .

Like Thölke, we argue that this definition of constraint does not make a lot of sense: We know that $X_{62} \not\models \text{Premise}(r_{63})$ itself is not a strong enough requirement such that the reason is respected because of the model collapse shown in Chapter 2. Thus, we also require that $\overline{\text{Premise}(r_{62})} \not\models \text{Premise}(r_{63})$. However, $\overline{\text{Premise}(r_{62})} \not\models \text{Premise}(r_{63})$ holds just because dimension d_{pet} is present in $\text{Premise}(r_{63})$ but not in $\overline{\text{Premise}(r_{62})}$. However, if d_{pet} were present in $\text{Premise}(r_{62})$ with value $X_{62}(d_{\text{pet}}) = \text{none}$, then we would have $\overline{\text{Premise}(r_{62})} \models \text{Premise}(r_{63})$. And because we also have $X_{60} \models \text{Premise}(r_{62})$, the decision would become constrained to δ , so the absence of d_{pet} blocks the constraint in favour of δ . But it is questionable why the absence of d_{pet} blocks the constraint. Because the addition of d_{pet} to $\text{Premise}(r_{62})$ generates a stronger reason that applies to fewer cases, fewer decisions should become constrained instead of more. This relates to the observation regarding strengthening reasons which we showed in Theorem 2. We challenge the revised dimension reason model, as we think a decision in favour of δ should not have been allowed in the first place here.

Let us argue semantically, why we believe that the given case should be constrained to π : Rule r_{62} in case c_{62} states that the decision in favour of δ is chosen on the basis that the applicant earns at least 1000\$ monthly ($M_{d_{\text{income}},1000\$}^{\delta}$). This property is also satisfied by X_{63} where the applicant earns 1100\$ monthly ($\langle d_{\text{income}}, 1100\$ \rangle$). For the other dimension d_{pet} in X_{62} , there is no threshold value in the reason of rule r_{62} . Thus, we assume that for requiring a decision in favour of δ , d_{pet} needs to be at least as strong for δ as given in X_{62} which is that they did not own a pet before ($\langle d_{\text{pet}}, \text{none} \rangle$). Since in X_{63} the applicant owned a fish before ($\langle d_{\text{pet}}, \text{fish} \rangle$), which is stronger for δ than not having owned a pet, we believe a decision in favour of δ should be required. Exactly this requirement is what the reduction model describes.

Let us now finally translate Γ_{21} to $\Gamma_{21}^{\times} = \{c_{62}^{\times}\}$ where $c_{62} = \langle X_{62}^{r_{62}}, \delta \rangle$ with

$$X_{62}^{r_{62}} = \{\langle d_{\text{income}}, 1000\$ \rangle, \langle d_{\text{pet}}, \text{none} \rangle\}.$$

X_{63} is stronger for δ than $X_{62}^{r_{62}}$, that is, $X_{62}^{r_{62}} \leq^{\delta} X_{63}$ because of $1000\$ \leq_{d_{\text{income}}^{\delta}} 1100\$$ and $\text{none} \leq_{d_{\text{pet}}^{\delta}} \text{fish}$. Thus, the dimensional reduction model requires that X_{63} is decided in favour of δ .

A final worry that remains when using dimensions in the reduction model is the basketball-height vignette discussed by Horty [Hor17]. In this example, there is only one relevant dimension. The issue at hand is for an ordinary person to judge whether a person is tall. The first individual to be classified is the former basketball player Kobe Bryant who is 1.98 m tall. The person judges that Kobe Bryant is tall and justifies the decision by stating that anyone over 1.8 m is considered tall. Next, the person is confronted with another former basketball player Isiah Thomas who is 1.85 m tall. Confronted with Isiah Thomas, the person might now realise that 1.85 m is, in fact, not that tall and wishes to decide that the basketball player is not to be considered particularly tall. By the reduction model, Isiah Thomas also has to be considered tall. Since this worry is not an artefact introduced by the translation of the case base, but rather represents a real change of judgement by decision-makers, we argue that the reduction model remains the simplest way of encoding stated thresholds, preserving monotonicity, and taking into account the reasons provided in cases.

3.3.2 Relation to the Reason Model

We now show formally that the reduction model has a stronger notion of constraint than the reason model. First, we assert that if a decision in favour of a side is required by the reason model, it is also required by the reduction model. Then, we show that there are cases constrained by the reduction model that are not constrained by the reason model.

Theorem 3 (Reason model constrains reduction model: dimensional). Let Γ be a dimensional case base and X be a dimensional fact situation. If a decision for X in favour of side s is required on Γ by the revised dimensional reason model, then a decision for X in favour of side s is required on Γ by the dimensional reduction model as well.

Proof. Let Γ be a dimensional case base and X be a dimensional fact situation. We argue by contraposition. Assume that Γ^\times does not require a decision in favour of either side. By Proposition 12, for case base Γ we can always select rule $r = U_X^s \rightarrow s$ for fact situation X , in case a decision in favour of some side s is possible. Since Γ^\times does not require a decision, for every case $\langle Y, \bar{s} \rangle \in \Gamma^\times$ we have $Y \not\prec_{\bar{s}} X$, that is, for some $\langle d, p \rangle \in Y$ and $\langle d, q \rangle \in X$ we have $p \not\prec_{\bar{s}} q$. Since

$$Y = \{\langle d, p \rangle \mid M_{d,p}^{\bar{s}} \in \text{Premise}(r')\} \cup \{\langle d, q \rangle \in Y' \mid \forall M_{d',p}^{\bar{s}} \in \text{Premise}(r') : d' \neq d\}$$

for some $\langle Y', r', \bar{s} \rangle \in \Gamma$, we know that for some $\langle d, q \rangle \in X$ there is either some $M_{d,p}^{\bar{s}} \in \text{Premise}(r')$ such that $p \not\prec_{\bar{s}} q$, or there is some $\langle d, p \rangle \in Y'$ with d not in $\text{Premise}(r')$ such that $p \not\prec_{\bar{s}} q$.

Case 1: There is some $M_{d,p}^{\bar{s}} \in \text{Premise}(r')$ such that $p \not\prec_{\bar{s}} q$. This means that $X \not\models \{M_{d,p}^{\bar{s}}\}$, because of q being weaker for side \bar{s} than p , and thus $X \not\models \text{Premise}(r')$. Since $M_{d,p}^{\bar{s}} \in U_X^{\bar{s}}$, we can also assert that $\{M_{d,q}^{\bar{s}}\} \not\models \{M_{d,p}^{\bar{s}}\}$ for the same reason that q is weaker than p for \bar{s} . Because $U_X^{\bar{s}} = \overline{\text{Premise}(r)}$, we get $\overline{\text{Premise}(r)} \not\models \text{Premise}(r')$.

Case 2: There is some $\langle d, p \rangle \in Y'$ with d not in $\text{Premise}(r')$ such that $p \not\leq_d^s q$. Because of duality, we know that $q \not\leq_d^s p$. Therefore, we also know that Y' cannot be stronger for side s than X —i.e., $X \not\leq^s Y'$ —, since p in Y' is not stronger than or equal to q in X for side s . Since $X \not\leq^s Y'$, we know that Y' cannot satisfy U_X^s either, that is, $Y' \not\models U_X^s$. Because $U_X^s = \text{Premise}(r)$, we get $Y' \not\models \text{Premise}(r)$. Since there is some dimension d in Y' that is not in $\text{Premise}(r')$ —and equivalently not in $\overline{\text{Premise}(r')}$ —, but U_X^s contains all dimensions, we know that $\overline{\text{Premise}(r')} \not\models U_X^s$ and, again because $U_X^s = \text{Premise}(r)$, we get $\overline{\text{Premise}(r')} \not\models \text{Premise}(r)$.

Since we have either

1. $X \not\models \text{Premise}(r')$ and $\overline{\text{Premise}(r')} \not\models \text{Premise}(r')$, or
2. $Y \not\models \text{Premise}(r)$ and $\overline{\text{Premise}(r')} \not\models \text{Premise}(r)$,

by Proposition 11 we know that this decision for arbitrary side s is consistent—since $\Gamma \cup \{\langle X, r, s \rangle\}$ is consistent—and thus a decision in favour of either side is not required on Γ either. \square

This illustrates that the decision constraints of the dimensional reduction model are at least as strong as the constraints in the dimensional reason model. As we have already seen, the opposite is not the case here, since we can find cases that are constrained by the reduction model, but not by the reason model, thus proving that the reduction model has a stronger notion of constraint than the reason model.

Theorem 4 (Reason model decision not constrained by reduction model decision: dimensional). Let Γ be a consistent dimensional case base. If a dimensional fact situation X can only be decided in favour of side s for case base Γ by the dimensional reduction model, then X is not necessarily required to be decided in favour of s for case base Γ by the revised dimensional reason model as well.

Proof. Shown in Example 40. \square

To further illustrate the notion of constraint in the reduction model, we can again graphically represent decision constraints.

Example 41. Let $\Gamma_{22} = \{c_{64}, c_{65}\}$ be a consistent case base with opposing cases

$$\begin{aligned} c_{64} &= \langle \{ \langle d_{\text{activity}}, 30 \rangle, \langle d_{\text{read}}, 20 \rangle \}, r_{64}, \pi \rangle && \text{where } r_{64} = \{ M_{d_{\text{read}}, 30}^\pi \} \rightarrow \pi, \text{ and} \\ c_{65} &= \langle \{ \langle d_{\text{activity}}, 60 \rangle, \langle d_{\text{read}}, 35 \rangle \}, r_{65}, \delta \rangle && \text{where } r_{65} = \{ M_{d_{\text{activity}}, 50}^\delta, M_{d_{\text{read}}, 25}^\delta \} \rightarrow \delta. \end{aligned}$$

Translating the case base Γ_{22} to $\Gamma_{22}^\times = \{c_{64}^\times, c_{65}^\times\}$, we retain the cases

$$\begin{aligned} c_{64}^\times &= \langle \{ \langle d_{\text{activity}}, 30 \rangle, \langle d_{\text{read}}, 30 \rangle \}, \pi \rangle, \text{ and} \\ c_{65}^\times &= \langle \{ \langle d_{\text{activity}}, 50 \rangle, \langle d_{\text{read}}, 25 \rangle \}, \delta \rangle. \end{aligned}$$

Suppose we now want to check the decision constraints of the following five fact situations

$$\begin{aligned} X_{66} &= \{\langle d_{\text{activity}}, 13 \rangle, \langle d_{\text{read}}, 25 \rangle\}, \\ X_{67} &= \{\langle d_{\text{activity}}, 25 \rangle, \langle d_{\text{read}}, 16 \rangle\}, \\ X_{68} &= \{\langle d_{\text{activity}}, 60 \rangle, \langle d_{\text{read}}, 30 \rangle\}, \\ X_{69} &= \{\langle d_{\text{activity}}, 75 \rangle, \langle d_{\text{read}}, 40 \rangle\}, \\ X_{70} &= \{\langle d_{\text{activity}}, 45 \rangle, \langle d_{\text{read}}, 10 \rangle\}. \end{aligned}$$

By case base Γ_{22} , fact situations X_{66} and X_{67} are constrained by case c_{64}^\times because $\text{Facts}(c_{64}^\times) \leq^\pi X_{66}$ and $\text{Facts}(c_{64}^\times) \leq^\pi X_{67}$, and fact situations X_{68} and X_{69} are constrained by case c_{65}^\times because $\text{Facts}(c_{65}^\times) \leq^\pi X_{68}$ and $\text{Facts}(c_{65}^\times) \leq^\pi X_{69}$. Since $X_{68} \models \text{Premise}(c_{65})$ and $\text{Premise}(c_{65}) \Vdash U_{X_{68}}^\pi$, fact situation X_{68} is also constrained by case c_{65} in the reason model. Since $\text{Facts}(c_{64}) \leq^\pi X_{67}$, fact situation X_{67} is also constrained by c_{64}^* in the result model, and since $\text{Facts}(c_{65}) \leq^\delta X_{69}$, fact situation X_{69} is also constrained by c_{65}^* in the result model. Fact situation X_{70} is not constrained in any model.

We can visualise these results in a two-dimensional coordinate system along the dimensions. This is shown in Figure 3.1. Any fact situation that falls into a red square bounded by a case is constrained to be decided in favour of that case in the result model. Any fact situation that falls into an orange square bounded by a case is constrained to be decided in favour of that case in the reason model. And any fact situation that falls into a yellow square bounded by a case is constrained to be decided in favour of that case in the reduction model.

Interestingly, in case c_{64} the constraints collapse with the result model, and in case c_{65} the constraints collapse with the reduction model. This is because of how the decision constraint is defined. By Proposition 12, a decision in favour of s for some fact situation X is required against a case base Γ if and only if there is a case $\langle Y, r, s \rangle \in \Gamma$ such that

1. $X \models \text{Premise}(r)$, and
2. $Y \leq^s X$ or $\overline{\text{Premise}(r)} \Vdash U_X^s$.

We can see that in the second condition $Y \leq^s X$ corresponds to the notion of constraint by the result model. What is more interesting is $\overline{\text{Premise}(r)} \Vdash U_X^s$. If $\text{Premise}(r)$ does not contain all dimensions—this is the case for c_{64} —then $\overline{\text{Premise}(r)} \Vdash U_X^s$ is unsatisfiable. Thus, if $\text{Premise}(r)$ does not contain all dimensions, the decision constraint is identical to the result model. But, if $\text{Premise}(r)$ does contain all dimensions—this is the case for c_{65} —then $\overline{\text{Premise}(r)} \Vdash U_X^s$ is equivalent to $Y^r \leq^s X$. Thus, if $\text{Premise}(r)$ does contain all dimensions, the decision constraint is identical to the reduction model. This poses an interesting question of whether the notion of constraint in the revised dimensional reason model may be ill-defined as well, since it collapses with the result model for every reason that does not contain all dimensions.

This example illustrates that the dimensional reduction model elevates the principle of monotonicity, that applies to the dimensional result model, to provide a stronger notion of constraint. Further, it provides an insight how the reason model relates to these models, and why this new notion of constraint is more reasonable.

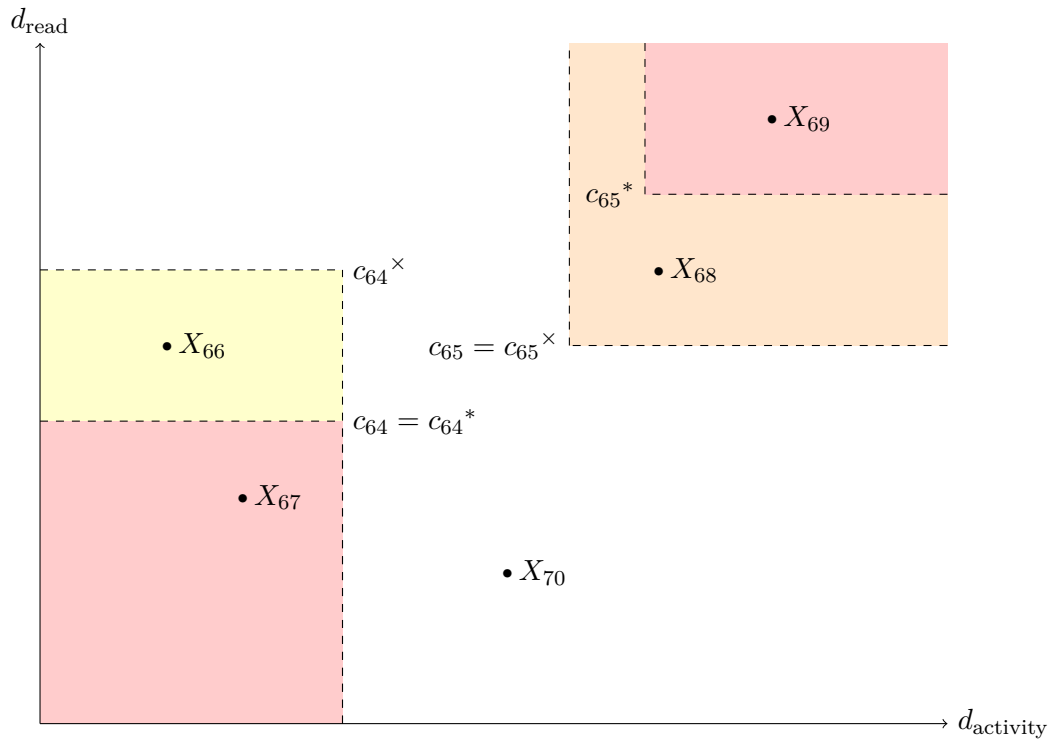


Figure 3.1: Decision constraints in the dimensional reduction model

3.3.3 Relation to the Reason Model with Complete Rules

The reduction model we presented is very similar to the dimensional reason model with complete rules Prakken proposed in [Pra21]. Where the dimensional reason model allows rules with reasons that include only a subset of dimensions present in the case, Prakken’s model requires that every reason includes *all* dimensions. He calls rules with such reasons *complete*. He defines constraint as follows.

Definition 36 (Reason model with complete rules decision constraint: dimensional). Let Γ be a consistent dimensional case base containing only complete rules, and let X be a dimensional fact situation. Then, a decision in favour of side s is required for X if and only if there is a case $\langle Y, r, s \rangle \in \Gamma$ such that $X \models \text{Premise}(r)$.

As we can easily see, the reduction model uses the same principle as the reason model with complete rules, since the reduced fact situations are analogous to complete rules. However, Prakken does not define a translation of the reason model to the reason model

with complete rules. Furthermore, our model is motivated differently through the idea that strengthening reasons should always be possible.

3.3.4 Strengthening Reasons

In Theorem 2 we outlined that in the dimensional reason model it is not always possible to make reasons stronger while preserving consistency. In fact, we argued that this is an important property that the model should have but lacks. Let us first revisit the counterexample from Theorem 2.

Example 42. We revisit case base $\Gamma_{19} = \{c_{54}, c_{55}\}$ with opposing cases

$$c_{54} = \langle \{ \langle d_{\text{activity}}, 90 \rangle, \langle d_{\text{read}}, 30 \rangle \}, r_{54}, \pi \rangle \quad \text{where } r_{54} = \{ M_{d_{\text{activity}},100}^{\pi}, M_{d_{\text{read}},40}^{\pi} \} \rightarrow \pi, \text{ and}$$

$$c_{55} = \langle \{ \langle d_{\text{activity}}, 120 \rangle, \langle d_{\text{read}}, 20 \rangle \}, r_{55}, \delta \rangle \quad \text{where } r_{55} = \{ M_{d_{\text{activity}},60}^{\delta} \} \rightarrow \delta.$$

We first perform the model reduction for the cases c_{54} and c_{55} to c_{54}^{\times} and c_{55}^{\times} where we get

$$\text{Facts}(c_{54}^{\times}) = \text{Facts}(c_{54})^{r_{54}} = \{ \langle d_{\text{activity}}, 100 \rangle, \langle d_{\text{read}}, 40 \rangle \}, \text{ and}$$

$$\text{Facts}(c_{55}^{\times}) = \text{Facts}(c_{55})^{r_{55}} = \{ \langle d_{\text{activity}}, 60 \rangle, \langle d_{\text{read}}, 20 \rangle \}.$$

The case base is consistent since we have $\text{Facts}(c_{54}^{\times}) \not\leq^{\pi} \text{Facts}(c_{55}^{\times})$, because neither dimension d_{activity} is stronger in c_{54}^{\times} than in c_{55}^{\times} , nor d_{read} . We can now replace rule r_{55} with the following rule

$$r_{56} = \{ M_{d_{\text{activity}},60}^{\delta}, M_{d_{\text{read}},20}^{\delta} \} \rightarrow \delta.$$

This rule merely adds the magnitude factor $M_{d_{\text{read}},20}^{\delta}$ to $\text{Premise}(r_{55})$. Clearly, $\text{Facts}(c_{55}) \models \text{Premise}(r_{56})$ and $\text{Premise}(r_{56}) \Vdash \text{Premise}(r_{55})$, thus it is a strengthened rule. When we now perform the model reduction with the changed rule, we get

$$\text{Facts}(c_{55})^{r_{56}} = \{ \langle d_{\text{activity}}, 60 \rangle, \langle d_{\text{read}}, 20 \rangle \}.$$

Since $\text{Facts}(c_{55})^{r_{56}} = \text{Facts}(c_{55}^{\times})$, the case reduction remains unchanged. Therefore, the case base with the strengthened rule r_{56} is also consistent.

We will now show formally that it is always possible to consistently strengthen reasons in the dimensional reduction model.

Theorem 5 (Reduction model consistency of strengthening a reason: dimensional). Let

$$\Gamma \cup \{ \langle X, W \rightarrow s, s \rangle \}$$

be a consistent case base in the dimensional reduction model and Z be a reason for side s such that

$$X \models Z \text{ and } Z \Vdash W.$$

Then, $\Gamma \cup \{ \langle X, Z \rightarrow s, s \rangle \}$ is also consistent in the dimensional reduction model.

Proof. Let $\Gamma \cup \{\langle X, r, s \rangle\}$ be a consistent case base in the dimensional reduction model where $r = W \rightarrow s$, and $r' = Z \rightarrow s$ be a rule where Z is reason such that $X \models Z$ and $Z \Vdash W$. Since the case base is consistent, we know that for every case $\langle Y', \bar{s} \rangle \in \Gamma^\times$ we have $Y' \not\leq^{\bar{s}} X^r$. We know that $X^r \leq^s X^{r'}$, because $Z \Vdash W$. This is equivalent to $X^{r'} \leq^{\bar{s}} X^r$. From this and $Y' \not\leq^{\bar{s}} X^r$ we can derive that $Y' \not\leq^{\bar{s}} X^{r'}$. Thus, $\Gamma \cup \{\langle X, r', s \rangle\}$ is consistent as well. \square

We motivated the necessity for this property by arguing that strengthening a reason such that it is satisfied by fewer cases should never cause more cases to be constrained, and showed that this property holds in the standard reason model. The dimensional reason model, however, does not meet this requirement, as shown in Theorem 2, and therefore cannot guarantee that consistency is preserved when a reason is strengthened. The dimensional reduction model improves upon the dimensional reason model by enabling the strengthening of reasons in the dimensional setting while also preserving the strength in constraint imposed by the dimensional reason model.

Summary and Outlook

In this chapter we introduced a definition for reducing a case base in the reason model to a case base in the result model. In the standard setting, we could reduce the case base in such a way that the reduction model had the same decision constraints as the reason model. We then adapted this idea to the dimensional setting which led to an even stronger notion of constraint. In Chapter 4 we will implement the dimensional reason model and the dimensional reduction model, and in Chapter 5, we will generate and analyse a case study that allows us to compare the behaviour of reason model and the reduction model on a larger dataset.

Implementation

We implement and compare the revised dimensional reason model, and the dimensional reduction model. As a programming language for our implementation we choose *answer set programming* (ASP) which is a logic programming language [Lif19]. Because ASP is closely related to first order logic, the constraints of the chosen models can be easily translated. We are interested in two properties:

1. Is a given case base consistent?
2. What decisions can and cannot be made for a matching fact situation against a consistent case base?

With our implementation we aim to automatically answer these questions.

The following chapter is divided into five sections. Since this thesis aims to be self-contained, in Section 4.1 we briefly describe the syntax and semantics of ASP. All of the remaining sections outline how the theoretical frameworks described in Chapters 2 and 3 are translated into ASP. In Section 4.2 we specify how the case base Γ along with the defined dimension set D is to be translated. In Section 4.3 we define rules that are shared between both implementations of the models. We then give a high-level overview of the approach in Section 4.4 and how non-monotonicity is utilised, before specifying the rules specific to the models in Sections 4.5 and 4.6.

Each of these sections will feature definitions of many ASP predicates. Every time we *define* a predicate we will first outline what information it describes semantically. Afterwards, we *specify* the rules in ASP necessary to retrieve that information. The words *define* and *specify* are used to distinguish between describing the semantics of predicates and putting them in the program.

Note that this implementation aims for simplicity. Predicates introduced generally have two purposes in this implementation, that is, either (a) they convey semantically-relevant

information, or (b) they simplify the implementation and are utilities. For predicates of form (a) the semantics of predicates will be related to the mathematical definition given by the reason model where useful.

The complete implementations of both the dimensional reason model and the dimensional reduction model can be found in the appendix as well as in the repository that can be found at <https://git.logic.at/cbr-asp/cbr-asp>.

We will now start with an introduction to ASP.

4.1 Answer Set Programming

Answer set programming (ASP) is a logic programming language with features that enable non-monotonic reasoning through negation as failure. To start with a short example of how an answer set program could look like, consider this set of formulas in first order logic:

$$\{ \text{is_child}(c), \text{is_crying}(c), \forall x((\text{is_child}(c) \wedge \text{is_crying}(x)) \rightarrow \text{comfort}(x)) \}$$

The formulas is a contain predicates $\text{is_child}(c)$ and $\text{is_crying}(c)$ for the constant c , and a for-all relation $\forall x((\text{is_child}(x) \wedge \text{is_crying}(x)) \rightarrow \text{comfort}(x))$ which states that for any x , if $\text{is_child}(x)$ and $\text{is_crying}(x)$ are present, then $\text{comfort}(x)$ is present as well. In ASP, we could model this set of formulas like so:

```
1 is_child(c).
2 is_crying(c).
3 comfort(X) :- is_child(X), is_crying(X).
```

Constants in ASP are written in lowercase, e.g., c , whereas variables in ASP are written in uppercase, e.g., x . \forall -statements are given implicitly by using variables in ASP. We can also see that implications $a \rightarrow b$ are written in reverse as $b :- a$. Where formulas in first order logic can admit *models*, in ASP we speak of *answer sets* that can contain positive and negative literals.

Formally, every rule in ASP consists of a head, and a body such that

$$\langle \text{head} \rangle :- \langle \text{body} \rangle .$$

If the body is empty, we do not write $:-$, for example, a . We also refer to such rules as *facts*. Conversely, if the head part of the rule is empty, e.g., $:- a$. we treat it as a *constraint* that forbids a from being present in the answer set.

A distinguishing feature of ASP is that there are two types of negation, that is, *strong negation* and *default negation*. The former, strong negation—written as “ \sim ”—is the same type of negation as in first order logic. For example, we can write $b :- \sim a$. which states that if $\sim a$ is present in the answer set, then b must be present in the answer set as well. Of course, any program that derives only answer sets containing both a and $\sim a$ is

unsatisfiable. Conversely, default negation—written as “**not**”—allows for non-monotonic reasoning. For example, we can write $b \text{ :- not } a$, which states that if a is *absent* in an answer set, then b must be present.

For a set S of literals to be an answer set of program \mathcal{P} , we require for every rule

$$H \text{ :- } B_1, B_2, \dots, B_n, \text{ not } C_1, \text{ not } C_2, \dots, \text{ not } C_m.$$

in \mathcal{P} that if every predicate B_1, B_2, \dots, B_n is present in S , and if every predicate C_1, C_2, \dots, C_m is absent in S , then H must be present in S as well. Furthermore, every literal in S has to be supported by some rule in \mathcal{P} .

Consider the following answer set program:

```
1 b :- not a.
```

This program has exactly one answer set $S = \{b\}$. The set $S' = \{a\}$ is not answer of the program, because a is not supported by any rule in the program. Because every literal has to be supported by some rule, the following program does not have an answer set:

```
1 a :- not a.
```

If we choose set $S = \emptyset$, the predicate a is missing despite being required by rule $a \text{ :- not } a$. Thus S cannot be an answer set. Conversely, if we choose set $S' = \{a\}$, then the predicate a is not supported by any rule, thus S' cannot be an answer set either. Of course, we can also define programs that have several answer sets, for example:

```
1 a :- not b.
2 b :- not a.
```

This program has two answer sets. The first answer set is $S_1 = \{a\}$, since a is supported by rule $a \text{ :- not } b$. and rule $b \text{ :- not } a$. is not applicable. The second answer set is $S_2 = \{b\}$, since b is supported by rule $b \text{ :- not } a$. and rule $a \text{ :- not } b$. is not applicable. Set $S_3 = \{a, b\}$ is not an answer set of the program, because neither a nor b are not supported by any rule.

ASP enables *non-monotonic reasoning*. In monotonic reasoning, the more knowledge we have of a situation, the more information we can derive. In non-monotonic reasoning this does not necessarily hold true. We illustrate this with a famous example. Consider the following program:

```
1 % facts
2 bird(tweety).
3
4 % rules
5 flies(X) :- bird(X), not penguin(X).
```

This program states that Tweety is a bird—given by `bird(tweety)`. We have one rule which states that any bird can fly, except if we know the bird to be a penguin. Since

we do not know, if Tweety is a penguin, we can assume that Tweety can fly. Thus, the answer set of the program is

$$S = \{\text{bird}(\text{tweety}), \text{flies}(\text{tweety})\}.$$

Let us now look at a variation of the program. By adding fact `penguin(tweety)` . we get:

```

1 % facts
2 bird(tweety).
3 penguin(tweety).
4
5 % rules
6 flies(X) :- bird(X), not penguin(X).

```

Since we now know that Tweety is a penguin, we can no longer apply the rule given in the program. Thus, the answer set of this program is

$$S' = \{\text{bird}(\text{tweety}), \text{penguin}(\text{tweety})\}.$$

Where in the first program, we could derive `flies(tweety)` ., in this new program that contains a larger set of facts, we can no longer derive that predicate. This illustrates how in non-monotonic contexts more knowledge of a situation does not necessarily enable us to derive more information.

4.2 Input

We now translate the case information passed to the programs. It consists of two parts. Firstly, the programs have to have knowledge of every dimension in $d \in D$. Particularly, the value relation \leq_d^s has to be constructed for every side s . Secondly, every case $c \in \Gamma$ has to be translated. Since the case base Γ depends on the definition of the dimension set D , we start by defining the dimensions and their relations.

4.2.1 Dimensions

Every dimension $d \in D$ will be assigned a unique id in ASP. So for dimension set $D = \{d_1, d_2, d_3, \dots\}$ we can define ids `d1`, `d2`, `d3`, `...`. These ids are just examples; they are not limited to numbers. For example, dimensions d_{screen} or d_{activity} could be assigned ids `screen` and `activity`.

Every dimension $d \in D$ contains a range $[d]$ of values ranging from favouring side \bar{s} to favour side s . The reason model does not specify, if $[d]$ is limited to a set of values or can have infinitely many. For many dimensions, like in our running example, integer values are used. However, in some dimensions, like in d_{protest} or d_{coop} , custom values are employed. While it is possible to translate these custom values to integers, we will provide a mechanism to specify these in the answer set program to increase readability.

When specifying a dimension d , the program has to know how the values in range $[d]$ are ordered. For a range $[d]$ we have a total ordering

$$\cdots \leq_d^s p_i \leq_d^s p_{i+1} \leq_d^s p_{i+2} \leq_d^s \cdots$$

for side s and by the dual condition we have the reverse ordering

$$\cdots \leq_d^{\bar{s}} p_i \leq_d^{\bar{s}} p_{i-1} \leq_d^{\bar{s}} p_{i-2} \leq_d^{\bar{s}} \cdots$$

for the opposing side \bar{s} . Thus, knowing the ordering for one side is sufficient.

Let us now consider the first case where a dimension d uses integer values. Usually, when integer values are used, they are ordered naturally, i.e., for side s we have either

$$\begin{aligned} \cdots &\leq_d^s 1 \leq_d^s 2 \leq_d^s 3 \leq_d^s \cdots, \text{ or} \\ \cdots &\leq_d^s 3 \leq_d^s 2 \leq_d^s 1 \leq_d^s \cdots \end{aligned}$$

and the opposite ordering for side \bar{s} . Thus, if a dimension has an ordering of integer values, we know that one side s has a relation \leq_d^s correspondent to the natural ordering \leq and the opposite side \bar{s} has a relation $\leq_d^{\bar{s}}$ correspondent to \geq . Thus, it is sufficient to specify which side corresponds to ordering \leq .

For the sides plaintiff π and defendant δ , we define the constants `plaintiff` and `defendant` in ASP. To tell the model which side corresponds to \leq , we define predicate

```
int_dimension_strengthens(Dimension, Side).
```

We use the word “strengthens” in the sense that a larger integer value corresponds to a stronger case for the described side. Here `Dimension` is the specified id of the dimension d to consider, and `Side` is the side s , which is either `plaintiff` or `defendant`. The presence of the predicate for dimension d and side s conveys the information that (a) range $[d]$ contains integer values, and (b) \leq_d^s corresponds to \leq .

Example 43. Out of the five dimensions specified in our running example, three use integer values. That is, d_{screen} describes the minutes of screen time the child had, d_{activity} describes the minutes of physical activity the child engaged in, and d_{read} describes the number of pages the child read that day. We know for π that $\leq_{d_{\text{screen}}}^{\pi}$ corresponds to \leq whereas for δ we know that $\leq_{d_{\text{activity}}}^{\delta}$ and $\leq_{d_{\text{read}}}^{\delta}$ correspond to \leq . We use ids `screen`, `activity`, and `read` for the dimensions and specify the predicates like so:

```
1 int_dimension_strengthens(screen, plaintiff).
2 int_dimension_strengthens(activity, defendant).
3 int_dimension_strengthens(read, defendant).
```

Listing 4.1: Integer dimensions example

Integers already admit a natural ordering. Therefore, defining a single predicate is sufficient to describe such dimensions. In the second case, where dimensions contain arbitrary values, we have to provide an ordering ourselves. In this scenario, the ordering only has to contain all values that occur in Γ to correctly model the relations between these values. So in the definition of such values for some dimension d , we can consider $[d]$ to be finite.

If a dimension d has range $[d] = \{p_1, p_2, \dots, p_n\}$ with arbitrary values for p_1, p_2, \dots, p_n ranging from favouring side \bar{s} to favouring side s , we define a predicate

```
custom_dimension_strengthens(Dimension, Side).
```

Again, this predicate identifies the side s given by `side` that the “larger” values favour for dimension d given by `Dimension`. We consider p_1 the “smallest” value where p_n is considered the “largest” here. Since compared to integers there is no natural ordering for arbitrary values, we define our own ordering through predicate

```
custom_dimension_ordering(Dimension, Current, Next).
```

For dimension with id `Dimension`, for every value `Current` in $[d]$, we define a value `Next` in $[d]$ that is the next value in the range, given there is one. So for $p_i \in [d]$, the next value is p_{i+1} . With this definition, we can define a custom ordering by specifying a successor for every value. Combined with the `custom_dimension_strengthens`-predicate we have an ordering and a side that the “larger” values strengthen. Note that if there is only one value in $[d]$, that is, $[d] = \{p\}$, then we do not need to specify an ordering.

Example 44. Out of the five dimensions used in the running example, two use custom values. That is, d_{protest} admits values $[d_{\text{protest}}] = \{\text{none, complained, cried, screamed, kicked}\}$ where “none” is the strongest value for δ and “kicked” is the strongest value for π , and d_{coop} admits values $[d_{\text{coop}}] = \{\text{physical, verbal, ignored, played}\}$ where “physical” is the strongest value for π and “played” is the strongest value for δ . We use ids `protest`, and `coop` for the dimensions and specify the predicates like so:

```
1 custom_dimension_strengthens(protest, plaintiff).
2 custom_dimension_strengthens(coop, defendant).
3
4 custom_dimension_ordering(protest, none, complained).
5 custom_dimension_ordering(protest, complained, cried).
6 custom_dimension_ordering(protest, cried, screamed).
7 custom_dimension_ordering(protest, screamed, kicked).
8
9 custom_dimension_ordering(coop, physical, verbal).
10 custom_dimension_ordering(coop, verbal, ignored).
11 custom_dimension_ordering(coop, ignored, played).
```

Listing 4.2: Custom dimensions example

Note that reversing the ordering while also switching out `plaintiff` and `defendant` with their opposing sides is equivalent by the dual condition.

4.2.2 Solved Cases

Next, we will translate all cases found in the case base Γ . Every case $\langle X, r, s \rangle \in \Gamma$ holds several pieces of information. That is, the fact situation X , the rule r consisting of a reason $\text{Premise}(r)$ and a conclusion $\text{Conclusion}(r)$, and a winning side s . Since $\text{Conclusion}(r) = s$, we only need to consider fact situation X , reason $\text{Premise}(r)$ and winning side s . In each fact situation for each dimension d we have an assigned value $X(d)$. In a reason $\text{Premise}(r)$ we have several magnitude factors $M_{d,q}^s \in \text{Premise}(r)$ for some but not necessarily all dimensions.

Like with every dimension, every case in the case base can be assigned an id. For case base $\Gamma = \{c_1, c_2, c_3, \dots\}$, we can define ids c_1, c_2, c_3, \dots in the program. Every fact situation and every reason belongs to some solved case in the case base. Therefore, every value assignment $\langle d, p \rangle$ and every magnitude factor $M_{d,p}^s$ belong to some case as well. We can define cases in a manner similar to relational databases where we store a foreign key of the case id in the predicates describing the value assignments and magnitude factors. Thus, for every case $c \in \Gamma$ where $c = \langle X, r, s \rangle$ we define the following predicates:

1. Exactly one `case(Case, WinningSide)` defines the case where `Case` refers to the id of the case c and `WinningSide` defines the winning side s as either `plaintiff` or `defendant`.
2. Several value assignment predicates `value_assignment(Case, Dimension, Value)` define the fact situation X of c . Every value assignment $\langle d, p \rangle \in X$ contains the case id `Case` referring to the id of case c it belongs to, the `Dimension` refers to the dimension d specified, and `Value` refers to the assigned value p . We link $\langle d, p \rangle$ directly to the case, instead of some fact situation id, since every fact situation in the case base always belongs to a case.
3. Several magnitude factor predicates `magnitude_factor(Case, Dimension, ThresholdValue)` define the reason $\text{Premise}(r)$ of c . Every magnitude factor $M_{d,p}^s \in \text{Premise}(r)$ contains the case id `Case` referring to the id of case c it belongs to, the `Dimension` refers to the id of dimension d , and the `ThresholdValue` refers to the value $p \in [d]$. Again, we link $M_{d,p}^s$ directly to the case instead of some reason id, since every reason in the case base always belongs to a case. Do note that in the `magnitude_factor`-predicate we leave out the side s found in $M_{d,p}^s$. Since s always matches with the winning side in c , we can use the `magnitude_factor`-predicate together with the `case`-predicate to infer the side it refers to.

A translation of a case c is straightforward. We pick a unique id for `Case`, translate $\text{Outcome}(c)$ for `WinningSide`, and define predicate `case(Case, WinningSide)`. For every value assignment $\langle d, p \rangle \in \text{Facts}(c)$ we set the id of the case c in `Case`, the id of d in `Dimension`, translate the value p in `Value`, and use these values to define `value_assignment(Case, Dimension, Value)`. For every magnitude factor $M_{d,p}^s$ we set the id of the case c

in `Case`, the id of d in `Dimension`, translate the value of p to `ThresholdValue`, and use these values to define `magnitude_factor(Case, Dimension, ThresholdValue)`.

How can we reconstruct a case $c = \langle X, r, s \rangle$ from this information? X is retrieved by collecting all `value_assignment`-predicates with the value for `Case` matching the id of c . `Premise(r)` is retrieved by collecting all `magnitude_factor`-predicates with the value for `Case` matching the id of c , and inferring the side s stored in `Side` in the case-predicate matching the value for `Case`. `Conclusion(r)` and s are retrieved by looking at the `WinningSide` value in the case-predicate.

Example 45. Let $\Gamma_{23} = \{c_{71}\}$ be a case base with case

$$c_{71} = \langle X_{71}, r_{71}, \pi \rangle \text{ where}$$

$$X_{71} = \{\langle d_{\text{protest}}, \text{screamed} \rangle, \langle d_{\text{screen}}, 100 \rangle, \langle d_{\text{activity}}, 50 \rangle, \langle d_{\text{coop}}, \text{ignored} \rangle, \langle d_{\text{read}}, 4 \rangle\} \text{ and}$$

$$r_{71} = \{M_{d_{\text{protest}}, \text{screamed}}^{\pi}, M_{d_{\text{screen}}, 60}^{\pi}, M_{d_{\text{read}}, 10}^{\pi}\} \rightarrow \pi.$$

Fact situation X_{71} states that the child screamed ($\langle d_{\text{protest}}, \text{screamed} \rangle$), had 100 minutes of screen time ($\langle d_{\text{screen}}, 100 \rangle$), 50 minutes of physical activity ($\langle d_{\text{activity}}, 50 \rangle$), ignored its sibling ($\langle d_{\text{coop}}, \text{ignored} \rangle$), and read 4 pages throughout the day ($d_{\text{read}}4$). It was sent to bed early because the child protested by at least screaming ($M_{d_{\text{protest}}, \text{screamed}}^{\pi}$), had at least 60 minutes of screen time ($M_{d_{\text{screen}}, 60}^{\pi}$), and read at most 10 pages ($M_{d_{\text{read}}, 10}^{\pi}$). To translate this information, we assign id c_{71} to case c_{71} in ASP, and define one case-predicate, five `value_assignment`-predicates, and three `magnitude_factor`-predicates:

```

1 case(c71, plaintiff).
2
3 value_assignment(c71, protest, screamed).
4 value_assignment(c71, screen, 100).
5 value_assignment(c71, activity, 50).
6 value_assignment(c71, coop, ignored).
7 value_assignment(c71, read, 4).
8
9 magnitude_factor(c71, protest, screamed).
10 magnitude_factor(c71, screen, 60).
11 magnitude_factor(c71, read, 10).

```

Listing 4.3: Case definition example

4.2.3 Unsolved Cases

In a similar fashion to the cases in the case base Γ , we can also define fact situations awaiting to be decided by the program. We will reuse the `value_assignment`-predicates to define the value assignments in a new fact situation X . Since the `value_assignment`-predicate requires every value assignment to belong to a case, we will from now on refer to new fact situations as *unsolved cases*, purely to simplify defining them in the program. We specify the predicate

$$\text{unsolved_case}(\text{Case}).$$

Given a fact situation X , we pre-emptively assign a new case id to `Case` that will refer to the fact situation X . For every value assignment $\langle d, p \rangle \in X$ we also use the new id in `Case` for `value_assignment(Case, Dimension, Value)`. Like before, we set the id of d in `Dimension`, translate the value p to `Value`. Note that it is important that the id of the unsolved case does not overlap with any solved case in the case base Γ .

Example 46. We want to translate a fact situation presented to the agent for a decision

$$X_{72} = \{\langle d_{\text{protest}}, \text{none} \rangle, \langle d_{\text{screen}}, 30 \rangle, \langle d_{\text{activity}}, 80 \rangle, \langle d_{\text{coop}}, \text{played} \rangle, \langle d_{\text{read}}, 15 \rangle\}$$

into an unsolved case in the answer set program. To do so, we generate an id `c72` to refer to X_{72} . Then, we translate every $\langle d, p \rangle \in X_{72}$ like so:

```

1 unsolved_case(c72).
2
3 value_assignment(c72, protest, none).
4 value_assignment(c72, screen, 30).
5 value_assignment(c72, activity, 80).
6 value_assignment(c72, coop, played).
7 value_assignment(c72, read, 15).

```

Listing 4.4: Unsolved case definition example

4.3 Shared Rules

The following section contains definitions for predicates that are useful to both models we aim to implement. In particular, predicates that define side s and its opposite \bar{s} as well as predicates for the value relations \leq_d^s and $\leq_d^{\bar{s}}$ of each dimension d will be defined. These definitions are independent from the notion of constraint used. Thus, both models will utilise the definitions given below.

4.3.1 Sides and Opposite Sides

To define the two adversarial parties π and δ we have previously reserved the constants `plaintiff` and `defendant`. For every side s , we now start to define a predicate for the side itself as well as a predicate that provides information about its opposing side \bar{s} . A definition of the side itself is given by predicate

$$\text{side}(\text{Side})$$

where `Side` refers to s . To refer to the opposite \bar{s} of a side s , we define predicate

$$\text{opposite_side}(\text{Side}, \text{OppositeSide})$$

where `Side` again refers to s and `OppositeSide` refers to \bar{s} . Since there are only two sides, we simply need to define π and δ and opposites $\bar{\pi}$ and $\bar{\delta}$. We specify the predicates as follows:

```

1 side(plaintiff).
2 side(defendant).
3
4 opposite_side(plaintiff, defendant).
5 opposite_side(defendant, plaintiff).

```

Listing 4.5: Sides and opposite sides

4.3.2 Dimension Value Relation

We will now build the value relations \leq_d^s and $\leq_d^{\bar{s}}$ for every dimension d . Remember that we distinguished between two different types of dimensions: integer dimensions and custom dimensions. We define a predicate for the value relation that can be used for both types of dimensions so that the model implementations do not have to distinguish between the different kinds.

Since every dimension can contain a potentially infinite number of values, it may be impossible to fully specify a value relation. Because we only need to compare values present in the case base or in unsolved cases, we merely need to specify the value relation for the values actually used in the input. For this, we define predicate

```
dimension_value(Dimension, Value)
```

which denotes that a value `Value` is present for a dimension `Dimension` which refers to some dimension $d \in D$. `Value` is a translated value $p \in [d]$ for some dimension d . Thus, we infer the value relation for all potential values given in the case base or unsolved fact situations. We first retrieve all present values like so:

```

1 dimension_value(Dimension, Value) :-
2     value_assignment(Case, Dimension, Value).
3
4 dimension_value(Dimension, Value) :-
5     magnitude_factor(Case, Dimension, Value).

```

Listing 4.6: Inferring dimension values

Now we have knowledge of which values occur either in some fact situation X or in some reason $\text{Premise}(r)$.

We define the predicate

```
value_relation(Side, Dimension, Value1, Value2)
```

which expresses that for a dimension d referred to by `Dimension`, $p_1 \in [d]$ referred to by `Value1` is at most as strong for side s referred to by `Side` as value $p_2 \in [d]$ referred to by `Value2`. In short, it represents that $p_1 \leq_d^s p_2$ holds.

To specify the `value_relation`-predicate, we will now need to distinguish between integer dimensions, identified by the presence of the `int_dimension_strengthens`-predicate, and

other dimensions, identified by the `custom_dimension_strengthens`-predicate. For both types of dimensions we will first perform an initialization, afterwards we recursively specify the relation fully by inferring reflexivity, transitivity, and duality for the relation.

We start with integer dimensions. For every integer dimension $d \in D$, we initialise the value relation for the side s that was set to be “strengthened” by larger values. Or equivalently, the side where the relation \leq_d^s is equal to the natural ordering \leq :

```

1 value_relation(Side, Dimension, Value1, Value2) :-
2   int_dimension_strengthens(Dimension, Side),
3   dimension_value(Dimension, Value1),
4   dimension_value(Dimension, Value2),
5   Value1 <= Value2.
```

Listing 4.7: Initializing value relation for integer dimensions

Notice that this specification generates a reflexive and transitive value relation for side s for every value that is present. Every value that is not present is not considered for the value relation. The dual remains undefined for now on that integer dimension.

For custom dimensions, we utilise the `custom_dimension_ordering`-predicate. We can assume that every value identified by the `dimension_value`-predicate can be found in the ordering that was specified in the input. Otherwise, the dimension definition would be incomplete. We make the ordering transitive in the same way that integers are transitive by default. After adding transitivity, we can specify the value relation in the same way as we did for integer dimensions:

```

1 custom_dimension_ordering(Dimension, Value1, Value3) :-
2   custom_dimension_ordering(Dimension, Value1, Value2),
3   custom_dimension_ordering(Dimension, Value2, Value3).
4
5 value_relation(Side, Dimension, Value1, Value2) :-
6   custom_dimension_strengthens(Dimension, Side),
7   dimension_value(Dimension, Value1),
8   dimension_value(Dimension, Value2),
9   custom_dimension_ordering(Dimension, Value1, Value2).
```

Listing 4.8: Initializing value relation for custom dimensions

Notice that this specification generates a transitive value relation for side s for every value that is present. Again, every value that is not present is not considered in the value relation. Reflexivity and the dual remain undefined.

To define the missing properties required by Definition 21, we now fully construct the value relation. While for a minimal program only reflexivity and duality are needed to fully specify the value relation from the previous specifications, for completeness we specify all four properties of reflexivity, transitivity, antisymmetry, and duality:

```

1 % reflexivity
2 value_relation(Side, Dimension, Value, Value) :-
```

```

3   side(Side),
4   dimension_value(Dimension, Value).
5
6  % transitivity
7  value_relation(Side, Dimension, Value1, Value3) :-
8     value_relation(Side, Dimension, Value1, Value2),
9     value_relation(Side, Dimension, Value2, Value3).
10
11 % antisymmetry
12 :-
13     value_relation(Side, Dimension, Value1, Value2),
14     value_relation(Side, Dimension, Value2, Value1),
15     Value1 != Value2.
16
17 % duality
18 value_relation(OppositeSide, Dimension, Value2, Value1) :-
19     value_relation(Side, Dimension, Value1, Value2),
20     opposite_side(Side, OppositeSide).

```

Listing 4.9: Constructing the value relation

In addition to the value relation \leq_d^s we also specify the strict value relation $<_d^s$ —the relation $p <_d^s q$ is satisfied if and only if $p \leq_d^s q$ and $p \neq q$. We describe it by defining predicate

```
strict_value_relation(Side, Dimension, Value1, Value2).
```

Using the regular value relation we can easily specify its strict version in the program:

```

1  strict_value_relation(Side, Dimension, Value1, Value2) :-
2     value_relation(Side, Dimension, Value1, Value2),
3     Value1 != Value2.

```

Listing 4.10: Constructing the strict value relation

Example 47. We will now outline how the value relations are generated based on present cases. For this, let us consider case base $\Gamma_{24} = \{c_{73}, c_{74}\}$ with cases

$$c_{73} = \langle \{ \langle d_{\text{read}}, 30 \rangle, \langle d_{\text{coop}}, \text{played} \rangle \}, \{ M_{d_{\text{read}}, 20}^\pi, M_{d_{\text{coop}}, \text{ignored}}^\pi \} \rightarrow \pi, \pi \rangle \text{ and}$$

$$c_{74} = \langle \{ \langle d_{\text{read}}, 10 \rangle, \langle d_{\text{coop}}, \text{verbal} \rangle \}, \{ M_{d_{\text{coop}}, \text{verbal}}^\delta \} \rightarrow \delta, \delta \rangle.$$

We can assume the same definition for d_{read} and d_{coop} as given in Examples 43 and 44. A translation of cases c_{73} and c_{74} looks as follows:

```

1  case(c73, plaintiff).
2  value_assignment(c73, read, 30).
3  value_assignment(c73, coop, played).
4  magnitude_factor(c73, read, 20).
5  magnitude_factor(c73, coop, ignored).
6

```

```

7 case(c74, defendant).
8 value_assignment(c74, read, 10).
9 value_assignment(c74, coop, verbal).
10 magnitude_factor(c74, coop, verbal).

```

Listing 4.11: Case translation example

We now want to analyse what `value_relation`-predicates are inferred from the given cases. We start by looking at d_{read} . Evidently, three different values are present for d_{read} in the case base, that is, 30 in $\langle d_{\text{read}}, 30 \rangle \in \text{Facts}(c_{73})$, 20 in $M_{d_{\text{read}}, 20}^{\pi} \in \text{Premise}(\text{Rule}(c_{73}))$, and 10 in $\langle d_{\text{read}}, 10 \rangle \in \text{Facts}(c_{74})$. Thus, the value relations $\leq_{d_{\text{read}}}^{\pi}$ and $\leq_{d_{\text{read}}}^{\delta}$ are defined as if 10, 20, and 30 were the only values in range d_{read} . This is because only these values are inferred for the `dimension_value`-predicates. For dimension d_{coop} , we have values “played” in $\langle d_{\text{coop}}, \text{played} \rangle \in \text{Facts}(c_{73})$, “ignored” in $M_{d_{\text{coop}}, \text{ignored}}^{\pi} \in \text{Premise}(\text{Rule}(c_{73}))$, and “verbal” in $\langle d_{\text{coop}}, \text{verbal} \rangle \in \text{Facts}(c_{74})$ and $M_{d_{\text{coop}}, \text{verbal}}^{\delta} \in \text{Premise}(\text{Rule}(c_{74}))$. Thus, the value relations $\leq_{d_{\text{coop}}}^{\pi}$ and $\leq_{d_{\text{coop}}}^{\delta}$ are defined as if “played”, “ignored”, and “verbal” were the only values in range d_{coop} .

The defined predicates will be central to the model implementations, as they enable to check for consistency and decision constraints. In the following sections we will specify definitions unique to the respective models. There will be overlaps in the predicate definitions, but these will not interfere with each other, since the models are considered to be completely separate programs that are based on the shared rules defined in this section.

4.4 Approach Outline

Before we describe the implementations in detail in Sections 4.5 and 4.6, let us outline our approach on a high level first. As described before, we want to verify both the consistency of a given case base as well as check what decisions can be made for a newly presented fact situation. In fact, the latter check can be reduced to the check of consistency, since a decision constraint for side s is merely a test of whether a decision in favour \bar{s} is inconsistent with the case base.

Our program therefore consists two types of case-like structures:

1. Cases that are found in the case base.
2. Proposed case decisions for new fact situations. For every newly presented fact situation we infer two possible case decisions, one in favour of π and one in favour of δ .

In our implementations, we first compare all opposing cases found in the case base with each other. If we find an inconsistency, the program outputs the inconsistent pairs. If the

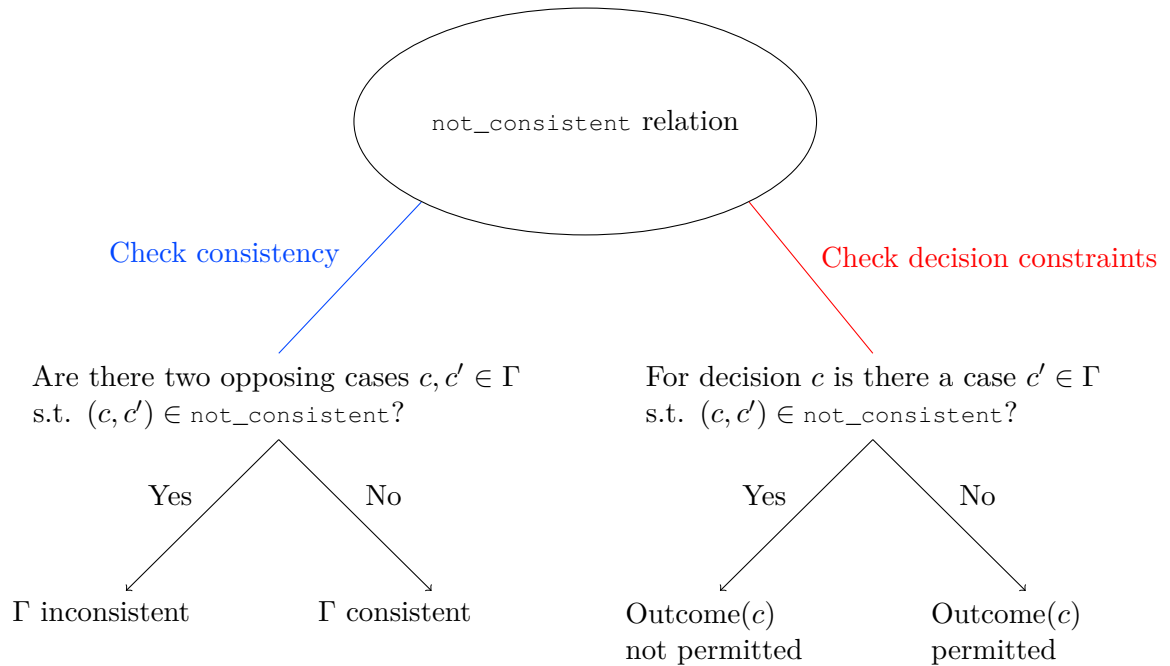


Figure 4.1: Visual representation of implementation: First the `not_consistent` relation is constructed. Then, checks for consistency and decision constraints are performed.

program is consistent, we continue to checking what case decisions are consistent with the case base. For every fact situation it is possible that either (1) both case decisions are possible, i.e., there is no decision constraint, or (2) a decision constraint applies in favour of one of the two sides. In the latter scenario where a decision constraint applies to side s for a fact situation, our program would find an inconsistency between the decision for the opposite side \bar{s} and the case base.

Since the check of consistency and decision constraint are essentially the same, but with different entities that are compared, we aim to first construct a relation that matches all case-like entities, independent of whether they are in the case base or a proposed decision. This relation will be given by the `not_consistent`-predicate which contains all case-like pairs that are not consistent with each other. To check for the consistency of the case base, we simply need to check if there is any pair of cases from the case base in the relation. If there is, the whole case base is inconsistent. To check if a decision in favour of side s is constrained, we have to find a case from the case base that forbids a decision in favour of the opposite side \bar{s} in the relation. This description is represented graphically in Figure 4.1.

Note that we use a “negative” predicate like `not_consistent` rather than a positive predicate like `consistent` to describe the relation between cases. As we will soon see, most predicates we define to compare case-like entities will be negative. But why do we define them this way? To give an intuition on why we primarily use predicates in the

negative form, consider the check of whether

$$W \Vdash Z$$

holds, where W and Z are reasons for side s . To test this condition, we need to check if for every magnitude factor for dimension d in W , there is either a magnitude factor for d in Z that is weaker for side s , or a magnitude factor for d does not exist in Z . But, non-monotonically we can more easily describe the condition as follows: $W \Vdash Z$ holds, *unless* there is a magnitude factor for dimension d that is either stronger for s in Z , or that is present in Z but not in W . In our problem domain it is mostly sufficient to check for witnesses that prove the opposite of the condition we want to assert, and then test for the condition non-monotonically using weak negation. So, a test of whether $W \not\Vdash Z$ holds could look as follows in ASP:

```

1 not_entails(W, Z) :- has_weaker_magnitude_factor_than(W, Z).
2 not_entails(W, Z) :-
3     has_magnitude_factor(Z, D),
4     not has_magnitude_factor(W, D).

```

If $W \not\Vdash Z$ holds, we can infer `not_entails(W, Z)`. If we now want to test for $W \Vdash Z$ in another predicate, we can now use weak negation:

```

1 entails(W, Z) :- reason(W), reason(Z), not not_entails(W, Z).

```

The same principle applies to comparing the consistency of cases: Two cases are consistent, *unless* there is some condition that causes an inconsistency. And a decision is allowed, *unless* there is some case in the case base that conflicts with that decision. This idea will be central to the implementations in the following sections.

Both the reason model and the reduction model benefit from a non-monotonic implementation, since a larger knowledge base does not necessitate that more knowledge can be derived. More specifically, the larger the case base grows the fewer decisions are permitted, which is why ASP is a good fit for implementing these models.

4.5 The Dimensional Reason Model

In this section, we will provide an implementation of the reason model in the dimensional setting, using the revised notion of consistency as it was proposed by Horty [Hor21] and given by Proposition 11. There are two main aspects of the reason model we want to capture: Firstly, we want to be able verify the consistency of a case base. If a case base is inconsistent we want to be able to pinpoint which cases are at fault for this. Secondly, we want to check what decisions are permitted for unsolved cases. Since the latter problem relies on the consistency evaluation, we will start by implementing the consistency check and then enable making decisions utilising the consistency framework.

4.5.1 Case Base Consistency

In the reason model, consistency is the measure that is used to check both the integrity of a case base Γ as well as what decisions are permissible for some matching fact situation X against a case base Γ . To account for both of these use cases, we will define predicates to be used for consistency that can be reused for the decision-making process. These generic predicates can describe solved cases in the case base and decisions made for unsolved cases alike.

Both solved and unsolved cases use the same `value_assignment`-predicates to describe fact situations. However, unsolved cases have neither a winning side nor a reason. Therefore, we define these predicates generically:

1. A generic case is defined by predicate `case_like(Case, Side)`. It represents a case-like object with an id `Case` and a winning side `s` referred to by `Side`. For now, we can simply infer `case`-predicates as case-likes as is, but we will see soon that when making decisions, that having generic cases will become useful.
2. We define generic magnitude factors through predicate `magnitude_factor_like(Case, Dimension, ThresholdValue)`. It represents a magnitude factor-like object belonging to a case-like identified by id `Case`. It is a magnitude factor-like for dimension `d` referred to by `Dimension` with threshold value `p` referred to by `ThresholdValue`. Again, for cases we can infer these predicates using `magnitude_factor`-predicates.

We specify that every case in the case base and every magnitude belonging to a case in the case base is considered case- and magnitude factor-like respectively:

```

1 case_like(Case, Side) :- case(Case, Side).
2
3 magnitude_factor_like(Case, Dimension, ThresholdValue) :-
4   magnitude_factor(Case, Dimension, ThresholdValue).
```

Listing 4.12: Generating case-likes

In the following definitions we will mainly use case-likes and magnitude factor-likes. However, for simplicity we will continue to refer to them as cases and magnitude factors.

As a utility to check for the presence of a magnitude factor in a case, we define the predicate

```
has_magnitude_factor_like(Case, Dimension).
```

It expresses that in a case with id `Case` a magnitude factor for dimension `d` referred to by `Dimension` is present. We specify the predicate as follows:

```

1 has_magnitude_factor_like(Case, Dimension) :-
2   magnitude_factor(Case, Dimension, ThresholdValue).
```

Listing 4.13: Utility to check presence of magnitude

Now, let us continue by examining what causes inconsistency. By Proposition 11, we can test whether the case base is inconsistent by finding two opposing cases $\langle X, r, s \rangle, \langle Y, r', \bar{s} \rangle \in \Gamma$ such that

1. $X \models \text{Premise}(r')$ or $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$, and
2. $Y \models \text{Premise}(r)$ or $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$.

If a case base is inconsistent, it will provide two witnessing cases that satisfy the given conditions. Since we are naturally interested in finding these contradictory cases, we will query inconsistency by trying to find witnesses matching the given proposition. If we cannot find any such witnesses, the case base is consistent.

We will first identify all opposing cases, since cases that are decided for the same side cannot cause inconsistency. For this, we introduce a utility predicate

```
opposing_case_likes(Case, Side, OppCase, OppSide).
```

It expresses that case `Case` decided for side `Side` and case `OppCase` decided for `OppSide` are opposing cases. We specify it as follows:

```
1 opposing_case_likes(Case, Side, OppCase, OppSide) :-
2   case_like(Case, Side),
3   case_like(OppCase, OppSide),
4   opposite_side(Side, OppSide),
5   Case != OppCase.
```

Listing 4.14: Identifying opposing case-likes

Every opposing pair of cases will be captured by this requirement. Take note of the check `Case != OppCase` that verifies that the case ids are different. Since every case id is unique in the case base, this check is redundant when simply performing a consistency check for the case base. However, we will check for decision constraints by testing decisions for both sides later on. In this scenario, it will occur that there are two `case_like`-predicates opposing each other that have the same id. Since we do not want to compare a case with itself, once favouring one side and once favouring the other side, we add the condition `Case != OppCase`.

Next, we need to model $X \models \text{Premise}(r')$ (equivalently $Y \models \text{Premise}(r)$), and $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$ (equivalently $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$). For this, we utilise the non-monotonicity of ASP. We cannot easily verify that $X \models \text{Premise}(r')$ in ASP, since we would need to check that every magnitude factor in $\text{Premise}(r')$ is satisfied by X . But we can do the opposite, i.e., $X \not\models \text{Premise}(r')$, by simply checking that there is one magnitude factor in $\text{Premise}(r')$ not satisfied by X . Through non-monotonicity, if we cannot find such a magnitude factor, using weak negation `not` we can infer that $X \models \text{Premise}(r')$ has to hold. The same idea will be applied for checking $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$.

Let us start with checking $X \not\models \text{Premise}(r')$. For this, we define the predicate

```
not_value_assignment_satisfies_opposing_magnitude_factor(Case, Side, OppCase,
                                                         OppSide, Dimension).
```

This predicate describes that the value assignment of case `Case` assigned for dimension `Dimension` does not satisfy the magnitude factor of opposing case `OppCase`, or equivalently $X \not\models \text{Premise}(r')$. To assert this statement, we have to pick value assignment $\langle d, q \rangle \in X$ and magnitude factor $M_{d,p}^{\bar{s}} \in \text{Premise}(r')$. If the magnitude factor is not present, it is implicitly satisfied, because the statement cannot be inferred if the magnitude factor cannot be selected. If it is present, we check if $p \not\prec_d^{\bar{s}} q$ holds. The condition $p \not\prec_d^{\bar{s}} q$ is equivalent to $q <_d^{\bar{s}} p$. We define the predicate as follows:

```
1 not_value_assignment_satisfies_opposing_magnitude_factor(Case, Side, OppCase,
2   OppSide, Dimension) :-
3   % pick two opposing cases
4   opposing_case_likes(Case, Side, OppCase, OppSide),
5   % pick a value assignment from the fact situation of the first case
6   value_assignment(Case, Dimension, Value),
7   % pick the matching magnitude factor from the reason of the second case
8   magnitude_factor_like(OppCase, Dimension, ThresholdValue),
9   % assert that the value assignment does not satisfy the magnitude factor
   strict_value_relation(OppSide, Dimension, Value, ThresholdValue).
```

Listing 4.15: Value assignment in case-like does not satisfy magnitude factor-like in opposing case-likes

Remember that `Value` is q and `ThresholdValue` is p . Thus, when using the predicate

```
strict_value_relation(OppSide, Dimension, Value, ThresholdValue)
```

referring to the strict value relation $<_d^{\bar{s}}$ we assert the fact that $q <_d^{\bar{s}} p$. This is equivalent $p \not\prec_d^{\bar{s}} q$, the condition we need to assert.

If one magnitude factor in $\text{Premise}(r')$ is not satisfied, the whole reason $\text{Premise}(r')$ cannot be satisfied either. Therefore, to propagate this knowledge of a violated magnitude factor to the whole reason we define predicate

```
not_fact_situation_satisfies_opposing_reason(Case, Side, OppCase, Side).
```

It expresses that the fact situation of case `Case` decided for side `Side` does not satisfy the reason of `OppCase` decided for side `OppSide`. We infer this predicate from the knowledge that at least one magnitude is not satisfied by the fact situation of case `Case`. Therefore, we specify it like so:

```
1 not_fact_situation_satisfies_opposing_reason(Case, Side, OppCase, OppSide) :-
2   not_value_assignment_satisfies_opposing_magnitude_factor(Case, Side,
3   OppCase, OppSide, Dimension).
```

Listing 4.16: Fact situation in case-like does not satisfy reason-like in opposing case-like

Similarly, we now verify that $\overline{\text{Premise}(r)} \not\models \text{Premise}(r')$. We first have to think in what scenarios this assertion can hold. Since reasons like $\overline{\text{Premise}(r)}$ and $\text{Premise}(r')$ do not require that all dimensions are present, there are four case distinctions we need to make for every dimension d :

1. Magnitude factors for d are present in $\overline{\text{Premise}(r)}$ and in $\text{Premise}(r')$: Then we need to compare the strength of the magnitude factors, ensuring that magnitude factor for d in $\overline{\text{Premise}(r)}$ is weaker for side \bar{s} than magnitude factor for d in $\text{Premise}(r')$.
2. A magnitude factor for d is present in $\text{Premise}(r')$, but not in $\overline{\text{Premise}(r)}$: Then we know that $\overline{\text{Premise}(r)}$ is not stronger than $\text{Premise}(r')$.
3. Magnitude factor for d is present in $\overline{\text{Premise}(r)}$, but not in $\text{Premise}(r')$: Then we can consider the magnitude factor for dimension d in $\overline{\text{Premise}(r)}$ stronger. It cannot be the cause for $\overline{\text{Premise}(r)} \not\models \text{Premise}(r')$ which is why we do not need to consider this any case further.
4. For dimension d a magnitude factor is present neither in $\overline{\text{Premise}(r)}$ nor in $\text{Premise}(r')$: Then dimension d cannot be responsible for $\overline{\text{Premise}(r)} \not\models \text{Premise}(r')$ and we do not need to consider this case any further.

Thus, only case distinctions 1 and 2 need to be considered for $\overline{\text{Premise}(r)} \not\models \text{Premise}(r')$, since 3 and 4 cannot cause the condition to be satisfied. Therefore, we can assume that a magnitude factor for d in $\text{Premise}(r')$ is present, but we distinguish the cases where a magnitude factor for d is either present in $\overline{\text{Premise}(r)}$ or not.

We first consider the case where the magnitude factor for d is not present in $\overline{\text{Premise}(r)}$. To model this, information we define the predicate

```
not_negated_magnitude_factor_present_for_opposing_magnitude_factor(Case, Side,
                                                                    OppCase, OppSide, Dimension).
```

This predicate expresses that there is no magnitude factor for dimension `Dimension` in case `Case` decided for side `Side`, but there is a magnitude factor for that dimension in `OppCase` decided for side `OppSide`. We specify the predicate as follows:

```
1 not_negated_magnitude_factor_present_for_opposing_magnitude_factor(Case, Side
2   , OppCase, OppSide, Dimension) :-
3   % pick two opposing cases
4   opposing_case_likes(Case, Side, OppCase, OppSide),
5   % ensure that the magnitude factor from the second case is present
6   has_magnitude_factor_like(OppCase, Dimension),
7   % ensure that the magnitude factor from the first case is not present
8   not has_magnitude_factor_like(Case, Dimension).
```

Listing 4.17: Negated magnitude factor-like in case-like does not exist while magnitude factor-like in opposing case-like does

Note that in this scenario it does not matter if we consider the negated reason $\overline{\text{Premise}(r)}$ or $\text{Premise}(r)$ since the magnitude factor for d is missing in both.

For the second case distinction where a magnitude factor for d is present in both $\overline{\text{Premise}(r)}$ and $\text{Premise}(r')$, we define the predicate

```
not_negated_magnitude_factor_satisfies_opposing_magnitude_factor(Case, Side,
                                                                    OppCase, OppSide, Dimension).
```

This predicate expresses that negated magnitude factor $M_{d,p}^{\bar{s}} \in \overline{\text{Premise}(r)}$ for dimension d given by dimension `Dimension` in case `Case` is not stronger for side \bar{s} than the magnitude factor in $M_{d,q}^{\bar{s}} \in \text{Premise}(r')$. The condition for $\overline{\text{Premise}(r)} \not\models \text{Premise}(r')$ therefore is that $p \not\leq_{\bar{s}}^d q$. This is equivalent to $q <_{\bar{s}}^d p$. We now specify the predicate as follows:

```
1 not_negated_magnitude_factor_satisfies_opposing_magnitude_factor(Case, Side,
2   OppCase, OppSide, Dimension) :-
3   % pick two opposing cases
4   opposing_case_likes(Case, Side, OppCase, OppSide),
5   % pick a magnitude factor from the reason of the first case
6   magnitude_factor_like(Case, Dimension, ThresholdValue1),
7   % pick the matching magnitude factor from the reason of the second case
8   magnitude_factor_like(OppCase, Dimension, ThresholdValue2),
9   % check that the magnitude factor in the first case is weaker
10  % than the magnitude factor in the second case
    strict_value_relation(OppSide, Dimension, ThresholdValue1,
                          ThresholdValue2).
```

Listing 4.18: Negated magnitude factor-like in case-like does not satisfy magnitude factor-like in opposing case-like

Note that we did not perform any explicit negation of $\text{Premise}(r)$ in this rule. Since the `magnitude_factor_like`-predicates do not have a side attached, they can be considered independent of a side. Because the satisfaction of a magnitude factor is given by the value relation, picking the opposite of the side that case `Case` is decided for negates the magnitude factor implicitly. Remember that `ThresholdValue1` is p and `ThresholdValue2` is q . Thus

```
strict_value_relation(OppSide, Dimension, ThresholdValue1, ThresholdValue2)
```

asserts $q <_{\bar{s}}^d p$. This is equivalent to $p \not\leq_{\bar{s}}^d q$ which is sufficient to show that $\overline{\text{Premise}(r)} \not\models \text{Premise}(r')$.

If one magnitude factor is not weaker or missing in $\overline{\text{Premise}(r)}$, the whole reason $\overline{\text{Premise}(r)}$ cannot entail $\text{Premise}(r')$. To propagate this information to the whole reason, we define predicate

```
not_negated_reason_satisfies_opposing_reason(Case, Side, OppCase, OppSide).
```

It expresses that the negated reason of case `Case` decided for side `Side` does not entail the reason of case `OppCase` decided for side `OppSide`. We specify the predicate by

distinguishing the two cases previously described, that is, either there is a magnitude factor for dimension d in $\text{Premise}(r')$ that is not in $\overline{\text{Premise}(r)}$ or a magnitude factor for dimension d in $\overline{\text{Premise}(r)}$ is weaker for side \bar{s} than the corresponding magnitude factor in $\text{Premise}(r')$:

```

1 not_negated_reason_satisfies_opposing_reason(Case, Side, OppCase, OppSide) :-
2   not_negated_magnitude_factor_present_for_opposing_magnitude_factor(Case,
3     Side, OppCase, OppSide, Dimension).
4 not_negated_reason_satisfies_opposing_reason(Case, Side, OppCase, OppSide) :-
5   not_negated_magnitude_factor_satisfies_opposing_magnitude_factor(Case,
6     Side, OppCase, OppSide, Dimension).
```

Listing 4.19: Negated reason-like in case-like does not satisfy reason-like in opposing case-like

We can now use these definitions to check for inconsistency. For a case base to be inconsistent, we require $X \models \text{Premise}(r')$ or $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$, and $Y \models \text{Premise}(r)$ or $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$ for two opposing cases $c, c' \in \Gamma$ where $c = \langle X, r, s \rangle$ and $c' = \langle Y, r', \bar{s} \rangle$.

Since the property is disjunctive, but ASP rules do not allow disjunction in the rule body, we will introduce a utility predicate. Before we define it, an intuition on how we model disjunction: Suppose we want predicate c to be true, if either a or b is true. To model this, we can define

$$c :- a. \quad c :- b.$$

In the same way, we can model that $X \models \text{Premise}(r')$ or $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$. We will define an intermediate predicate that captures this condition between the two cases:

```
not_consistent_cross_condition(Case, Side, OppCase, OppSide).
```

It expresses that either $X \models \text{Premise}(r')$ or $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$ holds. Note that with the predicates defined, we currently have knowledge of the negation $X \not\models \text{Premise}(r')$ and/or $\overline{\text{Premise}(r)} \not\Vdash \text{Premise}(r')$. To negate this information in the program, we use weak negation, i.e., **not**-statements. We define the cross condition like so:

```

1 not_consistent_cross_condition(Case, Side, OppCase, OppSide) :-
2   opposing_case_likes(Case, Side, OppCase, OppSide),
3   not not_negated_reason_satisfies_opposing_reason(Case, Side, OppCase,
4     OppSide).
5 not_consistent_cross_condition(Case, Side, OppCase, OppSide) :-
6   opposing_case_likes(Case, Side, OppCase, OppSide),
7   not not_fact_situation_satisfy_opposing_reason(Case, Side, OppCase,
8     OppSide).
```

Listing 4.20: Intermediate inconsistency between case-likes

Finally, we define the predicate

```
not_consistent(Case, Side, OppCase, OppSide)
```

which states that the two opposing cases `Case` decided for side `Side`, and `OppCase` decided for side `OppSide` are inconsistent. Per Proposition 11, $X \models \text{Premise}(r')$ or $\text{Premise}(r) \Vdash \text{Premise}(r')$, and $Y \models \text{Premise}(r)$ or $\text{Premise}(r') \Vdash \text{Premise}(r)$ has to hold for the cases to be inconsistent. Note that the two disjunctive statements are symmetrical but with the variables flipped. Thus, in the program this condition is satisfied if the cross condition defined holds symmetrically. We specify the predicate as follows:

```
1 not_consistent(Case, Side, OppCase, OppSide) :-
2   not_consistent_cross_condition(Case, Side, OppCase, OppSide),
3   not_consistent_cross_condition(OppCase, OppSide, Case, Side).
```

Listing 4.21: Mark case-likes inconsistent

Remember that we defined case-like and magnitude factor-like predicates earlier and used these to check for inconsistency. Therefore, the `not_consistent`-predicate actually performs the inconsistency check for case-likes instead of just cases in the case base. However, when asserting the consistency of the case base, we are only interested in verifying that cases in the case base are consistent, and not decisions that are only tested for consistency against the case base. Therefore, we define the predicate

```
not_cases_consistent(Case, Side, OppCase, OppSide).
```

It expresses that two opposing cases `Case` decided for side `Side`, and `OppCase` decided for `OppSide` are inconsistent. We specify the predicate such that it limits the inconsistency check only to cases:

```
1 not_cases_consistent(Case, Side, OppCase, OppSide) :-
2   case(Case, Side),
3   case(OppCase, OppSide),
4   not_consistent(Case, Side, OppCase, OppSide).
```

Listing 4.22: Mark cases inconsistent

Finally, if we find two inconsistent cases, we can also mark the whole case base as inconsistent. For this, we define predicate

```
not_case_base_consistent
```

which expresses that the case base is inconsistent, if it is present. If the predicate is absent, the case base is consistent. We specify the predicate as follows by inferring it from the knowledge of inconsistent cases in the case base:

```
1 not_case_base_consistent :-
2   not_cases_consistent(Case, Side, OppCase, OppSide).
```

Listing 4.23: Mark case base inconsistent

Thus, if predicate `not_case_base_consistent` is derived, we know that there are some cases that are inconsistent. Because of the previously-defined predicates, we can even pinpoint why inconsistencies happen on case-level. Cases that are consistent will have witnessing magnitude factors that prove their consistency whereas cases that are inconsistent will lack these witnesses.

4.5.2 Decisions

We will extend the existing definitions to enable decision-making using ASP. To test if a decision to some fact situation X in the dimensional reason model is possible, by Proposition 12 we need to verify that for some consistent case base Γ we can add case $\langle X, U_X^s \rightarrow s, s \rangle$ such that $\Gamma \cup \{\langle X, U_X^s \rightarrow s, s \rangle\}$ is also consistent. Firstly, we define the predicate

```
decision(Case, Side).
```

This predicate does not state that a decision for `Case` in favour of `Side` was made, merely that a decision is *proposed*. So the predicate helps the program keep track of all the decisions we want to test. That is, for every new fact situations we want to test two decisions in favour of the two sides π and δ , i.e., plaintiff and defendant. Given that the case base is consistent, we infer the predicate for each side:

```
1 decision(Case, Side) :-
2   unsolved_case(Case),
3   side(Side),
4   not not_case_base_consistent.
```

Listing 4.24: Generating case decision proposals

This infers two case proposals for each `unsolved_case`-predicate, one in favour of the plaintiff, and one in favour of the defendant. Recall that `unsolved_case`-predicates represent new fact situations that need to be decided by a model.

We will now utilise the definitions of case-likes and magnitude factor-likes from the consistency check, since the decision process relies on consistency. Since we want to verify that $\langle X, U_X^s \rightarrow s, s \rangle$ is consistent with the existing case base Γ , we can specify a case-like with winning side s and magnitude factor-likes matching the value assignments present in the fact situation:

```
1 case_like(Case, Side) :- decision(Case, Side).
2
3 magnitude_factor_like(Case, Dimension, Value) :-
4   decision(Case, Side),
5   value_assignment(Case, Dimension, Value).
```

Listing 4.25: Generating case-likes and magnitude-likes for decisions

The translation of the decision is straightforward. Note that the magnitude factor-likes do not have any attribute stating which side they favour. In our program, that information is

inferred from the `case_like`-predicate. Since U_X^s and $U_X^{\bar{s}}$ contain exactly the same values, but simply favour a different side, we can define `magnitude_factor_like`-predicates once for both sides. Because the information regarding which side is favoured comes from the `decision`- and thus `case_like`-predicate, the `magnitude_factor_like`-predicates are therefore assigned to both `case_like`-predicates defined for each case `Case`, i.e., for `Side = plaintiff` and for `Side = defendant`.

Since we define decisions as case-likes, we can now utilise the `not_consistent`-predicate to check whether the decision is inconsistent with some other case-like. We define

```
not_decision_consistent_with_case(Case, Side, OppCase, OppSide).
```

This predicate expresses that a decision on case `Case` in favour of side `Side` is inconsistent with existing case in the case base `OppCase` decided for side `OppSide`. We specify this condition like so:

```
1 not_decision_consistent_with_case(Case, Side, OppCase, OppSide) :-
2   decision(Case, Side),
3   case(OppCase, OppSide),
4   not_consistent(Case, Side, OppCase, OppSide).
```

Listing 4.26: Mark decision inconsistent with case

To reiterate, the `case_like`- and `magnitude_factor_like`-predicates causes the decisions to be compared with all other case-likes. If several `unsolved_case`-predicates are defined, then `not_consistent` also compares these. Thus, `not_consistent` compares any two opposing abstract case-likes defined. For consistency, we specified `not_cases_consistent` on top of `not_consistent` by requiring that the opposing case-likes are cases in the case base. Now, for `not_decision_consistent_with_case` we specify that the first case-like is a decision and the second case-like is a case in the case base.

If there is some case in the case base that is inconsistent with a proposed case decision, then we know that the proposed decision is not permitted. To model the permission of a decision we define predicate

```
not_decision_permmissible(Case, Side)
```

which states that a decision for unsolved case `Case` is not permitted in favour of side `Side`. We specify the predicate like so:

```
1 not_decision_permmissible(Case, Side) :-
2   not_decision_consistent_with_case(Case, Side, OppCase, OppSide).
```

Listing 4.27: Check if a decision is not permitted

Thus, if `not_decision_permmissible(Case, Side)` is present for unsolved case, then we know that `Side` cannot be selected. From this, we can conclude that a decision in favour of the opposite side is required. If the `not_decision_permmissible`-predicate cannot be inferred for either of the two sides, then the decision is not constrained.

4.5.3 Rule Selection

The reason model requires that for a decision a rule has to be selected. To test whether a decision is permitted, we select rule $U_X^s \rightarrow s$, because by Proposition 17 it is sufficient when testing for decision permission. However, we can also test if a certain rule can be selected simply by treating the unsolved case as part of the case base. Since the program can verify the consistency of case base Γ , it can also verify the consistency of case base $\Gamma \cup \{\langle X, r, s \rangle\}$, where r is the proposed rule for fact situation X . This is compliant with Definition 18, which states exactly that a rule r supporting s can only be selected for X if and only if $\Gamma \cup \{\langle X, r, s \rangle\}$ is consistent. Alternatively, if users of the program prefer distinct predicates for checking if a certain rule can be selected, the case-like and magnitude factor-like definitions could be reused in the same manner as in Subsection 4.5.2.

4.6 The Dimensional Reduction Model

We will implement the dimensional reduction model, as introduced in Chapter 3. While the following section describes a program different from the program described in Section 4.5, there is a lot of overlap in what predicates are defined. Again, we want to check (a) if a case base is consistent, and (b) what decisions can be made for new fact situations, i.e., unsolved cases, against the case base. Since this model performs a translation of the case base to the dimensional result model, and then uses this model as the basis for reasoning, we will start with a translation of the case base.

4.6.1 Translation

The translation step in the reduction model turns every case $\langle X, r, s \rangle \in \Gamma$ into a case $\langle X^r, s \rangle \in \Gamma^\times$. That is, the fact situation X and the rule r are combined to generate a reduced fact situation

$$X^r = \{\langle d, p \rangle \mid M_{d,p}^s \in \text{Premise}(r)\} \cup \{\langle d, q \rangle \in X \mid \forall M_{d',p}^s \in \text{Premise}(r) : d' \neq d\}.$$

This definition expresses that for each dimension $d \in D$, if there is a magnitude factor $M_{d,p}^s \in \text{Premise}(r)$, then pick p as the new value for the value assignment on d . If there is no magnitude factor for dimension d in $\text{Premise}(r)$, then pick $X(d)$ as the new value for the value assignment on d . In the following section we will perform the model translation from Γ to Γ^\times .

First, we define an auxiliary predicate

```
has_magnitude_factor(Case, Dimension).
```

The predicate asserts that a magnitude factor for dimension `Dimension` is present in case `Case`. We specify it like so:

```

1 has_magnitude_factor(Case, Dimension) :-
2   magnitude_factor(Case, Dimension, ThresholdValue).

```

Listing 4.28: Auxiliary predicate to check for existence of a magnitude factor

Next, we define a new predicate for value assignments, since we cannot overwrite the `value_assignment`-predicates. For every value assignment in a case we define the predicate

```
reduced_value_assignment(Case, Dimension, Value).
```

A reduced value assignment should represent a value assignment in the reduced fact situation that is the result of the model translation. If case `Case` has a magnitude factor for dimension `Dimension`, then `Value` is assigned the value of that magnitude factor. Otherwise, the value of the value assignment for `Dimension` is chosen. We specify the translation as follows:

```

1 % use magnitude factors if present
2 reduced_value_assignment(Case, Dimension, Value) :-
3   magnitude_factor(Case, Dimension, Value).
4
5 % use value assignments otherwise
6 reduced_value_assignment(Case, Dimension, Value) :-
7   case(Case, Side),
8   value_assignment(Case, Dimension, Value),
9   not has_magnitude_factor(Case, Dimension).

```

Listing 4.29: Constructing reduced value assignments

The resulting program defines `reduced_value_assignment`-predicates for every existing case in the case base. Since the reduction model uses only these reduced value assignments, the `reduced_value_assignment`-predicates will act as the basis for checking consistency between cases. Do note that the `value_assignment`-predicates for unsolved cases where *not* translated.

4.6.2 Consistency

In the same manner as in the dimensional reason model we define `case_like`-predicates to be used for consistency instead of `case`-predicates directly, to again be able to reuse the consistency check for the decision-making process. Instead of magnitude factor-likes, we now introduce reduced value assignment-likes by defining

```
reduced_value_assignment_like(Case, Dimension, Value).
```

It holds exactly the same information as the `reduced_value_assignment`-predicates where `Case` refers to the case, `Dimension` refers to the chosen dimension, and `Value` refers to the value of the value assignment. For cases, the `case_like`-predicates and `reduced_value_assignment_like`-predicates are simply specified like so:

```

1 case_like(Case, Side) :- case(Case, Side).
2
3 reduced_value_assignment_like(Case, Dimension, Value) :-
4   reduced_value_assignment(Case, Dimension, Value).

```

Listing 4.30: Generating case-likes and reduced value assignment-likes

Like before, for simplicity we will continue to speak of cases instead of case-likes when modelling consistency. Consistency in the result model is asserted according to Definition 7, that is, a case base Γ^\times is inconsistent if and only if there are two opposing cases $\langle X, s \rangle, \langle Y, \bar{s} \rangle \in \Gamma^\times$ such that

$$X \leq^s Y.$$

In a similar manner as in the reason model, we will first assert the opposite $X \not\leq^s Y$ and then use default negation `not` to assert $X \leq^s Y$. The condition $X \not\leq^s Y$ is satisfied if and only if there is some dimension d such that $X(d) \not\leq_d^s Y(d)$. This statement is equivalent to $Y(d) <_d^s X(d)$. That is, if we can find a dimension for which X is stronger in favour of s than Y , then $X \not\leq^s Y$ holds.

Recall that only opposing cases, i.e., $\langle X, s \rangle$ and $\langle Y, \bar{s} \rangle$, can be inconsistent. Therefore, we again need to identify these cases. Since we need to identify opposing cases several times throughout the program, we define auxiliary predicate

```
opposing_case_likes(Case, Side, OppCase, OppSide).
```

in the same way as before. That is, case `Case` decided in favour of side `Side` is an opposing case of case `OppCase` decided in favour of side `OppSide`. We specify the predicate as follows:

```

1 opposing_case_likes(Case, Side, OppCase, OppSide) :-
2   case_like(Case, Side),
3   case_like(OppCase, OppSide),
4   opposite_side(Side, OppSide),
5   Case != OppCase.

```

Listing 4.31: Identifying opposing case-likes

To model the condition that $X(d) \not\leq_d^s Y(d)$, we now define predicate

```
not_value_assignment_leq_opposing_value_assignment(Case, Side, OppCase,
                                                    OppSide, Dimension).
```

It states that for two opposing cases `Case` and `OppCase`, value assignment on dimension `Dimension` belonging to case `OppCase` is not stronger than or equally strong for side `Side` as value assignment on dimension `Dimension` belonging to case `Case`. If X belongs to `Case`, s refers to side `Side`, Y belongs to `OppCase`, and d refers to dimension `Dimension`, then the predicate models $X(d) \not\leq_d^s Y(d)$ which is equivalent to $Y(d) <_d^s X(d)$. We specify it as follows:

```

1 not_value_assignment_leq_opposing_value_assignment(Case, Side, OppCase,
  OppSide, Dimension) :-
2   % pick two opposing cases
3   opposing_case_likes(Case, Side, OppCase, OppSide),
4   % pick the reduced value assignment from the first case
5   reduced_value_assignment_like(Case, Dimension, Value1),
6   % pick the matching reduced value assignment from the second case
7   reduced_value_assignment_like(OppCase, Dimension, Value2),
8   % check if the reduced value assignment from the first case is stronger
9   strict_value_relation(Side, Dimension, Value2, Value1).

```

Listing 4.32: Value assignment in opposing case-like not stronger than or equal to value assignment in case-like

Next, we want to model the statement $X \not\leq^s Y$. For this, we define

```
not_fact_situation_leq_opposing_fact_situation(Case, Side, OppCase, OppSide).
```

It states that $X \not\leq^s Y$ where X belongs to case $Case$, s refers to side $Side$, and Y belongs to case $OppCase$. Since $X(d) \not\leq_d^s Y(d)$ for some dimension d implies $X \not\leq^s Y$, we can simply specify the predicate as follows:

```

1 not_fact_situation_leq_opposing_fact_situation(Case, Side, OppCase, OppSide)
  :-
2   not_value_assignment_leq_opposing_value_assignment(Case, Side, OppCase,
  OppSide, Dimension).

```

Listing 4.33: Fact situation in opposing case-like not stronger than or equal to fact situation in case-like

To model that two cases are inconsistent, we need to assert that $X \leq^s Y$. Since we have knowledge of $X \not\leq^s Y$ from the previous predicate, we can use default negation to infer knowledge of inconsistent cases. We reuse the predicate

```
not_consistent(Case, Side, OppCase, OppCase)
```

which states that case $Case$ decided for side $Side$ is inconsistent with opposing case $OppCase$ decided for side $OppSide$. We specify it like so:

```

1 not_consistent(Case, Side, OppCase, OppSide) :-
2   opposing_case_likes(Case, Side, OppCase, OppSide),
3   not not_fact_situation_leq_opposing_fact_situation(Case, Side, OppCase,
  OppSide).

```

Listing 4.34: Mark case-likes inconsistent

Note that the `not_consistent`-predicate actually applies to all case-likes. Since we again use case-likes for checking inconsistency and decision permissions alike, we filter out cases by defining

```
not_cases_consistent(Case, Side, OppCase, OppSide)
```

which checks for inconsistency, given that the two opposing case-likes are cases. We specify the predicate like so:

```
1 not_cases_consistent(Case, Side, OppCase, OppSide) :-
2   case(Case, Side),
3   case(OppCase, OppSide),
4   not_consistent(Case, Side, OppCase, OppSide).
```

Listing 4.35: Mark cases inconsistent

And lastly, just like in the reason model, we also define the predicate

```
not_case_base_consistent
```

which tells us if the case base is inconsistent. That is, if the `not_cases_consistent`-predicate is present in the answer set, then the case base is inconsistent. If the predicate is absent, then the case base is consistent. We specify it in the same way as before:

```
1 not_case_base_consistent :-
2   not_cases_consistent(Case, Side, OppCase, OppSide).
```

Listing 4.36: Mark case base inconsistent

4.6.3 Decisions

Analogous to the implementation of the reason model, the program can propose decisions for unsolved cases and then check their consistency. Given fact situation X and case base Γ , by Proposition 17 we can verify that a decision in favour of s is permitted if and only if $\Gamma^{\times} \cup \{\langle X, s \rangle\}$ is consistent. Since the decision permission is a check of consistency, analogous to the implementation of the reason model we will utilise the `not_consistent`-predicate to check for decision permissions as well. First let us again define the predicate

```
decision(Case, Side)
```

and specify it in the same manner as before. That is, the `decision`-predicate refers to a decision proposal and is only inferred for both sides, if the case base is consistent:

```
1 decision(Case, Side) :-
2   unsolved_case(Case),
3   side(Side),
4   not not_case_base_consistent.
```

Listing 4.37: Generating case decision proposals

To check for consistency, we reuse the `not_consistent`-predicate. The predicate in the reduction model requires us to define case-likes and reduced value assignment-likes. The case case-likes can be inferred directly from the `decision`-predicates. Since by Proposition 17 we do not need to perform a translation for fact situation X to

check if a decision is permissible, we can directly infer all `value_assignment`-predicates as `reduced_value_assignment_like`-predicates. We specify case- and reduced value assignment-likes as follows:

```

1 case_like(Case, Side) :- decision(Case, Side).
2
3 reduced_value_assignment_like(Case, Dimension, Value) :-
4     decision(Case, Side),
5     value_assignment(Case, Dimension, Value).
```

Listing 4.38: Generating case-likes from decisions

From here on, the program is identical with the specifications in the reason model. We say that a decision is consistent by reusing predicate

```
not_decision_consistent_with_case(Case, Side, OppCase, OppSide)
```

which states that decision for unsolved case `Case` in favour of side `Side` is inconsistent with case in the case base `OppCase` decided for side `OppSide`. We specify the predicate as follows:

```

1 not_decision_consistent_with_case(Case, Side, OppCase, OppSide) :-
2     decision(Case, Side),
3     case(OppCase, OppSide),
4     not_consistent(Case, Side, OppCase, OppSide).
```

Listing 4.39: Mark decision inconsistent with case

Further, we also reuse the predicate

```
not_decision_permissible(Case, Side)
```

which states that a decision for unsolved case `Case` in favour of side `Side` is not permitted. We specify it like so:

```

1 not_decision_permissible(Case, Side) :-
2     not_decision_consistent_with_case(Case, Side, OppCase, OppSide).
```

Listing 4.40: Check if a decision is not permitted

Therefore, if for an unsolved case a decision in favour of a side is not permitted, that is, some case in the case base is inconsistent with it, a `not_decision_permissible`-predicate is inferred. If the predicate is present, then there has to be some case `not_decision_consistent_with_case`-predicate present as well that causes the inconsistency. If a decision is not permitted for a side s that also means that it is constrained to side \bar{s} .

4.6.4 Rule Selection

Testing if a certain rule can be selected for a fact situation works in the same manner as in the implementation of the reason model, described in Subsection 4.5.3.

Case Study

Models of precedential constraint rely on past examples to guide future reasoning. In the previous chapters, we have discussed small-scale examples that were meant to aid the understanding of the various models introduced. However, to properly compare models of precedential constraint, we are interested in a larger dataset. In the following chapter we will generate and analyse a dataset aimed at the dimensional reason model. This dataset will not only allow us to examine how the dimensional reason model could work in practice, but will also enable us to further compare it with the dimensional reduction model we introduced in Chapter 3.

In Section 5.1 we will outline how the dataset is generated. In Section 5.2 we will evaluate the pair-wise consistency of cases in the reason model and in the reduction model. Further, we will check the decision constraints of both models. In Section 5.3 we will then analyse the results of the evaluation by comparing for which instances the models have diverging conclusions.

5.1 Generating a Dataset

First, we have to address the question of how we can retrieve a dataset that fits our needs. For the dimensional reason model, in particular, we require a set of cases where each case contains a fact situation, a binary decision, and justification for the decision. Further, value assignments and magnitude factors for each dimension d have to satisfy the total orderings \leq_d^s and $\leq_d^{\bar{s}}$. Unfortunately, finding a fully-featured dataset that matches all the criteria we require for the model turns out to be difficult. This is why instead we opt for generating a synthetic dataset. While the downside of generating a synthetic dataset is that it is potentially too far removed from a real-world example, the upside is that it enables us to continue our running example of the nanny robot as introduced in previous chapters.

To reiterate, in our hypothetical scenario, a nanny robot is tasked with making a binary decision each day. It has to decide whether a child needs to be sent to bed on time, or whether the bedtime can be extended, depending on the child’s behaviour. This decision should be made in accord with the parents’ preferences. In a real-world scenario retrieving the parents’ preferences could happen in the form of a questionnaire that asks the parents to fill out examples of how decisions should be made. These decisions then act as the case base for the nanny robot. It is the data the robot bases its decisions on.

We will now generate a synthetic dataset. In fact, the dataset is only semi-synthetic, since we define the five dimensions used by hand. The dimensions are the same as in the running example of the previous chapters. The dimension set is

$$D_{\text{study}} = \{d_{\text{protest}}, d_{\text{screen}}, d_{\text{activity}}, d_{\text{coop}}, d_{\text{read}}\}.$$

The contained dimensions are defined in the same way as before.

We consider 20 solved cases—10 in favour of the plaintiff and 10 in favour of the defendant—to be an adequate size for generating a dataset. Apart from this, we do not have any requirement regarding consistency. In fact, we aim for some level of inconsistency to not only compare the constraints of the models, but further to test if the implementation equally detects the inconsistencies.

We generate the case study by writing a simple script in Python 3. The exact script can be found in the appendix, but we will briefly outline here how it progresses:

1. For each dimension $d \in D_{\text{study}}$, we define a *finite* range of allowed values from side \bar{s} to the opposite side s . This range is a subset of $[d]$.
2. For each case, randomly pick values for the fact situation in the allowed range for every dimension $d \in D_{\text{study}}$.
3. Calculate a probability that, based on the present value assignments makes it more likely for the plaintiff or for the defendant to win. This probability is then taken to choose a winning side.
4. Having decided on a winning side, randomly pick magnitude factors. The values for the magnitude factors are also chosen randomly with values that are stronger for the winning side being more likely to be chosen.

The result of this random generation of cases can be viewed in Table 5.1. Here, cases c_1, c_2, \dots, c_{10} were decided in favour of the plaintiff whereas cases $c_{11}, c_{12}, \dots, c_{20}$ were decided in favour of the defendant. Every case has the winning side, and the value of each value assignment in the fact situation given in a column. Furthermore, for every magnitude factor in the reason there is a column as well. If the column is left empty, the dimension was not chosen for a magnitude factor. If the column contains a value, it

is the threshold value for the magnitude factor supporting the winning side s . So, for example, case c_1 contains fact situation X_1 and rule r_1 where

$$X_1 = \{\langle d_{\text{protest}}, \text{screamed} \rangle, \langle d_{\text{screen}}, 105 \rangle, \langle d_{\text{activity}}, 285 \rangle, \langle d_{\text{coop}}, \text{physical} \rangle, \langle d_{\text{read}}, 30 \rangle\}, \text{ and}$$

$$r_1 = \{M_{d_{\text{protest}}, \text{cried}}^\pi, M_{d_{\text{coop}}, \text{physical}}^\pi, M_{d_{\text{read}}, 30}^\pi\} \rightarrow \pi.$$

case	side	dimension values					magnitude factor values for winning side s				
		d_{protest}	d_{screen}	d_{activity}	d_{coop}	d_{read}	d_{protest}	d_{screen}	d_{activity}	d_{coop}	d_{read}
c_1	π	screamed	105	285	physical	30	cried			physical	30
c_2	π	screamed	90	75	ignored	10	screamed	90	90	played	20
c_3	π	kicked	135	255	played	20	kicked			played	80
c_4	π	cried	165	285	played	25		150			80
c_5	π	cried	75	285	physical	45				verbal	50
c_6	π	complained	120	30	verbal	60		90	60	ignored	
c_7	π	kicked	45	90	played	15	screamed		120		90
c_8	π	complained	255	105	played	55		150	180		90
c_9	π	cried	210	15	played	100		150	30		
c_{10}	π	kicked	15	150	physical	95	screamed		240	physical	
c_{11}	δ	screamed	210	90	ignored	95			60	20	
c_{12}	δ	kicked	150	60	verbal	35		270			10
c_{13}	δ	screamed	210	150	verbal	85					60
c_{14}	δ	none	255	75	played	5	none			played	
c_{15}	δ	complained	60	150	played	25	cried	60		ignored	
c_{16}	δ	kicked	180	30	physical	85					70
c_{17}	δ	none	195	75	verbal	0	cried				
c_{18}	δ	none	165	285	verbal	100	none	180	90		90
c_{19}	δ	kicked	180	60	verbal	50		240			30
c_{20}	δ	screamed	240	15	ignored	100					10

Table 5.1: Case study dataset

5.2 Model Comparison on the Dataset

In this section, we examine the dataset in the dimensional reason model and dimensional reduction model. First, we determine the consistency of the case base. In particular, this involves examining all opposing cases and checking if an inconsistency exists between these. In addition to consistency, we also examine how decision constraints are enforced in both datasets. For this, we pick a subset of cases that are consistent with each other for the case base. We can reuse the fact situations of the remaining cases by treating these as “new” fact situations that need to be checked for decision constraints.

Consistency To verify the consistency of the dataset, we can consider all pairs of opposing cases. Since we have 10 cases in favour of the plaintiff and 10 cases in favour of the defendant, we can model consistency across cases in a 10 by 10 table. This means that we need to compare 100 cases in total. We will start by evaluating consistency in the reason model. To reiterate, by Proposition 11 two cases $\langle X, r, s \rangle, \langle X, r', \bar{s} \rangle \in \Gamma_{\text{study}}$ are inconsistent if and only if

1. $X \models \text{Premise}(r')$ or $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$, and

2. $Y \models \text{Premise}(r)$ or $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$.

Conversely, the two cases are consistent if and only if

1. $X \not\models \text{Premise}(r')$ and $\overline{\text{Premise}(r)} \not\Vdash \text{Premise}(r')$, or
2. $Y \not\models \text{Premise}(r)$ and $\overline{\text{Premise}(r')} \not\Vdash \text{Premise}(r)$.

Thus, w.l.o.g. we simply need to find a witnessing dimension d that causes $X \not\models \text{Premise}(r')$, and a witnessing dimension d' that causes $\overline{\text{Premise}(r)} \not\Vdash \text{Premise}(r')$. For the latter this is achieved either through absence of that dimension in $\overline{\text{Premise}(r)}$, or because of it being stronger for side \bar{s} in $\text{Premise}(r')$.

A manual evaluation of the case base in the reason model yields the results given in Table 5.2. We performed 100 comparisons of opposing cases, that is, all cases decided in favour the plaintiff c_1, c_2, \dots, c_{10} were compared with all cases decided in favour of the defendant $c_{11}, c_{12}, \dots, c_{20}$. Out of the 100 comparisons, there are 20 pairs of cases that are inconsistent, leaving a rest of 80 pairs of cases that are consistent.

	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
c_1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
c_2	✓		✓	✓	✓	✓	✓	✓	✓	✓
c_3	✓		✓	✓	✓	✓	✓	✓	✓	✓
c_4	✓		✓	✓	✓	✓		✓		✓
c_5	✓		✓	✓	✓	✓		✓		✓
c_6	✓		✓	✓	✓	✓	✓	✓		
c_7	✓		✓	✓	✓		✓	✓	✓	✓
c_8	✓			✓	✓			✓		✓
c_9	✓	✓	✓	✓	✓		✓	✓	✓	
c_{10}	✓	✓	✓	✓	✓		✓	✓	✓	✓

Table 5.2: Reason model consistency for case study

We now perform the consistency check for the reduction model. First, we translate all cases in Γ_{study} to cases in $\Gamma_{\text{study}}^{\times}$. Every case $\langle X, r, s \rangle \in \Gamma_{\text{study}}$ is turned into a case $\langle X^r, s \rangle \in \Gamma_{\text{study}}^{\times}$. Two opposing cases $\langle X, s \rangle, \langle Y, \bar{s} \rangle \in \Gamma_{\text{study}}^{\times}$ are inconsistent if and only if

$$X \leq^s Y.$$

Conversely, the two cases are consistent if and only if

$$X \not\leq^s Y.$$

We can check this condition by finding a witness dimension d such that $X(d) \not\leq_d^s Y(d)$ which is equivalent to $Y(d) <_d^s X(d)$. We need to perform the evaluation only for the 80

case pairs that are consistent in the reason model, because by Theorem 3 any two cases that are inconsistent in the reason model have to be inconsistent in the reduction model as well.

A manual evaluation of the case base in reduction model yields the results given in Table 5.3. We performed 80 comparisons of opposing cases, and marked the ones red that are consistent in the reason model, but inconsistent in the reduction model. Out of the 80 comparisons, there are 15 more pairs of cases that are inconsistent. Thus, a rest of 65 pairs of cases is consistent and a total of 35 pairs is now inconsistent.

	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
c_1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
c_2			✓	✓	✓	✓	✓	✓	✓	
c_3	✓		✓	✓	✓		✓	✓		✓
c_4				✓	✓			✓		
c_5	✓		✓	✓	✓	✓		✓		✓
c_6			✓	✓	✓	✓	✓			
c_7			✓	✓	✓		✓	✓		
c_8				✓	✓			✓		
c_9	✓	✓	✓	✓	✓		✓	✓	✓	
c_{10}	✓	✓	✓	✓	✓		✓	✓	✓	✓

Table 5.3: Reduction model consistency for case study

Constraint In addition to consistency in the models, we also want to compare the decision constraints imposed by the models. To test what decision constraints are present in the case study, we have to test a fact situation against a consistent case base. We will pick cases from the case study that are consistent with each other for a case base, and treat the fact situations of the remaining cases as the “new” situations to be checked, if decision constraints apply to them. Since the case base Γ_{study} of all 20 cases is inconsistent for both models, we pick a subset of 8 cases that are consistent for the reduction model. If the case base is consistent in the reduction model, it is consistent in the reason model as well. This allows us to compare the decision constraints this case base enforces. We pick 8 cases for our case base

$$\Gamma_{\text{study:consistent}} = \{c_1, c_3, c_5, c_{10}, c_{13}, c_{14}, c_{18}, c_{20}\}.$$

4 cases in $\Gamma_{\text{study:consistent}}$ are decided in favour of the plaintiff and 4 cases in $\Gamma_{\text{study:consistent}}$ are decided in favour of the defendant. We can easily see that the case base is consistent by comparing all the opposing case pairs highlighted in Table 5.4. The selected cases (and their corresponding columns / rows) are marked yellow. Their intersections with other selected cases are marked green.

We will now use the fact situations of the remaining 12 cases and check if decision constraints apply to these fact situations against case base $\Gamma_{\text{study:consistent}}$. Note that

	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
c_1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
c_2			✓	✓	✓	✓	✓	✓	✓	
c_3	✓		✓	✓	✓		✓	✓		✓
c_4				✓	✓			✓		
c_5	✓		✓	✓	✓	✓		✓		✓
c_6			✓	✓	✓	✓	✓			
c_7			✓	✓	✓		✓	✓		
c_8				✓	✓			✓		
c_9	✓	✓	✓	✓	✓		✓	✓	✓	
c_{10}	✓	✓	✓	✓	✓		✓	✓	✓	✓

Table 5.4: Reduction model consistency for a subset of cases

while the cases these fact situations belong to may be inconsistent with the case base $\Gamma_{\text{study:consistent}}$, this does not necessarily mean that the fact situations themselves are constrained in how they can be decided. Many of the cases are inconsistent with the case base because of the reasons that were chosen to justify a decision in favour of the winning side. But, choosing a stronger reason for these cases could cause the inconsistencies with $\Gamma_{\text{study:consistent}}$ to disappear. Thus, we cannot tell from the inconsistencies of the cases if the fact situations themselves are constrained. This is why we continue to use these existing fact situations for comparing the decision constraints of the reason model and the reduction model, instead of introducing new ones.

For the following fact situations

$$X_2, X_4, X_6, X_7, X_8, X_{11}, X_{12}, X_{15}, X_{17}, X_{19}$$

we have test if a decision is constraint applies for either of the two sides. To test a fact situation X for a decision constraint in favour of s in the reason model, we need to assert that $\Gamma_{\text{study:consistent}} \cup \{\langle X, U_X^{\bar{s}} \rightarrow \bar{s}, \bar{s} \rangle\}$ is inconsistent. If it is, this means that the decision is constrained to s . Otherwise it is not, but it may still be constrained to \bar{s} . Therefore, we also need to test $\Gamma_{\text{study:consistent}} \cup \{\langle X, U_X^s \rightarrow s, s \rangle\}$.

The results for checking the decision constraints in the reason model can be found in Table 5.5. We can see that the reason model with case base $\Gamma_{\text{study:consistent}}$ only constrains the decision of fact situation X_9 to be decided in favour of the defendant—caused by case c_{20} —and the decision of fact situation X_{16} to be decided in favour of the plaintiff—caused by case c_{10} .

Because a decision constraint in the reason model implies a decision constraint in the reduction model, we do not need to check X_9 and X_{16} in the reduction model. This leaves us with only 10 more cases to check in the reduction model. To test that a fact situation is constrained for side s , we need to check if for some case $\langle Y, s \rangle \in \Gamma_{\text{study:consistent}}^\times$ we have $Y \leq^s X$.

	constraint	by case
X_2		
X_4		
X_6		
X_7		
X_8		
X_9	δ	c_{20}
X_{11}		
X_{12}		
X_{15}		
X_{16}	π	c_{10}
X_{17}		
X_{19}		

Table 5.5: Reason model decision constraints

	constraint	by case
X_2	δ	c_{20}
X_4	δ	c_{20}
X_6		
X_7		
X_8		
X_9	δ	c_{20}
X_{11}	δ	c_{20}
X_{12}	π	c_3
X_{15}	δ	c_{20}
X_{16}	π	c_{10}
X_{17}		
X_{19}	π	c_3

Table 5.6: Reduction model decision constraints

The results for the reduction model are given in Table 5.6. There we can see that the reduction model with case base $\Gamma_{\text{study:consistent}}$ constrains the decision for 6 more cases than the reason model. Therefore, 8 fact situations out of the given 12 are constrained in total. Out of these, 5 decision constraints are enforced in favour of the defendant—all of these by case c_{20} —, and 3 in favour of the plaintiff—by cases c_3 and c_{10} . This emphasises the stronger notion of constraint imposed by the reduction model.

5.3 Discussion

The presented case study shows in effect the stronger notion of precedential constraint. We will now analyse the differences presented in Tables 5.5 and 5.6. For this, we will pick out one example where the reason model does not require a decision, but the reduction model does.

Let us look at fact situation X_2 and case c_{20} . For the fact situation we have

$$X_2 = \{\langle d_{\text{protest}}, \text{screamed} \rangle, \langle d_{\text{screen}}, 90 \rangle, \langle d_{\text{activity}}, 75 \rangle, \langle d_{\text{coop}}, \text{ignored} \rangle, \langle d_{\text{read}}, 30 \rangle\}.$$

For the chosen case we have

$$c_{20} = \langle X_{20}, r_{20}, \delta \rangle \text{ where}$$

$$X_{20} = \{\langle d_{\text{protest}}, \text{screamed} \rangle, \langle d_{\text{screen}}, 240 \rangle, \langle d_{\text{activity}}, 15 \rangle, \langle d_{\text{coop}}, \text{ignored} \rangle, \langle d_{\text{read}}, 100 \rangle\} \text{ and}$$

$$r_{20} = \{M_{d_{\text{read}},10}^{\delta}\} \rightarrow \delta.$$

X_2 is not constrained to δ in the reason model. That is because we can find a rule in favour of π , for example

$$r_2 = \{M_{d_{\text{activity}},80}^{\pi}, M_{d_{\text{read}},20}^{\pi}\} \rightarrow \pi.$$

Selecting this rule does not cause any inconsistency with c_{20} , because

$$X_{20} \not\models \text{Premise}(r_2) \quad \text{and} \quad \overline{\text{Premise}(r_{20})} \not\models \text{Premise}(r_2).$$

The first condition $X_{20} \not\models \text{Premise}(r_2)$ holds, because $20 \not\leq_{d_{\text{read}}}^{\pi} 100$. The second condition $\overline{\text{Premise}(r_{20})} \not\models \text{Premise}(r_2)$ holds, because a magnitude factor for dimension d_{activity} is present in $\text{Premise}(r_2)$, but not in $\overline{\text{Premise}(r_{20})}$.

To check the decision constraint in the reduction model, we need to first translate c_{20} to c_{20}^{\times} . The fact situation of c_{20}^{\times} is given as

$$X_{20}^{r_{20}} = \{\langle d_{\text{protest}}, \text{screamed} \rangle, \langle d_{\text{screen}}, 240 \rangle, \langle d_{\text{activity}}, 15 \rangle, \langle d_{\text{coop}}, \text{ignored} \rangle, \langle d_{\text{read}}, 10 \rangle\}.$$

$X_{20}^{r_{20}}$ differs from X_{20} only for the dimension d_{read} where 100 is replaced with 10. Since X_2 is stronger for δ than $X_{20}^{r_{20}}$, i.e.,

$$X_{20}^{r_{20}} \leq^{\delta} X_2,$$

by the reduction model X_2 has to be decided in favour of the defendant as well.

This result is similar to what we have already discussed in Chapter 3. For one, we argue that a decision in favour of δ is necessary, because X_2 (1) meets the requirements of the reason in r_{20} , and (2) for all other dimensions is at least as strong for δ as X_{20} .

Another benefit of the manual evaluation of the models against a dataset is that it allows us to verify the correctness of the programs described in Chapter 4. In particular, the pair-wise consistency of Γ_{study} as well as the decision constraints in $\Gamma_{\text{study:consistent}}$ were tested with both the dimensional reason model and the dimensional reduction model implementations. Because the results of the implementation were the same as in the case study, we have a strong indication that the programs function correctly.

Related Work

In this chapter we will provide an insight into research related to this thesis. For one, we will outline alternative approaches to case-based reasoning that are similar to the models provided in Chapters 2 and 3. Further, we will discuss what frameworks for automating the reasoning process in case-based or rule-based settings exist, and how they differ from our approach presented in Chapter 4. Since related research in the field of normative AI is heavily motivated by legal reasoning, many of the discussed papers are rooted in precedent law. Since legal reasoning is beyond the scope of this thesis, we will focus solely on the normative aspects of the models and model implementations without discussing their application to legal systems in detail.

6.1 Modelling Precedential Constraint

In this section, we discuss work related to our approach of performing case-based reasoning from a theoretical perspective.

An alternative approach to representing reasons in the dimensional reason model is discussed by Rigoni [Rig18]. In his approach, he introduces dimensions that contain a polarity for a side. Where Horty's dimensions merely range from favouring the plaintiff to favouring the defendant, Rigoni's dimensions contain a switching point: Every value in the range before the switching point favours the plaintiff and every value in the range after it favours the defendant. His dimensional reason model also prevents a collapse with the dimensional result model, presenting a viable alternative approach for reasoning with dimensions. Furthermore, Rigoni also discusses dimensions with incomplete information in regard to their value relations or switching points.

As briefly discussed in Chapter 3, in his paper Prakken examines both Horty's and Rigoni's dimensional reason models in-depth [Pra21]. Prakken notices a problematic feature of Horty's revised reason model in [Hor21] where the model admits a rule that

should rather be constrained. This rule is similar to the rule described by Thölke, which we discussed in Example 40. Prakken proposes a model following an approach similar to the dimensional reduction model that rejects such a rule. Instead of translating the case base to the result model, he requires complete rules, i.e., rules where every dimension is present. Then, from both a legal and normative perspective he examines potential limitations posed by further restricting allowed decisions in this way. Further, he discusses the combination of using factors and dimensions, and why factors cannot be simply represented by two-valued dimensions.

An extension to the reason model proposed by Canavotto is reasoning using inconsistent case bases [Can25]. Instead of requiring that a case base is consistent to allow new case decisions to be made, Canavotto proposes that no further inconsistencies are introduced into the reason model when making a new decision. An alternative approach is presented by Morello et al. [MCG25]. They propose treating each applicable precedent as uncertain evidence, and learn its reliability from data. A decision is then proposed along with a confidence store. Especially in real-world scenarios where case decisions are made by different parties, e.g., two parents deciding the children’s bedtime differently, generalising the reason model to enable case decisions under inconsistency enhances the reasoning process.

Another problem with the result model and reason model of constraint is that it requires factual information to be complete. In their work, Odekerken et al. research precedent-based reasoning with incomplete case information for the dimensional result model [OBP23]. Where the dimensional models as we introduced them require every dimension to be assigned a value, Odekerken et al. propose that for each missing dimension all potential candidates for values are explored when making a new case decision. Adding a selection of the missing dimensions is referred to as a case completion. They introduce an alternative notion of constraint for incomplete case bases that requires a decision for a complete fact situation in favour of a side if and only if it is at least as strong as some completion of a potentially incomplete case decided for the same side. Since naively exploring all potential combinations of case decisions results in a combinatorial explosion, Odekerken et al. formulate efficient algorithms to check for consistency. They also model how decisions for incomplete case information can be supported by humans in a semi-automatic process.

Both Canavotto and Horty, and Bench-Capon and Atkinson explore different ideas of representing factual information in case-based reasoning [CH23, BCA22]. Canavotto and Horty discuss the idea that in legal contexts a set of present factors do not directly lead to a decision, but rather that factors should be combined into factor hierarchies of intermediate legal concepts that constitute a decision. This idea can also apply in general normative settings where, for example, a combination of many different factors can constitute “bad behaviour” by a child. Bench-Capon and Atkinson on the other hand explore a more specialised approach tailored towards legal reasoning. They split up the decision process into 4 steps, that is, 1. accepting facts based on evidence, 2. ascribing factors on the basis of the facts 3. resolving the issues on the basis of the factors, and 4.

deciding the outcome on the basis of the issues. Here, issues can be constructed from a hierarchy of factors that is similar to the hierarchy introduced by Canavotto and Horty.

Regarding the retrieval of case information, another notable mention is the work carried out by Bex inquiring about how unreliable information could be incorporated into case-based reasoning systems [Bex11, Bex19]. Bex introduces the notion of stories where different parties report coherent sequences of events in chronological order and outline causal relations between these events. The agent should then reason about what actually happened in the case, as the parties are assumed to tell the stories in their favour. For example, in scenarios regarding childminding, parents could evaluate the factors of a case where conflict between siblings arises, but the parents were not present.

6.2 Automating Case-based Reasoning

In this thesis, we implemented both the dimensional reason model and the dimensional reduction model using ASP. In this section, we will give an overview of related efforts that aim to automate reasoning about decision constraints and permissions.

Among the first efforts to automate case-based reasoning is HYPO, a system for trade secret law proposed by Rissland and Ashley [RA87]. HYPO introduced the notion of dimensions similar to how they are used in the models we introduced throughout the thesis. In particular, for the domain of trade secret law, it used 13 dimensions to model fact situations. To reason about decisions, it deployed a three-ply argument which looks as follows: First, the model selects a case in favour of a side, say the plaintiff, as precedent, and argues why this case is similar enough for constraining the decision. Second, the other side, the defendant, responds by arguing why the precedent should not be followed, and by providing counter examples of similar cases in favour of the defendant. Finally, the plaintiff attempts to rebut the weaknesses pointed out by the defendant by distinguishing the counterexamples mentioned, and showing that the weaknesses pointed out are not fatal.

HYPO influenced many automated reasoning systems. CABARET, developed by Rissland and Skalak, extended the ideas of HYPO to the general setting of case-based reasoning in the legal domain [RS91]. Like HYPO, it includes a series of moves that represent legal arguments. The systems BankXX, developed by Rissland et al, and SPIRE, developed by Rissland and Daniels, aimed at modelling personal bankruptcy [RSF96, RSF97, RD95, RD96, DR97]. BankXX constructs its arguments by combining piece-wise arguments from different cases in the case base. SPIRE on the other hand is a hybrid system that uses case-based reasoning to kickstart finding potential candidates for precedent cases, and then analysing these candidates in-depth by locating relevant text passages from the collection of excerpts.

Another influential system for reasoning about precedential constraint is CATO which was developed by Alevén and Ashley [AA93, Ale03]. CATO, unlike the systems described before, uses Boolean factors which favour either the plaintiff or the defendant. CATO

constructs a hierarchy of factors, and defines a set of argument moves that can be carried out to reason about constraint. These include, for example, analogising a case to a past case with a favourable outcome, downplaying the significance of a decision, or citing a favourable case to emphasise strengths. For CATO, cases in the domain of trade secret law were chosen as a case study.

Where CATO's main purpose was teaching law students about case-based reasoning, the system IBP, developed by Brünninghaus and Ashley, aimed to convert CATO into a program able to predict case outcomes [BA03]. IBP organises the factor hierarchies from CATO into a logical model that specifies under which circumstances a trade secret misappropriation is present.

A precursor to our approach using ASP is presented in a paper by Liu et al. where they aim to combine case-based reasoning with the logical specification of classifiers [LLRS22]. They specify both the standard result model and the standard reason model using the binary-input classifier logic BCL, which was introduced in [LL21]. Furthermore, they also discuss the principle of monotonicity in the result model, and how the reason model relates to it. In particular, they show that the constraints of the result model can be represented as a special case of the reason model. This idea is also the motivation for the reduction model introduced in this thesis.

Conclusion

In case-based reasoning, decisions are made based on past cases and their outcomes. Models of precedential constraint aim to describe how prior decisions influence or limit future ones. Where the result model constrains future decisions based on a fortiori information present in cases [Hor04], the reason model further requires to consider how decisions in past cases were justified to reason about future decisions [HBC12]. Many different approaches to how case information can be represented has been studied in the literature before [HBC12, Rig18, Pra21, CH23]. Where Boolean factors merely describe the presence or absence of an atomic piece of information, dimensions allow for a more nuanced description of information by allowing assignments from a range of values. The reason model was originally introduced using Boolean factors [HBC12]. Upon adjusting the model to the dimensional representation of information, the notion of constraint required revisions [Hor17, Hor21].

Our contribution in this field of research is the proposal of a new model of precedential constraint we call the dimensional reduction model. We identified a desirable property that is present in the standard reason model: In a consistent case base it is always possible to make a reason stronger without causing an inconsistency. However, in the dimensional reason model this property does not apply. Instead, strengthening a reason can cause inconsistencies in an otherwise consistent case base. We motivated the definition of the dimensional reduction model by first considering an alternative representation of the standard reason model, and then generalising this alternative representation to dimensions. Additionally, significant examples from the literature were addressed to further justify our model.

To enable automated reasoning, we implemented both the dimensional reason model and the dimensional reduction model in ASP. The implementation enables to check for inconsistencies in an existing case base where certain predicates in the answer set allow for pinpointing what causes certain cases to be consistent or inconsistent, enhancing the explainability of the program. Further, given a consistent case base, the program is able

to check what decision constraints apply for new fact situations. Again, information is provided pinpointing the cases that cause the inconsistency.

To further explore the differences between the dimensional reason model and the dimensional reduction model, we generated a synthetic case study which is set in the nanny robot scenario from [CH22a]. The dataset contains five hand-picked dimensions and spans 20 cases, 10 of which are decided in favour of the plaintiff, and 10 of which are decided in favour of the defendant. An analysis of the case base showed that out of 100 combinations of opposing cases (10 in favour of π times 10 in favour of δ), that needed to be compared for evaluating consistency, 20 are inconsistent in the reason model whereas 35 are inconsistent in the reduction model. Also, when picking a consistent subset of 8 cases for a case base, checking for decision constraints on 12 fact situations showed that the reason model only constrains 2 fact situations whereas the reduction model constrains 8 fact situations. Since the dataset was evaluated by hand, it was also used to help verify the correctness of the implementation, both for the consistency check as well as for checking applicable decision constraints.

7.1 Future Work

Several aspects of the discussed topics could be taken into account in future works. For one, we could further explore how precedential constraint can be modelled. Using dimensions, an open problem remains when a shift in how cases are argued occurs: When reading 10 pages is considered a lot for 5-year-olds, but very little for 20-year-olds this change in what is considered “a lot” cannot be simply captured by dimensions. For this, either exploring a different representation of information, or further refining how dimensions can be defined, could yield fruitful results. Another aspect that has not been explored in this thesis, but is subject to active investigation in related work [Can25, MCG25], is working with inconsistencies in the case base. In particular, when working with multiple parties such as the two parents involved in the nanny robot scenario, further investigations could be performed to inquire how conflicting resolutions could be considered.

Further research for automating case-based reasoning could explore involving sub-symbolic AI such as *large language models* (LLMs) for extracting case information. The challenge here lies within selecting suitable dimensions to use for a domain. As an alternative approach one could define the dimensions by hand and then extract the cases by trying to fit the present information to the defined dimensions. But this again yields the challenge of defining and parsing the values for each dimension. Instead, parsing the dimensions and cases semi-automatically by querying the user for guidance in case of uncertainty could greatly improve the reliability of generating a case base.

Another area of investigation is the exploration of cases beyond precedents. It would be interesting to inquire how meaningful rules could be proposed for new fact situations where no precedent applies. In the setting of dimensions, for example, a simple idea for a new fact situation without precedent is to try and find the cases in the case base with

the “closest” fact situations. This inquiry is subject to the question, of how preferences can be generalised from a finite case base not covering all potentials.

Overview of Generative AI Tools Used

Übersicht verwendeter Hilfsmittel

List of Figures

2.1	Decision constraints in the dimensional result model	35
2.2	Inconsistency in the dimensional result model	36
3.1	Decision constraints in the dimensional reduction model	68
4.1	Visual representation of implementation: First the <code>not_consistent</code> relation is constructed. Then, checks for consistency and decision constraints are performed.	84

List of Tables

5.1	Case study dataset	103
5.2	Reason model consistency for case study	104
5.3	Reduction model consistency for case study	105
5.4	Reduction model consistency for a subset of cases	106
5.5	Reason model decision constraints	107
5.6	Reduction model decision constraints	107

List of Algorithms

Reason Model Implementation

```
1  %%% SIDES %%%
2
3  side(plaintiff).
4  side(defendant).
5
6  opposite_side(plaintiff, defendant).
7  opposite_side(defendant, plaintiff).
8
9  %%% VALUE RELATION %%%
10
11 dimension_value(Dimension, Value) :- value_assignment(Case, Dimension, Value)
12      .
13 dimension_value(Dimension, Value) :- magnitude_factor(Case, Dimension, Value)
14      .
15
16 value_relation(Side, Dimension, Value1, Value2) :-
17     int_dimension_strengthens(Dimension, Side),
18     dimension_value(Dimension, Value1),
19     dimension_value(Dimension, Value2),
20     Value1 <= Value2.
21
22 custom_dimension_ordering(Dimension, Value1, Value3) :-
23     custom_dimension_ordering(Dimension, Value1, Value2),
24     custom_dimension_ordering(Dimension, Value2, Value3).
25
26 value_relation(Side, Dimension, Value1, Value2) :-
27     custom_dimension_strengthens(Dimension, Side),
28     dimension_value(Dimension, Value1),
29     dimension_value(Dimension, Value2),
30     custom_dimension_ordering(Dimension, Value1, Value2).
31
32 % reflexivity
33 value_relation(Side, Dimension, Value, Value) :-
34     side(Side),
35     dimension_value(Dimension, Value).
36
37 % transitivity
```

```
36 value_relation(Side, Dimension, Value1, Value3) :-
37     value_relation(Side, Dimension, Value1, Value2),
38     value_relation(Side, Dimension, Value2, Value3).
39
40 % antisymmetry
41 :-
42     value_relation(Side, Dimension, Value1, Value2),
43     value_relation(Side, Dimension, Value2, Value1),
44     Value1 != Value2.
45
46 % duality
47 value_relation(OppositeSide, Dimension, Value2, Value1) :-
48     value_relation(Side, Dimension, Value1, Value2),
49     opposite_side(Side, OppositeSide).
50
51 strict_value_relation(Side, Dimension, Value1, Value2) :-
52     value_relation(Side, Dimension, Value1, Value2),
53     Value1 != Value2.
54
55 %%% CONSISTENCY %%%
56
57 case_like(Case, Side) :-
58     case(Case, Side).
59
60 magnitude_factor_like(Case, Dimension, ThresholdValue) :-
61     magnitude_factor(Case, Dimension, ThresholdValue).
62
63 has_magnitude_factor_like(Case, Dimension) :-
64     magnitude_factor_like(Case, Dimension, ThresholdValue).
65
66 opposing_case_likes(Case, Side, OppCase, OppSide) :-
67     case_like(Case, Side),
68     case_like(OppCase, OppSide),
69     opposite_side(Side, OppSide),
70     Case != OppCase.
71
72 not_value_assignment_satisfies_opposing_magnitude_factor(Case, Side, OppCase,
73     OppSide, Dimension) :-
74     % pick two opposing cases
75     opposing_case_likes(Case, Side, OppCase, OppSide),
76     % pick a value assignment from the fact situation of the first case
77     value_assignment(Case, Dimension, Value),
78     % pick the matching magnitude factor from the reason of the second case
79     magnitude_factor_like(OppCase, Dimension, ThresholdValue),
80     % assert that the value assignment does not satisfy the magnitude factor
81     strict_value_relation(OppSide, Dimension, Value, ThresholdValue).
82
83 not_fact_situation_satisfies_opposing_reason(Case, Side, OppCase, OppSide) :-
84     not_value_assignment_satisfies_opposing_magnitude_factor(Case, Side,
85     OppCase, OppSide, Dimension).
```

```

86  % pick two opposing cases
87  opposing_case_likes(Case, Side, OppCase, OppSide),
88  % ensure that the magnitude factor from the second case is present
89  has_magnitude_factor_like(OppCase, Dimension),
90  % ensure that the magnitude factor from the first case is not present
91  not has_magnitude_factor_like(Case, Dimension).
92
93 not_negated_magnitude_factor_satisfies_opposing_magnitude_factor(Case, Side,
OppCase, OppSide, Dimension) :-
94  % pick two opposing cases
95  opposing_case_likes(Case, Side, OppCase, OppSide),
96  % pick a magnitude factor from the reason of the first case
97  magnitude_factor_like(Case, Dimension, ThresholdValue1),
98  % pick the matching magnitude factor from the reason of the second case
99  magnitude_factor_like(OppCase, Dimension, ThresholdValue2),
100 % check that the magnitude in the first case is weaker
101 % than the magnitude in the second case
102 strict_value_relation(OppSide, Dimension, ThresholdValue1,
ThresholdValue2).
103
104 not_negated_reason_satisfies_opposing_reason(Case, Side, OppCase, OppSide) :-
105  not_negated_magnitude_factor_present_for_opposing_magnitude_factor(Case,
Side, OppCase, OppSide, Dimension).
106
107 not_negated_reason_satisfies_opposing_reason(Case, Side, OppCase, OppSide) :-
108  not_negated_magnitude_factor_satisfies_opposing_magnitude_factor(Case,
Side, OppCase, OppSide, Dimension).
109
110 not_consistent_cross_condition(Case, Side, OppCase, OppSide) :-
111  opposing_case_likes(Case, Side, OppCase, OppSide),
112  not not_fact_situation_satisfies_opposing_reason(Case, Side, OppCase,
OppSide).
113
114 not_consistent_cross_condition(Case, Side, OppCase, OppSide) :-
115  opposing_case_likes(Case, Side, OppCase, OppSide),
116  not not_negated_reason_satisfies_opposing_reason(Case, Side, OppCase,
OppSide).
117
118 not_consistent(Case, Side, OppCase, OppSide) :-
119  not_consistent_cross_condition(Case, Side, OppCase, OppSide),
120  not_consistent_cross_condition(OppCase, OppSide, Case, Side).
121
122 not_cases_consistent(Case, Side, OppCase, OppSide) :-
123  case(Case, Side),
124  case(OppCase, OppSide),
125  not_consistent(Case, Side, OppCase, OppSide).
126
127 not_case_base_consistent :-
128  not_cases_consistent(Case, Side, OppCase, OppSide).
129
130 %%% DECISION %%%
131
132 decision(Case, Side) :-
    
```

```
133     unsolved_case(Case),
134     side(Side),
135     not not_case_base_consistent.
136
137 case_like(Case, Side) :-
138     decision(Case, Side).
139
140 magnitude_factor_like(Case, Dimension, Value) :-
141     decision(Case, Side),
142     value_assignment(Case, Dimension, Value).
143
144 not_decision_consistent_with_case(Case, Side, OppCase, OppSide) :-
145     decision(Case, Side),
146     case(OppCase, OppSide),
147     not_consistent(Case, Side, OppCase, OppSide).
148
149 not_decision_permmissible(Case, Side) :-
150     not_decision_consistent_with_case(Case, Side, OppCase, OppSide).
```

Reduction Model Implementation

```
1  %%% SIDES %%%
2
3  side(plaintiff).
4  side(defendant).
5
6  opposite_side(plaintiff, defendant).
7  opposite_side(defendant, plaintiff).
8
9  %%% VALUE RELATION %%%
10
11 dimension_value(Dimension, Value) :- value_assignment(Case, Dimension, Value)
12 .
13 dimension_value(Dimension, Value) :- magnitude_factor(Case, Dimension, Value)
14 .
15
16 value_relation(Side, Dimension, Value1, Value2) :-
17     int_dimension_strengthens(Dimension, Side),
18     dimension_value(Dimension, Value1),
19     dimension_value(Dimension, Value2),
20     Value1 <= Value2.
21
22 custom_dimension_ordering(Dimension, Value1, Value3) :-
23     custom_dimension_ordering(Dimension, Value1, Value2),
24     custom_dimension_ordering(Dimension, Value2, Value3).
25
26 value_relation(Side, Dimension, Value1, Value2) :-
27     custom_dimension_strengthens(Dimension, Side),
28     dimension_value(Dimension, Value1),
29     dimension_value(Dimension, Value2),
30     custom_dimension_ordering(Dimension, Value1, Value2).
```

```
30 % reflexivity
31 value_relation(Side, Dimension, Value, Value) :-
32     side(Side),
33     dimension_value(Dimension, Value).
34
35 % transitivity
36 value_relation(Side, Dimension, Value1, Value3) :-
37     value_relation(Side, Dimension, Value1, Value2),
38     value_relation(Side, Dimension, Value2, Value3).
39
40 % antisymmetry
41 :-
42     value_relation(Side, Dimension, Value1, Value2),
43     value_relation(Side, Dimension, Value2, Value1),
44     Value1 != Value2.
45
46 % duality
47 value_relation(OppositeSide, Dimension, Value2, Value1) :-
48     value_relation(Side, Dimension, Value1, Value2),
49     opposite_side(Side, OppositeSide).
50
51 strict_value_relation(Side, Dimension, Value1, Value2) :-
52     value_relation(Side, Dimension, Value1, Value2),
53     Value1 != Value2.
54
55 %%% TRANSLATION %%%
56
57 has_magnitude_factor(Case, Dimension) :-
58     magnitude_factor(Case, Dimension, ThresholdValue).
59
60 % use magnitude factors if present
61 reduced_value_assignment(Case, Dimension, Value) :-
62     magnitude_factor(Case, Dimension, Value).
63
64 % use value assignments otherwise
65 reduced_value_assignment(Case, Dimension, Value) :-
66     case(Case, Side),
67     value_assignment(Case, Dimension, Value),
68     not has_magnitude_factor(Case, Dimension).
69
70 %%% CONSISTENCY %%%
71
72 case_like(Case, Side) :- case(Case, Side).
73 reduced_value_assignment_like(Case, Dimension, Value) :-
74     reduced_value_assignment(Case, Dimension, Value).
75
76 opposing_case_likes(Case, Side, OppCase, OppSide) :-
77     case_like(Case, Side),
78     case_like(OppCase, OppSide),
79     opposite_side(Side, OppSide),
80     Case != OppCase.
81
```

```
82 not_value_assignment_leq_opposing_value_assignment(Case, Side, OppCase,
OppSide, Dimension) :-
83     % pick two opposing cases
84     opposing_case_likes(Case, Side, OppCase, OppSide),
85     % pick the reduced value assignment from the first case
86     reduced_value_assignment_like(Case, Dimension, Value1),
87     % pick the matching reduced value assignment from the second case
88     reduced_value_assignment_like(OppCase, Dimension, Value2),
89     % check if the reduced value assignment from the first case is stronger
90     strict_value_relation(Side, Dimension, Value2, Value1).
91
92 not_fact_situation_leq_opposing_fact_situation(Case, Side, OppCase, OppSide)
:-
93     not_value_assignment_leq_opposing_value_assignment(Case, Side, OppCase,
OppSide, Dimension).
94
95 not_consistent(Case, Side, OppCase, OppSide) :-
96     opposing_case_likes(Case, Side, OppCase, OppSide),
97     not not_fact_situation_leq_opposing_fact_situation(Case, Side, OppCase,
OppSide).
98
99 not_cases_consistent(Case, Side, OppCase, OppSide) :-
100     case(Case, Side),
101     case(OppCase, OppSide),
102     not_consistent(Case, Side, OppCase, OppSide).
103
104 not_case_base_consistent :-
105     not_cases_consistent(Case, Side, OppCase, OppSide).
106
107 %%% DECISION %%%
108
109 decision(Case, Side) :-
110     unsolved_case(Case),
111     side(Side),
112     not not_case_base_consistent.
113
114 case_like(Case, Side) :-
115     decision(Case, Side).
116
117 reduced_value_assignment_like(Case, Dimension, Value) :-
118     decision(Case, Side),
119     value_assignment(Case, Dimension, Value).
120
121 not_decision_consistent_with_case(Case, Side, OppCase, OppSide) :-
122     decision(Case, Side),
123     case(OppCase, OppSide),
124     not_consistent(Case, Side, OppCase, OppSide).
125
126 not_decision_permmissible(Case, Side) :-
127     not_decision_consistent_with_case(Case, Side, OppCase, OppSide).
```

Synthetic Dataset Generation

```
1 import random
2
3 ### INPUT ###
4
5 case_number = 20
6
7 def value_list(start, stop, step):
8     return list(range(start, stop + 1, step))
9
10 dimensions = [
11     {
12         "name": "protest",
13         "strengthens": "plaintiff",
14         "values": ["none", "complained", "cried", "screamed", "kicked"]
15     },
16     {
17         "name": "screen",
18         "strengthens": "plaintiff",
19         "values": value_list(0, 300, 15),
20         "magnitude_values": value_list(0, 300, 30)
21     },
22     {
23         "name": "activity",
24         "strengthens": "defendant",
25         "values": value_list(0, 300, 15),
26         "magnitude_values": value_list(0, 300, 30),
27     },
28     {
29         "name": "coop",
30         "strengthens": "defendant",
31         "values": ["physical", "verbal", "ignored", "played"]
32     },
33     {
34         "name": "read",
35         "strengthens": "defendant",
36         "values": value_list(0, 100, 5),
37         "magnitude_values": value_list(0, 100, 10)
38     }
39 ]
40
41 ### PROGRAM ###
42
43 dimensions_dict = dict()
44 dimensions_len = len(dimensions)
45
46 for dim in dimensions:
47     dimensions_dict[dim["name"]] = dim
48
49 cases = []
50
51 for case_id in range(1, case_number + 1):
```

```
52 facts = []
53 plaintiff_probs = []
54
55 for dim in dimensions:
56     values = dim["values"]
57     values_len = len(values)
58
59     value = random.choice(values)
60     index = values.index(value)
61
62     if dim["strengthens"] == "defendant":
63         index = values_len - index
64
65     plaintiff_prob = index / values_len
66     plaintiff_prob = max(0, min(1, plaintiff_prob))
67
68     plaintiff_probs.append(plaintiff_prob)
69
70     facts.append({
71         "name": dim["name"],
72         "value": value,
73         "plaintiff_prob": plaintiff_prob
74     })
75
76 plaintiff_total_prob = sum(plaintiff_probs) / dimensions_len
77 side = "plaintiff" if random.random() >= plaintiff_total_prob else "
78 defendant"
79
80 magnitudes = []
81
82 while len(magnitudes) == 0:
83     for fact in facts:
84         threshold = fact["plaintiff_prob"]
85
86         if side == "plaintiff":
87             threshold = 1 - threshold
88
89         if random.random() < threshold:
90             continue
91
92         fact_name = fact["name"]
93         fact_value = fact["value"]
94
95         dim = dimensions_dict[fact_name]
96         dim_values = dim["values"]
97         dim_magnitude_values = dim["magnitude_values"] if "
98 magnitude_values" in dim else dim_values
99
100         dim_allowed_values = []
101
102         if side == dim["strengthens"]:
103             for value in dim_values:
104                 dim_allowed_values.append(value)
```

```
103         if value == fact_value:
104             break
105     else:
106         value_reached = False
107
108         for value in dim_values:
109             if value == fact_value:
110                 value_reached = True
111
112             if value_reached:
113                 dim_allowed_values.append(value)
114
115         dim_allowed_values.reverse()
116
117     dim_allowed_magnitude_values = []
118
119     for value in dim_allowed_values:
120         if value in dim_magnitude_values:
121             dim_allowed_magnitude_values.append(value)
122
123     dim_scaled_allowed_magnitude_values = []
124
125     i = 1
126     for value in dim_allowed_magnitude_values:
127         for _ in range(i):
128             dim_scaled_allowed_magnitude_values.append(value)
129
130         i += 1
131
132     random.shuffle(dim_scaled_allowed_magnitude_values)
133     value = random.choice(dim_scaled_allowed_magnitude_values)
134
135     magnitudes.append({
136         "name": fact_name,
137         "value": value
138     })
139
140     cases.append({
141         "side": side,
142         "facts": facts,
143         "magnitudes": magnitudes
144     })
145
146 sep = r"%%%"
147
148 print(f"{sep} DIMENSIONS {sep}\n")
149
150 for dim in dimensions:
151     dim_name = dim["name"]
152     dim_strengthens = dim["strengthens"]
153     dim_values = dim["values"]
```

```

156     if isinstance(dim_values[0], int):
157         print(f"int_dimension_strengthens({dim_name}, {dim_strengthens}).")
158     else:
159         print(f"custom_dimension_strengthens({dim_name}, {dim_strengthens})."
160             )
161
162     for i in range(1, len(dim_values)):
163         print(f"custom_dimension_ordering({dim_name}, {dim_values[i -
164             1]}, {dim_values[i]}).")
165
166     print()
167 print(f"{sep} CASES {sep}\n")
168
169 cases.sort(key=lambda x: 0 if x["side"] == "plaintiff" else 1)
170
171 case_id = 1
172
173 for case in cases:
174     print(f"case({case_id}, {case["side"]}).")
175
176     for fact in case["facts"]:
177         print(f"value_assignment({case_id}, {fact["name"]}, {fact["value"]}.
178             ")
179
180     for magnitude in case["magnitudes"]:
181         print(f"magnitude_factor({case_id}, {magnitude["name"]}, {magnitude["
182             value"}]).")
183
184     print()
185
186     case_id += 1

```

Appendix: Background

Proposition 1: Let Γ be a consistent case base and X be a fact situation. A decision is constrained to side s if and only if $\Gamma \cup \{\langle X, \bar{s} \rangle\}$ is inconsistent. Equivalently, a decision in favour of side s is permitted if and only if $\Gamma \cup \{\langle X, s \rangle\}$ is consistent.

Proof. Let Γ be a consistent case base and X be a fact situation.

Let us first show that decision constraints using consistency are equivalent.

“ \implies ” Assume a decision in favour of side s is constrained. This means that for some case $\langle Y, s \rangle \in \Gamma$ we have $Y \leq^s X$. Thus, we know that $\Gamma \cup \{\langle X, \bar{s} \rangle\}$ must be inconsistent.

“ \impliedby ” Assume that $\Gamma \cup \{\langle X, \bar{s} \rangle\}$ is inconsistent. This means that for some $\langle Y, s \rangle \in \Gamma$ we have $Y \leq^s X$. Therefore, a decision in favour of s is required.

Let us now show that decision permissions using consistency are equivalent.

“ \implies ” Assume a decision in favour of side s is permitted. This means that for every case $\langle Y, \bar{s} \rangle \in \Gamma$ we have $Y \not\leq^{\bar{s}} X$. Thus, we know that $\Gamma \cup \{\langle X, s \rangle\}$ is consistent too.

“ \impliedby ” Assume that $\Gamma \cup \{\langle X, s \rangle\}$ is consistent. This means that for every $\langle Y, \bar{s} \rangle \in \Gamma$ we have $Y \not\leq^{\bar{s}} X$. Therefore, a decision in favour of \bar{s} is not required which means that a decision in favour of s is permitted. \square

Proposition 2 Let Γ be a consistent case base and X be a fact situation. By the result model, a decision in favour of at least one of the two sides π and δ is always possible.

Proof. Let Γ be a consistent case base and X be a fact situation. Assume for contradiction that X cannot be decided for any side. This means that there have to be cases $\langle Y, s \rangle, \langle Y', \bar{s} \rangle \in \Gamma$ such that $Y \leq^s X$ and $Y' \leq^{\bar{s}} X$. The latter is equivalent to $X \leq^s Y'$. By transitivity of \leq^s , we get $Y \leq^s Y'$. But then, Γ is inconsistent which is a contradiction. \square

Proposition 3: Let Γ be a case base. Then, Γ is inconsistent if and only if there are two cases $c, c' \in \Gamma$ where

$$c = \langle X, r, s \rangle \text{ and } c' = \langle Y, r', \bar{s} \rangle$$

such that

$$\text{Premise}(r') <_c \text{Premise}(r) \text{ and } \text{Premise}(r) <_{c'} \text{Premise}(r').$$

Because $\text{Premise}(r) \Vdash \text{Premise}(r)$ and $\text{Premise}(r') \Vdash \text{Premise}(r')$ always hold trivially, the statements are respectively equivalent to

$$X \models \text{Premise}(r') \text{ and } Y \models \text{Premise}(r).$$

Proof. To show Proposition 3, we need to we consider some case base Γ . The condition for inconsistency as per Definition 17 is that there are two reasons W and Z of opposing sides \bar{s} and s such that $W <_{\Gamma} Z$ and $Z <_{\Gamma} W$. As previously described, this is equivalent to the fact that there are two cases $c, c' \in \Gamma$ where $c = \langle X, r, s \rangle$ and $c' = \langle Y, r', \bar{s} \rangle$ such that $W <_c Z$ and $Z <_{c'} W$. We show both sides:

“ \implies ”: Assume Γ is consistent. Therefore, for all reasons W and Z of opposing sides \bar{s} and s , and cases c, c' where $c = \langle X, r, s \rangle$ and $c' = \langle Y, r', \bar{s} \rangle$, either $W \not<_c Z$ or $Z \not<_{c'} W$. Thus, there are no two cases $c, c' \in \Gamma$ where $c = \langle X, r, s \rangle$ and $c' = \langle Y, r', \bar{s} \rangle$ such that $\text{Premise}(r') <_c \text{Premise}(r)$ and $\text{Premise}(r) <_{c'} \text{Premise}(r')$.

“ \impliedby ”: Assume Γ is inconsistent. Therefore, there are two reasons W and Z of opposing sides \bar{s} and s such that $W <_{\Gamma} Z$ and $Z <_{\Gamma} W$. That is, there are two cases $c, c' \in \Gamma$ where $c = \langle X, r, s \rangle$ and $c' = \langle Y, r', \bar{s} \rangle$ such that $W <_c Z$ and $Z <_{c'} W$. We know that $W <_c Z$ is equivalent to $X \models W$ and $Z \Vdash \text{Premise}(r)$, and $Z <_{c'} W$ is equivalent to $Y \models Z$ and $W \Vdash \text{Premise}(r')$. Because of $X \models W$ and $W \Vdash \text{Premise}(r')$, we can derive that $X \models \text{Premise}(r')$, and because of $Y \models Z$ and $Z \Vdash \text{Premise}(r)$ we can derive that $Y \models R$. Thus, for these two cases c and c' we have $\text{Premise}(r') <_c \text{Premise}(r)$ and $\text{Premise}(r) <_{c'} \text{Premise}(r')$. \square

Proposition 5: Let Γ be a consistent case base and X be a fact standard situation. By the reason model, a decision in favour of at least one of the two sides π and δ is always possible.

Proof. Let Γ be a consistent case base and X be a fact situation. Assume for contradiction that X cannot be decided for any side. By Proposition 4, a decision for some side s is not permitted if and only if $\Gamma \cup \{\langle X, X^s, s \rangle\}$ is inconsistent. This means there is a case $\langle Y', r', s \rangle \in \Gamma$ such that $X \models \text{Premise}(r')$ and $Y' \models X^s$. So, if no decision is permitted, there must be two cases $\langle Y, r, s \rangle, \langle Y', r', \bar{s} \rangle \in \Gamma$ such that

1. $X \models \text{Premise}(r)$ and $Y \models X^{\bar{s}}$, and
2. $X \models \text{Premise}(r')$ and $Y' \models X^s$.

Since $Y \models X^{\bar{s}}$ and $X \models \text{Premise}(r')$, we know that $Y \models \text{Premise}(r')$. Conversely, since $Y' \models X^s$ and $X \models \text{Premise}(r)$, we know that $Y' \models \text{Premise}(r)$. Since $Y \models \text{Premise}(r')$ and $X \models \text{Premise}(r)$ is the condition for inconsistency by Proposition 3, we know that

case base Γ is inconsistent which contradicts our assumption. Therefore, a decision in favour of one of the two sides is always possible. \square

Proposition 11: Let Γ be a dimensional case base. Then, Γ is inconsistent if and only if there are two cases $c, c' \in \Gamma$ where

$$c = \langle X, r, s \rangle \text{ and } c' = \langle Y, r', \bar{s} \rangle$$

such that

$$\text{Premise}(r') <_c \text{Premise}(r) \text{ and } \text{Premise}(r) <_{c'} \text{Premise}(r').$$

Because $\text{Premise}(r) \Vdash \text{Premise}(r)$ and $\text{Premise}(r') \Vdash \text{Premise}(r')$ always hold trivially, the statements are equivalent to

1. $X \models \text{Premise}(r')$ or $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$, and
2. $Y \models \text{Premise}(r)$ or $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$

respectively.

Proof. Let Γ be a case base. The condition for inconsistency per Definition 17 is that there are two reasons W and Z of opposing sides \bar{s} and s such that $W <_{\Gamma} Z$ and $Z <_{\Gamma} W$. As previously described, this is equivalent to the fact that there are two cases $c, c' \in \Gamma$ where $c = \langle X, r, s \rangle$ and $c' = \langle Y, r', \bar{s} \rangle$ such that $W <_c Z$ and $Z <_{c'} W$. We show both sides:

“ \implies ”: Assume Γ is consistent. Therefore, for all reasons W and Z of opposing sides \bar{s} and s , and cases c, c' where $c = \langle X, r, s \rangle$ and $c' = \langle Y, r', \bar{s} \rangle$, either $W \not<_c Z$ or $Z \not<_{c'} W$. Thus, there are no two cases $c, c' \in \Gamma$ where $c = \langle X, r, s \rangle$ and $c' = \langle Y, r', \bar{s} \rangle$ such that $\text{Premise}(r') <_c \text{Premise}(r)$ and $\text{Premise}(r) <_{c'} \text{Premise}(r')$.

“ \impliedby ”: Assume Γ is inconsistent. Therefore, there are two reasons W and Z of opposing sides \bar{s} and s such that $W <_{\Gamma} Z$ and $Z <_{\Gamma} W$. That is, there are two cases $c, c' \in \Gamma$ where $c = \langle X, r, s \rangle$ and $c' = \langle Y, r', \bar{s} \rangle$ such that $W <_c Z$ and $Z <_{c'} W$.

We know that $W <_c Z$ is equivalent to $Z \Vdash \text{Premise}(r)$, and $X \models W$ or $\overline{\text{Premise}(r)} \Vdash W$. From $W \Vdash \text{Premise}(r')$ we can derive that if $X \models W$, then $X \models \text{Premise}(r')$, and if $\overline{\text{Premise}(r)} \Vdash W$, then $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$.

Likewise, we know that $Z <_{c'} W$ is equivalent to $W \Vdash \text{Premise}(r')$, and $Y \models Z$ or $\overline{\text{Premise}(r')} \Vdash Z$. From $Z \Vdash \text{Premise}(r)$ we can derive that if $Y \models Z$, then $Y \models \text{Premise}(r)$, and if $\overline{\text{Premise}(r')} \Vdash Z$, then $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$. \square

Proposition 13: Let Γ be a consistent dimensional case base and X be a dimensional fact situation. By the revised dimensional reason model, a decision in favour of at least one of the two sides π and δ is always possible.

Proof. Let Γ be a consistent case base and X be a fact situation. Assume for contradiction that X cannot be decided for any side. If a decision is not possible for some side s , by Proposition 11 there must be a case $\langle Y', r', \bar{s} \rangle \in \Gamma$ such that $X \models \text{Premise}(r)$ and $(X \leq^{\bar{s}} Y \text{ or } \overline{\text{Premise}(r)} \Vdash U_X^{\bar{s}})$. Thus, if no decision is possible, there must be two cases $c, c' \in \Gamma$ where $c = \langle Y, r, s \rangle$ and $c' = \langle Y', r', \bar{s} \rangle$ such that

1. $X \models \text{Premise}(r)$ and $(X \leq^{\bar{s}} Y \text{ or } \overline{\text{Premise}(r)} \Vdash U_X^{\bar{s}})$, and
2. $X \models \text{Premise}(r')$ and $(X \leq^s Y' \text{ or } \overline{\text{Premise}(r')} \Vdash U_X^s)$.

W.l.o.g., we make three case distinctions:

- Case 1: $X \leq^{\bar{s}} Y$ and $X \leq^s Y'$. From $X \leq^{\bar{s}} Y$ follows that $Y \leq^s X$. Thus, by transitivity, we have $Y \leq^s Y'$, and thus $Y' \leq^{\bar{s}} Y$ by duality. Since $Y \models \text{Premise}(r)$ and $Y \leq^s Y'$, we have $Y' \models \text{Premise}(r)$. By $Y' \models \text{Premise}(r)$ and $Y' \leq^{\bar{s}} Y$ we know that c and c' are inconsistent.
- Case 2: $X \leq^{\bar{s}} Y$ and $\overline{\text{Premise}(r')} \Vdash U_X^s$. Since $X \leq^{\bar{s}} Y$ and $X \models \text{Premise}(r')$, we have $Y \models \overline{\text{Premise}(r')}$. Since by $\overline{\text{Premise}(r')} \Vdash U_X^s$ we know that every magnitude factor in $\overline{\text{Premise}(r')}$ is at least as strong for side s as every value assignment in X , and by $X \models \text{Premise}(r)$ every value assignment in X is at least as strong for side s as every magnitude factor in $\text{Premise}(r)$, by transitivity we have $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$. By $Y \models \text{Premise}(r')$ and $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$, we know that c and c' are inconsistent.
- Case 3: $\overline{\text{Premise}(r)} \Vdash U_X^{\bar{s}}$ and $\overline{\text{Premise}(r')} \Vdash U_X^s$. Since $\overline{\text{Premise}(r)} \Vdash U_X^{\bar{s}}$, we know that each dimension $d \in D$ must be present in $\overline{\text{Premise}(r)}$. Thus, by duality we know that $U_X^s \Vdash \text{Premise}(r)$ holds as well. Since $\overline{\text{Premise}(r')} \Vdash U_X^s$ and $U_X^s \Vdash \text{Premise}(r)$, by transitivity we get $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$. We can make the same argument for $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$. Since $\overline{\text{Premise}(r')} \Vdash \text{Premise}(r)$ and $\overline{\text{Premise}(r)} \Vdash \text{Premise}(r')$, we know that c and c' are inconsistent.

In all three case distinctions Γ is inconsistent which contradicts our assumption. Thus, a decision in favour of some side is always possible. \square

Bibliography

- [AA93] Vincent Aleven and Kevin D. Ashley. What law students need to know to WIN. In *Proceedings of the 4th international conference on Artificial intelligence and law*, ICAIL '93, pages 152–161, New York, NY, USA, August 1993. Association for Computing Machinery.
- [Ale89] Larry Alexander. Constrained by Precedent. *Southern California Law Review*, 63:1, 1989.
- [Ale03] Vincent Aleven. Using background knowledge in case-based legal reasoning: A computational model and an intelligent learning environment. *Artificial Intelligence*, 150(1):183–237, 2003. AI and Law.
- [BA03] Stefanie Brüninghaus and Kevin D. Ashley. Combining case-based and model-based reasoning for predicting the outcome of legal cases. In Kevin D. Ashley and Derek G. Bridge, editors, *Case-Based Reasoning Research and Development*, pages 65–79, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [BCA22] Trevor Bench-Capon and Katie Atkinson. Using Argumentation Schemes to Model Legal Reasoning, October 2022. arXiv:2210.00315 [cs].
- [Bex11] Floris J. Bex. A Hybrid Theory of Stories and Arguments. In Floris J. Bex, editor, *Arguments, Stories and Criminal Evidence: A Formal Hybrid Theory*, pages 83–100. Springer Netherlands, Dordrecht, 2011.
- [Bex19] Floris Bex. The Hybrid Theory of Stories and Arguments Applied to the Simonshaven Case. *Topics in Cognitive Science*, 12, May 2019.
- [Can25] Ilaria Canavotto. Reasoning with inconsistent precedents. *Artificial Intelligence and Law*, 33(1):137–166, March 2025.
- [CH22a] Ilaria Canavotto and John Horty. Piecemeal knowledge acquisition for computational normative reasoning. In *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '22, page 171–180, New York, NY, USA, 2022. Association for Computing Machinery.

- [CH22b] Ilaria Canavotto and John Horty. Piecemeal Knowledge Acquisition for Computational Normative Reasoning. In *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '22, pages 171–180, New York, NY, USA, July 2022. Association for Computing Machinery.
- [CH23] Ilaria Canavotto and John Horty. Reasoning with hierarchies of open-textured predicates. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, ICAIL '23, pages 52–61, New York, NY, USA, September 2023. Association for Computing Machinery.
- [DR97] Jody J. Daniels and Edwina L. Rissland. Finding legally relevant passages in case opinions. In *Proceedings of the sixth international conference on Artificial intelligence and law - ICAIL '97*, pages 39–46, Melbourne, Australia, 1997. ACM Press.
- [HBC12] John F. Horty and Trevor J. M. Bench-Capon. A factor-based definition of precedential constraint. *Artificial Intelligence and Law*, 20(2):181–214, May 2012.
- [Hor04] John F. Horty. THE RESULT MODEL OF PRECEDENT. *Legal Theory*, 10(1):19–31, March 2004.
- [Hor17] John Horty. Reasoning with dimensions and magnitudes. In *Proceedings of the 16th Edition of the International Conference on Artificial Intelligence and Law*, ICAIL '17, page 109–118, New York, NY, USA, 2017. Association for Computing Machinery.
- [Hor21] John Horty. Modifying the reason model. *Artificial Intelligence and Law*, 29(2):271–285, Jun 2021.
- [Lif19] Vladimir Lifschitz. *Answer Set Programming*. Springer International Publishing, Cham, 2019.
- [LL21] Xinghan Liu and Emiliano Lorini. A Logic for Binary Classifiers and Their Explanation. In Pietro Baroni, Christoph Benzmüller, and Yi N. Wang, editors, *Logic and Argumentation*, pages 302–321, Cham, 2021. Springer International Publishing.
- [LLRS22] Xinghan Liu, Emiliano Lorini, Antonino Rotolo, and Giovanni Sartor. Modelling and Explaining Legal Case-Based Reasoners Through Classifiers. December 2022.
- [MCG25] Yoann Morello, Agata Ciabattoni, and Morgan Gray. The Result Model under Inconsistent Knowledge: Theory and Experiments. 2025.
- [OBP23] Daphne Odekerken, Floris Bex, and Henry Prakken. Precedent-Based Reasoning with Incomplete Cases. In Giovanni Sileno, Jerry Spanakis, and Gijs

Van Dijck, editors, *Frontiers in Artificial Intelligence and Applications*. IOS Press, December 2023.

- [Pra21] Henry Prakken. A formal analysis of some factor- and precedent-based accounts of precedential constraint. *Artificial Intelligence and Law*, 29(4):559–585, December 2021.
- [RA87] E. L. Rissland and K. D. Ashley. A case-based system for trade secrets law. In *Proceedings of the first international conference on Artificial intelligence and law - ICAIL '87*, pages 60–66, Boston, Massachusetts, United States, 1987. ACM Press.
- [RD95] Edwina L. Rissland and Jody J. Daniels. A hybrid CBR-IR approach to legal information retrieval. In *Proceedings of the fifth international conference on Artificial intelligence and law - ICAIL '95*, pages 52–61, College Park, Maryland, United States, 1995. ACM Press.
- [RD96] Edwina L. Rissland and Jody J. Daniels. The synergistic application of CBR to IR. *Artif. Intell. Rev.*, 10(5-6):441–475, October 1996.
- [Rig18] Adam Rigoni. Representing dimensions within the reason model of precedent. *Artificial Intelligence and Law*, 26(1):1–22, March 2018.
- [RS91] Edwina L. Rissland and David B. Skalak. CABARET: rule interpretation in a hybrid architecture. *International Journal of Man-Machine Studies*, 34(6):839–887, June 1991.
- [RSF96] Edwina L. Rissland, David B. Skalak, and M. Timur Friedman. BankXX: Supporting legal arguments through heuristic retrieval. *Artificial Intelligence and Law*, 4(1):1–71, March 1996.
- [RSF97] Edwina L. Rissland, David B. Skalak, and M. Timur Friedman. Evaluating a Legal Argument Program: The BankXX Experiments. *Artificial Intelligence and Law*, 5(1):1–74, March 1997.
- [Thö24] Henri Thölke. Towards a generalized reason model. Master’s thesis, Technische Universität Wien, 2024.