A Language-Independent Framework for Reasoning about Preferences for Declarative Problem Solving

by

Alireza Ensan

M.Sc., University of New Brunswick, 2012
B.Sc., Ferdowsi University of Mashhad, 2008

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy
in the
School of Computing Science
Faculty of Applied Science

© Alireza Ensan 2019
SIMON FRASER UNIVERSITY
Spring 2019

Copyright in this work rests with the author. Please ensure that any reproduction or re-use is done in accordance with the relevant national copyright legislation.
Approval

Name: Alireza Ensan

Degree: Doctor of Philosophy (Computer Science)

Title: A Language-Independent Framework for Reasoning about Preferences for Declarative Problem Solving

Examining Committee: Chair: Andrei Bulatov
Professor

Eugenia Ternovska
Senior Supervisor
Associate Professor

David Mitchell
Supervisor
Associate Professor

James Delgrande
Internal Examiner
Professor

Pedro Cabalar
External Examiner
Associate Professor
Department of Computer Science, University of Corunna

Date Defended: April 10, 2019
Automated decision making often requires solving difficult (e.g., NP-hard) problems. In many AI applications (e.g., planning, formal verification, robotics, etc.), users can assist decision making by specifying their preferences over some domain of interest. We take a language-independent, i.e., model-theoretic approach to preferences. Decision and search problems are abstractly formalized as Model Expansion, that is, the logical task of expanding an input structure to satisfy a specification. The specification can be in any language with a model-theoretic semantics, e.g. Answer Set Programming, Constraint Programming, etc. Preferences are defined as binary relations on (sets of) partial models. We prove that introducing preferences even in the simplest formulation leads to a significant rise of the computational complexity.

We develop a model-theoretic approach of combining specifications written in multiple languages with preferences. We demonstrate relationships with several preference frameworks in the literature, such as CP-nets, Answer Set Optimization, Preference-based planning, etc.

We propose an algorithm that solves Model Expansion problems with preferences using abstract solvers empowered by propagators. A solver provides symbolic explanations for rejecting and accepting models, and follows a preferred computation path to prune the search space.

Finally, we develop a preference-based approach to finding approximate solutions of over-constrained problems. The specifications of such problems may have several parts, potentially written in different languages, that may be not satisfiable in combination. We demonstrate how an approximate solution can be constructed, based on selecting models that are the closest to the models of more preferable specifications.

**Keywords:** Preference, Model Expansion, Modular Systems
Dedication

To my lovely wife, Mahdis, and my parents
Acknowledgements

First and foremost, I would like to express my sincere gratitude to my senior supervisor, Dr. Eugenia Ternovska, for the constant guidance and support she provided to me throughout my studies. None of this would be possible without her encouragement, patience, kindness, and help. Her brilliance and insightful research vision will be always the source of inspiration for me. I am grateful for the opportunity she provided to me to learn from her different scientific and communication skills to become a better scientist.

I would also like to thank my supervisor, Dr. David Mitchell, who guided me with his invaluable suggestions throughout my PhD studies. Moreover, I would like to thank the internal examiner, Dr. James Delgrande, and the external examiner, Dr. Pedro Cabalar, for their very helpful input and comments.

Finally, many thanks to my beloved wife, Mahdis, who supported me through all the ups and downs in these years with her love, passion, and encouragement, and to my parents for their never ending support.
# Table of Contents

 Approval ii  
 Abstract iii  
 Dedication iv  
 Acknowledgements v  
 Table of Contents vi  
 List of Tables ix  
 List of Figures x  

## 1 Introduction 1  
 1.1 Model Expansion and Modular Systems ........................................... 3  
 1.2 Preferences Handling ................................................................. 4  
 1.3 Thesis Organization ................................................................. 7  

## 2 Background 9  
 2.1 Finite Model Theory and Descriptive Complexity ............................... 9  
 2.1.1 Model Checking and Complexity ............................................... 9  
 2.1.2 Descriptive Complexity ......................................................... 10  
 2.2 Some Remarks on Complexity Classes ............................................. 11  
 2.3 Model Expansion ................................................................. 12  

## 3 Prioritized Model Expansion 14  
 3.1 Introduction ................................................................................. 14  
 3.2 Model Expansion with Preferences ................................................. 16  
 3.2.1 Preference Expression ............................................................. 16  
 3.2.2 Prioritized Model Expansion ..................................................... 18  
 3.2.3 Conditional Preferences ......................................................... 25  
 3.3 The Relation Between Prioritized Model Expansion and Other Preference Formalisms ................................................................. 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.1</td>
<td>Preference-based SAT</td>
<td>26</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Logic Programs with Preferences</td>
<td>28</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Answer Set Optimization</td>
<td>30</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Some More Examples</td>
<td>32</td>
</tr>
<tr>
<td>3.4</td>
<td>Conclusion</td>
<td>32</td>
</tr>
<tr>
<td>4</td>
<td>Modular System with Preferences</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>33</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Contributions</td>
<td>35</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Novelty</td>
<td>36</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Chapter Structure</td>
<td>36</td>
</tr>
<tr>
<td>4.2</td>
<td>Algebra of Modular Systems</td>
<td>36</td>
</tr>
<tr>
<td>4.3</td>
<td>(\mathcal{P})-MS: Preference-based Modular Systems</td>
<td>38</td>
</tr>
<tr>
<td>4.4</td>
<td>Relationship of (\mathcal{P})-MS to other Preference Formalisms</td>
<td>47</td>
</tr>
<tr>
<td>4.4.1</td>
<td>CP-Nets</td>
<td>47</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Answer Set Optimization</td>
<td>50</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Preference-based Planning</td>
<td>52</td>
</tr>
<tr>
<td>4.5</td>
<td>Conclusion and Future Work</td>
<td>54</td>
</tr>
<tr>
<td>5</td>
<td>Solving Prioritized Model Expansion Problems</td>
<td>55</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>55</td>
</tr>
<tr>
<td>5.2</td>
<td>Preference Relation Semantics</td>
<td>56</td>
</tr>
<tr>
<td>5.3</td>
<td>Naive Algorithm</td>
<td>57</td>
</tr>
<tr>
<td>5.4</td>
<td>Solvers for Model Expansion Problems and Modular Systems</td>
<td>60</td>
</tr>
<tr>
<td>5.5</td>
<td>Preference Solvers for Prioritized Model Expansion Problems</td>
<td>63</td>
</tr>
<tr>
<td>5.5.1</td>
<td>General Properties of Preference Solvers</td>
<td>63</td>
</tr>
<tr>
<td>5.5.2</td>
<td>An algorithm for Preference Solvers</td>
<td>65</td>
</tr>
<tr>
<td>5.6</td>
<td>Constructing Preference Solvers from Solvers of Modular Systems</td>
<td>66</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Prioritized Model Expansion Problems as Modular Systems</td>
<td>66</td>
</tr>
<tr>
<td>5.6.2</td>
<td>Modular Systems as Operators</td>
<td>69</td>
</tr>
<tr>
<td>5.6.3</td>
<td>Characterizing Prioritized Model Expansion in Dynamic Algebra</td>
<td>71</td>
</tr>
<tr>
<td>5.7</td>
<td>Conclusion</td>
<td>73</td>
</tr>
<tr>
<td>6</td>
<td>Distance-Optimal Approximate Solutions in Compound Model Expansion Problems</td>
<td>74</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>74</td>
</tr>
<tr>
<td>6.2</td>
<td>Compound Model Expansion</td>
<td>77</td>
</tr>
<tr>
<td>6.3</td>
<td>Distance-Optimal Solutions</td>
<td>78</td>
</tr>
<tr>
<td>6.4</td>
<td>Query Evaluation for Weighted Compound Model Expansion Problems</td>
<td>80</td>
</tr>
<tr>
<td>6.5</td>
<td>Distance-Optimal Solutions in Prioritized Model Expansion Problems</td>
<td>83</td>
</tr>
<tr>
<td>6.6</td>
<td>Discussion</td>
<td>85</td>
</tr>
</tbody>
</table>
List of Tables

Table 2.1  Complexity of Model Checking ................................. 10
Table 2.2  Complexity of Model Expansion ............................... 12
Table 4.1  Outcomes in Car Dealership Example ......................... 49
Table 7.1  Dessert Options .................................................. 100
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A Feature Model for an Online Shop Domain</td>
<td>2</td>
</tr>
<tr>
<td>3.1</td>
<td>A Prioritized Graph Three Colouring</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>Two Possible Hamiltonian Paths for Graph $G$ in Example 5</td>
<td>24</td>
</tr>
<tr>
<td>3.3</td>
<td>Conditional Graph Three-Colouring</td>
<td>26</td>
</tr>
<tr>
<td>4.1</td>
<td>Logistics Service Provider $M_{LSP} = (M_K \triangleright M_{TSP})[B = B']$</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>CP-net Example: Car Dealership</td>
<td>50</td>
</tr>
<tr>
<td>5.1</td>
<td>Preference Semantics Subsumption Hierarchy</td>
<td>58</td>
</tr>
<tr>
<td>5.2</td>
<td>A Preferred-Goal State Achieved by a Preference Solver for Graph Three-Colouring</td>
<td>62</td>
</tr>
<tr>
<td>5.3</td>
<td>A Procedure to Find Optimal Expansions</td>
<td>69</td>
</tr>
<tr>
<td>5.4</td>
<td>A Compound Action for Solving Prioritized Model Expansion Problems</td>
<td>72</td>
</tr>
<tr>
<td>6.1</td>
<td>Equipment Size Selection in Mining Engineering</td>
<td>76</td>
</tr>
<tr>
<td>6.2</td>
<td>Approximate Graph Colouring</td>
<td>85</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The last three decades have witnessed significant growth when it comes to the application of Artificial Intelligence (AI) technologies in software development, health care, manufacturing, transportation, and so on. AI tasks (e.g., reasoning, verification, theorem proving, etc.) often consist of solving several computational problems. The declarative programming paradigm plays a key role in problem solving in many AI tasks. Declarative languages specify what an algorithm for solving a problem should achieve without specifying how exactly to do it. Declarative programming has been employed for a variety of applications, such as database query languages, functional programming, multi-agent systems, multi-component problems, and so on. The popularity of declarative approaches is due to reducing the mutability of the implementation, scalability, and maintainability. Also, declarative approaches provide easier problem encoding for non-professional users. A programmer is only required to describe a problem in a particular declarative language and does not need to worry about using different heuristics, search techniques, or, generally, the control flow of the algorithm that is to be implemented. For example, SQL is a declarative language for querying relational databases based on logic that is being used by millions of non-specialist users.

Declarative problem representation and solving is one of the topics of interest in AI in general and in the Knowledge Representation (KR) community in particular. Logic, which is at the center of KR frameworks (which usually aim at modeling human thoughts [88]), includes a language in which formulas are written, a set of formulas that are called axioms and a set of rules of inference. A theorem is an axiom or can be derived from another theorem using the rules of inference. A model is an interpretation that satisfies all axioms. While there is no general consensus on its definition, computational logic provides a relation between the computation and logical reasoning [23] aimed at declarative problem solving. It originates from the field of automated theorem proving that started since 1950s and 1960s which sought to automate the proof of mathematical theorems. It was in the beginning of 1970s that computational logic found its way into computer science applications, particularly in declarative problem solving in AI and deductive databases, with the purpose of merging together different declarative approaches in databases, programming, and AI [88]. Relational database management is almost a $37 billion (per year) industry, based on logic. But now we need to address computationally harder tasks than answering relational databases queries. The fol-
Example 1. Software Product Line is a paradigm in software development (in the requirement engineering phase) that characterizes a family of software products that share a number of features. Each feature can represent a requirement, a qualitative measure (e.g., security of a transaction), or a functionality (e.g., measuring the similarity between two text documents for a text editor). A feature model is a graphical representation (with a tree structure) of a software product line that illustrates possible configurations of software products. There are structural constraints that can be represented by a feature model. For example, Figure 1.1, which is inspired by [60], shows a product line for an online store. Each node represents a feature and arrows with white circles represent optional child features while arrows with black circles represent mandatory child features. Each online store must have order approval, shipment, and payment features, but having customer verification is optional. Also, filled angles between two arrows represent logical OR and unfilled angles represent AND. For instance, the payment method options are debit, credit, or check. A set of integrity constraints are also defined for each feature model. For example, integrity constraints can be introduced so that for a credit card payment, an ebill is always issued and every new customer cannot use check for payment. The problem of determining whether there is a configuration (a selection of features) that satisfies all structural and integrity constraints can be reduced to a SAT problem [97]. This is a classical use of declarative approaches for formal verification of software programs (in the requirement engineering stage). For an incoherent feature model (i.e., a feature model with no configuration), a method has been proposed in [104] to transform a feature model into a Description Logic (DL) knowledge base. Then, by generating the Minimal Incoherence-Preserving Sub-TBoxes (MIPS), based on traversing the Reiter’s hitting set tree [108] for MIPS, the incoherency in the feature model is explained and a possible solution is proposed to resolve the incoherency.

Figure 1.1: A Feature Model for an Online Shop Domain

The computational complexity of problems representable in declarative frameworks has been one of the main topics of studies in AI. Another important topic is the complexity of describing problems (e.g., describing a planning problem by Answer Set Programming (ASP) rules). In recent
years, some research attention has been shifted toward handling the complexity of representing problems in declarative ways for the following reasons.

First, in the last two decades, there has been a significant development in declarative solvers, especially SAT solvers that are able to efficiently solve real-world problems with millions of variables. Due to the significant improvement in modern solvers, solving computationally hard (e.g., NP-hard) problems has become practical in many applications (e.g., software verification, planning, optimization, etc). Therefore, dealing with hard problems may not always be as big of an issue as it was twenty years ago.

Second, with regard to computational complexity, there is a school of thought led by Moshe Vardi [133], which is finding its way into the mainstream, that says it is necessary to introduce a new complexity theory that is able to distinguish problems in the same complexity class from a practical point of view. For example, there are SAT instances with fewer numbers of variables that cannot be solved by current available processors and memories, while there are SAT problems with much larger numbers of variables that can be solved very efficiently. Vardi argued in [133] that even solving whether P=NP, which is one of the most prominent unsolved problems in theoretical computer science, does not necessarily result in practical benefits. For example, if P=NP and an NP-hard problem can be solved by a polynomial algorithm in $n^{2300}$, this is not practically feasible by the extant hardware technologies for large inputs. On the other hand, proving that NP $\neq$ P does not necessarily mean that an NP-hard problem cannot be solved efficiently. For instance, imagine an NP-hard problem that is solved by an algorithm with running time $n^{\log \log \log n}$. Consequently, the current complexity theories are not necessarily helpful in describing some real-world problems.

We shall approach declarative problem solving in a way that overcomes representational hurdles while, at the same time, keeping an eye on the computational complexity aspects.

One of the main difficulties in declarative problem solving is dealing with multi-component problems where each sub-problem can be specified in a possibly different declarative language. These problems are encountered in many real-world applications, such as web-service composition in which each service, which is viewed as a black-box, is a software artifact, multi-agent systems where each agent may act independently, design optimization of mine sites and equipment in mining industry, software product lines with constructs from different programming languages, etc. For instance, in Example 1, credit card transactions need a database implementation with a number of security protocols that must work with other components of the online store (e.g., credit history). We take a model-theoretic approach to declarative problem solving that paves the way for combining declarative frameworks in a uniform way.

1.1 Model Expansion and Modular Systems

Model Expansion characterizes search and decision problems as expanding a problem instance (an input structure) over an instance vocabulary $\sigma$ to a solution (an expansion structure) over a vocabulary $\tau$, where $\sigma \subseteq \tau$, that satisfies a formula $\psi$ in a logic $\mathcal{L}$ with a model-theoretic semantics.
Formula $\psi$ characterizes the specifications of a problem. Each problem instance is completely separated from the problem specifications, which is an advantage specially for non-professional users. The problem specifications are written once and in the future it is only necessary to specify each instance of the problem.

Model Expansion underlies all main declarative approaches in AI, such as SAT, ASP, and CSP that have different constructs and syntactical limitations such that describing complex problems may become burdensome [127]. However, in all of them, a search problem is solved for a given problem instance. Model Expansion can be seen as a generalization of searching solutions of a problem instance.

A problem characterized as Model Expansion can also be viewed as a class $C$ of structures that are models of $\psi$ in $\mathcal{L}$. This view of Model Expansion, which is independent from the syntax and semantics of a specific formal language, is the basis for defining modular systems. The modular system framework is a declarative framework for combining modules (Model Expansion) in different declarative languages model-theoretically using several operations defined as the algebra of modular systems [127].

1.2 Preferences Handling

With exponentially growing sizes of inputs (problem instances) for search problems in AI tasks (e.g., flight planning with a large number of impacting factors in an airport or finding similar matches in a social network) and the possibility of finding a multitude of solutions, it is common to prioritize solutions based on the goals and preferences of decision makers and stakeholders (e.g., customization of different features of a software product for a specific user).

Decision theory [106], having roots in economics, studies the task of finding the best results when a decision maker deals with multiple options. The connection between decision theory and AI became the interest of several researchers in the late 1980s. One of the primary motivations was handling planning with preferences for goals, certain actions, or some states of plans. Modeling preferences gained attention in a wide range of AI applications, from database systems to recommender systems, web-service composition, planning, and so on. For instance, in Example 1, the goal could be to maximize the higher sale, which leads to selecting FedEx (because it is faster) and easier customer verification (e.g., customers verification based on purchase history instead of credit rating, etc). Handling preferences has been an important part of declarative frameworks and has been widely studied in the literature. Examples include [120], [19], [18], [84], [125], [117], [48], [53], [32], [33], [46], and [5].

Proposals found in these examples have similarities and differences. Delgrande et.al [48] introduced a number of criteria for comparing preference-based non-monotonic approaches. Some of those criteria can be used for classifying preference frameworks. A declarative language to which preferences are added is usually called a host system [48]. For example, the host systems of [36], [33], and [113] are logic programs with answer set semantics. Also, [46] is a preference framework
for default logic, [122] is a preference-based argumentation framework, and [120] is a standard planning framework with preferences. In some proposals, preference relations are represented as external entities that are added to a host system. For example, in [29], [36], [30], and [73], preferences are represented at the meta-level as binary relations over atoms appearing in the head of rules. Contrarily, preferences may be translated into elements of a particular language, such as in [47] or [46]. These approaches in which preferences are represented as objects of the host system’s language are called object-level. Also, the type of elements that are ordered with respect to a preference relation is an important factor. For instance, in [47], ordering is defined on rules and in [46] and [34], on defaults. In contrast, in [113], a preference relation is defined as an ordering on atoms. Another factor worth considering is the type of outputs a preference-based declarative approach generates. For example, in [36], an Answer Set Optimization (ASO) program defines a preorder over the answer sets of a logic program based on a vector of satisfaction degrees for each answer set. Answer sets with higher satisfaction degrees are more preferred. However, in [47], only preferred answer sets are produced and there is no preference over outputs. Moreover, we should consider whether a host system with preferences retains its original properties. For example, we should consider whether extensions (models) of a prioritized default theory are a subset of the extensions of the default theory without preferences. In addition to all of the above, we can study whether the complexity of reasoning (skeptical or credulous) with regard to the preferred solutions stays in the same class as the complexity of the host system (complexity of finding solutions of a problem described in a host system) or rises to a higher complexity class.

Existing declarative languages with preferences in the literature have some drawbacks that restrain them from dealing with complex real-world problems because:

First, they are mainly host dependent, which means that they are defined for a specific host language. For instance, preferences are represented as a binary relation over default rules [46] or as a set of context dependent preference rules with different rankings for disjunctive ASP programs [73]. This dependency on the syntax and semantics of a declarative language limits users to only encoding problems for solvers of that language. The task of describing a problem with preferences may become difficult when the language has some syntactical limitations (e.g., no arithmetic aggregate operators), especially for a user with no skill or related knowledge of that language.

Second, preference solvers, which find optimal solutions (the most preferred solutions) using existing generic solvers, are usually combinations of two solvers of a host declarative language [32]. One solver generates a candidate solution and another one (i.e., the tester) determines whether the candidate solution is preferable. The second solver requires axiomatizing the notion of preferred models in the solver’s (host) language, which is restrained by the constructs of the host language. The definition of a preferred model is very different in different declarative frameworks. For instance, in [36], the preferred answer set is the one with the greatest associated satisfaction vector. In [47], a preferable answer set is the result of an earlier application of more preferable rules. Also, in some prominent declarative approaches, such as SAT and ASP, specifications of a problem are not clearly separated from the problem instance. Solvers can speed up the search by cutting down
the search space based on the structure of a problem when there is a clear separation between an instance and specifications of a problem [127].

Third, real-world AI tasks often include solving modular problems where each sub-problem can be described in a different language and has its own solver. Combining solvers for such problems becomes burdensome for unskilled users, particularly, when preferences of a sub-problem are provided in a host system whose language is different than the declarative language in which the sub-problem is described. For instance, in Example 1, in order to increase the profit margin, looser rules for the verification of current customers are applied. Customer verification includes both credit score and purchase history. Assume credit score is calculated based on a set of rules described by an ASP program. Assume a customer with a longer purchase history is preferred to a customer with a longer period of positive balance. Suppose this preference is reflected as a set of conditional preference rules in the Answer Set Optimization (ASO) framework [36]. Suppose that there is a set of constraints that characterizes a current active customer (e.g., a minimum amount of purchases in the last three months), which are specified as a Constraint Sanctification Problem (CSP) [130]. Let preferences be provided by a CP-net, which is a graphical representation of conditional preferences [20], such that it is preferable to consider a customer to be active based on the maximum amount of purchases made rather than the time of the last purchase. Constructing a software product from the feature model in Figure 1.1 requires combining a number of preference solvers, each of which may be for a different declarative language.

We shall address these issues by extending Model Expansion to represent preferences, which is called Prioritized Model Expansion, and introducing preference-based modular systems, which is a framework for combining declarative languages with preferences. We take the following steps to this end:

First, for a Model Expansion problem, we represent preferences as binary relations over atoms. We define a binary (preference) relation over solutions (structures) by lifting the relation over atoms based on a certain preference semantics (lifting method). In fact, a lifting method characterizes what it means to be a preferred model. We introduce a number of lifting methods and show the flexibility of our framework, which allows a user to specify her own way of lifting as long as it meets the requirement that there exists a polynomial algorithm that, from two solutions, finds the preferable one. This requirement is essential in the complexity of several reasoning tasks associated with Prioritized Model Expansion problems. We show that introducing preferences leads to a rise in the complexity that is, intuitively, compatible with the expectation that optimization problems are, generally, harder. Prioritized Model Expansion is host independent (i.e., can be defined for any logic $\mathcal{L}$ with a model-theoretic semantics). Our proposal can also be placed in the category of describing preferences as external entities added to a host system. It is worth noting that object-level approaches are host dependent. Therefore, we do not focus on them in this thesis.

Second, we propose preference-based modular systems for combining Model Expansion problems with preferences. We introduce an algebra for preference relations over structures that corre-
responds to the algebra of modular systems. We also study the relation between our proposal and a number of other preference-based declarative frameworks in the literature.

Third, we propose an algorithm for solving Prioritized Model Expansion problems. After this, we formally define a preference solver and enumerate a number of properties that each preference solver must have. We propose a formal definition for solvers of modular systems and show that a preference solver can be constructed from an existing solver of modular systems. By using solvers of modular systems, a user is provided with options for encoding each constraint of a problem or the axiomatization of a lifting method in a declarative language that is more convenient (i.e., the language has some constructs that make the axiomatization of a problem’s constraint simpler).

Finally, we propose a method for dealing with combinations of Model Expansion problems (compound Model Expansion problems) with no solutions. We assume that a user prioritizes constraints of a compound problem (by assigning different weights). Inspired by model-based belief merging [86], we compute approximate solutions that are closer to the solutions of more important problems.

Kießlingin [84] argued that a viable preference framework for database systems should have the following properties:

- intuitive semantics, which is the capability of modelling quantitative and qualitative preferences compatible with human intuition,
- extensible and a constructive structure, which means that complex preferences can be inductively constructed from simple preferences,
- concise mathematical foundation, which means the framework should be adaptive to the intuitive semantics,
- the ability to handle conflicts that may arise due to the composition of complex preferences, and
- the ability to provide a declarative preference query language.

In our opinion, this set of properties should be a road map for any preference handling approach that tackles real world problems. Our proposal adopts these properties in an intuitive way such that preferences are qualitatively expressed by an ordering on atoms or partial models. The representation of preferences can simply be extended to more complex preference statements (e.g., conditional preferences can be represented in Prioritized Model Expansion problems or using logical operator, such as AND, OR, etc). We also deal with conflicting preferences by introducing the notion of a meta preference that prioritizes preferences. Finally, we provide a thorough mathematical analysis of the impact of introducing preferences on the computational complexity of query evaluation.

1.3 Thesis Organization

This thesis is organized as follows: First, in Chapter 2, we review some theoretical background behind our work, such as computational complexity, descriptive complexity, and Model Expans-
sion. Then, in Chapter 3, we introduce Prioritized Model Expansion and study the computational complexity of some associated problems and the relation between our proposal and several other preference-based frameworks. In Chapter 4, we define preference-based modular systems and introduce an algebra for combining preferences over structures. In Chapter 5, we propose an algorithm for solving Prioritized Model Expansion problems and characterize general properties of preference solvers. We prove that a preference solver can be constructed by employing solvers of modular systems. In Chapter 6, for a combination of Model Expansion problems with no solutions, we use preferences of a user over the constraints and find approximate solutions, which are closer to models that satisfy more important constraints. We conclude, in Chapter 7, with a literature review of different preference frameworks and their relation to our proposal, and a discussion of possible future work.
Chapter 2

Background

2.1 Finite Model Theory and Descriptive Complexity

A vocabulary $\tau$ is a set of non-logical relation symbols $R_i$ with associated arity $k_i$ and constant symbols $C_j$. A $\tau$-structure $\mathcal{A} = (A, R_1^A, ..., R_n^A, C_1^A, ..., C_m^A)$ is a tuple where $A$ is a domain, for each $R_i \in \tau$, $R_i^A \subseteq A^{k_i}$, and for each $C_j \in \tau$, $C_j^A \in A$. $R_i^A$ is called the interpretation of $R_i$ in $\mathcal{A}$. Similarly, $C_j^A$ is the interpretation of constant $C_j$ in $\mathcal{A}$. For a formula $\psi$ in a logic $\mathcal{L}$, $\text{vocab}(\psi)$ denotes the set of non-logical vocabulary symbols appearing in $\psi$.

Model theory [40] is the study of classes of structures and their properties. A structure is finite if its domain is finite. Finite model theory [72] studies properties of classes of finite structures.

2.1.1 Model Checking and Complexity

One of the topics of interest in finite model theory and particularly in descriptive complexity is to establish a relation between computational complexity and definability in a logic. Complexity is often associated with a logic based on the complexity of reasoning tasks such as satisfiability and model checking. However, satisfiability is undecidable in many logics, e.g., first-order logic. It is more common to describe the complexity of a logic $\mathcal{L}$ by the complexity of model checking in $\mathcal{L}$ (it is also called query evaluation). Given a finite structure $\mathcal{A}$ and a sentence $\phi$ in $\mathcal{L}$, the combined complexity of $\mathcal{L}$ is defined as the complexity of deciding whether $\mathcal{A} \models \phi$. The data complexity of $\mathcal{L}$ denotes the complexity of deciding whether $\mathcal{A} \models \phi$ holds for a fixed sentence $\phi$ in $\mathcal{L}$ and given a finite structure $\mathcal{A}$. Since $\phi$ is fixed, the syntax of $\phi$ does not impact the data complexity of model checking. Similarly, the expression complexity of $\mathcal{L}$ is the complexity of deciding whether $\mathcal{A} \models \phi$ for a fixed finite structure $\mathcal{A}$ and a given sentence $\phi$ in $\mathcal{L}$.

The data and expression complexity of some logics are given in Table 2.1. In this table, FO denotes first-order logic, FP is first-order logic with least fixed point operator, and $\exists \text{SO}$ denotes existential second-order logic. As can be observed from Table 2.1, there is an exponential gap between the data and expression complexities. Also, by considering a constant upper bound for the
Table 2.1: Complexity of Model Checking

<table>
<thead>
<tr>
<th>Logic</th>
<th>Data Complexity</th>
<th>Query Complexity</th>
<th>Combined Complexity</th>
<th>VC Query Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO</td>
<td>LOGSPACE</td>
<td>PSPACE</td>
<td>PSPACE</td>
<td>PTIME</td>
</tr>
<tr>
<td>FP</td>
<td>PTIME</td>
<td>EXPTIME</td>
<td>EXPTIME</td>
<td>PTIME</td>
</tr>
<tr>
<td>∃SO</td>
<td>NP</td>
<td>NEXPTIME</td>
<td>NEXPTIME</td>
<td>NP</td>
</tr>
</tbody>
</table>

number of variables in queries, which is called variable confinement (VC), the complexity reduces substantially.

The relation between the complexity classes in Table 2.1 is as follows: LOGSPACE ⊆ PTIME ⊆ NP ⊆ PSPACE ⊆ EXPTIME ⊆ NEXPTIME, LOGSPACE ∉ PSPACE, PTIME ∉ EXPTIME, and NP ∉ NEXPTIME [45].

In order to connect finite model theory and complexity, structures are translated into inputs of a Turing machine. Inputs of Turing machines are usually described as words over a fixed finite alphabet. Each structure can be viewed as a word. Loosely speaking, for a \( \tau \)-structure \( A \), consider an order \( > \) on the domain of \( A \). For each \( k \)-ary predicate \( R \in \tau \), the relation \( R^A \) (assume \( k \) is the maximum arity of relations in \( \tau \)) can be translated into \( l_1, \ldots, l_n \) where \( l_j = 1 \) if the \( j \)th tuple (among \( n^k \) tuples) is in \( R^A \) and \( l_j = 0 \) otherwise. Thus, there is a one-to-one correspondence between classes of structures and languages. A class of structures \( C \) can be encoded as a language \( L \) and vice versa. Therefore, in the context of Turing machines, the problem of model checking can be viewed as deciding whether a word belongs to a language.

In the following subsection, we take a look at capturing a complexity class by a logic, which is the main subject of study in the area of descriptive complexity.

### 2.1.2 Descriptive Complexity

Query evaluation is the basis for connecting logic and computation. Let \( \tau \) and \( \tau' \) be two vocabularies of symbols. A query \( Q \) is a map from the set of \( \tau \)-structures to \( \tau' \)-structures where for each \( \tau \)-structure \( A \), \( Q(A) \) has a domain bounded by a polynomial factor of the domain of \( A \) [79]. In first-order logic, based on Codd’s characterization [42], a non-Boolean first-order query \( Q \) can be described as a formula \( \phi(x_1, \ldots, x_k) \) such that \( Q(A) = \{ (\nu(x_1), \ldots, \nu(x_n)) | A \models \nu \phi(x_1, \ldots, x_k) \} \) where \( \nu \) is a valuation function that assigns an element in the domain of \( A \) to \( x_i \). In other words, based on [79], \( Q(A) \) can be viewed as a \( \tau' \)-structure such that \( R^{Q(A)} = \{ (a_1, \ldots, a_k) \in A^k | A \models \phi(a_1, \ldots, a_k) \} \) where \( \text{vocab}(\phi) \subseteq \tau \) and \( \tau' = \{ R \} \), where \( R \) is an \( k \)-ary predicate.

A boolean query is defined as a mapping from \( \tau \)-structures to \{0,1\} or sometimes as a set of \( \tau \)-structures \( B \) such that \( Q(B) = 1 \). A sentence \( \varphi \) where \( \text{vocab}(\varphi) = \tau \) is related to a boolean query \( Q \) such that for all \( \tau \)-structures \( B \), \( Q(B) = 1 \) if and only if \( B \models \varphi \). A boolean query over a class \( C \) of structures sometimes is called a property over \( C \). For example, a boolean query \( Q \) asks whether a given finite graph \( G \) belongs to the class of Hamiltonian graphs.

For a boolean query (property) \( Q \) over a class \( C \) of \( \tau \)-structures (i.e., \( Q \subseteq C \) when \( Q \) is considered to be a class of structures), we say that \( Q \) is definable (expressible) in logic \( \mathcal{L} \) over \( C \) if there is
a sentence $\varphi \in \mathcal{L}$ such that for every $\mathcal{A} \in \mathcal{C}$, $\mathcal{A} \models \varphi$ if and only if $\mathcal{A} \in Q$. A complexity class $X$ can be represented by the set of all $\tau$-properties that are computable in $X$.

The main definition in descriptive complexity that connects definability in a logic and complexity is: Complexity class $X$ is captured by a logic $\mathcal{L}$ (over $\mathcal{C}$) if and only if for all $\tau$-properties $Q$ computable in $X$, $Q$ is definable in $\mathcal{L}$. For example, based on Fagin’s theorem [66], existential second-order logic captures NP over the class of finite structures. Also, least fixed point logic captures P over the class of ordered finite structures [132].

### 2.2 Some Remarks on Complexity Classes

Let us define a relation $R(x, y_1, \ldots, y_n)$ for a word $x$. Relation $R$ is polynomial if there is a nondeterministic Turing machine $M$ that determines whether $R(x, y_1, \ldots, y_n)$ holds in polynomial time of the size of $x$. A language $\mathcal{L}$ is in NP if and only if for all $\tau$-properties $Q$ computable in $X$, $Q$ holds if $x \in \mathcal{L}$ if and only if $\exists y R(x, y)$ holds. For example, SAT is in NP where $x$ is a word that represents a boolean formula and $y$ is an assignment to $x$.

**Definition 1.** Oracle machine $M_L$ is a Turing machine $M$ augmented by an oracle tape that can decide whether string $x$ on the oracle tape belongs to a language $L$.

Let $X$ be a complexity class.

**Notation 1.** $P^X$ is the class of languages (complexity class) that can be computed in polynomial time by a deterministic Turing machine with an oracle in $X$. Also, $NP^X$ is the class of languages that can be computed in polynomial time by a nondeterministic Turing machine using an oracle in $X$.

**Notation 2.** $co-X$ is the complexity class of decision problems whose complements are in $X$.

For complexity classes beyond NP, consider the following: $\Sigma^P_k$ is the class of languages (or a complexity class) such that for every $\mathcal{L} \in \Sigma^P_k$, there is a polynomial relation $R$ such that $x \in \mathcal{L}$ if and only if $\exists y_1 \forall y_2 \ldots, \exists y_k R(x, y_1, \ldots, y_k)$ holds if $k$ is odd and $\exists y_1 \forall y_2 \ldots, \forall y_k R(x, y_1, \ldots, y_k)$ holds otherwise. The complexity class PH is defined as $PH = \bigcup_k \Sigma^P_k$. Also, $\Sigma^P_k$ is called the $k$th level of the Polynomial Hierarchy (PH). $NP^{\Sigma^P_{k-1}}$ is recursively defined as a complexity class or a family of languages that can be decided by a polynomial nondeterministic Turing machine with access to an oracle that can decide languages in $\Sigma^P_{k-1}$. Therefore, $P = \Sigma^P_0 = \Pi^P_0 = \Delta^P_0$ and $\Sigma^P_{k+1} = NP^{\Sigma^P_k}$, $\Delta^P_{k+1} = P^{\Delta^P_k}$, and $\Pi^P_{k+1} = coNP^{\Sigma^P_k}$ for $k > 0$. Also, NP, coNP $\subseteq \Sigma^P_2 \cap \Pi^P_2$ [126]. Complete problems in the $k$th level of the Polynomial Hierarchy have been widely studied [115]. For example, for an arbitrary Boolean formula $\phi$ and Boolean variables $x_1, \ldots, x_k$ appearing in $\phi$, deciding whether $\exists x_1, \forall x_2, \ldots, \exists x_k \phi(x_1, \ldots, x_k)$ is valid, which is also called True Quantified Boolean Formula (TQBF) problem, is $\Sigma^P_k$-complete. It is worth mentioning that any problem in PH can be encoded as an TQBF $\exists x_1, \forall x_2, \ldots, \exists x_n \phi(x_1, \ldots, x_n)$ with unbounded $n$ number of alternations.
of quantifiers, which is the canonical PSPACE-complete problem. As a result, $PH \subseteq PSPACE$. If $PH=PSpace$, the Polynomial Hierarchy collapses. Also, there is no language that belongs to PH-complete unless PH collapses to the $k$th level for some $k$. If there is a PH-complete language, say $\mathcal{L}$, it must belong to some $\Sigma^P_k$ class and all other languages of PH can be reduced to $\mathcal{L}$, that leads to the collapse of the hierarchy [126].

### 2.3 Model Expansion

Model Expansion (MX) is the logical task of expanding a structure to satisfy a formula in a logic $\mathcal{L}$ [100] that has a model-theoretic semantics, such as a logic with Tarski semantics (e.g., first-order logic) or Kripke semantics (e.g., modal logic). Note that Kripke semantics can be encoded in Tarski’s structures with an auxiliary predicate that encodes the reachability relation on possible worlds (for more details see guarded fragments [77]).

The definition of expansion is standard in logic and is defined as follows: Let $A$ be a $\sigma$-structure and $B$ be a $\tau$-structure where $\sigma \subseteq \tau$. $B$ expands $A$ if the domain of $B$ is the same as the domain of $A$ and for all relational symbols $R \in \sigma$ and all constant symbols $C \in \sigma$, $R^A = R^B$ and $C^A = C^B$.

**Definition 2.** *(Model Expansion (MX))*

*Given:* a formula $\psi$ and a $\sigma$-structure $I$ with a domain $\text{Dom}$ where $\sigma \subseteq \text{vocab}(\psi)$,

*Find:* a $\tau$-structure $A$, where $\tau = \text{vocab}(\psi)$, such that $A$ expands $I$ and $A \models \psi$.

We call $A$, which is a $\tau$-structure with domain $\text{Dom}$, an expansion structure.

**Definition 3.** *(Decision Version of Model Expansion)*

*Given:* a formula $\psi$ and a $\sigma$-structure $I$ with a domain $\text{Dom}$ where $\sigma \subseteq \text{vocab}(\psi)$,

*Question:* is there a structure $A$ such that $A$ expands $I$ and $A \models \psi$?

The Model Expansion problem is denoted by $\text{MX}_{\sigma,\psi}$ when the input vocabulary $\sigma$ and specification $\psi$ are fixed. In this thesis, we are interested in data complexity ($\psi$ is fixed and the domain of input varies). The data complexity of (the decision version of) Model Expansion for logic $\mathcal{L}$ always lies in between model checking (MC) and satisfiability (SAT) for $\mathcal{L}$. For example, for first-order logic, MC is $\text{AC}_0$, MX is in NP, and SAT is undecidable. The combined complexity of Model Expansion for first-order logic is $\text{NEXPTIME}$. A complexity analysis of the three tasks (MC, MX, SAT) for several logics of interest is performed in [85]. The data and combined complexity of Model Expansion in a number of logics are presented in Table 2.2.

---

Table 2.2: Complexity of Model Expansion

<table>
<thead>
<tr>
<th>Logic</th>
<th>Data Complexity</th>
<th>Combined Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO</td>
<td>NP</td>
<td>NEXPTIME-complete</td>
</tr>
<tr>
<td>FP</td>
<td>NP</td>
<td>NEXPTIME-complete</td>
</tr>
<tr>
<td>FO$^k$</td>
<td>NP-complete</td>
<td>NP-complete</td>
</tr>
</tbody>
</table>
Note that when the input vocabulary is empty, MX is often called model generation, and if the input vocabulary is equal to the vocabulary of formula $\psi$, it is equivalent to model checking.

A variety of problems in AI, such as the Airport Gate Scheduling problem [57], can be reduced to graph colouring, which is a well-known NP-complete problem. Graph colouring can be characterized as a first-order Model Expansion (i.e., the specifications of the problem are written in first-order logic) as follows:

**Example 2.** Let $E$ be a binary relation. Let relation symbols $R$, $G$, and $B$ denote the colours red, green, and blue, respectively. Formula $\psi$, in first-order logic, specifies three-colouring as:

\[
\psi = \forall x \left[ (R(x) \lor B(x) \lor G(x)) \\
\land \neg((R(x) \land B(x)) \lor (R(x) \land G(x)) \lor (B(x) \land G(x))) \\
\land \forall x \forall y \left[ E(x, y) \supset (\neg R(x) \land R(y)) \\
\land \neg(B(x) \land B(y)) \land \neg(G(x) \land G(y)) \right] \right].
\]

A graph $G = (V, E)$ is an instance structure with vocabulary $\sigma = \{E\}$ and domain $V$ that is the set of vertices. For an input graph $G$, Model Expansion problem $\text{MX}_{\{E\}, \psi}$ finds an expansion structure $A$ (i.e., three-colouring of $G$) that interprets symbols $R$, $B$, and $G$ satisfying $\psi$ as:

\[
\left( V; E^G, R^A, B^A, G^A \right) \models \psi.
\]
Chapter 3

Prioritized Model Expansion

We take a model-theoretic approach to computationally hard problems with preferences. Computational problems are characterized as Model Expansion, that is, the logical task of expanding an input structure to satisfy a set of specifications. The uniformity of the model-theoretic approach allows us to link preferences and computational problems by introducing a framework of Prioritized Model Expansion and allows us to combine problems with preferences regardless of the syntax of their specifications. The main technical contributions are as follows.

We introduce a simple but expressive formalism to encode preferences of users associated with computationally hard problems. Conditional preferences can also be expressed in the formalism. We introduce Prioritized Model Expansion, which is Model Expansion based on these preferences. We investigate properties of Prioritized Model Expansion and conduct a thorough study of the impact of introducing preferences on the computational complexity of $\Sigma^P_k$-complete Model Expansion problems. We also discuss how Prioritized Model Expansion is related to other preference-based declarative approaches, such as SAT with preferences and preference-based Logic Programming.

3.1 Introduction

Solving computationally hard problems (e.g., NP-hard) is in the core of many AI tasks. Due to the significant progress in the performance of modern solvers, finding solutions to such problems (e.g., planning, travelling salesman, graph colouring, etc.) has become feasible in many applications. For example, in the context of resource management, consider the well-known problem of Airport Gate Scheduling [57], which, in a nutshell, is the task of assigning flight arrivals and departures to different gates of an airport. The problem can be formalized as a Constraint Satisfaction Problem (CSP). Some variations of the problem are NP-hard and have been encoded as scheduling or clique partitioning [56] that itself can be transformed into a graph colouring problem [11].

We view such hard problems as Model Expansion [100]. By distinguishing between problem instances and problem specifications, Model Expansion provides a robust modelling framework and establishes a connection to Descriptive Complexity [78].
It is common that a decision maker prioritizes the solutions. For instance, in the Airport Gate Scheduling problem, there could be some preferences for scheduling gates such that a certain gate is preferred to be assigned domestic flights rather than international flights. Since preferences play a key role in AI, a large number of frameworks for handling preferences have been proposed during the last two decades, e.g., [2, 36, 41, 20, 138]. As discussed in [48], these proposals are often language-dependent because they are added to a host formal language such as ASP [36, 33, 47] or default logic [46, 34].

In real-world applications, search and decision problems often consist of a number of sub-problems that interact with each other. For example, suppose a vacation planner includes a component that generates a vacation package based on the constraints of a travel agency by solving answer set programs and a component that solves integer linear programs to find the best plan based on the needs and priorities of a traveler. The modular nature of many AI tasks necessitates the integrating of preference-based problems regardless of the language of their specifications. We tackle this issue by proposing a preference framework for Model Expansion.

We show how our language-independent preference-based framework corresponds to prominent declarative approaches with preferences, such as Answer Set Optimization [36], Logic Programming with Preferences [113], CSP with CP-nets [21], etc. The main motivation for our work is to connect model theory, descriptive complexity, and preference modelling to study computationally hard problems with preferences. To the best of our knowledge, this is the first proposal of this kind in the literature. We propose that preferences are expressed as an ordering relation on ground atoms. The relation among atoms is lifted to a preference ordering relation among structures using a number of different preference semantics (lifting methods). We define the Dominant Structure problem, which is the problem of deciding whether a structure is preferred to another structure. We prove that solving the Dominant Structure problem is polynomial in the size of the domain of structures. Our framework allows other methods of lifting as long as the Dominant Structure problem is polynomial, which makes the following complexity results applicable.

Using model-theoretic approaches has promising advantages. For example, properties of model theory [40] can be used to identify certain syntactic fragments of a logic for answering tractable queries with preferences. We presented a model-theoretic view on modular systems with preferences in [61]. We showed the connection of our proposal to other formalisms such as CP-nets [20]. We used Codd’s relational algebra [42] to combine modules with preferences.

This chapter is an extension of [64]. Our main contributions are as follows: First, we introduce the notion of Prioritized Model Expansion, a declarative framework for specifying computational problems with preferences. Prioritized Model Expansion extends Model Expansion by modelling preferences of a decision maker. Second, we show that adding preferences even in the simplest formulations leads to a rise of the computational complexity of $\Sigma^P_k$-complete Model Expansion problems to $\Sigma^P_{k+1}$-complete prioritized problems. Third, we study the relationships of some preference-based frameworks to Prioritized Model Expansion. We apply the computational complexity result to some associated reasoning tasks to obtain similar results for those frameworks.
3.2 Model Expansion with Preferences

In this section, we introduce the notion of Prioritized Model Expansion (PMX). We model preferences as an ordering relation on a set of ground atoms. We study the computational complexity of solving problems related to PMX including Dominant Structure (i.e., given two structures, whether one is preferred to another), Optimal Expansion (i.e., given a structure, whether it is an optimal expansion of a Model Expansion problem), and Goal-Oriented Optimal Expansion (i.e., deciding whether there is an optimal expansion satisfying a certain goal).

3.2.1 Preference Expression

Let Dom be a domain of elements. Consider first-order variables \( X = \{x_1, \ldots, x_n\} \) over Dom, vocabulary \( \tau \), and \( k \)-ary \( R \in \tau \). Let \( \nu : X \rightarrow \text{Dom} \) be an assignment function that assigns a domain element to each variable. For an ordered set of variables \( \bar{x} = (x_1, \ldots, x_k) \), we call \( \bar{a} = (a_1, \ldots, a_k) \) a \( k \)-ary tuple when there is an assignment \( \nu \) such that for \( 1 \leq i \leq k \), \( \nu(x_i) = a_i \). We use the symbol \( \bar{a}[x_i] \) to denote value \( a_i \). For \( k \)-ary predicate symbol \( R \in \tau \), we call \( R(\bar{a}) \) a ground atom of \( \tau \) over \( \text{Dom} \). We say a structure \( A \) satisfies a ground atom \( R(\bar{a}) \) (notation \( A \models R(\bar{a}) \)) if \( \bar{a} \in \text{Dom}^k \).

Definition 4. (Preference Expression) A preference expression \( P \) over \( \text{Dom} \) is defined as a pair \( P = (\mathcal{S}_\tau, \sqsupseteq P) \) where \( \mathcal{S}_\tau \) is the set of all ground atoms of vocabulary \( \tau \) over \( \text{Dom} \) and \( \sqsupseteq P \) is a preorder on \( \mathcal{S}_\tau \).

Let \( R(\bar{a}) \) and \( T(\bar{b}) \) be ground atoms where \( k \)-ary predicate \( R \) and \( k' \)-ary predicate \( T \) are in \( \tau \), \( k \)-ary tuple \( \bar{a} \in \text{Dom}^k \), and \( k' \)-ary tuple \( \bar{b} \in \text{Dom}^{k'} \). The expression \( R(\bar{a}) \sqsupseteq P T(\bar{b}) \) is read as \( R(\bar{a}) \) is preferred to \( T(\bar{b}) \). Also, \( R(\bar{a}) \) is called strictly preferred to \( T(\bar{b}) \) with notation \( R(\bar{a}) \sqsupset P T(\bar{b}) \) if \( R(\bar{a}) \sqsupseteq P T(\bar{b}) \) is true and \( T(\bar{b}) \sqsupset P R(\bar{a}) \) does not hold. Also, \( R(\bar{a}) \approx P T(\bar{b}) \) if \( R(\bar{a}) \sqsupseteq P T(\bar{b}) \) and \( T(\bar{b}) \sqsupseteq P R(\bar{a}) \).

Example 3. Consider a graph \( G = (V, E) \) and three colours \( R, B, \) and \( G \) as in Example 2. Suppose \( P = (\mathcal{S}_\tau, \sqsupseteq P) \) where domain \( V = \{v_1, v_2, \ldots, v_n\} \) is a finite set of nodes and \( \tau = \{E, R, G, B\} \). \( R(v_1) \approx P R(v_2) \sqsupset P R(v_3) \) states that it is equally preferred to have \( v_1 \) and \( v_2 \) in the colour red and either of which is strictly preferred to \( v_3 \) being red. Also, \( G(v_2) \sqsupset B(v_3) \) denotes that having \( v_2 \) green is favoured over a blue \( v_3 \).

We shall introduce a preference ordering \( \geq P \) on structures based on a preference expression \( P \) and a preference semantics (lifting method) \( s \). Lifting method \( s \) specifies how \( \geq s \) is constructed from \( \sqsupseteq P \). Comparing two sets with members that are prioritized has been widely studied in different areas, such as in database systems [124], or even beyond the realm of theoretical computer science, such as in economics and decision theory [39]. Inspired by [37, 124, 2, 29], here we introduce three different methods to lift a preference ordering on ground atoms to a preference ordering on structures. In each of the following methods, relation \( \geq P \) on \( \tau \)-structures with the same domain is constructed from \( \sqsupseteq P \).
Definition 5. (Preference Relations on Structures)

Given a preference expression \( P = (S_\tau, \succeq_P) \) over domain \( Dom \), let \( A \) and \( B \) be two \( \tau \)-structures with domain \( Dom \),

- **Weak Pareto (WP).** \( A \succeq_{WP} B \) iff for all \( R, S \in \tau \) and for all \( \bar{a} \in R^A \) and all \( \bar{b} \in S^B \), \( R(\bar{a}) \succeq_P S(\bar{b}) \).

- **Upper Bound Dominance (UBD).** \( A \succeq_{UBD} B \) iff for all \( S \in \tau \) and for all \( \bar{b} \in S^B \), there is \( \bar{a} \in R^A \) such that \( R(\bar{a}) \succeq_P S(\bar{b}) \).

- **Element Dominance (ED).** \( A \succeq_{ED} B \) iff for some \( R, S \in \tau \), there is \( \bar{b} \in S^B \) and there is \( \bar{a} \in R^A \) such that \( R(\bar{a}) \succeq_P S(\bar{b}) \) and for all \( T \in \tau \), there is no \( \bar{c} \) such that \( T(\bar{c}) \succeq_P R(\bar{a}) \).

The Weak Pareto semantic uses the idea of Pareto dominance [134] such that \( A \) is preferred to \( B \) if every ground atom that is satisfied by \( A \) is at least as preferred as any ground atom which \( B \) satisfies. In a stronger version, all ground atoms that are satisfied by \( A \) must be at least as preferred as ground atoms satisfied by \( B \) except one ground atom satisfied by \( A \) that is strictly preferred to a ground atom that \( B \) satisfies. The Upper Bound Dominance approach is a weaker version of the Weak Pareto such that if \( A \) satisfies any ground atom that is at least as preferred as the maximal element (based on preorder \( \succeq_P \)) of atoms satisfied by \( B \), then \( A \) is preferred to \( B \). Finally, based on the Element Dominance semantics, there is an adequate reason to drive that \( A \) is preferred to \( B \) if there is a ground atom, say \( R(\bar{a}) \), that \( A \) satisfies and is preferred to some ground atoms satisfied by \( B \) and no ground atom satisfied by \( B \) is strictly preferred to \( R(\bar{a}) \).

These semantics (lifting methods) may illustrate similar, or in some cases, different behavior. For example, the computational complexity of reasoning tasks associated with preferred models using each of these semantics is the same. However, they behave differently in solving Prioritized Model Expansion problems that will be discussed in Chapter 5. Also, the relation \( >_y \) is transitive when \( y \) is the Upper Bound Dominance or the Weak Pareto semantics. However, if \( y \) is the Element Dominance semantics, \( >_y \) is not necessarily transitive. For example, consider the case where the relational vocabulary is \( \tau = \{ R \} \) and the domain is \( Dom = \{ 1, 2, 3, 4 \} \). Assume the preorder over ground atoms is defined as \( R(1) \sqsupseteq R(2) \) and \( R(3) \sqsupseteq R(4) \). Assume \( A, B, \) and \( C \) are \( \tau \)-structures such that \( R^A = \{ 1 \}, R^B = \{ 2, 3 \}, \) and \( R^C = \{ 4 \} \). One can check that for the Element Dominance semantics, \( A >_{ed} B, B >_{ed} C, \) but \( A >_{ed} C \) does not hold. However, the transitivity is not a requirement for relation \( >_y \) in solving Optimal Expansion and Goal-oriented Optimal Expansion problems. We emphasize similarities and differences between the preference semantics when it is necessary throughout this thesis.

We note that the lifting methods are not limited to what was proposed in Definition 5. Other lifting methods are allowed if the Dominant structure problem (see Definition 6 below) remains in polynomial time. Defining different preference semantics on the condition that the Dominant
Structure problem is in polynomial time gives us the alternatives to pick stronger or weaker versions of preference semantics based on a particular application while the computational complexity of associated reasoning tasks does not vary. We take a closer look at different preference semantics and study the underlying relations among them in Chapter 5.

The strict version of $\geq_p^s$ where $s \in \{wp, ubd, ed\}$ is defined as $A >_p^s B$ if $A \geq_p^s B$ holds but $B \geq_p^s A$ does not hold. Also, we say $A$ is dominant to $B$ based on a preference semantics $s \in \{wp, ubd, ed\}$ whenever $A >_p^s B$. We may drop $s$ when the preference semantics is clear from the context.

The problem of deciding whether $A$ is dominant is called the Dominant Structure problem and is characterized as follows:

**Definition 6. (Dominant Structure)**

**Input:** A preference expression $P = (S_\tau, \sqsupseteq_p)$ over Dom, $\tau$-structures $A$ and $B$ with domain Dom, and a preference semantics $s \in \{wp, ubd, ed\}$.

**Question:** is $A >_p^s B$?

The following result indicates that the decision problem of deciding whether a structure is dominant to another structure using one of the preference semantics in Definition 5 is polynomial in the size of the domain of the structures.

**Proposition 1.** The Dominant Structure problem is solvable in polynomial time in the size of Dom.

**Proof.** As stated by Definition 5, at most, we compare all tuples in $R^A$ and $S^B$ for all $R, S \in \tau$. The total possible number of $k$-ary tuples is $|Dom|^k$ where $k$ is the maximum arity of predicate symbols in $\tau$. Therefore, $O(|Dom|^{2k})$ comparisons are required for each $R \in \tau$. Thus, deciding whether $A >_p^s B$ is in $O(m \cdot |Dom|^{2k})$ (polynomial in the size of Dom) where $m$ is the number of elements in $\tau$. □

We note that vocabulary $\tau$ is considered to be fixed and our discussion of computational complexity is focused on the size of the domain of $A$ and $B$ in the Dominant Structure problem.

### 3.2.2 Prioritized Model Expansion

We characterize search and decision problems with preferences as Prioritized Model Expansion (PMX), which is the task of expanding an input structure to the most preferred expansion structures with respect to a preference expression.

**Definition 7. (Prioritized Model Expansion Problem)**

**Input:** An input $\sigma$-structure $I$, formula $\psi$, input vocabulary $\sigma \subseteq \text{vocab}(\psi)$, a preference expression $P = (S_\tau, \sqsupseteq_p)$ over the domain of $I$, and a preference semantics $s$

**Find:** Structure $A$ such that $A$ is an expansion structure of $\text{MX}_{\sigma, \psi}$ and there is no expansion structure $B$ such that $B >_p^s A$. 

18
Notation 3. $\Pi_{\sigma,\psi} = (MX_{\sigma,\psi}, P)$ is a Prioritized Model Expansion problem (based on a preference semantics $s$) where $MX_{\sigma,\psi}$ is a Model Expansion problem and $P = (S_r, \sqsubseteq_P)$ is a preference expression over the domain of the input structure.

Each solution of a Prioritized Model Expansion problem $\Pi_{\sigma,\psi}$ is called an optimal expansion of $\Pi_{\sigma,\psi}$. It is worth noting that all expansion structures retain the domain of input structure $I$. Also, $\psi$ and $\text{vocab}(\psi)$ are assumed to be fixed. Hereafter, unless it is mentioned otherwise, we only study the data complexity of associated reasoning tasks. The data complexity points to the fact that the domain of input $I$ (and hence the size of $I$) varies while the vocabulary of input (i.e., $\sigma$) and specifications (and therefore $\text{vocab}(\psi) = \tau$) are considered to be fixed.

Example 4. Consider the problem of graph colouring that was described as Model Expansion in Example 2. Let $G = (V, E)$ be the input graph where $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $E^G = \{(v_1, v_2), (v_2, v_1), (v_1, v_3), (v_1, v_3), (v_3, v_1), (v_2, v_4), (v_4, v_2), (v_4, v_5), (v_5, v_4), (v_3, v_5), (v_5, v_3)\}$.

Assume that we prefer red for $v_1$. Also, a red $v_4$ is favoured over a red $v_5$ and a blue $v_2$ is preferred to a green $v_2$. These preference statements can be encoded by a preference expression $P$ such that $R(v_1) \sqsubseteq_P B(v_1)$ and $R(v_4) \sqsubseteq_P R(v_5)$. Also, $R(v_4) \sqsubseteq_P R(v_5)$ and $B(v_2) \sqsubseteq_P G(v_2)$. The Prioritized Model Expansion problem $\Pi_{\{E\},\psi} = (MX_{\{E\},\psi}, P)$ where $MX_{\{E\},\psi}$ is the characterization of three-colouring for input graph $G$ and $P$ is the preference expression. The input graph $G$ has 18 possible three-colourings. Among these solutions, $A$, as it is exhibited in Figure 3.1, is an optimal expansion of $G$ (based on the Element Dominance semantics) where $R^A = \{v_1, v_4\}$, $B^A = \{v_2, v_5\}$, and $G^A = \{v_3\}$.

In the rest of this subsection, we discuss some decision problems that are associated with Prioritized Model Expansion.

The Optimal Expansion problem asks whether a given structure is an optimal expansion of a Prioritized Model Expansion problem.

Definition 8. (Optimal Expansion)
**Input:** A $\tau$-structure $A$ and a Prioritized Model Expansion problem $\Pi_{\sigma,\psi} = (MX_{\sigma,\psi}, P)$ based on a preference semantics $s$ where $MX_{\sigma,\psi}$ is a Model Expansion problem with an arbitrary input $\sigma$-structure $\mathcal{I}$ such that $\sigma \subseteq \tau = \text{vocab}(\psi)$ and $P = (S, \sqsupseteq P)$ is a preference expression over the domain of $\mathcal{I}$.

**Question:** is $A$ an optimal expansion of $\Pi_{\sigma,\psi}$?

**Proposition 2.** For a Model Expansion problem $MX_{\sigma,\psi}$, let model checking of $\psi$ (given a structure $A$, decide whether $A \models \psi$) be in a complexity class $Y$. The problem of Optimal Expansion is in $\text{co-NP}^Y$.

**Proof.** The complementary problem is deciding whether there is an expansion structure $B$ such that $B >_P A$. The complementary problem can be solved by a non-deterministic polynomial Turing machine guessing $B$ with access to an oracle in $Y$ that decides whether $B$ is an expansion of $MX_{\sigma,\psi}$ (this includes checking whether $B$ expands $\mathcal{I}$ in polynomial time and whether $B \models \psi$ in complexity $Y$) and, based on Proposition 1, in polynomial time checks whether $B >_P A$. Thus, the complementary problem is in $\text{NP}^Y$ and the original problem is in $\text{co-NP}^Y$.

To put the impact of defining preferences in perspective, consider that deciding whether a given structure $A$ is an expansion of $MX_{\sigma,\psi}$ has complexity $Y$ because it is required to first check whether $A$ is an expansion of $\mathcal{I}$ that can be decided in polynomial time and then determine whether $A \models \psi$ with complexity $Y$. On the other hand, deciding whether $A$ is an optimal expansion is in $\text{co-NP}^Y$. In fact, to decide whether a structure is an optimal expansion, not only must it be verified as an expansion, but it must also be compared to all possible expansion structures.

The decision version of a Model Expansion problem asks whether there is an expansion structure. One might ask whether the complexity of deciding whether there is an optimal expansion differs from the Model Expansion problem. Interestingly, the answer is that the complexities of these two problems are the same. The reason is that $\geq_P$ that is constructed from $P$ is a binary relation over all possible $\tau$-structures with the domain of input $\mathcal{I}$. If there is an expansion, it is guaranteed that there is an optimal expansion as well. However, a question might arise about the complexity of deciding the existence of optimal expansion satisfying certain properties. One of the common tasks in many AI applications is to determine whether a certain goal is achieved by solutions to a problem, e.g., in automated AI planning [5]. In the context of Prioritized Model Expansion, we ask whether there is an optimal expansion that satisfies a certain formula (goal). The problem is formulated as follows:

**Definition 9. (Goal-Oriented Optimal Expansion)**

**Input:** A Prioritized Model Expansion $\Pi_{\sigma,\psi} = (MX_{\sigma,\psi}, P)$ where $MX_{\sigma,\psi}$ is a Model Expansion problem with an arbitrary input $\sigma$-structure $\mathcal{I}$, $\tau = \text{vocab}(\psi)$, $P = (S, \sqsupseteq P)$ is a preference expression over the domain of $\mathcal{I}$, and $\phi$ is a formula of the form $R_i(\overline{a}_j) \land ... \land R_l(\overline{a}_k)$ where $R_i, ..., R_l \in \tau$ and $R_i(\overline{a}_j), ..., R_l(\overline{a}_k)$ are ground atoms over a finite domain $\text{Dom}^*$.

**Question:** is there an optimal expansion $A$ of $\Pi_{\sigma,\psi}$ such that $A \models \phi$?
**Proposition 3.** Let solving Optimal Expansion problem $\Pi_{\sigma,\psi} = (MX_{\sigma,\psi}, P)$ be in a complexity class $X$. The problem of Goal-Oriented Optimal Expansion is in $NP^X$.

**Proof.** First, we non-deterministically guess a $\tau$-structure $A$ and in polynomial time check if $\bar{a} \in R^A$, for all ground atoms $R(\bar{a})$ appearing in $\phi$, which can be done by means of a non-deterministic polynomial Turing machine. Second, we check whether our guess is an optimal expansion, which is in complexity class $X$ by the assumption. Thus, the problem can be solved by a non-deterministic polynomial Turing machine using an oracle in $X$. Hence, the problem is in $NP^X$.  

A generalization of the Goal-Oriented Optimal Expansion problem is to find an optimal expansion satisfying a formula $\phi$ in a certain logic $L^*$. In this case, the complexity of model checking in logic $L^*$ is taken into account. However, for the sake of simplicity, in this chapter we consider goal $\phi$ as a conjunction of ground atoms. Hence, deciding whether a structure $A$ satisfies $\phi$ can be verified in polynomial time.

**Prioritized $\Sigma^P_k$-complete Model Expansion problems**

In this subsection, we discuss the impact on computational complexity of introducing preferences on $\Sigma^P_k$-complete Model Expansion problems.

As was discussed in [100], any boolean query computable in NP can be expressed as a first-order Model Expansion $MX_{\sigma,\psi}$ where $\psi$ is a first-order formula. Based on Fagin’s theorem [66], NP is the class of boolean queries expressible in existential second-order logic ($\exists SO$). This shows that a first-order MX and existential second-order logic have the same expressive power. Similarly, the Polynomial Hierarchy is the set of boolean queries expressible in second-order logic and any query computable in $\Sigma^P_k$ ($k > 1$) can be encoded as $MX_{\sigma,\psi}$ where $\psi$ is a formula of the form $Q_1, ..., Q_{k-1}\psi^*$ such that for $1 \leq i \leq k$, $Q_i$’s are alternating second-order quantifiers (alternation between $\exists$ and $\forall$), $Q_1 = \forall$, $Q_{k-1}$ is $\forall$ if $k$ is even and $\exists$ otherwise, and $\psi^*$ is a first-order formula. If the decision version of a Model Expansion problem $MX_{\sigma,\psi}$ is in $\Sigma^P_k$, then the problem has a second-order specification $\psi$ of the form $Q_1, ..., Q_{k-1}\psi^*$. Thus, the complexity of model checking of $\psi$ is in $\Pi^P_{k-1}$ and, hence, based on Proposition 2, solving the Optimal Expansion problem $(MX_{\sigma,\psi}, P)$ is in $\Sigma^P_k$.

For $\Sigma^P_1$-complete Model Expansion problems, we show that the problem of deciding the existence of minimal solutions to an abductive logic program [59] satisfying a goal can be reduced to Goal-Oriented Optimal Expansion similarly [113]. An abductive logic program is a tuple $ALP = \langle H, M, P \rangle$ over a set $A$ of propositional atoms where $P$ is a normal logic program, $H \subseteq A$ is called the hypothesis and $M \subseteq A \cup \{\neg a | a \in A\}$ is the manifestation. A solution of $ALP$ is a set $N \subseteq H$ such that there is a stable model $S$ of $P \cup N$ and $M \subseteq S$. A solution $N$ is called $(H)$ minimal if there is no solution $N'$ such that $N' \subset N$. For a given hypothesis $h \in H$, deciding whether there is a minimal solution $N$ such that $h \in N$ is $\Sigma^P_2$-complete [59].

21
For problems in the higher levels of the Polynomial Hierarchy, we consider the following: $\Sigma_k^P$-complete problems can be encoded as a combined logic program [15]. $\Pi = (\mathcal{P}_g, \mathcal{P}_t)$ is called a combined logic program where $\mathcal{P}_g$ and $\mathcal{P}_t$ are logic programs over a set of propositional variables $G$ and $T$, respectively. $M$ is a model of $\Pi$ if it is a stable model of $\mathcal{P}_g$ and there is not a stable model $N$ of $\mathcal{P}_t$ such that $M \cap G = N \cap T$. The decision version of this problem is $\Sigma_k^P$-complete. Recursively, the existence of a model of a combined program in depth 2 defined as $\Pi_2 = (\mathcal{P}_{g_2}, (\mathcal{P}_{g_1}, \mathcal{P}_t))$ is $\Sigma_k^P$-complete and, similarly, in depth $k$, the existence of a model of $(\mathcal{P}_{g_k-1}, \Pi_{k-2})$ is $\Sigma_k^P$-complete.

We introduce abductive combined program as $\mathcal{C}$ of complete problems can be encoded as a combined logic program [15].

$\Pi$ the existence of a model of a combined program in depth 2 defined as $\Pi_2 = (\mathcal{P}_{g_2}, (\mathcal{P}_{g_1}, \mathcal{P}_t))$ is $\Sigma_k^P$-complete and, similarly, in depth $k$, the existence of a model of $(\mathcal{P}_{g_k-1}, \Pi_{k-2})$ is $\Sigma_k^P$-complete.

We introduce abductive combined program as $\mathcal{C} = (H, M, \Pi)$ where $\Pi = (\mathcal{P}_g, \mathcal{P}_t)$ is a combined logic program. $W$ is a solution of $\mathcal{C}$ if there is a model of $\Pi$ such that $M \subseteq S$. $W$ is minimal if there does not exist a solution $W'$ such that $W' \subset W$.

**Lemma 1.** The problem of deciding whether $\mathcal{C}_k = (H, M, \Pi_k)$, where $k > 0$, for a given $h \in H$ has a minimal solution containing $h$ is $\Sigma_{k+1}^P$-complete.

**Proof:** The proof includes a translation from a quantified boolean formula (QBF) to $\mathcal{C}$ for the base case $k = 2$ and then, by induction on $k$ for $k > 2$, the result follows. In the first part, let $\varphi$ be a boolean formula in CNF and $X = \{x_1, ..., x_m\}$, $W = \{w_1, ..., w_m\}$, $X' = \{x_1', ..., x_m'\}$, $Y = \{y_1, ..., y_n\}$, and $Z = \{z_1, ..., z_l\}$ be a set of boolean variables in $\varphi$. Let $t$, $h$, and $f$ also be boolean variables. Consider $\mathcal{P}_g$ to be a set of rules of the form $\{t \leftarrow x_i, x_i'\}, \{w_i \leftarrow x_i\}, \{w_i \leftarrow x_i'\}$, $\{t \leftarrow y_1, ..., y_n, h\}$, and $\{f \leftarrow l_1, ..., l_r\}$ where $-(l_1 \land, ..., \land l_r) \in \varphi$ similar to [59]. For $X \cup X' \subseteq H$, an $H$-minimal solution of $\langle H, \{t\} \cup W, \mathcal{P}_g \rangle$ does not contain $f$ and it has either $x_i$ or $x_i'$. In the second part, on the other hand, similar to [15], assume $\mathcal{P}_t$ determines the truth value of a set of boolean variables $Z$. Also, for each clause $C \in \varphi$, suppose that $\mathcal{P}_t$ includes a set of rules of the form $t \leftarrow -C$ and $f \leftarrow -f, t$ that means $t$ must not be in any stable model of $\mathcal{P}_t$. This implies that the validity of $\exists X \forall Y \exists Z \varphi$ is equivalent to the existence of an $H$-minimal solution of $\mathcal{C}$ that contains $h$. So, for $k = 2$, the existence of a minimal solution to an abductive combined logic program containing an atom $h$ is $\Sigma_3^P$-complete. For the induction step, assume the problem in Lemma 1 for $\mathcal{C}_k = (H, M, \Pi_k)$ is $\Sigma_{k+1}^P$-complete. For $\mathcal{C}_{k+1} = (H, M, \Pi_{k+1})$, the reduction from QBF in the first part is the same as in the base case. Also, for $\Pi_{k+1} = (\mathcal{P}_{g_k}, \Pi_{k-1})$, deciding whether $\Pi_{k-1}$ has a stable model is NP-complete using an oracle in $\Sigma_{k-2}^P$. Therefore, in the second part, the reduction is from an QBF problem of the form $\exists X_1 \forall X_2 ... Q_{k-1} X_{k-1} \varphi$ for $k - 1$ quantifier alternations where $Q$ is $\exists$ when $k$ is even and $Q$ is $\forall$ otherwise. Therefore, for $\mathcal{C}_{k+1} = (H, M, \Pi_{k+1})$ and a given $h \in H$, deciding whether $\mathcal{C}_{k+1}$ has a minimal solution containing $h$ is $\Sigma_{k+2}^P$-complete. \[\blacksquare\]

The following result shows the impact of introducing preferences to $\Sigma_k^P$-complete Model Expansion problems.

**Theorem 1.** Let the decision version of a Model Expansion problem $\text{MX}_{\sigma, \psi}$ be $\Sigma_k^P$-complete. The problem of Goal-Oriented Optimal Expansion for $\text{MX}_{\sigma, \psi}$ is $\Sigma_{k+1}^P$-complete.

**Proof.** The membership to $\Sigma_{k+1}^P$ follows from the results of Proposition 2, Proposition 3, and properties of Model Expansion. Since the Model Expansion problem is in $\Sigma_k^P$, it has a second-order
specification with \( k - 1 \) number of alternations between second-order quantifiers. Therefore, the complexity of model checking of \( \psi \) is in \( \Pi^P_{k-1} \). Thus, based on proposition 2, the complexity of the Optimal Expansion problem is in co-NP^{\Sigma^P_{k-1}} which is equal to \( \Pi^P_k \). Also, based on Proposition 3, the Goal-Oriented Optimal Expansion problem is in NP^{\Pi^P_k} or NP^{\Sigma^P_k} that is equal to \( \Sigma^P_{k+1} \).

For the proof of hardness, we consider an abductive logic program \( ALP = \langle H, M, P \rangle \). Let us define a logic program \( P' \) as a set of rules of the form \( r : R(a) \leftarrow \neg S(b) \) for any \( R(a) \sqsubseteq P S(b) \). Rule \( r \) says that a better conclusion is drawn from not making a less preferred assumption. Define \( P^* = P \cup P' \). The problem of deciding the existence of a stable model of a logic program is NP-complete and it can be translated into the decision version of a Model Expansion problem \( MX_{\sigma, \psi} \). The program can be represented by an instance structure and the stable model semantics is characterized by \( \psi \) (e.g., a first-order Model Expansion characterization of ASP was shown in \([100]\)). The problem of finding out whether there is an \( H \)-minimal solution of \( ALP \) can be reduced to deciding whether there is an optimal expansion in \( (MX_{\sigma, \psi}, P) \) where \( P^* \) is translated into \( MX_{\sigma, \psi} \). Assume \( X_1 \) and \( X_2 \) are two stable models of \( P^* \). If \( X_1 \) is preferred to \( X_2 \) with respect to one of the preference semantics in Definition 5, then there is \( R(\pi) \in X_1 \) and \( S(\overline{b}) \in X_2 \) such that \( R(\pi) \sqsubseteq_P S(\overline{b}) \). So, we have \( X_1 \cap H \subseteq X_2 \cap H \) and therefore, each preferred answer set is \( H \)-minimal. Hence, finding a minimal solution for \( \langle H, M, P \rangle \) is reduced to finding an optimal expansion of \( \Pi_{\sigma, \psi} \) that satisfies a goal \( M \). Thus, Goal-Oriented Optimal Expansion for an NP-complete MX is \( \Sigma^P_2 \)-complete. By using the same argument and according to Lemma 1, finding a minimal solution for an abductive combined logic program in level \( k \) can be translated into a Goal-Oriented Optimal Expansion where the Model Expansion problem is \( \Sigma^P_{k+1} \)-complete and hence the result follows.

\[ \square \]

Theorem 1 presents an important consequence of adding preferences to a \( \Sigma^P_k \)-complete Model Expansion problem. For the problem of deciding whether there is an expansion that satisfies a goal \( \phi \), adding preferences leads to a jump in the Polynomial Hierarchy. So, the preference relation between expansion structures derived from a preference expression can not be translated into axiomatization \( \psi \) with polynomial time model checking unless P=NP or the Polynomial Hierarchy collapses.

**Example 5.** Deciding whether a graph is Hamiltonian (i.e., whether a graph has a Hamiltonian cycle) is a well-known NP-complete problem. We first characterize the Hamiltonian graph problem as first-order Model Expansion and then examine the Goal-Oriented Prioritized Model Expansion problem for an input graph with some preferences. Consider vocabulary \( \tau = \{ E, H \} \). For an arbitrary graph \( G = (V, E^G) \) (represented as a structure) such that \( V \) is the set of vertices of the graph and \( E^G \) specifies edges of \( G \), the Hamiltonian graph problem is defined as a Model Expansion problem \( MX_{(E)}, \Psi \) where \( \{ E \} \) is the vocabulary of input and \( \Psi = \psi_1 \land \psi_2 \land \psi_3 \) such that
(a) The preferred Hamiltonian path that satisfies goal $\phi$ of Example 5

(b) A valid Hamiltonian path that is not preferred

Figure 3.2: Two Possible Hamiltonian Paths for Graph $G$ in Example 5

$$
\psi_1 = \forall x \forall y (H(x, y) \lor H(y, x)) \\
\psi_2 = \forall x \forall y \forall z (H(x, y) \land H(y, x) \supset H(x, z)) \\
\psi_3 = \forall x \forall y ((H(x, y) \land \neg \exists z [H(x, z) \land H(z, y)]) \supset E(x, y))
$$

where $\psi_1$ indicates there is a Hamiltonian path between any arbitrary pair of vertices $x$ and $y$, $\psi_2$ stipulates the transitivity property of the path, and, based on $\psi_3$, if there are two adjacent vertices in the path, they must be connected through an edge of the graph. A $\tau$-structure $A$ is an expansion structure of $MX_{E, \psi}$ if $A$ expands $G$ and satisfies $\psi$ as follows:

$$
\begin{aligned}
\text{Model Expansion problem } MX_{E, \psi} & \text{ asks whether there is a Hamiltonian path (and therefore a Hamiltonian cycle) in the input graph. If the answer is yes, then there is a $\tau$-structure } B \text{ such that } H^B \\
\text{constitutes a Hamiltonian path. Assume } V = \{v_1, v_2, v_3, v_4\} \text{ is a set of vertices and } G = (V, E^G) \text{ is defined } E^G = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_1), (v_1, v_4), (v_3, v_1), (v_4, v_2)\}. A \text{ preference expression } P \text{ is defined as } H(v_4, v_2) \sqsupset p H(v_3, v_4) \text{ which means it is preferred that a Hamiltonian path includes } H(v_4, v_2) \text{ (i.e., edge } E(v_4, v_2) \text{ due to specification } \psi_3) \text{ rather than } H(v_3, v_4) \text{ (i.e., edge } E(v_4, v_2) \text{ because of specification } \psi_3). \\
\text{The Prioritized Model Expansion problem } \Pi_{\{E\}, \psi} = (MX_{\{E\}, \psi}, P) \text{ (in the search version) finds the preferred Hamiltonian paths of } G \text{ if there are such paths. Let us define a goal formula } \phi = H(v_3, v_1). \text{ The Goal-Oriented Model Expansion Problem asks whether there is a preferred expansion structure (i.e., Hamiltonian path) that includes } H(v_3, v_1) \text{ and hence edge } E(v_3, v_1). \text{ As is illustrated in Figure 3.2, there are two possible Hamiltonian paths } H^A = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_1)\} \text{ and } H^B = \{(v_3, v_1), (v_1, v_4), (v_4, v_2), (v_2, v_3)\}. \text{ Based on Definition 5, } A \text{ characterizes a path that is preferred and it is comprised of } H(v_3, v_4).
\end{aligned}
$$
3.2.3 Conditional Preferences

Preferences of users are often expressed in conditional statements. For example, A is preferred to B if C is true. Contextual preferences are widely studied in the literature. Examples include [20], [26], [95], and [28]. Here, we show how the Prioritized Model Expansion framework handles conditional preferences.

Let \( p \) be a conditional preference of the form \( p : R_q(\overline{a}_u), ..., R_s(\overline{a}_v) \geq S_i(\overline{b}_1) \geq_p S_j(\overline{b}_m) \).

Conditional preference \( p \) is read as \( S_i(\overline{b}_1) \) is preferred to \( S_j(\overline{b}_m) \) if \( R_q(\overline{a}_u) \land ... \land R_s(\overline{a}_v) \) is true. Intuitively, we aim to construct a partial order \( \geq_p \) from \( p \) such that for structures \( A \) and \( B \) with the same domain and vocabulary, \( A \) is preferred to \( B \) with respect to \( p \) if \( A \models R_q(\overline{a}_u) \land ... \land R_s(\overline{a}_v) \land S_i(\overline{b}_1) \land \) and \( B \models R_q(\overline{a}_u) \land ... \land R_s(\overline{a}_v) \land S_j(\overline{b}_m) \). We call \( R_q(\overline{a}_u) \land ... \land R_s(\overline{a}_v) \) the body of \( p \) with notation \( \text{body}(p) \).

**Definition 10.** \( \Pi_{\sigma,\psi} = (MX_{\sigma,\psi}, \mathcal{P}) \) is called a General Prioritized Model Expansion Problem where \( \mathcal{P} = \{p_1, \ldots, p_n\} \) is a set of conditional preferences.

A translation of \( \Pi_{\sigma,\psi} \) into a standard Prioritized Model Expansion problem \( \Pi_{\sigma,\psi}^\ast = (MX_{\sigma,\psi}^\ast, P^\ast) \) is as follows: First, we add each element of \( \text{Dom} \) as a constant symbol to \( \tau \). We assume that for all \( a \in \tau \) such that \( a \in \text{Dom} \) and for all optimal expansions \( A, a^A = a \). For conditional preference \( p \), let us introduce two new (ground) auxiliary atoms \( T \) and \( T' \). Consider formulas \( \psi_1 : (R_q(\overline{a}_u) \land ... \land R_s(\overline{a}_v) \land S_i(\overline{b}_1)) \supset T \) and \( \psi_2 : (R_q(\overline{a}_u) \land ... \land R_s(\overline{a}_v) \land S_j(\overline{b}_m)) \supset T' \). Set \( \psi^\ast = \psi \land \psi_1 \land \psi_2, T \supseteq_{\tau^\ast} T' \), and \( \tau^\ast = \tau \cup \{T, T'\} \). It is clear that an expansion \( \tau^\ast \)-structure \( A \) is preferred to another expansion \( \tau^\ast \)-structure \( B \) with respect to \( p \) if \( A \) and \( B \) satisfy \( R_q(\overline{a}_u) \land ... \land R_s(\overline{a}_v), A \models S_i(\overline{b}_1) \), and \( B \models S_j(\overline{b}_m) \). The binary relation \( \geq_{\tau^\ast} \) is constructed from \( \mathcal{P} \) based on the preference semantics Weak Pareto, Upper Bound Dominance, and Element Dominance similarly to before.

**Example 6.** For a graph three-colouring problem with conditional preferences \( \Pi_{\{E\},\psi} = (MX_{\{E\},\psi}, \mathcal{P}) \), consider the following conditional preferences: red \( v_1 \) is preferred to blue \( v_2 \) if \( v_5 \) is green which is expressed as \( p : R(v_1) \supset (B(v_2) \supset G(v_5)) \). We introduce new atoms \( T_1 \) and \( T_2 \) such that \( \psi_1 : (R(v_1) \land B(v_2)) \supset T_1 \) and \( \psi_2 : (R(v_1) \land G(v_5)) \supset T_2 \). Also, we set \( \psi^\ast = \psi \land \psi_1 \land \psi_2 \) and \( T_1 \supseteq_p T_2 \). For an input graph \( G = (V, E^G) \) where \( V = \{v_1, ..., v_5\} \) and \( E^G = \{(v_1, v_2), (v_2, v_1), (v_1, v_3), (v_3, v_1), (v_1, v_4), (v_4, v_2), (v_4, v_1), (v_5, v_4), (v_3, v_5), (v_5, v_3)\} \), two three-colourings of \( G \) are compared in Figure 3.3 based on the Week Pareto semantics.

3.3 The Relation Between Prioritized Model Expansion and Other Preference Formalisms

In this section, we study some examples of declarative specifications of problems with preferences and their relation to Prioritized Model Expansion.
Figure 3.3: Conditional Graph Three-Colouring
If $v_1$ is red, then blue $v_2$ is preferred to green $v_5$. Three-colouring (a) is preferred to three-colouring (b).

### 3.3.1 Preference-based SAT

3-SAT is a canonical NP-complete problem. Let $\varphi$ be a Conjunctive Normal Form (CNF) boolean formula and $X = \{x_1, \ldots, x_n\}$ be the set of boolean variables appearing in $\varphi$. A truth assignment $I$ is a mapping from $X$ to \{true, false\}. The problem of deciding whether $\varphi$ is satisfiable (i.e., there is a truth assignment $I$ that satisfies $\varphi$) can be converted into a Model Expansion problem $MX_{\sigma, \psi}$ with an input $\sigma$-structure $I$. The general idea is to represent formula $\varphi$ by $I$. The domain of $I$ is $X$. Also, the expansion vocabulary (i.e., $\tau \backslash \sigma$) includes unary predicates $T$ and $F$ that specify each boolean variable as true or false. An interpretation of the expansion vocabulary represents a truth assignment to the boolean variables in $\varphi$. Formula $\psi$ specifies the notion of satisfying a boolean CNF formula.

As an example, we encode 3-SAT for $\varphi$ as first-order Model Expansion. Let $MX_{\sigma, \psi}$ be a Model Expansion problem with input $I$ where $\psi$ is a first-order formula of the form $\forall x \forall y \forall z (C(x, y, z) \rightarrow T(x) \lor T(y) \lor T(z)) \land \forall x (T(x) \equiv \neg F(x)) \land \forall x \forall y (\text{Neg}(x, y) \rightarrow (T(x) \equiv F(y)))$ where $C(x, y, z)$ denotes a clause containing variables $x, y$ and $z$, $T(x)$ and $F(x)$ represent the value true and the value false for $x$, respectively, and $\text{Neg}(x, y)$ denotes that $x$ is the negation of $y$. Also, assume a structure $I$ represents $\varphi$ such that $C^I$, which is the interpretation of predicate symbol $C$, is the set of clauses appearing in $\varphi$. More precisely, for each $(x_i, x_j, x_k) \in C^I$, $(x_i \lor x_j \lor x_k)$ is a clause in formula $\varphi$. The vocabulary of input structure $I$ is $\{C\}$ and the domain of $I$ is $I = \{x, \neg x | x \in X\}$. Let $A$ be an expansion structure such that the vocabulary of $A$ is $\tau = \{C, T, F\}$. We say $A$ represents a truth assignment $I$ (or $I$ is constructed from $A$) if for each $x \in T^A$, we have $\mathcal{I}(x) = true$ and for each $x \in F^A$, we have $\mathcal{I}(x) = false$.

The associated Model Expansion problem is defined as deciding whether there is an expansion structure $A$ such that

$$\frac{I}{A} \models_{\psi}$$
It is clear that a truth assignment $I$ that satisfies $\varphi$ can be constructed from an expansion structure $A$ in polynomial time in the size of $X$.

Lemma 2. Let $MX_{\sigma,\psi}$ be a Model Expansion problem that characterizes a SAT problem for a boolean formula $\varphi$ and boolean variables $X$ appearing in $\varphi$. For each expansion structure $A$, a truth assignment $I$ satisfying $\varphi$ can be constructed from $A$ in polynomial time in the size of $X$.

Proof. For each ground atom $T(a)$ where $a \in I$, if $a \in T^A$, we assign $I(a) = \text{true}$ and $I(a) = \text{false}$ for $a \in F^A$. The total number of ground atoms is a polynomial factor of the size of $I$. □

Preference-based SAT, which is related to the problem of max-SAT [13], is the problem of finding truth assignments satisfying a boolean formula when some variables are favoured to be assigned the value true. A preference-based SAT problem for a boolean formula $\varphi$ and the set $X$ of boolean variables appearing in $\varphi$ is defined as a pair $(\varphi, (X, \geq))$ where $\geq$ is a preorder on $X$ that specifies a preference over variables in $X$. Let $I$ and $I'$ be truth assignments that satisfy $\varphi$. We say $I$ is preferred to $I'$ if each variable assigned the value true by $I$ is preferred to every variable that $I'$ maps to true. The following result indicates how preference-based SAT is related to Prioritized Model Expansion.

Theorem 2. Let $S = (\varphi, (X, \geq))$ be a preference-based SAT problem where $\varphi$ is a CNF boolean formula, $X = \{x_1, ..., x_n\}$ is the set of boolean variables appearing in $\varphi$, and $\geq$ is a partial order over $X$. There is a Prioritized Model Expansion problem $\Pi = (MX_{\sigma,\psi}, P)$ with an input $I$ such that if $A$ is a solution of $\Pi$, then there is a preferred truth assignment $I$ satisfying $\varphi$ that can be constructed from $A$ in polynomial time.

Proof. Assume $I$ represents $\varphi$ and $\psi$ characterizes the satisfaction of $\varphi$ by a truth assignment. Preference order $\geq$ over elements of $X$ in the preference-based SAT can be translated into preference expression $P$ in $\Pi$ where $\exists_P$ specifies a preorder over $\{T(x) \mid x \in X\}$. Therefore, $I$ is preferable to $I'$ if and only if $B$ is preferred to $B'$ based on the Weak Pareto semantics where $B$ and $B'$ represent $I$ and $I'$, respectively. □

As a result, deciding whether a preferred truth assignment maps a set of boolean variables to true can be encoded as a Goal-Oriented Optimal Expansion problem.

Let $S = (\varphi, (X, \geq))$ be a preference-based SAT problem where $\varphi$ is a CNF boolean formula, $X = \{x_1, ..., x_n\}$ is the set of boolean variables appearing in $\varphi$, and $\geq$ is a partial order over $X$.

Theorem 3. The problem of deciding whether there is a preferred truth assignment $I$ satisfying $\varphi$ that maps all variables in some $Y \subset X$ to true is $\Sigma^P_2$-complete.

Proof: Consider a Prioritized NP-complete Model Expansion problem $\Pi_{\sigma,\psi} = (MX_{\sigma,\psi}, P)$. Every NP-complete problem can be reduced to SAT in polynomial time. Also, by considering the
Weak Pareto semantics in the Prioritized Model Expansion framework, preference relation $\geq$ in the preference-based SAT framework is matched with $\equiv_P$. For each relation $R(\overline{a}) \equiv_P S(\overline{b})$, we consider $x_i \geq x_j$ in the preference-based SAT framework where $x_i$ and $x_j$ are boolean variables representing ground atoms $R(\overline{a})$ and $S(\overline{b})$, respectively. In order to reduce an NP-complete $MX_{\sigma,\psi}$ problem to SAT, some auxiliary propositional variables are introduced that are considered to be equally preferred with respect to relation $\equiv_P$. Therefore, Prioritized NP-complete Model Expansion problem $\Pi_{\sigma,\psi}$ is reduced to a preference-based SAT $S$ in polynomial time. Moreover, a formula $\varphi$ that is a conjunction of ground atoms can be viewed as a set of propositional atoms $Y$. Thus, the Goal-Oriented Optimal Expansion problem for formula $\varphi$ and Prioritized Model Expansion problem $\Pi_{\sigma,\psi} = (MX_{\sigma,\psi}, P)$ where $MX_{\sigma,\psi}$ is an NP-complete problem can be reduced (in polynomial time) to deciding whether there is a preferred truth assignment $\mathcal{I}$ satisfying $\varphi$ that maps all variables in some $Y \subseteq X$ to true, which is, based on Theorem 1, $\Sigma_2^P$-complete.

### 3.3.2 Logic Programs with Preferences

Logic programming with stable model semantics is one of the main declarative approaches for specifying problems in NP. Deciding whether a normal logic program has a stable model is NP-complete. Let $P$ be a normal program that is defined as a set of rules of the form $r : c_1, \ldots, c_l \leftarrow a_1, \ldots, a_m, \not b_1, \ldots, \not b_n$ where $a_i$ for $i \in [1, m]$, $b_j$ for $j \in [1, n]$, and $c_k$ for $k \in [1, l]$ are propositional atoms. We use notation $\text{body}^+(r)$ to refer to $\{a_1, \ldots, a_m\}$, $\text{body}^-(r)$ to refer to $\{b_1, \ldots, b_n\}$, and $\text{head}(r)$ to denote $\{c_1, \ldots, c_l\}$. Program $P$ is called disjunctive if for some rules $r$ in $P$, $\text{head}(r)$ is of the form $c_1 \lor c_2 \lor \ldots \lor c_l$. We use notation $\text{Atom}(P)$ to denote all propositional atoms appearing in $P$. An answer set for a normal program $P$ is a minimal set of atoms $X \subseteq \text{Atom}(P)$ such that for every rule $r \in P$, if $\text{body}^+(r) \subseteq X$ and $\text{body}^- \cap X = \emptyset$, then $\text{head}(r) \subseteq X$. Formally speaking, for a set of atoms $X$, $P^X$ is called the reduct of $P$ to $X$ where $P^X = \{\text{head}(r) \leftarrow \text{body}^+(r) | r \in P \text{ and } \text{body}^-(r) \cap X = \emptyset\}$. We say a minimal set of atoms $X \subseteq \text{Atom}(P)$ is an answer set of $P$ if and only if for every rule $r \in P^X$, if $\text{body}^+ \subseteq X$, then $\text{head}(r) \subseteq X$.

In Model Expansion framework, problem instances and specifications are separated. The problem of deciding whether a propositional program has a stable model can be cast as a Model Expansion problem where the program can be thought as an instance structure and the problem specification characterizes the stable model semantics. For example, we show a first-order characterization of the stable model semantics as follows:

Let us define vocabulary symbols $\text{Rule}$, $\text{Stable}$, $\text{Body}^+$, $\text{Body}^-$, and $\text{Head}$. Symbol $\text{Stable}(x)$ denotes that $x$ belongs to a stable model of program $P$, $\text{Rule}(y)$ means that $y$ is a rule of program $P$, $\text{Body}^+(y, x)$ indicates that $x$ is a positive atom in the body of rule $y$ and, similarly, $\text{Body}^-(y, x)$ points out that $x$ is an atom in the negative part of the body of $y$. Also, $\text{Head}(y, x)$ means that atom $x$ is in the head of rule $y$. For $\sigma = \{\text{Rule}, \text{Stable}, \text{Body}^+, \text{Body}^-, \text{Head}\}$, a $\sigma$-structure $\mathcal{P}$ specifies a program by interpreting predicate symbols in $\sigma$. A first-order formula $\psi$ characterizes the stable...
model semantics as

\[ \psi = \forall x (\exists y (\text{Rule}(y) \land \text{Head}(x, y) \land \forall z [\text{Body}^+(y, z) \supset \text{Stable}(z)] \land \forall w [\text{Body}^-(y, w) \supset \neg \text{Stable}(w)]) \supset \text{Stable}(x)) \land \forall x (\exists y (\text{Rule}(y)) \land \text{Head}(x, y) \land \forall z [\neg \text{Body}^+(y, z)] \land \forall w [\neg \text{Body}^-(y, w) \supset \text{Stable}(x)]. \]

A stable model of program \( P \) is represented by an expansion structure \( M \) that expands input \( \sigma \)-structure \( \mathcal{P} \), which represents program \( P \), and satisfies \( \psi \) as follows:

\[ \frac{\mathcal{P}}{A; \text{Rule}^\mathcal{P}, \text{Head}^\mathcal{P}, \text{Body}^+\mathcal{P}, \text{Body}^-\mathcal{P}, \text{Stable}^\mathcal{M}} \models \psi. \]

\( A \) contains the set of all propositional atoms (i.e., the domain of \( \mathcal{P} \)) and \( M \) is a \( \tau \)-structure (\( \tau = \sigma \cup \{\text{Stable}\} \)) that expands \( \mathcal{P} \) and satisfies \( \psi \). Relation \( \text{Stable}^\mathcal{M} \) represents a stable model of \( P \). For disjunctive programs, a second-order universal quantifier over \( \text{Stable} \) ensures that all interpretations of \( \text{Stable} \) are minimal. We say a program \( P \) is represented by a \( \sigma \)-structure \( \mathcal{P} \), if for each rule \( r \) in \( P \), \( r \in \text{Rule}^\mathcal{P} \), for each atom \( x \) in the head of \( r \), \( (r, x) \in \text{Head}^\mathcal{P} \), for each \( y \) in \( \text{body}^+(r) \), \( (r, y) \in \text{Body}^+\mathcal{P} \), and for each \( z \) in \( \text{body}^-(r) \), \( (r, z) \in \text{Body}^-\mathcal{P} \). Also, we say \( M \) represents a stable model of \( P \) if unary relation \( \text{Stable}^\mathcal{M} \) is a stable model of \( P \).

**Preferred Models**

Prioritized logic program (PLP) [113] is one of the impactful frameworks proposed for logic programming with preferences. For many preference-based logic programming proposals in the literature that define preferences over atoms, PLP has been a source of inspiration [73, 36, 30]. A PLP program is a pair \((Pr, \Phi)\) where \( Pr \) is a general extended disjunctive logic program with answer set semantics and \( \Phi \) is a set of preference relations among propositional atoms of the form \( a \succeq b \) that means \( a \) is preferred to \( b \). The transitive closure of \( \Phi \) is denoted by \( \Phi^\bullet \). The reflexive transitive binary relation \( \sqsupseteq \) among answer sets of \( Pr \) is defined as:

1. \( X_1 \sqsupseteq X_1 \).
2. If there exist \( a \in X_1 - X_2 \) and \( b \in X_2 - X_1 \) where \( (a \succeq b) \in \Phi_c \) and there is no \( d \in X_1 - X_2 \) such that \( (b \triangleright d) \in \Phi_c \), then \( X_1 \sqsupset X_2 \), and
3. If \( X_1 \sqsupset X_2 \) and \( X_2 \sqsupset X_3 \), then \( X_1 \sqsupset X_3 \).

\( X \) is called a preferred answer set if there is no answer set \( Y \) such that \( Y \sqsubseteq X \). One could examine conditions 1 and 2 in polynomial time in the size of the input (the number of propositional atoms). However, condition 3 requires possibly an exponential number of comparisons over the answer sets of \( Pr \). The complexity results of the decision problems associated with a PLP program are based on the assumption that deciding whether \( X \) is preferable to \( Y \) is in polynomial time that is not accurate due to condition 3.

The role of condition 3 is to make relation \( \sqsupset \) transitive. On the other hand, relation \( \triangleright^\mathcal{P} \) in the Prioritized Model Expansion framework is not necessarily transitive for some preference semantics.
could decide whether an answer set \( \Gamma = (Pr, \Pi, \sigma, \psi) \) is preferable to an answer set \( \Phi \) with respect to the preference expression \( \sigma \). Therefore, if condition 3 is removed, then the following holds: For a PLP program \( \Gamma = (Pr, \Phi_c) \), there is a Prioritized Model Expansion problem \( \Pi_{\sigma, \psi} = (MX_{\sigma, \psi}, P) \) with an input \( \sigma \)-structure \( I \) where \( I \) represents \( Pr \), and \( P \) specifies \( \Phi_c \). If there are expansion structures \( A \) and \( B \), then \( Pr \) has answer sets \( M_1 \) represented by \( A \) and \( M_2 \) represented by \( B \) such that if \( A >_P B \), then \( M_1 \sqsupset M_2 \).

**Proof.** \( \Phi_c \) can be viewed as a preference expression in the PMX framework and finding answer sets of generalized extended disjunctive program \( Pr \) can be expressed by an MX,\( \sigma, \psi \) problem. As discussed in the previous subsection, there is a correspondence between the expansion structures of \( MX_{\sigma, \psi} \) and the answer sets of \( Pr \). Each expansion structure of \( MX_{\sigma, \psi} \) can be one-to-one mapped in polynomial time to an answer set of \( Pr \) and vice versa. The relation \( >_P \) with the Element Dominance semantics is a subset of relation \( \sqsupset \) in the PLP because it satisfies conditions 1 and 2 in the definition of \( \sqsupset \) while it does not guarantee the transitivity of the preference relation among the expansion structures. So, for structures \( A \) and \( B \), \( Pr \) has answer sets \( M_1 \) and \( M_2 \) such that \( M_1 \sqsupset M_2 \) and \( M_1 \) and \( M_2 \) are represented by \( A \) and \( B \), respectively.

It is worth mentioning that in the definition of relation \( \sqsupset \) in the PLP, if we drop condition 3, which guarantees the transitivity of \( \sqsupset \), then the complexity results of [113] are applicable. One could decide whether an answer set \( X \) is preferable to an answer set \( Y \) by a direct comparison between \( X \) and \( Y \). Therefore, if condition 3 is removed, then the following holds: For a PLP program \( \Gamma = (Pr, \Phi_c) \), there is a Prioritized Model Expansion problem \( \Pi_{\sigma, \psi} = (MX_{\sigma, \psi}, P) \) with an input \( \sigma \)-structure \( I \) where \( I \) represents \( Pr \), \( \psi \) characterizes the stable model semantics, and \( P \) is a preference expression representing \( \Phi_c \) such that \( A \) is an optimal expansion of \( \Pi_{\sigma, \psi} \) if and only if there is a preferred answer set \( M \) in \( \Gamma \) that is represented by \( A \). The computational complexity of brave reasoning (i.e., deciding the existence of an answer set that satisfies a conjunction of atoms) in general extended disjunctive logic program \( Pr \) is \( \Sigma_2 \) -complete and the complexity of brave reasoning in \( \Gamma = (Pr, \Phi_c) \) is \( \Sigma_2 \) -complete [113]. Therefore, deciding the existence of an optimal expansion of \( \Pi_{\sigma, \psi} \) that satisfies a goal \( \phi \) (Goal-oriented Prioritized Model Expansion problem) is \( \Sigma_2 \) -complete.

### 3.3.3 Answer Set Optimization

An Answer Set Optimization (ASO) program [36] is a pair \( (P_g, R) \) where \( P_g \) is a generating normal logic program and \( R \) is a set of rules of the form \( r : C_1 > ... > C_k \leftarrow a_1, ..., a_n, \text{not } b_1, ..., \text{not } b_m \). In each rule, \( a_i \) and \( b_i \) are literals. Also, \( C_i \) is defined as a combination of atoms integrated by conjunction, disjunction, default negation (not) and strong negation (¬) that must appear only before atoms. \( C_i > C_j \leftarrow \text{body} \) means that if \( \text{body} \) is satisfied, \( C_i \) is preferred to \( C_j \). Given a set of \( l \) rules \( r_1, ..., r_l \), each answer set \( M \) of \( P_g \) is associated with a satisfaction vector \( d(M) = \)
satisfies a goal formula $\phi$ programs with stable model semantics), and determining the existence of a solution of ASO that of an answer set that satisfies a set of ground atoms is NP-complete (i.e., brave reasoning in logic to deciding whether some answer sets of an ASO satisfy a formula in polynomial time. The existence 1, is $\Sigma^p_2$-complete. 

Let $\text{ASO} = (P_g, R)$ be an ASO program where $P_g$ is a normal logic program and $R$ is a set of preference rules. There is a Prioritized Model Expansion problem $\Pi_{\sigma,\psi} = (\text{MX}_{\sigma,\psi}, P)$ with an input $I$ such that each preferred answer set of ASO is represented by an optimal expansion of $\Pi_{\sigma,\psi}$.

Proof. Let $P_g^*$ be a logic program such that $P_g^* = P_g \cup R^*$ where $R^*$ is a set of $r_1^*$ and $r_2^*$ rules that are constructed as follows: For each rule $r$ in $R$ of the form $C_1 > C_2 \leftarrow \text{body}(r)$, we introduce auxiliary atoms $n_1$ and $n_2$ and define $r_1^*: n_1 \leftarrow C_1$, body$(r)$ and $r_2^*: n_2 \leftarrow C_2$, body$(r)$. Normal logic program $P_g^*$ with stable model semantics can be cast as Model Expansion problem $\text{MX}_{\sigma,\psi}$ such that $\psi$ specifies stable model semantics and $I$ represents $P_g^*$. Also, let us define preference expression $P$ such that for auxiliary atoms $n_1$ and $n_2$, we have $n_2 \sqsupseteq_P n_1$. We consider all other ground atoms in $P_g$ as equally preferred. Let $M_1$ and $M_2$ be answer sets of $P_g$. Assume $A$ and $B$ are expansion structures of $\text{MX}_{\sigma,\psi}$ that represent $M_1$, and $M_2$, respectively. If $M_2 \geq M_1$ in ASO, for each $C_i$ in the head of each rule $r \in R$ such that $M_1 \models C_i$, there is $C_j$ such that $M_2 \models C_j$ and $C_j > C_i$. This is equivalent to say that for each auxiliary atom $n_i$ satisfied by $A$, there is an auxiliary atom $n_j$ satisfied by $B$ such that $n_j \sqsupseteq_P n_i$ that is matched with the Upper Bound Dominance semantics. So, $A >_{\text{UBD}}^P B$ and the result follows.

We showed that an ASO program can be viewed as a Prioritized Model Expansion. Also every Prioritized NP-complete Model Expansion problem with the Upper Bound Dominance semantics can be encoded as an ASO program. The problem of deciding the existence of a stable model of a normal program is NP-complete and, obviously, an NP-complete problem can be reduced in polynomial time to another NP-complete problem. Preference expression $P$ is encoded by a single rule $r \in R$ of the form $a_1 > a_2, \ldots > a_n \leftarrow$ where $a_i$s are ground atoms and $a_i > a_j$ if $a_i \sqsupseteq_P a_j$.

As a result, Goal-oriented Prioritized NP-complete Model Expansion problems can be reduced to deciding whether some answer sets of an ASO satisfy a formula in polynomial time. The existence of an answer set that satisfies a set of ground atoms is NP-complete (i.e., brave reasoning in logic programs with stable model semantics), and determining the existence of a solution of ASO that satisfies a goal formula $\phi$ is $\Sigma^p_2$-complete [36]. Likewise, for an NP-complete $\text{MX}_{\sigma,\psi}$, the problem of deciding whether there is an optimal expansion of $\Pi_{\sigma,\psi}$ that satisfies a goal $\phi$, based on Theorem 1, is $\Sigma^p_2$-complete.
3.3.4 Some More Examples

Our proposal can also be related to a variety of other preference frameworks, such as CP-nets [20] that model conditional preferences. A CP-net can be approximated by a Prioritized Model Expansion problem such that if an outcome \( o_1 \) is preferred to an outcome \( o_2 \) in a CP-net, then for associated expansion structures \( A \) and \( B \) in the Prioritized Model Expansion problem, \( A \) is preferred to \( B \). A similar approach was taken in [36]. Finding preferred stable models of first-order logic programs with ordered disjunction, which were characterized as structures satisfying a second-order logic formula in [3], can be encoded as a Prioritized Model Expansion problem using the similar idea of reducing an ASO to a Prioritized Model Expansion problem. Moreover, finding preferred repairs of a database that violates some integrity constraints [124] can be translated into a Prioritized Model Expansion problem using the Upper Bound Dominance semantics.

3.4 Conclusion

We proposed a novel language-independent preference framework and connected it to Model Expansion for characterizing preference-based computational decision and search problems. We demonstrated that adding preferences raises the computational complexity of deciding the existence of an expansion structure satisfying a goal. In Chapter 5, we will devise an algorithm that solves Prioritized Model Expansion problems using generic solvers empowered by propagators. The solver would provide symbolic explanations for rejecting and accepting models, and would follow a preferred computation path to prune the search space.
Chapter 4

Modular System with Preferences

We propose a versatile framework for combining knowledge bases in modular systems with preferences. In our proposal, each module (knowledge base) can be specified in a different language. We define the notion of a preference-based modular system that includes a formalization of meta-preferences. We show that our framework is robust in the sense that the operations for combining modules preserve the notion of a preference-based modular system. Finally, we formally demonstrate relationships between our framework and the related preference formalisms of CP-nets, Answer Set Optimization, and preference-based planning. Our framework allows one to use these preference formalisms (and others) in combination, in the same modular system.

4.1 Introduction

Combining knowledge bases (KBs) is very important when common sense reasoning is involved. For example, in planning, we may want to combine temporal and spatial reasoning, or reasoning from the point of view of several agents. Here, we consider each knowledge base, called module, as a set of structures. Modular Systems (MS) [127] is a framework to combine heterogeneous knowledge bases. Modules are combined through the operators of composition, union, projection, and feedback. An algorithm for finding models of MSs was proposed in [128]. An improved version of the algorithm, in the same paper, uses approximations to reduce the search space. Connections to Satisfiability Modulo Theory and other systems were discussed in [128]. In [99], lazy clause generation has been applied to decide whether a given modular system has any models.

An important aspect of knowledge representation systems is the capability to represent preferences. The literature presents a variety of approaches to formalize preferences, e.g., [25], [114], [52], [37], [117], [20], [53], [136] and [65]. Several surveys have appeared in recent years [25], [37], [5], and [48] categorizing preference formalisms from various perspectives. For example, in [5], a set of preference formalisms for planning were introduced. The authors of [48] classified preference frameworks in non-monotonic reasoning. Preferences in database systems have also been studied by different researchers such as [84], [18] and [125]. A primary well-known preference language in database systems was proposed in [84]. In this language, some preference constructors were in-
troduced to express basic preference terms. For example, consider the following statement: When buying a car, it is preferable to buy a BMW or Mercedes-Benz rather than another type of car. This statement can be expressed by a constructor that is called positive preference (POS). In the aforementioned example, POS includes BMW and Mercedes-Benz. Formally speaking, given two \( n \)-ary tuples \( a = (a_1, \ldots, a_n) \) and \( b = (b_1, \ldots, b_n) \) and a set called POS, \( a \) is preferred to \( b \) (notation \( a >_P b \)) with respect to the \( i \)th attribute (column) in a database table if \( a[i] \in \text{POS} \) and \( b[i] \notin \text{POS} \). Likewise, \( P \) is an NEG preference and \( a >_P b \) if \( a[i] \notin \text{NEG} \) and \( b[i] \in \text{NEG} \). There are a number of methods, such as Pareto and prioritized accumulation, for combining preferences when there are multiple preference statements [84]. The pareto operator combines two preferences such that \( A \) is preferred to \( B \) with respect to the composition of \( P_1 \) and \( P_2 \) (notation \( P = P_1 \otimes P_2 \)) if \( A \) is preferred to \( B \) with respect to at least one of preferences \( P_1 \) and \( P_2 \) such that \( [(A >_{P_1} B) \land \neg(B >_{P_2} A)] \lor [(A >_{P_2} B) \land \neg(B >_{P_1} A)] \). Prioritized accumulation that is denoted by \& gives priority to a preference. \( A \) is preferred to \( B \) (notation \( A >_{P_1 \& P_2} B \)) if \( A >_{P_1} B \lor \neg(A >_{P_1} B) \land \neg(B >_{P_1} A) \land (A >_{P_2} B) \).

In practical settings, systems such as web services, planners, business process controllers, and so on often consist of intricate interconnected parts. Each component may interact with other parts and its associated knowledge base may be updated during the execution of a process. Current frameworks are incapable of modelling preferences in such dynamic environments. For example, in [32] and [36], preferences are defined by rules in Answer Set Programming (ASP). However, it is not clear how a knowledge base of ASP preferences and rules can be connected to other preference-based knowledge bases. Similarly, in [138], a language for preference representation and inference is proposed, but the framework does not address how to reason about preferences when preferences are combined. Some attempts have been made to combine preferences, such as in [111], [84], and [1]. But these proposals have two main shortcomings: 1) Preferences of two components can be combined only when their languages are the same. For instance in [111] and [84] preferences are expressed in first-order logic. So, these languages cannot model heterogeneous data systems, such as web services when each part may have its own language. 2) Two components are sequentially composed. Nonetheless, many real world systems have a more complicated structure (see the feedback connection in Example 1 below). The following example shows the complexity of formalizing a modular system with preferences.

**Example 7.** A Logistic Service Provider (LSP) is a modular system that can be used by a company that provides logistic services. It decides how to pack goods and deliver them. It solves two NP-complete tasks interactively, – the multiple Knapsack (module \( M_K \)) and the Traveling Salesman Problem (module \( M_{TSP} \)). The system takes orders from customers. Each order is specified by inputs Items\((i)\) that denotes the item to be delivered, \( p(i) \) that specifies the price of item \( i \), \( w(i) \) that indicates the weight of item \( i \), and \( c(t) \) that stands for the capacity of truck \( t \). By solving a multiple Knapsack problem in module \( M_K \), a candidate solution that is a valid packing of items is sent to module \( M_{TSP} \) that solves a TSP problem by using the output of \( M_K \) together with the graph of
cities and routes with distances, allowable distance limit, and destinations for each product. The output of this module is a route for a truck specified by \( \text{Route}(t, n, c_r) \) where \( t \) is a truck, \( n \) is the number in the sequence, and \( c_r \) is a city. The Knapsack problem is written, in, e.g., Integer Linear Programming (ILP), and the TSP problem in Answer Set Programming (ASP). Modules \( M_K \) and \( M_{TSP} \) are composed in sequence, with feedback going from an output of \( M_{TSP} \) to an input of \( M_K \) deciding whether a package of items in Module \( M_K \) is suitable for the capacity of truck \( t \). A solution to the compound module, \( M_{LSP} \), to be acceptable, must satisfy both sub-systems.

Also, the company may have preferences for packing and delivery of products. For example, if a fragile item is packed in a truck, it may be preferable to exclude heavy items or, among certain routes with equal costs, some of them may be preferable to others. Assume that preferences in the Knapsack problem are formalized by CP-nets [20] and the TSP’s preferences are represented in the preference-based Answer Set Programming framework [36]. In Figure 1, \( P_k \) denotes the preferences of the Knapsack module and \( P_{TSP} \) denotes the TSP module’s preferences. Formalizing this modular problem with preferences is not easy because: 1) the Knapsack and the TSP are axiomatized in different languages, 2) preferences of each module are represented by a different formalism, 3) preference formalisms use different languages than the axiomatizations of the modules themselves, and 4) two modules communicate in a complex way that includes a feedback loop from \( M_{TSP} \) to \( M_K \).

![Figure 4.1: Logistics Service Provider \( M_{LSP} = (M_K \triangleright M_{TSP})[B = B'] \)](image)

4.1.1 Contributions

We propose a model-theoretic foundation for combining KBs with preferences in modular systems. On the logic level, each module is represented by a KB in some logic \( L^1 \), and its preferences (and meta-preferences) are represented by (strict) partial orders on partial structures in a preference formalism named \( \mathcal{P}-\mathcal{M}S \). Different logics and preference formalisms can be used for modules in the same system. Operations for combining modules are generalized to combine preferences of each module. We prove that our formalism is closed in the sense that the operations for combining modules preserve the notion of preference-based modular systems. Our formalism is consistent with

---

1Any logic with model-theoretic semantics can be used, including logic programs.
(and extends) the model-theoretic semantics of modular systems [127]. In model theoretic semantics, each module is viewed as a set of structures. We also show the relationship between our formalism and CP-nets, Answer Set Optimization, and preference-based planning. Thus, we can combine them in one modular system.

4.1.2 Novelty

With our formalism, each module can be formalized in a different framework. To our knowledge, this is the first multi-language preference formalism. This generality is achieved through the model-theoretic semantics of modular systems. Another novelty is the ability to handle preferences when knowledge bases are combined in complex ways. For instance, in Example 1, there is a complex combination of Knapsack and TSP problems (feedback from TSP to Knapsack). In contrast, these complex systems were not representable in previous work, e.g., in [84].

4.1.3 Chapter Structure

After covering preliminaries, we discuss how preferences in an atomic module are represented. Then we extend the idea to modular systems and study preference specifications in modular systems when some preferences of their components are given. After that, we formally analyze the relation between three well-known preference formalisms and our proposed approach. We conclude the chapter with a summary and future work.

This chapter significantly extends IJCAI paper [61], IfCoLog journal paper [63], and [62] and by adding complete and detailed proofs and, by novel examples, illustrates how the proposed formalism generalizes and combines other approaches to handling preferences.

4.2 Algebra of Modular Systems

Based on the model theoretic-semantics of modular systems [127], a module is defined as a set (class) of structures. A module can be given by any decision procedure, be a set of models of a KB, be given by an inductive definition, be a C or an ASP program, or be given by an agent making decisions. Modules that have inputs and outputs are very common. Many software programs and hardware devices are of that form. In the Logistics Service Provider (Example 7), for example, users’ requests could be the input, and the truck route and packing solutions the output. Fixing input and output vocabularies in some modules allows us to talk about the Model Expansion (MX) task [100].

**Definition 11** (Module). A module is a set of $\sigma \cup \varepsilon$-structures, where $\sigma$ is the input vocabulary and $\varepsilon$ is the output vocabulary and $\sigma \cap \varepsilon = \emptyset$.

Modules of [128] were introduced as Model Expansion tasks. Recall that in [100], the authors formalize combinatorial search problems as the task of Model Expansion (MX), the logical task of expanding a given structure with new relations. Formally, the user axiomatizes the problem in some
logic $L$. This axiomatization relates an instance of the problem (a finite structure) to its solutions (expansions of that structure with new relations or functions). Logic $L$ corresponds to a specification/modelling language. It could be an ASP program, or a specification in a language from the CP community, or even a Java program, as long as model-theoretic semantics can be provided. Complexity-wise, MX lies in between model checking (full structure is given) and satisfiability (no structure is given). The task generalizes to the formalism of Modular Systems. A fixpoint semantics was also defined for modular system [127] in which modular systems act as operators on structures.

Modules are combined using the following algebraic operations: Composition ($M_1 \triangleright M_2$) connects outputs of $M_1$ to inputs of $M_2$. Union ($M_1 \cup M_2$) models choice. Projection ($\pi_M(M)$) hides some vocabulary of a module. Feedback ($M[R = S]$), which was inspired by feedbacks in logical circuits, connects output $S$ of $M$ to its input $R$. Intuitively, the operations correspond to conjunction, disjunction, and existential quantifier. Feedback represents fixpoints (not necessarily minimal) of modules viewed as operators. One can introduce other operations, e.g., least fixpoints or combinations of the operations above.

We shall now define the syntax of the algebra of modular systems. Following [80], we say modules $M_1$ and $M_2$ are composable if $\varepsilon_{M_1} \cap \varepsilon_{M_2} = \emptyset$ (that insures there is no interference between outputs). Module $M_2$ is independent from $M_1$ if $\sigma_{M_2} \cap \varepsilon_{M_1} = \emptyset$ (that guarantees there is no cyclic dependency between input and outputs). A module is primitive if the only sub-module (algebraic sub-formula) of it is itself. Well-formed modular systems $MS(\sigma, \varepsilon)$, with instance ($\sigma$) and expansion ($\varepsilon$) vocabularies, are defined recursively as follows:

- If $M$ is a primitive module with instance (input) vocabulary $\sigma$ and expansion (output) vocabulary $\varepsilon$, then $M \in MS(\sigma, \varepsilon)$.
- If $M \in MS(\sigma, \varepsilon)$, $\tau \subseteq \sigma \cup \varepsilon$, then $\pi_{\tau}(M) \in MS(\sigma \cap \tau, \varepsilon \cap \tau)$.
- If $M \in MS(\sigma, \varepsilon)$, $M' \in MS(\sigma', \varepsilon')$, and $M$ is composable with and independent from $M'$, then $(M \triangleright M') \in MS(\sigma \cup \sigma', \varepsilon \cup \varepsilon')$.
- If $M \in MS(\sigma, \varepsilon)$, $M' \in MS(\sigma', \varepsilon')$, and they are independent, then $(M \cup M') \in MS(\sigma \cup \sigma', \varepsilon \cup \varepsilon')$.
- If $M \in MS(\sigma, \varepsilon)$, $R \in \sigma$, $S \in \varepsilon$, and $R$ and $S$ are of the same type and arity, then $M[R = S] \in MS(\sigma \setminus \{R\}, \varepsilon \cup \{R\})$.

We may use the terms primitive module and atomic module interchangeably. A modular system is given by an algebraic formula, with input-output vocabulary specified for each primitive module. Subsystems correspond to sub-formulas and are modules themselves.

**Model-theoretic semantics** associates a set of structures with each modular system. Each such structure is called a model of the modular system. The semantics does not put any finiteness restriction on the domains. Thus, the framework works for modules with infinite structures. However, in this thesis, we assume that all structures have a finite domain that is included in a universal domain $U$.

**Definition 12 (Models of a Modular System).** Let $M \in MS(\sigma, \varepsilon)$ be a modular system and $B$ be a $(\sigma \cup \varepsilon)$-structure. We construct the set of models of module $M$ by structural induction on the structure of a module.
Primitive Module: $\mathcal{B}$ is a model of $\mathcal{M}$ if $\mathcal{B} \in \mathcal{M}$.

Projection: $\mathcal{B}$ is a model of $\mathcal{M} := \pi_{(\sigma \cup \varepsilon)}(\mathcal{M}')$ (with $\mathcal{M}' \in \text{MS}(\sigma', \varepsilon')$) if a $(\sigma' \cup \varepsilon')$-structure $\mathcal{B}'$ exists such that $\mathcal{B}'$ is a model of $\mathcal{M}'$ and $\mathcal{B}'$ expands $\mathcal{B}$.

Composition: $\mathcal{B}$ is a model of $\mathcal{M} := \mathcal{M}_1 \triangleright \mathcal{M}_2$ (with $\mathcal{M}_1 \in \text{MS}(\sigma_1, \varepsilon_1)$ and $\mathcal{M}_2 \in \text{MS}(\sigma_2, \varepsilon_2)$) if $\mathcal{B}|_{(\sigma_1 \cup \varepsilon_1)}$ is a model of $\mathcal{M}_1$ and $\mathcal{B}|_{(\sigma_2 \cup \varepsilon_2)}$ is a model of $\mathcal{M}_2$, where $\mathcal{A}|_\tau$ is the projection of structure $\mathcal{A}$ into vocabulary $\tau$.

Union: $\mathcal{B}$ is a model of $\mathcal{M} := \mathcal{M}_1 \cup \mathcal{M}_2$ (with $\mathcal{M}_1 \in \text{MS}(\sigma_1, \varepsilon_1)$ and $\mathcal{M}_2 \in \text{MS}(\sigma_2, \varepsilon_2)$) if $\mathcal{B}|_{(\sigma_1 \cup \varepsilon_1)}$ is a model of $\mathcal{M}_1$ or $\mathcal{B}|_{(\sigma_2 \cup \varepsilon_2)}$ is a model of $\mathcal{M}_2$.

Feedback: $\mathcal{B}$ is a model of $\mathcal{M} := \mathcal{M}'|_{R = S}$ (with $\mathcal{M}' \in \text{MS}(\sigma', \varepsilon')$) if $\mathcal{R}^\mathcal{B} = \mathcal{S}^\mathcal{B}$ and $\mathcal{B}$ is a model of $\mathcal{M}'$.

We remark that for sequential composition, the direction of information propagation is specified by distinguishing the vocabulary of input and output. For example, the Knapsack-TSP system in Example 1 is formalized as $\mathcal{M}_{\text{LSP}} = [\mathcal{M}_K \triangleright \mathcal{M}_{\text{TSP}}]|_{B = B'}$ where the input vocabulary of $\mathcal{M}_K$ is $\sigma_{\mathcal{M}_K} = \{\text{Item}, p, w, c, B\}'$, the output vocabulary of $\mathcal{M}_K$ is $\varepsilon_{\mathcal{M}_K} = \{\text{Pack}\}$, the input vocabulary of $\mathcal{M}_{\text{TSP}}$ is $\sigma_{\mathcal{M}_{\text{TSP}}} = \{\text{Pack}, E, L, \text{Dest}\}$, and the output vocabulary of $\mathcal{M}_K$ is $\varepsilon_{\mathcal{M}_K} = \{\text{Route}, B\}$.

Also, the input vocabulary of modular system $\mathcal{M}$ is $\sigma_{\mathcal{M}} = \{\text{Item}, p, w, c, E, L, \text{Dest}\}$ and the output vocabulary of $\mathcal{M}$ is $\varepsilon_{\mathcal{M}} = \{\text{Pack}, \text{Route}\}$.

Partial structures allow interpretation of some vocabulary symbols to be partially specified [127].

Definition 13. For $\delta \subseteq \tau$, $\mathcal{B}$ is a $\delta$-partial $\tau$-structure if for any $n$-ary predicate symbol $S \in \tau \setminus \delta$, the interpretation of $S$ is total, and for any $n$-ary $T \in \delta$ and for all $\mathbf{t} \in [\text{dom}(\mathcal{B})]^n$, $\mathcal{T}^{\mathcal{B}}$ is characterized by auxiliary vocabulary symbols $T^+$ and $T^-$ such that $\mathbf{t} \in T^{\mathcal{B}}$ if $\mathbf{t} \in T^\mathcal{B}$ and $\mathbf{t} \in T^{-\mathcal{B}}$ if $\mathbf{t} \notin T^\mathcal{B}$.

In fact, $\mathcal{B}$ totally interprets the symbols in $\tau \setminus \delta$ while the interpretation of the symbols in $\delta$ is partial. $\mathcal{T}^+$ denotes the interpretation of all $T^+$ by $\mathcal{B}$ where $T \in \tau$. Also, $\mathcal{T}^-$ is, similarly, defined based on the interpretation of $T^-$ for each $T \in \tau$. Let us define the notion of extension that is a relation between two partial structures.

Definition 14. For two partial structures $\mathcal{A}$ and $\mathcal{B}$ over the same vocabulary and domain, we say that $\mathcal{B}$ extends $\mathcal{A}$ if $\mathcal{A}^+ \subseteq \mathcal{B}^+$ and $\mathcal{A}^- \subseteq \mathcal{B}^-$.

Notation 4. Let $\mathcal{A}$ be a $\delta$-partial $\tau$-structure that interprets $\delta \subseteq \tau$ partially. $\mathcal{A}$ is represented by a set of ground atoms $X = \{a_1, \ldots, a_n\}$ where for each $n$-ary $R \in \delta$, if $\mathbf{t} \in R^\mathcal{A}$ or $\mathbf{t} \notin R^\mathcal{A}$, there is $a_i \in X$ such that either $a_i = R^+(\mathbf{t})$ or $a_i = R^-(\mathbf{t})$. We say $X$ is the representation of $\mathcal{A}$ or simply partial structure $\mathcal{A}$.

4.3 $\mathcal{P}$-MS: Preference-based Modular Systems

In this section, we define preference $\mathcal{P}$ in a primitive module $\mathcal{M}$. Then, an atomic relation $\triangleright_{\mathcal{P}}$ is introduced to compare structures in $\mathcal{M}$ with respect to $\mathcal{P}$. After that, we consider a primitive module
with a set of preferences \( \Pi = \{ P_1, \ldots, P_n \} \) and a preference relation \( M\mathcal{P} \) on elements of \( \Pi \). In this setting, a preferred structure is defined based on binary relation \( \succeq_{M\mathcal{P}} \). For modular systems, we introduce the notion of compound preference relations constructed by modular systems operators. A modular system with preference relations (atomic or compound) is called a preference-based modular system. We show that when we combine modular systems with preferences, the result is also a preference-based modular system.

To have a formalism compatible with model theoretic-semantics of modular systems, we aim to define preference relations over structures. However, using structures to model preferences is not always practical because a decision maker usually does not have complete information about the world. Informally speaking, some interpreted symbols may be preferred to others, and there might not be enough information to decide about the rest. Unlike structures, partial structures interpret a subset of vocabulary symbols, while the interpretation of other symbols is unknown. In our formalism, a preference statement is an order over a set of partial structures when certain conditions hold.

To unify notations, we first define strict partial order and strict total order in the definition below.

**Definition 15.** A strict partial order \( O \) over a set \( S \) is a pair \( O = (S, \prec) \) such that \( \prec \) is a binary relation over elements of \( S \) that is anti-reflexive, asymmetric, and transitive. If for all \( S, S' \in S \), either \( S \prec S' \) or \( S' \prec S \), then \( O \) is called a strict total order.

Preference statements in natural language are often represented as conditional statements, e.g., Mary prefers to buy a Ford over a Toyota if the car’s body-type is SUV or coupe. If it is sedan, Mary prefers Toyota over Ford. We model this statement by a pair \( P = (O, \Gamma) \) where \( \Gamma \) represents the premise of the statement (i.e., if body-type is SUV or coupe) and \( O \) denotes the conclusion (i.e., Ford is preferred to Toyota).

Let \( U \) be a finite domain of elements. We define primitive preferences for an atomic module as follows:

**Definition 16.** Let \( M \in \text{MS}(\sigma, \varepsilon) \) be an atomic module and \( \text{vocab}(M) = \tau \). A \( \tau_o \)-preference (or, simply, preference) \( P = (O, \Gamma) \) in \( M \) is a pair where \( O = (S, \prec) \) is a strict total order over \( S \) that is a set of partial structures \( A \) where \( A \) is a \( \tau_o \)-partial \( \tau_o \)-structure with domain \( U \) and \( \tau_o \subseteq \tau \). As well, \( \Gamma = \{ \gamma_1, \gamma_2, \ldots, \gamma_m \} \) is a set of \( \tau_{p_i} \)-partial \( \tau \)-structures with domain \( U \) where \( \tau_{p_i} \subseteq \tau \), for \( 1 \leq i \leq m \).

For a primitive module \( M \in \text{MS}(\sigma, \varepsilon) \), it is intuitive to define preferences over (partial) interpretations of the output vocabulary. For example, consider a primitive module \( M \in \text{MS}(\{E\}, \{R, G, B\}) \) that solves the three-colouring problem. The interpretation of the input vocabulary, which is \( \{E\} \), in an input graph \( G \) is the same for all possible three-colourings of \( G \). Hence, we define preferences on \( \tau_o \)-partial structures where \( \tau_o \subseteq \varepsilon \). There is no such restriction for the vocabulary of \( \Gamma \) since preference over outputs may vary based on different graph structures. In the buying a car scenario, based on the above definition, \( \Gamma \) is a set of partial structures \( \Gamma = \{\text{Type(SUV)}, \text{Type(coupe)}\} \) and
\( \mathcal{O} = \{ \text{Model(Ford)} \succ \text{Model(Toyota)} \} \). The size of \( \mathcal{S} \) can be an exponential factor of the size of domain \( U \). Hereafter, we assume that the size of \( \mathcal{S} \) is bounded by a given constant \( K \).

By slight abuse of notation, we say a \( \tau \)-structure \( \mathcal{A} \) extends a partial \( \delta \)-structure \( \mathcal{B}_j \) if \( \mathcal{A}|_{\delta} \) extends \( \mathcal{B}_j \). Binary relation \( \succ_{\mathcal{P}} \) is defined as follows:

**Definition 17.** Let \( M \) be an atomic module, and \( \mathcal{B} \) and \( \mathcal{B}' \) be two structures in \( M \). Given a \( \tau_0 \)-preference \( \mathcal{P} = (\mathcal{O}, \Gamma) \) in \( M \), let \( \Delta \) be the set of all structures in \( M \) that extend at least one member of \( \Gamma \). We say structure \( \mathcal{B} \) is preferred to \( \mathcal{B}' \) with respect to \( \mathcal{P} \) (denoted by \( \mathcal{B} \succ_{\mathcal{P}} \mathcal{B}' \)) if and only if 1) \( \mathcal{B}, \mathcal{B}' \in \Delta \), 2) there are partial structures \( \mathcal{B}_i \) and \( \mathcal{B}_j \) in \( \mathcal{S} \) that can be extended to structures in \( M \) such that \( \mathcal{B}_i \succ \mathcal{B}_j \), and \( \mathcal{B} \) is an extension of \( \mathcal{B}_i \), where \( \mathcal{B}' \) extends \( \mathcal{B}_j \), and 3) there are no partial structures \( \mathcal{B}_k \) and \( \mathcal{B}_m \) in \( \mathcal{S} \) such that \( \mathcal{B} \) and \( \mathcal{B}' \) extend them, respectively, and \( \mathcal{B}_m \succ \mathcal{B}_k \).

This definition states that when a part of \( \mathcal{B} \) is preferred to \( \mathcal{B}' \), if a condition specified by \( \Gamma \) is satisfied by both structures, we can conclude that \( \mathcal{B} \) is preferred to \( \mathcal{B}' \). It makes no difference how the rest of the vocabulary is interpreted because it is irrelevant to \( \mathcal{P} \). For a fixed vocabulary \( \tau \), the set \( \Gamma \) can be cast as a class of isomorphic \( \tau \)-structures that are models of a (set of) variable-free specification(s) \( \phi \) (notation \( \text{Mod}(\phi) \)) where \( \tau = \text{vocab}(\phi) \). In other words, \( \tau \)-structures \( \mathcal{A} \) and \( \mathcal{B} \) can be compared based on \( \mathcal{P} \) only if both of them satisfy \( \phi \).

**Example 8.** In Example 7, consider that safely delivering items is an important preference for the company. So, it is preferable to avoid packing heavy and light items together to reduce the risk of damage to the light items. The input structure specifies price, weight, and capacity of each item. Let \( \mathcal{P}_{\text{safe}} = (\mathcal{O}_{\text{safe}}, \Gamma_{\text{safe}}) \) be the safety preference where \( \mathcal{O}_{\text{safe}} = (\mathcal{S}, \prec) \) is a partial order over \( \mathcal{S} \) that is a set of \{pack\}-partial structures. Each element of \( \mathcal{S} \) specifies some items that can be placed in the pack. Relation "\( \prec \)" is defined as \{pack(i) \( \prec \) pack\(^{-}\)(i)\}. It means that it is preferable to not put item \( i \) in the pack. According to Notation 4, pack\(^{-}\)(i) is a representation of a partial structure that interprets ground atom pack(i) as false. The premise of the conditional statement is formalized by \( \Gamma_{\text{safe}} = \{ \mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_m \} \). Each \( \mathcal{C}_i \in \Gamma \) satisfies

\[
\forall x \left( (\text{item}(x) \land x = i) \supset \left[ (w(x) > W \land \exists y (\text{item}(y) \land w(y) < W') \lor (w(x) < W' \land \exists z (\text{item}(z) \land w(z) > W)) \right] \right).
\]

According to this formula and safety preference \( \mathcal{O}_{\text{safe}} \), for constant weights \( W \) and \( W' \), when an item \( i \) is heavier than \( W \) and there is an item lighter than \( W' \), it is preferable to avoid putting \( i \) in the pack. Similarly, if item \( i \) is lighter than \( W' \) and there is an item heavier than \( W \) in the pack, it is not preferable to put \( i \) in the pack.

POS preferences can be represented by a preference \( \mathcal{P} \) in our formalism. The idea is to characterize each tuple of a database table as a structure. The following example shows how a POS is encoded.
Example 9. Imagine that we want to buy a car from a dealership. Assume the POS set includes BMW and the POS preference indicates BMW cars are preferred to others. In the context of modular systems, we consider a module $M$ that specifies a set of available cars in the dealership. Each structure of the module characterizes properties of a car such as model, color, and body-type. Suppose $Model$ is a unary relation (predicate) that indicates a car model, e.g., $Model(BMW)$. Recall that the interpretation of an $n$-ary predicate $R$ is a set of $n$-ary tuples $\mathcal{T}$. Each tuple in a database table can be viewed as a structure where the interpretation of each unary predicate, which represents an attribute, is singleton.

Assume $\mathcal{P} = (\mathcal{O}, \Gamma)$ where $\mathcal{O} = (\mathcal{S}, \prec)$ is an order over car properties such that $\prec$ is defined as: $Model(BMW) \succ Model^-(BMW)$. According to Notation 4, this statement means that all structures that interpret $Model$ as $Model(BMW)$ are preferred. If car $A$ is specified as $A = \{Model(BMW), Color(white), Body-type(sedan)\}$ and for car $B$ we have $B = \{Model(Benz), Color(Black), Body-type(SUV)\}$, then $A$ is preferred to $B$ with respect to $\succ$. An NEG constructor can also be similarly expressed.

The notion that two structures are equally preferred is defined in the following.

Definition 18. For two structures $\mathcal{B}, \mathcal{B}' \in M$, if a) neither $\mathcal{B} \succ_{\mathcal{P}} \mathcal{B}'$ nor $\mathcal{B}' \succ_{\mathcal{P}} \mathcal{B}$, b) for all $\mathcal{B}'' \in M$, if $\mathcal{B}'' \succ_{\mathcal{P}} \mathcal{B}$ then $\mathcal{B}'' \succ_{\mathcal{P}} \mathcal{B}'$, and c) if $\mathcal{B} \succ_{\mathcal{P}} \mathcal{B}''$ then $\mathcal{B}' \succ_{\mathcal{P}} \mathcal{B}''$, they are called equally preferred with respect to $\mathcal{P}$ and are represented by $\mathcal{B} \equiv_{\mathcal{P}} \mathcal{B}'$. Also, $\mathcal{B} \succeq_{\mathcal{P}} \mathcal{B}'$ means that $\mathcal{B} \succ_{\mathcal{P}} \mathcal{B}'$ or $\mathcal{B} \equiv_{\mathcal{P}} \mathcal{B}'$.

The following results characterize some properties of binary relations $\succ_{\mathcal{P}}$, $\approx_{\mathcal{P}}$, and $\succeq_{\mathcal{P}}$ based on Definition 16, 17, and 18.

Proposition 4. Given a preference $\mathcal{P} = (\mathcal{O}, \Gamma)$ in a module $M$, $\succ_{\mathcal{P}}$ is a strict partial order, $\approx_{\mathcal{P}}$ is an equivalence relation over structures of $M$, and $\succeq_{\mathcal{P}}$ is a transitive and reflexive binary relation over structures of $M$.

Proof: As specified in Definition 15, a binary relation is a strict partial order if it is anti-reflexive, asymmetrical, and transitive. Also, it can be simply shown that a transitive binary relation is asymmetric if and only if it is irreflexive. For relation $\succ_{\mathcal{P}}$, assume for an arbitrary structure $\mathcal{B}$, the relation $\mathcal{B} \succ_{\mathcal{P}} \mathcal{B}$ holds. Let us assume $\mathcal{A}_1 = \mathcal{A}_2 = \mathcal{B}$ which implies $\mathcal{A}_1 \succ_{\mathcal{P}} \mathcal{A}_2$ and $\mathcal{A}_2 \succ_{\mathcal{P}} \mathcal{A}_1$. Based on Definition 17, there are $\mathcal{B}_i$ and $\mathcal{B}_j$ that can be extended to $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, such that $\mathcal{B}_i \succ \mathcal{B}_j$. However, $\mathcal{A}_1$ also extends $\mathcal{B}_j$ and $\mathcal{A}_2$ extends $\mathcal{B}_i$, which leads to the violation of condition 3 in Definition 17. Hence, $\mathcal{B} \succ_{\mathcal{P}} \mathcal{B}$ does not hold. Regarding the symmetry property, if $\mathcal{A} \succ_{\mathcal{P}} \mathcal{B}$ and $\mathcal{B} \succ_{\mathcal{P}} \mathcal{A}$, there are $\mathcal{B}_i, \mathcal{B}_j$ that can be extended to $\mathcal{A}$ and $\mathcal{B}$, respectively, and $\mathcal{B}_i \succ \mathcal{B}_j$. Also, there are $\mathcal{B}_m$ and $\mathcal{B}_n$ that are extended to $\mathcal{A}$ and $\mathcal{B}$, respectively, and $\mathcal{B}_m \succ \mathcal{B}_n$ that is in contradiction with condition 3 in Definition 17. So, $\mathcal{A} \succ_{\mathcal{P}} \mathcal{B}$ and $\mathcal{B} \succ_{\mathcal{P}} \mathcal{A}$ do not hold at the same time, which proves $\succ_{\mathcal{P}}$ is asymmetric. For the transitivity property, assume $\mathcal{A} \succ_{\mathcal{P}} \mathcal{B}$ and $\mathcal{B} \succ_{\mathcal{P}} \mathcal{C}$.

There are partial structures $\mathcal{A}_i, \mathcal{B}_j, \mathcal{B}_k$, and $\mathcal{C}_m$ such that $\mathcal{A}$ extends $\mathcal{A}_i$, $\mathcal{B}_j$ and $\mathcal{B}_k$ can be extended to $\mathcal{B}$, and $\mathcal{C}$ is an extension of $\mathcal{C}_m$. Also, $\mathcal{A}_i \succ \mathcal{B}_j$ and $\mathcal{B}_k \succ \mathcal{C}_m$. Since $\succ$ is a total order, $\mathcal{A}_i \succ \mathcal{B}_k$ otherwise
where we can conclude that $A_i > C_m$ and, therefore, $A > C$.

With regard to relation $\approx_{\rho}$, the reflexivity is trivial because $\succ_{\rho}$ is a strict partial order and in Definition 18, by replacing an arbitrary structure $A$ with $B$ and $B'$, all three conditions (a), (b), and (c) for *equally preferred* hold. So, $A$ is *equally preferred* to $A$ and therefore $\approx_{\rho}$ is a reflexive relation. Similarly, according to Definition 18, $\approx_{\rho}$ is trivially symmetric. Also, conditions (b) and (c) guarantee that $\approx_{\rho}$ is transitive. Therefore, $\approx_{\rho}$ is an equivalence relation. Since $\succeq_{\rho}$ means that $\succ_{\rho}$ or $\approx_{\rho}$, it is clear that $\succeq_{\rho}$ is transitive because both $\approx_{\rho}$ and $\succ_{\rho}$ are transitive. Also, $\succeq_{\rho}$ is reflexive since $\approx_{\rho}$ is reflexive. However, $\succeq_{\rho}$ is not guaranteed to be symmetric because $\succ_{\rho}$ is not symmetric. So, $\succeq_{\rho}$ is a reflexive and transitive (pre-order) relation.

For a preference $P = (O, \Gamma)$ where $O$ is a strict partial order, the above results do not necessarily hold. For example, $\succ_{\rho}$ is not always transitive. Consider $A \succ_{\rho} B$ and $B \succ_{\rho} C$. There are partial structures $A_i, B_j, B_k,$ and $C_m$ that can be extended to $A, B, B,$ and $C,$ respectively, such that $A_i > B_j$ and $B_k > C_m.$ There could be no relation between $A_i$ and $B_k$ without violating condition 3 in Definition 17. Similarly, for the symmetry property, there may exist partial structures $A_i$ and $A_j$ that can be extended to $A$ and partial structures $B_i, B_j$ extendible to $B$ where $A_i > B_i$ and $A_j > B_j$ and there is no relation between $A_i$ and $A_j$ or between $B_i$ and $B_j.$ In this case, $A \succ_{\rho} B$ and $B \succ_{\rho} A$ would not hold.

For a preference $P = (O, \Gamma)$ where $O$ is a strict partial order, the above results do not necessarily hold. For example, $\succ_{\rho}$ is not always transitive. Consider $A \succ_{\rho} B$ and $B \succ_{\rho} C$. There are partial structures $A_i, B_j, B_k,$ and $C_m$ that can be extended to $A, B, B,$ and $C,$ respectively, such that $A_i > B_j$ and $B_k > C_m.$ There could be no relation between $A_i$ and $B_k$ without violating condition 3 in Definition 17. Similarly, for the symmetry property, there may exist partial structures $A_i$ and $A_j$ that can be extended to $A$ and partial structures $B_i, B_j$ extendible to $B$ where $A_i > B_i$ and $A_j > B_j$ and there is no relation between $A_i$ and $A_j$ or between $B_i$ and $B_j.$ In this case, $A \succ_{\rho} B$ and $B \succ_{\rho} A$ would not hold.

We call a $\delta$-partial $\tau$-structure $A$ an *absolute* partial structure if for all $R \in \delta, R^+_{\tau_{\rho}}$ and $R^-_{\tau_{\rho}}$ are empty and the interpretation of each $S \in \tau \setminus \delta$ is total. Let $B$ be a structure in $M$ and $P = (O, \Gamma)$ be a $\tau_o$-preference where $\tau_o \subseteq vocab(M)$, $O$ is a strict partial order over *absolute* $\delta$-partial structures where $\delta \subseteq \tau_o$, and $\succ_{\rho}$ is a strict partial order. Assume that $B \succ_{\rho} B$. According to Definition 17, there are partial structures $B_i$ and $B_j$ over $vocab(M)$ that can be extended to $B$ such that $B_i > B_j.$ Since both $B_i$ and $B_j$ are $\tau_o$-partial structures and they can be extended to the same total structure $B$, we can conclude that $B_i \equiv B_j.$ An immediate result is $B_i > B_i$. This contradicts the assumption that $O$ is a strict partial order. Therefore, the binary relation $\succ_{\rho}$ is anti-reflexive. To prove the asymmetry property, assume two structures $B, B' \in M$ and $B \succ_{\rho} B'$. From Definition 17, there exist partial structures $B_i$ and $B_j,$ that can be extended to $B$ and $B',$ respectively such that $B_i > B_j.$ If $\succ_{\rho}$ is symmetrical, then we have $B' \succ_{\rho} B$, which implies there exist partial structures $B_m,$ extendible to $B$, and $B_n$, extendible to $B'$, over $vocab(M)$ such that $B_n > B_m.$ According to Definition 17, $B_i, B_j, B_m,$ and $B_n$ are $\tau_o$-partial structures, where $\tau_o \subseteq vocab(M).$ Since absolute partial structures $B_i$ and $B_n$ are extendible to the same total structure $B$, an immediate result is $B_i = B_n.$ Having the same argument, we conclude $B_j = B_m.$ Consequently, we have $B_i \succ B_j$ and at the same time $B_j \succ B_i.$ This is a contradiction, so binary relation $\prec_{\rho}$ is not symmetrical. It is straightforward to prove that the relation $\prec_{\rho}$ is transitive. Let $B, B', \text{ and } B''$ be total structures in $M.$ Assume that $B \succ_{\rho} B'$ and $B' \succ_{\rho} B''$. Definition 17 states that there exist partial structures $B_i, B_j, B_m, B_n$ with $vocab(M)$ that $B$ is an extension of $B_i, B_j$ and $B_m$ are extendible to $B'$, and $B_n$ can be extended to
Also, we have $B \succ B'$ and $B' \succ B''$. Considering the fact that partial structures $B_j$ and $B_m$ can be extended to the same total structure, it can be concluded that they are the same. Consequently, we have $B_i \succ B_j$ and $B_j \succ B_n$. Since $O$ is a strict partial order, we have $B_i \succ B_n$, which results in $B \succ_P B''$. Consequently, $\succ_P$ is a strict partial order.

**Meta-Preferences**

Modules may have more than one preference. Some of the preferences may have higher priority than others. For example, consider the following scenario: A transportation planner decides on one of three methods of transportation: walking, taxi, or bus. Let $P_t$ be a preference regarding the time of travel and $P_c$ be a preference regarding the cost of travel. Assume that a decision maker considers time to be a more important factor than cost. We model the priority of $P_t$ with the notion of meta-preferences. We say $P_t$ is preferred to $P_c$ with notation $P_t > P_c$.

**Definition 19.** Consider a module $M$ and a set of preferences $\Pi = \{P_1, P_2, ..., P_n\}$ that are defined over the same domain. Assume $O_{MP} = (\Pi, \succ)$ is a strict total order over elements of $\Pi$. Meta-preference $M\Pi$ is characterized as $M\Pi := O_{MP}$. We say structure $B$ is preferred to $B'$ with respect to binary relation $\succ_{M\Pi} \subseteq M \times M$ (notation $B \succ_{M\Pi} B'$) if

- there is a preference $P_i \in \Pi$ such that $B \succ_{P_i} B'$, and
- for all $P_j \in \Pi$ where $P_j > P_i$, $B' \approx_{P_j} B$.

If $O_{MP}$ specifies an empty binary relation (there is no priority over elements of $\Pi$) denoted by $O_{MP} = \emptyset$, then $A \succ_{M\Pi} B$ whenever there is a preference $P_i \in \Pi$ such that $A \succ_{P_i} B$ and for all $P_j \in \Pi$, where $i \neq j$, $A \succeq_{P_j} B$.

This definition states that structure $B$ is preferred to $B'$ with respect to $M\Pi$ if we can find a preference $P_i$ such that $B \succ_{P_i} B'$ and for all $P_j$ in $\Pi$ such that $P_j > P_i$, $B$ and $B'$ are equally preferred. If there is a preference $P_j$ that is more important than $P_i$ and $B' \succ_{P_j} B$, then $B$ is not preferred to $B'$ with respect to the meta-preference. The notion of a meta-preference is related to prioritized accumulation operator $&$ in [84]. The prioritized preference $P_1 & P_2$ means that $P_2$ is applied when $P_1$ is not applicable.

**Example 10.** In Example 7, assume that the company has more than one preference. If an expensive item is selected for delivery, it is not secure to have another precious item in the pack that is specified by $P_{secur}$. Assume we have a meta-preference $M\Pi$ such that $\Pi_K = \{P_{safe}, P_{secur}\}$ and $M\Pi = \{P_{safe} > P_{secur}\}$. To have a preferred packing in accordance with the Knapsack module, when there is an item in the package that is both heavy and expensive, it is preferable to not include another heavy item, but it is fine to have another expensive item in the pack.
Preference-based Modular Systems

Up to now, we defined primitive preference $\mathcal{P}$ in atomic modules. In what follows, we examine how a compound preference in a modular system is constructed when preferences of its components are given. First, we define the serial composition of preferences.

**Definition 20.** Let $M = M_1 \triangleright M_2$ be a modular system where $M_1 \in MS(\sigma_1, \varepsilon_1)$ and $M_2 \in MS(\sigma_2, \varepsilon_2)$. Given a $\tau_{o_1}$-preference $\mathcal{P}_1 = (O_1, \Gamma_1)$ in $M_1$ where $\tau_{o_1} \subseteq \varepsilon_1$ and a $\tau_{o_2}$-preference $\mathcal{P}_2 = (O_2, \Gamma_2)$ in $M_2$ where $\tau_{o_2} \subseteq \varepsilon_2$, for all $B, B' \in M$, $B$ is preferred to $B'$ with respect to $\mathcal{P}_1$ and $\mathcal{P}_2$, with notation $B \succ_{\mathcal{P}_1 \triangleright \mathcal{P}_2} B'$ when $B|_{\varepsilon_1} \succ_{\mathcal{P}_1} B'|_{\varepsilon_1}$ and $B|_{\varepsilon_2} \succ_{\mathcal{P}_2} B'|_{\varepsilon_2}$, where $B|_{\tau}$ is the restriction of $B$ by vocabulary $\tau$.

Informally speaking, $B$ is preferred to $B'$ with respect to $\mathcal{P} = \mathcal{P}_1 \triangleright \mathcal{P}_2$ if $B$ is preferred to $B'$ with respect to $\mathcal{P}_1$ when $B$ and $B'$ are restricted by the output vocabulary of $M_1$ and with respect to $\mathcal{P}_2$ when they are restricted by the output vocabulary of $M_2$. $\mathcal{P}_1$ specifies a preference relation over some portion of the output vocabulary of $M_1$, say $\varepsilon_1^*$, that can be linked to the input vocabulary of $M_2$. Since the input of $M_2$ is (partially) determined by the output of $M_1$, the preferred output of $M_2$ is dependent on $\mathcal{P}_1$. Also, when $M_1$ and $M_2$ are independent, that is, when $(\varepsilon_1, \varepsilon_2)$, $(\sigma_1, \varepsilon_2)$, and $(\sigma_2, \varepsilon_1)$ are disjoint pairs of vocabularies of symbols, $M_1 \triangleright M_2$ and $M_2 \triangleright M_1$ are equivalent [80]. Given independent modular systems $M_1$ and $M_2$, and for all $\tau$-structures $A$ and $B$ where $\tau = \sigma_1 \cup \varepsilon_1 \cup \sigma_2 \cup \varepsilon_2$, $A \succ_{\mathcal{P}_1 \triangleright \mathcal{P}_2} B$ if and only if $A \succ_{\mathcal{P}_2 \triangleright \mathcal{P}_1} B$.

**Example 11.** In Example 7, for module $M_{tsp}$, suppose that if cities $c_1, c_2, c_3, c_4, c_5$ are in the set of destinations, there is a path from $c_1$ to $c_4$ through $c_2$ that is preferred to the path from $c_1$ to $c_3$ through $c_2$. This can be formalized by preference $\mathcal{P}_{tsp} = (O_{tsp}, \Gamma_{tsp})$ where $O_{tsp} = (S_{tsp}, <)$ is an order over $S_{tsp}$ that is a set of possible routes. For a positive integer $k$ and truck $t$,

\[
\{Route(k, c_1, t), Route(k + 1, c_2, t), Route(k + 2, c_4, t) \succ \\
Route(k, c_1, t), Route(k + 1, c_3, t), Route(k + 2, c_4, t)\}
\] and
\[
\Gamma_{tsp} = \{\{Dest(c_1), Dest(c_2), Dest(c_3), Dest(c_4), Dest(c_5)\}\}.
\]

The preferred plan for packing and delivery with respect to $\mathcal{P}_{safe} \triangleright \mathcal{P}_{tsp}$ is the one where heavy and light items are not in the same pack and if the truck is supposed to visit cities $c_1, c_2, c_3, c_4$, then taking road $(c_1, c_2)$ is preferred to $(c_1, c_3)$.

In cases where there is a set of preferences with a meta-preference in each module, preference relations with respect to the meta-preferences can be serially composed similarly to the composition of singleton preferences. Given $\mathcal{M}\mathcal{P}_1$ over $\Pi_1 = \{\mathcal{P}_{11}, ..., \mathcal{P}_{1r}\}$ in $M_1$ and $\mathcal{M}\mathcal{P}_2$ over $\Pi_2 = \{\mathcal{P}_{21}, ..., \mathcal{P}_{2s}\}$ in $M_2$, we introduce binary relation $\succ_{\mathcal{M}\mathcal{P}}$ as $\{\{A, B\}|A, B \in M_1 \triangleright M_2, A|_{\varepsilon_1} \succ_{\mathcal{M}\mathcal{P}_1} B|_{\varepsilon_1}$, and $A|_{\varepsilon_2} \succ_{\mathcal{M}\mathcal{P}_2} B|_{\varepsilon_2}\}$.

The union operator introduces choice between the outputs of modules (similar to parallel circuits). Preferred outputs are also selected non-deterministically using the union operator.
Definition 21. Let $M = M_1 \cup M_2$ be a modular system where $M_1 \in MS(\sigma_1, \varepsilon_1)$ and $M_2 \in MS(\sigma_2, \varepsilon_2)$. Assume $\mathcal{P}_1$ and $\mathcal{P}_2$ are preferences in $M_1$ and $M_2$ respectively. For $B, B' \in M$, if $B|_{\varepsilon_1} \succ_{\mathcal{P}_1} B'|_{\varepsilon_1}$ and $B|_{\varepsilon_2} \succ_{\mathcal{P}_2} B'|_{\varepsilon_2}$ or $B|_{\varepsilon_2} \succ_{\mathcal{P}_2} B'|_{\varepsilon_2}$ and $B|_{\varepsilon_1} \succ_{\mathcal{P}_1} B'|_{\varepsilon_1}$, then $B$ is preferred to $B'$ with respect to $\mathcal{P}_1 \cup \mathcal{P}_2$ and is denoted by $B \succ_{\mathcal{P}_1 \cup \mathcal{P}_2} B'$.

For meta-preferences, similar to the above definition, given $\mathcal{MP} = \mathcal{MP}_1 \cup \mathcal{MP}_2$, binary relation $\succ_{\mathcal{MP}}$ is defined as $\{(A, B) \mid A, B \in M_1 \cup M_2$ and $A|_{\varepsilon_1} \succ_{\mathcal{MP}_1} B|_{\varepsilon_1}$ and $A|_{\varepsilon_2} \succ_{\mathcal{MP}_2} B|_{\varepsilon_2}$ or $A|_{\varepsilon_2} \succ_{\mathcal{MP}_2} B|_{\varepsilon_2}$ and $A|_{\varepsilon_1} \succ_{\mathcal{MP}_1} B|_{\varepsilon_1}\}$.

Let us comment briefly on the feedback operator. Let $M_f = M[R = S]$ where $M \in MS(\sigma, \varepsilon)$, $R \in \sigma$, and $S \in \varepsilon$. The feedback operator selects structures $B$ in module $M$ such that $R^B = S^B$.

With regard to preferences, when $B \succ_{\mathcal{P}} B'$ holds in $M$, if $B$ and $B'$ are also structures of $M_f$, then $B$ is preferred to $B'$ in $M_f$.

Definition 22. Let $M_f = M[R = S]$ and $\mathcal{P} = (O, \Gamma)$ be a preference in $M$. $\succ_{\mathcal{P}[R=S]}$ is a binary relation over models (structures) of $M_f$. For all structures $B, B' \in M$, whenever $R^B = S^B$ and $R^B' = S^B'$, if $B \succ_{\mathcal{P}} B'$, then $B \succ_{\mathcal{P}[R=S]} B'$.

This definition says that if two structures $B$ and $B'$ are in $M$, and $B$ is preferred to $B'$ with respect to $\mathcal{P}$, then $B$ remains preferable to $B'$ in module $M_f$, which is module $M$ with feedback. In other words, $\succ_{\mathcal{P}[R=S]}$ is a subset of relation $\succ_{\mathcal{P}}$. For module $M$ with meta-preference $\mathcal{MP}$ over preferences $\Pi$, binary relation $\succ_{\mathcal{MP}[R=S]}$ is defined as $\succ_{\mathcal{MP}[R=S]} = \{(A, B) \mid A, B \in M, M_f$ and $A \succ_{\mathcal{MP}} B\}$.

For the projection operator, assume $M' = \pi_{\delta}(M)$ where $M \in MS(\sigma, \varepsilon)$. The projection operator hides some vocabulary symbols. We define the construction of relation $\succ_{\pi_{\delta}\mathcal{P}}$ in $M'$ (which is called the projection of preference relation $\succ_{\mathcal{P}}$ in $M$) using the intuition behind Definition 17. Let us consider a $\delta$-preference $\mathcal{P} = (O, \Gamma)$ where $\Gamma$ is empty and $O = (S, \succ)$ where $S$ is the set of structures $A|_{\delta}$ where $A \in M$. For all $A', B' \in M'$, $A' \succ_{\mathcal{P}} B'$ if there are $A_i$ and $A_j$ in $S$ such that $A_i \succ_{\mathcal{P}} A_j$ and $A'|_{\delta} = A_i$ and $B'|_{\delta} = A_j$. Also, for all $A$ and $B$ in $M$ such that $A|_{\delta} = A_i$ and $B|_{\delta} = A_j$, we have $A \succ_{\mathcal{P}} B$. To ensure that no relevant information about preferences is lost by the projection, we consider $A_i \succ_{\mathcal{P}} A_j$ if each structure $A$ where $A|_{\delta} = A_i$ is preferred with respect to $\mathcal{P}$ to every structure $B$ where $B|_{\delta} = A_j$. This shows how $\succ_{\mathcal{P}}$ and $\succ_{\pi_{\delta}\mathcal{P}}$ are related. Therefore, given two structures $A'$ and $B'$ in $M' = \pi_{\delta}(M)$, if all structures in $M$ which their projection to $\delta$ are equal to $A'|_{\delta}$ are preferred to all structures in $M$ which their projection to $\delta$ are equal to $B'|_{\delta}$, we say $A'$ is preferred to $B'$. The definition is made precise in the following:

Definition 23. Assume $M' = \pi_{\delta}(M)$ where $\text{vocab}(M) = \tau$ and $\text{vocab}(M') = \delta$. Let $\mathcal{P}$ be a preference in $M$. We define $\mathcal{P}' = \pi_{\delta}(\mathcal{P})$ and binary relation $\succ_{\mathcal{P}'} = \{(A', B') \mid A', B' \in M' \land \forall A \in M(A|_{\delta} = A'|_{\delta} \Rightarrow (\forall B \in M(B|_{\delta} = B'|_{\delta} \Rightarrow A \succ_{\mathcal{P}} B)))\}$. 

45
Analogously, we introduce the concept of projected meta-preference \( MP' \) in \( M' \). Binary relation \( \triangleright_{\pi_\delta(MP)} \) is defined as \( \{(A', B'), A', B' \in M' \land \forall A \in M (A|_\delta = A'|_\delta \Rightarrow (\forall B \in M (B|_\delta = B'|_\delta \Rightarrow A \triangleright_{MP} B)))\}. \)

It is worth mentioning that for a \( \tau_o\)-preference \( P \), if \( \tau_o \subseteq \delta \), then, for all structures \( A \in M \), no information related to \( P \) in \( A \) is lost by projection \( \pi_\delta \). Projection in this case is called preference preserving.

A modular system \( M \) with a strict partial order \( \triangleright \) over models of \( M \) is called a preference-based modular system with notation \( (M, \triangleright) \). The following result indicates that our framework is closed in the sense that the operations for combining modules preserve the notion of preference-based modular systems. This property is proven by structural induction.

Let \( (M_1, \triangleright_{MP_1}), \ldots, (M_n, \triangleright_{MP_n}) \) (for a constant \( n \)) be preference-based modular systems where \( M_i \) is a modular system and \( MP_i \) is a meta-preference for module \( M_i \). Assume \( M \) is an algebraic expression in the algebra of modular systems (a compound module) of the form \( M = (\bigoplus M_1) \otimes (\bigoplus M_2) \otimes \ldots \otimes (\bigoplus M_n) \) and \( \triangleright_{MP} = \triangleright_{(\bigoplus MP_1)} \otimes (\bigoplus MP_2) \otimes \ldots \otimes (\bigoplus MP_n) \) where \( \otimes \) is sequential composition or union and \( \bigoplus \) is feedback, projection, or does nothing.

**Theorem 6.** The pair \( (M, \triangleright_{MP}) \) is a preference-based modular system.

**Proof:** It follows from Definitions 15, 17, 20, 21, 22, 23 and Proposition 4. \]

For a module \( M \) and a set of preferences defined in \( M \) with a meta-preference \( MP \) over \( \Pi \), the pair \( (M, \triangleright_{MP}) \) is a preference-based modular system. Let us define operator \( F \) as follows: \( \Omega = F(M, \triangleright_{MP}) \) where \( \Omega \subseteq M \) and \( A \in \Omega \) if and only if there is no \( B \in M \) such that \( B \triangleright_{MP} A \). \( \Omega \) that represents a set of structures called a prioritized modular system. Each structure \( A \in \Omega \) is called an optimal structure.

Assume a modular system \( M \) is constructed as \( M_1 \triangleright_{MP} M_2 \), where \( M_i, 1 \leq i \leq 2 \) are modular systems. Also, let \( \triangleright_{MP} \) be a preference relation over models of \( M_i \) generated from a set of preferences \( \Pi_i \) with a meta-preference \( MP_i \) over \( \Pi_i \). Let \( \Omega_i = F(M_i, \triangleright_{MP_i}) \) be a prioritized modular system for \( 1 \leq i \leq 2 \). It can be shown that \( \Omega = \Omega_1 \triangleright_{MP} \Omega_2 \) is a prioritized modular system such that \( \Omega = F(M, \triangleright_{MP}) \) where \( \triangleright_{MP} = \triangleright_{MP_1 \triangleright_{MP_2}} \). Let \( B \) be a model of \( \Omega \). Assume there is a structure \( C \) in \( M \) such that \( C \triangleright_{MP} B \). So, \( C|_{\varepsilon_1} \triangleright_{MP_1} B|_{\varepsilon_1} \) and \( C|_{\varepsilon_2} \triangleright_{MP_2} B|_{\varepsilon_2} \) where \( \varepsilon_1 \) is the output vocabulary of \( M_1 \) and \( \varepsilon_2 \) is the output vocabulary of \( M_2 \). Thus, neither \( \Omega_1 \) nor \( \Omega_2 \) is a prioritized modular system that contradicts our assumption. Also, if \( A|_{\varepsilon_1} \) and \( A|_{\varepsilon_2} \) are optimal structures with respect to \( \triangleright_{MP_1} \) and \( \triangleright_{MP_2} \), respectively, \( A \) is an optimal expansion with respect to \( \triangleright_{MP_1 \triangleright_{MP_2}} \). Therefore, \( \Omega = F(M, \triangleright_{MP}) \) is a prioritized modular system.

For the union operation, \( \Omega_1 \cup \Omega_2 \) is not equal to \( F(M_1 \cup M_2, \triangleright_{MP_1 \cup_{MP_2}}) \). For a structure \( A \in \Omega_1 \cup \Omega_2 \), \( A \) is not necessarily an optimal structure because, for example, \( A|_{\varepsilon_2} \) could be an arbitrary structure. By applying operator \( F \) on \( \Omega_1 \cup \Omega_2 \), we can filter out all non-optimal structures. In other words, \( F(\Omega_1 \cup \Omega_2, \triangleright_{MP_1 \cup_{MP_2}}) = F(M_1 \cup M_2, \triangleright_{MP_1 \cup_{MP_2}}) \). For the projection operator, since some information about preferences might be hidden by the projection, we cannot generally
establish a relation between \( \pi_\delta(F(M, \succ_M^p)) \) and \( F(\pi_\delta M, \succ_p) \) similar to those for the composition and union. Likewise, for the feedback, \( F(M[R = S], \succ_M^p[R = S]) \) is not equal to \( F(M, \succ_M^p) \) with a feedback.

### 4.4 Relationship of \( P-\mathcal{MS} \) to other Preference Formalisms

We shall now study the relationship between CP-nets [20], Answer Set Optimization [36], and Preference-based Planning [120] and our proposal of the \( P-\mathcal{MS} \) framework.

#### 4.4.1 CP-Nets

A CP-net [20] is a graphical representation of conditional ceteris paribus (CP) preference statements. Based on CP assumptions that are broadly used in economics, when a variable is in a cause and effect relationship with another variable, all variables that can affect the other variable are assumed to be constant. A CP-net \( \mathcal{N} \) over variables \( V = \{v_1, ..., v_n\} \) is a directed graph such that each node \( v_i \) in \( \mathcal{N} \) represents a variable \( v_i \). An outcome \( o \) of \( \mathcal{N} \) is an assignment to the members of \( V \). Each variable represents an attribute of an outcome. We may henceforth use the terms "variable" and "attribute" interchangeably. Preference over values of \( v_i \) is dependent on a set of variables that are called parents of \( v_i \) (notation \( Pr(v_i) \)). A subset \( Y \) of \( V \) is called preferentially independent of \( V \setminus Y \) when the preferences over values of \( Y \) do not depend on the values of variables in \( V \setminus Y \). In CP-net \( \mathcal{N} \), variable \( v_i \) is preferentially independent of \( V - (Pr(v_i)) \). There is a directed edge between each member of \( Pr(v_i) \) to \( v_i \). Each node \( v_i \) is associated with a conditional preference table (CPT) with notation \( CPT_i \) or \( CPT(v_i) \) that specifies preferences over \( dom(v_i) \) (\( dom(v_i) \) denotes the domain of variable \( v_i) \) based on value assignments to \( Pr(v_i) \). The induced graph derived from CP-net \( \mathcal{N} \) shows an ordering relationship among outcomes of \( \mathcal{N} \). Each node in the induced graph represents an outcome and each directed edge exhibits an ordering relationship between nodes. An outcome \( o_1 \) is preferred to \( o_2 \) if in the induced graph, there is a path from \( o_1 \) to \( o_2 \). The induced graph contains all information about preferences over outcomes that can be derived from a CP-net. The complexity of determining whether an outcome is preferred to another outcome varies based on the structure of CP-nets from polynomial to undecidable with regard to the number of variables [20]. In this thesis, we only consider acyclic CP-nets where there is no cyclic conditional dependency among variables.

From the syntactic point of view, each variable in a CP-net can be viewed as a unary predicate symbol in the context of the \( P-\mathcal{MS} \). For a CP-net \( \mathcal{N} \), let module \( M \) specifies all possible outcomes of \( \mathcal{N} \). For each variable \( v \in V \), consider a unary predicate \( A_v \) and a formula of the form \( \phi = \forall x (A_v(x) \supset x \in dom(v)) \land \exists y (y \in dom(v) \land A_v(y)) \) in the axiomatization of \( M \) (\( \phi \) can be written in any logic with a model-theoretic semantics) that indicates each interpretation of \( A_v \) is a singleton containing only one member of the domain of \( v \). As a result, for vocabulary \( \tau = \{A_v_1, ..., A_v_n\} \) and domain \( Dom = \{dom(v_1) \cup ... \cup dom(v_n)\} \), an outcome \( o \) in CP-net \( \mathcal{N} \) can be represented by a \( \tau \)-structure with domain \( Dom \). \( M \) is defined as the class of \( \tau \)-structures with domain \( Dom \) satisfying \( \phi = \phi_1 \land ... \land \phi_n \). It is clear that \( M \) represents the set of all possible outcomes of \( \mathcal{N} \).
Each CPT_i table in \( N \) can be encoded as a preference \( \mathcal{P} = (\mathcal{O}, \Gamma) \) in the \( \mathcal{P}MS \) framework as follows: Consider that, in a sense, partial structures in the \( \mathcal{P}MS \) are collections of some interpreted vocabulary symbols. Thus, a partial structure can stipulate a value assigned to an attribute. In fact, each ordering over a set of partial structures in the \( \mathcal{P}MS \) is an ordering over an attribute’s values in a CP-net. To be more precise, the preference relation specified by CPT_i can be considered as a \( \tau_o \)-preference \( \mathcal{O} = (\mathcal{S}, \succ) \) where \( \tau_o = A_{v_i} \). Transforming the conditional part of a preference statement in a CP-net into a conditional preference in the \( \mathcal{P}MS \) framework is straightforward. \( \Gamma \) can be a singleton containing a \( \tau_p \)-partial \( \tau \)-structure such that \( \tau_p = \{ A_{v_k}, \ldots, A_{v_m} \} \) where \( \mathcal{P}(v_i) = \{ v_k, \ldots, v_m \} \) is the set of variables that are parents of \( v_i \) in CP-net \( N \). Therefore, it can be observed that the parents of each CP-net variable can be represented by \( \Gamma \).

In order to establish the correspondence between the semantics of CP-nets and preference-based modular systems in the \( \mathcal{P}MS \) framework, we first explain the concept of \( \text{flip} \) in CP-nets. In the induced graph derived from a CP-net \( N \), each outcome node has one attribute value preferred to its child’s while other attribute values of the parent and child are equal. Therefore, by moving from a node to its children, one attribute value is changed and that change is called a \( \text{flip} \). A path in the induced graph is a chain of \( \text{flips} \) between two outcomes. Hence, an outcome is preferred to another when a single or multiple \( \text{flip(s)} \) exist between them. For outcomes \( o \) and \( o' \), \( N \models o \succ o' \) denotes that there is a path (worsening \( \text{flip(s)} \)) from \( o \) to \( o' \) in the induced graph of \( N \). Now we show how a \( \text{flip} \) can be expressed in the \( \mathcal{P}MS \) framework. Consider two structures \( B \) and \( B' \) that represent outcomes \( o \) and \( o' \), respectively. Assume \( B \succ \mathcal{M} \mathcal{P} \ B' \) where \( \mathcal{M} \mathcal{P} \) is a meta-preference over a set of preferences \( \Pi = \{ \mathcal{P}_1, \ldots, \mathcal{P}_n \} \) and each CPT_i in \( N \) is expressed by preference \( \mathcal{P}_i \) in \( \mathcal{P}MS \). Based on this assumption, there is a \( \tau_{o_j} \)-preference \( \mathcal{P}_j \in \Pi (\tau_{o_j} = \{ A_{v_j} \}) \) such that \( B \succ \mathcal{P}_j \ B' \). So, \( B \) is preferred to \( B' \) based on the interpretation of at least one vocabulary symbol (i.e., \( \tau_{o_j} \)). Relation \( \succ \mathcal{M} \mathcal{P} \) guarantees that there is also no other \( \tau_{o_k} \)-preference \( \mathcal{P}_k \in \Pi \) such that the interpretation of \( \tau_{o_k} \) by \( B' \) is preferred to \( B' \)'s. In fact, the concept of a single \( \text{flip} \) can be characterized by \( \succ \mathcal{M} \mathcal{P} \) when \( \mathcal{O}_{\mathcal{M} \mathcal{P}} = \emptyset \) (there is no meta-preference in CP-nets). Relation \( \succ \mathcal{M} \mathcal{P} \) has the transitivity property and if there is a sequence of \( \text{flips} \) between \( o \) and \( o' \), then \( B \succ \mathcal{M} \mathcal{P} \ B' \). If \( \mathcal{O}_{\mathcal{M} \mathcal{P}} \) is not empty, then \( \mathcal{M} \mathcal{P} \) represents the notion of relative importance (meta-preference) in a TCP-net [28] that is an extension of CP-nets to model meta-preferences. This leads to the following theorem, relating CP-nets and the \( \mathcal{P}MS \) framework.

**Theorem 7.** Let \( N \) be a CP-net and \( M \) be a modular system in the \( \mathcal{P}MS \) framework generating outcomes of \( N \). Let \( \Pi = \{ \mathcal{P}_1, \ldots, \mathcal{P}_n \} \) be a set of preferences in \( M \) such that \( \mathcal{P}_i \) is the representation of the CPT_i table in \( N \). Assume \( \mathcal{M} \mathcal{P} \) is a meta-preference over \( \Pi \) such that \( \mathcal{O}_{\mathcal{M} \mathcal{P}} = \emptyset \). Let \( o \) and \( o' \) be outcomes of \( N \) and \( A \) and \( A' \) be structures in \( M \) that represent \( o \) and \( o' \), respectively. If \( N \models o \succ o' \), then \( A \succ \mathcal{M} \mathcal{P} \ A' \).

**Proof:** Recall that a structure is a domain together with an interpretation of a set of non-logical symbols. Assume \( N \) is defined over a set of variables \( V = \{ v_1, v_2, \ldots, v_n \} \). Let \( \tau = \{ A_{v_1}, \ldots, A_{v_n} \} \) be a vocabulary of symbols where \( A_{v_i} \) is a unary predicate representing attribute \( v_i \). \( A \) and \( A' \) are
<table>
<thead>
<tr>
<th>Outcome</th>
<th>Model</th>
<th>Color</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>o₁</td>
<td>Benz</td>
<td>Black</td>
<td>SUV</td>
</tr>
<tr>
<td>o₂</td>
<td>BMW</td>
<td>Black</td>
<td>SUV</td>
</tr>
<tr>
<td>o₃</td>
<td>Benz</td>
<td>White</td>
<td>SUV</td>
</tr>
<tr>
<td>o₄</td>
<td>Benz</td>
<td>Black</td>
<td>Sedan</td>
</tr>
<tr>
<td>o₅</td>
<td>Benz</td>
<td>White</td>
<td>Sedan</td>
</tr>
<tr>
<td>o₆</td>
<td>BMW</td>
<td>White</td>
<td>SUV</td>
</tr>
<tr>
<td>o₇</td>
<td>BMW</td>
<td>Black</td>
<td>Sedan</td>
</tr>
<tr>
<td>o₈</td>
<td>BMW</td>
<td>White</td>
<td>Sedan</td>
</tr>
</tbody>
</table>

Table 4.1: Outcomes in Car Dealership Example

$\tau$-structures with domain $\text{dom}(v_1) \cup \ldots \cup \text{dom}(v_n)$. In the induced graph $G$ that is derived from $N$, assume that outcome node $o$ is preferred to its child $o'$ and they are all similarly preferred except in one attribute, say $v_d$. Assume in $o$, the value of $v_d$ is $d$ and in $o'$ it is $d'$. Since $o$ is preferred to $o'$ based on the assumption, the following holds: $d \succ d'$ given an assignment $t$ to $Pr(v_d)$. Consider two $\{A_{vd}\}$-partial $\tau$-structures $A_i$ and $A_j$ such that $A_{vd}^A_i = \{d\}$ and $A_{vd}^A_j = \{d'\}$. Assume $P = (O, \Gamma)$ is an $\{A_{vd}\}$-preference where, based on $O$, $A_i \succ A_j$. Also, there is no other attribute in $N$ such that its values are preferentially different in $o$ and $o'$. According to Definition 17, we conclude that $A \succ P B$. If $o_1$ and $o_2$ are not adjacent nodes in $G$ and there is a path from $o_1$ to $o_2$, by induction we derive that $A \succ_{MP} B$ where $O_{MP} = \emptyset$ (there is no meta-preference in CP-nets).

The following example shows how CP-nets are connected to the $\mathcal{P}$-$\mathcal{M}$-$\mathcal{S}$ framework.

**Example 12.** Consider the car dealership scenario in Example 9. Each car has three attributes, including model, type, and color. The model can be Benz or BMW, the colour can be white or black, and the type can be SUV or sedan. The cars available at the dealership are listed in Table 4.1. It can be observed from the CTP table in Figure 4.2a that black cars are always better than white ones regardless of type or model. For a black car, Benz is preferred to BMW, and vice versa for a white car. Similarly, a BMW sedan is preferable to an SUV. Conversely, for a Benz, an SUV is preferable to a sedan. The graph induced from this CP-net is illustrated in Figure 4.2. The induced graph shows whether a certain outcome is preferred to another one. For example, outcome $o_1$, a black SUV Benz, is preferred to $o_6$, a white SUV BMW because there is a path form $o_1$ to $o_6$.

To represent this CP-net in the $\mathcal{P}$-$\mathcal{M}$-$\mathcal{S}$ framework, we consider a primitive module (there is no combination of CP-nets) $M$. Let us assume $M$ is the class of structures representing the outcomes of $N$. Consider a unary relation (predicate) that indicates a car model, i.e., $M(\text{Benz})$ or $M(\text{BMW})$. Similarly, $T$ specifies type, including $T(\text{sedan})$ and $T(\text{SUV})$. For colour, predicate $C$ can take value white or black ($C(\text{white})$ or $C(\text{black})$). Let $\mathcal{P}_1 = (O_1, \Gamma_1)$ be a preference where $O = (S_1, \succ_1)$ is a strict partial order over color attribute such that $\succ_1$ is defined as: $C(\text{black}) \succ_1 C(\text{white})$, and $\Gamma_1 = \emptyset$. In fact, $\mathcal{P}_1$ represents the first row in Figure 2.a. In a similar way, $\mathcal{P}_2 =$
(O₁, Γ₁) is defined as Γ₂ = C(black) and O₂ = (S₂, ≽₂) where S₂ = T(BMW) ≽₂ T(Benz).
We construct P₃, P₄, and P₅ likewise. For a set of preferences Π = {P₁, P₂, P₃, P₄, P₅}, meta-preference O₅(MP) = (Π, >) is defined as >= ∅ because there is no priority over preferences. Assume structures A_i and A_j represent outcome o_i and o_j, respectively. According to Definition 19, if there is a path from o_i to o_j in Figure 4.2b, then A_i is preferred to A_j.

### 4.4.2 Answer Set Optimization

Recall from the previous chapter that Answer Set Optimization (ASO) is a framework for representing preferences in the context of Answer Set Programming [36]. An ASO program P is a pair (P_gen, P_pref) where P_gen is a generating program and P_pref, which is called a preference program, is a set of rules of the form r : C₁ > ... > C_k ← a₁, ..., a_n, not b₁, ..., not b_m. In each rule, C_i is a combination of atoms integrated by using logical connectives and default negation. Moreover, a_i and b_i are literals. Given a set of l rules r₁, ..., r_l, each stable model of P_gen is associated with a vector V(X) = {v₁(X), ..., v_l(X)} where v_i(M) is the rank of model M in r_i, v_i(M) is the minimum j of C_j s in r_i that are satisfied by M if M satisfies body, and 1 in other case. Let X₁ and X₂ be stable models of P_gen. X₁ is preferred to X₂ with respect to P_pref if V(X₁) < V(X₂).

We now show the connection between the ASO and P-MS frameworks. Assume X₁, ..., X_n are stable models of an ASO program P = (P_gen, P_pref). Let M be a modular system that solves P_gen (finds the stable models of P_gen) that is encoded as a Model Expansion problem as discussed in Chapter 3. M is represented as a class of structures such that each structure in M represents a stable model of P_gen.

Consider a preference rule r : C₁ > ... > C_k ← a₁, ..., a_n, not b₁, ..., not b_m. A partial structure B_i (or a set of partial structures in the case of logical combinations of literals) can represent each C_i. The body of the rule is a set of positive and negative literals that can be simply expressed...
by a partial structure as well. In light of this observation, a preference in the form of \( \mathcal{P} = (\Gamma, \mathcal{O}) \) can assert a preference rule in ASO. The body of such a rule is modelled by \( \Gamma \), and \( \mathcal{O} \) represents the head of the rule. So, each rule \( r_i \) is represented by a preference \( \mathcal{P}_i \) and a set of rules \( \mathcal{P}_{\text{pref}} \) can be presented by \( \Pi = \{ \mathcal{P}_1, \ldots, \mathcal{P}_m \} \).

**Theorem 8.** Let \( (\mathcal{P}_{\text{gen}}, \mathcal{P}_{\text{pref}}) \) be an ASO program and \( \Pi \) be the representation of \( \mathcal{P}_{\text{pref}} \) in the \( \mathcal{P} \)-\( \mathcal{M} \)-\( \mathcal{S} \) framework. Given two ASP models \( X_1 \) and \( X_2 \) of \( \mathcal{P}_{\text{gen}} \), assume structures \( \mathcal{A} \) and \( \mathcal{B} \) in the \( \mathcal{P} \)-\( \mathcal{M} \)-\( \mathcal{S} \) represent \( X_1 \) and \( X_2 \), respectively. If \( X_1 \) is preferred to \( X_2 \) with respect to \( \mathcal{P}_{\text{pref}} \), then we have \( \mathcal{A} \succ^{\mathcal{M}\mathcal{P}} \mathcal{B} \).

**Proof:** For ASO program \( (\mathcal{P}_{\text{gen}}, \mathcal{P}_{\text{pref}}) \), assume \( M \) is a modular system that solves \( \mathcal{P}_{\text{gen}} \) and \( \mathcal{A}, \mathcal{B} \in M \). Consider a preference rule \( r : C_1 > \ldots > C_k \leftarrow a_1, \ldots, a_n, \text{not } b_1, \ldots, \text{not } b_n \). A partial structure \( B_i \) (or a set of partial structures in cases of logical combination of literals) can represent each \( C_i \). The body of the rule is a set of positive and negative literals that can be simply expressed by a partial structure as well. A preference \( \mathcal{P} = (\Gamma, \mathcal{O}) \) can represent \( r \). The body of such a rule is modelled by \( \Gamma \), and \( \mathcal{O} \) represents the head of the rule. Let \( \Pi = \{ \mathcal{P}_1, \ldots, \mathcal{P}_m \} \) be a set of preferences such that \( \mathcal{P}_i \) encodes \( r_i \in \mathcal{P}_{\text{pref}} \) for \( i \in [1, m] \). \( X_1 \) is preferred to \( X_2 \) if there is rule \( r_i \in \mathcal{P}_{\text{pref}} \) such that \( v_i(X_1) < v_i(X_2) \) and for all other rules \( r_j \) either \( v_j(X_1) < v_j(X_2) \) or \( v_j(X_1) = v_j(X_2) \). In the \( \mathcal{P} \)-\( \mathcal{M} \)-\( \mathcal{S} \) framework, for a meta-preference \( \mathcal{M}\mathcal{P} = \mathcal{O}_{\mathcal{M}\mathcal{P}} \), if \( \mathcal{O}_{\mathcal{M}\mathcal{P}} \) specifies an empty binary relation, according to Definition 19, \( \mathcal{A} \) is preferred to \( \mathcal{B} \) if there is a preference \( \mathcal{P} \in \Pi \) such that \( \mathcal{A} \succ^\mathcal{P} \mathcal{B} \) and for every \( \mathcal{P}' \in \Pi \), \( \mathcal{A} \succeq^{\mathcal{P}'} \mathcal{B} \). The preference of \( X_1 \) over \( X_2 \) ensures that there is a rule \( r_i \) and, hence, a preference \( \mathcal{P} \) such that \( \mathcal{A} \succ^\mathcal{P} \mathcal{B} \). Also, for each rule \( r_j, v_j(X_1) \geq v_j(X_2) \), which implies that for each preference \( \mathcal{P}' \in \Pi \), \( \mathcal{A} \succeq^{\mathcal{P}'} \mathcal{B} \). Therefore, \( \mathcal{A} \) is preferred to \( \mathcal{B} \) with respect to \( \mathcal{M}\mathcal{P} \).

To clarify the relationship between the ASO and \( \mathcal{P} \)-\( \mathcal{M} \)-\( \mathcal{S} \), let us consider the food planner example that was discussed in [36]. Suppose a food menu consists of four sections: appetizer, main-course, drink, and dessert. The domain of each variable is defined as: \( \text{dom(food)} = \{\text{fish, beef}\} \), \( \text{dom(drink)} = \{\text{red-wine, white-wine, beer}\} \), \( \text{dom(appetizer)} = \{\text{soup, salad}\} \), and \( \text{dom(dessert)} = \{\text{ice-cream, pie}\} \). Suppose the preference rules are as below:

- \( r_{p_1} = \text{white-wine} > \text{red-wine} > \text{beer} \leftarrow \text{fish} \)
- \( r_{p_2} = \text{red-wine} > \text{beer} > \text{white-wine} \leftarrow \text{beef} \)
- \( r_{p_3} = \text{pie} > \text{ice-cream} \leftarrow \text{beef} \)

Also, consider a program \( \mathcal{P}_{\text{gen}} = \{r_1, \ldots, r_7\} \) where

- \( r_1 = 1\{\text{soup, salad}\} \)
- \( r_2 = 1\{\text{beef, fish}\} \)
- \( r_3 = 1\{\text{ice-cream, pie}\} \)
- \( r_4 = 1\{\text{red-wine, white-wine, beer}\} \)
- \( r_5 = \leftarrow \text{fish}, \text{not white-wine} \)

51
\[ r_0 = \leftarrow \text{beef}, \text{not red-wine}, \text{not beer} \]
\[ r_7 = \leftarrow \text{beef}, \text{not pie} \]

\( P_{\text{gen}} \) generates the following models:
\[ X_1 = \{\text{soup, beef, beer, ice-cream} \} \]
\[ X_2 = \{\text{salad, beef, beer, ice-cream} \} \]
\[ X_3 = \{\text{soup, fish, beer, ice-cream} \} \]
\[ X_4 = \{\text{salad, fish, beer, ice-cream} \} \]

As noted above, the representation of \( r_{p_1} \) in \( \mathcal{P} \)-\( M \mathcal{S} \) is: \( \mathcal{P}_1 = (\mathcal{O}_1, \Gamma_1) \) where \( \mathcal{O}_1 = \{\text{drink(white-wine)} \gg \text{drink(red-wine)} \gg \text{drink(beer)}\} \) and \( \Gamma_1 = \{\text{main-food(fish)}\} \). For \( r_{p_2} \), we consider \( \mathcal{P}_2 = (\mathcal{O}_2, \Gamma_2) \), \( \mathcal{O}_2 = \{\text{drink(red-wine)} \gg \text{drink(white-wine)} \gg \text{drink(beer)}\} \), and \( \Gamma_2 = \{\text{main-food(beef)}\} \). Furthermore, \( r_{p_3} \) is specified as \( \mathcal{P}_3 = (\mathcal{O}_3, \Gamma_3) \), \( \mathcal{O}_3 = \{\text{appetizer(pie)} \gg \text{appetizer(ice-cream)}\} \), and \( \Gamma_3 = \{\text{drink(beer)}\} \). Finally, we have \( \Pi = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\} \) and \( \mathcal{O}_{\text{MP}} = \emptyset \). \( X_2 \) is preferred to \( X_3 \). Similarly, in the \( \mathcal{P} \)-\( M \mathcal{S} \) framework, we have \( \mathcal{A}_2 \gg_{\text{MP}} \mathcal{A}_3 \) where \( \mathcal{A}_2 \) and \( \mathcal{A}_3 \) represent \( X_2 \) and \( X_3 \), respectively.

### 4.4.3 Preference-based Planning

In what follows, we show how the \( \mathcal{P} \)-\( M \mathcal{S} \) framework is able to assert preference statements expressed in the \( \mathcal{PP} \) [120], which is a preference language for planning problems. While we do not discuss the full details of the \( \mathcal{PP} \) here, we recall the main definitions found in [120]. Given a set of fluent symbols \( \mathcal{F} \) and a set of actions \( \mathcal{A} \), a state is defined as a subset of \( \mathcal{F} \). A planning problem is a triple \( \langle D, I, G \rangle \) where \( D \) indicates pre-conditions and effects of actions, \( I \) is the initial state, and \( G \) stands for the goal state. A solution to a planning problem, which is called a plan, is a chain of actions and states \( I, a_1, ...a_n, G \) that starts from \( I \) and ends at \( G \). A basic desire \( \phi \) is identified as one of the following: 1) a certain action that occurs in the plan denoted by \( \phi \equiv \text{occ}(a) \), 2) a set of certain fluents that are satisfied and is denoted by \( \phi \equiv (f_i \land ... \land f_{i+n}) \), 3) any combination of basic desires based on classical logic connectives (e.g., \( \land, \lor, \neg \)) or temporal connectives stemming from temporal logic, such as \( \text{Next}(\phi_1), \text{Until}(\phi_1, \phi_2), \text{Always}(\phi), \text{Eventually}(\phi) \).

Planning problem \( \langle D, I, G \rangle \) can be reduced to an Answer Set Programming (ASP) program \( \Pi(D, I, G) \) such that for every feasible plan \( p \) there is an answer set \( X_p \) in program \( \Pi \) [119]. We say \( X_p \) represents \( p \). For two plans \( p_1 \) and \( p_2 \), we say \( p_1 \) is preferred to \( p_2 \) with respect to a basic desire \( \phi \) if \( \phi \) is satisfied in \( p_1 \) but not in \( p_2 \). Likewise, in the context of ASP, if \( X_1 \) and \( X_2 \) are two answer sets representing \( p_1 \) and \( p_2 \), respectively, \( X_1 \) satisfies \( \phi \) but \( X_2 \) does not.

A basic desire can be thought of as a constraint that preferred models of a program \( \Pi \) must satisfy. This approach to model preferences is different than preference frameworks introduced in [20] and [36]. In these frameworks, models are generated (that are outcomes in CP-nets and stable models in ASO) and are then compared with respect to an order over some partial assignments. By contrast, in [120], once the set of all preferred models is generated, we are only able to determine whether a model belongs to the set. Also, there is no preference relation over the models (plans)
that do not satisfy the basic desire formula. This is one of the shortcomings of the $PP$, which gives only a binary choice such that either a model is preferred or not and its relation to other models is unknown. The notion of preferred models can be axiomatized as a set of Answer Set rules as discussed in [120]. Therefore, computing the preferred models does not impact the complexity of finding models of $\Pi$.

To express *basic desires* in the $P-MS$ framework, consider a module $M_1$ that finds stable models of Answer Set Program $\Pi$ by solving the associated Model Expansion problem (see Chapter 3). Also, assume $M_2$ solves $\Pi_{\phi}$, which is the translation of $\phi$ into an ASP program. We say $M_1$ and $M_2$ characterize $\Pi$ and $\Pi_{\phi}$, respectively. It is only necessary to sequentially compose $M_1$ and $M_2$ so that the result is the set of the most preferred structures. Also, the relation $\succ_{MP}$ is empty. The following theorem results from what we discussed.

**Theorem 9.** Let $p$ be a feasible plan for a planning problem $\langle D, I, G \rangle$ that is translated into an ASP program $\Pi(D, I, G)$. Let $X_p$ be an answer set of $\Pi$ that represents $p$. Assume modules $M_1$ and $M_2$ characterize program $\Pi$ and a basic desire formula $\Pi_{\phi}$, respectively. Suppose that $X_p$ is represented by a $\tau$-structure $B$ where $\tau = vocab(M_1 \triangleright M_2)$. If $p$ is a preferred plan, then $B$ is a model of $M_1 \triangleright M_2$.

**Proof:**

As discussed in [120], basic desire $\phi$ can be encoded as an ASP program $\Pi_{\phi}$. Let $M_1$ and $M_2$ be modular systems that solve $\Pi$ and $\Pi_{\phi}$, respectively. Assume $M_1 \in MS(\sigma_1, \varepsilon_1)$ and $M_2 \in MS(\sigma_2, \varepsilon_2)$ such that $\varepsilon_1$ and $\varepsilon_2$, $\sigma_1$ and $\varepsilon_2$, and $\sigma_2$ and $\varepsilon_1$ are disjoint pairs of vocabularies of symbols. Thus, $M = M_1 \triangleright M_2$ is equivalent to $M = M_1 \cap M_2$. A plan is preferred if it satisfies $\phi$. Therefore, for each $A \in M$, $A|_{vocab(M_2)}$ represents a plan and $A|_{vocab(M_2)}$ is a structure satisfying basic desire $\phi$. In this setting, $\Pi = \emptyset$, and $\succ_{MP}$ is also empty. 

**Example 13.** Assume we want to travel from location $l_1$ to $l_3$ through $l_2$. The action Travel (notation $T_1$) from $l_1$ to $l_2$ can be done by taxi, bus, or train. Similarly, $T_2$ specifies travel from $l_2$ to $l_3$ by train or walking. Because of the expensive cost, two consecutive travels cannot be completed by taking a train. Also, walking is preferred to taking a train or bus. These preferences are specified by a basic desire formula, say $\phi$. The valid travel plans in the $PP$ are $p_1 = (T_1(\text{train}), T_2(\text{walk}))$, $p_2 = (T_1(\text{bus}), T_2(\text{walk}))$, $p_3 = (T_1(\text{bus}), T_2(\text{train}))$, $p_4 = (T_1(\text{taxi}), T_2(\text{walk}))$, and $p_5 = (T_1(\text{taxi}), T_2(\text{train}))$. Assume structures $A_{p_2}$ and $A_{p_3}$ are the translation of $p_2$ and $p_3$, respectively, in the $P-MS$ framework. Let $A_{p_2}$ and $A_{p_3}$ be models of a module $M_1$ such that $A_{p_2} = (D, T_1^{A_{p_2}}, T_2^{A_{p_2}})$ where $D = \{\text{taxi, walk, bus, train}\}$ is the domain and $T_1^{A_{p_2}} = (T_1(\text{bus}))$ and $T_2^{A_{p_2}} = (T_2(\text{walk}))$ are the interpretations of $T_1$ and $T_2$. Also, $A_{p_3} = (D, T_1^{A_{p_3}}, T_2^{A_{p_3}})$ where $D = \{\text{taxi, walk, bus, train}\}$, interprets $T_1$ as $T_1^{A_{p_3}} = (T_1(\text{bus}))$ and $T_2$ as $T_2^{A_{p_3}} = (T_2(\text{train}))$. Assume $M_2$ solves program $P_{\phi}$ such that for each structure $B \in M_2$, $B \not\models T_1(\text{train}) \wedge T_2(\text{train})$. For each structure $C \in M_1 \triangleright M_2$, $C|_{\{T_1, T_2\}}$ characterizes a preferred plan based on basic desire $\phi$.  

53
4.5 Conclusion and Future Work

In this chapter, we proposed an abstract framework for unifying preference-based declarative approaches in modular systems. We introduced the notion of preference-based modular systems (P-MS). We demonstrated that a system obtained through the combination of more than one preference-based modular systems is also a preference-based modular system. We studied how preferences expressed in other approaches (three preference frameworks as examples) can be related to our framework. Examples included: CP-nets, Answer Set Optimization, and planning with preferences. For future work, we will continue our study of practical aspects of our framework in AI applications, in particular, Business Processes that have complex modular structures and different users who may communicate through different formal languages.
Chapter 5

Solving Prioritized Model Expansion Problems

In this chapter, we shall discuss solving Prioritized Model Expansion problems. We propose a framework that is called Solving Prioritized Model Expansion (SPMX) to find optimal expansion structures. We present an abstract characterization of solvers in the SPMX framework, which are external tools that implement a non-deterministic algorithm to find expansion structures of Model Expansion problems or models of modular systems. Solvers would provide symbolic explanations for rejecting and accepting models, and would follow a preferred computation path to prune the search space. The notion of solvers is then generalized to preference solvers that return the most preferred structures. We propose a modular system that characterizes the optimal expansions of a Prioritized Model Expansion problem and prove a solver for that modular system is a preference solver.

5.1 Introduction

Solving a Prioritized Model Expansion problem is the task of finding its optimal expansions. We introduce Solving Prioritized Model Expansion (SPMX), a framework to solve Prioritized Model Expansion problems. In our proposal, a solver is defined as (an external tool that runs) a non-deterministic algorithm that finds models of a Model Expansion problem or models of a modular system. We define a number of requirements for solvers that guarantee the soundness of the algorithm that they implement. Solvers in AI, such as CSP solvers [89], ILP solvers [44], and SAT solvers [13] typically perform a depth-first search with propagation after each assignment (e.g., unit propagation in SAT), use back-jumping when they are unsuccessful in a search path (e.g., back-jumping in CSP [130]), and employ a number of learning techniques (e.g., clause learning in SAT solvers) to avoid going through paths with no possible solution. For example, the conflict-driven clause learning (CDCL) algorithm empowers SAT solvers by clause learning and back-jumping [14]. Our proposed approach uses a similar idea by gradually adding information to a partial model
(structure) and generates reasons (learning) if the partial model gets rejected and receives guidance (propagation) to avoid going through certain paths in the search space.

A preference solver, which is a generalization of solvers in the SPMX framework, finds optimal expansions. Every preference solver in the SPMX framework executes a basic generator-verifier algorithm as follows: A solver $s_a$ generates an expansion structure (if there exists an expansion) and then a solver $s_b$ finds a more preferable expansion. The output of $s_b$ is assigned to its input and $s_b$ finds another preferable expansion. The process continues until $s_b$ fails at generating an expansion structure. The final output of $s_b$ is an optimal expansion. This algorithm is tightly related to the idea of optimization solvers in logic programming[32, 36] in which a solver guesses a model of a generator program and another solver for a tester program verifies if the model is optimal. Similar to solvers in the SPMX framework, a preference solver can also be advised to select search paths which wind up reaching preferred structures or avoid some paths which do not end up in a preferred structure.

Interestingly, the generator-verifier algorithm can be expressed by operations of modular systems using a special type of fixed point, composition, and projection. The main focus of this chapter is to establish a relationship between modular systems and Prioritized Model Expansion. Thereby, a preference solver can be constructed using solvers of modular systems. We propose a modular system $M$ that characterizes optimal expansions of a Prioritized Model Expansion problem $\Pi_{\sigma,\psi}$. We prove that a solver for $M$ is a preference solver for $\Pi_{\sigma,\psi}$.

This chapter is organized as follows: We begin by taking a closer look at the lifting methods (preference semantics) introduced in Chapter 3 that are used in constructing our proposed preference solvers. After that, we introduce a naive algorithm for solving Prioritized Model Expansion problems. The naive algorithm simply constructs a model and compares it with already computed models to determine whether it is optimal. Next, solvers are defined by adding guidance (propagation) and reasoning (learning) to the naive algorithm. After laying out the notion of solvers in the SPMX framework, we introduce preference solvers that find optimal expansions. We define a number of properties that preference solvers share. Finally, we introduce a method to construct preference solvers by means of solvers of modular systems.

### 5.2 Preference Relation Semantics

Recall from Chapter 3 that a preference expression is an order over the set of ground atoms of an input domain $Dom$ and a fixed vocabulary $\tau$. In Definition 5, we proposed three preference semantics to build a preference relation among structures: Weak Pareto, Upper Bound Dominance, and Element Dominance. The idea behind Definition 5 is to lift a preference relation over ground atoms to a preference relation among structures as long as the Dominant Structure problem in Chapter 3 is in polynomial time. Our proposed framework is flexible such that the preference semantics are not limited to Definition 5 and other lifting methods are allowed on the condition that for a lifting method, the Dominant structure problem is polynomially solvable. This condition guarantees that
the other complexity results in Chapter 3 are applicable. For example, the Strong Element Domi-
nance (SED) semantics is another weaker type of the Weak Pareto semantics such that a structure is
preferred to another structure if every ground atom satisfied by the former is preferred to a ground
atom satisfied by the latter. Formally,
\[ A \geq_{\text{sed}} P B \]
if for all \( R \in \tau \) and for each \( \bar{a} \in R^A \), there is a \( \bar{b} \in S^B \) for some \( S \in \tau \) such that \( R(\bar{a}) \sqsupset P S(\bar{b}) \).

As another example, for \( \tau \)-structures \( A \) and \( B \) with domain \( \text{Dom} \) and preference expression \( P = (S_\tau, \succeq_P) \), the Weak Element Dominance (WED) semantics indicates that it is enough to find a ground atom that is satisfied by \( A \) and is preferred to a ground atom satisfied by \( B \) to derive that \( A \) is favoured over \( B \). In other words, \( A \geq_{\text{ed}} P B \) if for some \( R, S \in \tau \), there is \( \bar{b} \in S^B \) and there is \( \bar{a} \in R^A \) such that \( R(\bar{a}) \sqsupset P S(\bar{b}) \).

The relation \( \geq_{\text{wed}} P \) is not transitive and also there may be a cyclic preference relation such that \( A >_{\text{wed}} P B \) and \( B >_{\text{wed}} P A \) (where there are elements in \( A \) and \( B \) that are preferred to each other) that is not intuitively sound.

The Strong Element Dominance semantics models the Smyth method of lifting in [37]. Also, the
well-known notion of the Hoare lifting method [37] corresponds to the Upper Bound Dominance
(UBD).

We take a closer look at an underlying hierarchical relationship among these lifting methods.

**Definition 24.** Let \( s \) and \( s' \) be two preference semantics. We say that \( s' \) subsumes \( s \) when for all \( A \) and \( B \), if \( A \geq_{s'} P B \), then \( A \geq_{s'} P B \).

A graphical representation of the subsumption relation between lifting methods is illustrated in
Figure 5.1 such that a directed edge exhibits a subsumption relation. This graph is called the prefer-
ence semantics hierarchy. In Figure 24, WP denotes the Weak Pareto semantics, UBD stands for the
Upper Bound Dominance semantics, ED is the Element Dominance semantics, WED is the Weak
Element Dominance semantics, and SED stands for the Strong Element Dominance semantics.

As before, the symbol \( \geq_{s'} P \) denotes a binary relation among structures under the semantics of
\( s \) that can be one of the semantics in the preference semantics hierarchy. It is clear that the sub-
sumption relation is transitive and the Upper Bound Dominance and Strong Element Dominance
semantics subsume the Weak Pareto semantics. Also, the Weak Element Dominance semantics sub-
sumes the Element Dominance and Upper Bound Dominance semantics. Furthermore, deciding
whether \( A \geq_{s'} P B \) is solvable in polynomial time of the size of \( \text{Dom} \) due to the fact that for every
preference semantics in the hierarchy, we need to compare at most all tuples in \( R^A \) and \( S^B \) for
all \( R, S \in \tau \). The total possible number of \( k \)-ary tuples is \( |\text{Dom}|^k \). Therefore, \( O(|\text{Dom}|^{2k}) \) com-
parisons are required for each \( R \in \tau \) where \( k \) is the maximum arity of the predicate symbols in
\( \tau \).

### 5.3 Naive Algorithm

In this section, we propose an algorithm for finding optimal expansions of Prioritized Model Ex-
pansion problems. Let us first elaborate on the notion of a partial structure, which was introduced
in Chapter 4, that is a tuple containing a domain with partial interpretation of some vocabulary
symbols.
**Definition 25.** For each predicate $R \in \tau$, where $R$ is a $k$-ary predicate, and for all $k$-ary tuple $\pi \in \text{Dom}^k$, a partial $\tau$-structure $\mathfrak{B}$ interprets $R(\bar{t})$ as true (i.e., $\bar{t} \in R^\mathfrak{B}$), false (i.e., $\bar{t} \not\in R^\mathfrak{B}$), or unknown (i.e., neither $\bar{t} \in R^\mathfrak{B}$ nor $\bar{t} \not\in R^\mathfrak{B}$), which are abbreviated as $t, f$, and $u$, respectively, where $\mathfrak{B} \models R(\bar{t})$ if $R(\bar{t})$ is interpreted as true, $\mathfrak{B} \not\models R(\bar{t})$ when $R(\bar{t})$ is interpreted as false, and neither $\mathfrak{B} \models R(\bar{t})$ nor $\mathfrak{B} \models R(\bar{t})$ when $R(\bar{t})$ is interpreted as unknown.

We define $S^f_B$ as the set of all ground atoms that are assigned as true under the interpretation $\mathfrak{B}$. Similarly, $S^I_B$, and $S^u_B$ are associated with each partial structure $\mathfrak{B}$. A partial structure $\mathfrak{B}$ with no unknown value is a full or total structure. A partial structure is augmented if it has less unknown information.

**Definition 26.** For two partial structures $\mathfrak{B}$ and $\mathfrak{B}'$ of a vocabulary $\tau$ and a domain $\text{Dom}$, we say $\mathfrak{B}$ is augmented to $\mathfrak{B}'$ with notation $\mathfrak{B} \Rightarrow \mathfrak{B}'$, if $S^f_B \subseteq S^f_{B'}$ and $S^I_B \subseteq S^I_{B'}$.

From the above definition, if $\mathfrak{B}$ is augmented to $\mathfrak{B}'$, then $S^u_B \subseteq S^u_{B'}$.

A formula is falsified by a partial structure if it is falsified by the known part (the interpretation of ground atoms that are either true or false) provided by the partial structure. Given a sentence $\phi$ in a certain logic $\mathcal{L}$ and a partial $\tau$-structure $\mathfrak{B}$, we say $\mathfrak{B}$ falsifies $\phi$ with notation $\mathfrak{B} \not\models \phi$ if $\phi$ is false based on $\{\bar{t} | \bar{t} \in R^\mathfrak{B} \text{ or } \bar{t} \not\in R^\mathfrak{B} \text{ for } R \in \tau\}$.

For example, let $\tau = \{R, S\}$ and $\text{Dom} = \{v\}$ where $R$ and $S$ are unary predicates. Let $\phi$ be defined as $\forall x (R(x) \land \neg S(x))$. Assume a partial $\tau$-structure $\mathfrak{B}$ interprets $S$ such that $v \in S^\mathfrak{B}$ while it is unknown whether $v \in R^\mathfrak{B}$. It is clear that $\mathfrak{B}$ falsifies $\phi$ because $v \in S^\mathfrak{B}$ provides enough information to deduce that $\phi$ is false under the interpretation of $\mathfrak{B}$ no matter what the interpretation of $R$ is.

We propose Algorithm 1 that uses a brute-force search to find optimal expansions of a Prioritized Model Expansion problem. In this algorithm, the input structure is a $\sigma$-structure $\mathcal{I}$ and output $S$ is the set of all preferred $\tau$-structures (where $\tau = \text{vocab}(\psi)$ and $\sigma \subseteq \tau$) that expands $\mathcal{I}$ and satisfies $\psi$. Partial structure $\mathfrak{B}$ is initialized to $\mathcal{I}$ such that for all $R \in \sigma$, $R^\mathfrak{B}(\bar{t}) = R^\mathcal{I}(\bar{t})$ and for all other $T \in \tau \setminus \sigma$, the value of $T^\mathfrak{B}(\bar{t})$ is set to unknown ($\mathfrak{B}$ is called an absolute $(\tau \setminus \sigma)$-partial structure).

**Theorem 10.** The naive algorithm is sound and complete and its running time is $O(n^{2k} \times 2^{O(n^{2k})})$ where $n$ is the size of the domain of $\mathcal{I}$ and $k$ is the maximum arity of the predicates in vocabulary $\tau$.
Algorithm 1 Naive Algorithm for Prioritized Model Expansion

**procedure** SOLVE \((\mathbf{M}, \sigma, \psi, \mathcal{I}, \mathcal{P}, s)\)
- set \(S = \emptyset\)
- while there is an element that can be added to \(S\) do
  - Initialize partial \(\tau\)-structure \(\mathcal{B}\) to \(\mathcal{I}\)
  - while there is \(R(\pi)\) where \(R^\mathcal{B}(\pi)\) is unknown do
    - Assign true or false to \(R^\mathcal{B}(\pi)\)
  - end while
  - if \(\mathcal{B}|\mathcal{I}\models \psi\) then
    - if there is \(S \in S\) such that \(\mathcal{B} >^s_p S\) then
      - set \(S = S - \{S\}\)
    - end if
  - if there is no \(S \in S\) such that \(S >^s_p \mathcal{B}\) then
    - set \(S = S \cup \{\mathcal{B}\}\)
  - end if
- end while
- return \(S\)

**Proof.** The soundness comes from the fact that \(\mathcal{B}\) is augmented to a (full) structure. Then, the algorithm verifies whether \(\mathcal{B}\) satisfies \(\psi\) and whether there is a structure in \(S\) that dominates \(\mathcal{B}\). If there is such a structure, then \(\mathcal{B}\) will not be in \(S\). Otherwise, it is added to \(S\) and will remain in the set of preferred expansion structures unless in successive iterations it is replaced by better structure. Since the algorithm searches through the space of all possible structures, it is also complete. According to the definition of preference semantics in the subsumption hierarchy, given two structures, finding the preferred one (which is also called the Dominant Structure problem) requires comparing, at most, all pairs of tuples in the interpretation of predicate symbols. The maximum possible number of tuples of arity \(k\) is \(n^k\). Therefore, given the fact that the size of vocabulary \(\tau\) is fixed, the total number of comparisons is \(O(\mathcal{n}^{2k})\). On the other hand, there are at most \(2^{O(n^k)}\) expansion structures that can be generated over a domain with size \(n\) and predicates with maximum arity size \(k\). Starting from the initial iteration where \(S\) is empty, until the last iteration \(2^{O(n^k)}\) (in the worst case scenario), we have at most \(1, 2, ..., 2^{O(n^k)}\) Dominant Structure problems between two structures at each step, respectively. Thus, the total number of operations is \(O(n^{2k} \times 2^{O(n^2k)})\). \(\square\)

If the task is to determine the optimal structures of a prioritized modular system \((\mathbf{M}, \succ_P)\), the step to check whether \(\mathcal{B} \models \psi\) in Algorithm 1 is replaced by whether \(\mathcal{B} \models M\). Since module \(M\) is a set of structures and has no computational power, we assume a decision procedure \(D\) is associated with \(M\) that accepts a structure \(A\) if and only if \(A \models M\). Also, \(D\) may provide guidances and reasons of a rejection to the solver, as will be discussed in the next section.
5.4 Solvers for Model Expansion Problems and Modular Systems

The authors of [127] introduced an abstract characterization of solvers in modular systems inspired by the lazy approach in Satisfiability Modulo Theory (SMT) [6]. DPLL(T) [103] is a popular architecture for SMT solvers based on the lazy approach, which is a SAT solver combined with one or more theory solvers. The SAT solver finds a model of the problem specifications formula and sends it to the theory solver. If the model is $T$-satisfiable ($T$ is the background theory), then it is accepted. Otherwise, the SAT solver finds another model. The process continues until a model consistent with the background theory is found. If all propositional models of the formula are checked without finding a $T$-consistent model, then the solver returns "unsatisfiable". It was shown in [127] that DPLL(T) can be modeled as a modular system and, hence, a solver for modular systems can be used for DPLL(T). Likewise, constraint ASP solvers [7] and ILP solvers [44] can be modeled by modular systems.

As discussed in Chapter 4, a module with an input structure specifies a Model Expansion problem. Thus, a solver for a modular system can be viewed as a solver for its related Model Expansion problem. Our proposal and [127] share a main principle: A solver continues to augment a partial structure if the partial structure is not rejected based on a knowledge base provided by the specifications of a Prioritized Model Expansion problem or a decision procedure associated with a module. The soundness of the method is guaranteed because a solver of a module (or a Model Expansion problem) does not reject any partial structure that can be augmented to a model of the module. However, a partial structure that will not be augmented to any model of the module (or expansion structures of a Model Expansion problem) may not be rejected until the very end of the process.

We always assume a decision procedure $D$ is associated with a modular system $M$ that assists the solver. Decision process $D$ accepts a structure if it is a model of $M$ and rejects it otherwise. Also, $D$ helps the solver by giving reasons and guidances. We first define the notion of a solver of modular systems in the SPMX framework. Solver $s$ is specified by $s(M, I, \Gamma_s)$ where $M \in MS(\sigma, \epsilon)$ is a module, $I$ is an input structure, and $\Gamma_s$ is a set of sentences (all variables are bound) in a certain language $L_s$. Set $\Gamma_s$ is called the solver knowledge base and consists of formulas generated by decision process $D$.

**Definition 27.** Solver $s(M, I, \Gamma_s)$ is a non-deterministic algorithm that either returns a $\tau$-structure $A \in M$ ($\tau = \text{vocab}(M)$) such that $A \models \Gamma_s$, $A \models M$, and $A \xrightarrow{\Delta_s} A$ where $A$ is an absolute $\epsilon$-partial $\tau$-structure and $A|_\sigma = I|_\sigma$ or it returns $\bot$, which denotes $M$ has no model.

Solver $s$ for Model Expansion problem $MX_{\sigma, \psi}$ with input $\sigma$-structure $I$ returns an expansion (if there is an expansion) structure of $MX_{\sigma, \psi}$. Solvers can be implemented by a variety of methods using different heuristics to reduce the search space and accelerate the search. Generally speaking, $s$ gradually augments $A$ and at each step checks whether $A$ does not falsify $\Gamma$ to prevent an augmentation to a partial structure that cannot be augmented to a model of $M$. The solver determines the next augmentation of $A$ non-deterministically or by the guidance of formulas in $\Gamma_s$.  

60
Decision procedure \(D\) provides reason and guidance in the form of formulas in a language \(L_s\), which is called the solver language. Here, we consider a simple language such that each ground atom \(T(\pi)\) is a formula in \(L_s\). Also, for formulas \(\phi_1\) and \(\phi_2\) in \(L_s\), \(\phi_1 \land \phi_2\), \(\phi_1 \lor \phi_2\), and \(\neg \phi_1\) are in \(L_s\). The logical connectives are interpreted the same as in propositional logic. The solver language can have more expressive power with a larger set of logical connectives. We expect a solver language to be monotonic and at least as expressive as propositional logic similar to [127].

For a Model Expansion problem \(M_{\sigma,\psi}\) with an input \(I\), the decision process determines whether a partial structure \(A\) augments \(A_0\) (\(A_0|\sigma = I|\sigma\)) and whether it falsifies \(\psi\) or formulas in \(G_s\). If a partial structure is rejected, the reason is created and added as a formula to \(G_s\).

In our framework, for a fixed vocabulary \(\tau\) and a given domain \(Dom\), the search space is the set of all possible partial \(\tau\)-structures with domain \(Dom\). Solver \(s\) assigns true or false to all ground atoms \(R(\bar{t})\) with unknown value where \(R \in \tau\) is a \(k\)-ary predicate and \(\bar{t} \in Dom^k\). For a vocabulary of \(m\) predicate symbols of arity \(k\), the size of the search space is \(3^{nmk}\) where \(n\) is the size of \(Dom\).

A partial \(\tau\)-structure \(B\) is called a state of a solver \(s(M, I, G_s)\) if \(|B|_{\sigma} = I|_{\sigma}\), and \(B\) does not falsify \(G_s\). Let \(B\) be a partial structure. We say \(B'\) is a successor of \(B\) if \(B \models B'\) (\(B'\) is an augmentation of \(B\)) and \(B'\) has exactly one more ground atom that is unknown in \(B\) (which is called a distinguishing ground atom) with a true or false value. Also, \(B\) is called a predecessor of \(B'\). A sentence \(\phi\) in language \(L\) is satisfied in a solver state \(A\) if and only if \(A \models \phi\). When the current state of the solver is augmented to a partial \(\tau\)-structure that is rejected by \(D\) (or it falsifies \(\psi\) for an \(MX_{\sigma,\psi}\) problem) or formulas in \(G_s\), we say that the solver is in a dead state. When all search paths starting from the initial state end in a dead state, the solver returns \(\bot\). When the current state of the solver has no ground atom with unknown value, we say that the solver has reached to a final state. We call \(B\) a goal state if \(B\) is a final state and a model of \(M\). A solver path is a sequence \(\rho = B_0, ..., B_q\) that starts from initial state \(B_0\) (i.e., \(B_0|\sigma = I|\sigma\)) and ends at a final state \(B_q\).

The reason for moving to a dead state or generating a guidance for augmenting the current state of the solver toward a certain type of states can be used by the solver for walking through the search space more efficiently. Any reason or guidance formula in the solver language \(L_s\) is added to the solver knowledge base \(G_s\). The knowledge base \(G_s\) of a solver \(s\) is defined as a set of formulas in language \(L_s\) that are satisfied in all goal states. At each step of the search, the solver tries to move to a new state that is not inconsistent with the knowledge base.

The reason for transitioning to a dead state is specified as a formula in the solver language \(L_s\) as follows:

**Definition 28.** For a solver \(s\), the reason for transitioning to a dead state \(B\) is \(\neg \phi\) where \(\phi\) is a formula in the solver language \(L_s\) that is satisfied in \(B\) while it is not satisfied in any goal state.

Let us comment briefly on the solver reasoning mechanism. Assume \(M\) is a module which specifies a Model Expansion problem \(MX_{\sigma,\psi}\). For example, if \(C\) is a dead state of solver \(s(M, I, G_s)\), a formula \(\neg \phi\) in the solver language is added to knowledge base \(G_s\) such that \(C \models \phi\) and \(C \models \phi \rightarrow \)
Figure 5.2: A Preferred-Goal State Achieved by a Preference Solver for Graph Three-Colouring

$\neg \psi$. In the subsequent iterations, the solver will not go to state $C$ and also will not move to any state in which, similar to $C$, the formula $\phi$ is satisfied.

In addition to adding reasons to avoid dead states, solvers can also be assisted by a decision process $D$ to move to certain states.

**Definition 29.** A formula $\phi$ in the solver language $L_s$ is a guidance to solver $s$ in state $B$, if it is of the form $(\phi_{11} \lor ... \lor \phi_{1n_1}) \land ... \land (\phi_{l1} \lor ... \lor \phi_{ln_l})$ where $\phi_{ij}$, $1 \leq i \leq l$ and all $j \in \{1, ..., n_1\} \cup ... \cup \{1, ..., n_l\}$, is a literal in $L_s$ (an atomic formula or its negation) and some $\phi_{ij}$s are satisfied in $B$ and $\phi$ is satisfied in all goal states.

According to the above definition, a guidance for a solver $s$ is a formula in clausal (conjunctive) normal form (CNF) in the solver language. Since some literals are satisfied, the truth value of some atomic formulas (i.e., ground atoms) can be decided by the propagation of truth values.

**Example 14.** Consider the N-queen problem axiomatization in first-order logic where predicate $Q(x,y)$ denotes the presence of a queen in the location $(x,y)$ on a board. The formula $\psi_1 = \forall x \forall y, \forall z (Q(x,y) \land y \neq z \supset \neg Q(x,z)) \land \forall x \exists y Q(x,y)$ specifies the constraint of each column. Let $\psi_2$ and $\psi_3$ be the constraints for each row and diameter, respectively (similar to $\psi_1$, but we do not elaborate to avoid redundancy). The formula $\psi = \psi_1 \land \psi_2 \land \psi_3$ is the axiomatization of the problem. For an $8 \times 8$ board, assume solver $s$ assigns true to $Q(1,3)$. When $Q(2,2)$ is assigned true, $s$ goes to a dead state. Thus, the solver adds reason $\neg Q(2,2) \lor \neg Q(1,3)$ to $\Gamma_s$ to prevent making both $Q(1,3)$ and $Q(2,2)$ true in the future.

Algorithm 2 outlines a procedure for solver $s$. When solver $s$ reaches to a dead state, similar to the idea of solvers with propagators in [16], it backtracks to the previous step and adds possible reasons to avoid returning to the same dead state in following iterations. Also, after the augmentation of $\mathcal{B}$, a guidance formula may be added to $\Gamma_s$. After moving to a new state and adding any reason or guidance to $\Gamma_s$, the classical logic closure of $\Gamma_s$ (deductive closure) with notation $Th(\Gamma_s)$ is calculated such that for a formula $\phi \in L_s$, if $\Gamma_s \vdash \phi$, then $\phi \in Th(\Gamma_s)$. 
Algorithm 2 Solver in the SPMX Framework

procedure $s(M, I, \Gamma_s)$
set $B|_{\sigma} = I|_{\sigma}$
while TRUE do
    if $B$ is a final state then
        if $B$ is a goal state then
            return $B$
        else
            set $B = B'$ where $B'$ is a predecessor of $B$
            add reason $\phi$ to $\Gamma_s$
        end if
    end if
    if there is a successor state $B'$ then
        set $B = B'$
        add guidance $\phi'$ to $\Gamma_s$
    else
        set $B = B'$ where $B'$ is the predecessor of $B$
        add reason $\phi''$ to $\Gamma_s$
    end if
    if all successors states of $B$ are dead states then
        mark $B$ as a dead state
        set $B = B'$ where $B'$ is the predecessor of $B$
    end if
end while
return $\perp$
end procedure

5.5 Preference Solvers for Prioritized Model Expansion Problems

In this section, we introduce preference solvers for Prioritized Model Expansion problems. The idea behind our proposal can also be used to find preferred structures of preference-based modular systems, introduced in Chapter 4.

5.5.1 General Properties of Preference Solvers

A preference solver $s_p$ is denoted by $s_p(\Pi_{\pi,\psi}, I, \Gamma_s, y)$ where $\Pi_{\pi,\psi} = (MX_{\pi,\psi}, P)$ is a Prioritized Model Expansion problem, $I$ is a problem instance (input), $\Gamma_s$ is the solver knowledge base, $P$ is a preference expression, and $y$ is a preference semantics in the preference semantics hierarchy.

Definition 30. A preference solver $s_p(\Pi_{\pi,\psi}, I, \Gamma_s, y)$ is a non-deterministic algorithm that returns either an optimal expansion structure of $\Pi_{\pi,\psi}$ or $\perp$ if there is no expansion.

It is worth mentioning that $s_p$ only returns $\perp$ when $MX_{\pi,\psi}$ has no expansion structure. A goal state $A$ is dominant to another goal state $B$ with respect to $P$ based on semantics $y$ if $A >^y_p B$. For a preference solver $s_p$, a preference-goal state is a goal state that is not dominated by other goal
states. Preference-dead state $\mathfrak{A}$ is a state of the solver when it will only transit (can be augmented) to final states that are not preference-goal states. A solver goes to a preference-dead state when the only available paths are paths to final states that are not preference-goal states. It is evident that a preference-dead state is not necessarily a dead state and a preference-goal state is a goal state.

Similar to solvers in modular systems, a preference solver can also be advised to avoid transitioning to preference-dead states. We introduce the reason for reaching a preference-dead state or guidance to move to a preferable state based on preferences of a solver.

**Definition 31.** A preference reason for preference solver $s_p(\Pi_{\sigma,\psi}, I, \Gamma_s, y)$ that moved to preference-dead state $\mathfrak{B}$, is the negation of a formula $\phi$ in the solver language $L_s$ that is not satisfied in any preference-goal state but it is satisfied in $\mathfrak{B}$.

It is worth noting that this definition characterizes the general property of any preference reason and does not take any preference semantics into account. A reason to refrain from entering into a preference-dead state for the Weak Pareto semantics is illustrated in the following example.

**Example 15.** Let $s_p(\Pi_{\sigma,\psi}, I, \Gamma_s, y)$ be a preference solver. Assume $\varphi = T(\bar{\pi}_1) \lor T(\bar{\pi}_2)$ is a formula such that $\varphi \in \Gamma_s$. Assume $s$ reaches to a preference-dead state $\mathfrak{B}$ because $T(\bar{\pi}_1)$ and $T(\bar{\pi}_2)$ are atomic formulas in $L_s$ where $T(\pi_1) \sqsupset_p T(\pi_2)$ and $\mathfrak{B} \models T(\pi_1) \land \neg T(\pi_2)$. Assigning $T(\pi_2)$ the value true and $T(\pi_1)$ the value false would lead $s_p$ to a final state that is dominated by another final state in which $T(\bar{\pi}_2)$ is false and $T(\bar{\pi}_1)$ is true. So, adding $\neg(T(\bar{\pi}_1) \land \neg T(\bar{\pi}_2))$ to the knowledge base prevents the solver from moving to a final state where it would be dominated. Therefore, $\neg(T(\bar{\pi}_1) \land \neg T(\bar{\pi}_2))$ is a preference reason.

A preference solver can be assisted by preferences of the decision maker. A preference guidance leads the solver to move to a preference-goal state.

**Definition 32.** A preference guidance in a solver state $\mathfrak{B}$ is a formula in the solver language if it is of the form $(\phi_{i1} \lor ... \lor \phi_{in_1}) \land ... \land (\phi_{i1} \lor ... \lor \phi_{in_l})$ where $\phi_{ij}$, for $1 \leq i \leq l$ and all $j \in \{1, ..., n_1\} \cup ... \cup \{1, ..., n_l\}$, is a literal in $L_s$, some $\phi_{ij}$s are satisfied in $\mathfrak{B}$, and $\phi$ is satisfied in all preference-goal states.

From the above definition it can be observed that the difference between a guidance formula and a preference guidance formula is that the former must be satisfied in all goal states while the latter is satisfied in all preference-goal states. This is a general characterization of a preference guidance, similar to preference reasons, regardless of preference semantics.

**Example 16.** Consider the formula $\neg S(T_3) \lor R(T_1) \lor R(T_2)$ and let $R(T_1)$ be preferred to $R(T_2)$. By assigning $S(T_3)$ as true, the solver also assigns $R(T_1)$ true and $R(T_2)$ false.

**Example 17.** Consider the problem of graph three-colouring. Let $G = (V, E)$ be the input graph where $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $E^G = \{(v_1, v_2), (v_2, v_1), (v_1, v_3), (v_3, v_1), (v_2, v_3), (v_3, v_2), (v_3, v_4), \ldots\}$.
Let \( v_4 \) with the colour red be preferable to red \( v_5 \). Assume \( \psi \) is the characterization of three-colouring in first-order logic. Preference solver \( s_p \) initializes a \( \{E, R, G, B\} \) structure \( \mathcal{B}_0 \) such that \( E^\mathcal{G} = E^A \) and the interpretation of \( R, G, \) and \( B \) are unknown. Assume in the first step, \( s_p \) goes to state \( \mathcal{B}_1 \) with distinguishing atom \( R(v_1) \) having the value true. Based on the axiomatization \( \psi \) and preferences, \( s \) adds \( (R(v_4) \lor R(v_5)) \land (\neg R(v_5) \lor R(v_4)) \) to \( \Gamma \). In the next two steps, \( s_p \) moves to \( \mathcal{B}_2 \) and \( \mathcal{B}_3 \) where \( G(v_2) \) and \( B(v_3) \) are assigned the value true. Then, \( s_p \) goes to \( \mathcal{B}_4 \) in which \( \mathcal{B}_4 \models B(v_5) \) and, based on \( \Gamma \), \( s_p \) is enforced to go to goal state \( \mathcal{B}_5 \) in which \( R(v_4) \) is true and \( R^{\mathcal{B}_5} = \{v_1, v_4\}, G^{\mathcal{B}_5} = \{v_2\}, \) and \( B^{\mathcal{B}_5} = \{v_3, v_5\} \).

The following example shows a preference guidance for the Upper Bound semantics in the preference semantics hierarchy.

**Example 18.** Let \( \phi_1, \phi_2, \) and \( \phi_3 \) be ground atoms \( R(\bar{t}), R(\bar{t}') \), and \( R(\bar{t}'') \) respectively. Assume for a prioritized model expansion \( \Pi = (MX_{\mathcal{I},\psi}, P) \), solver \( s_p(\Pi_{\sigma,\psi}, \mathcal{I}, \Gamma_\sigma, ubd) \) \( \phi_1 \lor \phi_2 \lor \phi_3 \) is in the knowledge base \( \Gamma_\sigma \) and \( R(t) \triangleright_P R(t') \triangleright_P R(t'') \). Formula \( \Phi = (\neg \phi_3 \lor \phi_2) \land (\neg \phi_2 \lor \phi_1) \) is a preference guidance. The reason is as follows: \( \Phi \) is a CNF formula in the solver language such that by assigning truth to \( \phi_3 \), \( \phi_2 \) must be true, otherwise, when the solver reaches a goal state \( \mathcal{B} \), there exists another goal state in which \( \phi_2 \) is true that dominates \( \mathcal{B} \) based on the Upper Bound Dominance semantics. Following the same reasoning, clause \( (\neg \phi_2 \lor \phi_1) \) is also a preference guidance. In other words, when \( \phi_3 \) is true, \( \phi_1 \) and \( \phi_2 \) must be true. Otherwise, the solver will go to a state that is not a preference-goal state.

### 5.5.2 An algorithm for Preference Solvers

We propose an algorithm for preference solvers based on an iterative procedure as follows:

**Algorithm 3** A Program for Finding an Optimal Expansion of \( \Pi_{\sigma,\psi} \)

```markdown
procedure \( F(\Pi_{\sigma,\psi}, \mathcal{I}, y) \)
Find an expansion structure of \( MX_{\sigma,\psi} \) and assign it to \( \mathcal{A} \)
while TRUE do
  if \( MX_{\sigma,\psi} \) has an expansion \( \mathcal{B} \) such that \( \mathcal{B} \succ_P^y \mathcal{A} \) then
    set \( \mathcal{A} = \mathcal{B} \)
  else return \( \mathcal{A} \)
end if
end while
return \( \mathcal{A} \)
end procedure
```

In Algorithm 3, an expansion of \( MX_{\sigma,\psi} \) (if there is an expansion) is found and assigned to \( \mathcal{A} \). In the While loop, an expansion of \( MX_{\sigma,\psi} \) that is preferred to \( \mathcal{A} \) is generated until there is no more such expansion is found. The final result is an optimal expansion.

Algorithm 4 demonstrates a pseudo code for a preference solver, based on the idea behind Algorithm 3, that finds an optimal expansion of a Prioritized Model Expansion problem \( \Pi_{\sigma,\psi} \).
the first stage, an expansion of $\text{MX}_{\sigma,\psi}$, say $\mathfrak{A}$, is generated by running Algorithm 2. Then, partial structure $\mathfrak{B}$ is augmented. If $\mathfrak{B}$ does not falsify any formula in $\Gamma_s$ and $\mathfrak{B} \succ_P \mathfrak{A}$ is not false, then related guidances and preference guidances are added to $\Gamma_s$ and the solver moves to a successor state. Otherwise, related reasons or preference reasons are added to $\Gamma_s$ and the solver moves back to the predecessor state. When the solver reaches to a goal state, say $\mathfrak{C}$, it considers it to be a candidate preferred-goal state. If the solver finds another goal state, say $\mathfrak{F}$, where $\mathfrak{F} \succ_P \mathfrak{C}$, then $\mathfrak{C}$ is updated to $\mathfrak{F}$. While $\mathfrak{C}$ is a candidate preference-goal state, the preference reasons and the preference guidances are generated with respect to $\mathfrak{C}$. This process continues until $\mathfrak{C}$ can not be updated to a new goal state. The result, which is the last update of $\mathfrak{C}$, is an optimal expansion of $\Pi_{\sigma,\psi}$.

**Proposition 5.** Algorithm 4, which returns an optimal expansion of $\Pi_{\sigma,\psi}$, is sound.

**Proof:** Partial structure $\mathfrak{B}$ is augmented if it does not falsify the solver knowledge base and it backtracks if $\mathfrak{B} \not\succ_P \mathfrak{A}$. At each step, the solver moves to a new state that does not falsify any formula in $\Gamma_s$, which guarantees that no reason, guidance, preference reason, or preference guidance formula is falsified when a preference-goal state is achieved. Thus, Algorithm 4, which describes preference solver $s_P$, is sound. 

### 5.6 Constructing Preference Solvers from Solvers of Modular Systems

In this section, we introduce a method to construct a preference solver for finding optimal expansions of a Prioritized Model Expansion problem $\Pi_{\sigma,\psi} = (\text{MX}_{\sigma,\psi}, P)$ by using solvers of modular systems.

#### 5.6.1 Prioritized Model Expansion Problems as Modular Systems

First, we characterize each Model Expansion problem in Algorithm 3 as a modular system as shown in Figure 5.3.

Assume $\text{Dom}$ is the domain of the input and expansion structures of $\text{MX}_{\sigma,\psi}$. Consider modules $M_1$ and $M_2$ that specify Model Expansion problems $\text{MX}_{\sigma_1,\psi_1}$ and $\text{MX}_{\sigma_2,\psi_2}$, respectively. As shown in Figure 5.3, module $M_1$ outputs an expansion of $\text{MX}_{\sigma,\psi}$ (if there is such an expansion). The output of $M_1$ is fed to the input of $M_2$, which finds an expansion of $\text{MX}_{\sigma,\psi}$ that is preferred to the output of $M_1$ with respect to preference expression $P$. Then, if $M_2$ has such an output, by means of a loop, its output is set as its input and the process continues until there is no more change or, in other words, an optimal expansion is obtained.

Let us introduce auxiliary unary predicates $\text{Mod}_1$ and $\text{Mod}_2$. Predicate $\text{Mod}_1$ represents the ground atoms satisfied by an expansion structure of $\text{MX}_{\sigma,\psi}$. Predicate $\text{Mod}_2$ specifies ground atoms satisfied by an expansion of $\text{MX}_{\sigma,\psi}$, which is preferred to the structure specified by $\text{Mod}_1$. Let $\mathcal{S}_\tau = \{a_1, \ldots, a_l\}$ be the set of all ground atoms over vocabulary $\tau$ and domain $\text{Dom}$. We define a bijective function $f: \mathcal{S}_\tau \to G$ that maps each ground atom in $\mathcal{S}_\tau$ to an element in $G$ where $G =$
Algorithm 4 Preference Solver in SPMX Framework

```plaintext
procedure \( s_p(\Pi_{\sigma,\psi}, I, \Gamma_s, y) \)
    set \( \mathcal{B}_{|\sigma} = I \)
    if \( s(MX_{\sigma,\psi}, I, \Gamma_s) = \bot \) then
        return \( \bot \)
    else
        set \( \mathcal{A} = s(MX_{\sigma,\psi}, I, \Gamma_s) \)
        while TRUE do
            if \( \mathcal{B} \) is a final state then
                if \( \mathcal{B} \) is a goal state then
                    if \( \mathcal{B} >^y_p \mathcal{A} \) then
                        set \( \mathcal{A} = \mathcal{B} \)
                        assign unknown to \( \mathcal{B}_{|\varepsilon} \)
                    else
                        add preference-reasons to \( \Gamma_s \)
                        set \( \mathcal{B} = \mathcal{B}' \) where \( \mathcal{B}' \) is the predecessor of \( \mathcal{B} \)
                    end if
                else
                    set \( \mathcal{B} = \mathcal{B}' \) where \( \mathcal{B}' \) is the predecessor of \( \mathcal{B} \)
                    add reason \( \phi \) to \( \Gamma_s \)
                end if
            else
                if there is a successor state \( \mathcal{B}' \) then
                    set \( \mathcal{B} = \mathcal{B}' \)
                    add guidance \( \phi' \) to \( \Gamma_s \)
                    if \( \mathcal{B}' \neq_p \mathcal{A} \) then
                        add preference reasons
                        replace \( \mathcal{B} \) with its predecessor
                    else
                        add preference guidance to \( \Gamma_s \)
                    end if
                else
                    set \( \mathcal{B} = \mathcal{B}' \) where \( \mathcal{B}' \) is the predecessor of \( \mathcal{B} \)
                    add reason \( \phi'' \) to \( \Gamma_s \)
                end if
            end if
            if all successor states of \( \mathcal{B} \) are (preference) dead states then
                mark \( \mathcal{B} \) as a (preference) dead state
                set \( \mathcal{B} = \mathcal{B}' \) where \( \mathcal{B}' \) is the predecessor of \( \mathcal{B} \)
            end if
        end while
        return \( \mathcal{B} \)
    end procedure
```

\( \{ g_1, ..., g_l \} \) is a set of elements with the same size as \( S_\tau \). Let \( Pr \) be a binary predicate that represents the preference relation among ground atoms. Assume \( \tau = vocab(\psi) \) and \( Dom^* = Dom \cup G \). Let
us define function \( h_i^* \) such that for \( k \)-ary \( R_i \in \tau \) and each \( \overline{a} \in \text{Dom}^k \), we have \( h_i^*(\overline{a}) = f(R_i(\overline{a})) \). Now, for each \( k \)-ary predicate \( R_i \in \tau \), we consider a function symbol \( h_i \). The intended interpretation of \( h_i \) for each expansion structure \( A \) of \( M_1 \) is a function \( h_i^A : \text{Dom}^k \rightarrow G \) such that \( h_i^A = h_i^* \).

Assume \( \tau_1 = \tau \cup \{\text{Mod}_1, h_1, ..., h_n\} \cup \text{Dom}^*, \) and \( \tau_2 = \tau \cup \{\text{Mod}_2, Pr, h_1, ..., h_n\} \cup \text{Dom}^* \). We assume for all structures \( A \) in \( M_1 \) and for all \( a \in \text{Dom}^* \), we have \( a^A = a \). Similarly, for all structures \( B \) in \( M_2 \) and for all \( a \in \text{Dom}^* \), we have \( a^B = a \).

Specification \( \psi_1 \) is defined as: \( \forall x \exists \overline{y}([h_1](\overline{y}) = x \land R_1(\overline{y})] \supset \text{Mod}_1(x)) \land ... \land \forall x \exists \overline{y}([h_n](\overline{y}) = x \land R_n(\overline{y})] \supset \text{Mod}_1(x)) \). Formula \( \psi_1 \) indicates that each expansion structure of \( \text{MX}_{\sigma, \psi} \) is represented by a subset of \( G \). More precisely, each expansion \( \tau_2 \)-structure \( A \) with domain \( \text{Dom}^* \) interprets \( \text{Mod}_1 \) such that \( \text{Mod}_1^A = \{g | g \in G \text{ and } \overline{a} \in R^A | \tau \text{ for } f(R(\overline{a})) = g\} \). We note that if \( A \) is an expansion of \( \text{MX}_{\sigma, \psi} \), then \( A|_{\tau} \) is an expansion of \( \text{MX}_{\sigma, \psi} \). Formula \( \psi_1 \) could be written in any language with a model-theoretic semantics.

Module \( M_2 \) is sequentially composed to \( M_1 \) with a loop from its output \( \text{Mod}_2 \) to its input \( \text{Mod}_1 \). Module \( M_2 \) solves a Model Expansion problem \( \text{MX}_{\sigma_2, \psi_2} \) where \( \psi_2 \) is a formula that is defined as \( \psi_2 = \psi \land \psi_p \) where \( \psi_p \) specifies preferred expansions of \( \text{MX}_{\sigma, \psi} \) such that \( \psi_p = \varphi_1 \land \varphi_2 \land \varphi_3 \) where

\[
\begin{align*}
\varphi_1 &= \forall x \exists \overline{y}([h_1](\overline{y}) = x \land R_1(\overline{y})] \supset \text{Mod}_2(x)) \land ... \land \forall x \exists \overline{y}([h_n](\overline{y}) = x \land R_n(\overline{y})] \supset \text{Mod}_2(x)); \\
\varphi_2 &= \forall x \forall y (\text{Mod}_1(x) \land \text{Mod}_2(y) \supset \text{Pr}(y, x)); \text{ and} \\
\varphi_3 &= \text{Pr}(g, g') \text{ if } R(\overline{t}) \supset P S(\sigma) \text{ where } f(R(\overline{t})) = g \text{ and } f(S(\sigma)) = g' \text{ for all pairs of ground atoms that have a preference relation with respect to preference expression } P.
\end{align*}
\]

Formula \( \varphi_1 \) indicates that if a ground atom \( R(\overline{a}) \) is assigned true by an expansion structure, then ground atom \( \text{Mod}_2(g) \), where \( f(R(\overline{a})) = g \), must also be assigned true by the same expansion structure. So, for a \( \tau_2 \)-structure \( B \), \( \text{Mod}_2^B = \{g | g \in G \text{ and } \overline{a} \in R^B | \tau \text{ for } f(R(\overline{a})) = g\} \). Formula \( \varphi_2 \) specifies that each ground atom that is assigned true by an expansion structure of \( \text{MX}_{\sigma_2, \psi_2} \) must be preferred to all ground atoms that are satisfied by expansion structures of \( \text{MX}_{\sigma_1, \psi_1} \). This reflects the Weak Pareto preference semantics. Also, \( \varphi_3 \) characterizes preferences among ground atoms expressed by preference expression \( P \). Similar to \( \psi_1 \), formula \( \psi_2 \) can be characterized in any language with a model-theoretic semantics.

We note that \( \varphi_2 \) varies for different preference semantics. For the Upper Bound Dominance semantics, \( \varphi_2 \) is defined as \( \varphi_2 = \forall x \exists y (\text{Mod}_1(x) \land \text{Mod}_2(y) \supset \text{Pr}(y, x)) \). The Element Dominance semantics is characterized as \( \varphi_2 = \exists x \exists y (\text{Mod}_1(x) \land \text{Mod}_2(y) \supset \text{Pr}(y, x) \land \neg \exists z (\text{Mod}_2(z) \land \text{Pr}(z, x))) \). Also, the Weak Element Dominance semantics is encoded as \( \varphi_2 = \exists x \exists y (\text{Mod}_1(x) \land \text{Mod}_2(y) \supset \text{Pr}(y, x)) \). Moreover, for the Strong Element Dominance semantics, \( \varphi_2 = \forall y \exists x (\text{Mod}_1(x) \land \text{Mod}_2(y) \supset \text{Pr}(y, x)) \).
5.6.2 Modular Systems as Operators

As mentioned in Chapter 4, two semantics for modular systems were introduced in [127]: 1) modular systems as sets of structures where the algebra of modular systems corresponds to classical logic and Codd’s algebra [42], and 2) fixed point semantics in which modular systems are viewed as operators (actions) that change the state of the world (the interpretation of some vocabulary symbols). The algebra of modular systems as operators (called the dynamic algebra of modular systems) corresponds to modal logic. It has been proved that these two semantics coincide in input and output of modules.

In order to characterize Algorithm 3 by a modular system, an operational view is crucial. It is convenient to show a transition over the states of the world (changing the interpretation of some vocabulary symbols) when modular systems are viewed as actions or operators. We introduce an algebraic expression in dynamic algebra to represent Algorithm 3 and the procedure in Figure 5.3. Then, we utilize a generic solver for modular systems to construct a preference solver.

The dynamic algebra of modular systems is also called the algebra of information flow when there is a flow of information from input to output. The operations in the algebra of modular systems in Chapter 4 are definable in dynamic algebra [129]. A non-deterministic action module with specified inputs and outputs vocabularies is a dynamic object that formalizes information propagation. In dynamic algebra, an action can be a logical characterization of computationally hard search problems such as scheduling, planning, etc. We first start with a brief review of dynamic algebra [129].

Let \( \text{Vars} = (X_1, X_2, \ldots) \) be a sequence of relational variables (second-order variables) with an associated arity. Assume \( \mathcal{M} = \{M_1, M_2, \ldots\} \) is a fixed vocabulary of atomic module symbols. Each module is associated with some variables in \( \text{Vars} \). Symbol \( \nuoc(M) = (X_i, \ldots, X_k) \) denotes second-order variables in module \( M \) that are visible from the outside world. We may use the symbol \( M(X_{i_1}, \ldots, X_{i_k}) \) to denote predicate variables that are associated with \( M \). To distinguish between input and output, the input variables are underlined. The syntax of dynamic algebra specifies a binary calculus on structures as follows:

\[
\alpha ::= id \mid M_i \mid Z_j \mid \alpha \cup \alpha \mid \alpha^{-} \mid \pi_\delta(\alpha) \mid \sigma_\Theta(\alpha) \mid \mu Z_j . \alpha. \tag{5.1}
\]

In this syntax, \( \alpha \) is an action (process) formula, \( M_i \in \mathcal{M} \) is an action module, and \( Z_j \) is an action variable. Operation \( \cup \) expresses a non-deterministic choice of actions, \( - \) is the negation, and
\(\sigma_\alpha\) is the selection of pairs structures satisfying condition \(\Theta\), which can be of the form \(X = Y\), \(Z \neq Y\), or a combination of both. Operation \(\pi\) is the projection and \(\mu Z_j.\alpha\) denotes the least fixed point where \(\alpha\) is assigned to action variable \(Z_j\).

Let us consider a fixed relational vocabulary \(\tau\) and domain \(\text{Dom}\). Let \(s : \text{Vars} \rightarrow \tau\) be a function that maps symbols in \(\tau\) to relational variables in \(\text{Vars}\). Symbol \([\cdot]^s\) specifies the semantics of the algebraic expressions in dynamic algebra. For an atomic module \(M\), \([M]^s\) is defined as:

\[
[M]^s = \{(A, B) \in U \times U | \exists C \in M (C|_{s(I(M))} = A|_{s(I(M))} \land C|_{s(O(M))} = B|_{s(O(M))} \land A|_{\tau|s(O(M))} = B|_{\tau|s(O(M))})\}
\]

where \(U\) is the set of all \(\tau\)-structures with domain \(\text{Dom}\) and \(s(I(M))\) and \(s(O(M))\) denote the vocabulary of the input and output of \(M\), respectively. In dynamic algebra, an atomic module is interpreted as a set of pairs of structures. For \((A, B) \in [M]^s\), \(A\) is an input and \(B\) is an output.

Module \(M\) as an operator only changes the output vocabulary \((s(O(M)))\) of each input structure. The interpretation of the rest of vocabulary \(\tau\) remains unchanged. This definition results in the following: \(A \in M\) if and only if \((A, A) \in [M]^s\) and, also, if \((B_1, B_2) \in [M]^s\), then \((B_2, B_2) \in [M]^s\).

For an algebraic expression \(\alpha\), the semantics of \(\alpha\) with notation \([\alpha]^s\) is defined inductively as follows:

\[
[id]^s := \{(A, A) | A \in U\},
\]

\[
[\alpha_1 \cup \alpha_2]^s := [\alpha_1]^s \cup [\alpha_2]^s,
\]

\[
[\alpha^-]^s := U \times U \setminus [\alpha]^s,
\]

\[
[\pi_1^δ(\alpha)]^s := \{(B_1, B_2) \in U \times U | \exists C((C, B_2) \in [\alpha]^s \land C|_{s(δ)} = B_1|_{s(δ)})\},
\]

\[
[\pi_2^δ(\alpha)]^s := \{(B_1, B_2) \in U \times U | \exists C((B_1, C) \in [\alpha]^s \land C|_{s(δ)} = B_2|_{s(δ)})\},
\]

\[
[\mu Z_j.\phi]^s := \bigcap\{R \subseteq U \times U | \phi|_{Z := R} \subseteq R\},
\]

\[
[\sigma^i_{L_1 = L_2}(\alpha)]^s := \{(B_1, B_2) \in [\alpha]^s | (s(L_1))^{B_1} = (s(L_2))^{B_1}\},
\]

\[
[\sigma^o_{L_1 = L_2}(\alpha)]^s := \{(B_1, B_2) \in [\alpha]^s | (s(L_1))^{B_2} = (s(L_2))^{B_2}\}.
\]

Here, \(\pi^i\) and \(\pi^o\) denote the projection onto a subset of the input and output vocabulary, respectively. More precisely, by operation \(\pi^i_δ\), the interpretation of vocabulary \(\delta \subseteq s(I(M))\), which is a subset of the input vocabulary, in the input structures (recall that \(\alpha\) defines a binary relation over structures in which the first element of each pair belongs to that binary relation is called the input and the second element is called the output of \(\alpha\)) remains the same as before and the interpretation of the rest of vocabulary of the input is projected out (cylinderified). Likewise, \(\pi^o_\delta\) projects out the interpretation of \(\tau \delta\), where \(\delta \subseteq s(O(M))\), in the output structures while keeps the interpretation of \(\delta\). Selection operator \(\sigma^i_{L_1 = L_2}(\alpha)\), where \(L_1, L_2 \in \text{Var}\), stands for the selection of all pairs of
structures where the interpretations of vocabulary symbols \( s(L_1) \) and \( s(L_2) \) in the input are the same. Also, \( \sigma^o \) and \( \sigma^{\sigma_0} \) denote the selection when variables \( L_1 \) and \( L_2 \) are in the output and input-output, respectively. Operation \( id \) is an action with no impact, such that the input is transferred to the output. The satisfaction relation is defined as \((A, B) \models_s \alpha \) if \((A, B) \in [\alpha]^s\). We say there is an \( \alpha \) transition from \( A \) to \( B \) if \((A, B) \models_s \alpha \).

Forward unary negation \( \lhd \) is a useful operation, which is definable from the above-mentioned primitive operations. Forward unary negation \( \lhd \alpha \) represents all (pairs of) structures from which there is no \( \alpha \) transition. This operation is used when we want to specify the continuation of a process until no more \( \alpha \) transition is possible. We define \( \lhd \alpha \) as: \( \lhd \alpha = (\pi_I(\alpha))^c \cap id \) and, semantically, \([\lhd \alpha]^s = \{(A, A)| \) there is no \( B \) such that\((A, B) \in [\alpha]^s\} \).

### 5.6.3 Characterizing Prioritized Model Expansion in Dynamic Algebra

Let \( \alpha_1 \) and \( \alpha_2 \) be actions in dynamic algebra such that \([\alpha_1]^s = [M_1]^s \) and \([\alpha_2]^s = [M_2]^s\). Figure 5.3, which demonstrates Algorithm 3, includes a While program, which is shown by a loop from the output of \( M_2 \) to its input. In the While loop, action \( \alpha_2 \) is executed for a non-deterministic and finite number of times until no more change is possible. The output of \( \alpha_2 \) is connected to its input. After reaching a fixed point, the final output of \( \alpha_2 \) is an optimal expansion of \( \Pi_{\sigma, \psi} \). Generally speaking, the While program is related to the notion of the least fixed point in logic. In what follows, for modelling the procedure in Figure 5.3, we shall use a specific form of the least fixed point in which there is no \( \alpha_2 \) transition to an output that is not an optimal expansion of \( \Pi_{\sigma, \psi} \).

Each action \( \alpha \) defines a binary relation, say \( R \), on \( U \). Algebraic expression \( \beta = \mu Z.(id \cup Z; \alpha) \) is the reflexive transitive closure of \( R \) such that \((A, A') \in [\beta]^s \) if, for some \( n > 0 \), there are \( B_1, \ldots, B_n \) in \( U \) such that \( A = B_1, A' = B_n \), and for all \( 0 < i < n + 1 \), \( (B_i, B_{i+1}) \in [\beta]^s \). The reflexive transitive closure of \( R \) corresponds to the Kleene star operation in Dynamic Logic [75].

In Figure 5.3, in each iteration, the execution of \( \alpha_2 \) generates a more preferable expansion structure of \( \Pi_{\sigma, \psi} \). When there is no more \( \alpha_2 \) transition, a fixed point is reached. So, if \((A, B) \in \mu Z.(id \cup Z; \alpha_2) \), then \( B \) is preferred to \( A \). However, it is not guaranteed that there is no \( C \) that is preferred to \( B \) and \((B, C) \in \mu Z.(id \cup Z; \alpha_2) \). To generate only the longest \( \alpha_2 \) transition, we consider a restricted form (a determinization) of Kleene star in the following way:

If there is a transition from input \( A \) to an output \( B \), there is no structure \( C \) where there is an \( \alpha_2 \) transition from \( A \) to \( C \) and from \( C \) to \( B \) or there is no structure \( B' \) such that there is an \( \alpha_2 \) transition from \( B \) to \( B' \). This restricted form of Kleene star is called max iterate operation with notation \( \alpha_2^\uparrow \) which is defined as \( \alpha_2^\uparrow = \mu Z.(\lhd \alpha_2 \cup \alpha_2; Z) \). For an input \( A \), algebraic expression \( \alpha_2^\uparrow \) represents the maximum possible iterations of \( \alpha_2 \) until for an output, say \( B \), no more execution of \( \alpha_2 \) is possible. In this case, \( B \) is an optimal expansion of \( \Pi_{\sigma, \psi} \).

As it can be observed from Figure 5.3, in each iteration that action \( \alpha_2 \) is executed, its output is connected to the input of \( \alpha_2 \) in the next iteration. To express the connection of the input and output of a module in dynamic algebra, we use an intermediary action module \( M_I \) that is defined as \([M_I]^s = \{(A, B)| A \in M_2, \text{ Mod}_I^A = \text{ Mod}_2^B, \text{ and } A|_{\tau_2 \setminus \{\text{Mod}_1}\}} = B|_{\tau_2 \setminus \{\text{Mod}_1}\}} \\} \). Action module \( M_I \)
connects the output of $M_2$, which is $\text{Mod}_2$, to the input of $M_2$, which is $\text{Mod}_1$, in the next iteration of the loop and the interpretation of $\tau_2 \setminus \{\text{Mod}_1\}$ remains unchanged. Putting all this together, the compound action $\lambda = \pi_{\text{Mod}_2}(\alpha_1; \mu Z, \mu \zeta \cup (\alpha_2; \alpha_I); Z)$ represents the procedure in Figure 5.3.

Figure 5.4 shows how compound action $\lambda$ is executed. Action $\alpha_1$ generates a structure with output vocabulary $\text{Mod}_1$ and the result is sent as a part of the input to action $\alpha_2$, which changes the interpretation of vocabulary symbol $\text{Mod}_2$. Action $\alpha_I$, where $[\alpha_I]^s = [M_I]^s$, is used to connect the output of $\alpha_2$ in one iteration to its input in the next iteration. This process continues for a non-deterministic finite number of times (say, $n$ times) until it reaches a fixed point where the output characterizes an optimal expansion of $\Pi_{\sigma, \psi}$.

**Theorem 11.** Assume algebraic expression $\lambda$ is defined as: $\lambda = \pi_{\text{Mod}_2}(\alpha_1; \mu Z, \mu \zeta \cup (\alpha_2; \alpha_I); Z)$. Let $\Pi_{\sigma, \psi}$ be a Prioritized Model Expansion problem that is solved by the procedure in Figure 5.3. If $(\mathcal{I}, \mathcal{B}) \in [\lambda]^s$, then, $\mathcal{B}$ is an optimal expansion of $\Pi_{\sigma, \psi}$ for input $\mathcal{I}$.

**Proof:** Formula $\psi_2$ in $M_2$ ensures that in each iteration, the output of $\alpha_2$ is preferred to the output of the previous iteration. Also, operation $(\alpha_2; \alpha_I)^\Delta = \mu Z, \mu \zeta \cup (\alpha_2; \alpha_I); Z$ returns the longest transition from the input. So, for an output $\mathcal{B}$, it is guaranteed that there is no possible output $\mathcal{C}$ where there is an $\alpha_2$ transition from $\mathcal{B}$ to $\mathcal{C}$. In other word, there is no output $\mathcal{C}$ that is preferred to $\mathcal{B}$. Therefore, the final output of $\lambda$ is an optimal expansion of $\Pi_{\sigma, \psi}$.

An immediate consequence is that a solver for $\lambda$ is a preference solver for $\Pi_{\sigma, \psi}$.

**Theorem 12.** A solver $s$ for compound action $\lambda$ is a preference solver for $\Pi_{\sigma, \psi} = (MX_{\sigma, \psi}, P)$.

**Proof:** Solver $s$ incrementally adds information to an initial partial structure until the partial structure is rejected because it is not an expansion of $MX_{\sigma, \psi}$ or the partial structure will not be augmented to a structure that is optimal based on $\psi_2$. Thus, $s$ is a preference solver for $MX_{\sigma, \psi}$.

A preference reason can be generated by solver $s$ for $M$ when a partial assignment is rejected because it falsifies formula $\varphi_2$ in $M_2$. Preference-guidances can also be provided in the similar way.
5.7 Conclusion

In this chapter, we studied a framework for solving Prioritized Model Expansion problems. We proposed a naive algorithm for finding the optimal expansions. We defined a declarative characterization of solvers for modular systems. After that, solvers for modular systems were generalized to preference solvers. We proposed a procedure to solve Prioritized Model Expansion problems and defined a modular system in dynamic algebra to model this procedure. Finally, we showed that a solver for a such modular system is a preference solver for Prioritized Model Expansion problems. Our proposal is related to the solvers in logic programming [32, 36, 113] in which an ASP solver returns a stable model and tries to find a more preferable stable model. However, here, we abstract away preference solvers and model them using solvers of modular systems.

One possibility for future work is to implement a solver for modular systems and, hence, for Prioritized Model Expansion problems. Declarative heuristics [32] can help the solver rank ground atoms and consider atoms with higher ranks sooner, which can impact the performance of the solver.
Chapter 6

Distance-Optimal Approximate Solutions in Compound Model Expansion Problems

Over-constrained problems, which do not have a solution that satisfy all the constraints, arise in many AI applications (e.g., planning, formal verification, robotics, etc.). We develop a preference-based model-theoretic approach to address over-constrained problems that arise when a number of computational problems are combined. We introduce compound Model Expansion problems, which are combinations of several Model Expansion problems, possibly specified in different languages, that may have no solutions. For such problems, users specify their preferences over the specifications (constraints). We compute the approximate solutions of a combination of Model Expansion problems, based on the Hamming distance measure.

6.1 Introduction

In Chapter 2, we characterized computational problems as Model Expansion that is the logical task of expanding an input structure (problem instance) to satisfy a specification $\psi$ in a certain language with a model-theoretic semantics. Here, we consider a more general case in which several Model Expansion problems are combined. A combination of Model Expansion problems may have no solutions. A problem is called over-constrained when it does not have a solution that satisfies all the specifications (constraints). Over-constrained problems arise in many AI applications, such as in planning [5, 82], constraint satisfaction problems [83, 98], requirement engineering for software development [90], etc. We study a real-world example of an over-constrained problem in mining engineering below.

Example 19. In mining engineering, equipment size selection is a vital task in each mining project due to its impact on the safety of a site and its workers, efficiency, marginal profits, etc. Computing the proper equipment size is an important topic of research in mining engineering [24]. As discussed in [112], a number of parameters affect the proper size of the equipment. These parameters are
divided into three categories: 1) Mine site characteristics, 2) Mining design properties, and 3) Equipment properties. The category of mine site contains a number of parameters describing a mine site based on the reality in the field, e.g., climate, skilled workers, water flow, rock characteristics, etc. The parameters in the category of mining design properties characterize different aspects of a mining project, such as capacity, roadways, geometry, etc. The properties of equipment describe criteria must be considered in designing a mining machine, such as safety, maintainability, gas emission, etc. The parameters in these categories may impact each other such that assigning a specific value for a parameter may impact the selection of a value for another parameter. The problem of equipment size selection comprises two sub-problems: the mechanical design of the equipment that includes computing a safety measure of the equipment based on a set of criteria (assume there is a safety measure threshold that should be considered by the equipment design) and a mine design, which determines the roadways and the capacity of a mine. Generally, there are a number of conditions that must be met for an accepted equipment size. For example, an equipment mechanical design, which determines the safety of the equipment, considers the climate of the project site. As another example, the roadways must be designed based on the water inflow and the maximum capacity of drilling. However, it might be impossible to satisfy all conditions simultaneously. For instance, certain roadways, constructed based on the solutions of the mine design problem, do not allow some types of equipment, despite their very high degree of safety and maintainability. Therefore, considering all factors cannot always lead to a solution for the problem of equipment size selection.

It is a common practice in engineering design to approximate the solutions based on the realities of a project site. For example, the safety of workers is a more important constraint compared to the maximum capacity of drilling. An accepted design is the closest one to satisfy more important constraints. Therefore, an accepted design of the equipment and roadways is as closest possible to allow moving safer equipment and then a larger capacity of drilling if possible.

This problem becomes more complicated when different groups of engineers are responsible for each sub-problem, possibly using different notations and standards. In the context of Knowledge Representation, this means that sub-problems are specified in different languages. It is necessary to aggregate the design of each sub-problem (i.e., mine design and equipment design), prioritize constraints and find a set of approximate solutions.

We tackle this problem by translating it into a compound Model Expansion problem, which is a combination of multiple Model Expansion problems, prioritizing the specifications, and finding approximate solutions. Each sub-problem can be specified in a different language with a model-theoretic semantics. Approximate solutions are solutions that are the closest to the expansion structures of Model Expansion problems with specifications in higher priorities.

In the previous chapters, we defined preferences as ordering relations on a set of ground atoms or on a set of partial models (partial structures). In this chapter, we consider preferences of a decision maker over the specifications of problems that are combined. We compute the proximity of two structures based on the Hamming distance measure [74]. For a compound Model Expansion
problem, which is a combination of multiple Model Expansion problems, without a solution, we find the Hamming distance optimal solutions of the problem. Our proposal has a tight relationship with model-based belief merging [87] in which a set of knowledge bases are merged. Each knowledge base is defined as a set of formulas in a certain language, e.g., propositional [86], first-order [71], ASP [49], etc. When knowledge comes from different sources, conflict may happen and a model for all knowledge bases may not be found. Our approach to solving compound Model Expansion problems without solutions can be viewed as a generalization of model-based merging. In our proposal, there is a clear separation of problem instance and problem specifications, where each problem specification can be written in a different language $\mathcal{L}$ with a model-theoretic semantics.

Finding approximate solutions, which can be viewed as an optimization task, is generally a harder task than solving a Model Expansion problem, similar to over-constrained CSP that is harder than usual CSPs [24]. We study the complexity of several problems related to compound Model Expansion problems without solutions.

This chapter is organized as follows: We first introduce the notion of a compound Model Expansion problem, which is a combination of a set of Model Expansion problems. Then, we define the notion of a distance between two structures and between a structure and the solutions of a Model Expansion problem. These distance measures are used to compute the solutions of compound Model Expansion problems. After that, we briefly discuss constructing quantitative preferences from qualitative preferences over the specifications of a compound problem, which is assigning a numerical weight to each specification. Furthermore, we study several computational problems related to compound Model Expansion problems and their complexity. Finally, we extend our proposal to compound Prioritized Model Expansion problems that may have no solution.
6.2 Compound Model Expansion

We introduce a more general form of Model Expansion, which is expanding a problem instance to satisfy a set of problem specifications \( \Psi = \{\psi_1, \ldots, \psi_n\} \). Each \( \psi_i \in \Psi \) can be written in a different language \( \mathcal{L}_i \). For the sake of simplicity, we consider all specifications are written in the same language.

**Definition 33. (Compound Model Expansion Problem MX\( _{\sigma,\Psi} \))**

*Given:* an input \( \sigma \)-structure \( I \) and a fixed set of specifications \( \Psi = \{\psi_1, \ldots, \psi_n\} \) written in a language \( \mathcal{L} \) where \( \text{vocab}(\Psi) = \tau \) and \( \sigma \subseteq \tau \).

*Find:* a \( \tau \)-structure \( B \) that expands \( I \) and for all \( i \in [1, n] \), \( B \models \psi_i \). The decision version is to decide whether there exists a \( \tau \)-structure \( B \) that expands \( I \) and for all \( i \in [1, n] \), \( B \models \psi_i \).

The solutions to a compound Model Expansion problem \( \text{MX}_{\sigma,\Psi} \) are the intersection of the solutions of \( n \) Model Expansion problems \( \text{MX}_{\sigma,\psi_1}, \ldots, \text{MX}_{\sigma,\psi_n} \). A \( \tau \)-structure \( B \) is an expansion of \( \text{MX}_{\sigma,\Psi^i} \) if it is an expansion of \( \text{MX}_{\sigma,\psi^i} \), for all \( i \in [1, n] \).

We call \( \text{MX}_{\sigma,\Psi} \) *inconsistent* when it has no expansion structure. We assume each Model Expansion problem \( \text{MX}_{\sigma,\psi_i} \), where \( 1 \leq i \leq n \), is consistent (i.e., has a solution).

**Example 20.** Let \( \mathcal{G} = (V, E^G) \) be a graph such that \( V = \{v_1, v_2, v_3, v_4\} \) is a set of vertices and \( E^G = \{(v_1, v_2), (v_2, v_1), (v_2, v_3), (v_3, v_2), (v_2, v_4), (v_2, v_1), (v_4, v_3), (v_1, v_4), (v_4, v_1)\} \). Let \( R, B \), and \( G \) be unary predicate symbols, which denote red, blue, and green colour. Assume formula \( \varphi \) is defined as \( \varphi = \forall x((R(x) \lor B(x) \lor G(x)) \land (\neg R(x) \lor \neg B(x)) \land (\neg R(x) \lor \neg G(x)) \land (\neg G(x) \lor \neg B(x))) \). Formula \( \varphi \) indicates that each node of a graph only has one colour.

Consider a more limited version of the graph three-colouring problem as a Model Expansion problem \( \text{MX}_{\{E\},\Psi} \) where \( \Psi = \{\psi_1, \psi_2, \psi_3\} \) and

\[
\psi_1 = \varphi \land \forall x \forall y (E(x, y) \rightarrow (\neg R(x) \lor \neg R(y)) \land (\neg B(x) \lor \neg B(y)) \land (\neg G(x) \lor \neg G(y))) \\
\psi_2 = \varphi \land \forall x \forall y (R(x) \land E(x, y) \rightarrow B(y)) \\
\psi_3 = \varphi \land B(v_1)
\]

Here, \( \text{vocab}(\Psi) = \{E, R, B, G, v_1\} \) where \( v_1 \) is a constant symbol. We assume that for each \( \tau \)-structure \( A \) that satisfies \( \psi_3 \), we have \( v_1^A = v_1 \). It is obvious that there does not exist any three-colouring for graph \( \mathcal{G} \). In fact, any graph that contains a cycle of an odd size (the number of edges in the cycle is odd) does not have a colouring satisfying \( \psi_1, \psi_2, \psi_3 \) simultaneously.

We employ the well-known Hamming distance metric [74] to measure the closeness of two structures. Hamming distance (originates from information technology and error correcting [94]) between two strings of the same length is the number of different characters at each position. In the context of Model Expansion, for two \( \tau \)-structures \( A \) and \( B \) with the same domain \( \text{Dom} \), the distance between \( A \) and \( B \), notation \( |A - B| \), is defined as \( |A - B| = \sum_{i=1}^{l} w(a_i) \) where for each
ground atom $a_i$, if $A \models a_i$ and $B \models a_i$ or $A \not\models a_i$ and $B \not\models a_i$, then $w(a_i) = 0$, and $w(a_i) = 1$ otherwise. Also, $l$ is the total number of possible ground atoms based on the size of $Dom$ and the size of vocabulary $\tau$, which is assumed to be fixed.

### 6.3 Distance-Optimal Solutions

We define the distance between a $\tau$-structure $A$ and the solutions of a Model Expansion problem $MX_{\sigma,\psi}$ to be the minimum (Hamming) distance between $A$ and each solution of $MX_{\sigma,\psi}$ as follows:\(^1\)

$$\text{Dist}(A, MX_{\sigma,\psi}) := \min_{B \text{ is an expansion of } MX_{\sigma,\psi}} |A - B|.$$  

(6.1)

Assume that specifications are weighted (as a quantitative way of modelling preferences). A function $\omega : \Psi \to \mathbb{Q}^+$ is called a weighting function. A compound Model Expansion problem with weighted specifications is called a weighted (ranked) compound Model Expansion problem:

**Notation 1. (Weighted Compound Model Expansion Problem)**

A weighted compound Model Expansion problem is denoted by a pair $\Omega_{\sigma,\psi} = (MX_{\sigma,\psi}, \omega)$ where $MX_{\sigma,\psi}$ is a compound Model Expansion problem and $\omega$ is a weighting function.

For a given $\tau$-structure $A$ with a domain $Dom$ and a weighted compound Model Expansion problem $\Omega_{\sigma,\psi} = (MX_{\sigma,\psi}, \omega)$ with an arbitrary input over domain $Dom$ where $\Psi = \{\psi_1, ..., \psi_n\}$ and $\tau = vocab(\Psi)$, let us define $\text{Dist}(A, \Omega_{\sigma,\psi})$ as

$$\text{Dist}(A, \Omega_{\sigma,\psi}) := \sum_{i=1}^{n} \omega(\psi_i) \cdot \text{Dist}(A, MX_{\sigma,\psi_i}).$$  

(6.2)

It can be observed from (6.2) that the more preferable a specification of a Model Expansion problem is, the more influential the specification is in computing $\text{Dist}(A, \Omega_{\sigma,\psi})$.

For a given vocabulary of symbols $\tau$, a domain of elements $Dom$, and a weighted compound Model Expansion problem $\Omega_{\sigma,\psi} = (MX_{\sigma,\psi}, \omega)$ with an input $I$ with domain $Dom$ where $\tau = vocab(\Psi)$, $\text{Dist}(A, \Omega_{\sigma,\psi})$ defines a preorder relation $\succeq$ among $\tau$-structures with domain $Dom$ expanding $I$ as follows:

**Definition 34.** Let $\Omega_{\sigma,\psi} = (MX_{\sigma,\psi}, \omega)$ be a weighted Model Expansion problem where $MX_{\sigma,\psi}$ is a compound Model Expansion problem with an arbitrary input $I$ with domain $Dom$, where $\omega$ is a weighting function. We define for all $\tau$-structures $A$ and $B$ with domain $Dom$, $A \triangleright B$ if $\text{Dist}(A, \Omega_{\sigma,\psi}) < \text{Dist}(B, \Omega_{\sigma,\psi})$. Also, $A \triangleright B$ if $\text{Dist}(A, \Omega_{\sigma,\psi}) = \text{Dist}(B, \Omega_{\sigma,\psi})$ or $\text{Dist}(A, \Omega_{\sigma,\psi}) < \text{Dist}(B, \Omega_{\sigma,\psi})$.

A distance-optimal approximate solution of a weighted compound Model Expansion problem is a structure that is maximal with respect to $\triangleright$.

\(^1\)We write “:=" to denote "is by definition"
Definition 35. (Weighted Compound Model Expansion Problem $\Omega_{\sigma, \Psi}$)

Given: a compound Model Expansion problem $MX_{\sigma, \Psi}$ with an input $\sigma$-structure $I$ and a weighting function $\omega$ over $\Psi$.

Find: a $\tau$-structure $A$ ($\tau = vocab(\Psi)$) such that there is no $\tau$-structure $B$ that expands $I$ and $B \triangleright A$.

A structure $A$ that is a solution to the weighted compound Model Expansion problem is called a distance-optimal approximate solution (or approximate solution) of $\Omega_{\sigma, \Psi}$. Note that $\Omega_{\sigma, \Psi}$ always has an approximate solution. The set of distance-optimal approximate solutions of $\Omega_{\sigma, \Psi}$ is denoted by $approx(\Omega_{\sigma, \Psi})$.

From Qualitative Models to Quantitative Models

We can characterize the importance of each specification of a problem by a partial order $\preceq$, similar to qualitative preference relations we defined in the previous chapters. Let $\prec$ be a partial order over $\Psi = \{\psi_1, \ldots, \psi_n\}$ where $\psi_i \prec \psi_j$ means that $\psi_j$ is preferred to $\psi_i$. For a compound Model Expansion problem $MX_{\sigma, \Psi}$ and a partial order $\prec$ over $\Psi = \{\psi_1, \ldots, \psi_n\}$, we define $A$ as an approximate expansion (solution) of $MX_{\sigma, \Psi}$ when $A$ is the closest to expansions of $MX_{\sigma, \psi_i}$ for $\psi_i$s that are more preferable with respect to $\prec$. A qualitative preference model can be translated into a quantitative model, which is called the linearization of a partial order [83], by a weighting function that assigns a weight to each specification such that order $\preceq$ is respected. A function $\omega_{\prec}: \Psi \to \mathbb{Q}^+$ is called a weighting function for $\preceq$ whenever for all $\psi_i$ and $\psi_j$, if $\psi_i \preceq \psi_j$, then we have $\omega_{\prec}(\psi_i) \geq \omega_{\prec}(\psi_j)$.

We may drop the reference to $\preceq$ in $\omega_{\prec}$ when it is clear from the context. Expression $\omega_{\prec}(\psi)$ is called the weight (rank) of $\psi$ in partial order $\preceq$.

As an example, a simple method to derive a weighting function from $\preceq$ is described as follows: Assume relation $\preceq$ is represented by a graph $G_{\Psi}$ where each node $v_i$ in $G_{\Psi}$ represents $\psi_i \in \Psi$. There is a directed edge between node $v_i$ and $v_j$ if $v_i \preceq v_j$. A path $\rho$ is a shortest path starting from a node $v_0$ with no incoming edge and ending at a node $v_d$ with no outgoing edge. The level of $v_0$ (notation $L(v_0)$) is defined as 1. The length of a node $v$ in $\rho$ is defined as $L_\rho(v) = \frac{\text{len}(\rho)}{\text{len}(\rho)-k}$, where $k$ is the number of nodes in $\rho$ from $v_0$ to $v$ and $\text{len}(\rho)$ is the length of $\rho$, which is the number of nodes appearing in $\rho$. The level of $v$ in $G_{\Psi}$ is defined as $L(v) = \max_j \{L_{\rho_j}(v)\}$ for any $\rho_j$ visiting $v$. It is clear that function

$$\omega(\psi) = \frac{1}{L(v_\psi)}$$

(6.3)

is a weighting function. We note that for a partial order $\preceq$ over a set $S$ and $s \in S$, computing $\omega_{\prec}(s)$, based on (6.5), requires, at most, finding all shortest paths starting from nodes with no incoming edges toward $s$, which is in polynomial time of the size of $S$. Henceforth, we assume that any weighting function $\omega$ for a partial order $\preceq$ over a set $S$ computes $\omega(s)$, where $s \in S$, in polynomial time of the size of $S$.
6.4 Query Evaluation for Weighted Compound Model Expansion Problems

Let $\Psi$ be specified in a logic $\mathcal{L}$. For a formula $\varphi$ in $\mathcal{L}$, we say $\text{approx}(\Omega_{\sigma,\Psi}) \models \varphi$ if there is a structure $\mathcal{A} \in \text{approx}(\Omega_{\sigma,\Psi})$ such that $\mathcal{A} \models \varphi$. Note that this formalization corresponds to the notion of brave reasoning. Also, cautious reasoning (i.e., for all structures $\mathcal{A} \in \text{approx}(\Omega_{\sigma,\Psi})$, whether $\mathcal{A} \models \varphi$) can be defined analogously. The problem of Approximate Support asks whether a distance-optimal approximate solution of a weighted compound Model Expansion problem satisfies a property $\varphi$.

**Definition 36. (Approximate Support Problem)**

| Given: $\Omega_{\sigma,\Psi} = (MX_{\sigma,\Psi}, \omega)$ where $\Psi$ is a finite set of specifications in a logic $\mathcal{L}$ and $\text{vocab}(\Psi) = \tau$, an arbitrary input $\mathcal{I}$ with a domain $\text{Dom}$, and a formula $\varphi$ in $\mathcal{L}$, |
| Question: does $\text{approx}(\Omega_{\sigma,\Psi}) \models \varphi$ hold? |

Below, we present an algorithm, which is similar to the idea of query evaluation in belief merging [86], for solving the Approximate Support problem. Before proceeding to the algorithm, we present a few complexity results. First, we show that computing the distance between two structures is solvable in polynomial time in the size of the domain of the structures.

**Definition 37. (Distance Problem)**

| Given: $\tau$-structures $\mathcal{A}$ and $\mathcal{B}$ with a domain $\text{Dom}$ and a constant $C$, |
| Question: is $|\mathcal{A} - \mathcal{B}| < C$? |

**Proposition 6.** The data complexity of the Distance problem is in PTIME.

**Proof:** The total number of ground atoms is in $O(m \cdot |\text{Dom}|^k)$ where $k$ is the maximum arity of predicates in $\tau$ and $m$ is the number of elements in $\tau$. Also, computing $w(a_i)$ takes a constant time. Therefore, the total number of operations to compute the Hamming distance between $\mathcal{A}$ and $\mathcal{B}$ is in $O(|\text{Dom}|^k)$.

**Notation 5.** Let $\Omega_{\sigma,\Psi} = (MX_{\sigma,\Psi}, \omega)$ be a weighted compound Model Expansion problem with input $\mathcal{I}$ with a domain $\text{Dom}$. Let $\text{vocab}(\Psi) = \tau$. Notation $\min_{\mathcal{A}} \text{Dis}(\mathcal{A}, \Omega_{\sigma,\Psi})$ denotes the minimum value of $\text{Dis}(\mathcal{A}, \Omega_{\sigma,\Psi})$ for any $\tau$-structure $\mathcal{A}$ that expands $\mathcal{I}$.

In order to determine whether there is a structure $\mathcal{A}$ in $\text{approx}(\Omega_{\sigma,\Psi})$ that satisfies a formula $\varphi$, we first solve the problem of deciding whether $\min_{\mathcal{A}} \text{Dis}(\mathcal{A}, \Omega_{\sigma,\Psi})$ is less than a constant $C$.

**Definition 38. (Upper Bound Distance Problem)**

| Given: a weighted compound Model Expansion problem $\Omega_{\sigma,\Psi} = (MX_{\sigma,\Psi}, \omega)$ where $\Psi = \{\psi_1, \ldots, \psi_n\}$ is in $\mathcal{L}$ and $\tau = \text{vocab}(\Psi)$, an input $\mathcal{I}$ with a domain $\text{Dom}$, and a constant $C$, |
| Question: is $\min_{\mathcal{A}} \text{Dis}(\mathcal{A}, \Omega_{\sigma,\Psi}) < C$? |
Proposition 7. Let model checking in $\mathcal{L}$ be in a complexity class $X$. The Upper Bound Distance problem is in $\text{NP}^X$.

Proof: First, non-deterministically guess a $\tau$-structure $B$ with domain $\text{Dom}$ and $\tau$-structures $B_1, ..., B_n$ with domain $\text{Dom}$. Second, by means of an oracle in $X$ check whether $B$ expands $\mathcal{I}$ and whether $B_i$ is an expansion of $\mathcal{M}_{\sigma, \psi_i}$ for $i \in [1, n]$. Third, compute $|B - B_i|$ for $i \in [1, n]$ in polynomial time, based on Proposition 6. Finally, determine whether $\sum_{i=1}^n \omega(\psi_i) \cdot \text{Dist}(B, \mathcal{M}_{\sigma, \psi_i}) < C$ in constant time.

The complexity of computing $\min_A \text{Dis}(A, \Omega_{\sigma, \psi})$ is as follows.

Proposition 8. Let $\Omega_{\sigma, \psi}$ be a weighted compound Model Expansion problem with an input $\mathcal{I}$ with domain $\text{Dom}$. Let model checking of $\Psi$ be in a complexity class $X$. The value of $\min_A \text{Dis}(A, \Omega_{\sigma, \psi})$ can be computed by polynomial number of calls to an $\text{NP}^X$ oracle.

Proof: For every $\tau$-structure $C$ that expands $\mathcal{I}$, $2^p(|\text{Dom}|)$ is an upper bound for $\text{Dist}(C, \Omega_{\sigma, \psi})$ where $|\text{Dom}|$ is the size of $\text{Dom}$ and $p$ is a polynomial similar to [86]. As a result, $\min_A \text{Dis}(A, \Omega_{\sigma, \psi})$ is also bounded by $2^p(|\text{Dom}|)$.

To compute $\min_A \text{Dis}(A, \Omega_{\sigma, \psi})$, we perform a binary search on the sorted arrays of natural numbers $1, 2, 3, ..., 2^p(|\text{Dom}|)$ (assume all weights are natural numbers and, therefore, $\min A \text{Dis}(A, \Omega_{\sigma, \psi})$ is also a natural number). The maximum number of steps in the binary search is $\log(2^p(|\text{Dom}|))$, which is equal to $p(|\text{Dom}|)$. Assume that we obtain number $C_i$ at $i$th step of the binary search. We check whether $\min A \text{Dis}(A, \Omega_{\sigma, \psi}) < C_i$. This is an instance of the Upper Bound Distance problem, which is in $\text{NP}^X$ based Proposition 7. Therefore, we solve the Upper Bound Distance problem for at most $p(|\text{Dom}|)$ number of times until we obtain the value of $\min A \text{Dis}(A, \Omega_{\sigma, \psi})$. Consequently, $\min A \text{Dis}(A, \Omega_{\sigma, \psi})$ can be computed by polynomial number of calls to an $\text{NP}^X$ oracle.

We define the problem of Constant Distance Support as follows.

Definition 39. (Constant Distance Support Problem)

Given: a weighted compound Model Expansion problem $\Omega_{\sigma, \psi}$ with an input structure $\mathcal{I}$ with domain $\text{Dom}$ where $\Psi = \{\psi_1, ..., \psi_n\}$ are defined in a logic $\mathcal{L}$, a formula $\varphi$ in $\mathcal{L}$, and a constant $C$.

Question: is there a $\tau$-structure $B$ such that $B$ expands $\mathcal{I}$, $B \models \varphi$, and $\text{Dis}(B, \Omega_{\sigma, \psi}) = C$?

Proposition 9. Let model checking in $\mathcal{L}$ in the Constant Distance Support problem be in a complexity class $X$. The problem of Constant Distance Support is in $\text{NP}^X$.

Proof: First, by using a non-deterministic Turing machine, non-deterministically guess a $\tau$-structure $B$ with domain $\text{Dom}$ and $\tau$-structures $B_1, ..., B_n$ with domain $\text{Dom}$. Then, check whether $B$ expands $\mathcal{I}$, which can be done in polynomial time. After that, by means of an oracle in $X$ check
whether $B \models \varphi$ and check whether $B_i$ is an expansion structure of $M_{X,\psi_i}$ for $i \in [1,n]$. Next, compute $|B - B_i|$ for $i \in [1,n]$ in polynomial time, based on Proposition 6. Finally, in constant time, compute $\text{Dist}(B, \Omega_{\sigma,\psi})$ which is equal to $\sum_{i=1}^{n} \omega(\psi_i) \cdot |B - B_i|$ and check whether $\text{Dist}(B, \Omega_{\sigma,\psi}) = C$. Since we use a non-deterministic polynomial Turing machine with an oracle in $X$, the Constant Distance problem is in NP.

We now analyze the upper bound on the complexity of the problem of Approximate Support.

**Theorem 13.** Let model checking in logic $\mathcal{L}$ in the Approximate Support problem be in a complexity class $X$. The Approximate Support problem is in $P^{NP^X}$.

**Proof:** We solve the Approximate Support problem by solving two sub-problems subsequently. In the first part, we compute $\min_A \text{Dis}(A, \Omega_{\sigma,\psi})$ by polynomial number of calls to an NP oracle, based on Proposition 8. We assume that $m = \min_A \text{Dis}(A, \Omega_{\sigma,\psi})$. It is worth noting that the value of $m$ is unique and computed only for one time.

In the second part, we solve an instance of the Constant Distance Support problem as follows. First, by using a non-deterministic Turing machine, non-deterministically guess a $\tau$-structure $B$ with domain $\text{Dom}$, and $\tau$-structures $B_1$, ..., $B_n$ with domain $\text{Dom}$. Then, check whether $B$ expands $I$, which can be done in polynomial time. Next, By means of an $X$ oracle, check whether $B \models \varphi$ and check whether $B_i$ is an expansion structure of $M_{X,\psi_i}$ for $i \in [1,n]$. After that, compute $|B - B_i|$ for $i \in [1,n]$, in polynomial time, based on Proposition 6. Finally, in constant time, compute $\text{Dist}(B, \Omega_{\sigma,\psi})$ which is equal to $\sum_{i=1}^{n} \omega(\psi_i) \cdot |B - B_i|$ and check whether $\text{Dist}(B, \Omega_{\sigma,\psi}) = m$. Based on Proposition 9, the Constant Distance Support problem can be solved by one call to an NP oracle.

Hence, we can solve the Approximate Support problem by polynomial number of calls to an NP oracle for the sub-problem in the first part, and one more call to an NP oracle for the sub-problem in the second part. Therefore, the Approximate Support problem is in $P^{NP^X}$.

**Theorem 14.** Let a compound Model Expansion problem $M_{X,\psi}$ be in the complexity class $\Sigma_{k+1}^P$. The Approximate Support problem for $M_{X,\psi}$ is in $\Delta_{k+1}^P$.

**Proof:** Since the problem is in $\Sigma_k^P$, it has a Model Expansion specification $\phi$ in second-order logic of the form $Q_1, ..., Q_k \phi'$ with $k-1$ number of alternations of second-order quantifiers such that $Q_1$ is $\forall$, $Q_2$ is $\exists$, and $Q_{k-1}$ is $\forall$ if $k$ is even and $Q_{k-1}$ is $\exists$ otherwise. Consequently, the model checking of $\phi$ is in $\Pi_k^P$. Thus, based on Proposition 3, the Approximate Support problem is in $P^{NP^P_{k-1}}$ that is equal to $P^{\Sigma_k^P}$ and $\Delta_{k+1}^P$.

We remark that each specification (constraint) $\psi_i \in \Psi$ can be written in a different language (e.g., first-order, QBF, ASP, etc.).

Algorithm 5 solves the Approximate Support problem for a weighted compound Model Expansion problem $\Omega_{\sigma,\psi} = (M_{X,\psi}, \omega)$ with an input structure $I$ and a formula (query) $\varphi$. 82
Algorithm 5 Approximate Support Problem

procedure \( F(\Omega_{\sigma,\psi}, I, \varphi) \)

\[
\begin{align*}
&\text{set } m = \min_A \text{Dis}(A, \Omega_{\sigma,\psi}) \\
&\text{while TRUE do} \\
&\quad \text{guess a structure } A \text{ that expands } I \\
&\quad \text{guess structures } A_1, \ldots, A_n \text{ that are expansion structures of } MX_{\sigma,\psi_1}, \ldots, MX_{\sigma,\psi_n}, \text{ respectively} \\
&\quad \text{if } A| = \varphi \text{ then} \\
&\quad \quad \text{if } m = \text{Dist}(A, \Omega_{\sigma,\psi}) \text{ then} \\
&\quad \quad \quad \text{return } A \\
&\quad \text{end if} \\
&\quad \text{end if} \\
&\text{end while} \\
&\text{end procedure}
\]

Example 21. We solve the problem of equipment size selection by translating it into a problem of finding an approximate solution of a weighted compound Model Expansion problem. Let \( \sigma = \{ \text{rock-characteristics, climate, water-inflow, skilled-worker} \} \) be a vocabulary of relational symbols. Let \( I \) be an input \( \sigma \)-structure that specifies the properties of the mine site. The goal is to solve two sub-problems with input \( I \): mine design and mechanical design. Assume \( MX_{\sigma,\psi_1} \) is a Model Expansion problem that solves the mine design problem. Suppose \( \psi_1 \) is a set of constraints (e.g., written in ASP or first-order logic) that must be satisfied for a mine design to maximize the capacity of drilling. Let \( B_1 \) be a solution to \( MX_{\sigma,\psi_1} \) that specifies the capacity of drilling. Assume \( MX_{\sigma,\psi_2} \) is a mechanical design problem that maximizes the safety of the workers. Let \( \psi_2 \) be a set of constraints (e.g., written in ASP or first-order logic). A solution \( B_2 \) to \( MX_{\sigma,\psi_2} \) characterizes a mechanical design based on the input structure. Assume \( \omega(\psi_1) = 1/3 \) and \( \omega(\psi_2) = 2/3 \) that indicates the safety is twice as important as the maximum capacity. Assume there is a negative correlation (i.e., by increasing the safety, the maximum of capacity of drilling declines) between safety and the maximum drilling. To achieve higher safety we need to compromise on the maximum drilling. To solve this problem, we consider \( \Omega_{\sigma,\psi} = (MX_{\sigma,\psi}, \omega) \) where \( \Psi = \{ \psi_1, \psi_2 \} \). Let \( B \) be an approximate solution of \( \Omega_{\sigma,\psi} \). There is no \( C \) such that \( (1/3 \text{Dist}(C, MX_{\sigma,\psi_1}) + 2/3 \text{Dist}(C, MX_{\sigma,\psi_2})) < (1/3 \text{Dist}(B, MX_{\sigma,\psi_1}) + 2/3 \text{Dist}(B, MX_{\sigma,\psi_2})) \). Since the complexity of model checking is in \( P \) in first-order logic and ASP, based on Theorem 14, the Approximate Support problem is in \( \Delta_1^P \).

6.5 Distance-Optimal Solutions in Prioritized Model Expansion Problems

Recall from Chapter 3 that a Prioritized Model Expansion problem is denoted by a pair \( \Pi_{\sigma,\psi} = (MX_{\sigma,\psi}, P) \) where \( MX_{\sigma,\psi} \) is a Model Expansion problem and \( P = (S_\tau, \sqsupseteq_P) \) is a preference expression where \( \sqsupseteq_P \) is a preorder over the set of all ground atoms of vocabulary \( \tau = \text{vocab}(\psi) \) and a do-
main $Dom$. For two $\tau$-structures $A$ and $B$ with domain $Dom$, we define the notion of the weighted distance based on a Preference expression $P = (S, \Sigma)$ as $|A - B|_P := \sum_{a_i \in S} \omega_{\Sigma}(a_i) \cdot w(a_i)$ where $\omega_{\Sigma}$ is a weighting function for $\Sigma$ and $w(a_i)$ is 1 if $a_i$ is satisfied by both $A$ and $B$ or is not satisfied by neither $A$ nor $B$ and $w(a_i) = 0$ otherwise. The intuition behind the weighted distance is to penalize any mismatch (i.e., a ground atom is satisfied by one structure and is not satisfied by another structure) between two structures by greater weights for more preferable atoms. We define $\text{Dist}(A, \Pi_{\sigma, \psi})$ where $A$ is a $\tau$-structure with domain $Dom$ as

$$\text{Dist}(A, \Pi_{\sigma, \psi}) := \min_{B \text{ is an optimal expansion}} |A - B|_P. \quad (6.4)$$

A weighted compound Prioritized Model Expansion problem is denoted by a pair $\Theta_{\sigma, \psi} = (\Pi_{\sigma, \psi}, \omega)$ where $\Pi_{\sigma, \psi}$ is called compound prioritized Model Expansion problem and $\omega$ is a weighting function over $\Psi$. We define $\text{Dist}(A, \Theta_{\sigma, \psi})$ as

$$\text{Dist}(A, \Theta_{\sigma, \psi}) := \sum_{\psi_j \in \Psi} \omega(\psi_j) \cdot \text{Dist}(A, \Pi_{\sigma, \psi_j}). \quad (6.5)$$

In (6.5), both the importance of each ground atom and the weight of each problem specification are taken into account.

For a weighted compound Prioritized Model Expansion problem $\Theta_{\sigma, \psi} = (\Pi_{\sigma, \psi}, \omega)$ with an input $I$ with a domain $Dom$ where $\tau = vocab(\Psi)$, assume $m$ is the minimum value of $\text{Dist}(A, \Theta_{\sigma, \psi})$ for all $\tau$-structures $A$ with domain $Dom$. A $\tau$-structure $B$ is called a distance-optimal approximate solution of $\Theta_{\sigma, \psi}$ if $m = \text{Dist}(B, \Theta_{\sigma, \psi})$.

**Example 22. (Continuing Example 20)**

Let $\Omega_{\sigma, \psi} = (\Pi_{\sigma, \psi}, \omega)$ be a weighted compound Prioritized Model Expansion problem where $\Pi_{\sigma, \psi}$ is a prioritized graph-three colouring problem where $\Psi = \{\psi_1, \psi_2, \psi_3\}$, and $P$ is a preference expression such that $R(v_2) \supseteq P R(v_3) \supseteq P R(v_4)$ and $G(v_3) \supseteq P B(v_3)$. Also, assume that $\psi_1 > \psi_2 > \psi_3$. We use (6.3) to define $\omega_1$ and $\omega_\geq$. There are 6 possible three-colourings satisfying specification $\psi_1$ for this graph as follows:

- $A_1$: $R^{A_1} = \{v_1, v_3\}$, $B^{A_1} = \{v_4\}$, $C^{A_1} = \{v_2\}$,
- $A_2$: $R^{A_2} = \{v_1, v_3\}$, $B^{A_2} = \{v_2\}$, $C^{A_2} = \{v_4\}$,
- $A_3$: $R^{A_3} = \{v_2\}$, $B^{A_3} = \{v_1, v_3\}$, $C^{A_3} = \{v_4\}$,
- $A_4$: $R^{A_4} = \{v_4\}$, $B^{A_4} = \{v_1, v_3\}$, $C^{A_4} = \{v_2\}$,
- $A_5$: $R^{A_5} = \{v_1\}$, $B^{A_5} = \{v_2\}$, $C^{A_5} = \{v_1, v_3\}$, and
- $A_6$: $R^{A_6} = \{v_2\}$, $B^{A_6} = \{v_4\}$, $C^{A_6} = \{v_1, v_3\}$.

$A_6$ is the only optimal three-colouring of $G$ that satisfies $\psi_1$. Also, there are 24 possible ways of colouring $G$ that satisfy $\psi_2$ and 27 ways of colouring $G$ that satisfy $\psi_3$. Consider Prioritized Model Expansion problems $\Pi_{\sigma, \psi_1} = (\text{MX}_{\sigma_1, \psi}, P)$, $\Pi_{\sigma, \psi_2} = (\text{MX}_{\sigma, \psi_2}, P)$, and $\Pi_{\sigma, \psi_3} = (\text{MX}_{\sigma, \psi_3}, P)$. The distance between $A_6$ and $\Omega_{\sigma, \psi}$ is computed as:
\[ \text{Dist}(A_6, \Omega_{\sigma, \psi}) = \text{Dist}(A_6, \Pi_{\sigma, \psi_1}) + 2/3 \text{Dist}(A_6, \Pi_{\sigma, \psi_2}) + 1/3 \text{Dist}(A_6, \Pi_{\sigma, \psi_3}) = 0 + 4/3 + 2/3 = 2, \] which is the minimum distance. Therefore, \( A_6 \) is a distance-optimal approximate solution of \( \Pi_{\sigma, \psi} \).

Now, assume that a preference over the specifications is defined as \( \psi_3 \succ \psi_2 \succ \psi_1 \). A Structure \( B \) where \( R^B = \{v_1\}, B^B = \{v_2, v_4\}, \) and \( G^B = \{v_3\} \) is an approximate solution of \( \Omega_{\sigma, \psi} \) where
\[ \text{Dist}(B, \Pi_{\sigma, \psi_3}) + 2/3 \text{Dist}(B, \Pi_{\sigma, \psi_2}) + 1/3 \text{Dist}(B, \Pi_{\sigma, \psi_1}) = 0 + 0 + 4/3 = 4/3. \]

### 6.6 Discussion

In this chapter, we studied approximating the solutions of compound Model Expansion problems that do not have any solutions by prioritizing the specifications and finding structures with the least distance from those structures that expand the input structure and satisfy more important specifications. Our proposal can be seen as a generalization of \( \text{DA}^2 \) model-based merging operators using Hamming distance [86] in the sense that each Model Expansion problem \( MX_{\sigma, \psi_i} \) characterizes a knowledge base that is not necessarily propositional. \( \text{DA}^2 \) is a parameterized model-based operator with a distance function and two aggregation functions. One aggregation function is employed to select consistent information from each knowledge base, and another aggregation function aggregates information from different knowledge bases. There are two major approaches in merging (possibly conflicting) knowledge bases: model-based [91], which is focused on selecting the closet models of different knowledge bases, and formula-based, such as [8], which, roughly speaking, selects a subset of consistent formulas. The formula-based approach is mainly dependent on the syntax of the knowledge bases, and some information may get lost because only a subset of formulas are selected.

Belief merging and belief revision [67] are closely related. Belief merging can be cast as an extension of belief revision in which the knowledge base is updated by new information. The priority is given to more recent information. However, in belief merging, the new information is not necessarily accepted. Belief merging, in early stages, has been mainly focused on propositional knowledge bases [87, 51]. However, it is not merely limited to propositional logic, and there are proposals for merging first-order knowledge bases [71], merging logic programs [49], and argumentation theories.
[43], etc. In our proposed approach, we find solutions of a combination of several problems. Our proposal can be viewed as merging several knowledge bases in different logics.

Our work can be related to max CSP [98] and max-SAT [13] where each constraint (or a clause) is defined as a formula $\psi$ and all weights are considered to be equal.

6.7 Conclusion

We presented a framework to characterize a combination of problems without solution (over-constrained) as compound Model Expansion problems. By giving priority to some specifications of a combined problem, we proposed a model-theoretic approach to compute approximate solutions. An approximate solution is in the least distance to structures that expand the input and satisfy specifications in a higher priority. We proposed an algorithm to solve over-constrained compound Model Expansion problems. We also established a relationship between our proposal and model-based belief merging. In future work, we plan to use the operations in the algebra of modular systems to combine Model Expansion problems in different ways. We will study the impact of applying the operations of modular systems to the approximate solutions of compound Model Expansion problems with no solutions.
Chapter 7

Literature Review and Discussion

In this chapter, we review some of the most influential approaches to handle preferences. We categorize these approaches based on several criteria and compare them with our proposed approach. We begin with a brief description of qualitative and quantitative preference models. Then, we review some of the major preference-based approaches in nonmonotonic frameworks. Next, we take a closer look at conditional preferences with a graphical representation, such as CP-nets. After that, we review the role of preferences in argumentation theory and discuss preference-based planning. Finally, we briefly review preference reasoning with lexicographic models.

7.1 Qualitative and Quantitative Approaches

Since the very beginning of the research on handling preferences, two distinct approaches to model preferences emerged: Qualitative models in which preferences of a user or a decision maker are expressed by an ordering (total or partial) on values of some variables [120, 19] and quantitative approaches [22, 117], which specify preferences by assigning numerical weights to values of variables. It is not always practical to have a value function that weights different choices, specifically, for a large number of variables. Every quantitative model can be viewed as a special case of a qualitative model in which all choices are totally ordered based on the corresponding weights. In our proposed frameworks for Prioritized Model Expansion and preference-based modular systems, we used qualitative methods to represent preferences of a decision maker. In solving approximate solutions of compound Model Expansion problems, we constructed quantitative preference models from qualitative preferences over specifications of a compound problem.

7.2 Preferences in Nonmonotonic Reasoning

Nonmonotonic frameworks are defined based on commonsense reasoning such that adding new information may refute a consequence that has been already derived. One of the motivation of introducing nonmonotonic reasoning was reasoning with defaults or rules with exceptions. For example, unlike universal quantifiers in first-order logic, rules may have exceptions, such as $\forall x (bird(x) \Rightarrow$
fly(x)), which with exception becomes: ∀x(bird(x) ∧ ¬abnormal(x) ⇒ fly(x)). Nonmonotonic frameworks have been employed for various applications, such as to address the frame problem in situation calculus, belief revision, and database query languages [131].

Representing, reasoning with, and reasoning about preferences in nonmonotonic frameworks have been widely studied [48]. The primary focus of nonmonotonic reasoning was on default logic [107]. However, since early 2000s, Answer Set Programming (ASP) [35] has received much more attention for nonmonotonic approaches and declarative problem solving due to the development of significantly effective solvers and simple problem encoding.

We categorize preference handling approaches in nonmonotonic frameworks into two distinct categories: First, a preference relation is defined as an ordering on some ground atoms. Syntactically, each preference relation is represented as a specific form of ASP or default rules. The set of preference rules are aggregated based on a certain aggregation semantics. For example, in [36], answer sets are compared with respect to the level of satisfaction of each preference rule in each answer set. Second, preferences are represented as ordering relations on generating ASP or default rules. In this approach, more preferable rules are applied before less preferable ones, e.g., [34, 47].

We take a closer look at these approaches in the following:

7.2.1 Preferences over Atoms

In Chapter 3 and Chapter 4, we studied the relation between the Answer Set Optimization (ASO) [36] framework, Prioritized Model Expansion, and preference-based modular systems. The ASO framework is a distinguished example of preferences over atoms. There are a number of proposals in the category of preferences over atoms that are closely related to ASO. For example, in [29], which is a special case of ASO, preferences are represented as disjunctive orders in the head of a program’s rules. The authors of [29] incorporated an extension to propositional logic, which is called Qualitative Choice Logic (QCL), into logic programming to represent context dependent preferences. QCL, which expresses priorities in propositional logic, has all propositional logic connectives plus a new connective × that is called ordered disjunction. For two literals l₁ and l₂, expression l₁ × l₂ means that one of l₁ or l₂ must be true. It is preferred that l₁ is true. If this is not the case, then l₂ is true. Ordered disjunction is applied in the head of logic programs. The new logic program is called a logic program with ordered disjunction (LPOD). The semantics of LPODs is different than the semantics of disjunctive logic programs proposed in [68] because disjunctive programs generate minimal models, where it may not be the case in LPODs. The authors of [29] argued that by considering only minimal models, some more preferable models in a program may be missed. For example, assume a program with two rules r₁ = C₁ × C₂ × C₃ ← and r₂ = C₂ × C₄ ← has a minimal model X₁ = {C₂} that satisfies r₁ and r₂. However, X₂ = {C₁, C₂} satisfies both rules and has the most preferred literal in the head of each rule, which, according to [29], makes X₂ the most preferred model. Since X₁ ⊆ X₂, the model minimality is violated in ordered disjunction. To address this issue, a particular semantics was proposed for LPODs in [29] such that an LPOD program consists of a set of rules of the form r = C₁ × ... × Cₙ ← A₁, ..., Aₘ, not B₁, ..., not Bₚ,
is converted to a set of split programs. In each split program, the ordered disjunctions are removed (rule \( r \) is replaced by \( C_k \leftarrow \text{body}(r), \neg C_1, ..., \neg C_{k-1}, \) for some \( k \)). A set \( X \) is an answer set of an LPOD program \( P \) if it is a consistent answer set of some split programs of \( P \). The most preferred answer set is the one with a higher satisfaction of each rule, similar to [36].

One of the main differences between ASO programs and LPODs is that preferences and generating programs are completely separated in ASO programs, while in an LPOD program, preferences are represented within a generating program. Similarly, in Prioritized Model Expansion, preferences are separated from the problem’s axiomatizations. Likewise, in preference-based modular systems, a problem is characterized as a set of structures and preferences are defined independently.

Preference description language (PDL) [30] is an extension of ASO for modeling quantitative and more complex preference expressions. In PDL, each preference rule is associated with a certain penalty, which is a basic preference expression. Also, PDL allows to combine basic preference expressions to express more complex preferences statements. PDL is a combination of a qualitative approach (preference rules are qualitative models that are associated with penalties) and a quantitative approach (preference rules are combined based on some quantitative criteria). The complexity of deciding whether there is a preferred answer set that satisfies a set of atoms remains the same as in ASO.

Moreover, in Chapter 3, we studied the relation between Prioritized Model Expansion and PLP [113], which is one of the primary approaches in defining preferences over atoms in logic programming. The proposal in [73] is another example of preferences over atoms, which is a generalization of PLP by employing the syntax of preference rules in ASO [36].

7.2.2 Preferences over Rules

A large number of earlier preference handling methods in nonmonotonic frameworks have been proposed based on a basic principle: a program rule or a default rule in a higher priority is applied before a rule in a lower priority. We review some well-known works in this family of frameworks in default logic and logic programming.

Default Logic and Preferences over defaults

A default rule in the standard default logic [107] is defined as: \( q : \frac{\alpha(x) ; \beta_1(x), ..., \beta_n(x)}{\gamma(x)} \) where \( \alpha(x) \) is called the prerequisite of \( q \) (notation \( \text{Prereq}(q) \)), set \( \{\beta_1, ..., \beta_n\} \) is the justification of \( q \) denoted by \( \text{Justif}(q) \), and \( \gamma \) is the consequent of \( q \) with notation \( \text{Conseq}(q) \). A default is called categorical or prerequisite free when it has no prerequisite. When the justification and conclusion of a default are equivalent, it is called normal. A default \( q \) that \( \text{Prereq}(q) \), \( \text{Justif}(q) \), and \( \text{Conseq}(q) \) are first-order sentences is called closed. A default theory is a pair \((Q, W)\) where \( Q \) is a set of defaults and \( W \) is a set of first-order sentences. A default theory may generate a set or multiple sets of beliefs that are called extensions. Reiter [107] characterized extensions of a default theory in a non-constructive
way as follows: Assume \((Q, W)\) is a default theory and \(E\) is a set of first-order formulas. Let \(E_0 = W\) and \(Th(E)\) be (classic) logical closure of \(E\). If 
\[ E_{i+1} = Th(E_i) \cup \{ \gamma | \Box_{i+1}^{\alpha} ... \Box_{i+1}^{\beta_n} \in Q, E_i \models \alpha \land \forall j \in \{1, ..., n\} \ E \not\models \Box_{i+1}^{\beta_j} \} \]
then \(E\) is an extension of \((Q, W)\) if and only if \(E = \bigcup_{i=0}^{\infty} E_i\). We assume \(GD(Q, E)\) denotes all defaults in \(Q\) that contribute to generate an extension \(E\). \(GD(Q, E)\) is called the generator of \(E\).

An ordered (prioritized) default theory \(T\) is defined as \((Q, W, <)\) where binary relation \(<\) is a strict partial order over defaults such that \(< \subseteq Q \times Q\). If \(<\) is a total order, then \(T\) is called fully prioritized default theory. A preference statement in natural language can be translated into an ordering on defaults. For example, for a user’s choice of food, fish is preferred to steak can be expressed as:
\[ \frac{\text{fish}}{\text{steak}}. \]

**Brewka’s and Eiter’s Principles**

Brewka and Eiter defined two principles as basic necessary properties for handling preferences in default logic [34]. They argued that these requirements were not met in the previous work. Principle 1 suggests that for an ordered default theory \(T_o = (Q, W, <), X_1\) and \(X_2\) as extensions of \((Q, W)\), and \(Q' \subseteq Q\), if \(GD(Q, X_1) = Q' \cup \{q_1\}\), \(GD(Q, X_2) = Q' \cup \{q_2\}\), and \(q_1 < q_2\), then \(X_2\) is preferred to \(X_1\) and \(X_1\) is not in the set of preferred extensions of \(T_o\). According to principle 1, for two extensions \(X_1\) and \(X_2\), if the generators of \(X_1\) and \(X_2\) differ only in \(q_1\) and \(q_2\) (\(q_1\) is in the generator of \(X_1\) and \(q_2\) is in the generator of \(X_2\)) and \(q_2 > q_1\), then \(X_2\) is preferred to \(X_1\). Principle 2 states that a preferred extension of a prioritized default theory is not affected by adding a new default rule that is not contributing in generating that preferred extension. Let \(T_o = (Q, W, <)\) and \(T_a = (Q \cup \{q_a\}, W, <_{a})\) be prioritized default theories where for all \(q, q' \in Q\), if \(q < q'\) then \(q <_{a} q'\). For an extension \(X\) of \(T_o\), if \(q_a\) is a default such that \(Prereq(q_a) \not\in X\), then \(X\) is also an extension of \(T_a = (Q \cup \{q_a\}, W, <_{a})\). The authors of [34] examined some prioritized default frameworks in the literature with their proposed principles. For example, the proposal in [4] violates the first principle and [110] does not satisfy principle 2.

Brewka and Eiter proposed a new framework [34] that satisfies required conditions in principle 1 and principle 2. In their proposed approach, first, a method to construct preferred extensions in super normal default theories (prerequisite free and normal default theories) was defined. Then, the method was generalized to prerequisite free default theories. Finally, in the most general case, the extensions of prioritized standard default theories that can be reduced to prerequisite free theories were determined. Given a fully prioritized super normal default theory \(T = (Q, W, <)\), the following constructive definition specifies extension \(X\) of \(T\). We call a default \(q\) active in a set \(Y\) if \(\text{prereq}(q) \in Y\) and \(\neg \text{just}(q) \cap Y = \emptyset\) and \(q\) is not applied yet (\(\text{cons}(q) \not\in Y\)). An operator \(\Gamma\) is defined as \(\Gamma(T) = \bigcup_{i>0} X_i\) where \(X_0 = Th(W)\) and \(X_i\) is

\[
X_i = \begin{cases} 
\bigcup_{j<i} X_j & \text{if there is no default } q \text{ active in } \bigcup_{j<i} X_j \\
Th(\bigcup_{j<i} X_j \cup \{\text{cons}(q)\}) & \text{otherwise, for } q \text{ as the most preferred active default in } \bigcup_{j<i} X_j 
\end{cases}
\]

90
$X$ is a preferred extension of $T$ if and only if $X = \Gamma(T)$. Default theory $T$ has only one preferred extension $X$ that is the fixed point of operator $\Gamma$.

To determine extensions of prioritized default theories, they are reduced into prerequisite free theories. This characterization of prioritized defaults respects both principle 1 and principle 2. The complexity of credulous reasoning is $\Sigma^p_2$-complete and for skeptical reasoning it is $\Pi^p_2$-complete.

Prioritized answer set programming in its early stages was also characterized as a program with a preference ordering on rules. A prioritized (ordered) program $P$ is a pair $P = (\Pi, \prec)$ where $\Pi$ is a logic program with answer set semantics and $\prec$ is a strict partial order on rules in $\Pi$. A preferred answer set of $P$ is an answer set of $\Pi$ when rules are applied orderly with respect to $\prec$ [47].

**Axiomatic Approach**

An axiomatic approach for handling preferences in ordered logic programs was proposed in [47] in which relation $\prec$ over rules of a logic program is translated into a set of a rules of a new logic program. This axiomatic approach has a number of benefits, such as: 1) the flexibility to express static preferences (i.e., preferences are not dependent to each other) and dynamic preferences (i.e., preferences on rules can be expressed by some new program rules), 2) the complexity of answering queries (e.g., deciding whether a model is a preferred answer set of an ordered program) remains the same as in the original programs, and 3) existing ASP solvers can be used to find preferred answer sets.

Let $P = (\Pi, \prec)$ be an ordered program defined over a propositional language $\mathcal{L}$. A function $\mathcal{F}$ translates $P$ to $P^*$ that is a standard program (a program with no order on rules). The idea is to convert $\prec$ to some rules and embed those rules into $P^*$ based on the following two basic principles: First, if $r_1 > r_2$ where $r_1, r_2 \in \Pi$, $r_1$ is considered to be applied or blocked before $r_2$. Second, $r_2$ is ok to be checked if $r_1$ is already either applied or blocked. For the complete list of the translation rules see [47]. An answer set $S$ of $P = (\Pi, \prec)$ is called preferred if $\mathcal{F}(P)$ has answer set $S'$ such that $S = S' \cap \mathcal{L}$. Since the translation of $P = (\Pi, \prec)$ is in polynomial time, deciding whether $P$ has a preferred answer set can be reduced in polynomial time to decide whether $\mathcal{F}(P)$ has an answer set, which is NP-complete.

A similar axiomatic approach for prioritized default logic was proposed in [46] where unique names are assigned to default rules. A new auxiliary predicate $\prec$ is defined to represent a relation among default names such that for defaults $d_1$ and $d_2$, the relation $d_1 \prec d_2$ is converted to $n_{r_1} \prec n_{r_2}$ where $n_{r_1}$ and $n_{r_2}$ are names assigned to $d_1$ and $d_2$, respectively. The proposed framework allows to represent both static and dynamic preferences in default logic. The relation between defaults is axiomatized and added as a set of facts to the knowledge base. Because of a polynomial translation, the complexity of answering queries does not change. Also, the existing default theory provers can be employed to find preferred extensions.

Prioritized default theories can be extended to the case that an order relation is defined on a set of subsets of defaults. For instance, for a default theory $(Q, W)$, an order can be defined as
\[ S_1 > S_2 > S_3 \] where \( S_1, S_2, S_3 \subseteq Q \). This means that, for example, \( S_1 \) is preferred to \( S_3 \). This generalization of prioritized default theories was proposed in [50]. The idea of the translation is similar to prioritized theories. Defaults in a set with a higher priority are applied before defaults in sets with a lower priority.

**Static preferences** are defined as preferences ordering on default rules. They are not context-dependent and always remain the same.

**Dynamic Preferences** are also defined as preferences ordering on defaults (or rules of a logic program). A dynamic preference can also be represented by some default rules. This type of ordering is called dynamic. A predicate \( \prec \) is defined to express ordering on defaults. For example, given an extension \( X \), for default \( \frac{n_q \prec n_{q'}}{n_q \prec n_{q'}} \), if \( n_q \prec n_{q'} \) is consistent with \( X \), then \( n_q \prec n_{q'} \) is preferred to \( n_{q'} \).

Similar to static ordering, a set of constant names \( N = \{n_1, ..., n_k\} \) are added to the language of the default theory. Each default \( q \) is associated with a name \( n_q \) in \( N \). There is a relation between static and dynamic preferences as follows: Assume a prioritized default \( (Q, W, \prec) \) and its translation \( Tr(Q, W, \prec) \) is defined over a propositional language \( L \). Let \( Tr_d(Q, W \cup \{n_q \prec n_{q'}|q < q'\}) \) be the translation of a default theory with dynamic preferences. If \( X_s \) is an extension of \( Tr(D, W, \prec) \), then \( Tr_d(Q, W \cup \{n_q \prec n_{q'}|q < q'\}) \) has an extension \( X_r \) such that \( X_s \cap L = X_r \cap L \).

By translating a prioritized default theory into a default theory, the properties of default theories can be employed to determine the existence of an extension in that prioritized default theory. In [105], it was proved that certain types of semi-normal default theories are guaranteed to have extensions. A default is called semi-normal when its consequent is entailed by its justification (e.g., \( \frac{\alpha \beta \gamma \gamma}{\gamma} \)). A default theory is semi-normal if all defaults in the theory are semi-normal. For a set of defaults \( Q \), the dependency graph \( G_d \) is a graph that each node represents a default in \( Q \). There is an edge between two nodes \( q_1 \) and \( q_2 \) if the consequent of \( q_1 \) may contribute in applying or blocking \( q_2 \). It was proved in [105] that a semi-normal default is guaranteed to have an extension if all cycles in the dependency graph have even weight (assume the weight of each edge is one). This result can be used for prioritized semi-normal propositional default theories. Assume \( T = (Q, W, \prec) \) is a prioritized semi-normal propositional default theory. Let \( G_d = (V, E) \) be the dependency graph of \( (Q, W) \). Consider \( G^*_d = (V, E \cup \{(q, q')|q > q'\}) \) extends \( G_d \) such that if \( q > q' \), an edge \((q, q')\) is added to \( G_d \). If all cycles in \( G^*_d \) have even weight and if each cycle \( C \) in \( G^*_d \) is a cycle in \( G_d \), then \( Tr(Q, W, \prec) \) is guaranteed to have an extension. In other words, if the ordering on the defaults is compatible with the dependency graph, then there exists an extension for the translation of the prioritized semi-normal default theory [46].

At the end, we note that one of the distinctive properties of the axiomatic approaches is that they generate only preferred extensions [46]. On the other hand, there are some proposals in the category of preferences over rules, such as [110], in which the set of all extensions are generated and then a preorder over the extensions is deduced from an ordering on the defaults. It is worth noting that in this thesis, our approach to model preferences, intuitively, is similar to the category of preferences
over atoms. Our goal is to construct a language-independent preference framework for computational problems. Finding preferred models when a preference relation is defined over the rules in a certain language depends on the syntax and semantics of that language, which is not aligned with our goal of modelling preferences language-independently. Also, preference handling approaches in which preferences are defined over rules mostly generate optimal (the most preferred) models and in many cases cannot determine a preference relation over non-optimal models. However, one might need a comparison between two models (e.g., the Dominant Structure problem) instead of computing the most preferable models.

Preferences in Autoepistemic Logic

Autoepistemic logic was proposed by Moore [102]. It is a nonmonotonic modal logic that specifies the belief of an agent about its own knowledge. The modal language $L_k$ is constructed by modal formulas, boolean connectives and a modal operator $K$. A modal formula is either a propositional formula or it is of the form $K\varphi$ that means that modal formula $\varphi$ is believed by a rational agent. Stalnaker [123] characterized the properties of a belief set for a rational agent such that for a set $S$ of modal formulas in a modal language $L_k$: 1) $Cn(S) \subseteq S$, 2) if $\varphi \in S$ then $K\varphi \in S$, and 3) if $\varphi \notin S$ then $\neg K\varphi \in S$. In this characterization of belief sets, $Cn$ is the propositional consequence operator. As suggested in [102], for a set of modal formulas $W$ as the assumption, a belief set $S$ of modal formulas is an expansion of $W$ if $Cn(W \cup \{K\varphi|\varphi \in S\} \cup \{\neg K\varphi|\varphi \notin S\}) = S$. Moore also characterized a constructive definition of the expansion of belief sets based on possible world structures. For more details see [102].

A preference framework for autoepistemic logic was proposed in [109] with the similar idea the same author used for prioritized default logic in [110]. Consider a pair $O = (Q, <)$ where $Q$ is a set of modal formulas and $<$ is a strict partial order relation on $Q$. Assume $X = \{X_1, ..., X_n\}$ is the set of all expansions. $X_i$ is a preferred expansion with respect to $O$, if there is a strict total order $<_t$ such that $<_t \subseteq <$ and for all $X_j \in X$ ($j \neq i$) and for all $\varphi \in Q$, if $\varphi \in X_i - X_j$, then there exists a modal formula $\psi \in X_j - X_i$ such that $\psi <_t \varphi$. This proposal uses a lifting method similar to the Element Dominance semantics we introduced in Chapter 5. However, in our proposal, preference are defined over ground atoms, but in [109] preference are defined over a set of modal formulas.

7.3 CP-nets

As discussed in Chapter 4, CP-nets are among the most popular preference modelling approaches. The idea behind CP-nets was inspired by Bayesian networks and originally was proposed in [20]. CP-nets provide an intuitive framework with a graphical representation for modelling conditional preferences. In Chapter 4, we studied the relation between preference-based modular systems and CP-nets. Several problems associated with CP-nets have been widely studied in the literature. For example, the authors of [20] defined the dominance query for a CP-net $N$ as follows: Given outcomes $o_1$ and $o_2$, decide whether $N \models o_1 > o_2$. The complexity of the dominance query varies,
based on the structure of a CP-net, from polynomial to undecidable in the size of \( N \) (the number of variables). The authors of [20] studied the complexity of the dominance query for four different binary-valued CP-nets (variables with binary values) with an assumption that the size of \( Pr(v) \) (the parents of node \( v \)) for each variable \( v \) is bounded by a constant factor. First, consider a CP-net with a tree structure. An algorithm called TreeDT was introduced in [20] for dominance queries in tree structured CP-nets. The running time of TreeDT is \( O(n^2) \) where \( n \) is the number of variables in the CP-net. Second, for polytree (trees with multiple roots) CP-nets, dominance queries are also in polynomial time [27]. Third, for a CP-net that there is either no path or one path between every two nodes in the graph (that is called directed path singly connected CP-nets), dominance queries are NP-complete. Finally, as discussed in [27, 55, 20], dominance queries for more general cases of \( m \)-connected CP-nets where there are at most \( m \) paths between any two nodes are also NP-complete.

In a case that values of a variable are partially ordered, the complexity of dominance queries is not changed.

Ordering queries ask whether \( N \not\models o_2 \succ o_1 \) holds. If the answer is yes, \( o_1 \) is called orderable over \( o_2 \). We note that \( N \not\models o_2 \succ o_1 \) does not necessarily imply that \( N \models o_1 \succ o_2 \). Thus, ordering queries are weaker than dominance queries. It was proved in [20] that ordering queries are in linear time in the size of \( N \).

The task of finding the most preferred outcome (outcome optimization) given an assignment to a subset of variables \( A \subseteq V \) can be performed in linear time in the size of \( N \). An algorithm called forward sweep was introduced in [20]. The idea of forward sweep is to assign the most preferred value to each variable \( v \) in \( V - A \) according to \( CPT(v) \) given an assignment \( a \in \bar{A} \).

There are a large number of studies on the extension of CP-nets. For example, gCP-net [70] is a generalization of CP-nets such that a CP table associated with each variable is allowed to be incomplete. A more recent example is a proposal in [76], which is a framework to aggregate generalized CP-nets (gCP-nets). In order to combine gCP-nets, [76] employed the idea of integrating CP-nets that is called mCP-nets from [93] using four different semantics, including pareto, majority, max, and ranking. The authors of [76] studied several reasoning tasks associated with the aggregation of multiple gCP-nets (mgCP-nets), such as dominancy and optimality.

### 7.4 Preferences in Planning

Preference-based planning is one of the primary topics focused on the role of preferences in AI. A planning problem is the task of finding a sequence of actions that are applied on an initial state to reach a certain goal state. Recall from Chapter 4 that a planning problem is defined as a triple \( T = \langle D, s_0, G \rangle \) where \( D \) indicates the pre-conditions and effects of actions, \( s_0 \) is the initial state, and \( G \) stands for the goal of plans. A solution to a planning problem, that is called a plan, is a chain of actions and states \( s_0, a_1, \ldots, a_n, s_n \) that starts from \( s_0 \) and ends at \( s_n \) such that \( s_n \models G \).

A large amount of research has been conducted for modeling planning problems. For example, PDDL [96] and STRIP [38] are two prominent languages for formalizing planning problems. A
prioritized planning problem is defined as a pair \((T, \succeq)\) where \(T\) is a standard planning problem and \(\succeq\) is a preference relation on \(P\) that is the set of all acceptable plans of \(T\). For \(p, p' \in P\), \(p \succeq p'\) stands for \(p\) is at least as preferred as \(p'\). The interpretation of relation \(\succeq\) varies in different frameworks. For example, \(p\) is preferred to \(p'\) if a certain action is completed or a formula is true in a certain state. In the context of planning, preferences can be temporal properties of plans, about the final state, or a combination of both. For example, in [31], preferences are defined in the final state based on a ranked knowledge base while in [54, 69, 12, 120] temporal preferences are defined for the plans.

In Chapter 4, we studied \(PP\) [120], which is a preference-based planning framework, and its relation to preference-based modular systems. In \(PP\), preferences are viewed as constraints. A preferable plan satisfies the preference constraints while a plan that does not satisfy those preference constraints can not be preferred. In \(PP\), only preferred plans are generated and there is no way to compare two non-preferred plans. However, in our proposals of Prioritized Model Expansion and preference-based modular systems, structures (models) are generated and then compared with respect to preferences.

An extension to \(PP\) was proposed in [12] to combine preferences. The proposed framework is called a language for preference-based planning (LPP). A satisfaction degree from a set \(V\) is assigned to each preference formula. Set \(V\) can be a continuous range of real numbers, such as \([0,1]\), or a totally ordered discrete set, such as \{excellent, good, ok, bad\}. As an example, according to [81], consider atomic preferences \(\phi_1, \phi_2, \phi_3\). Assume an atomic preference \(\Phi\) in LPP that is defined as \(\Phi = \phi_1 \land \phi_2[\text{excellent}] > \phi_2[\text{good}] > \phi_1[\text{ok}]\). This means that it is excellent that \(\phi_1\) and \(\phi_2\) are satisfied together (i.e., the most preferred) while the satisfaction of \(\phi_2\) is good and \(\phi_1\) is ok to be satisfied. Now, assume another atomic preference \(\Phi' = \phi_3[\text{excellent}] > \phi_4[\text{good}]\). Similar to general preferences in \(PP\), \(\Phi\) and \(\Phi'\) can be aggregated by connectives \&, |, and !. To generate a ranking among plans based on preferences in LPP, consider a set \(V\) such that \(V = \{v_0, ..., v_n\}\). Let \(\Phi\) be of the form \(\phi_0[v_0] >, ..., > \phi_n[v_n]\) and \(w(\Phi)\) denotes the weight of \(\Phi\). For a plan \(\rho\), define \(w_\rho(\Phi) = v_k\) where \(k = \min \rho | \models \phi_i\). If there is no \(\phi\) that is satisfied by \(\rho\), then assign \(w_\rho(\Phi) = v_{\text{max}}\) where \(v_{\text{max}}\) is the maximal member of \(V\). A total order on plans is deduced from the satisfaction level of each preference formula based on totally ordered set \(V\). This differs from a partial order relation (e.g., \(\succeq_\phi\)) generated by preferences in \(PP\).

Some more examples

PDDL3 [69] is a quantitative language with temporal preferences using connectives similar to Linear Temporal Logic (LTL) temporal connectives such as \textit{next}, \textit{at-most-once}, \textit{sometime}, and \textit{always}. Preferences in the goal state is expressed by \textit{at-end-state}. Quality of a plan with respect to a preference \(\Phi\) is measured by the maximization or minimization of a numerical function based on the satisfaction or falsification of \(\Phi\). The proposal in [118] extends PDDL3 to model preferences in hierarchical task network (HTN), which is a generalization of the standard planning problem. Temporal preferences are represented based on the LTL semantics and HTN decomposition is translated
into situation calculus. The satisfaction of preferences in a plan is defined based on the logical entailment and plans are compared with respect to the value of a metric function similar to PDDL3.

Planners for preference-based planning problems
A preference-based planning problem \((T, \succ)\) is optimal if it generates an optimal plan with respect to \(\succ\). The authors of [81] surveyed a variety of algorithms employed by different planners to find optimal plans using search techniques, heuristic methods, approximation, and so on. An algorithm for a planner is called \(k\)-optimal if it generates an optimal plan from a set of plans with a makespan (the number of steps to be taken to implement a plan) at most \(k\). Also, an algorithm is incremental if there is a family of preference-based planning problems \(\mathcal{T}\) such that for all \((T, \succ)\) in \(\mathcal{T}\), the algorithm produces a series of plans \(\rho_1, \ldots, \rho_n\) where \(n \geq 1\) and \(\rho_i \succ \rho_j\) when \(i < j\). Preference-based planners are divided into two major categories: planners that use search-based algorithms and planners that employ general solvers.

Search algorithms are the main core of search-based planners. For example, [81] introduced \(P_{\text{REF-S SEARCH}}\) that is a general best-first search algorithm with branch and bound that can be applied for both qualitative and quantitative approaches. The plan with the best quality found so far is stored and an evaluation function measures the search branch and estimates the quality of the plan that will be made at the end of the branch. The search branch will not be examined if it does not generate a plan with a better quality based on the estimation. \(P_{\text{REF-S SEARCH}}\) is incremental and can be \(k\)-optimal. As an example, BBOP-LP [10], which is an instance of \(P_{\text{REF-S SEARCH}}\), is an incremental planner that uses a branch and bound technique to prune the search space and deals with preferences about the final states of plans. BBOP-LP employs linear programming to find the optimal solution to the planning problem that is encoded in integer programming. There are some search-based planners that can handle temporal preferences such as PP LAN, which is a best-first search planner primarily proposed for LPP language [12]. Also, HTNPP LAN was suggested in [118] for HTN planning tasks as an extension to PDDL3. Furthermore, there are several planners that use general solvers (such as SAT solvers or ASP solvers). For example, [121] proposed ASPlan for \(P^P\) language that converts a preference-based planning problem into an ASP problem and Smodels, which is an ASP solver, finds the optimal solution.

In our proposed algorithm for solving Prioritized Model Expansion problems in Chapter 5, we use solvers of modular system to generate a solution at the current iteration and compare it with the solution found in the previous iteration until the most preferred solution is achieved. But, in \(P^P\), since preferences are considered as constraints, models that are generated by a general solver are preferable and it is not required to compare them with other models. Our proposed algorithm can also be viewed as an incremental algorithm because it outputs a totally ordered set of structures (each output is preferentially improved in compare to the outputs in the previous iterations).
7.5 Preference-based Argumentation

Argumentation theory [58] studies conflicting knowledge bases and methods to derive conclusions from knowledge bases in the presence of conflict. One of the important applications of preferences in AI is in argumentation theory. Preferences in an argumentation framework can be used to handle conflicts that cannot be resolved without preferences. The argumentation framework proposed by Dung [58] is defined as $AF = \langle A, R \rangle$ where $A$ is a set of arguments. $R$ is a binary relation over $A$, which is called an attack relation. Given two arguments $X$ and $Y$, $X \triangleright Y$ means that $X$ attacks $Y$. A set $B$ where $B \subseteq A$ is called conflict-free when there do not exist $X, Y \in B$ such that $X \triangleright Y$ ($X$ attacks $Y$). Also, $X \in A$ is acceptable with respect to $B$ if and only if for all $Y \in A$ if $Y \triangleright X$, there is some $Z \in B$ that attacks $Y$. In other words, $B$ defends $X$ from external attacks. A set of arguments $B$ where $B \subseteq A$ is called an admissible extension of $AF = \langle A, R \rangle$ if $B$ is conflict-free and every argument $X$ in $B$ is acceptable with respect to $B$. In other words, if there are no two arguments in $B$ that attack each other and every argument in $B$ is defended by $B$, then $B$ is an admissible extension. $B$ is a stable extension of $AF$ if and only if it is admissible, there is no $C$ such that $B \subset C$, and it attacks each argument $Y \in A - B$. Furthermore, $B$ is a complete extension of $AF$ when it is an admissible extension and every argument $X$ that is acceptable with respect to $B$ is in $B$. A minimal (based on set inclusion) complete extension is called ground extension.

A preference-based argumentation framework [116] is defined as a triple $AF_p = \langle A, R_p, \preceq \rangle$ such that $A$ is a set of arguments, $R_p$ is an attack relation over $A$, and $\preceq$ is a preorder over $A$. $AF_p$ represents a standard argumentation framework $AF = \langle A, R \rangle$ whenever for all $X$ and $Y$ in $A$, $X \triangleright Y$ if and only if $X \triangleright_p Y$ and $Y \succ X$ does not hold. Each preference-based argumentation framework $AF_p$ represents a unique standard argumentation framework, namely $AF$. Acceptable extensions of $AF_p$ are acceptable extensions of $AF$.

Amgoud and Vesic [2] argued that the proposal in [116] has some drawbacks. For example, consider $AF_p = \langle \{X, Y\}, Y \triangleright_p X, X \succ Y\rangle$. $AF_p$ represents $AF = \langle \{X, Y\}, \emptyset \rangle$. Thus, $\{X, Y\}$ is an acceptable extension of $AF$ while $X$ and $Y$ are in conflict with respect to $R_p$. To address this issue, Amgoud and Vesic proposed a new framework [2] with a conflict relation that is symmetric. Consider $AF_s = \langle A, R_s, \succeq \rangle$ where $A$ is a set of arguments, $\succeq$ is a partial order over $A$, and $R_s$ is a symmetric conflict relation over $A$ such that if $X \triangleright_s Y$, then $Y \triangleright_s X$. Preference-based framework $AF_s$ represents a standard framework $AF = \langle A, R \rangle$ such that for all $X, Y \in A$, $X \triangleright Y$ if and only if $X \triangleright_s Y$, $Y \triangleright_s X$, and $Y \succ X$ does not hold. In other words, if $X$ and $Y$ are in conflict with respect to $AF_s$ and $Y$ is not preferred to $X$, then we can conclude that $X$ attacks $Y$ in $AF$. In the aforementioned example, if the conflict relation is symmetric, then $\{X\}$ is the only acceptable extension, which is, intuitively, compatible with the notion of conflict-free extensions.

In both [2] and [116], preferences over arguments are used to generate acceptable extensions containing preferred (or stronger) arguments (similar to $PP$ in preference-based planning in which only preferred plans are generated). In [122], a method was suggested to compare extensions (even if they do not have preferred arguments) based on preferences over arguments. Given a preference-
based framework $AF_p = \langle A, R_p, \succeq \rangle$, to rank the extensions of $AF_p$, consider the following three steps: First, the acceptable extensions of $AF_p$ are generated. Second, a complete preorder, notation $\leq_c$, over $A$ is constructed from partial preorder $\succeq$. Third, extensions of $AF_p$ are ordered with respect to $\leq_c$. A complete preorder $\leq_c$ can be constructed from two different perspectives: I) minimal specificity principle, and II) maximal specificity principle.

Let a set of arguments $A$ be partitioned into $B_1, ..., B_n$ such that $A = \bigcup_{i=1}^n B_i$. Assume these partitions are ordered in a sequence $B_1, ..., B_n$ such that for $i < j$, $B_i$ is favored over $B_j$. Based on the minimal specificity principle, for an argument $X$ in $B_i$, there is no argument $Y$ such that $Y \succ X$ ($X$ is not dominated). The minimal specificity denotes that each argument is preferred except there is an evidence against it based on relation $\succeq$. Elements in $B_2$ can be dominated only by elements of $B_1$. Similarly, members of $B_i$ can be dominated only by elements of $B_1, ..., B_{i-1}$. For arguments $X$ and $Y$, we have $X \geq_h Y$ if $X \in B_i$ and $Y \in B_j$ such that $i < j$.

Based on the maximal specificity principle, all arguments are not preferred unless partial order $\geq$ states otherwise. Each argument in $B_n$ is not preferred to any argument in $A$. Arguments in $B_{n-1}$ are only preferred (dominant) to arguments in $B_n$. Every argument $X$ in $B_i$ is preferred to every argument $Y$ in $B_j$ when $i < j$ (notation $Y \leq_i X$).

To compare extensions with respect to $\leq_1^c$ or $\leq_h^c$ (for simplicity, $\leq^c$), three different comparison semantics were proposed in [122] that are listed as follows:

1. For two sets of arguments $B$ and $C$ such that $B, C \subseteq A$ and for a complete preorder $\leq^c$ over $\mathcal{A}$, we say $B$ is preferred or equally preferred to $C$ (notation $B \sqsupseteq C$) if $C \subseteq B$ or for all $X$ in $B - C$ and for all $Y$ in $C - B$, $Y \leq^c X$.

2. $B$ is at least as optimistically preferred as $C$ (notation $B \sqsupseteq_{\text{opt}}^c C$) if $C \subseteq B$ or for all $X$ in $\text{Max}(B - C, \leq^c)$ and for all $Y$ in $\text{Max}(C - B, \leq^c)$, we have $X \geq^c Y$ where $\text{Max}(\mathcal{S}, \leq^c) = \{Z \mid Z \in \mathcal{S} \text{ and } \not\exists U \in \mathcal{S} : U >^c Z\}$.

3. $B$ is pessimistically preferred or equally preferred to $C$ (notation $B \sqsupseteq_{\text{pes}}^c C$) if $C \subseteq B$ or for all $X$ in $\text{Min}(B - C, \leq^c)$ and for all $Y$ in $\text{Min}(C - B, \leq^c)$, we have $X \geq^c Y$ where $\text{Min}(\mathcal{S}, \leq^c) = \{Z \mid Z \in \mathcal{S} \text{ and } \not\exists U \in \mathcal{S} : U <^c Z\}$.

These comparison semantics can be viewed as lifting semantics similar to what we proposed in Chapter 3. For example, (1) describes the Weak Pareto semantics and (2) and (3) are special cases of the Weak Pareto semantics.

A preference relation between two arguments can be expressed as an argument [101] similar to the idea of dynamic preferences in nonmonotonic frameworks. For instance, arguments $Z$ claims that argument $X$ is preferred to argument $Y$. According to [101], $AF_e = \langle A, \mathcal{R}_e, \mathcal{D} \rangle$ is an argumentation framework with arguments about preferences where $A$ is a set of arguments and $\mathcal{R}_e$ is a conflict relation that is symmetric and irreflexive. Also, $\mathcal{D} \subseteq A \times \mathcal{R}_e$ such that for $(Z, (X, Y)) \in \mathcal{D}$, $X$ is preferred to $Y$ based on the claim of $Z$. It is defined that if $Z_1 \mathcal{D}(X, Y)$ and $Z_2 \mathcal{D}(Y, X)$, then $Z_1 \mathcal{R}_e Z_2$ and $Z_2 \mathcal{R}_e Z_1$. $AF_e$ can be translated into a standard framework $AF = \langle A, \mathcal{R} \rangle$. For $X \mathcal{R}_e Y$
and Y ∉ \mathcal{A}, if there is no argument Z ∈ \mathcal{A} such that (Z, (Y, X)), then we have X ∉ Y. For example, suppose \mathcal{A} = \{X_1, X_2, X_3, X_4, X_5\} and \mathcal{R}_c = \{(X_1, X_2), (X_2, X_1), (X_3, X_4), (X_4, X_3)\}. Also, assume \mathcal{D} = \{(X_3, (X_2, X_1)), (X_4, (X_1, X_2)), (X_5, (X_3, X_4))\}. The acceptable extension is \{X_2, X_4, X_5\}.

Preferences over arguments are either specified by a decision maker or they can be constructed from relations among arguments. In [116], a preference relation is elicited based on the specificity relation among arguments in defeasible reasoning. In another example that was proposed in [9], certainty values assigned to arguments determine a preference relation among arguments.

### 7.6 Lexicographic Models and Preference Reasoning

Lexicographic preference models have been widely studied in the literature, e.g., preference trees [17, 92] and hierarchical models [138, 137, 139]. There are a variety of preference-based frameworks in which lexicographic models are employed for preference representation, such as non-monotonic frameworks [36], database systems [84], planning [120], etc.

Preference inference is a topic of interest in lexicographic models [138] and is defined as the task of deriving new preference relations from a given partial preference relation that satisfy a given lexicographic model. As discussed in [138, 137, 139], preference inference based on a set of possible lexicographic models is defined in the following steps: First, an input preference language \mathcal{L}^I is defined to express input preference statements. Second, a preference query language \mathcal{L}^Q is defined for preference queries. Finally, for a set of preferences \mathcal{P} in a language \mathcal{L}, a set of lexicographic models \mathcal{M} = \{m_1, ..., m_i\}, a set of input preference statements \Gamma in \mathcal{L}^I, and a preference query \mathcal{Q} in \mathcal{L}^Q, the task of deciding whether \mathcal{Q} is entailed by \Gamma based on \mathcal{M} is called a preference inference problem. This reasoning task can be in polynomial or computationally hard (NP-hard) depending on the structure of the lexicographic models.

As an example, we take a closer look at a preference reasoning approach [139] that was inspired by a generalization of hierarchical constraint logic programming (HCLP) [135]. An HCLP model is a lexicographic preference model that is defined based on an HCLP structure. An HCLP structure \mathcal{S} is a tuple \langle \mathcal{A}, \oplus, \mathcal{C} \rangle where \mathcal{A} is set of alternatives, \mathcal{C} is a set of evaluation function such that for each c ∈ \mathcal{C}, c : \mathcal{A} → \mathbb{Q}^+ where for a ∈ \mathcal{A}, c(a) = 0 means the most preferred value assigned to a, and \oplus is a binary operator on \mathcal{C} that is monotonic, associative, and commutative. For a_1, a_2 ∈ \mathcal{A} and C ⊆ \mathcal{A}, a_1 is preferred to a_2 with respect to C and operator \oplus (notation a_2 \sim^C_{\oplus} a_1) if \forall c ∈ \mathcal{C} c(a_1) ≤ \oplus_{c ∈ \mathcal{C}} c(a_2). The binary relation \sim^C_{\oplus} is a total pre-order. An HCLP model \mathcal{H} is an ordered series of disjoint subsets of \mathcal{C} of the form (C_1, ..., C_k) where \bigcup_{1 ≤ i ≤ k} C_i ⊆ \mathcal{C}. Given an HCLP model \mathcal{H} = (C_1, ..., C_k) and an alternative a, a vector V(a) is defined as V(a) = [\oplus_{c ∈ C_1} c(a), ..., \oplus_{c ∈ C_k} c(a)]. The relation a_1 \sim^H_{\oplus} a_2 between alternatives a_1 and a_2 holds if either V(a_1)[r] = V(a_2)[r] (1 ≤ r ≤ k) or there exists 1 ≤ i ≤ k such that
Table 7.1: Dessert Options

<table>
<thead>
<tr>
<th></th>
<th>Per 100 gram</th>
<th>Apple Pie</th>
<th>Chocolate Cake</th>
<th>Ice Cream</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calories</td>
<td>11</td>
<td>13</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Fat</td>
<td>23</td>
<td>23</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Sugar</td>
<td>20</td>
<td>17</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>Fat+Sugar</td>
<td>43</td>
<td>40</td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>

Preference Inference. Let $\mathcal{L}^A_{\preceq}$ be an input language that is a set of statements of the form $\phi = a_2 \leq a_1$. Given an HCLP model $H$ and a structure $\langle A, \mathbin{\oplus}, \mathcal{C} \rangle$, preference inference (entailment) based on model $H$ is defined as $H \models \mathbin{\oplus} a_1 \leq a_2$ if $a_1 \preceq H^a a_2$. For a set of preference statements $\lambda$ in $\mathcal{L}^A$, $H \models \mathbin{\oplus} \lambda$ if for all $\phi \in \lambda$, we have $H \models \mathbin{\oplus} \phi$. Given a set of HCLP models $M$ and $\psi = a_1 \leq a_2$, we say that $\lambda \models M^\mathbin{\oplus} \psi$ (where $\lambda$ is a set of preference statements in $\mathcal{L}^A$) if for all $H \in M$ that $H \models \mathbin{\oplus} \lambda$, also $H \models \mathbin{\oplus} \psi$.

Example 23. Imagine in a restaurant, alternatives for dessert are apple pie, chocolate cake, and ice cream. Each alternative has three attributes including: calories, sugar, and fat. An evaluation function (as provided in Table 7.1) assigns a value to each alternative’s attribute. Let us assume that $H = \langle \{Fat, Sugar\}, \{Calories\} \rangle$. Let operation $\mathbin{\oplus}$ be the standard addition on $\mathbb{Q}^+$. According to Table 7.1, we have $V(Ice Cream) = [40, 11]$ and $V(Chocolate Cake) = [40, 13]$. It is derived that $Chocolate Cake \preceq H^a Ice Cream$.

Example 24. Assume that $\lambda = \{(Chocolate Cake \geq Ice Cream)\}$. We want to check if $\lambda \models M^\mathbin{\oplus} \phi$ where $\phi = Apple Pie \geq Ice Cream$. If there is a model $H$ that satisfies $\lambda$ but does not satisfy $\phi$, then $\phi$ is not entailed by $\lambda$. Let us consider $H = \{(Fat, Sugar)\}$. $H$ satisfies $\lambda$ but $\phi$ is not satisfied by $H$. Therefore, we have $\lambda \not\models M^\mathbin{\oplus} \phi$. If we change $\lambda$ to $\{(Chocolate Cake > Ice cream)\}$ and $\phi$ to Apple Pie $< Ice Cream$, then $\lambda \models M^\mathbin{\oplus} \phi$ holds.

It was proved in [139] that if each $C_i$ in an HCLP model $H = \langle C_1, \ldots, C_k \rangle$ is a singleton, the problem of deciding whether $\lambda \models M^\mathbin{\oplus} \psi$ is in polynomial time and it is coNP-complete otherwise. In Chapter 4, we introduced the notion of a meta-preference that can be viewed as a lexicographic model. A comprehensive study of preference reasoning based on lexicographic models in the context of Prioritized Model Expansion and preference-based modular systems will be left for future work.
Chapter 8

Summary and Future Work

8.1 Summary

In this thesis, we studied handling preferences in (modular) Model Expansion problems. We summarize our contributions as follows:

In Chapter 3, we developed a language-independent approach to handle preferences in computational problems by introducing the Prioritized Model Expansion framework. We showed that some prominent declarative approaches (e.g., SAT and ASP) with preferences can be viewed as Prioritized Model Expansion. We studied the impact of introducing preferences to the complexity of query evaluations in Model Expansion problems.

In Chapter 4, we defined preference-based modular systems, which is a framework to combine Model Expansion problems with preferences. To the best of our knowledge, this is the first proposal for combining preferences with possible different host languages. We also studied the relationship between preference-based modular systems and a number of prominent preference-based approaches, such as CP-nets, Answer Set Optimization, and preference-based planning.

In Chapter 5, we presented a formal characterization of solvers of modular systems, which perform learning and reasoning to cut down the search space. Then, we proposed a naive algorithm for solving Prioritized Model Expansion problems. Next, we formally defined preference solvers, which finds the most preferred solutions, and their proprieties. We proved that a preference solver can be built from solvers of modular systems.

Finally, in Chapter 6, we presented a preference-based approach to find solutions to a combination of computational problems, possibly specified in different languages, that may have no solution. By prioritizing the constraints, an approximate solution was defined as the most similar (closest) structure to the solutions of the individual problems with more preferable constraints.

8.2 Future Work

In Chapter 5, we showed that a solver for a modular system can be used as a preference solver. One possible future work is to implement Algorithm 2 (solvers in the SPMX framework) that effi-
ciently performs operations in the algebra of modular systems, specially, the least fixed point, the Kleene star, and the max iterate. An efficient implementation of these iterative operations includes strategies that reduce the search space, such as ranking ground atoms, learning, and reasoning. Also, by introducing preferences to a Model Expansion problem, the complexity rises. For example, for first-order Prioritized Model Expansion, query evaluation is in $\Sigma^P_2$. Designing tractable algorithms using approximation would be an important priority for future work.

Another possible interesting direction could be designing a solver for finding approximate solutions of weighted compound Model Expansion problems. It seems possible to use solvers of modular systems in a similar way used for preference solvers. For such solvers, designing declarative heuristics that reduce the search space could be the topic of further study.

One more promising future work is to study the application of our proposed preference solvers in designing efficient solvers to find optimal outcomes in multi-component systems in which each sub-component solves a Model Expansion problem, such as business processes, large-scale software systems, engineering design and optimization problems, etc.

Lastly, as a long term plan, we are interested in applying mathematical modeling of preferences to the current trends in AI technologies, such as social networks analysis, recommender systems, image processing, etc. For instance, in supervised learning approaches, it could be interesting to study how a predictive model is built when context-dependent preferences of users are included or some feature values in a training data set are prioritized.
Bibliography


