Matching of Service Feature Diagrams based on Linear Logic

Thesis submitted for the degree of
Doctor of Philosophy
at the University of Leicester

by

Muhammad Naeem MIT (Pakistan)
Department of Computer Science
University of Leicester

2012
Matching of Service Feature Diagrams based on Linear Logic

Muhammad Naeem

Abstract

Managing variability is essential for an efficient implementation of end-user services that can be customised to individual needs. Apart from variations in selection and orchestration, also third-party services may have to be customisable. Although feature diagrams provide a high-level visual notation for variability, their use for specifying variability of services raises the problem of matching a required feature diagram against a set of provided ones.

In particular, the established interpretation of feature diagrams in Propositional Logic is not expressive enough for matching in the context of service variability. The problem becomes more visible when a certain requirement is going to be satisfied by a combination of multiple offers with overlapping features, which is a consequence of idempotence in Propositional Logic.

To address this problem, we propose service feature diagrams with semantics in Linear Logic. Linear Logic only allows the use of idempotence on the propositions with modalities. The permissible selection of features of a service feature diagram is called an instance diagram. We provide rules to obtain instance diagrams from the service feature diagram. The semantics of instance diagrams are also supported by Linear Logic.

This thesis not only introduces service feature diagrams, but also formalises their matching as linear deduction. We propose two categories of rules to verify diagrammatically if a collection of service descriptions satisfy the requirements. First, graphical matching rules are used to match service feature diagrams of requestor and provider. Second, graphical merging rules are used to merge multiple feature diagrams contributing to satisfy the requestor’s demands. We prove the correctness of these rules using the inference system of Linear Logic. We also provide the analysis of graphical rules and show that the application of the graphical rules is independent of the context in service feature diagram, i.e., graphical rule can be applied anywhere in a service feature diagram.
my mother and the sweet memories of my father,
the greatest sources of inspiration
in my life.
Acknowledgement

First of all, I would pay my gratitude to Allah (S.W.T.), The Most Merciful and The Mighty, for all His blessings. One of His great blessings on me was supervision of Professor Reiko Heckel. I learned from him what is research about. This thesis would not have been possible without his supervision. He was always available for help with an encouraging attitude whenever I got stuck. His comments and feedback opened new directions in my work. I sincerely appreciate his time and efforts which he has been providing from the start of my PhD. I will never forget his efforts in reading and giving very useful and in-time feedback on drafts of my thesis.

I would probably have missed the opportunity to work with Professor Reiko, had not Hazara University, Pakistan started Faculty Development Scheme under Hazara University Post-Quake Development Plan. I also thank Higher Education Commission, Pakistan for providing financial assistance in this scheme.

I am grateful to Professor José Luiz Fiadeiro. He was always available whenever I needed his help. His critical questions on the earlier versions of our work were very helpful, especially in formalising the matching. I would like to thank Dr. Krzysztof Czarnecki for introducing us to the Propositional Logic semantics of classical feature diagrams. I can not forget Dr. Dénes Bicztray for suggesting to use feature diagrams in our work during one of the group meetings. I want to thank Professor Fernando Orejas, whom I met during his visit to the University of Leicester in Summer 2009. His visit contributed to the partial matching of visual contracts.

I would like to thank my examiners Professor Richard Paige and Dr. Emilio Tuosto for their thorough and detailed review of this thesis. Their questions in a friendly environment opened new directions of my research and enabled the substantial improvement of this thesis.

The time spent in Computer Science Department has been great fun due to the presence of my friendly colleagues. It has been a real pleasure to share the office with Ajab, Fawad, Nosheen, Shakeel, and Stelious. The friendly discussions of Niaz, Ishrat, Dursun, and Muzammil were always enjoyable for me during my stay at the department. I am very much thankful to the teaching and administrative staff of our department for their in-time support, especially Dr. Fer-Jan de Vries for his help.

Of course, I thank my family for their support, their interest and for always being there when I needed them most. My father was very willing to see me as a PhD, but he passed away in January 2012. I really miss my father at this moment. His golden words will always remain with me as the guiding principles of my life. The continuous support and prays of my family make it possible for me to pass the long years of my PhD studies.
# Table of Contents

**Preliminary Pages** i – xviii

1 Introduction
---
1.1 Motivation .................................................. 1
1.2 Problem Statement ......................................... 4
1.3 Solution ...................................................... 4
   1.3.1 Contributions ........................................... 6
1.4 Thesis Outline .............................................. 7
   1.4.1 How to read thesis ..................................... 10

2 Background
---
2.1 Classical Feature Diagrams ................................. 11
   2.1.1 FODA Feature Diagrams ............................... 12
   2.1.2 Feature-RSEB ........................................... 14
   2.1.3 Cardinality-based Feature Diagrams ................. 15
   2.1.4 Interpretation ......................................... 17
2.2 Linear Logic ................................................ 19
   2.2.1 Concepts and Notations ............................... 19
   2.2.2 Inference System ..................................... 21
2.3 Service Specification and Matching ....................... 23
2.4 Summary ................................................... 25

3 Related Work
---
3.1 Matching and Merging of Models ......................... 27
3.2 Matching of Feature Diagrams ............................ 28
3.3 Composing Feature Diagrams .............................. 29
3.4 Feature Diagrams and First-order Logic .................. 31
3.5 Algebra of Features ..................................... 33
3.6 Communicating Sequential Processes ..................... 35
3.7 Feature Diagrams and Services ........................... 36
3.8 Linear Logic and Services ................................ 38
3.9 Summary ................................................... 38
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>From Classical to Service Feature Diagrams</td>
<td>39</td>
</tr>
<tr>
<td>4.1</td>
<td>Matching Challenges</td>
<td>39</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Resource vs. Shareable Features</td>
<td>40</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Your Choice or Mine</td>
<td>41</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Keeping my Options</td>
<td>42</td>
</tr>
<tr>
<td>4.2</td>
<td>Proposed Framework</td>
<td>43</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Concrete Syntax</td>
<td>45</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Metamodel</td>
<td>47</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Instance Diagrams</td>
<td>50</td>
</tr>
<tr>
<td>4.3</td>
<td>Discussion</td>
<td>53</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Resource vs. Shareable Features</td>
<td>53</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Your Choice or Mine</td>
<td>54</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Keeping my Options</td>
<td>54</td>
</tr>
<tr>
<td>4.4</td>
<td>Running Example</td>
<td>55</td>
</tr>
<tr>
<td>4.5</td>
<td>Summary</td>
<td>58</td>
</tr>
<tr>
<td>5</td>
<td>Semantics of Service Feature Diagrams</td>
<td>59</td>
</tr>
<tr>
<td>5.1</td>
<td>Interpretation</td>
<td>60</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Interpretation in Linear Logic</td>
<td>60</td>
</tr>
<tr>
<td>5.2</td>
<td>Instance Formulas</td>
<td>68</td>
</tr>
<tr>
<td>5.3</td>
<td>Encoding of Instance Diagrams</td>
<td>70</td>
</tr>
<tr>
<td>5.4</td>
<td>Validation of Instance Formulas</td>
<td>75</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Mandatory Feature</td>
<td>75</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Optional Features</td>
<td>76</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Alternative-groups</td>
<td>77</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Or-groups</td>
<td>78</td>
</tr>
<tr>
<td>5.5</td>
<td>Summary</td>
<td>80</td>
</tr>
<tr>
<td>6</td>
<td>Graphical Matching Rules for Service Feature Diagrams</td>
<td>82</td>
</tr>
<tr>
<td>6.1</td>
<td>Diagrams used in Generic Rules</td>
<td>83</td>
</tr>
<tr>
<td>6.2</td>
<td>Matching Rules for And-groups of Solitary Features</td>
<td>84</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Rules for And-groups of Mandatory Features</td>
<td>84</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Rules for And-groups of Optional (R) Features</td>
<td>85</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Rules for Mixed And-groups</td>
<td>86</td>
</tr>
<tr>
<td>6.3</td>
<td>Matching Rules for Alternative-groups</td>
<td>88</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Rules for Requestor-based Choice</td>
<td>88</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Rules for Provider-based Choice</td>
<td>90</td>
</tr>
<tr>
<td>6.4</td>
<td>Matching Rules for Or-groups</td>
<td>90</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Rules for Requestor-based Choice</td>
<td>91</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Rule for Provider-based Choice</td>
<td>92</td>
</tr>
</tbody>
</table>
# Table of Contents

6.5 Matching Rules for And-groups of Solitary and Group Features . 93  
6.5.1 Rules for And-groups of Solitary and Alternative-groups . 93  
6.5.2 Rules for And-groups of Solitary and Or-groups . . . . . . 95  
6.6 Matching Rules for Shareable Features . . . . . . . . . . . . . 98  
6.6.1 Rule for Shareable to Resource Feature . . . . . . . . . . 98  
6.7 Combining Multiple Graphical Rules . . . . . . . . . . . . . . 98  
6.8 Graphical Rules for Merging . . . . . . . . . . . . . . . . . . . . 99  
6.8.1 Rules for Merging of Service Feature Diagrams . . . . . . . 99  
6.8.2 Rules for Merging of Shareable Features . . . . . . . . . . 100  
6.9 Analysis of Graphical Matching Rules . . . . . . . . . . . . . . 105  
6.10 Application of Graphical Rules . . . . . . . . . . . . . . . . . . 112  
6.11 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114  

7 Formalisation and Verification of Matching 115  
7.1 Matching of Service Feature Diagrams . . . . . . . . . . . . . . . 115  
7.1.1 Definitions of Matching . . . . . . . . . . . . . . . . . . . . 116  
7.1.2 Relationships . . . . . . . . . . . . . . . . . . . . . . . . . 120  
7.1.3 Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123  
7.2 Verification of Graphical Rules . . . . . . . . . . . . . . . . . . . 125  
7.2.1 Inference Figures . . . . . . . . . . . . . . . . . . . . . . . 126  
7.2.2 Verification . . . . . . . . . . . . . . . . . . . . . . . . . . 128  
7.3 Verification of the Matching Process . . . . . . . . . . . . . . . . 130  
7.3.1 Context Extension . . . . . . . . . . . . . . . . . . . . . . 132  
7.3.2 Leaf Extension . . . . . . . . . . . . . . . . . . . . . . . . 135  
7.4 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 145  

8 Evaluation 146  
8.1 Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . 146  
8.1.1 Extension in Classical Feature Diagrams . . . . . . . . . . . 147  
8.1.2 Matching Rules of Service Feature Diagrams . . . . . . . . 149  
8.2 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 151  

9 Future Work 152  
9.1 Visual Contracts and Service Feature Diagrams . . . . . . . . . . 152  
9.2 Tool Support . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 155  
9.3 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 156  

Appendix: A 157  
A.1 Deduction of First Axiom . . . . . . . . . . . . . . . . . . . . . . 157  
A.2 Deduction of Second Axiom . . . . . . . . . . . . . . . . . . . . 157
| B.1 | Rules for And-groups of Optional (R) Features | 159 |
| B.2 | Rules for Mixed And-groups (R) | 160 |
| B.3 | Rule for Mixed And-group (P) | 163 |
| B.4 | Rules for Mixed And-group with Mixed Choices | 164 |
| B.5 | Rules for Alternative-groups (R) | 165 |
| B.6 | Rules for Alternative-groups (P) | 166 |
| B.7 | Rules for Or-groups (R) | 167 |
| B.8 | Rule for Or-group (P) | 168 |
| B.9 | Rules for And-groups of an Alternative-group (R) and Solitary Features | 169 |
| B.10 | Rules for And-groups of an Alternative-group (P) and Solitary Features | 170 |
| B.11 | Rules for And-groups of an Or-group (R) and Solitary Features | 171 |
| B.12 | Rules for And-groups of an Or-group (P) and Solitary Features | 172 |
| B.13 | Rule for Shareable Feature | 173 |

| C.1 | Merging Rule for Service Feature Diagrams | 190 |
| C.2 | Rules for Merging of Shareable Features | 191 |
| C.2.1 | Merging of Solitary and Group Features | 192 |
| C.2.2 | Merging of Group Features | 193 |

References | 211 |
# List of Figures

1.1 Architecture of Our Case Study ........................................... 2  
1.2 Manufacturer (left) and Client (right) ................................. 3  
1.3 How to Read Thesis .......................................................... 10  

2.1 A Mandatory Feature .......................................................... 13  
2.2 An Optional Feature ........................................................... 13  
2.3 An Alternative-group .......................................................... 13  
2.4 An Or-group ................................................................. 14  
2.5 A Feature Diagram showing Seats of Car ............................... 15  
2.6 A Group with Set Cardinality .............................................. 15  
2.7 A Clonable Feature ............................................................ 16  
2.8 Cardinality-based Feature Diagram showing Seats of Car .......... 17  
2.9 Service-oriented Architecture .............................................. 24  

4.1 Requirements and Offers for an Entertainment System of Car ... 40  
4.2 Provider (left) and Requestor (right) ..................................... 42  
4.3 Provider (left) and Requestor (right) ..................................... 44  
4.4 Service Feature Diagrams of Restaurant and Requests .............. 44  
4.5 A Sample Service Feature Diagram ....................................... 45  
4.6 Metamodel of a Service Feature Diagram (adapted from metamodel shown by Czarnecki et al in [CHE05b]) .................. 48  
4.7 Manufacturer’s Options for the Entertainment System of Car ... 49  
4.8 An Instance Diagram from the Feature Diagram Shown in Fig. 4.7 51  
4.9 Instance Diagrams of the Service Feature Diagram shown in Fig. 4.7 53  
4.10 Service Feature Diagram showing Client’s Requirements .......... 55  
4.11 Service Feature Diagram of a Car Manufacturer ..................... 56  
4.12 Requirements of Entertainment and Suppliers Providing Entertain-

5.1 An Example Service Feature Diagram .................................. 65  
5.2 A Service Feature Diagram Showing Provider’s Offer ............... 67
6.27 And-groups of Solitary and Or-groups Satisfy And-group of Optional (P) Features ................................. 97
6.28 Shareable Feature Satisfying Resource Feature ......................... 98
6.29 Merging of Feature Diagrams (with shared features) ............... 99
6.30 Merging of Feature Diagrams (without shared features) ............ 100
6.31 Merging of Solitary Features ................................................ 101
6.32 Merging of Solitary Feature and Alternative-group (R) ............ 104
6.33 Merging of Solitary Features and Or-group (P) ...................... 104
6.34 Merging of Group Features .................................................. 105
6.35 Merged Feature Diagram of Supplier 1 and Supplier 2 ............. 113

7.1 Travel Agent (left) and Requestor (right) ............................... 116
7.2 Instance Diagrams of Fig. 7.1 .................................................. 119
7.3 Description (Left) and Requirements (Right) .......................... 124
7.4 Sets of Instance Diagrams of Descriptions and Requirements of Fig. 7.3 ......................................................... 124

9.1 Type Graph labelled by features: $BT$ for Bank Transfer, $CC$ for Credit Card, $Gd$ for Guide, $F$ for Flight, $H$ for Hotel, $Tra$ for Transport ......................................................... 153
9.2 Visual contracts specifying booking operations from requester (top) and provider (bottom) point of view ...................... 153
9.3 Feature Diagrams of Requestor, Provider, and their Common Features ................................................................. 154
List of Tables

2.1 Encoding Features in Propositional Logic . . . . . . . . . . . . . . 18
4.1 Concrete Syntax of Service Feature Diagrams . . . . . . . . . . . . 46
4.2 Instance Diagrams of a Service Feature Diagram . . . . . . . . . . 52
5.1 Encoding of Service Feature Diagrams in Linear Logic . . . . . . . 62
5.2 Table Showing the Borrowed Encoding . . . . . . . . . . . . . . . 66
5.3 Encoding of Instance Diagrams in Linear Logic . . . . . . . . . . . 72
6.1 Table Showing the Diagrams of Generic Rule . . . . . . . . . . . . 84
6.2 Analysis of Graphical Rules for Solitary and Group Features . . . 106
6.3 Analysis of Graphical Rules for the Combination of Solitary and
Group Features . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 108
Glossary

A Meta-operator (⋆)

⋆ is a meta operator which can be replaced by either of \{⊗, ⊕, &\}.

AGG

Attributed Graph Grammar (AGG) is graphical rule based language supporting an algebraic approach to graph transformation.

Alternative-group (R)

Exactly one feature from an Alternative-group (R) must be chosen by the requestor, if its parent is chosen in an instance diagram of a service feature diagram.

Analysis of Matching Rule

The following list explains the terminologies used while analysing matching rules.

Combination of Multiple Rules

✓ was used to represent a case, where a provider description satisfies the requirements.

Identity Case

✓ id was used to represent a case when description and requirements are same in a matching rule.

Match

✓ is used to represent a case, where a provider description satisfies the requirements.

Mismatch

× is used to represent a case, where a provider description does not satisfy the requirement.

Mismatch due to Flexibility of Variable Features

× was used to represent a case, where a provider description does not satisfy the requirement due to the level of flexibility.
offered and required by the description and the requirements, respectively.

**Mismatch due to Selection Rights of Variable Features**

is used to represent a case, where a provider description does not satisfy the requirement when both parties want to keep the selection rights of their variable features.

**C**

**Capital Letters**

Capital letters $A, B, \ldots$ are used to represent the multiplicative conjunction of literals.

**Cardinality-based Feature Diagrams**

Cardinality-based feature diagram introduced by Czarnecki et al in [CBUE02].

**Classical Feature Diagrams**

We consider FODA, Feature-RSEB and Cardinality-based feature diagrams as classical feature diagrams.

**Core Feature**

A feature that must exists in all the instances of a feature diagram.

**CSP**

Communicating Sequential Processes (CSP) is a formal language for describing patterns of interaction in concurrent systems.

**D**

**Discovery Service**

The discovery service works as a service repository, where service provider publish their services and service requestors find their required services.

**F**

**FAMA**

FAMA stands for a Framework for automated analyses of feature models integrating some of the most commonly used logic representations and solvers proposed in the literature (BDD, SAT and CSP solvers are implemented).

**Feature-RSEB**

An extension to the FODA feature diagrams introducing Or-group in [GFd98].

**FODA**

Feature Oriented Domain Analysis which introduced feature modelling
in [KCH+90]. Its is a very long description, to check whether glossary package automatically cuts the extra bits or.

**FOSD**

Feature Oriented Software Development.

**G**

**Generic Feature**

A feature that may be converted into multiple concrete features, based on the type of generic feature.

**Alternative-group (RP)**

A generic feature that is either be replaced by an Alternative-group (R), or an Alternative-group (P).

**OptP-Man**

A generic feature that is either be replaced by an Optional (P), or a Mandatory feature.

**OptR-Man**

A generic feature that is either be replaced by an Optional (R), or a Mandatory feature.

**OptRP**

A generic feature that is either be replaced by an Optional (R), or an Optional (P) feature.

**Or-group (RP)**

A generic feature that is either be replaced by an Or-group (R), or an Or-group (P).

**Graphical Rule**

A rule that show the derivable combinations of a provider specification that can be one of the following categories.

**Generic Rule**

A graphical rule that consists of generic feature.

**Matching Rule**

A graphical rule showing the matchable combination of requestor and provider specifications.

**Merging Rule**

A graphical rule showing the merge-able combination of provider specification.

**Greek Letters**

Greek letters $\alpha, \beta, \ldots$ are used to encode the features represented by the diamond symbols of a service feature diagram.
Inference Figure

Inference figure is derived inference rule which is used to avoid the repeating patterns in deduction of graphical rules.

Inference Rule

An inference rule of a sequent calculus is of the form $\text{Rule} \frac{\text{Hyp}_1 \text{Hyp}_2}{\text{Concl}}$ where hypotheses (Hyp1 and Hyp2) and conclusion (Concl) are represented in the form of sequents, while Rule states the name of inference rule applied to Hyp1 and Hyp2 to get to the Concl.

Instance

A subset of the set of permissible selection of features of a feature diagram.

Instance Diagram

A subset of the set of permissible selection of features of a service feature diagram made by requestor.

Instance Formula

Instance formula linear encoding of an instance diagram.

L

Linear Logic

Linear Logic introduced by Girard in [Gir87], also called Classical Linear Logic (CLL). The following list shows the different fragments of CLL.

MALL

Multiplicative Additive Linear Logic, it is formed by the combination of multiplicative and additive connectives.

MELL

Multiplicative Exponential Linear Logic, it is formed by the combination of multiplicative and exponential connectives.

MLL

Multiplicative Linear Logic, it only allows the use of multiplicative connectives of Linear Logic.

Linear Logical Connectives

The following list represents the list of connectives allowed in Linear Logic.

Additive Conjunction ($\&$)

If literals $a$ and $b$ refer the features $a$ and $b$ then $a \& b$ encodes the existence of an alternative choice made by the requestor between features $a$ and $b$.

Additive Disjunction ($\oplus$)

If literals $a$ and $b$ refer the features $a$ and $b$ then $a \oplus b$ encodes
the existence of an alternative choice made by the provider between features a and b.

**Linear Implication** (→)
If literals $a$ and $b$ refer the features a and b then $a \rightarrow b$ encodes the fact that selection of a feature b requires the prior selection of feature a.

**Multiplicative Conjunction** (⊗)
If literals $a$ and $b$ refer the features a and b then $a \otimes b$ states the fact that the features a and b must be selected.

**Storage Operator** (!)
Storage operator (also called an ofcourse modality) is used to generate the multiple copies of a proposition. We use it to encode shareable features.

**Lower-case Letters**
Lower-case letters $a, b, \ldots$ are the literals representing the features of a service feature diagram.

**M**

**Mandatory Feature**
A mandatory feature must be chosen if its parent is chosen in an instance of a feature diagram.

**Mathematical Letters**
Mathematical letters $A, B, \ldots$ are used to represent arbitrary linear logical formulas.

**Multiset**
A multiset is generalization of the notion of set in which repetition of its members are allowed.

**N**

**Normal Form**
The Normal Form (NF) is an irreducible logical expression.

**O**

**Optional (P)**
Optional features chosen by the provider.

**Optional (R)**
Optional features chosen by the requestor.
Or-group (P)
At least one feature from an Or-group (P) must be chosen by the provider, if its parent is chosen in an instance diagram of a service feature diagram.

Or-group (R)
At least one feature from an Or-group (R) must be chosen by the requestor, if its parent is chosen in an instance diagram of a service feature diagram.

P

Pairwise Disjoint
The property of a collection of sets such that every two members of the collection are uncommon. Let \{m_A \mid A \in S\} is an arbitrary collection of sets. These sets are said to be pairwise disjoint if for every pair of distinct members m, we have \(m_A \cap m_B = \phi\).

Provider
A service provider creates a web service and publishes its interface and access information to the discovery service.

R

Requestor
A service requestor queries the requires services in the discovery service using various find operations and then binds to the service provider in order to invoke the required services.

Resource Feature
A feature that can be used only once.

S

Sequent
Sequent is an expression of the form \(\Gamma \vdash \Delta\) written in Gentzen’s style.

Sequent Calculus
A formal theory in which inference rules are represented in terms of sequents is called sequent calculus.

Service Feature Diagram
An extension to the classical feature diagrams that can meet matching challenges (cf. Chapter 4).

Shareable Feature
A feature that can be used multiple times.

SOA
Service-oriented Architecture.
Variable Feature

A feature that does not exist in all the instances of a feature diagram.
Chapter 1

Introduction

1.1 Motivation

In order to reuse existing business level functionality in a flexible and distributed way, service-oriented systems resemble supply chains where end-user products are manufactured from parts delivered by suppliers. Similarly, delivering a complex service to the end user requires the integration of elementary services from third-party providers [Erl09]. If end-user services are to be customisable to individual needs, the resulting variability has to be managed both within the end-user interface and the implementation of the service. Apart from variations in the selection of supplied services or their orchestration, these third-party services themselves may have to allow for customisation, leading to a service-oriented product line architecture [SCS07].

The prototypical example is a car manufacturer offering variations on a model, which requires suppliers for variable engines, gearboxes, audio and entertainment systems, just to name a few. In the case of software services, an online travel agency may use third-party services for booking hotels, flights, invoicing, and so on. If the agency offers different payment options or variable conditions for reservations, it may require this variability to be realised by third-party services.

In many scenarios, if a single provider offer can not satisfy the requestor’s demands, multiple offers have to be combined. For instance, if an agency is arranging a trip on behalf of a client who wishes to attend a conference, it may not only have
to book a flight and hotel, but also arrange for tickets for the opera and dinner at a well-known restaurant. If none of the backend services can meet all these requirements, the agency may have to use more than one provider, for example book separately with services for hotels and flights as well as for cultural activities.

In the thesis, we focus on the interface between the requestor integrating services for the end user and the providers supplying more elementary variable services. As put in [SCS07], these can be seen as provider product lines supplying a requestor product line in a service-oriented product line architecture.

For example, let us consider a Client, interested to customise a Car with specific gears, engine type and air conditioning. They find a Car Manufacturer offering the required variability. Please see Fig. 1.1 showing the architectural diagram of the running example we will use throughout the thesis. The Manufacturer gets some parts from external Suppliers. The rectangular boxes in the Fig. 1.1 show services along with their names at the top. Services use requires interfaces to define their requirements, whereas a provides interface is used to offer functionality to other services.

![Figure 1.1: Architecture of Our Case Study](image)

For example, a Client service states its requirements to be satisfied by the provides interface of the Manufacturer service, as shown in Fig. 1.1. The dotted arrows labelled \( \rightarrow \) show the direction of satisfaction. For example, the provides interfaces of the supplier services satisfy the requirements shown in the requires interfaces of the Manufacturer service, as shown in Fig. 1.1.

Feature diagrams provide a simple notation for service specification and variability [BSC10]. We use feature diagrams to model the requirements and offers. To keep
it simple we do not show the feature diagrams here, only refer to figures describing the requirements and offers. Although feature diagrams provide a high-level visual notation for variability in software product lines [KCH+90], their use for specifying variability of services raises the following questions.

1. Is there a way to distinguish the requirements that must be satisfied once from those that may be satisfied more than once?

2. Is there a way to decide, who will choose from the variable features?

For example, assume a Car Manufacturer is interested to get a CD Player for the Entertainment System of a Car finds Supplier 1 and Supplier 2 providing CD Players. In this case, both suppliers satisfy the Manufacturer’s requirement, but choosing both would be against the requirement of incorporating one CD Player only. However, classical feature diagrams¹ do not support the distinction between one or more occurrences of features. Their semantic interpretation in terms of propositional logic is not able to distinguish a single proposition from multiple occurrences as a consequence of an idempotence. For example, $A \land A = A$, where $A$ is a literal referring to a feature of a feature diagram. Hence, we propose to extend feature diagrams distinguishing feature types for single and multiple occurrences.

With requestor and provider being independent entities, there are different ways to resolve alternatives. For example, if a Client looking for a Car is interested to buy an Automatic, and finds a Manufacturer offering a Car with Automatic or Manual gears, the Client will be satisfied if allowed to choose the type of gear. If the gear type is chosen by the Manufacturer, satisfaction is not guaranteed, as shown in the Fig. 1.2.

![Figure 1.2: Manufacturer (left) and Client (right)](#)

¹We consider FODA, Feature-RSEB, and Cardinality-based feature diagrams as classical.
Again, we need to extend the classical notion of feature diagrams, such that they can mark, for variable features, if the choice will be made by the requestor or the provider. For example, the gear type can be selected by the requestor whereas the provider may have a choice of selecting the company of the gear for the Car. Such a fine-grained distinction is not possible in classical feature diagrams.

1.2 Problem Statement

For the matching of feature diagrams in the context of service-oriented architecture we need to

1. Extend classical feature diagrams to be able to differentiate the features that can only be used once from those that can be used multiple times, and variable features that are chosen by requestors from those that are left with the provider

2. Formalize the notion of matching of feature diagrams, i.e., when the requirements are satisfied.

1.3 Solution

We propose service feature diagrams with semantics based on Linear Logic [Gir87], an extension of classical features diagrams able to provide the flexibility discussed above. Service feature diagrams provide the following functionalities.

Feature Types: Service feature diagrams can differentiate features that must be used only once, which we call resource features, from those that may be used multiple times, which we call shareable features. Shareable features are modelled by rectangular boxes with grey background whereas a rectangle represents a resource feature. We use Linear Logic to encode these types.

Selection Rights of Variable Features: Service feature diagrams can distinguish variable features selected by the requestor from those which are left with the
provider to choose. Again, Linear Logic supports the encoding of these types of features.

Instance Diagrams: Traditionally, a permissible selection of features is called an instance [KCH+90, KKL+98, CE00, BSC10]. In the case of service feature diagrams, the traditional notion of instance needs to be extended to allow for the difference of resource from shareable features and requestor’s from provider’s choice of variable features.

An instance diagram consists of a permissible selection of features made by the requestor, while preserving the variability captured for the provider by the service feature diagram. That means, first the requestor has to choose one from the available instance diagrams of a service feature diagram followed by the provider from the features which are left for provider to choose from.

Instance formulas show the encoding of instance diagrams. The linear encoding of a service feature diagram derives the instance formulas of a service feature diagram.

Matching of Feature Diagrams: We provide graphical rules to match service feature diagrams. Each rule consists of two feature diagrams, one showing the precondition and other the effects of the matching rule. If the pre-condition of a graphical rule exists in the provider feature diagram then we apply that rule by replacing its pre-condition by the effects in provider feature diagram.

A rule is applied over a provider feature diagram and after the application of series of matching rules, if the provider feature diagram is transformed to show a requestor features diagram, we say that the provider satisfies the requirements.

Formal Support: We use Linear Logic as a formalism for service feature diagrams. For example, in Section 5.1 we provide the rules to encode the feature types of a service feature diagrams into Linear Logic. In Section 5.3, we present encoding rules of instance diagrams into Linear Logic. We also show that the encoding
of service feature diagram derives the encoding of its instance diagrams by using LLProver [Tam95]–an online tool capable of generating proofs by using inference system of Linear Logic.

To provide the formal support for graphical rules, we show their correctness using the inference system of Linear Logic. We also discuss that the application of matching rule over a provider feature diagram does not depend on the context.

It is worth mentioning that the use of service feature diagrams is not limited to the services domain. In classical feature diagrams, all features are treated as shareable [NH11], whereas requestor’s choice is used for the variable features [WZ10]. So, service feature diagrams not only add functionality to classical feature diagrams but, support them as well. For instance, a classical feature diagram can be remodelled using the notations of service feature diagrams, if shareable is used as a feature type and requestor’s choice for the variable features.

1.3.1 Contributions

We have presented the following contributions in the thesis:

1. Introduction of service feature diagrams

2. Semantics of service feature diagrams in Linear Logic

3. Graphical rules covering all cases for matching and merging service feature diagrams

4. Formalisation and verification of matching

We have published a paper in a Workshop at the International Conference on Software Product Lines 2011:

- Muhammad Naeem, Reiko Heckel. Towards Matching of Service Feature Models based on Linear Logic. In proceedings of 1st Workshop on Services, Clouds,

In the above paper, we have discussed the encoding and matching of basic version of Service Feature Diagrams, with only requestor’s choice and without any graphical matching rules.

We have also published the following papers which are not directly relevant to service feature diagrams, but motivated us in their development:


1.4 Thesis Outline

We structure the thesis as follows: In Chapter 2, we discuss background information which includes

- classical feature diagrams, where we discuss FODA, Feature-RSEB and Cardinality-based Feature Diagrams

- instance diagrams and encoding of classical feature diagrams in propositional logic
– basic information about service discovery and matching in the context of service-oriented architecture

Chapter 3 highlights the approaches that are related to ours, and also discusses their differences with our approach. We categorise the discussion about related work into:

– matching and merging of models: We discuss matching and merging of models at general level.

– matching of feature diagrams: We consider the approaches for the matching of feature diagrams.

– composing feature diagrams: We discuss different operations which are used to compose feature diagrams.

– feature diagrams and first order logic: We discuss the approaches connecting feature diagrams and first order logic.

– algebra of features, we discuss the algebra exists in the literature for feature diagrams.

– communicating sequential processes (CSP): We discuss the basics of CSP and state their possible use for capturing the semantics of service feature diagrams.

– feature diagrams and services: We discuss the use of feature diagrams in services specification.

– linear logic and services: We discuss the connection of Linear Logic and services.

From Chapter 4 we start presenting the contributions of the thesis, where we introduce service feature diagrams and discuss

– the diagrammatic representation of resource and shareable features
– the diagrammatic representation of variable features to be selected by requestor and provider

– a metamodel of service feature diagrams, which ensures the confirmation of requirements discussed so far

– visual rules to generate instance diagrams of a service feature diagram

In Chapter 5, we move further by providing the semantics of service feature diagrams, where we

– discuss that propositional logic is not expressive enough to capture the semantics of service feature diagrams.

– propose to use Linear Logic as formal semantics for service feature diagrams, it respect their requirements.

– encode instance diagrams as linear formulas and show that the linear formula of a service feature diagram derives the encoding of its instance diagrams.

In Chapter 6, we

– provide the graphical rules that are used to match feature diagrams.

– present rules to merge service feature diagrams.

– show the analysis of the graphical matching rules

– show application of graphical rules over a running example

In Chapter 7, we provide the proofs of the rules presented in Chapter 6, by encoding the feature diagrams of each rule in Linear Logic. We

– explain our approach for providing the correctness of graphical rules based on the inference system of Linear Logic

– show that the application of graphical rules over a service feature diagram is independent of the context.
Chapter 8 consists of the evaluation of our approach. We show how many of our claims have been achieved by using service feature diagrams. In Chapter 9, we discuss our future work, where we discuss aligning of service feature diagrams with our previous work about visual contracts published in [NH09, NHO09, NHOH10] and our plan to develop tool support for service feature diagrams.

1.4.1 How to read thesis

The readers who want to know the basics of feature diagrams are advised to start reading from Chapter 2. Contributions of the thesis are presented in Chapters 4 to 7. Chapters 4 and 6 are self contained, so readers who are not interested to go through the logics, can skip Chapters 5 and 7. We have shown it in the form of flowchart in Fig. 1.3.

Figure 1.3: How to Read Thesis
Chapter 2

Background

In this chapter, we provide the background information that is required to understand the technical contribution. In Section 2.1, we discuss feature diagrams an understanding of which will be required throughout the thesis. In Section 2.2, we discuss Linear Logic and its inference system, which will be required in Chapter 5 and 7. Section 2.3 gives information about service specification and matching, which will be useful to understand the contents of Chapter 9, whereas Section 2.4 concludes the chapter.

2.1 Classical Feature Diagrams

Feature diagrams are used to model the commonalities and variabilities of a product line in a hierarchical form [KCH+90]. A feature is an increment to the product functionality. For example, engine type, gear type, or seat type can be used as features in a feature diagram of a Car Manufacturer.

A feature diagram represents all combinations of features of a product line in a single diagram. A permissible selection of features of a feature diagram is called an instance [KCH+90, CE00]. A feature diagram consists of the following set of constraints:

Parent-child: These constraints are caused due to the vertical connection of features with its subfeatures in a feature diagram, i.e., by the parent-child relationship.
For example, a feature $B$ is a subfeature of feature $A$ in Fig. 2.1.

Cross-tree: These constraints are caused by presence of horizontal dependence of features in a feature diagram. Cross-tree constraints are of two types:

Requires: This constraint imposes the selection of a target feature if its source is selected, in an instance. A requires constraint is represented by a dashed arrow starting from the source and heading towards the target feature.

Excludes: This constraint imposes the non-selection of source and target in an instance. An excludes constraint is represented by a double headed dashed arrow between source and target.

Feature diagrams were proposed in 1990 by Kang et al [KCH+90], which are called as FODA (Feature Oriented Domain Analysis). There is no consensus on the standard notation of feature diagrams. Multiple authors have proposed extensions to the original feature diagrams. For example, Feature-RSEB was the first extension of FODA by Griss et al in 1998 [GFd98], where they have added a new parent-child relationship. Finally, Czarnecki et al [CHE05a] have proposed an extension to the basic feature models where they have introduced cardinalities in the parent-child relationship of feature diagrams. Let us now discuss the notations used by these types of feature diagrams.

### 2.1.1 FODA Feature Diagrams

FODA defines three basic categories of parent-child constraints:

**Mandatory** If a feature is selected in an instance, then its mandatory subfeature must also be selected. A mandatory relationship is represented by a filled circle, as shown below, where $B$ is a mandatory subfeature of $A$.

For example, it is mandatory to have the Front Seat, if Seating Capacity from a supplier shown in the Fig. 2.5.
Optional If a feature is selected then its optional subfeature may be selected or rejected in an instance. An optional relationship is represented by an unfilled circle, as shown below. In the figure below, $B$ is an optional subfeature of $A$.

Alternative-group If a feature is selected then exactly one subfeature from its Alternative-group must be selected in an instance. It is represented by an arc with the label ‘choose 1’, as shown below. In the figure below, features $B_1, \ldots, B_n$ forms an Alternative-group of the feature $A$.

For example, the Car Manufacturer of Fig. 2.5 allows either of with or without Headrest for Rear seats for a Car.

In addition to the above mentioned parent-child constraints, FODA also allows the cross-tree constraints mentioned above.
2.1.2 Feature-RSEB

In 1998, Griss et al [GFd98] presented an extension of FODA feature diagrams, usually referred as Feature-RSEB. Although they presented a new graphical notation for the feature types, the semantics of the diagrams were very similar to FODA. Feature-RSEB, while giving full support to FODA, provided one more parent-child relationship:

**Or-group** If a feature is selected then at least one subfeature from its Or-group must be selected in an instance. This is represented by a filled arc with the label ‘1 +’. In the figure below, features $B_1, \ldots, B_n$ form an Or-group of the feature $A$.

![Figure 2.4: An Or-group](image)

For example, the supplier of Fig. 2.5 allows the selection of either Liver-based or Power-seat for Front seats of a Car.

Feature-RSEB also uses a composed-of group, which is formed by the combination of several individual and/or group features. We will refer to this group as an And-group. For example, the SeatingCapacity forms an And-group of Front and Rear seats in Fig. 2.5.

To present the concepts discussed so far in the form of a feature diagram, let us discuss the running example which we will use throughout the thesis. An architectural diagram of the running example is already shown in Fig. 1.1. Let us consider a requestor, interested to get a Car with specific gears and engine type, seating and entertainment system in the Car. This requestor finds a Car Manufacturer who can fulfill the needs of the requestor. This Car Manufacturer gets some parts for the Car from the external suppliers, like seats and entertainment system. For example, Fig. 2.5 shows a feature diagram of the supplier providing seating to the Car.
Manufacturer by using the notations proposed by FODA and Feature-RSEB.

![Figure 2.5: A Feature Diagram showing Seats of Car]

The above diagram states that each request of SeatingCapacity must require Front, whereas choice of either LiverBased or PowerSeat can be made for the Front seat. Rear seat is an optional feature so, it may be asked in a request, but if it is required then Rear seats can be provided either of headrest or without but not both in a single request.

### 2.1.3 Cardinality-based Feature Diagrams

Metthias et al in [RBSP02] and [RSP03] proposed an extension of feature diagrams using UML multiplicities, in group features and keeping the solitary features of FODA. Feature diagrams with multiplicities generalise the group of feature proposed in FODA by using $[m..n]$ multiplicity, as shown below.

![Figure 2.6: A Group with Set Cardinality]

If the parent of a group is included in an instance, then the number of features included depends on the multiplicity specified. For example, the $[1..1]$ multiplicity is used for an Alternative-group and $[1..n]$ is used for an Or-group, where $n$ is the maximum number of features in the Or-group.

The set cardinality can start from an arbitrary number. Feature models with multiplicities enable us to model cases not possible before. For example, consider a
case where at least two features are required to be selected, if their parent is chosen.

The feature models with multiplicities only support group features, these can not be used in the scenarios where an individual feature requires multiplicity. For example, consider a feature that needs some multiplicity is the parent of a subtree containing variable features. So in [CBUE02], Czarnecki et al proposed cardinality-based feature diagrams. Cardinality-based feature diagrams not only support the use of cardinalities in the group features but also for individual features. Later, Czarnecki et al [CK05, KC05] provided the semantics of cardinality-based feature diagrams.

The main difference between cardinality-based feature diagrams and earlier versions is the use of clonable features with [m..n] cardinality, as shown below. For example, an optional relationship is equivalent to a clonable feature with cardinality [0..1] and a mandatory relationship has [1..1] cardinality. The range of clonable features can also start from an arbitrary number.

The cardinality of a clonable feature states the number of repetitions of a feature in an instance along with its parent. For example, a clonable feature having [2..4] cardinality means that the parent of a clonable feature has an And-group, where this feature will be added 2 to 4 times, with subscripts differentiating the feature names. If a feature B being a subfeature of a feature A gets [2..3] cardinality, then feature A either has an And-group of B₁ and B₂ or an And-group of B₁, B₂, and B₃. The repetition of a clonable feature also causes repetition of its subfeatures.

In Fig. 2.8, we have re-modelled the Car Manufacturer of Fig. 2.5 using the notation of cardinality-based feature diagrams. Please note that group and solitary features are shown using the range of features that can be selected along with their parent.
The number of instances of a feature diagram depends on the number of variable features it contains. A variable feature is a feature that may not exist in every instance of a feature diagram, whereas a core feature is part of each instance [BSC10]. For example, the feature diagram shown in Fig. 2.5 provides 6 instances, where \{SC, F, LB, R, wH \}^{2} is one of them.

The more instances a feature diagram has, the greater is its flexibility. The most flexible feature diagram is the one which only contains optional features. If we consider only one type of features in a feature diagram, then the order of flexibility from high to low is optional, Or-group, Alternative-group, and mandatory [BSC10].

### 2.1.4 Interpretation

The notion of instance can be formalised by interpreting feature diagrams in logic. Jonge and Vesser were the first to connect feature diagrams and grammars [dJV02]. Mannion provided a mapping in propositional formulas [Man02]. Batory in [Bat05] has summarised the work to date about the conversion of feature diagrams in propositional formulas and provided a connection between feature diagrams, grammars and propositional logic. Furthermore, Batory proposed to use Logic Truth Maintenance System for the analysis of propositional formulas based on a SAT solver [FdK93].

Czarnecki et al have provided an algorithm to extract a propositional formula from a feature diagram and vice versa [CW07]. While following their work, they

---

For brevity, we use underlined characters while referring to the features in the instances, and later in the formulas.
have re-engineered the feature diagram from a propositional formula in [SLB+11]. Benavides et al in [BSC10] have provided a comprehensive survey about the interpretation of feature diagrams.

Table 2.1: Encoding Features in Propositional Logic

<table>
<thead>
<tr>
<th>Features</th>
<th>Feature Representation</th>
<th>Propositional Formulas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mandatory</td>
<td>$x \leftrightarrow y$</td>
<td>$x \leftrightarrow y$</td>
</tr>
<tr>
<td>Optional</td>
<td>$y \rightarrow x$</td>
<td>$y \rightarrow x$</td>
</tr>
<tr>
<td>Alternative</td>
<td>$y_1 \leftrightarrow (\neg y_2 \land \cdots \land \neg y_n \land x) \land \neg y_1 \land \cdots \land \neg y_n \land x \land \cdots \land \neg y_n \land x)$</td>
<td></td>
</tr>
<tr>
<td>Or</td>
<td>$x \leftrightarrow (y_1 \lor y_2 \lor \cdots \lor y_n)$</td>
<td></td>
</tr>
<tr>
<td>Implies</td>
<td>$x \rightarrow y$</td>
<td>$x \rightarrow y$</td>
</tr>
<tr>
<td>Exclude</td>
<td>$\neg (x \land y)$</td>
<td>$\neg (x \land y)$</td>
</tr>
</tbody>
</table>

In our work, we follow the notation of Czarnecki et al [CE00]. A propositional formula consists of a set of variables and a set of logical connectives constraining the values of the variables, e.g. $\land$, $\lor$, $\neg$, $\rightarrow$. A variable in a propositional formula refers to a feature of a feature diagram. Table 2.1 states the encodings of basic feature types (mandatory, optional, Alternative- and Or-group) into propositional formulas.

For example, the propositional formula of the feature diagram in Fig. 2.5 is $(SC \leftrightarrow F) \land (F \leftrightarrow (LB \lor PS)) \land (R \rightarrow SC) \land (H \leftrightarrow (\neg wH \land R) \land wH \leftrightarrow (\neg H \land R))$. Valuations for this formula giving $true$ define the set of all valid instances. For example, $SC \land F \land LB \land R \land H \land \neg PS \land \neg wH$ is a valid instance of the feature diagram in Fig. 2.5.
2.2 Linear Logic

Linear Logic was introduced by Jean-Yves Girard in 1987 [Gir87]. Multiple occurrences of a proposition in a linear logical formula are distinguishable from a single occurrence, i.e., $A \otimes A \neq A$, where $A$ is a proposition in Linear Logic. This does not apply in classical logic.

Consider an example (inspired by [Kan00]) where a proposition $P$ represents one pound and propositions $Choco$ and $Snk$ refer to a chocolate and a pack of snacks, respectively. In Propositional Logic, $P \vdash Snk$ and $P \vdash Choco$ mean that one pound can be used to get one pack of snacks and a chocolate, respectively. Using the inference rules of Propositional Logic, we may be misled into believing that one pound can be used to get both a chocolate and a pack of snacks:

Furthermore, iterative use of the above deduction leads us to believe that with one pound we can get as many packs of snacks and chocolates as we want, which is the consequence of idempotence in Propositional Logic. One can avoid this problem by using more sophisticated encodings, although such encodings suffer from the frame problem [Sha97, Sha00, Sha09]. However, limiting the use of weakening and contraction to the propositions having modalities allows Linear Logic to avoid such kind of reasoning by using very simple rules. Let us discuss the concepts and notations of Linear Logic.

2.2.1 Concepts and Notations

There are three main categories of Linear Logical connectives: Multiplicative connectives, additive connectives, and exponential connectives [CM10, Gir87, Gir95, Tro92]. Multiple fragments of Linear Logic can be formed due to the combination of these connectives:
2.2 Linear Logic

**MALL:** MALL involves multiplicative and additive connectives.

**MLL:** MLL is MALL without additive connectives.

**MELL:** MELL involves multiplicative and exponential connectives. It is Linear Logic without additive connectives.

Let us introduce the concepts we will be using in the thesis. We use two linear propositions $A$, $B$ which refer to features $A$ and $B$, respectively, of a feature diagram.

1. $\otimes$ is called Multiplicative Conjunction with 1 as an identity. A linear formula $A \otimes B$ shows the selection of both features $A$ and $B$.

2. $\&$ is called Additive Conjunction with $\top$ as an identity. A linear formula $A \& B$ states the existence of an internal choice between $A$ and $B$.

3. $\oplus$ is called Additive Disjunction with 0 as an identity. A linear expression $A \oplus B$ also states the existence of an external choice between $A$ and $B$.

4. $\rightarrow$ is called Linear Implication. A linear expression $A \rightarrow B$ means that a feature $B$ can only be selected if we have already chosen the feature $A$. We use linear implication to impose the condition where we want to select a feature before the other feature. For example, a subfeature can only be selected if its parent is already chosen.

5. $!$ is called Storage Operator. It is used to copy a linear proposition. A linear expression $!A$ states the selection of a feature $A$ as many times as required.

In our case, to show the connection of feature diagrams and Linear Logic, we use Classical Linear Logic (CLL). We may require the connectives which are not provided by the subsets described above. For example, the encoding of a feature diagram with $A$ as root and $B$ and $C$ making an Alternative-group with provider’s choice, which we will explain in Chapter 5, is

$$A \otimes (A \otimes (A \rightarrow ((B \otimes C^\perp) \oplus (B^\perp \otimes C))))$$
Please note, we use three copies of the parent feature $A$ in the above formula. The reason behind an extra copy is the intuitive meaning of linear implication, i.e., $A \rightarrow B$ means after selecting a proposition $B$, a proposition $A$ gets consumed. In our case, features cannot be consumed so, we provide an extra copy of parent feature to preserve them in the instances.

In the above formula, we have used multiplicative and additive connectives of Linear Logic, in the cases where we encode shareable features—use of shareable features will be explained in Chapter 5—we have to use of-course modality (!). So, we use Classical Linear Logic to map the feature diagrams with linear formulas.

### 2.2.2 Inference System

The basic component of the linear inference system is a sequent, written in Gentzen’s style [CM10]. A sequent is composed of two sequences of formulas separated by turnstile $\vdash$ (also read as yields or derives). If $\Gamma$ and $\Delta$ are the multisets of the finite sequences of formulas then

$$\Gamma \vdash \Delta$$

represents a sequent in Linear Logic which states that the multiplicative conjunction of the formulas inside $\Gamma$ derives the multiplicative disjunction of the formulas in $\Delta$ [LMSS92].

An inference rule of a sequent calculus is of the form

$$\text{Rule} \quad \frac{Hypothesis 1 \quad Hypothesis 2}{Conclusion}$$

where hypotheses and conclusion are represented in the form of sequents, while Rule states the name of inference rule applied to Hypothesis 1 and Hypothesis 2 to get to the Conclusion. The sequent calculus of Linear Logic supports two categories of inference rules: Unary—which consists of single hypothesis; Binary—which consists of two hypotheses [Gir87]. The inference rule shown above is binary because it contains two hypotheses.
The deduction system of CLL consists of the basic rule and introduction rules for the connectives described above [Gir87, Tro92, CM10]. We have only one basic rule, i.e., the identity rule:

\[
\frac{\text{id}}{A \vdash A}
\]

which states that a formula \( A \) can be derived from the assumption of a formula \( A \). Linear propositions can be moved from one side of a sequent to the other, as shown by the following rules:

\[
L(.): \frac{\Gamma \vdash A, \Delta}{\Gamma, A^\perp \vdash \Delta} \hspace{2cm} R(.): \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash A^\perp, \Delta}
\]

The multiplicative conjunction \( \otimes \) has two introduction rules. First for introducing \( \otimes \) on the left, second for introducing \( \otimes \) on the right of a sequent:

\[
L_{\otimes}: \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta} \hspace{2cm} R_{\otimes}: \frac{\Gamma \vdash A, \Delta, \Gamma' \vdash B, \Delta}{\Gamma, A \otimes B, \Delta}
\]

The additive conjunction \( \& \) and additive disjunction \( \oplus \) both have two introduction rules for the left and the right of a sequent, as shown below:

\[
L_{\&} \quad \frac{\Gamma, A \vdash \Delta}{\Gamma, \Gamma', A \oplus B \vdash \Delta} \quad \frac{\Gamma, B \vdash \Delta}{\Gamma, \Gamma', A \oplus B \vdash \Delta} \hspace{2cm} R_{\&} \quad \frac{\Gamma \vdash B, \Delta}{\Gamma, B_1 \& B_2 \vdash \Delta} \quad \frac{\Gamma \vdash C, \Delta}{\Gamma \vdash B \& C, \Delta}
\]

The linear implication \( \rightarrow \) also has two rules for introducing it on the left and right of the sequent:

\[
L_{\rightarrow} \quad \frac{\Gamma \vdash A, \Delta}{\Gamma, A \rightarrow B \vdash \Delta, \Delta'} \quad \frac{\Gamma \vdash A, \Delta}{\Gamma, A \rightarrow \Delta, \Delta'} \hspace{2cm} R_{\rightarrow} \quad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \rightarrow B, \Delta}
\]

In CLL, weakening and contraction rules are only allowed for the propositions having modalities.

\[
W! \quad \frac{\Gamma \vdash \Delta}{\Gamma, !B \vdash \Delta} \quad \frac{\Gamma, !B \vdash \Delta}{\Gamma, !B \vdash \Delta}
\]
The !-modality can be introduced on the left and the right side of a sequent, as shown by the following rules:

\[
\frac{D! \Gamma, B \vdash \Delta}{\Gamma, !B \vdash \Delta} \quad \frac{!\Gamma \vdash B}{!!\Gamma \vdash !B}
\]

For a multiset \( \Gamma = A_1, A_2, \ldots, A_n \), the notation \( !\Gamma \) represents the multiset \( !A_1, !A_2, \ldots, !A_n \). All the connectives of the Linear Logic satisfies the De-Morgan’s Laws of duality [Gir87]. Linear connectives also conforms to the property of distributivity of multiplicative over additives

\[
A \otimes (B \oplus C) = (A \otimes B) \oplus (A \otimes C)
\]

Idempotence in Linear Logic is only allowed with additives, i.e.,

\[
A \& A = A, A \oplus A = A
\]

2.3 Service Specification and Matching

Service-oriented Architecture (SOA) supports dynamic discovery and binding of services. A service is a modular application which can be described, published, invoked, and composed over a variety of networks [CGL+08, Erl07]. The core idea of SOA is to decouple functional units and expose them as independent services to other programs to increase reusability. Moreover, this allows to create new applications or services by combining already available ones in new ways.

SOA forms a triangle of three participants: service provider, discovery service, and service requestor, as shown in Fig. 2.9 [FK02]. A service provider offers services and descriptions. They may publish their service descriptions in a repository e.g., using semantic web languages, such as OWL (Web Ontology Language) [HP06]. A discovery service is a repository where service descriptions can be published and searched [New02, PKPS02, CHvRR04].
It is up to the provider of the service to decide which information should be published with the discovery service. A service requestor queries the discovery service to find required services. The published information is used to select an appropriate service from the discovery service, based on matching the query of a service requestor with the available service descriptions. Once the appropriate service is selected and a requestor agrees with the service-level agreement (SLA) then they start interacting with each other. The SLA is also referred as a contract between provider and requestor.

The main aim behind the use of SOA is automatic discovery and binding of services [FK02]. This continues to pose major challenges. One of them is the level of flexibility required to find a service to specific requirements. Any automated selection requires specification not only of the signatures and data types of the services, but also of their actions and protocols. A more detailed description will, however, be less likely to be matched by any existing service.

For example, a requestor while planning to arrange a trip to attend a conference may not only be interested in booking a flight and a hotel but also to get tickets for a play and to have dinner at a well-known restaurant. No travel agency takes care of all these activities. In a scenario where a single offer cannot satisfy all requirements, there are two solutions: To compromise on the requirements, e.g., by abandoning cultural activities, or use more than one provider, e.g., by booking directly with the restaurant.
Feature models provide flexibility in service specification and matching [dAKS+11]. Using feature models, one can identify the negotiable and compulsory elements of the models as optional and mandatory features, respectively. Feature diagrams can provide the flexibility in service specification.

2.4 Summary

In this chapter, we have provided information about classical feature diagrams. We have explained three earlier versions FODA, feature-RSEB and cardinality-based feature diagrams with the help of examples. We also showed the interpretation of FODA and Feature-RSEB feature diagrams in the form of Propositional Logic.

We moved on by giving information about Linear Logic and its inference systems. There is no attempt in the literature to connect feature diagrams with Linear Logic, so we have not mentioned this connection, which we will discuss in forthcoming chapters. At the end we discussed service-oriented architecture in the context of service specification and discovery.
Chapter 3

Related Work

In this chapter we discuss the techniques, addressing similar or related problems. To the best of our knowledge, the interpretation of feature diagrams in Linear Logic, which we motivated in [NH11], has not been explored elsewhere, one of the reasons being that the problem of matching required features with multiple providers has not deserved so much attention in the literature on software product lines.

We start by the matching and merging of models in Section 3.1, which is followed by the discussion of the approaches available for matching of feature diagrams in Section 3.2. We discuss the different operations available for the composition of feature diagram in Section 3.3, whereas Section 3.4 explains the connection of service feature diagrams and first order logic and stating how those approaches lack the support of service feature diagrams.

Section 3.5 discusses algebras for feature composition, whereas Section 3.6 briefly explains Communicating Sequential Processes which share with linear logic the distinction of internal and external choice and discuss their possible use for capturing the semantics of service feature diagrams. Sections 3.7 and 3.8 discuss the use of feature diagrams and Linear Logic, respectively in services. Section 3.9 concludes the chapter.
3.1 Matching and Merging of Models

In this section, we will discuss the matching and merging of general models. Traditional approaches for model matching consider them as trees and use dedicated algorithms to get the difference between those models. This difference is computed by matching the elements of one model with the elements of other. The algorithms used for the matching of models resides in one of the following categories:

Static-identity [AP03, FGC*06]: In this category the basic approach is to identify the matched elements of models based on id. It is assumed that each model element has a static identity.

Signature-based [RFG*05]: The matching of elements is done on the basis of their signatures, which are calculated dynamically by a model query language using the values of its features.

Similarity-based [KPP06b, Fou07, LGJ07, RV08, TBWK07]: This category treats models as typed attribute graphs and matching of elements is based on the aggregated similarity of their features.

Language-based [Fou07, KPP06b, TBWK07]: This category involves matching algorithms based on a particular modelling language.

Kolovos et al in [KRPP09] argued that none of the above categories provide satisfactory solution to model matching, but that the problem should be treated by deciding on the best trade-off within the constraints imposed in the context, and for the particular task at stake. Brand et al in [vdBPV10] provided an algorithm for matching of models, which is able to use any of the above four techniques on UML models.

There are some efforts in the literature for the composition/merging of models. Kolovos et al in [KPP06a] used the Epsilon Merging Language [KRPGD12] to automatically compose a model from two separate models, whereas Mens in [Men02] provided a survey on software merging.
Our approach is different in specifically targeting feature diagrams. Although, there are some approaches which map feature diagrams with UML models [CH06, CA05, CKK06, CAK+05, KAG+07, HKW08], but the generic solutions can not be used in our case because, our matching is based on the semantics of the models, while the generic algorithms only consider their structure. In particular, as service feature diagrams are newly introduced in this thesis, there is no existing solution for their matching and merging. As it is the first attempt to provide the semantics of service feature diagrams (cf. Chapter 4 and 5), there is a need for rules for its matching and merging, which we have provided in Chapter 6.

3.2 Matching of Feature Diagrams

Fahrenberg et al in [FLW11], provided an algorithm to get the semantic difference of two feature diagrams. For example, if $S_P$ and $S_R$ are the semantics of provider and requestor feature diagrams, their algorithm gives $D = S_P - S_R$, the clauses which exist in $S_P$ but not in $S_R$. It is related to our work in the sense that their algorithm can be used to check whether an offer satisfies the requirements by computing the difference $D$. Indeed this difference, if non-empty, can be used as remaining requirements to to match with the next available offer.

They use the Propositional Logic-based semantics of feature diagrams, where $S_P$ and $S_R$ are considered to be in conjunctive normal form (CNF) [FdK93]. The algorithm only works on propositional formulas, but service feature diagrams can not be interpreted into Propositional Logic (Please see Chapter 5 for details). So, the algorithm discussed in [FLW11] can not be used for service feature diagrams.

Eichberg et al in [EKMM10] have proposed to use feature models for the automated composition of components to capture the variability of component interfaces. They allow variability at the provides interface of a component, whereas a requires interface does not have any variability. To check whether the descriptions at the provides interface satisfy the requirements at a requires interface, they check the inclusion of FRS in FMI, where FRS (Feature Requirements Specification) repre-
sents the client’s requirements and FMI (Feature Model Instance) is the set of all the instances of the provider feature diagram. If FRS is contained in FMI, they proceed to bind the components. For the implementation of their approach they used a Java-based environment called Columbus.

Apart from the fact that their approach does not support service feature diagrams, our work can be considered as building on theirs. In future work, the authors suggest to use a shared feature model that can be used by multiple requests at a time. For example, if a service describes cryptographic protocols once and is referenced by multiple services requiring cryptography, it should be shared among them. In our approach, shareable features in service feature diagrams can be used by multiple requestor service feature diagrams.

In [NH09], we have used feature models as requirements and descriptions to provide flexibility in matching services specified by graph transformation systems. We map the features of both feature models with the underlying visual contracts. To get the agreeable features of requestor and provider we compute the intersection of their feature diagrams based on their semantics in Propositional Logic. Once, we get the non-empty intersection, we proceed further by generating a variant of the underlying model.

Although the approach presented in [NH09] provides flexible matching of services, it could not address the two problems: Firstly, the approach does not work for matching against multiple providers, which we can address using service feature diagrams. Secondly, use of propositional semantics with feature diagrams causes the same problems discussed in Chapters 4 and 5.

3.3 Composing Feature Diagrams

Ancher et al [ACLF09, ACLF10] and Broek et al [vdBGN10] propose operations for merging feature models, but their emphasis is on composing, not matching feature models. As a consequence, they can follow the interpretation of feature models in Propositional Logic, which we have shown in Chapter 4 not to be applicable to
matching (see also [NH11]). The same applies to Rosenmüller et al. [RS10], who propose to combine different product lines into a single, virtual variability model, which they call *multiple product lines*. Again, they emphasize the composition of different software product lines and require a central point of control (the integration model).

Elçin et al in [AODK11] and Mannion et al in [MSA09] provided rules to merge different views of a large feature diagram. They provided rules to resolve conflicts between features appearing in different views. Features occurring in the view of an engineering perspective can have a different relevance than the same features occurring in a marketing view of a feature diagram of that organization. For example, in a home security service, a marketing view can have room surveillance and admittance control as mandatory, but intrusion detection as optional features, but these three features can be considered as mandatory in the engineering view. Their rules can be used to merge multiple views into one large feature diagram of a security system, for instance.

Their work overlaps with ours in the sense that we also merge service feature diagrams if more than one offer contributes towards a requestor’s goal. They use instances to get a permissible selection of feature and then use rules to detect the conflicting features. However their approach cannot be used in the case of service feature diagrams, because the classical notion of instance does not meet the semantics of service feature diagrams (please see Chapter 4 for details). Secondly, our merging rules are used at the level of service feature diagrams, unlike [AODK11, MSA09] where they use local views of their feature diagram.

Czarnecki et al [CHE05a] have used cardinalities in feature diagrams. The interpretation in Propositional Logic uses indexed propositions $a_1 \land \cdots \land a_n$ to represent multiple occurrence of a feature $a$. Apart from the fact that this use of names to encode multiplicity is not reflected in the semantics and calculus of Propositional Logic, it would not solve our problem: the use of such *clonable* features in our scenario does not guarantee that a requestor is given exactly one CD Player. For
example, if we use cardinality 1..1 for all parent-child relations, which is the closest we can get to considering them as resources, the feature diagrams in Fig. 4.1(b), (c) would still satisfy the requestor goal in Fig. 4.1(a). Again, the problem here is matching multiple providers with overlapping offers.

Sergio et al in [SBRCT08] have provided a technique for automated merging of feature models using graph transformation systems. Their work is partly inspired from [AGM+06]. They provide a catalogue of rules for the merging of feature models. The correctness is checked by a feature modeling tool suite, FAMA [TBC+08]. They used AGG (a tool for graph transformation systems) [Tae99] as automated support for the merging of feature models. They have provided the catalogue of rules to merge the feature diagrams. Their work can not be used in context of service feature diagrams, because the FAMA framework only supports classical feature diagrams.

In [CZZM05], Chen et al have presented a semi-automatic approach for composing feature models based on requirements clustering. They provided an algorithm to compose an application feature model from clustering related individual functional requirements of applications into features. They label the variable features in a domain feature model obtained by merging the application feature models.

Their approach differs from ours in many aspects. The algorithm they proposed only works for classical feature diagrams, i.e., does not support service feature diagrams. Their merging algorithm expects all application feature models to share a common root, while we only require one diagram to have a common feature as root.

There are more tools and techniques available that compose/merge feature diagrams, such as [AC04, ACLF11, AKL09, Beu11, BSR03, BSR04, KTS+09, LAMS05, MBC09, TBKC07, TBC+08]. All available approaches/tools use classical feature diagrams, hence do not provide support for service feature diagrams.

3.4 Feature Diagrams and First-order Logic

First-order Logic is the generalization of classical logic which allows more compact and flexible representation of knowledge [Fit96, Smu95]. First-order Logic consists...
3.4 Feature Diagrams and First-order Logic

of logical and non-logical connectives [Fit96]. Logical connectives may include:

- quantifiers like ∀ and ∃
- logical connectives like ∨, ∧, ⇒, ⇔ etc.

Non-logical connectives may include operation symbols consisting of an operation name followed by parenthesis. For example, if \( st \) represents a student then

\[
\text{Pass}(st, \text{APG}) \land \text{Deposit}(st, \text{fee}) \Rightarrow \text{Promote}(st, \text{APG}, \text{PhD})
\]

captures the fact that if student passes APG examination and deposits her/his fee, then s/he is promoted from APG to PhD student.

One can use non-logical connectives to capture some of the requirements of service feature diagrams. For example, let \( \text{choiceR}(f_1) \) and \( \text{choiceP}(f_2) \) state that the selection rights of the features \( f_1 \) and \( f_2 \) are left with the requestor and provider, respectively. By using the interpretation on hand, the following service feature diagram shown in the left is encoded into first-order logic and shown on the right:

\[
a \Leftarrow \text{choiceR}(b) \land a \Leftarrow \text{choiceP}(c)
\]

In above encoding \( \text{choiceR}(b) \) and \( \text{choiceP}(c) \) only hold if the choice of selecting from the features \( b \) and \( c \) is made by the requestor and provider, respectively. The relation \( a \Leftarrow \text{choiceR}(b) \) holds either if \( a \) is true and \( \text{choiceR}(b) \) is false or if both of \( a \) and \( \text{choiceR}(b) \) are true, i.e., feature \( a \) must be selected before requestor uses their selection rights for the feature \( b \).

Since First-order Logic supports the connectives used by classical logic the presence of idempotence on a proposition in first-order logic faces the matching problems when multiple providers contribute towards a requestor goal with overlapping offers (cf. Section 4.1.1 for detailed discussion). On the other hand, Linear Logic restricts the idempotence to the propositions having modalities.
There are some efforts in the literature of software product lines for connecting feature diagrams with first order logic. For example, Elflaki used first order logic to automate the validation of feature models in [Elf10, Elf12], extending their previous work in [EPAH09]. Like we do, they distinguish diagrammatic and logic representation.

Although, main() and min() operations used by Elflaki can control the minimum and maximum number of features selectable from a Group (cf. Section 4.2), these can not be used for Solitary features (cf. Section 4.2). Hence, their approach is not able to differentiate shareable features from resource like ones. Furthermore, the work in [EPAH09, Elf10, Elf12] is not able to mark the selector of variable feature of a service feature diagram. So, their approach does not support the semantics of service feature diagrams.

### 3.5 Algebra of Features

Höfner et al in [HKM11] have developed an algebra for expressing software and hardware variability in the form of features, extending their own work in [HKM06, HM09]. As in our work, they allow multiple existence of a feature (shareable features, in our case) in an instance using bags (multisets). They use PFS and PFB to capture the notion of instance, where $P$ is a possible product (instance in our case), $F$ shows the feature and $S$ and $B$ represent a set and a bag, respectively. Since they used FODA feature models that have no distinction between resource and shareable features, but their approach support the resource-like and a shareable features by the bag-based products of $F$ (PFB).

In contrast to our work, the step-wise transformation of feature diagrams does not keep the structure. The algebraic encoding of a feature diagram only contains root and leaf features. We preserve the structure of service feature diagrams at the level of linear formulas. Secondly their approach can not be used to encode service feature diagrams because they have not distinguished the variable features selected by the requestor and the provider.
Apel and Hutchins have provided a language-independent calculus which they call gDEEP for feature composition [AH10], an extended version of [AH07]. This calculus can be used to compose and validate feature diagrams. Their work differs from ours in the sense that their calculus does not distinguish variable features selected by the requestor from those which are to be selected by the provider.

In [ALMK08, ALMK10], the authors present an algebra for Feature-Oriented Software Development (FOSD). As a part of their approach, they introduce Feature Structure Trees (FST) as a mechanism to organize features hierarchically in composed feature diagram. The authors present a procedure for composing features based on merging FST using tree superimposition. Their algorithm for tree superimposition works by starting from the root and proceeding down recursively. A node is composed from two nodes if they are the same and their parents are already composed. If the children of composed node are different then they are added as separate nodes in the superimposed tree (resulting feature tree). This recursion continues till the leaves of the tree.

As in our work, they assume that nodes with the same name refer to the same features. As they have used the classical feature diagrams, their work can not be used to merge service feature diagrams. For example, no rules are there to merge feature diagrams where the selection rights of a variable feature is given to the requestor whereas in the second diagram the provider wants to keep the choice. As shown in Chapter 7, our approach can handle these types of variable features.

Furthermore, their tree superimposition works at the level of features, they have not discussed the use of the same features at different hierarchical position in source feature trees. In our merging rules, we provide a more comprehensive list of merge-able cases. As discussed in Section 6.8, our approach can handle cases where a feature appears at different levels in source diagrams.
3.6 Communicating Sequential Processes

Communicating Sequential Processes (CSP) [SDBR84, Hoa85, Abr94] is a formal language to state the interaction in concurrent systems, which was first introduced by Chris A. R. Hoare in 1978 [Hoa78]. As its name suggests, CSP deals with communication of component processes of a system. CSP has a wide range of algebraic operators. Some of them which are closely related to our work are:

Prefix: The prefix operator consumes an event to produce a new process. For example, \( a \rightarrow P \) is the process which is willing to communicate \( a \) with its environment, and, after \( a \), behaves like the process \( P \).

Let us consider an example of a vending machine (VM) which breaks after giving \( choco \), if provided a \( coin \). The formulation of this VM can be shown as:

\[
VM = coin \rightarrow (choco \rightarrow STOP)
\]

Recursion: A repetitive behaviour of a Prefix is called recursion. Let us consider a formula of the same VM (described above) for its maximum design life.

\[
VM = coin \rightarrow (choco \rightarrow VM) \quad \text{[Original formulation]}
\]

\[
VM = coin \rightarrow (choco \rightarrow (coin \rightarrow (choco \rightarrow VM))) \quad \text{[Substitution]}
\]

\[
\vdots \quad \text{[In the same way]}
\]

Internal or Nondeterministic Choice: The nondeterministic choice operator allows the future evolution of a process to be defined as a choice between two component processes, but does not allow the environment any control over which one of the component processes will be selected.

For example, let us consider a VM that accepts a \( coin \) and gives an option of selecting either of \( choco \) or \( snacks \) to the customer. The formulation of this VM can be shown as:

\[
VM = coin \rightarrow ((choco \sqcap snack) \rightarrow STOP)
\]
External or Deterministic Choice: The deterministic (or external) choice operator allows the future evolution of a process to be defined as a choice between two component processes, and allows the environment to resolve the choice between the processes.

For example, let us consider the same VM with a different configuration, i.e., it allows gambling. VM accepts a coin and chooses either from choco or snacks depending on availability. The formulation of this VM can be shown as:

\[ VM = \text{coin} \rightarrow ((\text{choco} \mathbin{\square} \text{snack}) \rightarrow \text{STOP}) \]

In our case, we are interested to differentiate resource from shareable features and requestor’s choice from provider’s choice of variable features. Internal and external choice operators of CSP can be used to encode the requestor’s and the provider’s choice of variable features, respectively, whereas the recursive application of a Prefix operator can be used to encode multiple occurrences of a feature (a shareable feature in our case). Prefix forces us to give a temporal order of features, which is more than we want because we are just interested in a static match, not a process. As already stated, CSP deals with the communication of component processes which is more than what we require for service feature diagrams. On the other hand, Linear Logic provides connectives that completely suites the semantics of service feature diagrams.

### 3.7 Feature Diagrams and Services

A number of authors have explored the use of feature diagrams in the context of service-oriented architectures. Their focus has been on improving reuse in a feature-oriented product line-engineering [LMN08] based on feature analysis technique to identify services; on developing domain-specific services and optimising discovery based on customising preexisting results [WYF+08]; and on the overall methodology of variation-oriented engineering of services, accounting for variations in service
development, composition, and configuration [NPSB08].

Wada et al in [WSO07] and Fantinato et al in [FdSGdT07] propose to use non-functional attributes of services in feature modeling to provide variability in service specification. Robak and Franczyk in [RF02] have proposed using classical feature diagrams to model variability in web services. They claimed that the established notion of commonalities and the differences may speed-up the automated creation of web services. They have classified web services from the point of view of the requestor. The authors have not provided any formal support for the implementation of their approach.

Our approach is more specifically targeted at the matching of service providers’ descriptions against requestor’s requirements. Matching multiple descriptions, variability may arise from how they are combined rather than the individual services themselves. More generally, we provide a new semantic interpretation of feature modelling motivated by service matching, but not necessarily limited to this domain.

Writtern and Zirpins in [WZ10] have proposed to use value attributes in classical feature diagrams and provided the mapping with services, so they call their feature models service feature models. They propose to use constraint programming to interpret their diagrams for automated analysis. Although extended feature models [Ben07]–an extension of FODA feature models– support using attributes in classical feature diagrams, they have not been interpreted yet [BSC10]. There exists an interpretation of classical feature diagrams in constraint programming provided by Benavides et al in [BTC05], which does not support extended feature diagrams.

In contrast to our proposal, where we have provided the formal semantics of service feature diagrams in Linear Logic, they do not give any formal semantics for their service feature models. Furthermore, service feature diagrams provide more fine-grained specification of features, where one can distinguish between the variable features selected by the requestor and those which are to be chosen by the provider.
3.8 Linear Logic and Services

The use of Linear Logic in the context of services has been suggested in [RKM06], focussing on composition rather than matching. A proof in Intuitionistic Linear Logic is constructed deriving the global pre/post condition of the composition from pre/postconditions of its constituent services. The proof itself is taken as a template for the composition. The interpretation of a linear formula in this approach is computational, with linear implication representing state transitions brought about by the invocation of elementary services. In our case a formula represents the resources sought after or on offer and implication represents a step in the top-down instantiation of the corresponding feature diagram.

3.9 Summary

In this chapter, we provided a compilation of the most relevant feature modelling techniques reported in the literature and classified them into eight parts: Matching and merging of models, matching of feature diagrams, composing feature diagrams, feature diagrams and first order logic, algebras of features, Communicating Sequential Processes, feature diagrams and services, and Linear Logic and services. In addition, we discussed that no available technique based on classical feature diagrams meets the requirements of service feature diagrams.
Chapter 4

From Classical to Service Feature Diagrams

This chapter introduces service feature diagrams, an extended form of classical feature diagrams [CE00, KCH+90]. The motivation behind the implementation of web service is satisfaction of some goals [HHL05]. To check how much of the requirements are satisfied by an offer, one needs to match them [HC07]. In this chapter, we start with the challenges raised by matching feature diagrams in Section 4.1. In Section 4.2, we discuss proposed framework that can address the challenges. In Sections 4.2.1, 4.2.2, and 4.2.3 we discuss the concrete syntax, metamodel and the instance diagrams, respectively of service feature diagrams, whereas Section 4.3 summarizes how service feature diagrams meet the matchmaking challenges. In Section 4.4, we present the running example which we will be using throughout the thesis, whereas Section 4.5 concludes the chapter.

4.1 Matching Challenges

Traditionally, feature diagrams represent variability in software product lines in hierarchical form [KCH+90]. We use feature diagrams to represent the variability in services implemented by product lines. Using feature diagrams for specifying requirements and descriptions, we can take account of the variability allowed for
by providers and required by requestors when verifying that a collection of services jointly satisfy a set of requirements.

For example, a requestor while planning to arrange a trip to attend a conference, may not only be interested in booking a flight and a hotel but also to get tickets for a play and to have dinner at a well-known restaurant. To check, whether the available descriptions can satisfy the requirements, we need to match them [NHO09, NHOH10]. If a single travel agency is not able to satisfy the requirements, there are two solutions: To compromise on the requirements, e.g., by abandoning cultural activities, or use more than one provider, e.g., by booking directly with the restaurant. In the rest of this section, we discuss the challenges raised by matching services in the classical feature diagrams.

4.1.1 Resource vs. Shareable Features

One question that may arise when a requirement is to be satisfied by a combination of multiple offers is whether a specific feature can be satisfied more than once. The classical feature diagrams do not differentiate the features that should be satisfied once from those that can be satisfied more than once.

For example, consider a small part of the running example which is provided in Section 4.4, where a Car Manufacturer wants to put a CD Player and a Radio into the Entertainment system of a Car along with a Contract for unlimited downloads of music, as shown in Fig. 4.1(a). The Car Manufacturer of Fig. 4.1(a) finds two offers shown in Fig. 4.1(b) and 4.1(c), respectively. The first offer provides a CD Player

![Figure 4.1: Requirements and Offers for an Entertainment System of Car](image-url)
along with the Contract and the second offers Radio and CD Player. However, both specify their CD Player to be mandatory. Let us further assume that the provider wants to get one CD Player only for their Entertainment system.

Intuitively, combined effect of both the offers should not satisfy the requirements shown in Fig. 4.1(a), because both offers of the Fig. 4.1(b,c) expect the selection of CD Player—they marked it mandatory. But, the match between offers and requirements shown in Fig. 4.1 could mislead us into believing that the two offers jointly satisfy the requirements, resulting in booking CD Player twice—that contradicts with the requirements. This is because of the absence of distinction between shareable feature and a resource feature in classical feature diagrams. That means, a feature that is required only once may also be satisfied by more than one offer providing that feature.

For a meaningful notion of matching of multiple feature diagrams we have to distinguish such features from resource-like ones that can be used or satisfied only once. For example, if the CD Player in Fig. 4.1(a) is marked as a resource, it means that a combination of providers resulting in more than one CD Players can not satisfy the requirements.

Because the distinction between resource and shareable features does not exist at the level of feature diagrams, we propose an extension of feature diagrams in which one can differentiate between single and multiple instances of features.

### 4.1.2 Your Choice or Mine

With requestor and provider being independent entities, there are different ways to resolve alternatives. For example, if a customer looking for a room with sea view finds a hotel offering rooms with sea view or facing inland, the requestor is sure to be satisfied if they get to choose the type of room. If the rooms are assigned by the hotel based on availability, satisfaction is not guaranteed, as shown in Fig. 4.2.

More generally, we would like diagrams to declare, for each alternative set of features, if the choice will be made by the requestor or the provider. For example,
while rooms may be allocated based on availability by the hotel, the requestor could have the choice between a continental or English breakfast, thus combining both cases in a single diagram. Such a fine-grained distinction is not possible in classical feature diagrams.

While distinguishing requestor and providers choice in our case was motivated by the need for matching, this may also be a useful extension to classical feature diagrams. Those who do not care about such a fine-grained distinction may use requestor’s choice at global level of diagram. Using requestor’s choice at global level would mean that the variable features used in the diagram are selected by the requestor. Because the use of requestor’s choice for variable features in a diagram does not contradict the constraints captured by the metamodel of service feature diagrams, as shown in Fig. 4.6, requestor’s choice can be used in service feature diagrams at a global level. Furthermore, the provider’s choice of variable features can be used later in a diagram if required.

### 4.1.3 Keeping my Options

Consider the case of planning a dinner with friends at a restaurant. We may be interested in the option of vegetarian meals, but will let the guests choose for themselves once they have arrived. To model this, a variable feature in the requestor diagram should not be disregarded in the matching but, instead, be matched by a corresponding variable feature, i.e., a restaurant giving the choice of vegetarian meals to the requestor.

This is not possible if one takes the simpler approach where requirements are represented by instances and matching corresponds to checking if the instance is
permissible based on the feature diagram specifying the offered service, as we provided previously in [NH09]. Instead, for capturing requirements about variability we propose that both requestor and provider be represented by feature diagrams and that matching be done at the level of diagrams.

To address the above challenges again, we propose an extension of feature diagrams which we call as service feature diagrams.

4.2 Proposed Framework

Service feature diagrams provide greater flexibility than classical feature diagrams. Using service feature diagrams, one can mark explicitly which of the features are shareable and which are required to be used only once. Service feature diagrams also provide different choices of variable features, by allowing either requestor or provider to make the choice.

Before starting discussion of service feature diagrams let us first check whether the propositional logic based semantics of classical feature diagrams support the limitations discussed earlier. To see the support of service feature diagrams at Propositional Logic, let us recall the requirements described in the previous section.

**Resource vs. Shareable Features:** Availability of weakening and contraction rules for a proposition in the sequent calculus of Propositional Logic [FS76, PL98, Hun02, Gue10] makes each proposition idempotent. For example, \((A \lor A) \equiv (A \land A) \equiv A\). So, a formula in Propositional Logic can not distinguish a proposition representing a shareable feature from one which represents a resource feature.

The sequent calculus of Linear Logic [Gir87, Tro92] does not allow the weakening/contraction rules on every proposition in linear logical formulas, but only on propositions with modalities. For example, \((A \otimes A) \neq A\) but \(!A \otimes !A\) \(\equiv !A\). So, a proposition with a ! modality refers to a shareable feature and propositions without modalities encode resource features. In other words, Linear Logic meets this requirement of service feature diagrams.
Your Choice or Mine: Propositional Logic provides one type of disjunction (\(\lor\)) that is used to encode variable features of classical feature diagrams [FS76, PL98, Hun02, Gue10]. Not declaring who will choose from the alternate, we can not guarantee the satisfaction of requirements in all cases. For example, consider the feature diagrams for provider and requestor in Fig. 4.3.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{provider_requestor.png}
\caption{Provider (left) and Requestor (right)}
\end{figure}

If the provider chooses between \(B\) and \(C\), there is no guarantee that requirement \(B\) is satisfied. In terms of Propositional Logic, if the provider makes the choice, the offer is rejected because \((B \lor C) \Rightarrow B\) is not valid. However, if the requestor makes the choice, we should consider the implication to be satisfiable, which is the case for \(B \equiv \text{true}\) in the example, meaning that the offer can be accepted.

Keeping my Options: Let us re-visit the scenario where a requestor is looking for a restaurant providing vegetarian or non-vegetarian food, considering two different types of requests. In one request, the requestor is happy with the choice made by the restaurant owner (the provider) for them, whereas the second request wants to keep the choice between the vegetarian and non-vegetarian food with them, as shown in Fig. 4.4(b) and 4.4(c), respectively. These requests find a provider that only offers vegetarian food, as shown in Fig. 4.4(a).

\begin{figure}[h]
\centering
\includegraphics[width=0.7\textwidth]{restaurant_requests.png}
\caption{Service Feature Diagrams of Restaurant and Requests}
\end{figure}

Intuitively, the offer shown in Fig. 4.4(a) should be able to satisfy the require-
ments depicted in Fig. 4.4(b), whereas this offer should not be able to satisfy the demands shown in Fig. 4.4(c). Propositional Logic does not provide connectives to encode the two types of choice. That means, Propositional Logic can not be used to interpret service feature diagrams.

As already said, we would like diagrams to declare, for each alternative set of features, if the choice will be made by the requestor or the provider. Linear Logic provides additive disjunction ($\oplus$) and additive conjunction ($\&$), which can be used to encode both types of variable features proposed in service feature diagrams.

Let us now discuss the syntax of the service feature diagrams.

### 4.2.1 Concrete Syntax

Table 4.1 shows the syntax of basic feature types used in service feature diagrams. Feature names are shown under ‘Features’ in the second column and their graphical representation is shown under ‘Feature Representation’. The first two rows show the types of feature allowed in service feature diagrams, i.e., resource feature and shareable feature. The first column refers to the concepts of the metamodel. For example, fType is the type of the feature, which can either be a resource or a shareable feature, whereas a Solitary feature can either be mandatory or optional. Group refers to either an Alternative- or an Or-group.

The diamond symbol in Solitary and Group features has a variable behaviour. It will be replaced by a rectangle for a resource feature, or a rectangle with grey background for a shareable feature. For example, a service feature diagram with shareable feature $x$ as root and resource feature $y$ as mandatory subfeature is shown in Fig. 4.5. Please note that diamond symbols for features $x$ and $y$ have been replaced by boxes for the shareable and the resource features, respectively.

![Figure 4.5: A Sample Service Feature Diagram](image)
### Table 4.1: Concrete Syntax of Service Feature Diagrams

<table>
<thead>
<tr>
<th>Features</th>
<th>Feature Representation</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>fType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource</td>
<td>Resource</td>
<td>A feature that can be used only once.</td>
</tr>
<tr>
<td>Shareable</td>
<td>Shareable</td>
<td>A feature that can be used multiple times.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solitary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mandatory</td>
<td>Mandatory</td>
<td>Feature y must be selected if x is.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Feature y may be selected or rejected with x, depending on the requestor’s choice.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Feature y may be selected or rejected with x, depending on the provider’s choice.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alternative</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Exactly one feature from the group of y₁, . . . , yₙ must be selected with x based on the requestor’s preference.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Exactly one feature from the group of y₁, . . . , yₙ must be selected with x based on the provider’s preference.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>At least one feature from the group of y₁, . . . , yₙ must be selected with x based on the requestor’s preference.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At least one feature from the group of y₁, . . . , yₙ must be selected with x based on the provider’s preference.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Or</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constraint</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implies</td>
<td>Implies</td>
<td>Target feature y must be selected if the source feature x is.</td>
</tr>
<tr>
<td>Exclude</td>
<td>Exclude</td>
<td>Features x and y can not be selected in one instance.</td>
</tr>
</tbody>
</table>

It is to be noted that choices are made by either the requestor or the provider. For example, an optional feature can either be selected by the requestor or the provider, and the same is for Group features. At the level of service feature diagram, we use a dashed edge or a solid edge to specify selection rights of variable features with the provider or the requestor, respectively.

The last two rows consist of cross-tree constraints: 1) An `implies constraint` is represented by a dashed arrow starting from a source feature and heading towards
the target feature. If the source of implies constraint is selected then target must also be selected. 2) An exclude constraint is represented by double headed dashed arrow between a source and a target feature. If any one from these features is selected then other can not be selected in that configuration.

4.2.2 Metamodel

In order to make precise concepts of the service feature diagrams, we present their abstract syntax by means of the metamodel in Fig. 4.6. A ServiceFeatureDiagram must have at least one Feature. Root is a special type of Feature, which exists only once and appears at the top of a service feature diagram. A Feature of a ServiceFeatureDiagram must have a name and a type, specified by the string attribute name and a boolean attribute fType, whereas childOf attribute stores the name of parent of Feature. The childOf attribute will only be empty for Root of service feature diagram. The name attribute should be unique in a ServiceFeatureDiagram, whereas a boolean attribute fType is used to differentiate between resource and shareable features. A Feature is a resource/shareable for fType being false/true. Feature has one more boolean attribute core, which is used to judge a Feature which gets selected in every configuration of a ServiceFeatureDiagram.

The core attribute is always true for the Root or for the features satisfying the condition

\[
\{ \text{If rType=} \text{false } \& (\text{Feature.core=} \text{true } \text{where Feature.name=} \text{childOf}) \}\]

That means, a feature will be core if it is a Root or a mandatory subfeature of a core feature. Using a core feature in the implies constraint causes an unnecessary condition, because the target feature will also behave like a core feature. In the case of excludes constraint, using core features in source/target will create dead features. A dead feature is a feature which never gets selected in an instance [Ben07, Men09].

To force that the Boolean attributes must always be defined, we use a condition
{fType and core must not be undefined} ensuring that the fType and core must be defined. This constraint can be shown formally using OCL [CW02, RG98, KW00] as

\[
\{\text{not self.fType } \rightarrow \text{oclUndefined}()\}
\]

and the same type of encoding can be used to enforce the definition of a core attribute. One can also use subclasses to avoid the undefined Boolean attribute. For example, using resource and shareable as subclasses of Feature in the metamodel, but to keep the diagram simple we prefer to use OCL constraints in avoiding such situations.

Other types of Feature are Solitary and GroupFeature. Solitary refers to a single feature whereas, GroupFeature refers to the member of a Group. The gName attribute in Group stores the name of a group to which a GroupFeature belongs to. Cardinarity between Group and GroupFeature confirms that a Group in a Service-FeatureDiagram must contain at least two GroupFeature. The rType is a boolean attribute of Solitary which captures its relevance with its parent, i.e., rType being

![Diagram of the metamodel of a Service Feature Diagram](image-url)
true means that a Feature is optional, whereas \textit{false} makes a Feature mandatory.

An And-group is special in the sense that it is not captured by Group, but is formed by the combination of multiple Group and/or Solitary features. Let us use an example to show the diagrammatic of the concepts discussed so far, where a Car Manufacturer gets an Entertainment system for their Car from external Suppliers, models their requirements for the Entertainment system, as shown in the Fig. 4.7.

![Figure 4.7: Manufacturer’s Options for the Entertainment System of Car](image)

In the service feature diagram shown above, core = \textit{false}, rType = \textit{true}, fType = \textit{true}, and childOf = ‘Entertainment’ makes the Contract as an optional (R) subfeature of an Entertainment Feature. In the same way, the attributes for the other features also makes them members of Or-group (R) and Alternative-group (P) in Fig. 4.7.

In a ServiceFeatureDiagram, a variable feature (either Solitary or a Group) must have a connection with Selector. A boolean attribute sType of Selector captures the information about the one who selects the variable feature. For example, sType = \textit{false} for an Or-group (R) and sType = \textit{true} for both the Alternative-groups (P) shown in Fig. 4.7.

Mandatory features are required to be selected in an instance if their parent is \cite{BSC10, KCH+90}, so there is no need to declare the selector of mandatory features in service feature diagram. To ensure that only optional features from Solitary should have a selector type, we use a condition \{Disallow: If rType=\textit{false} \} for the connection of Solitary and Selector.

A ServiceFeatureDiagram may have cross-tree constraints captured by a Constraint element in the metamodel. These constraints are of two types: An implies
constraint, when selection of a feature requires the selection of another feature, and an exclude constraint, when selection of a feature requires the rejection of another feature. In classical feature diagrams, cross-tree constraints are required to be used with variable features [CE00, KCH+90], and all the features in classical feature diagrams are treated as shareable [NH11].

In our case, we also require the source and targets of the cross-tree constraints to be shareable. Because the features used in these constraints can be selected two times once due to the hierarchical constraints, e.g., if its parent is selected, secondly due to the cross-tree constraint, e.g., if its source is selected. We have to use the boolean attributes fType and core of Feature to ensure the requirement.

A condition \{Disallow: If core=True or fType=False\} ensures that Constraint and Feature can only be connected for non-core feature that are shareable, as well [FS03]. In other words, if the type of Feature is resource and if a Solitary feature is mandatory at the level of service feature diagram then a feature can neither be considered as a Source nor as Target in the \textit{implies/exclude} constraint. Constraint type is judged by the value of a boolean attribute cType of Constraint. Type of constraint is implies for cType being true otherwise exclude. Source and target of a Constraint should be a Feature, shown by the edges headed by Source and Target.

Let us now discuss the instances that can be generated from a service feature diagram.

4.2.3 Instance Diagrams

An instance diagram shows the diagrammatic representation of an instance of a service feature diagram. Traditionally, an instance is defined as a permissible subset of the set of features of a feature diagram [KCH+90, CE00, BSC10]. However, this definition is not enough to capture the notion of instance for service feature diagrams. First, variable features allow a distinction between requestor’s choice and provider’s choice in a service feature diagram. The permissible selection of features for the requestor is different from that of the provider. Requestor and provider being
independent entities, will not make the selection of features at the same time, but one after the other.

Instance diagrams of a service feature diagram are generated from the point of view of the requestor, while preserving the variability captured for the provider. For example, let us consider an instance diagram of the Fig. 4.7, where the requestor chooses a CD Player for the Entertainment system of the Car, as shown in Fig. 4.8. As the choice of the manufacturer of CD Player is left with the provider, the instance diagram keeps this variability.

We show the rules to generate the instance diagrams of a service feature diagram in Table 4.2. The first two columns state the names and the diagrammatic representation of features, whereas the third column under ‘Instance Diagrams’ shows the rules to get the instance diagrams of the corresponding feature diagram. Because the instance diagrams do not unfold the provider’s choice, the concrete choice of features is shown either for mandatory feature (as shown in 1st row) or for the variable features with requestor’s choice (as shown in 2nd, 3rd and 4th row).

Again, the diamond symbol used in the diagrams of Table 4.2 is a meta variable which represents either a shareable feature or a resource feature. For the Alternative- and Or-groups shown in the diagram \( n \geq 2, i \in s, s = \{ k : 1 \leq k \leq n \}, j \in s', \) and \( s' = \{ l : 1 \leq l \leq n \mid l \notin s \}. \) We may come across a situation where \( s' = \phi, \) then we only consider the set \( s \) for an instance diagram and ignore the \( s'. \) For example, for an instance diagram of an Or-group (R) with two subfeatures, i.e., \( n = 2 \) and if \( s = \{1, 2\} \) then we will be left with empty \( s'. \)

Application of the rules shown in Table 4.2 is done by replacing the subtree of
Table 4.2: Instance Diagrams of a Service Feature Diagram

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Feature Diagram</th>
<th>Instance Diagrams</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mandatory</strong></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Optional (R)</strong></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Alternative-group (R)</strong></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Or-group (R)</strong></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Optional (P)</strong></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Alternative-group (P)</strong></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Or-group (P)</strong></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

the 2\textsuperscript{nd} column by one of the instance diagrams shown in the 3\textsuperscript{rd} column. We start from the root and check the existing subtrees. Then we start replacing each subtree with its respective instance diagram shown in Table 4.2.

For example, the service feature diagram shown in Fig. 4.7 has 6 instance diagrams, as shown in Fig. 4.9. The instance diagrams shown in Fig. 4.9 depict the choices available for the requestor of this service. The requestor is happy with any choice made by the provider. It is reflected in the instance diagrams by keeping the variable features which are left to be chosen by the provider and are directly connected to the selected features.

For example, if the requestor decides to get the Radio for the Entertainment, then
4.3 Discussion

As discussed, service feature diagrams are not only able to distinguish the features that are required to be used only once from those which can be used once, but also the variable features which are to be chosen by the requestor from those which are left for the provider to choose from. Let us revisit the challenges raised by matching feature diagrams, and check how service feature diagrams address those challenges.

4.3.1 Resource vs. Shareable Features

A service feature diagram can differentiate between resource and shareable features. The metamodel of service feature diagrams provided in Fig. 4.6 enforces this constraint by the boolean attribute $fType$ of Feature. If its value is $true$, the feature is shareable, otherwise a resource. Hence, using service feature diagrams, one can declare explicitly which features can be used/selected multiple times and which are required to be used only once.

At the level of diagram, a resource feature is represented by rectangle, whereas
4.3 Discussion

A shareable feature is shown by a rectangle with a grey background, as shown in Table 4.1.

4.3.2 Your Choice or Mine

A service feature diagram provides two basic categories of variable features, selected by the requestor or the provider. Each variable feature, Solitary or Group, is required to reside in one of these categories. Connections of Selector element in the metamodel enforce this constraint, whereas the boolean attribute $sType$ of Selector stores information about the selector of a variable feature. So using service feature diagrams, one can identify features required to be selected by the requestor and the features that are left for the provider to choose from.

Variable features required to be selected by the requestor are represented by a solid line, whereas variable features with provider’s choice are represented by a dashed line, as shown in Table 4.1. So, one can use both types of choice in a single diagram, as shown in the service feature diagram of the requirements for an Entertainment System of a Car in Fig. 4.7.

4.3.3 Keeping my Options

Let us revisit the scenario explained in Section 4.1.3 of a customer looking for a restaurant providing vegetarian and non-vegetarian food. The idea is to let the guests choose their food, once they arrive. Only restaurants offering vegetarian and non-vegetarian food as an alternative choice for the requestor are able to satisfy this demand. As classical feature diagrams do not make the distinction of who will choose the variable features, a group of variable features can be satisfied by a feature diagram providing a subset of that group. For example, a feature diagram providing vegetarian food could satisfy the request, which is not consistent with our requirements.

We observe that this type of problem arises due to the absence of explicitly marking the selector of variable features. For example, if a customer while declaring
their requirements says, vegetarian and non-vegetarian foods are alternatives, but I want to make the selection, only those restaurants offering vegetarian and non-vegetarian food with requestor’s choice will be able to satisfy this demand.

The attribute $sType$ of Selector in the metamodel is used to store the information of who will make the selection of variable features.

### 4.4 Running Example

We have only shown the architecture view of the running example in the Introduction, where a Client interested to customise a Car by keeping the choice of variable features, is shown. Client finds a Car Manufacturer who can satisfy their demands. A Car Manufacturer gets some parts of the Car from the Suppliers. Here, we explain the requirements and offers in detail.

![Figure 4.10: Service Feature Diagram showing Client’s Requirements](image)

Let us consider a Client interested to get a Car with specific variability in the Gear Type, Engine Type, and Seating Capacity, as shown in Fig. 4.10. The Client marks Entertainment and Air Conditioning as optional features, but wants to keep the choice. The Client of Fig. 4.10 is also interested to get the Contract of unlimited downloads of music with the Entertainment system. The Client has modelled the Entertainment types in an Or-group, so they have marked the Contract as shareable.
because they may get the Contract with both the Entertainment types.

The Manufacturer offers a Car with specific Gear Type, Engine Type, and Body, as shown in Fig. 4.11. The Manufacturer leaves the selection of Gears, Body and Engine to the Client, i.e., the Client chooses from the Or-group of the Car. Selection rights of Gear Type and Engine Type are left with the Client as well. The Manufacturer marks Seating Capacity as mandatory subfeature of Body, i.e., the requestor of this service is required to choose the Seating Capacity if they select Body.

![Figure 4.11: Service Feature Diagram of a Car Manufacturer](image)

The Manufacturer expects from the Client to choose at least one from Radio and CD Player if they want to have an Entertainment system. The Manufacturer also offers a Contract for unlimited download of music with their Entertainment system. The Contract can be given with at least one of the Entertainment systems. The Contract can be shared between the Entertainment Types, Client pays for the Entertainment System and gets the free downloads for both of them. So the Contract is marked as a shareable feature.

The Manufacturer gets the Entertainment system and the Seating Capacity from
external suppliers. So, the Manufacturer in Fig. 4.12(a) models the requirements for an Entertainment system to be satisfied by the suppliers shown in Fig. 4.12(b) and 4.12(c), respectively.

![Figure 4.12: Requirements of Entertainment and Suppliers Providing Entertainment System](image)

The Car Manufacturer gets the Car Seats from Supplier3 of Fig. 4.13(b), Fig. 4.13(a) shows the requirements for Seating Capacity. The Manufacturer, while interacting with the suppliers may use the information which is kept between the Supplier and the Manufacturer, i.e., not shown to the Client. The requirements defined in the requires interface are partially embedded into the provides interface of the Manufacturer service, i.e., not all the features obtained from the Suppliers are repeated in the provides interface, as shown in Fig. 1.1.

![Figure 4.13: Requirements of Seating Capacity (left), Seating Capacity Provided by Supplier 3 (right)](image)

For example, the Manufacturer’s requirements for an Entertainment system says that Supplier1 has to provide an Entertainment system from any of the companies shown under the Entertainment types. That means, the Manufacturer will only accept the Radio made by either ALPINE(Rd) or SONY(Rd). Details about the
companies providing Entertainment systems are not required by the Client (please see Fig. 4.10). That means, the Client can accept any company, so the information is not to them. The same applies to the case of Supplier2 where the Manufacturer is interested to keep the choice of having a Rear seat with/without a Headrest.

4.5 Summary

In this chapter, we have discussed the challenges raised by the matching of classical feature diagrams. Then we provided a solution to those challenges by proposing an extension which we call service feature diagrams. We also supported our claims by the help of a metamodel of service feature diagrams. Then we have provided the rules to get the instance diagrams from a service feature diagram. At the end of the chapter, running example which we will use throughout the thesis is presented.

In the following Chapter, we will discuss why the current semantics of the classical feature diagrams do not meet the requirements of service feature diagrams. We provide the solution by proposing the semantics for service feature diagrams based on Linear Logic [Gir87, Tro92, LMSS92, Gir95, CM10].
Chapter 5

Semantics of Service Feature Diagrams

In this chapter, we check how much support can be provided to service feature diagrams by the traditional semantics based on Propositional Logic. We will also discuss why propositional logic is not enough for the service feature diagrams. We move on by providing the semantics of service feature diagrams in Linear Logic [Gir87, Tro92, Gir95], including a variant of the traditional notion of instance that we call instance formula.

The structure of the chapter is shown in the figure below, where $ID(fd)$, $IF(fd)$, and $LF(fd)$ are the set of instance diagrams, set of instance formulas and the linear formula of a service feature diagram $fd$, respectively.

The encoding of $fd$ into a linear formula $LF(fd)$ is discussed in Section 5.1, where we provide the rules to interpret service feature diagrams in Linear Logic. Section 5.2 discusses the set of instance formulas $IF(fd)$ of a service feature diagram. We also discuss the derivation of instance formulas from $LF(fd)$. $ID(fd)$ is the collection of all instance diagrams that can be obtained from a service feature diagram $fd$. 
5.1 Interpretation

as we discussed in Chapter 4. In Section 5.3, we encode the instance diagrams of $ID(fd)$ into Linear Instance Formulas $LI(fd)$, where each element of $LI(fd)$ is the encoding of an instance diagram.

We also discuss that the set of instance formulas $IF(fd)$ derived from $LF(fd)$ and the set of linear instance $LI(fd)$ are equivalent, which makes the above diagram commute. In Section 5.4, we validate instance formulas by means of examples, whereas Section 5.5 summarizes the chapter.

5.1 Interpretation

The traditional notion of instance does not support service feature diagrams. For example, a traditional instance will not allow the manager of a Car Manufacturer shown in Fig. 4.11 to get a feature combination that can be offered to a requestor. Traditional instances can not differentiate between variable features selected by the requestor from those which are selected by the provider, and resource features from shareable features. Instead, we propose to use a linear formula to capture a permissible selection of features of the service feature diagram, which we call instance formula, as explained in Section 5.2. To formalise the notion of instance formula, we interpret service feature diagrams in Linear Logic.

Let us consider the same Car Manufacturer for a different case, where the provider wants to check whether they can satisfy certain requirements. Interpreting service feature diagrams in Linear Logic and checking the derivation of the resulting linear formula from the formula representing the descriptions is one of the ways to check whether the available description satisfies the requirements [NH11].

5.1.1 Interpretation in Linear Logic

We want to encode a service feature diagram into a logical formula giving the same result as expected from the diagram [Ben07]. For example, in the scenario shown in Fig. 5.2, a provider offering a deal would not be happy with a requestor who accepts
this deal by selecting the root feature R while rejecting any of Burger B, Coke C, Fries F, as they all are mandatory features.

To interpret service feature diagrams in Linear Logic, we provide the rules in Table 5.1 to encode the basic feature types. The first column contains rule names used for the encoding. The second column shows the linear formulas of different service feature diagrams and the third column explains the encoding. A diamond is used as a meta variable for representing features that can either be resource or shareable. If \( x \) is a resource, \( \Diamond x \) is replaced by \( x \) and if \( x \) is shareable, \( \boxed{x} \) is replaced by \( x \). Recursive applications of the rules are required to encode multi-level service feature diagrams.

Let us use the example of a vending machine (inspired by Hoare’s example [Hoa85]) while elaborating the encoding of Table 5.1. Let choco be used as an atomic proposition to represent a chocolate bar, while a pound is represented by £1. To state the fact that one can get a choco for £1, we use a linear implication £1 \( \rightarrow \) choco which states that after getting a choco, £1 will be consumed. So, the situation where a requestor has £1 and wants to get a choco can be written as £1 \( \otimes \) (£1 \( \rightarrow \) choco), which entails choco.

In our case, we use linear implication \( a \rightarrow b \) to express the fact that we have to choose the feature \( a \) to select the feature \( b \). Due to the intuitive meaning of \( A \rightarrow B \), i.e, the proposition \( A \) gets consumed once \( B \) is obtained, we need to use an extra copy of the parent feature to keep the intermediate features in the possible instance formula. For example, \( a \otimes (a \otimes (a \rightarrow b)) \) represents the fact that a feature \( a \) is required to be chosen for the selection of the feature \( b \), whereas we have already chosen feature \( a \), while an extra copy of feature \( a \) is used to keep the feature in possible instance formula. In other words, \( a \otimes (a \otimes (a \rightarrow b)) \vdash a \otimes b \), if we do not use extra copy of the feature \( a \) we will never get this feature at in the instance.

It is important to note that the use of an extra copy of a feature before the \( \rightarrow \) does not depend on the depth of a service feature diagram. Whenever we want to encode a service feature diagram, we require three copies of a parent feature before
Table 5.1: Encoding of Different Feature Types in Linear Logic.

<table>
<thead>
<tr>
<th>Rule Names</th>
<th>Linear Formulae of Features</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule::Res</td>
<td>$LF([x]) = x$</td>
<td>A resource feature maps to a linear variable.</td>
</tr>
<tr>
<td>Rule::Share</td>
<td>$LF([x]) = !x$</td>
<td>Feature $x$ is <em>shareable</em>, behaving like a classical proposition. From $!x$ we can derive $x \otimes \cdots \otimes x$.</td>
</tr>
<tr>
<td>Rule::Man</td>
<td>$LF(\mathcal{X} \bullet X) = LF(\mathcal{X}) \otimes (LF(\mathcal{X}) \otimes (LF(\mathcal{X}) \rightarrow LF(X)))$</td>
<td>Subtree $X$ is <em>mandatory</em>, i.e., must be present with feature $x$. In this case, we can derive $\mathcal{X} \otimes LF(X)$.</td>
</tr>
<tr>
<td>Rule::OptR</td>
<td>$LF(\mathcal{X} \rightarrow_\text{opt} X) = LF(\mathcal{X}) \otimes (LF(\mathcal{X}) \otimes (LF(\mathcal{X}) \rightarrow (LF(X) &amp; LF(X)^\perp)))$</td>
<td>$X$ is <em>optional</em>, expressed by requestors’ choice between $LF(X)$ and $LF(X)^\perp$. If $x$ is selected, one can derive both $LF(X)$ and $LF(X)^\perp$.</td>
</tr>
<tr>
<td>Rule::OptP</td>
<td>$LF(\mathcal{X} \rightarrow_\text{opt} X) = LF(\mathcal{X}) \otimes (LF(\mathcal{X}) \otimes (LF(\mathcal{X}) \rightarrow (LF(X) \oplus LF(X)^\perp)))$</td>
<td>$X$ is <em>optional</em>, expressed by choice made by provider between $LF(X)$ and $LF(X)^\perp$. If $x$ is selected, one can derive $LF(X) \oplus LF(X)^\perp$.</td>
</tr>
<tr>
<td>Rule::AlterR</td>
<td>$LF(\mathcal{X}<em>1 \cdots X_n) = \frac{1}{n} \sum</em>{i=1}^{n} (LF(X_i) \otimes (LF(X_j)^\perp))$</td>
<td>The exclusive OR of subtrees $X_1, X_2, \ldots, X_n$ is expressed by the choice made by requestor between selecting any $X_i$ while deselecting the rest. All of these choices are derivable from $x$.</td>
</tr>
<tr>
<td>Rule::AlterP</td>
<td>$LF(\mathcal{X}<em>1 \cdots X_n) = \frac{1}{n} \sum</em>{i=1}^{n} (LF(X_i) \otimes (LF(X_j)^\perp))$</td>
<td>The exclusive OR of subtrees $X_1, X_2, \ldots, X_n$ is expressed by the choice made by provider between selecting any $X_i$ while deselecting the rest. All of these choices are derivable from $x$.</td>
</tr>
</tbody>
</table>

Continued on the next page
Table 5.1 – continued from the previous page

<table>
<thead>
<tr>
<th>Rule Names</th>
<th>Linear Formulae of Features</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule::Orₐₕ</td>
<td>( LF\left( \oplus_{i=1}^{n} LF(X_i) \otimes \bigotimes_{j \in s^3} LF(X_j) \right) \rightarrow )</td>
<td>The OR group of the subtrees ( X_1, X_2, \ldots, X_n ) allow for a requestor’s choice between all subsets of features, deselecting their respective complements.</td>
</tr>
<tr>
<td>Rule::Orₜₚ</td>
<td>( LF\left( \oplus_{i=1}^{n} LF(X_i) \otimes \bigotimes_{j \in s^3} LF(X_j) \right) \rightarrow )</td>
<td>The OR group of the subtrees ( X_1, X_2, \ldots, X_n ) is expressed by the choice made by provider between selecting any ( X_i ) while deselecting all others.</td>
</tr>
<tr>
<td>Rule::And</td>
<td>( LF\left( \left( \bigotimes_{i=1}^{k} BE(S_i) \otimes \bigotimes_{j=k+1}^{n} BE(G_j) \right) \right) \rightarrow )</td>
<td>The And group combines a set of Solitary and/or Group subtrees, the selection of Solitary subtrees ( S_1 ) to ( S_k ) and Group subtrees ( G_{k+1} ) to ( G_n ) depend on their types and selection rights, further elaborated in the explanation.</td>
</tr>
<tr>
<td>Rule::Implies</td>
<td>Replace ( LF(x_1) ) by ( LF(x_1) \otimes !x_2 )</td>
<td>Implies constraint enforces the selection of ( x_2 ) with ( x_1 ) in an instance.</td>
</tr>
<tr>
<td>Rule::Exclude</td>
<td>Replace ( LF(x_1) ) by ( LF(x_1) \otimes !x_2 )</td>
<td>Exclude constraint enforces not to select both ( x_1 ) and ( x_2 ) in an instance.</td>
</tr>
</tbody>
</table>

\(^3s = \{ k : 1 \leq k \leq n \wedge k \neq i \}.\)

\(^4v \in \{ 0, 1 \}, LF(X)^0 = LF(X)^1\) and \( LF(X)^1 = LF(X).\)

\(^5BE\) stands for Borrowed Encoding.
the linear implication. For example, $a \otimes (a \otimes (a \rightarrow b))$ entail $a \otimes b$ is used to encode a feature diagram having $a$ as parent of a mandatory subfeature $b$, as shown in the Rule::Man. The first two rules Rule::Res and Rule::Share encode the resource and the shareable features, respectively.

Additive conjunction ($a \& b$) represents alternative occurrences of features, the choice of which the requestor controls. In the vending machine, offering different flavours of coffee means that one can get a cup of cappuccino, a cup of hot chocolate, and a cup of espresso, each costing £1. Thus the encoding £1 $\rightarrow (cappuccino \& hot-choco \& espresso)$ would state the same fact—£1 can be used to get any one, but the choice of coffee flavour is left with the requestor.

In feature modelling, we use additive conjunction ($\&$) to leave the choice of a variable feature with the requestor. For example, rule Rule::Opt$_R$ of Table 5.1 says that selection or rejection of an optional subtree $X$ depends on the requestor’s choice. So, the encoding in Rule::Opt$_R$ entails two instance formulas. The first shows the selection of an optional subtree $X$ along with its parent, whereas the second states that an optional subtree $X$ can be rejected even if its parent is selected. In the same way we use $\&$ in the rules Rule::Alter$_R$, Rule::Or$_R$ in Table 5.1 while encoding variable features with requestor’s choice.

Additive disjunction ($a \oplus b$) represents alternative choice where the machine controls the selection of each alternative. For example, suppose the vending machine permits gambling: Insert £1 and the machine may provide a cup of cappuccino, a hot chocolate, or espresso. This situation, in Linear Logic, is expressed as £1 $\rightarrow (cappuccino \oplus hot-choco \oplus espresso)$. This means to give the choice of a variable feature to the provider. In other words, we use this encoding if a requestor is happy to get any feature from the set of alternatives provided to them. We use additive disjunction ($\oplus$) in the rules Rule::Opt$_P$, Rule::Alter$_P$, Rule::Or$_P$ to give the choice of variable features to the provider.

Let us use an example to discuss the concepts presented so far, where a resource feature $a$ is a parent of an Alternative-group (R) of two subfeatures $b$ and $c$. Further
feature $c$ has a mandatory subfeature $d$, as shown in Fig. 5.1.

![Figure 5.1: An Example Service Feature Diagram](image)

The step-wise encoding of the service feature diagrams using the rules mentioned in Table 5.1 is shown below:

1. $LF(a) \otimes (LF(a) \otimes (LF(a) \rightarrow ((LF(b) \otimes LF(c) \perp) \& (LF(b) \perp \otimes LF(c)))))$
2. $a \otimes (a \otimes (a \rightarrow ((b \otimes c \perp \otimes d \perp) \& (b \perp \otimes (LF(c) \otimes (LF(c) \otimes (LF(c) \rightarrow LF(d)))))])])$
3. $a \otimes (a \otimes (a \rightarrow ((b \otimes c \perp \otimes d \perp) \& (b \perp \otimes (c \otimes (c \otimes (c \rightarrow d)))))))$

Please note, $LF(c) \perp = c \perp \otimes d \perp$ in the above encoding because rejection of a subtree $X$, $X^\perp$ is the multiplicative conjunction of the rejected features existing in the subtree.

The Rule::And provides an abstract encoding of an And-group, where features in $\{S_1, S_2, \ldots, S_k\}$ are Solitary and $\{G_{k+1}, G_{k+2}, \ldots, G_n\}$ represent Group features. $BE$ inside the encoding refers to the borrowed encoding for each $S_i$ or $G_j$ from their respective rules. Each variable feature discussed so far can either be selected by a requestor or by a provider. The And-group is special in the sense that it may have variable features with different selection rights. In other words, an And-group can have multiple variable features, some of which are selected by the requestor and others by the provider.

For example, the And-group shown in Fig. 5.2 consists of two Alternative-groups, one is selected by the requestor, the other by the provider. The Rule::And shows the generalized form of And-group, where we use meta edge with solid and dashed lines for each subfeature. Each meta edge is replaced by the dashed line for a variable feature to be selected by the provider, or by a solid line otherwise.

Each encoding shown in Table 5.1, is split by $\rightarrow$: before the implication, and after
the implication. While encoding an And-group, we borrow the second part of the encoding of its member from their corresponding rules, called BE in the encoding of Rule::And. The encoding of an And-group also has two parts: The first consists of the multiplicative conjunction (⊗) of three copies of parent features, whereas the second consists of the multiplicative conjunction of the borrowed encodings.

Table 5.2 shows the borrowed encoding for the rules shown in Table 5.1, where X is a subtree for which the encoding is being borrowed and n is the maximum index of the subtree. The second column in Table 5.2 under ‘Borrowed Encoding (BE)’ shows the encoding of the respective features shown in the first column under ‘Features’. The borrowed part of the encoding of an And-group will be the multiplicative conjunction of the borrowed encodings of each subtree which is the part of the And-group. For example, if an And-group has a mandatory feature shown by $X_1$ and an optional (R) feature shown by $X_2$ then $BE = X_1 \otimes (X_2 \& X_2^\perp)$.

<table>
<thead>
<tr>
<th>Features</th>
<th>Borrowed Encoding (BE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mandatory</td>
<td>$LF(X)$</td>
</tr>
<tr>
<td>Optional (R)</td>
<td>$LF(X) &amp; LF(X)^\perp$</td>
</tr>
<tr>
<td>Optional (P)</td>
<td>$LF(X) \oplus LF(X)^\perp$</td>
</tr>
<tr>
<td>Alternative-group (R)</td>
<td>$\bigwedge_{i=1}^{n} (LF(X_i) \otimes \bigotimes_{j \in s^3} LF(X_j)^\perp)$</td>
</tr>
<tr>
<td>Alternative-group (P)</td>
<td>$\bigoplus_{i=1}^{n} (LF(X_i) \otimes \bigotimes_{j \in s^3} LF(X_j)^\perp)$</td>
</tr>
<tr>
<td>Or-group (R)</td>
<td>$\bigwedge_{i=1}^{n} (&amp;(LF(X_i) \otimes \bigotimes_{j \in s^3} LF(X_j)^{a4}))$</td>
</tr>
<tr>
<td>Or-group (P)</td>
<td>$\bigoplus_{i=1}^{n} (\bigoplus_{j \in s^3} (LF(X_i) \otimes \bigotimes_{j \in s^3} LF(X_j)^{a4}))$</td>
</tr>
</tbody>
</table>

As an example, let us consider a scenario (inspired by [LMSS92]) to apply the encoding rules on a service feature diagram. A requestor $Req$ is looking for a restaurant that offers as a deal a burger ($B$), a coke ($C$), either onion soup ($O$) or salad ($S$) depending on the requestor’s choice, as much fries as the requestor can eat ($F$), and a pie ($P$) or an ice cream ($I$) depending on availability (provider’s choice), as
shown in the Fig. 5.2.

\[
\begin{align*}
\text{Fig. 5.2: A Service Feature Diagram Showing Provider’s Offer}
\end{align*}
\]

The diagram shown consists of an And-group with solitary features \((B, C, \text{ and } F)\) and group features \((O, S \text{ and } I, P)\). As said earlier, all the features are encoded directly using the rules in Table 5.1, except the features of the And-group. Here, we borrow the encoding from their corresponding rules. So, the linear formula of the service feature diagram shown in Fig. 5.2 is:

\[
R \otimes (R \otimes (R \rightarrow (B \otimes C \otimes ((O \otimes S) \& (O \otimes S)) \otimes ((I \otimes P) \oplus (I \otimes P)) \otimes !F))).
\]

For an instance to reflect the semantics of service feature diagrams, we introduce the more general notion of instance formula. Instance formulas are Linear Logic formulas capable of capturing the complete functionalities of a service feature diagram. As explained further below, the set of instance formulas of this service feature diagram is:

\[
\begin{align*}
\{(R \otimes B \otimes C \otimes O \otimes S \otimes I \otimes P \otimes !F) \oplus (R \otimes B \otimes C \otimes O \otimes S \otimes I \otimes P \otimes !F), \\
(R \otimes B \otimes C \otimes O \otimes S \otimes I \otimes P \otimes !F) \oplus (R \otimes B \otimes C \otimes O \otimes S \otimes I \otimes P \otimes !F)\}
\end{align*}
\]

At the level of set of instance formula, we consider each element as requestor’s choice. Providers can make the selection between the options connected through \(\oplus\) inside a formula. E.g., \(R, B, C, \text{ and } F\) are core features so they exist in each instance formula, whereas both formulas are distinguished by the choice made by the requestor between \(O\) and \(S\), i.e., the requestor chooses the first instance formula if they want to have \(O\), otherwise the second instance formula. The provider’s choice is kept inside an instance formula. For example, if a requestor wants to get \(O\), they
select the first instance formula. The provider can make the required selection locally, by choosing the conjunct separated by ⊕ from the selected instance formula.

5.2 Instance Formulas

Let us check whether the traditional notion of instance gives answer to the following questions. We consider the service feature diagram shown in Fig. 5.2:

- How to capture resource and shareable features, e.g., a shareable feature \( F \) and a resource feature \( B \) of Fig. 5.2?

- How to differentiate a variable feature selected by the requestor from one selected by the provider, e.g., features of the Alternative-groups shown in the Fig. 5.2?

The traditional instance, being a set, cannot contain multiple copies of an element [Hal60]. In other words, the instance cannot capture shareable features, because shareable feature can be selected more than once. Furthermore, a service feature diagram may contain variable features chosen by either requestor or by provider. Again, the traditional notion of instance is not able to reflect this, because a variable feature is either selected or rejected, but instance does not give information about the selector [BSC10].

The notion of instance has to be extended to allow for features occurring more than once, and to distinguish between requestor and provider choice. We introduce instance formulas. An instance formula of a diagram \( fd \) with \( n \) features \( \{a_1, a_2, \ldots, a_n\} \) is a Linear Logic formula of the form

\[ \bigoplus (RES \otimes SHA \otimes NEG) \]

where

- \( RES \) is of the form \( (a_1 \otimes \cdots \otimes a_j) \) where \( \{a_1, \ldots, a_j\} \) are resources
- SHA is of the form \((!a_{j+1} \otimes \cdots \otimes !a_m)\) where \(\{a_{j+1}, \ldots, a_m\}\) are shareable

- NEG is of the form \((a_{m+1}^\perp \otimes \cdots \otimes a_n^\perp)\) where \(\{a_{m+1}, \ldots, a_n\}\) are negative (rejected) features

- RES, SHA and NEG are pairwise disjoint.

In an instance formula, provider’s choice is captured by \(\oplus\), whereas each instance formula is a possible choice for the requestor. \(IF(fd)\), the set of all instance formulas of a service feature diagram \(fd\), captures the requestor’s choice as well as the provider’s choice of variable features presented in the diagram. Operationally, this means that the requestor chooses first among a number of instance formulas followed by the provider, who chooses from a number of conjuncts separated by \(\oplus\) in the selected instance formula.

Please note that selected and rejected features of a service feature diagram should be disjoint. The conjuncts in the instance formula can have positive and negative literals corresponding to the selected and rejected features. Shareable features are decorated with \(!\). For example, consider Fig. 4.9 which shows the instances of the service feature diagram of Fig. 4.7. The requestor\(^6\) chooses Radio for the Entertainment system while leaving the choice of the company with the provider\(^7\). In other words, the requestor chooses \((Ent \otimes Rd \otimes SO(Rd)) \oplus (Ent \otimes Rd \otimes AL(Rd))\) and now the provider has to choose a conjunct from the pair separated by \(\oplus\).

The number of instance formulas of a service feature diagram depends on the level of variability captured. A service feature diagram must have at least one instance formula. We define a restricted notation of deduction to derive instance formulas from the linear encoding of service feature diagrams. This restricted deduction \(\vdash_\alpha\) does not allow to use expansion of identity [PT95, Gir87, DC97] and an inference rule \(R\&\).

The proof \(a \vdash a\) is an atomic identity, if \(a\) is atomic formula. An atomic formula does not contain Linear Logic connectives. If the proof \(A \vdash A\) is an expanded identity,

\(^6\)The requestor in this case is the Car Manufacturer Fig. 4.11,
\(^7\)The provider in this case is Supplier of Fig. 4.12.
this means that \( A \) is an arbitrary linear formula which involves the linear logical connective. We only allow to use atomic identities in the proofs.

An instance formula is a choice of features left with the requestor, i.e., we do not want to derive a formula of the form \( A \& B \), where both \( A \) and \( B \) are linear formulas showing the choice of features of the requestor. Restricting the use of an expanded identity and/or the application of the inference rule ‘\( R\& \)’

\[
R\& \quad \frac{\Gamma \vdash B, \Delta \quad \Gamma \vdash C, \Delta}{\Gamma \vdash B \& C, \Delta}
\]

will lead to a formula meeting the format of an instance formula described above.

A formula \( F \) is in normal form (NF) with respect to a set of rules \( R_0 \), if \( F \) can not be rewritten any further using \( R_0 \). Here \( R_0 \) is the set of all inference rules of the sequent calculus of CLL except for the additive Rules \( R\oplus \), \( R\& \) and the exponential Rules \( W? \), \( D? \). More formally, a formula \( F \) will be in \( R_0-NF \) if there does not exist a formula \( F' \) such that \( F \vdash F' \) using the rules in \( R_0 \). If we can derive a formula \( if \) from a linear formula \( LF(fd) \) of a service feature diagram \( fd \), such that \( if \) is in \( R_0-NF \) and does not violate notion of instance formula (e.g. propositions representing the selected and the rejected features inside that formula should be disjoint), then \( if \) is an instance formula.

So, a formula \( if \) is an instance formula for a service feature diagram \( fd \) with \( LF(fd) \) as its linear formula, iff \( LF(fd) \vdash_{a} if \) and \( if \) is in \( R_0-NF \). In the following section, we will explain the encoding of instance diagrams as instance formulas.

### 5.3 Encoding of Instance Diagrams

To encode the instances of service feature diagrams, let us revisit the notion of instance as explained in Section 4.2.3. An instance is a concrete choice of features made by a requestor, so it does not provide any variability for the requestor. For example, if \( a \) and \( b \) are features of a service feature diagram then \( a \rightleftharpoons b \) is an instance that shows the selection of features \( a \) and \( b \). The instance \( a \rightleftharpoons b \) can be obtained from \( a \rightleftharpoons b \) or from \( a \rightleftharpoons b \).
At the level of logic, if \( a \) and \( b \) represent propositions referring to the features of a diagram, then \( a \otimes b \) reflects the fact that both \( a \) and \( b \) are selected. The encoding \( LI \) of an instance diagram is the multiplicative conjunction \( (\otimes) \) of the literals representing the features that exist in an instance diagram. For example, the encoding of the diagram \( a \rightarrow b \) is \( a \otimes b \), derivable from \( a \otimes (a \otimes (a \rightarrow (b \& b^\perp))) \) or from \( a \otimes (a \otimes (a \rightarrow b)) \), the linear formulas of \( a \rightarrow b \) or \( a \rightarrow \neg b \), respectively.

Table 5.3 shows the encoding of instance diagrams, where the 3\(^{rd}\) column shows the encoding of the instance diagrams shown in 2\(^{nd}\) column. The rules in Table 5.3 show the encoding of instance diagrams generated by choices made by the requestor, whereas the provider’s choice for variable features are preserved, as shown in the 4\(^{th}\), 5\(^{th}\) and 6\(^{th}\) rows. Again, the diamond symbol has a variable behaviour, it will be replaced with the respective boxes for resource and shareable features. Further \( \alpha \in \{a, \neg a\}, \beta_i \in \{b_i, \neg b_i\}, \) and \( \beta_i^\perp \in \{b_i^\perp, \neg b_i^\perp\} \). Literal \( \alpha \) refers to the root of a service feature diagram which never gets rejected, whereas \( \beta \) refers to a subfeature in the instance that may get rejected. Root and subfeatures can either be resource or shareable feature, hence we use \( \neg \).

The instance diagrams shown in the rows 4 to 6 have different encoding if considered as service feature diagrams. For example, considering the diagram shown in the row 4 as a service feature diagram would mean that the selection/rejection of the feature \( b \) by the provider require the selection of the feature \( a \). On the other hand, considering it as an instance diagram would mean that the feature \( a \) is already selected now provider has an option to select/reject the feature \( b \). So encoding of an instance diagram does not involve \( \rightarrow \).

In the case of an instance diagrams having more than one cases shown in 2\(^{nd}\) column, one can combine the corresponding cases shown the 3\(^{rd}\) column of Table 5.3. For example, encoding of the instance diagrams shown below is shown on the right:

\[
\begin{align*}
& a \\
& \downarrow \quad \downarrow \\
& b \quad c
\end{align*}
\]

\[ a \otimes b \otimes (c \oplus c^\perp) \]
Table 5.3: Encoding of Instance Diagrams in Linear Logic

<table>
<thead>
<tr>
<th>Sr No.</th>
<th>Instance Diagram</th>
<th>Linear Encoding (LE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1" alt="Diagram 1" /></td>
<td>$\alpha$</td>
</tr>
<tr>
<td>2</td>
<td><img src="image2" alt="Diagram 2" /></td>
<td>$\alpha \otimes \beta$</td>
</tr>
<tr>
<td>3</td>
<td><img src="image3" alt="Diagram 3" /></td>
<td>$\alpha \otimes \bigotimes_{i=1}^{n} \beta_i$</td>
</tr>
<tr>
<td>4</td>
<td><img src="image4" alt="Diagram 4" /></td>
<td>$\alpha \otimes (\beta \oplus \beta^\perp)$</td>
</tr>
<tr>
<td>5</td>
<td><img src="image5" alt="Diagram 5" /></td>
<td>$\alpha \otimes \bigoplus_{i=1}^{n} (\beta_i \otimes \bigotimes_{j \in s} \beta_j^\perp))$</td>
</tr>
<tr>
<td>6</td>
<td><img src="image6" alt="Diagram 6" /></td>
<td>$\alpha \otimes \bigoplus_{i=1}^{n} (\bigoplus_{j \in s} \beta_i \otimes \beta_j^\perp))$</td>
</tr>
</tbody>
</table>

We will follow the same technique to encode the multilevel feature diagrams, as well. If $LE$ is the linear encoding of an instance diagram of diagram $fd$ then linear instance is:

$$li = LE \otimes \bigotimes (RF)^\perp$$

where $RF \in \{A_{fd} \setminus A_{inst}\}$, while $A_{fd}$ and $A_{inst}$ are the sets of feature names of the service feature diagram and the instance, respectively. So, $RF$ is the set of literals that refer to the features that exist in the diagram, but are not selected in an instance. We use $\bigotimes (RF)^\perp$ to capture the multiplicative conjunction of the literals referring to the rejected features.

We may come across a case when all the features are selected in an instance diagram, for example the diagram with all mandatory features. Then we use $RF = \perp$, and $li = LE \otimes (\perp)^\perp$ which leads us to $li = LE$ because $\perp^\perp = 1$ and $a \otimes 1 = a$.
Using distributivity of multiplicatives over additives

\[ a \otimes (b \oplus c) \equiv (a \otimes b) \oplus (a \otimes c) \]

over the linear instance \( li \) of a service feature diagram \( fd \) gives us the formula that meets the general form of the instance formula \( if \) described earlier.

The set of all the linear instances \( LI(fd) \) and the set of all instance formulas \( IF(fd) \) of a service feature diagram \( fd \) are equivalent to each other, i.e.,

\[ \forall li \in LI(fd). IF(fd) \vdash li \land \forall if \in IF(fd). LI(fd) \vdash if \]

Let us proof the above expression by an example of a service feature diagram \( fd \) with shareable feature \( x \) as root and two optional subfeatures \( b \) and \( c \) with requestor’s choice and provider’s choice respectively, as shown in Fig. 5.3. Using the encoding shown in Table 5.1,

\[
LF(fd) = !a \otimes (!a \otimes (!a \rightarrow (b \& b^\perp) \otimes (c \oplus c^\perp))).
\]

On the other hand:

\[
IF(fd) = \{(!a \otimes b \otimes c) \oplus (!a \otimes c^\perp), (!a \otimes b^\perp \otimes c) \oplus (!a \otimes b^\perp \otimes c^\perp)\}
\]

represents the fact that \( b \) and \( c \) may be selected or rejected depending on the requestor’s choice and provider’s choice respectively. We can say that, if the requestor wants to select feature \( b \), they have to choose the first instance formula from the set \( IF \), otherwise the second instance formula. The provider has to decide about
Indeed, using the inference rules of the sequent calculus of CLL, we can prove that \( LF(fd) \) entails each instance formula \( if \) of the set \( IF(fd) \). Let us derive the first instance formula of \( IF(fd) \) shown above.

In the above proof, we have shown the deduction of \( LF(fd) \vdash if \), where \( if \in IF(fd) \). We also checked the above proof with llprover—an online prover for CLL [Tam95]. The proof for the second element (and indeed for every element \( if \) of \( IF(fd) \)) of a service feature diagram \( fd \) can also be generated in the same way. The derived instance formula \( if \) is in \( R_0-NF \), because we do not find a formula \( if' \) for which \( if \vdash if' \) using the \( R_0 \).

Using the rules shown in Table 4.2, one can generate the instance diagrams from a service feature diagram (cf. Chapter 4). The Fig. 5.3 has two instance diagrams as shown in Fig. 5.4.

\[
\begin{align*}
(a) & \quad \begin{array}{c}
\text{Fig. 5.4: Instance Diagrams of a Service Feature Diagram of Fig. 5.3}
\end{array} \\
\end{align*}
\]

On the other hand, using the encoding rules presented in Table 5.3, the set of linear instances of Fig. 5.4 is:

\[
LI(fd) = \{!a \otimes b \otimes (c \leftrightarrow c^\perp), !a \otimes (c \leftrightarrow c^\perp) \otimes b^\perp\}
\]
Please note that $LF(fd)$ and $LI(fd)$ contain equivalent formulas. In the following, we validate the instance formulas for the feature types shown in the Table 4.1, by using a number of examples.

## 5.4 Validation of Instance Formulas

We consider the derivation $LF(fd) \vdash if$, where $LF(fd)$ is the linear formula of a service feature diagram $fd$ and $if$ are the corresponding instance formulas. We will only consider service feature diagrams with two levels, because we will validate each feature type individually. The same approach can be followed to validate the instance formulas of multi-level diagrams. All the proofs presented below are also checked by the Ilprover–online tool available to generate the proofs for CLL [Tam95].

Greek letters used in proofs refer to the diamond symbols in the diagrams. For example, $\alpha$ refers to feature $a$ used inside the diamond symbol in the diagram. The diamond symbol has a variable behaviour: it can either be replaced by a box representing a resource or a shareable feature. Greek letters are therefore to be replaced by a proposition followed by $!$ modality for shareable feature, or a proposition otherwise. Let us now validate the instance formula for each feature type.

### 5.4.1 Mandatory Feature

A mandatory feature is always selected along with its parent. Let us consider the service feature diagram $fd$ shown on the right. The linear formula $LF(fd)$ can be obtained by replacing $LF(X)$ by $\beta$ in Rule::Man of Table 5.1, i.e., $\alpha \otimes (\alpha \otimes (\alpha \rightarrow \beta))$. On the other hand, the set of instance formulas is $IF(fd) = \{\alpha \otimes \beta\}$. Now, let us check $LF(fd) \vdash if$ where $if \in IF(fd)$. 
5.4 Validation of Instance Formulas

5.4.2 Optional Features

If a feature is selected, its optional subfeature may be selected or rejected.

Let us consider a service feature diagram \( fd \) with optional feature of requester’s choice, as shown in the right. The linear formula \( LF(fd) \) can be obtained by replacing \( LF(X) \) by \( \beta \) in the Rule::Opt\(_R\) of Table 5.1, i.e., \( \alpha \otimes (\alpha \rightarrow (\beta \& \beta^\perp)) \). On the other hand, instance formula \( IF(fd) = \{\alpha \otimes \beta, \alpha \otimes \beta^\perp\} \). Now, let us check \( LF(fd) \vdash if \) for each \( if \).

The proof tree for the second instance formula \( (if) \) of the set of instance formulas \( IF(fd) \) is

Selection rights of an optional feature may also be left with the provider.

Let us consider a service feature diagram \( fd \) with an optional subfeature of provider’s choice, as shown in the right. The linear formula \( LF(fd) = \alpha \otimes (\alpha \rightarrow (\beta \& \beta^\perp)) \) can be obtained by replacing \( LF(X) \) and \( LF(X)^\perp \).
with $\beta$ and $\beta^\perp$, respectively in the Rule::Opt$_P$ of Table 5.1. On the other hand, $IF(fd) = \{(\alpha \otimes \beta) \oplus (\alpha \otimes \beta^\perp)\}$. Now let us check $LF(fd) \vdash if$:

\[
\begin{array}{c}
\alpha \vdash \alpha \\
\beta \vdash \beta \\
\alpha, \beta \vdash (\alpha \otimes \beta) \oplus (\alpha \otimes \beta^\perp) \\
\alpha, \beta^\perp \vdash (\alpha \otimes \beta) \oplus (\alpha \otimes \beta^\perp) \\
\alpha, \beta \vdash (\alpha \otimes (\alpha \rightarrow (\beta \oplus \beta^\perp))) \vdash (\alpha \otimes (\alpha \otimes (\alpha \rightarrow (\beta \oplus \beta^\perp)))) \vdash (\alpha \otimes (\alpha \otimes (\alpha \otimes (\beta \oplus \beta^\perp)))) \\
\end{array}
\]

\[\text{5.4.3 Alternative-groups}\]

If a feature is selected, exactly one member of its Alternative-group must be selected. Let us consider a service feature diagram $fd$ having an Alternative-group with requestor’s choice, as shown in the figure on the right. The linear formula $LF(fd) = \alpha \otimes (\alpha \otimes (\alpha \rightarrow ((\beta_1 \otimes \beta_2^\perp) \& (\beta_1^\perp \otimes \beta_2))))$ is obtained by replacing $LF(X_1)$ and $LF(X_n)$ by $\beta_1$ and $\beta_2$, respectively, in the Rule::Alter$_R$ of Table 5.1. The set of instance formulas $IF(fd)$ is $\{\alpha \otimes \beta_1 \otimes \beta_2^\perp, \alpha \otimes \beta_1^\perp \otimes \beta_2\}$. Now, let us check $LF(fd) \vdash if$:

\[
\begin{array}{c}
\beta \vdash \beta \\
\beta \vdash \beta \\
\beta \vdash \beta \\
\beta \vdash \beta \\
\beta \vdash \beta \\
\beta \vdash \beta \\
\beta \vdash \beta \\
\beta \vdash \beta \\
\beta \vdash \beta \\
\end{array}
\]

The proof tree for the second element $if$ of the set of instance formulas $IF(fd)$ is
5.4 Validation of Instance Formulas

5.4.4 Or-groups

If a feature is selected, at least one member from its Or group must be selected. An Or-group also provide variability, hence we consider it for requestor’s and provider’s choice. Let us consider a service feature diagram having an Alternative-group with provider’s choice, i.e., group features are to be selected by the provider, as shown in figure on the right. The set of instance formulas $IF(fd)$ is $\{(\alpha \otimes \beta_1 \otimes \beta_2^2) \oplus (\alpha \otimes \beta_1^2 \otimes \beta_2)\}$. The linear formula $LF(fd)$ can be obtained by replacing $LF(X_1)$ by $\beta_1$ and $LF(X_n)$ by $\beta_2$ in Rule::$Alter_p$ of Table 5.1, i.e., $\alpha \otimes (\alpha \rightarrow ((\beta_1 \otimes \beta_2^2) \oplus (\beta_1^2 \otimes \beta_2)))$. Now, let us check $LF(fd) \vdash if$.

5.4.4 Or-groups

If a feature is selected, at least one member from its Or group must be selected. An Or-group also provide variability, hence we consider it for requestor’s and provider’s choice.

Let us consider a service feature diagram $fd$ having an Or-group with requestor’s choice, as shown in the figure on the right. The set of instance formulas $IF(fd)$ is $\\{\alpha \otimes \beta_1 \otimes \beta_2^2, \alpha \otimes \beta_1^2 \otimes \beta_2, \alpha \otimes \beta_3 \otimes \beta_2\}$. The linear formula $LF(fd)$ can be obtained by replacing $LF(X_1)$ and $LF(X_n)$ by
The proof tree for the second element if $(5.4$ Validation of Instance Formulas $79)$ if $(\beta_1 \otimes \beta_2) \& (\beta_1 \otimes \beta_2))$. The proof of $LF(fd) \vdash if$ for the first element if is

The proof tree for the second element if of the set of instance formulas is

The proof tree for the third element if of the set of instance formulas is
Now, let us consider derivation of $LF(fd) \vdash if(fd)$ when $fd$ consists of an Or-group with provider’s choice, i.e., the selection of features from this group is left with the provider, as shown in the figure on the right. The set of instance formulas $IF(fd)$ is $\{(\alpha \otimes \beta_1 \otimes \beta_2^\perp) \oplus (\alpha \otimes \beta_1 \otimes \beta_2)\}$. The linear formula $LF(fd)$ is obtained by replacing $LF(X_1)$ by $\beta_1$ and $LF(X_n)$ by $\beta_2$ in the Rule::OrP of Table 5.1, i.e., $\alpha \otimes (\alpha \otimes (\alpha \rightarrow ((\beta_1 \otimes \beta_2^\perp) \oplus (\beta_1 \otimes \beta_2) \oplus (\beta_1 \otimes \beta_2))))$. Due to depth of the proof we have shown $LF(fd) \vdash if$ on the next page in landscape mode.

5.5 Summary

In this chapter, we have introduced and discussed the semantics of service feature diagrams in Linear Logic. We discussed the limitations of traditional instances, then provided the solution to those limitations by introducing a set of instance formulas. At the end, we have provided the validation of instance formulas by using examples.
Graphical Matching Rules for Service Feature Diagrams

Graphical rules provide a simple way to present the required semantics while hiding the logical semantics. So, we define a set of graphical rules to check whether a set of descriptions satisfies given requirements. Each graphical rule is of the form $P F D \xrightarrow{\text{Satisfies}} R F D$ where $P F D$ is the provider and $R F D$ the requestor service feature diagram. For example, if $V e g$ and $N - V e g$ are the features representing vegetarian and non-vegetarian food, respectively then,

![Diagram]

means that a requirement of an And-group of vegetable and non-vegetarian food can be satisfied by an Or-group (R) of vegetarian and non-vegetarian food.

In general, for the graphical rules, the total number of features on the left and right side of each rule should be the same. Since no matching rule deletes a feature, the number of features in description and requirements remain constant. The graphical rules can be used if multiple offers contribute towards the requestor’s needs. The diagrams used in the graphical follows the notions captured by the metamodel of service feature diagrams (cf. Metamodel in Chapter 4).
In the example stated above, we have used two subfeatures on each side of the rule. To make the rules as general as possible we have formulated the rules for \( n \) features. For every rule, the index \( n \) refers to the highest index in each group. In the case of And-groups of solitary features, \( n \geq 1 \) and in case of group features, \( n \geq 2 \). For example, the rule shown in Fig. 6.1 also works for a single feature on both sides, one mandatory feature satisfies the requirement of an optional (P) feature.

We provide the graphical rules for all the feature types of service feature diagrams in Sections 6.2 to 6.8 with the use of generic rule wherever possible. A generic rule consists of a generic feature diagram capable of capturing multiple concrete rules. Section 6.1 discusses generic rules and generic feature diagrams in detail. In Section 6.9, we analyse the graphical rules to check whether all desired combinations of requirements can be derived from the description. In Section 6.10, we discuss the application of matching rules to the case study presented in Chapter 4.

### 6.1 Diagrams used in Generic Rules

Generic rule captures multiple concrete rules in a single diagram. Table 6.1 shows the diagrams which can be used in a generic rule in the 3\(^{rd}\) column, whereas the 2\(^{nd}\) column consists of the names of the diagrams used in the rule, where Opt and Man represent optional and mandatory features, R and P refer to requestor and provider, respectively. The 4\(^{th}\) to 6\(^{th}\) columns show the concrete rules that can be obtained from a generic rule.

Whenever we have a generic rule, then it may be replaced by one of its variants shown in the last three columns. For example, a generic rule shown in Fig. 6.3 can be considered as a combination of two concrete rules: 1) And-group of optional (R) features satisfying And-group of an optional (P) features, 2) And-group of Optional (R) features satisfying an And-group of mandatory features. Please note, this rule became generic rule due to the presence of double circles at the end of double edges. It is worth mentioning that the variants shown in 4\(^{th}\) to 6\(^{th}\) columns of Table 6.1 follow the constraints captured by the metamodel of service feature diagrams.
6.2 Matching Rules for And-groups of Solitary Features

An And-group can either consist of solitary, or group features, or both. Here we only consider And-groups of solitary features. We will discuss And-groups of solitary and group features after explaining the graphical rules for group features in Section 6.5. Let us first consider the And-group of mandatory features.

### 6.2.1 Rules for And-groups of Mandatory Features

A mandatory feature must be selected if its parent is. Mandatory features are not flexible in nature, so there is no distinction of who makes the selection. Let us provide the graphical rules for requirements to be satisfied by And-groups of mandatory features.

**And-group of Mandatory Features to And-group of Optional (P) Features:** An And-group of mandatory features satisfies an And-group of optional...
features with provider’s choice as shown in Fig. 6.1, where (P) means that the optional features are selected by the provider. That means, if the requestor allows the provider to choose the optional features in the requirements, the And-group of mandatory features in the descriptions satisfies this requirement.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{figure6.1}
\caption{And-group of Mandatory Features Satisfying And-group of Optional (P) Features}
\end{figure}

**And-group of Mandatory Features to Or-group (P):** An And-group of mandatory features satisfies an Or-group with provider’s choice, as shown in Fig. 6.2.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{figure6.2}
\caption{And-group of Mandatory Features Satisfying Or-group (P)}
\end{figure}

### 6.2.2 Rules for And-groups of Optional (R) Features

If the parent of an optional (R) feature is selected, the requestor may select or reject an optional subfeature. In the following, we will show the cases where an And-group of optional (R) features satisfies the requirements.

**And-group of Optional Features to And-group of OptP-Man Features:** If a provider allows a requestor to choose from an And-group of optional features, this satisfies an And-group of optP-man features, as shown in Fig. 6.3.

To know how much of the cases are captured by a generic rule, we need to replace the generic feature diagram of the rule by its corresponding concrete diagrams shown in the last column of Table 6.1. For example, a generic feature diagram (OptP-Man)
used in the right of the Fig. 6.3 consists of two concrete rules: optional (P) feature and mandatory feature. Hence, the multi-rule shown in Fig. 6.3 captures the following cases:

1. And-group of optional (R) features satisfying an And-group of optional (P) features,

2. And-group of optional (R) features satisfying an And-group of mandatory features

We will use the same technique to unfold the cases captured by a multi-rule.

**And-group of Optional Features to Or-group (RP):** An And-group of optional (R) features satisfies the Or-group (RP) (Or-groups with requestor’s or provider’s choice), as shown in a generic rule of Fig. 6.4.

**And-groups of Optional Features to Alternative-group (RP):** An And-group of optional (R) features satisfies an Alternative-group (RP) (Alternative-group with requestor’s or provider’s choice), as shown in a multi-rule of Fig. 6.5.

### 6.2.3 Rules for Mixed And-groups

An And-group of solitary features can have a combination of mandatory and optional features, which we call a mixed And-group. If the optional features in the mixed
And-group are selected by the requestor, we call it Mixed And-group (R) otherwise Mixed And-group (P). If some of the optional features of an And-group are selected by the requestor and some by the provider then we call it Mixed And-group with mixed choices. Let us check which requirements are satisfied by a mixed And-groups in descriptions.

**Mixed And-group (R) to Alternative-group (P):** A mixed And-group (R) satisfies the requirements of an Alternative-group (RP), as shown in Fig. 6.6. Please note that the mixed And-group (R) shown in the rule must have only one mandatory feature, otherwise the rule shown in Fig. 6.6 will not be applicable.

**Mixed And-group with Mixed Choices to Alternative-group (P):** A mixed And-group with mixed choices satisfies the requirements of an Alternative-group (P), as shown in Fig. 6.7.

**Mixed And-group of OptR-Man and OptRP to Or-group (P):** A mixed And-group of OptR-Man and OptRP satisfies the requirements of an Or-group (P),
as shown by a generic rule in Fig. 6.8. From here onwards $i < n$ such that $n$ does not violate the condition described above, i.e., $n \geq 1$.

![Figure 6.8: And-group of OpR-Man and OpRP Features Satisfying Or-group (P)](image)

The generic rule shown in Fig. 6.8 captures the following cases:

1. And-group of optional (R) and optional (P) features satisfying Or-group (P),
2. And-group of optional (R) features satisfying Or-group (P),
3. And-group of mandatory and optional (P) features satisfying Or-group (P),
4. And-group of mandatory and optional (P) features satisfying Or-group (P)

### 6.3 Matching Rules for Alternative-groups

If a feature is chosen, exactly one subfeature from its Alternative-group must also be selected. An Alternative-group provides flexibility in selecting its members, so we consider two cases: 1) when a requestor makes a selection from the Alternative-group; 2) when a provider makes a selection from the Alternative-group. Let us discuss the matching rules for both cases.

#### 6.3.1 Rules for Requestor-based Choice

An Alternative-group with requestor’s choice satisfies the following cases.

**Alternative-group to And-group of Optional (P) Features:** The alternative-group (R) satisfies an And-group of optional (P) features, as shown in Fig. 6.9.

**Alternative-group to Or-group (P):** An Alternative-group (R) satisfies an or-group (P), as shown in Fig. 6.10.
Alternative-group to Alternative-group (P): An Alternative-group (R) satisfies an Alternative-group (P), as shown in Fig. 6.11.

Alternative-groups to And-groups: An Alternative-group (R) satisfies the requirements of an And-group of optional (P) and Alternative-group (P) features, and an And-group of optional (P) and Or-group (P), respectively, as shown in Fig. 6.12.

Alternative-group to Mixed And-group (P) An Alternative-group (R) satisfies the requirements of Mixed And-group (P), as shown in Fig. 6.13.
6.3.2 Rules for Provider-based Choice

When the provider wants to keep the choice of an Alternative-group, it satisfies the following cases.

**Alternative-group to And-group of Optional (P) Features:** An Alternative-group (P) satisfies an And-group of optional (P) features, as shown in Fig. 6.14.

![Figure 6.14: Alternative-group (P) Satisfying Optional (P) Features](image)

**Alternative-group to Or-group (P):** An Alternative-group (P) satisfies an Or-group (P), as shown in Fig. 6.15.

![Figure 6.15: Alternative-group (P) Satisfying Or-group (P)](image)

6.4 Matching Rules for Or-groups

At least one subfeature of an Or-group must be selected, if the parent of this group is chosen. Again, we consider two cases while matching the Or-group: 1) when a requestor makes a selection from this group; 2) when a provider makes a selection.
6.4.1 Rules for Requestor-based Choice

Let us discuss the requirements that can be satisfied by an or-group (R).

**Or-group to Or-group (P):** An Or-group (P) of the requirements can be satisfied by an Or-group (R) of the description, as shown in Fig. 6.16.

![Figure 6.16: Or-group (R) Satisfying Or-group (P)](image)

**Or-group to Alternative-groups:** An Or-group (R) satisfies an Alternate-group with requestor’s choice as well as an Alternative-group with provider’s choice, as shown in Fig. 6.17.

![Figure 6.17: Or-groups (R) Satisfying Alternative-group (RP)](image)

The generic rule shown in Fig. 6.17 captures the following cases:

1. Or-group (R) satisfying Alternative-group (R),
2. Or-group (R) satisfying Alternative-group (P)

**Or-groups to And-groups of OptP-Man Features:** The Or-groups (R) satisfies an And-group of optP-man features, as shown in Fig. 6.18. Again the rule shown in Fig. 6.18 is generic rule which captures the following cases:

1. Or-group (R) satisfying And-group of mandatory features,
2. Or-group (R) satisfying And-group of optional (P) features,
3. Or-group (R) satisfying Mixed And-group (P)
Or-groups to And-groups of Solitary and Group Features: An Or-group (R) satisfies the And-groups of optP-Man features and an Alternative-group (P) and the And-groups of optP-Man features and Or-group (P), as shown in Fig. 6.19.

The generic rules shown above in Fig. 6.19 can represent the following cases:

1. Or-group (R) satisfying And-group of mandatory features and an Alternative-group (P),
2. Or-group (R) satisfying And-group of optional (P) features and an Alternative-group (P),
3. Or-group (R) satisfying And-group of mandatory features and an Or-group (P),
4. Or-group (R) satisfying And-group of optional (P) features and an Or-group (P)

6.4.2 Rule for Provider-based Choice

If a provider wants to keep the choice of the Or-group, it satisfies the following set of requirements.

Or-group to And-group of Optional (P) Features: An Or-group (P) satisfies an And-group of optional (P) features, as shown in Fig. 6.20.
6.5 Matching Rules for And-groups of Solitary and Group Features

Here we present the rules where an And-group combining solitary and group features satisfies a certain set of requirements. We consider all types of solitary features with each group feature and see which type of requirement are satisfied by this combination.

Let us first consider an And-group of solitary features and an Alternative-group.

6.5.1 Rules for And-groups of Solitary and Alternative-groups

An And-group of solitary and Alternative-groups satisfies the following set of requirements.

And-groups of Optional (R) and Alternative-groups to Alternative-group (P): An And-group of optional (R) features and Alternative-groups satisfies an Alternative-group of provider-based choice, as shown in Fig. 6.21.

The generic rule shown above captures the following cases:

1. And-group of optional (R) features and an Alternative-group (R) satisfying an Alternative-group (P),
2. And-group of optional (R) features and an Alternative-group (P) satisfying an Alternative-group (P)

**And-groups of Solitary and Alternative-groups to Or-group (P):** An And-group of solitary features and Alternative-groups satisfies an Or-group (P), as shown in Fig. 6.22.

![Figure 6.22: And-groups of Solitary and Alternative-groups (R) Satisfying Or-group (P)](image)

The above generic rule captures the following cases:

1. And-group of optional (R) features and an Alternative-group (P) satisfying Or-group (P),

2. And-group of mandatory features and an Alternative-group (P) satisfying Or-group (P),

3. And-group of optional (R) features and an Alternative-group (R) satisfying Or-group (P),

4. And-group of mandatory features and an Alternative-group (R) satisfying Or-group (P)

**And-groups of Solitary and Alternative-groups to an And-group of Optional (P) Features:** An And-group of solitary features and Alternative-groups satisfies an And-group of optional (P) features, as shown in Fig. 6.23.

The generic rule shown in Fig. 6.23 captures the following cases:

1. And-group of optional (R) features and an Alternative-group (P) satisfying optional (P) features,
6.5 Matching Rules for And-groups of Solitary and Group Features

2. And-group of optional (P) features and an Alternative-group (P) satisfying optional (P) features,

3. And-group of mandatory features and an Alternative-group (P) satisfying optional (P) features,

4. And-group of optional (R) features and an Alternative-group (R) satisfying optional (P) features,

5. And-group of optional (P) features and an Alternative-group (R) satisfying optional (P) features,

6. And-group of mandatory features and an Alternative-group (R) satisfying optional (P) features

6.5.2 Rules for And-groups of Solitary and Or-groups

Requestor-based choice of an Or-group along with solitary features satisfies the following set of requirements.

And-groups of Solitary and Or-groups to Or-group (P): The And-groups of solitary features and Or-groups satisfies the Or-group (P), as shown in Fig. 6.24.

The generic rule shown above captures the following cases:
1. And-group of optional (R) features and an Or-group (P) satisfying Or-group (P),

2. And-group of mandatory features and an Or-group (P) satisfying Or-group (P),

3. And-group of optional (R) features and an Or-group (R) satisfying Or-group (P),

4. And-group of mandatory features and an Or-group (R) satisfying Or-group (P)

**And-group of Optional (R) Features and Or-group (R) to Alternative-groups:** An And-group of optional (R) features and an Or-group (R) satisfies Alternative-groups, as shown in Fig. 6.25.

![Figure 6.25: And-group of optional (R) Features and Or-group (R) Satisfying Alternative-groups](image)

The generic rule shown above captures the following cases:

1. And-group of optional (R) features and an Or-group (R) satisfying Alternative-group (P),

2. And-group of optional (P) features and an Or-group (R) satisfying Alternative-group (P)

**And-group of Optional (R) Features and Or-group (R) to And-group of Mandatory Features:** An And-group of optional (R) features and an Or-group (R) satisfies the And-group of mandatory features, as shown in Fig. 6.26.
6.6 Matching Rules for And-groups of Solitary and Group Features

Figure 6.26: And-group of Optional (R) Features and Or-group (R) Satisfying And-group of Mandatory Features

And-groups of Solitary and Or-groups to And-group of Optional (P) Features: And-groups of solitary features and Or-groups satisfies an And-group of optional (P) features, as shown in Fig. 6.27.

The generic rule shown in Fig. 6.27 captures the following cases:

1. And-group of optional (R) features and an Or-group (P) satisfying optional (P) features,

2. And-group of optional (P) features and an Or-group (P) satisfying optional (P) features,

3. And-group of mandatory features and an Or-group (P) satisfying optional (P) features,

4. And-group of optional (R) features and an Or-group (R) satisfying optional (P) features,

5. And-group of optional (P) features and an Or-group (R) satisfying optional (P) features,

6. And-group of mandatory features and an Or-group (R) satisfying optional (P) features
6.6 Matching Rules for Shareable Features

In contrast to a resource-feature which can be used only once, a shareable-feature can be used as often as required. At the level of service feature diagrams, a shareable feature is represented by a grey background. The following rules explain the derivable combinations of shareable feature.

6.6.1 Rule for Shareable to Resource Feature

A shareable feature satisfies a requirement of a resource feature as shown in Fig. 6.28. The crossed edges in the rule states that the rule is only applicable for leaf nodes.

![Figure 6.28: Shareable Feature Satisfying Resource Feature](image)

All the matching rules described in Section 6.2 to 6.5 give the same result for shareable features if the combination of shareable and resource features does not violate the rule shown in Fig. 6.28.

6.7 Combining Multiple Graphical Rules

If more than one rule can be applied over the description, we can combine those graphical rules. We provide a rule that can be used to combine multiple rules so that more than one rule can be applied at once. For example, let the two rules shown in Fig. 6.29(a) and 6.29(b) applied over a description. We can either apply these rules sequentially or we can combine these rules to be applied in one go. The combined rule of Fig. 6.29(a) and 6.29(b) is shown in Fig. 6.29(c).

That means,

for $s_i$ and $t_i$ being pairwise distinct and having the same root

$$s_1 \vdash t_1, \ldots, s_n \vdash t_n \implies \text{And}(s_1, s_2, \ldots, s_n) \vdash \text{And}(t_1, t_2, \ldots, t_n)$$
where \( s_i \vdash t_i \) is a matching rule and \( \text{And}(a_1, a_2, \ldots, a_n) \) forms the And-group of the subtrees \( a_1, a_2, \ldots, a_n \). In the case of common features among subtrees, we have to use graphical rules for merging, which are explained in the following section.

### 6.8 Graphical Rules for Merging

If several offers jointly satisfy a specific set of requirements, those offers need to be combined by the match, so that a single combined offer can be provided to the requestor. This rule requires to merge feature diagrams.

### 6.8.1 Rules for Merging of Service Feature Diagrams

For the merging of feature diagrams, all the diagrams must have at least one common feature, which may be a resource feature in at most one of the diagrams. A common feature may be non-root in at most one of the diagrams being merged, as shown in Fig. 6.29. The rule shown in Fig. 6.29 creates an And-group of the given diagrams under the common feature, where \( sf_1, \ldots, sf_n \) represent the subfeatures of the common feature in these diagrams.

![Figure 6.29: Merging of Feature Diagrams (with shared features)](image-url)
shareable or a resource feature. The diamond symbols of the feature diagrams being merged and the resulting feature diagram should have the same type, i.e, either a resource or a shareable feature. For example, if the diamond symbol in the feature diagrams being merged is a resource feature, then the root of the merged feature diagram will be a resource feature. Hence, the above rule can produce the following results:

1. If at least one root in the feature diagrams being merged is a resource then the root of the merged diagram will be a resource feature,

2. If all the roots in the features diagrams being merged are shareable features then the root of resulting diagram may be a shareable feature,

3. If all the roots in the feature diagrams being merged are shareable then the root of a merged feature diagram may be a resource feature

An arbitrary root can be used to merge feature diagrams if they do not share any feature. Use of arbitrary root will not effect the semantics of merged feature diagrams. That means, the use of arbitrary root is for the sake of showing a single offer to the requestor instead of multiple offers. For example, the feature diagrams shown in left part of the Fig. 6.30 are merged using arbitrary root as shown in right of Fig. 6.30, where $fd_1, \ldots, fd_n$ are the feature diagrams being merged with no common feature and $A$-Root is an arbitrary root of the resulting diagram.

![Figure 6.30: Merging of Feature Diagrams (without shared features)](image)

### 6.8.2 Rules for Merging of Shareable Features

Once merging of service feature diagrams is achieved using the rule shown in Fig. 6.29, one can have more than one feature with the same name. For example, merging
of provider offers providing the same payment mechanism would result in duplicating the payment type in the resulting diagram. We introduce a merge rule for features, so that the resulting diagram does not violate the metamodel of service feature diagrams shown in Fig. 4.6. Let us consider merge rules for two categories: 1) features being merged are solitary, 2) features being merged are either solitary or group.

**Merging of Solitary Features:** Solitary features are of two types: mandatory and optional. Optional features are further categorized into optional (R) and optional (P) features. We consider all types of solitary features in this rule. In Fig. 6.31, we provide two generic rules representing the merging of solitary features.

![Figure 6.31: Merging of Solitary Features](image)

The generic rules shown above can provide variety of rules depending on the relevance which we use for OptRP-Man edges (cf. Table 6.1). A generic rule shown in the left of Fig. 6.31 captures the following cases:

1. Replacing left and the right OptRP-Man edges by the Optional (R) features in a feature diagram on the left may result a diagram with the Optional (R) relevance in place of OptRP-Man edge in a diagram on the right,

2. Replacing left and the right OptRP-Man edges by the Optional (R) features in a feature diagram on the left may result a diagram with the Mandatory relevance in place of OptRP-Man edge in a diagram on the right,

3. Replacing left and the right OptRP-Man edges by the Mandatory features in a feature diagram on the left results a diagram with the Mandatory relevance in place of OptRP-Man edge in a diagram on the right,
4. Replacing left and the right OpRP-Man edges by the Mandatory and Optional (R) features, respectively in a feature diagram on the left results a diagram with the Mandatory relevance in place of OpRP-Man edge in a diagram on the right,

5. Replacing left and the right OpRP-Man edges by the Optional (R) and Mandatory features, respectively in a feature diagram on the left results a diagram with the Optional (R) edge in place of OpRP-Man edge in a diagram on the right,

6. Replacing left and the right OpRP-Man edges by the Optional (R) and Mandatory features, respectively in a feature diagram on the left may result a diagram with the Mandatory feature in place of OpRP-Man edge in a diagram on the right,

7. Replacing left and the right OpRP-Man edges by the Optional (P) and Optional (R) features, respectively in a feature diagram on the left may result a diagram with the Optional (P) feature in place of OpRP-Man edge in a diagram on the right,

8. Replacing left and the right OpRP-Man edges by the Optional (P) and Mandatory features, respectively in a feature diagram on the left may result a diagram with the Optional (P) feature in place of OpRP-Man edge in a diagram on the right,

9. Replacing left and the right OpRP-Man edges by the Optional (R) and Optional (P) features, respectively in a feature diagram on the left may result a diagram with the Optional (P) feature in place of OpRP-Man edge in a diagram on the right,

10. Replacing left and the right OpRP-Man edges by the Mandatory and Optional (P) features, respectively in a feature diagram on the left may result a diagram
with the Optional (P) feature in place of OpRP-Man edge in a diagram on the right,

11. Replacing left and the right OpRP-Man edges by the Optional (P) features in a feature diagram on the left results a diagram with the Optional (P) feature in place of OpRP-Man edge in a diagram on the right.

Cases from 1 to 8 listed above, can also show three more variations, i.e., 1) replacing diamond symbols on the left and right by the resource features, 2) replacing diamond symbols on the left and right by the shareable features, and 3) replacing diamond symbols on the left by the shareable features and right by the resource features. The cases from 9 to 11 only works if the diamond symbols are replaced by the shareable features. In total the generic rule shown in the left of Fig. 6.31 represents 27 different cases for the merging of solitary features.

Generic rule shown in the right of Fig. 6.31 can also provide the same cases which we discussed for the generic rule on the left. The difference between the generic rules shown in Fig. 6.31 is the position of the features being merged. The feature \( b \) being merged is at the same level in the generic rule shown in left, whereas the generic rule shown on the right merges the features \( b \) existing at different level.

Consider the same example, where two offers are merged, both allowing payment by credit card. There is a possibility that one offer marks the credit card directly under the reservation feature, while the other may mark it under payment type, a subfeature of reservation. In that case we can use this rule to get a service feature diagram with the credit card under the payment type feature.

**Merging of Solitary and Group Features:** Merging of feature diagrams may result in duplication of features. In Fig. 6.32, we show the rules to merge solitary features and Alternative-groups with duplicate features. The duplication may arise because offers already merged are made independently. For example, a shareable feature marked as a solitary by one offer may be part of a group feature in another offer. A credit card may be marked as a single payment type in one offer, while being
part of an Alternative-group in another.

![Diagram of merging solitary feature and alternative-group](image)

Figure 6.32: Merging of Solitary Feature and Alternative-group (R)

In the first rule of Fig. 6.32, we have used a single mandatory feature with an Alternative-group, because an Alternative-group does not allow the selection of more than one subfeature. Merging more than one mandatory features with an Alternative-group is not acceptable for one of the offers that have been merged.

We can use an And-group of mandatory features while merging it with an Or-group, as shown in Fig. 6.33.

![Diagram of merging solitary features and or-group](image)

Figure 6.33: Merging of Solitary Features and Or-group (P)

**Merging of Group Features:** Let us discuss a slight variation of the rule shown in Fig. 6.32 and 6.33 where we merge group features as shown in Fig. 6.34.

Merging of Alternative-groups can only be done if one group is the subset of the other group. Otherwise, these groups can not be merged due to the basic semantics of Alternative-groups where exactly one feature must be chosen. In the case of Or-groups we can merge any two (and indeed more Or-groups) with common parents, as shown in Fig. 6.34.

The above merging rules can only be used when at least one of the features being merged is shareable. If merging of service feature diagrams results in duplicate
resource features, we can not use the rules for merging. So, in that case we will not be able to get a service feature diagram meeting the semantics shown by the metamodel in Fig. 4.6.

### 6.9 Analysis of Graphical Matching Rules

As already stated, we provide rules for deriving requirements from descriptions only for the cases where the requirements are actually satisfied. Here, we analyse the rules to provide the derivable combinations of operations. We have considered all combinations of operations of provider and requestor service feature diagrams irrespective of the result of the match. We use ✓ for a successful match and × otherwise. First, we analyse the comparison of solitary and group features, as shown in Table 6.2. Secondly, we show the comparison of And-groups of solitary and group features with group features and vice versa. The results are shown in Table 6.3. Let us discuss both categories.

In Table 6.2, Source and Target in the 2nd and 3rd columns represent the diagrams on the left and right-hand side of each rule, respectively. Columns 4 to 7 show the result of Source to Target matching: \( S(R), T(R) \) means that both Source and Target are chosen by the requestor; the same choice is given to the provider under \( S(P), T(P) \); \( S(R) \) means that Source is chosen by a requestor and Target by a provider and vice versa under \( (S(P)) \). Each rule gives the same result for shareable features if and only if the combination of a shareable and a resource feature do not
violate the rules of shareable features.

The identity match between the source and target is represented by $\checkmark^{id}$ in Table 6.2 and Table 6.3. We have not considered the identity case for mandatory features. A mandatory feature is always chosen if its parent is, so there is no fight for the selection of this feature. Hence, mandatory features of the description always satisfy the mandatory requirement for that feature.

### Table 6.2: Analysis of Graphical Rules for Solitary and Group Features

<table>
<thead>
<tr>
<th>No.</th>
<th>Source ($S$)</th>
<th>Target ($T$)</th>
<th>Source to Target Matching ($S \vdash T$)</th>
<th>$S(R), T(R)$</th>
<th>$S(P), T(P)$</th>
<th>$S(R)$</th>
<th>$S(P)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Optional</td>
<td>Optional</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>2</td>
<td>Optional</td>
<td>Or</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>3</td>
<td>Optional</td>
<td>Alternative</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>4</td>
<td>Optional</td>
<td>Mandatory</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>5</td>
<td>Or</td>
<td>Optional</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>6</td>
<td>Or</td>
<td>Or</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>7</td>
<td>Or</td>
<td>Alternative</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>8</td>
<td>Or</td>
<td>Mandatory</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>9</td>
<td>Alternative</td>
<td>Optional</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>10</td>
<td>Alternative</td>
<td>Or</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>11</td>
<td>Alternative</td>
<td>Alternative</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>12</td>
<td>Alternative</td>
<td>Mandatory</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>13</td>
<td>Mandatory</td>
<td>Optional</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>14</td>
<td>Mandatory</td>
<td>Or</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>15</td>
<td>Mandatory</td>
<td>Alternative</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

The analysis of the data presented in Table 6.2 leads us to the following:

1. An And-group of mandatory features and an Alternative-group can not be matched to each other, as shown by $\times$ in 12\textsuperscript{th} and last row,

2. $Source$ always satisfies $Target$ if both parties are willing to give selection rights
of variable features to each other which is represented by the √ in the 2nd last column,

3. If both Source and Target are selected by a single party then (less flexible features) $\not\Rightarrow$ flexible as shown by $\times$ in 4th and 5th column,

4. If both parties want to keep the selection rights of variable features with them then a Source does not satisfy a Target which is reflected by $\times$ in the last column.

The order of flexibility of features in classical feature diagrams from highest to lowest is Optional, Or-group, Alternative-group, and Mandatory [BSC10]. In our case, we use two categories of variable features: The first is selected by a requestor; the second is selected by a provider. The more options are provided to a requestor, the more flexible will be the service feature diagram. The order of flexibility from highest to lowest: Optional (R), Or-group (R), Alternative-group (R), Mandatory, Alternative-group (R), Or-group (P), and Optional (P).

Our analysis leads us to conclude that while matching an And-group of solitary and group features, flexible features satisfy the less flexible features. This is what service feature diagrams claimed to provide in Chapter 4. For example, we do not want to get a match between Alternative-group and an And-group of mandatory features. The requirements of an And-group having optional (R) features should only be satisfied by an And-group of optional (R) features, as shown in the 4th column of the 1st row in Table 6.2.

Table 6.3 shows the analysis of the graphical rules having an And-group of solitary and group features. The 2nd and 3rd columns in Table 6.3 show the sources $S_1$ and $S_2$ respectively, whereas the 4th column represents the Target of the matching rule. The And-group of $S_1$ and $S_2$ is represented by $S$ in Table 6.3. The result of the matching is shown in Columns 5 to 8. Columns 5 and 6 show the matching result of source to target, i.e., $S \vdash T$. The matching result of target to source, i.e., $T \vdash S$ is shown in columns 7 and 8. Columns 5 and 7 show the matching result when $S_1$ is
selected by the requestor, whereas columns 6 and 8 show the matching result when $S_1$ is chosen by the provider.

In Table 6.3, ✓ shows an entry which is already provided in Table 6.2, whereas ✓$^c$ represent an entry which can be obtained by rule combination, as provided in Section 6.7. Hence, to avoid the duplication in presenting the rules, matches that are represented by checkmarks with superscript in Table 6.3 are not provided in the graphical rules. A ✗ shows an entry that does not match due to a conflict of selection rights. A ✗ shows a conflict due to a matching of flexible features by less flexible ones.

Table 6.3: Analysis of Graphical Rules for the Combination of Solitary and Group Features

<table>
<thead>
<tr>
<th>No.</th>
<th>Source($S_1$)</th>
<th>Source($S_2$)</th>
<th>Target($T$)</th>
<th>$S \vdash T$</th>
<th>$T \vdash S$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$S_1(R)$</td>
<td>$S_1(P)$</td>
</tr>
<tr>
<td>1</td>
<td>Optional</td>
<td>Optional</td>
<td>Optional</td>
<td>✓$^c$</td>
<td>✓$^id$</td>
</tr>
<tr>
<td>2</td>
<td>Optional</td>
<td>Optional</td>
<td>Optional</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>3</td>
<td>Optional</td>
<td>Optional</td>
<td>Optional</td>
<td>✓$^id$</td>
<td>✗</td>
</tr>
<tr>
<td>4</td>
<td>Optional</td>
<td>Optional</td>
<td>Optional</td>
<td>✓$^id$</td>
<td>✗</td>
</tr>
<tr>
<td>5</td>
<td>Optional</td>
<td>Optional</td>
<td>Mandatory</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>6</td>
<td>Optional</td>
<td>Optional</td>
<td>Mandatory</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>7</td>
<td>Optional</td>
<td>Optional</td>
<td>Alternative</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>8</td>
<td>Optional</td>
<td>Optional</td>
<td>Alternative</td>
<td>✗</td>
<td>×</td>
</tr>
<tr>
<td>9</td>
<td>Optional</td>
<td>Optional</td>
<td>Alternative</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>10</td>
<td>Optional</td>
<td>Optional</td>
<td>Alternative</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>11</td>
<td>Optional</td>
<td>Or</td>
<td></td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>12</td>
<td>Optional</td>
<td>Or</td>
<td></td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>13</td>
<td>Optional</td>
<td>Or</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>14</td>
<td>Optional</td>
<td>Or</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>15</td>
<td>Optional</td>
<td>Mandatory</td>
<td>Optional</td>
<td>✓$^c$</td>
<td>✓$^c$</td>
</tr>
<tr>
<td>16</td>
<td>Optional</td>
<td>Mandatory</td>
<td>Optional</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>No.</th>
<th>Source($S_1$)</th>
<th>Source($S_2$)</th>
<th>Target($T$)</th>
<th>$S \vdash T$</th>
<th>$T \vdash S$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$S_1(P)$</td>
<td>$S_1(R)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$S_1(P)$</td>
<td>$S_1(R)$</td>
</tr>
<tr>
<td>17</td>
<td>Optional</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>18</td>
<td>Optional</td>
<td>Mandatory</td>
<td>Alternative (P)</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>19</td>
<td>Optional</td>
<td>Mandatory</td>
<td>Alternative (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>20</td>
<td>Optional</td>
<td>Mandatory</td>
<td>Or (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>21</td>
<td>Optional</td>
<td>Mandatory</td>
<td>Or (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>22</td>
<td>Optional</td>
<td>Alternative (P)</td>
<td>Optional (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>23</td>
<td>Optional</td>
<td>Alternative (P)</td>
<td>Optional (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>24</td>
<td>Optional</td>
<td>Alternative (R)</td>
<td>Optional (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>25</td>
<td>Optional</td>
<td>Alternative (R)</td>
<td>Optional (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>26</td>
<td>Optional</td>
<td>Alternative (P)</td>
<td>Mandatory</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>27</td>
<td>Optional</td>
<td>Alternative (R)</td>
<td>Mandatory</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>28</td>
<td>Optional</td>
<td>Alternative (P)</td>
<td>Optional (P)</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>29</td>
<td>Optional</td>
<td>Alternative (P)</td>
<td>Alternative (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>30</td>
<td>Optional</td>
<td>Alternative (R)</td>
<td>Alternative (P)</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>31</td>
<td>Optional</td>
<td>Alternative (R)</td>
<td>Alternative (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>32</td>
<td>Optional</td>
<td>Alternative (P)</td>
<td>Or (P)</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>33</td>
<td>Optional</td>
<td>Alternative (P)</td>
<td>Or (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>34</td>
<td>Optional</td>
<td>Alternative (R)</td>
<td>Or (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>35</td>
<td>Optional</td>
<td>Alternative (R)</td>
<td>Or (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>36</td>
<td>Optional</td>
<td>Or (P)</td>
<td>Optional (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>37</td>
<td>Optional</td>
<td>Or (P)</td>
<td>Optional (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>38</td>
<td>Optional</td>
<td>Or (R)</td>
<td>Optional (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>39</td>
<td>Optional</td>
<td>Or (R)</td>
<td>Optional (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>40</td>
<td>Optional</td>
<td>Or (P)</td>
<td>Mandatory</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>41</td>
<td>Optional</td>
<td>Or (R)</td>
<td>Mandatory</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>42</td>
<td>Optional</td>
<td>Or (P)</td>
<td>Alternative (P)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>43</td>
<td>Optional</td>
<td>Or (P)</td>
<td>Alternative (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>44</td>
<td>Optional</td>
<td>Or (R)</td>
<td>Alternative (P)</td>
<td>✓</td>
<td>×</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>No.</th>
<th>Source($S_1$)</th>
<th>Source($S_2$)</th>
<th>Target($T$)</th>
<th>$S \vdash T$</th>
<th>$T \vdash S$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$S_1(R)$ $S_1(P)$</td>
<td>$S_1(R)$ $S_1(P)$</td>
</tr>
<tr>
<td>45</td>
<td>Optional</td>
<td>Or (R)</td>
<td>Alternative (R)</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>46</td>
<td>Optional</td>
<td>Or (P)</td>
<td>Or (P)</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>47</td>
<td>Optional</td>
<td>Or (P)</td>
<td>Or (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>48</td>
<td>Optional</td>
<td>Or (R)</td>
<td>Or (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>49</td>
<td>Optional</td>
<td>Or (R)</td>
<td>Or (R)</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>50</td>
<td>Or</td>
<td>Mandatory</td>
<td>Optional (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>51</td>
<td>Or</td>
<td>Mandatory</td>
<td>Optional (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>52</td>
<td>Or</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>53</td>
<td>Or</td>
<td>Mandatory</td>
<td>Alternative (P)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>54</td>
<td>Or</td>
<td>Mandatory</td>
<td>Alternative (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>55</td>
<td>Or</td>
<td>Mandatory</td>
<td>Or (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>56</td>
<td>Or</td>
<td>Mandatory</td>
<td>Or (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>57</td>
<td>Or</td>
<td>Or (R)</td>
<td>Or (P)</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>58</td>
<td>Or</td>
<td>Or (R)</td>
<td>Or (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>59</td>
<td>Or</td>
<td>Or (P)</td>
<td>Or (P)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>60</td>
<td>Or</td>
<td>Or (P)</td>
<td>Or (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>61</td>
<td>Alternative</td>
<td>Mandatory</td>
<td>Optional (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>62</td>
<td>Alternative</td>
<td>Mandatory</td>
<td>Optional (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>63</td>
<td>Alternative</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>64</td>
<td>Alternative</td>
<td>Mandatory</td>
<td>Alternative (P)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>65</td>
<td>Alternative</td>
<td>Mandatory</td>
<td>Alternative (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>66</td>
<td>Alternative</td>
<td>Mandatory</td>
<td>Or (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>67</td>
<td>Alternative</td>
<td>Mandatory</td>
<td>Or (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>68</td>
<td>Alternative</td>
<td>Alternative (R)</td>
<td>Alternative (P)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>69</td>
<td>Alternative</td>
<td>Alternative (R)</td>
<td>Alternative (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>70</td>
<td>Alternative</td>
<td>Alternative (P)</td>
<td>Alternative (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>71</td>
<td>Alternative</td>
<td>Alternative (P)</td>
<td>Alternative (P)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>72</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>Optional (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Continued on next page
Table 6.3 – continued from previous page

<table>
<thead>
<tr>
<th>No.</th>
<th>Source ($S_1$)</th>
<th>Source ($S_2$)</th>
<th>Target ($T$)</th>
<th>$S \vdash T$</th>
<th>$T \vdash S$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$S_1$</td>
<td>$S_1(P)$</td>
<td>$S_1(R)$</td>
</tr>
<tr>
<td>73</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>Optional (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>74</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>Or (P)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>75</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>Or (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>76</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>Alternative (P)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>77</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>Alternative (R)</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>78</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

The data presented in Table 6.3 leads us to the following conclusions:

1. An And-group of mandatory features and an Alternative-group can not be matched.

2. An And-group of mandatory and optional features or an And-group of mandatory and Group features do not satisfy Alternative-groups. There is only one exception to this, mentioned in number 18 of Table 6.3, where an And-group of optional (R) and mandatory features satisfies an Alternative-group (P), but please note that this rule is only applicable for single mandatory feature, as depicted in the rule in Fig. 6.6.

3. If a requestor wants to keep the choice of a group-feature or if a provider leaves the selection rights of a group-feature with the requestor, we cannot match an And-group of mixed choices and vice versa.

4. A group-feature does not satisfy the requirements of an And-group of that group-feature and solitary features. Groups at both ends should have the same type, i.e., selection rights should only be given either to requestor or provider.

5. A requirement of an And-group of mandatory features can not be satisfied by an And-group of mixed choices, and vice versa.
6. An Alternative-group does not satisfy the requirements of an And-group of mixed choices. There is one exception to this shown in No. 19 in Table 6.3, where Alternative (R) satisfies an And-group (P).

6.10 Application of Graphical Rules

To check if the requirements are realised by the provided models, we have to match them [HC07]. In this Section, we show the matching of feature diagram by using the graphical rules. As shown in Section 4.4, we have a set of provided models and one required one. We apply the matching rules provided to match the provided models with the requirements.

To apply a rule, we select the description and start looking for a subtree where the description differs from the requirements, i.e., a subtree $S$ of the description that differs in selection rights or in relevance from the subtree $S'$ in the requirements. Once we find $S$, then we use the graphical rules to check if $S \vdash S'$, and then replace $S$ in the description by $S'$. We move on by replacing all $S$ of the description with their respective $S'$ of the requirement. After replacing all $S$ by $S'$, if the description is identical to the requirement then we say that the description satisfies the requirement. So, our technique reduces the matching process into the number of small transformations, where each transformation is performed by the application of a graphical rule.

To see the application of graphical rules in the accompanying example, we have to consider the application of matching rules in three different parts, i.e., wherever we have shown $\vdash$ in Fig. 1.1. First, let us check whether Supplier 1 and Supplier 2 of Fig. 4.12(b) and 4.12(c) satisfy the Manufacturer’s requirement of the Entertainment system of Fig. 4.12(a).

The deviating subtrees in both Fig. 4.12(b) and 4.12(c) differ in selection rights of the Alternative-group underneath Radio and CDPlayer, respectively. The Alternative to Alternative matching shown in Table 6.2 says that an Alternative-group of requestor’s choice satisfies an Alternative-group with provider’s choice, i.e., $S \vdash S'$. 
Hence we replace $\mathcal{S}$ by the respective $\mathcal{S}'$ in Fig. 4.12(b) and 4.12(c). We do not find any more deviating subtrees in Fig. 4.12(b) and 4.12(c), so using the rules shown in the Fig. 6.29 and in 6.31 over Fig. 4.12(b) and 4.12(c) gives us the Fig. 6.35.

![Feature Diagram](image)

Figure 6.35: Merged Feature Diagram of Supplier 1 and Supplier 2

Again checking the merged feature diagram of Fig. 6.35 for deviating subtrees, we observe that optional (R) features *Radio* and *CDPlayer* of *Entertainment* come in the form of an Or-group (R) in Fig. 4.12(a). The matching result of optional (R) to Or-group (R) in Table 6.2 says that optional features of requestor’s choice satisfies the Or-group of requestor’s choice. Hence the application of the rule shown in Fig. 6.4 over Fig. 6.35 transforms it further to show the Manufacturer’s requirements of Fig. 4.12(a). We conclude that both suppliers of Fig. 4.12(b) and 4.12(c) can jointly satisfy the Manufacturer’s requirement of an *Entertainment* shown in Fig. 4.12(a).

Now, let us apply the matching rules to check whether the service feature diagram shown in Fig. 4.13(b) satisfies the Manufacturer’s requirement of *SeatingCapacity* shown in Fig. 4.13(a). The first deviating subtree in Fig. 4.13(b) is the Alternative-group underneath the *Rear* feature. The matching result of Alternative to Alternative in Table 6.2 says that an Alternative-group of requestor’s choice satisfies the requirements of the same group with provider’s choice, i.e., $\mathcal{S} \vdash \mathcal{S}'$. So, we replace $\mathcal{S}$ by $\mathcal{S}'$. Now the transformed service feature diagram of 4.13(b) shows the Manufacturer’s requirements of an *Entertainment* system shown in Fig. 4.13(a). Supplier 3 satisfies the Manufacturer’s requirements shown in Fig. 4.13(a).

Let us see whether the Car Manufacturer shown in the Fig. 4.11 satisfies the Client’s requirement in Fig. 4.10. Our observation says that features that come in the form of an Or-group underneath the *Car* feature in Fig. 4.11 are required as an
And-group of mandatory features in Fig. 4.10. Table 6.2 says that an Or-group of requestor’s choice satisfies an And-group of mandatory features, i.e., $S \vdash S'$. So, we replace $S$ by the corresponding $S'$ in Fig. 4.10.

The next deviating subtree in Fig. 4.11 is an optional (R) feature Rear under SeatingCapacity that comes as a mandatory subfeature of SeatingCapacity in the requirements. Again, the matching result of optional to mandatory features shown in Table 6.2 says that optional (R) features satisfy mandatory requirements. So, we change the relevance of the optional feature underneath the Rear feature to mandatory. Now, a transformed feature diagram of the Car Manufacturer of Fig. 4.11 depicts the Client’s requirements shown in Fig. 4.10. The Car Manufacturer shown in the Fig. 4.11 satisfies the Client’s requirements of Fig. 4.10.

### 6.11 Summary

In this chapter, we provided graphical matching rules that are used to check whether available descriptions satisfy a requirement or not. We also presented the rules for the cases, where we may have to merge feature diagrams or features during a match. We tried to use generic rule wherever possible. A single generic rule captures multiple graphical rules. We proceeded by analysing the graphical rules and then applying matching rules over the running example presented in Chapter 4.
Chapter 7

Formalisation and Verification of Matching

In this chapter, we discuss the formalisation of the matching process of service feature diagrams using the inference system of Linear Logic [Gir87, Tro92]. From the matching of feature diagrams, we mean to state formally, when the requirements are said to be satisfied by the descriptions, where requirements and descriptions are modelled by service feature diagrams.

This chapter is arranged as follows: In Section 7.1, we discuss the definitions of matching requestor and provider feature diagrams. We also elaborate the connection of those definitions. In Section 7.2, we provide the verification of graphical rules by using the inference system of CLL. We show that the graphical rules are correct with respect to the inference rules of CLL. In Section 7.3, we discuss the verification of the matching process of requestor and provider feature diagrams, where we discuss that the application of graphical rules over a service feature diagram are independent of the context. Section 7.4 concludes the chapter.

7.1 Matching of Service Feature Diagrams

Based on the discussions so far, there are 4 possible definitions of matching requestor and provider feature diagrams:
(M1) $PFD \Rightarrow^* RFD$: The iterative application of graphical matching rules over provider feature diagram $PFD$ leads to requestor feature diagram $RFD$.

(M2) $LF(PFD) \vdash LF(RFD)$: The linear formula of provider feature diagram $LF(PFD)$ derives the linear formula of requestor feature diagram $LF(RFD)$.

(M3) $Inst(PFD) \Rightarrow^* Inst(RFD)$: The set of instance diagrams of requestor feature diagram $RFD$ can be obtained by the application of matching rules over the set of instance diagrams of provider feature diagram $PFD$.

(M4) $IF(PFD) \vdash IF(RFD)$: The set of instance formulas of provider diagram $PFD$ derives the set of instance formulas of requestor feature diagram $RFD$.

To elaborate the above mentioned definitions let us consider a requestor, interested to book a flight and a hotel, who finds a provider offering the requirements shown in Fig. 7.1. The provider gives the selection of an Or-group of Hotel and Flight to the requestor, who marks these feature as mandatory. In principle, this offer should be able to satisfy the requestors’ demands captured in Fig. 7.1. Let us check the matching of feature diagrams using the definitions above.

![Figure 7.1: Travel Agent (left) and Requestor (right)](image)

7.1.1 Definitions of Matching

**M1:** The definition M1 says that $PFD \Rightarrow^* RFD$ where $PFD$ and $RFD$ are the feature diagrams of provider and requestor, respectively. That means, an offer satisfies the requirements if applications of matching rules over $PFD$ transform it into $RFD$. As stated in Table 6.2 and 6.3, an Or-group (R) satisfies an And-group of mandatory features. So, the application of the multi-rule shown in Fig.
6.18 transforms the provider feature diagram into requestor feature diagram shown on the right of Fig. 7.1.

**M2:** The definition M2 says that $LF(PFD) \vdash LF(RFD)$ where $LF(PFD)$ and $LF(RFD)$ are the linear formulas of $PFD$ and $RFD$, respectively. Using the encoding shown in Table 5.1, the linear formulas of $PFD$ and $RFD$ are $R \otimes (R \otimes (R \rightarrow ((H \& F \& (H \otimes F)) \otimes P)))$ and $R \otimes (R \otimes (R \rightarrow (H \otimes F \otimes P)))$, respectively. The deduction of the requestor formula from the provider formula is shown below.

\[
\begin{align*}
L \& id & \frac{H \& F \vdash H \otimes F}{F \& (H \otimes F) \vdash H \otimes F} \\
L \& id & \frac{H \& F \& (H \otimes F) \vdash H \otimes F}{H \& F \& (H \otimes F), P \vdash H \otimes F \otimes P} \\
R \otimes id & \frac{R \vdash R}{R \rightarrow ((H \& F \& (H \otimes F)) \otimes P) \vdash R \rightarrow (H \otimes F \otimes P)} \\
L \rightarrow id & \frac{R \rightarrow R}{(H \& F \& (H \otimes F)) \otimes P \vdash R \rightarrow (H \otimes F \otimes P)}
\end{align*}
\]

The above deduction shows that $PFD$ satisfies $RFD$.

**M3:** The definition M3 says that $Inst(PFD) \Rightarrow^\ast Inst(RFD)$ where $Inst(PFD)$ and $Inst(RFD)$ are the sets of the instance diagrams of requestor and provider, respectively. Because we have no matching rule considering relevance less edges (which we will call solid edges in the current discussion) in a feature diagram, so the obtaining an $Inst(PFD)$ from $Inst(RFD)$ may have more cases then just the application of matching rule over $Inst(PFD)$. To explain it further, let us first enlist the cases which we may come across, while trying to get $Inst(PFD)$ from $Inst(RFD)$. In the following list $id_R \in Inst(RFD)$ whereas $id_P \in Inst(PFD)$

1. when the features connected through solid edges in $id_P$ are also represented by the solid edges in $id_R$,
2. when the features connected through dashed edges in $id_P$ are also represented by the dashed edges in $id_R$,
3. when the features connected through solid edges in $id_p$ are represented by the dashed edges in $id_R$

Please note, we do not consider a case where $id_p$ consists of the dashed edges and same features are connected through solid edges in $id_R$, because if both parties (requestor and provider) want to keep the choice of variable features with them then their models do not match (please see Chapter 6, for details). Let us now discuss the above list.

When the features which are connected through the solid edges in $id_p$ are also connected by the solid edges in $id_R$ then we obtain $Inst(RFD)$ from $Inst(PFD)$ if $Inst(PFD) \supseteq Inst(RFD)$. In other words, if we get the solid edges everywhere then we check, if the set of instance diagrams of requestor are subset of the set of the instance diagrams of provider.

If the features are connected through dashed in both sets ($Inst(PFD)$ and $Inst(RFD)$) then we apply matching rules over $Inst(PFD)$ and see if it can be transformed to $Inst(RFD)$. The reason of applying matching rule is the existence of instance diagrams with relevance in both sets, so we can find an applicable matching rule in catalogue of matching rules.

In case, if the features contained by $id_R$ are connected through the solid edges are connected through the dashed edges in $id_p$, we check if the features connected the dashed edges in $id_R$ exists either separately of in the form of group in $id_p$. There is only one exception to this category, the dashed edges forming an Alternative-group in $id_R$ must only exist separately in the form of solid edges in $id_p$, due to the nature of Alternative-group—more than one features can not be selected from this group.

We can also combine the above mentioned cases while trying to get $Inst(RFD)$ from $Inst(PFD)$. We use the rules shown in Table 5.3 to derive the instance diagrams from the feature diagrams of Fig. 7.1. Fig. 7.2(a) shows the instance diagrams of $PFD$ and Fig. 7.2(b) the instance diagram of $RFD$. As we have solid edges everywhere hence we check $Inst(PFD) \supseteq Inst(RFD)$. It is clear from Fig. 7.2 that the instance diagram of the requestor feature diagram exists in the instance diagrams
7.1 Matching of Service Feature Diagrams

![Instance Diagrams of Fig. 7.1](image)

Figure 7.2: Instance Diagrams of Fig. 7.1

of the provider feature diagram, i.e., $\text{Inst}(PFD) \Rightarrow \text{Inst}(RFD)$.

**M4:** The last definition of matching $M4$ states that $IF(PFD) \vdash IF(RFD)$ where $IF(PFD)$ and $IF(RFD)$ are the sets of instance formulas of provider and requestor feature diagrams, respectively. From $IF(PFD) \vdash IF(RFD)$, mean that individually all $if_{Rj} \in IF(RFD)$ can be derived from a formula $if_{Pi} \in IF(PFD)$.

Using the notions discussed in Chapter 5, instance-formulas of the provider feature diagram are derived below:

\[
\begin{align*}
\text{id} & \vdash H \vdash H \\
L \& & \frac{H & F & (H \otimes F) \vdash H}{H \& F \& (H \otimes F)} \\
R \otimes & \frac{P \vdash P}{P \vdash P}
\end{align*}
\]

\[
\begin{align*}
\text{id} & \vdash R \vdash R \\
L & \frac{R \vdash R}{R \vdash R}
\end{align*}
\]

\[
\begin{align*}
\text{id} & \vdash (H & F & (H \otimes F)) \otimes P \vdash H \otimes P \\
L \otimes & \frac{R, R \vdash ((H & F & (H \otimes F)) \otimes P)}{R \& (R \vdash ((H & F & (H \otimes F)) \otimes P)) \vdash R \& H \otimes P}
\end{align*}
\]

\[
\begin{align*}
\text{id} & \vdash F \vdash F \\
L \& & \frac{F \vdash F}{F \vdash F}
\end{align*}
\]

\[
\begin{align*}
\text{id} & \vdash R \vdash R \\
L & \frac{R \vdash R}{R \vdash R}
\end{align*}
\]

\[
\begin{align*}
\text{id} & \vdash (H & F & (H \otimes F)) \otimes P \vdash F \otimes P \\
L \otimes & \frac{R, R \vdash ((H & F & (H \otimes F)) \otimes P)}{R \& (R \vdash ((H & F & (H \otimes F)) \otimes P)) \vdash R \& F \otimes P}
\end{align*}
\]

\[
\begin{align*}
\text{id} & \vdash R \vdash R \\
L & \frac{R \vdash R}{R \vdash R}
\end{align*}
\]

\[
\begin{align*}
\text{id} & \vdash (H & F & (H \otimes F)) \otimes P \vdash F \otimes P \\
L \otimes & \frac{R, R \vdash ((H & F & (H \otimes F)) \otimes P)}{R \& (R \vdash ((H & F & (H \otimes F)) \otimes P)) \vdash R \& F \otimes P}
\end{align*}
\]

\[
\begin{align*}
\text{id} & \vdash R \vdash R \\
L & \frac{R \vdash R}{R \vdash R}
\end{align*}
\]

\[
\begin{align*}
\text{id} & \vdash (H & F & (H \otimes F)) \otimes P \vdash F \otimes P \\
L \otimes & \frac{R, R \vdash ((H & F & (H \otimes F)) \otimes P)}{R \& (R \vdash ((H & F & (H \otimes F)) \otimes P)) \vdash R \& F \otimes P}
\end{align*}
\]
The instance formula of a requestor feature diagram can also be derived as:

\[
\begin{align*}
\{R \otimes H \otimes P, R \otimes F \otimes P, R \otimes H \otimes F \otimes P\} & \vdash \{R \otimes H \otimes F \otimes P\}, \\
\text{IF}(PFD) & \vdash \text{IF}(RFD),
\end{align*}
\]

Let us now discuss the connection of the above mentioned definitions of matching.

### 7.1.2 Relationships

In general, the above-mentioned definitions are connected to each other as depicted in the figure below.

\[
\begin{align*}
\text{PFD} & \implies^* \text{RFD} & \implies_1 & \text{Inst}(PFD) \implies^* \text{Inst}(RFD) \\
\text{Conj} 1 & \implies 4 & \implies_4 & \text{Inst}(PFD) \implies^* \text{Inst}(RFD) \\
\text{LF}(PFD) & \implies \text{LF}(RFD) & \implies_3 & \text{IF}(PFD) \implies \text{IF}(RFD) \\
\text{Conj} 2 & \implies 2 & \implies_2 & \text{IF}(PFD) \implies \text{IF}(RFD)
\end{align*}
\]
Implies 1: Let us first discuss *Implies 1* which states that if we get the steps of transformations leading *PFD* to *RFD* then *Inst(RFD)* can be obtained from *Inst(PFD)*. We can get *Inst(PFD)* and *Inst(RFD)* from *PFD* and *RFD*, respectively, using the rules shown in Table 5.3. As stated in Chapter 6, *PFD* satisfies *RFD* if *PFD* provides at least the flexibility required by *RFD*. In other words, if *PFD* can be transformed to *RFD* then the set of instance diagrams of *RFD*, *Inst(RFD)*, can be obtained from the set of instance diagrams of *PFD*, *Inst(PFD)*.

Implies 2: As we state in *Implies 2*, if the set of instance diagrams of *RFD* can be obtained from the the set of instance diagrams of *PFD*, the set of instance formulas of *PFD* derives the set of instance formulas of *RFD*. In Chapter 5, we have shown the encoding of instance diagrams into linear formulas

\[
ID \xrightarrow{\text{Encoded}} LF
\]

where *id* and *LF* are the instance diagram and linear formula of a service feature diagram, respectively. If *Inst(PFD)* and *Inst(RFD)* are the sets of the instance diagrams of provider and requestor feature diagram, respectively, then

\[
\text{Inst}(PFD) \xrightarrow{\text{Encoded}} LF(\text{Inst}(PFD)) = LI(PFD)
\]

\[
\text{Inst}(RFD) \xrightarrow{\text{Encoded}} LF(\text{Inst}(RFD)) = LI(RFD)
\]

evaluate them into corresponding linear logical formulas, where the set of linear encodings of instance diagrams of a service feature diagram *fd* is represented by *LI(fd)*. We have also discussed in Chapter 5 that the set of linear instances and the set of instance formulas of a service feature diagram are equivalent. Hence,

\[
LI(PFD) \implies LI(RFD)
\]

\[
\text{IF}(PFD) \vdash \text{IF}(RFD)
\]

which leads to *Implies 2*.

Implies 3: Let us discuss *Implies 3*: In general, *LF* \(1 \vdash LF* \(2\) means that all formulas derivable from *LF* \(2\) are also derivable from *LF* \(1\), but not necessarily the other
way round. That means

\[ F_1 \vdash F_2 \land F_2 \vdash f \implies F_1 \vdash f \]

We already discussed in Chapter 5 that \( LF(fd) \vdash_{\alpha} if_i \), where \( LF(fd) \) and \( if \) are the linear and instance formulas of a service feature diagram \( fd \), respectively, and \( \vdash_{\alpha} \) is the restricted deduction. We know that \( F \vdash f \implies F \vdash_{\alpha} f \), if \( F \) and \( f \) are the linear and instance formulas, respectively, obtained by the encoding rules discussed in Chapter 5. From this it follows that

\[
\begin{align*}
LF(FD) \vdash LF(RD) \\
IF(FD) \vdash IF(RD)
\end{align*}
\]

**Implies 4:** We will demonstrate Implies 4 in Section 7.3, where we will show that \( PFD \implies^* RD \) implies \( LF(PFD) \vdash LF(RFD) \). For example, let \( L \) and \( R \) be the left and right hand sides of a matching rule \( m \), and \( FD \) a service feature diagram which is transformed to \( FD' \) by the application of the graphical rule \( m \), as shown in the left of the figure below.

\[
\begin{align*}
L & \quad \text{applies} \quad R \\
FD & \quad \Downarrow \quad FD'
\end{align*}
\]

We will check the correctness of the rule \( m \) by providing the deduction of \( LF(L) \vdash LF(R) \), where \( LF(L) \) and \( LF(R) \) are the linear formulas of the left and right sides of the matching rule \( m \), as shown in top right of the figure above, and \( LF(FD) \) and \( LF(FD') \) are the linear formulas of the feature diagrams before and after the application of rule \( m \). The verification of the matching process is provided using the inference system of Linear Logic to prove the deduction of \( LF(FD) \vdash LF(FD') \).

Apart from Implies 1, Implies 2, Implies 3, and Implies 4, we have conjectures for additional implications shown as Conj 1 and Conj 2. We believe that the following are true. Let us discuss the conjunctures.
Conjuncture 1: If the linear encoding of PFD derives the linear encoding of RFD, there exists a sequence of transformations from PFD to RFD.

\[ \text{PFD} \implies^* \text{RFD} \]

\[ LF(\text{PFD}) \vdash LF(\text{RFD}) \]

where \( LF(\text{PFD}) \) and \( LF(\text{RFD}) \) are the linear formulas of provider and requestor feature diagrams, respectively.

Conjuncture 2: If the set of instance formulas of PFD derives the instance formulas of RFD, the set of instance diagrams of RFD can be obtained from the set of instance diagrams of PFD.

\[ \text{Inst}(\text{PFD}) \implies^* \text{Inst}(\text{RFD}) \]

\[ IF(\text{PFD}) \vdash IF(\text{RFD}) \]

where \( \text{Inst}(\text{PFD}) \) and \( \text{Inst}(\text{RFD}) \) are the sets of instance diagrams of PFD and RFD, respectively, and \( IF(\text{PFD}) \) and \( IF(\text{RFD}) \) are the sets of instance formulas of PFD and RFD, respectively.

Let us now conclude the discussion by analysing what we have discussed so far.

7.1.3 Analysis

As discussed above, we are dealing with two definitions of matching. One is based on instance diagrams, which states that if the set of instance diagrams of RFD can be obtained from the set of instance diagrams of PFD, then PFD satisfies RFD. The second is based on the graphical rules, which states that if we get a sequence of transformations of PFD such that we end up with RFD, PFD satisfies RFD.

These definitions are not equivalent, rule-based matching is more demanding than the instance-based matching. In other words, all those requirements that can be satisfied by the graphical rules are also satisfied by the instance-based matching,
but not necessarily the other way round. For example, let us consider a case of four subfeatures $a, b, c,$ and $d$ making two Or-groups (R) in the requirements, whereas description marks those features in an Or-group (R), as shown in Fig. 7.3.

![Figure 7.3: Description (Left) and Requirements (Right)](image)

We do not find a matching rule in the catalogue where an Or-group (R) satisfies the And-group of two Or-groups (R). Furthermore, $LF(PFD) \not\subseteq LF(RFD)$ where $LF(PFD)$ and $LF(RFD)$ are the linear formulas of the descriptions and requirements, as shown in the left and the right of Fig. 7.3, respectively. Hence, requirements can not be satisfied by using rule-based matching.

On the other hand, to check the instance-based matching for the diagrams shown in Fig. 7.3, let us first generate their instance diagrams. The set of instance diagrams of $PFD$ and $RFD$ are shown in Fig. 7.4 with the labels $Inst(PFD)$ and $Inst(RFD)$, respectively. As we have solid edges in all the instance diagrams shown above, so we will check $Inst(PFD) \supseteq Inst(RFD)$. We noticed that all instance diagrams of $Inst(RFD)$ are included in $Inst(PFD)$ so we conclude that
7.2 Verification of Graphical Rules

Inst(PFD) ⊇ Inst(RFD), i.e., instance-based matching satisfies the requirements shown in Fig. 7.3.

We say that the graphical rules are not complete with respect to the instance-based matching of feature diagrams, i.e., rule-based matching is more demanding than instance-based matching. In other words, diagrams that are matched by the graphical rules are also satisfied by the instance-based matching, but not necessarily the other way round.

To discuss the correctness of the graphical rules, we provide their verification in the following section, using the theorem proving of CLL.

7.2 Verification of Graphical Rules

To verify the graphical rules, we generate the proofs showing the deduction of the linear formulas of the right side of the graphical rule from the left side. The result will be a proof of a sequent of the form:

\[ A \otimes (a \rightarrow (B_1 \star B_2 \cdots \star B_m)) \vdash A \otimes (a \rightarrow (B'_1 \star B'_2 \cdots \star B'_n)) \]

where \( \star \in \{ \otimes, \& , \oplus \} \) and \( m \geq 1 \leq n \). In formulas from here onwards, capital letters \( A, B, \ldots \) refer to the conjuncts of literals, lower-case letters \( a, b, \ldots \) are literals referring to the features of a feature diagram, and the Greek letters \( \alpha, \beta, \ldots \) refer to the features modelled by diamond symbols in diagrams. We will also use mathematical letters \( A, B, \ldots \) which will refer to arbitrary linear logical formulas.

The deduction of the rules shows that they are correct with respect to the inference system of Linear Logic (please see Chapter 2 for the inference rules). Our strategy for the deduction is to derive the innermost operator first using the identity rule. From the innermost operator we refer to the segment of a linear formula that exists in the innermost braces. Once we get the innermost operator, we move on by adding the next proposition to the formulas on the left and right of the sequent by applying the relevant inference rules until the formulas show the required deriva-
7.2 Verification of Graphical Rules

While applying the inference, we may come across situations, where we have to select one from a pair of left and right rules. We will always go for the binary rule due to the requirements of inference rules shown in the sequent calculus.

For example, if we have to choose a rule from $L\otimes$ or $R\otimes$, then we will always apply $R\otimes$ before $L\otimes$. We will never get a deduction step, where we have to choose from $L\otimes$ and $R\otimes$, while having more than one formulas in the antecedent. This is because of the balance of the operators used in the cedents. So after applying $R\otimes$, we will be left with two formulas in the antecedent, which then joined by using $L\otimes$.

In the case when we do not have a pair of operators to choose from, we may have to choose one from two unary rules. For example, if we have to choose a rule from $L\&$ or $R\oplus$, we apply any rule. The order of application of these rules does not effect the deduction.

While providing the deduction of the graphical rules, we have noticed that we have to repeat the patterns in different proofs. To avoid repeating such patterns, we define inference figures [RKM06]. An inference figure is a sequent derived from the inference rules of CLL. Before going to discuss the verification of graphical rules, let us first provide the inference figures which we will use in the proofs.

7.2.1 Inference Figures

The benefit of using inference figures is two-fold: first, it avoids the duplication of pattern in proofs; secondly, it decreases the length of proofs which makes them more easy to understand. We have generated the inference figures by analysing the repeated patterns in proofs. Let us provide the inference figures and derive their respective proofs:

**Inference Figure 7.1** (IF1) The general form of this figure is

$$f_1, \ldots, f_j \vdash \bigotimes_{k=i}^j (f_k \oplus f_k^\perp)$$
where $f$ is a literal representing a feature and $v \in \{0, 1\}, f^o = f^1, f^1 = f, i \leq n \geq j \wedge j \geq i$ and $n$ is maximum index used in the rule.

We use $IF1$ where we want to get the (multiplicative) conjunction of (additive) disjunctions on the right side of a sequent. The proof of this figure is shown below:

$$
\begin{align*}
\text{id} & \quad f^v_i \vdash f^v_i \\
\text{id} & \quad f^v_i \vdash f^v_i \oplus f^v_i \\
\text{id} & \quad f^v_i \vdash (f^v_i \oplus f^v_i) \otimes (f^v_{i+1} \oplus f^v_{i+1}) \\
\text{id} & \quad f^v_i \vdash (f^v_i \oplus f^v_i) \otimes (f^v_{i+1} \oplus f^v_{i+1}) \\
\text{id} & \quad f^v_i \vdash \cdots \\
\text{id} & \quad f^v_i, f^v_{i+1}, \ldots, f^v_j \vdash \bigotimes_{k=i}^j (f^v_k \oplus f^v_k)
\end{align*}
$$

In the above proof, $s \in \{0, 1\} \wedge s \neq v$ so the last step of the above sequent becomes:

$$
\begin{align*}
\text{id} & \quad f^v_i, f^v_{i+1}, \ldots, f^v_j \vdash \bigotimes_{k=i}^j (f^v_k \oplus f^v_k)
\end{align*}
$$

**Inference Figure 7.2 (IF2)** The general form of this figure is

$$
\begin{align*}
& r \otimes (r \otimes (r \rightarrow (A))) \vdash r \otimes (r \otimes (r \rightarrow (B)))
\end{align*}
$$

where $A$ and $B$ are arbitrary linear formulas of subtrees having $r$ as root and $A \vdash B$.

We use this inference figure where we know $A \vdash B$ and want to add the encoding of the root on both sides of the sequent. The proof of this figure is shown below:

$$
\begin{align*}
\text{id} & \quad \frac{r \vdash r}{A \vdash B} \\
\text{id} & \quad \frac{r \vdash r}{r, r \rightarrow A \vdash \overline{B}} \\
\text{id} & \quad \frac{r \vdash r}{r \rightarrow A \vdash r \otimes (r \rightarrow B)} \\
\text{id} & \quad \frac{r \vdash r}{r \otimes (r \rightarrow A) \vdash r \otimes (r \otimes (r \rightarrow B))} \\
\text{id} & \quad \frac{r \vdash r}{r \otimes (r \rightarrow A) \vdash r \otimes (r \otimes (r \rightarrow B))}
\end{align*}
$$

Apart from the lemmas described above, we also use a derived identity rule having the general for as

$$
\begin{align*}
\frac{f^v_i, f^v_{i+1}, \ldots, f^v_j \vdash \bigotimes_{k=i}^j f^v_k}{f^v_i, f^v_{i+1}, \ldots, f^v_j \vdash \bigotimes_{k=i}^j f^v_k}
\end{align*}
$$
, where \( f \) is a literal representing the feature of a service feature diagram and \( v \in \{0, 1\}, f^0 = f^\perp, f^1 = f \), \( i \leq n \geq j \land j \geq i \) and \( n \) is the maximum index used in the rule and \( f^v_i \) on both sides of the sequent must not be pairwise distinct.

We use this identity where we want to get the multiplicative conjunction of literals on the right side of a sequent. The proof of this figure is shown below:

\[
\begin{align*}
\text{id} & \quad \frac{f^v_i \mid f^v_i}{R \otimes f^v_i, f^v_{i+1} \mid f^v_i \otimes f^v_{i+1}} \\
\text{id} & \quad \frac{f^v_{i+1} \mid f^v_{i+1}}{R \otimes f^v_{i-1}, f^v_{i} \mid f^v_{i-1} \otimes f^v_{i}} \\
& \quad \vdots \\
\text{id} & \quad \frac{f^v_n \mid f^v_n}{R \otimes f^v_i, f^v_{i+1}, \ldots, f^v_n \mid \bigotimes_{k=i}^n f^v_k}
\end{align*}
\]

The horizontal dots in the above proof represents the series of sequents (and serve the same purpose in all deductions), whereas the vertical dots are used to show the repetitive applications of a rule which comes after vertical dots. For example, in the above proof, vertical dots show the multiple applications of rule \( R \otimes \).

The application of these inference figures is very simple: We analyse the sequents which we get in each deduction step by comparing them with the staring sequents of the figures. After that we check, if the ending sequent of the figure contributes towards the deduction then we apply that inference figure. The inference figures mentioned above can be used with any linear logical formulas, i.e., they are general in nature. Here, we use them to show the verification of graphical rules.

### 7.2.2 Verification

Now, we elaborate the use of inference figures and the inference rules of CLL while providing the proofs of the graphical rules. Let us show the verification of a matching rule where an And-group of mandatory features satisfies the And-group of Optional (P) Features, the graphical form of this rule is shown below:
As said earlier, the proof of the rule shown above should end with

\[ a \otimes (a \otimes (a \rightarrow (\bigotimes_{i=1}^{n} b_i))) \vdash a \otimes (a \otimes (a \rightarrow (\bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp)))) \]

where linear formulas before and after \( \vdash \) are obtained by encoding the left and the right sides of the rule, respectively. The innermost operator of the above derivation is:

\[ \bigotimes_{i=1}^{n} b_i \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp) \]

Analysis of the above sequent states, to get the inner most operator, we have to apply \( IF1 \) where \( i = 1, j = n, f = b, \) and \( \forall f \ v = 1: \)

\[ IF1 \]

\[
\begin{array}{c}
\vdash \\
\bigotimes_{i=1}^{n} b_i \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp) \\
\end{array}
\]

We have thus constructed the right part of the innermost operator. The left part can be obtained by the repetitive use of \( L\otimes \) on the above result:

\[ L\otimes \]

\[
\begin{array}{c}
\vdash \\
\bigotimes_{i=1}^{n} b_i \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp) \\
\end{array}
\]

At the next level, we need to add \( a \) on both sides of above sequent, which can be obtained by the use of \( IF2 \). The application of \( IF2 \) on the above result yields:

\[ IF2 \]

\[
\begin{array}{c}
\vdash \\
a \otimes (a \otimes (a \rightarrow (\bigotimes_{i=1}^{n} b_i))) \vdash a \otimes (a \otimes (a \rightarrow (\bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp)))) \\
\end{array}
\]

Please note, the final step of deduction contains the encoding of the service feature diagrams on the left and the right side of the matching rule. The above derivation is correct with respect to the inference rules of the sequent calculus of Linear Logic.

In short, the overall strategy of deduction will remain the same for each derivation, i.e., we start by the identity sequent or an inference figure and move on by applying applicable rules or inference figures. Vertical dots in the deduction mean that the rule mentioned in the deduction step has been applied multiple times (\( < n \) times), whereas horizontal dots show the sequence of sequents.
Let us now discuss the verification of one more matching rule, where an And-group of Optional Features (R) satisfies an Alternative-group (R). The diagrammatic representation of this rule is shown below which is followed by its proof:

Let us now discuss the verification of one more matching rule, where an And-group of Optional Features (R) satisfies an Alternative-group (R). The diagrammatic representation of this rule is shown below which is followed by its proof:

\[
\begin{align*}
\text{id} & \quad Satisfies \\
\quad id & \quad Satisfies
\end{align*}
\]

\[
\begin{align*}
\text{id} & \quad b_1, b_2^\perp, \ldots, b_n^\perp \vdash b_1 \otimes \bigotimes_{j=2}^{n} b_j^\perp & \quad \ldots \quad \text{id} & \quad b_1^\perp, \ldots, b_{n-1}^\perp, b_n^\perp \vdash \bigotimes_{j=2}^{n} b_j^\perp \otimes b_n
\end{align*}
\]

\[
\begin{align*}
\text{L&} & \quad \vdash b_1 \& b_1^\perp, \ldots, b_n \& b_n^\perp \vdash \bigotimes_{j=2}^{n} b_j \otimes b_n & \quad \text{L&} & \quad \vdash b_1 \& b_1^\perp, \ldots, b_n \& b_n^\perp \vdash \bigotimes_{j=2}^{n} b_j \otimes b_n
\end{align*}
\]

\[
\begin{align*}
\text{R&} & \quad \vdash b_1 \& b_1^\perp, b_2 \& b_2^\perp, \ldots, b_n \& b_n^\perp \vdash \bigotimes_{i=1}^{n} \& \bigotimes_{j \in s} b_j^\perp & \quad \text{R&} & \quad \vdash b_1 \& b_1^\perp, \ldots, b_n \& b_n^\perp \vdash \bigotimes_{i=1}^{n} \& \bigotimes_{j \in s} b_j^\perp
\end{align*}
\]

\[
\begin{align*}
\text{L\otimes} & \quad \vdash \bigotimes_{i=1}^{n} (b_i \& b_i^\perp) \vdash \bigotimes_{i=1}^{n} \& \bigotimes_{j \in s} b_j^\perp & \quad \text{R\otimes} & \quad \vdash \bigotimes_{i=1}^{n} (b_i \& b_i^\perp) \vdash \bigotimes_{i=1}^{n} \& \bigotimes_{j \in s} b_j^\perp
\end{align*}
\]

\[
\begin{align*}
\text{IF2} & \quad \vdash a \otimes (a \otimes (a \rightarrow \bigotimes_{i=1}^{n} (b_i \& b_i^\perp))) \vdash a \otimes (a \rightarrow \bigotimes_{i=1}^{n} \& \bigotimes_{j \in s} b_j^\perp))
\end{align*}
\]

We have provided the verification of matching and merging rules in Appendices B and C, respectively.

### 7.3 Verification of the Matching Process

We have shown the correctness of the matching rules in the previous section, where each rule is checked individually using the inference system of Linear Logic. Here, we want to check, whether the application of rules over \(PFD\) effects a derivation \(LF(PFD) \vdash LF(RFD)\):
We call it as verification of the matching process. So the verification of matching means, if the iterative application of the matching rules over \( PFD \) gives \( RFD \), it leads the derivation of \( LF(RFD) \) from \( LF(PFD) \). In other words, application of a matching rule does not depend on the position of a replacing subtree in \( PFD \).

We show that a matching rule \( m :: S \vdash S' \) that replaces a subtree \( S \) by a subtree \( S' \) provides a proof \( C[S] \vdash C[S'] \), where \( C \) is a context of an arbitrary feature diagram. For example, consider a service feature diagram where the subtree \( S \) being replaced is a member of an And-group. An abstract encoding of this service feature diagram is \( A \otimes (a \rightarrow (S \otimes C)) \), where \( a \) refers to the parent of an And-group and \( A = a \otimes a \), whereas \( C \) shows the encoding of other members of the And-group. The application of a matching rule \( m \) over this feature diagram giving \( A \otimes (a \rightarrow (S' \otimes C)) \) means that rule \( m \) is independent of the context in the service feature diagram.

In the linear encoding of a service feature diagram, the context \( C \) must be connected to a subtree by at least one of the connectives \( \otimes, \rightarrow, \&, \oplus \), etc. The position of a subtree \( S \) in the description tells us which connectives are used to link \( S \) to the context. For example, a subtree being replaced \( S \) is connected with the context by \( \otimes \) and \( \rightarrow \) in the example described in previous paragraph.

On the basis of the position of subtrees in \( PFD \), we consider two categories. If the subtree exists at leaf level, then all the other features, e.g., siblings and parent of the subtree, are considered as a context. In this case, the subtree \( S \) is embedded into a context, we speak of context extension.

On the other hand, if the subtree appears above leaf level, at the root or at the interior node, then the leaf nodes of the subtree are extended to the leaf nodes of the diagram, we speak of leaf extension. We will discuss different cases inside each category. For example, by considering the presence of the subtree in an And-group, or in an Alternative-group.

We use the general rule \( m :: S \vdash S' \) in the first category, whereas for leaf extension we use \( m :: A \otimes (a \rightarrow (B_1 \star B_2 \star \ldots \star B_m)) \vdash A \otimes (a \rightarrow (B'_1 \star B'_2 \star \ldots \star B'_n)) \) as a matching rule, where \( \star \in \{ \otimes, \&, \oplus \} \) and \( m \geq 1 \leq n \). The replacement of \( \star \) with
the connective depends on the group to which the subtree belongs to.

For instance, consider a rule where an Or-group (R) satisfies an Alternative-group (P), as shown in number 7 of Table 6.2 depicted in the right part of Fig. 6.17. If we consider \( b_1 \) and \( b_2 \) as members in the matching rule with \( a \) as parent then the matching rule \( m \) will be written as

\[
A \otimes (a \to ((b_1^+ \otimes b_2) \& (b_1 \otimes b_2^+) \& (b_1 \otimes b_2))) \vdash A \otimes (a \to ((b_1^+ \otimes b_2) \oplus (b_1 \otimes b_2))).
\]

Please note, \( m = 3, n = 2, \star \) is replaced by \& on the left, and by \( \oplus \) on the right of the rule.

### 7.3.1 Context Extension

This category includes the cases where the subtree appears at the level of leafs.

**Case-I:** Let us consider a service feature diagram, where subtree \( S \) exists at the leaf level and has ancestors. If \( C \) shows the encoding of the context in the service feature diagram, then \( C \otimes (p \to S) \) is the linear formula of the overall diagram, where \( p \) refers to the encoding of the parent of \( S \). If the application of a matching rule \( m :: S \vdash S' \) on this feature diagram results in \( C \otimes (p \to S') \), it is independent of the context.

The result of the rule application can be judged by deducing a sequent \( C \otimes (p \to S) \vdash C \otimes (p \to S') \). The deduction of this sequent is shown as:

\[
\begin{align*}
\text{id} & \quad p \vdash p \\
L & \to p, p \to S \vdash S' \\
R & \to p, p \to S \vdash p \to S' \\
R\otimes & \quad C, p \to S \vdash C \otimes (p \to S') \\
L\otimes & \quad C \otimes (p \to S) \vdash C \otimes (p \to S')
\end{align*}
\]

From now on, we will use \( p \) to show the encoding of the parent of a replacing subtree and \( C \) to represent the context appearing on the left of \( p \) in the linear formula of a service feature diagram, as shown in the resulting sequent in the above deduction.
**Case-II:** Let us consider a service feature diagram, where a subtree $S$ exists at leaf level and, while being a member of an And-group, it has ancestors as well, i.e., $S$ has siblings in the service feature diagram. If $C_1$ shows the encoding of the members of the And-group except $S$, then $C \otimes (p \rightarrow (S \otimes C_1))$ is the linear formula of the service feature diagram. Let us check whether the application of a matching rule $m :: S \vdash S'$ on this feature diagram gives us $C \otimes (p \rightarrow (S' \otimes C_1))$:

\[
\begin{array}{c}
\text{Axiom} \quad \frac{S \vdash S'}{C_1 \vdash C_1} \\
\text{id} \quad \frac{C \vdash C}{C \vdash C_1} \\
\text{id} \quad \frac{p \vdash p}{p \vdash p} \\
L \quad \frac{p \vdash p}{S \otimes C_1 \vdash S' \otimes C_1} \\
L \quad \frac{p \vdash p}{S \otimes C_1 \vdash S' \otimes C_1} \\
R \quad \frac{p \vdash p}{(S \otimes C_1) \vdash (S' \otimes C_1)} \\
R \quad \frac{p \vdash p}{(S \otimes C_1) \vdash (S' \otimes C_1)} \\
\end{array}
\]

So, the use of an Axiom $S \vdash S'$ leads the above deduction to $C \otimes (p \rightarrow (S \otimes C_1)) \vdash C \otimes (p \rightarrow (S' \otimes C_1))$. That means, the application of a matching rule $m$ is independent of the context.

**Case-III:** Let us consider a service feature diagram, where a subtree $S$ exists at leaf level and its parent is member of a group (R). Linear formula of this service feature diagram is $C \otimes (p \rightarrow ((C_1 \otimes S) \& C_2))$, where $C_1$ and $C_2$ show the linear encoding of the descendants of $S$ and the rest of the group (R), respectively. If the application of the matching rule $m :: S \vdash S'$ over this service feature diagram gives $C \otimes (p \rightarrow ((C_1 \otimes S') \& C_2))$, we say that it application is independent of the context in this diagram. Again, the proof of the sequent

\[
C \otimes (p \rightarrow ((C_1 \otimes S) \& C_2)) \vdash C \otimes (p \rightarrow ((C_1 \otimes S') \& C_2))
\]

leads us to decide whether the application of rule $m$ depends on the context or not:
The proof above states that the application of a matching rule \( m \) over a service feature diagram is independent of the context.

**Case-IV:** Let us consider a slight variation of the case discussed in Case-III, where a subtree \( S \) is a member of a group \( (P) \) in a service feature diagram. The linear formula of this service feature diagram is 

\[
C \otimes (p \leadsto ((C_1 \otimes S) \oplus C_2))
\]

where \( C_1 \) and \( C_2 \) show the linear encoding of the descendants of \( S \) and the rest of the group \( (P) \). If the application of matching rule \( m :: S \vdash S' \) over this service feature diagram gives 

\[
C \otimes (p \leadsto ((C_1 \otimes S') \oplus C_2))
\]

we say that it is independent of the context. Again, proof of the sequent

\[
C \otimes (p \leadsto ((C_1 \otimes S') \oplus C_2)) \vdash C \otimes (p \leadsto ((C_1 \otimes S') \oplus C_2))
\]

leads us to decide whether the application of rule \( m \) depends on the context or not:

The above proof states that the application of the matching rule \( m \) over a service feature diagram is independent of the context, where the parent of the subtree is a member of a group \( (P) \).
7.3 Verification of the Matching Process

7.3.2 Leaf Extension

Here, we will discuss those cases, where descendants of a replacing subtree are embedded into its leafs.

Case-I: Let us consider a service feature diagram where a subtree has descendants, but no ancestors, i.e., the subtree exists at root level. If $A \otimes (a \rightarrow B)$ and $D$ represent the linear formulas of the subtree and its descendants, the linear formula of a service feature diagram will be $A \otimes (a \rightarrow (B \otimes D))$. Please note that the context becomes part of the linear encoding of the subtree, so we use

$$m :: A \otimes (a \rightarrow B) \vdash A \otimes (a \rightarrow B')$$

as a matching rule, where $a$ and $B$ are linear formulas of parent and the leafs of the subtree.

If the application of a matching rule $m$ over this feature diagram gives $A \otimes (a \rightarrow (B' \otimes D))$, we say that it is independent of the context. The application of the rule over the service feature diagram can be judged by the deduction of $A \otimes (a \rightarrow (B \otimes D)) \vdash A \otimes (a \rightarrow (B' \otimes D))$.

Here, in contrast to the first category descendants are the embedded into the linear encoding of the leafs of the subtree, as shown in above sequent, where $D$ gets multiplied with the leaf of the subtree. In all those cases, where leafs of the subtree embed the descendants, we decompose the matching rule. We decompose a sequent into further sequents in such a way that at least one sequent must consist of an identity sequent. For example, $A \otimes B_1 \vdash A \otimes B_2$ is decomposed into $A \vdash A$ and $B_1 \vdash B_2$.

The descendants $D$ are embedded into the linear encoding of the leafs of the subtree, as shown above. Hence, we decompose the matching $m :: A \otimes (a \rightarrow B) \vdash A \otimes (a \rightarrow B')$ into $A \vdash A$, $a \vdash a$, and $B \vdash B'$. We use non-identity sequent in the proofs because it leads us to the required proof. Let us use $B \vdash B'$ as an axiom in the proof:
It is worth mentioning that we keep decomposing the matching rule until the leaves of the subtree and $\mathcal{D}$ gets separated. For example, in the above proof, $\mathcal{D}$ gets multiplied with $B$, so we decompose the matching rule until we get $B$ on both sides of the rule. That is why we use $B \vdash B'$ as an axiom in the proof shown above. The last deduction step in above proof says that the application of a matching rule $m$ is independent of the context in the feature diagram.

**Case-II:** Let us consider a service feature diagram, where a replacing subtree at root level forms a group $(R)$ and contains descendants as well. The linear formula of such a feature diagram can be written as $A \otimes (a \rightarrow \bigotimes_{i=1}^{m} (B_i \otimes \mathcal{D}_i))$, where $\mathcal{D}_i$ is a linear formula of the descendants appearing with each $B_i$. As we know from the analysis shown in Tables 6.2 and 6.3:

1. An Alternative-group $(R)$ satisfies an Alternative-group $(P)$,
2. An Alternative-group $(R)$ satisfies an Or-group $(P)$,
3. An Or-group $(R)$ satisfies an Alternative-group $(P)$,
4. An Or-group $(R)$ satisfies an Or-group $(P)$,
5. An Or-group $(R)$ satisfies an Alternative-group $(R)$.

That means, a group $(R)$ satisfies five different types including Alternative-groups and Or-groups. We make two categories from this: a) a subtree representing a group $(R)$ is replaced by a group $(P)$; b) a subtree representing an Or-group $(R)$ is replaced by an Alternative-group $(R)$. Let us explain both cases below:
Case-II(a): Let us consider the first category, where a subtree represents a group (R) and is replaced by a group (P). The matching rule \( m \) for this category will become

\[
A \otimes (a \rightarrow \bigwedge_{i=1}^{m} B_i) \vdash A \otimes (a \rightarrow \bigoplus_{j=1}^{k} B'_j)^9 \text{ by replacing } \ast \text{ with } \& \text{ on the left and with } \oplus \text{ on the right of the general matching rule shown earlier. The linear formula of a service feature diagram, where a subtree exists at root level and forms a group (R) is } A \otimes (a \rightarrow \bigwedge_{i=1}^{m} (B_i \otimes D_i)), \text{ where } D_i \text{ represents the descendants of each } B_i.
\]

If the application of a matching rule \( m \) over this feature diagram gives \( A \otimes (a \rightarrow \bigoplus_{j=1}^{k} (B'_j \otimes D_j)) \), we say that it is independent of the context. Again, it can be judged if we can derive

\[
A \otimes (a \rightarrow \bigwedge_{i=1}^{m} (B_i \otimes D_i)) \vdash A \otimes (a \rightarrow \bigoplus_{j=1}^{k} (B'_j \otimes D_j))
\]

As mentioned above, if the parent of the replacing subtree implies \( D \), we need to decompose the matching rule \( m \). We know that \( m :: A \otimes (a \rightarrow \bigwedge_{i=1}^{m} B_i) \vdash A \otimes (a \rightarrow \bigoplus_{j=1}^{k} (B'_j \otimes D_j)) \) can be decomposed into

\[
A \vdash A, a \vdash a, \text{ and } \bigwedge_{i=1}^{m} B_i \vdash \bigoplus_{j=1}^{k} B'_j
\]

By selecting a suitable hypothesis and conclusion from the third sequent, we can say that:

\[ B_s \vdash B'_s \]

where \( m \geq s \leq k \wedge s \geq 1 \), because iterative use of the inference rules \( L\& \) and \( R\oplus \) on the above sequent give us back \( \bigwedge_{i=1}^{m} B_i \vdash \bigoplus_{j=1}^{k} B'_j \). As mentioned earlier, we keep decomposing the matching rule until we get a proposition that can be separated from \( D \) in the linear encoding. That is why we decomposed the rule \( m \) until we get a proposition \( B \) at both ends. Let us use \( B_s \vdash B'_s \) as an axiom while deriving the required proof:

\[ ^9 \text{where } k > 1 < m. \]
The final sequent in the deduction above shows the required proof, so that application of matching rule \( m \) is independent of the context where the subtree makes a group (R) at the top level in a multilevel service feature diagram. Now, let us discuss the second category.

Case-II(b): A replacing subtree represents an Or-group (R) and is replaced by an Alternative-group (R). The matching rule \( m \) for this category will become \( A \otimes (a \rightarrow m \{B_i \otimes D_i\}) \vdash A \otimes (a \rightarrow k \{B'_j \otimes D_j\}) \) by replacing \( \ast \) with \( \& \) and \( \oplus \) on the left and the right side of the matching rule, respectively. Please note that an Or-group (R) consists of more conjuncts connected by \( \& \) then conjuncts for an Alternative-group (R), so \( m > k > 1 \).

The linear formula of a service feature diagram, where a conflicting subtree forms an Or-group (R) with no ancestor, but contains subfeatures, is \( A \otimes (a \rightarrow m \{B_i \otimes D_i\}) \), where \( D_i \) represents the encoding of descendants of \( B_i \). If the application of the matching rule \( m \) over this feature diagram gives \( A \otimes (a \rightarrow k \{B'_j \otimes D_j\}) \), we
say that it is independent of the context. Again, it can be judged if we can derive:

\[ A \otimes (a \rightarrow \bigvee_{i=1}^{m} (B_i \otimes D_i)) \vdash A \otimes (a \rightarrow \bigvee_{j=1}^{k} (B'_j \otimes D_j)) \]

We need to decompose the matching rule \( m \) if the parent of a replacing subtree implies \( D \). The matching rule \( m : A \otimes (a \rightarrow \bigvee_{i=1}^{m} B_i) \vdash A \otimes (a \rightarrow \bigoplus_{j=1}^{k} B'_j) \) can be decomposed into

\[ A \vdash A, a \vdash a, \text{ and } \bigvee_{i=1}^{m} B_i \vdash \bigoplus_{j=1}^{k} B'_j \]

By selecting a suitable hypothesis and conclusion from the third sequent, we can say that:

\[ B_1 \vdash B'_1, B_2 \vdash B'_2, \ldots, B_k \vdash B'_k \]

because iterative use of the inference rule \( Lk \) and connecting all sequents by \( R\& \) gives us back \( \bigvee_{i=1}^{m} B_i \vdash \bigoplus_{j=1}^{k} B'_j \). Let us use these sequents to derive the required proof:

\[
\begin{align*}
A \otimes (a \rightarrow \bigvee_{i=1}^{m} (B_i \otimes D_i)) & \vdash A \otimes (a \rightarrow \bigvee_{j=1}^{k} (B'_j \otimes D_j)) \\
& \vdash A, a \vdash a, \text{ and } \bigvee_{i=1}^{m} B_i \vdash \bigoplus_{j=1}^{k} B'_j \\
& \vdash A, B_1 \vdash B'_1, B_2 \vdash B'_2, \ldots, B_k \vdash B'_k \\
& \vdash A \otimes (a \rightarrow \bigvee_{i=1}^{m} (B_i \otimes D_i)) \vdash A \otimes (a \rightarrow \bigvee_{j=1}^{k} (B'_j \otimes D_j))
\end{align*}
\]

The final sequent in the deduction above shows the required proof, so the application of matching rule \( m \) is independent of the context where the subtree makes an Or-group (R) at the top level in a multilevel service feature diagram.

\(^{10}\text{Ax stands for Axiom.}\)
Case-III: Let us consider a slight variation of the scenario discussed in Case-II, where a replacing subtree at root level forms a group (P) and contains subfeatures as well. The linear formula of such a feature diagram can be written as $A \otimes (a \rightarrow \bigoplus_{i=1}^{m} (B_i \otimes D_i))$, where $D_i$ is a linear formula of the context which appears with each $B_i$. As we know from the analysis shown in Tables 6.2 and 6.3:

1. An Alternative-group (P) satisfies an Or-group (P),

2. An Alternative-group (P) satisfies an And-group of optional (P) features,

3. An Or-group (P) satisfies an And-group of optional (P) features.

That means, group (P) satisfies three different types of groups. We make two categories from this: a) a subtree representing an Alternative-group (P) is replaced by an Or-group (P); b) a subtree representing a group (P) is replaced by an And-group of optional (P) features. Let us explain both cases below:

Case-III(a): A subtree represents an Alternative-group (P) and is replaced by an Or-group (P). The matching rule $m$ for this category becomes $A \otimes (a \rightarrow \bigoplus_{i=1}^{k} B_i) \vdash A \otimes (a \rightarrow \bigoplus_{j=1}^{m} B'_j)$ after replacing $\star$ with $\oplus$ on the left and the right of the general matching rule. Please note that the Or-group (P) consists of more conjuncts connected by $\oplus$ then conjuncts for an Alternative-group (P), so $m > k > 1$.

The linear formula of a service feature diagram, where a conflicting subtree forms an Alternative-group (P) with no ancestor, but subfeatures, is $A \otimes (a \rightarrow \bigoplus_{i=1}^{k} (B_i \otimes D_i))$, where $D_i$ represents the descendants of $B_i$. If the application of the matching rule $m$ over this feature diagram gives $A \otimes (a \rightarrow \bigoplus_{j=1}^{m} (B'_j \otimes D_j))$, we say that it is independent of the context. Again, it can be judged if we can derive:

$$A \otimes (a \rightarrow \bigoplus_{i=1}^{k} (B_i \otimes D_i)) \vdash A \otimes (a \rightarrow \bigoplus_{j=1}^{m} (B'_j \otimes D_j))$$

As mentioned above, the parent of the replacing subtree implies $D$, so we need to decompose the matching rule. The matching rule $m :: A \otimes (a \rightarrow \bigoplus_{i=1}^{k} B_i) \vdash A \otimes (a \rightarrow$
7.3 Verification of the Matching Process

$m \bigoplus_{j=1}^{m} B'_j$ can be decomposed into

$$A \vdash A, a \vdash a, \text{ and } \bigoplus_{i=1}^{k} B_i \vdash \bigoplus_{j=1}^{m} B'_j$$

the third sequent from the above sequents can further be decomposed to:

$$B_1 \vdash \bigoplus_{j=1}^{m} B'_j, B_2 \vdash \bigoplus_{j=1}^{m} B'_j, B_3 \vdash \bigoplus_{j=1}^{m} B'_j, \cdots, B_k \vdash \bigoplus_{j=1}^{m} B'_j$$

because the use of an inference rule $L \oplus$ on these sequents gives us back the original sequent. Selection of a suitable conclusion leads us to the sequents:

$$B_1 \vdash B'_1, B_2 \vdash B'_2, B_3 \vdash B'_3, \cdots, B_k \vdash B'_k$$

Let us use these sequents to generate the required proof:

The final sequent in above deduction shows the required proof, so the application of matching rule $m$ is independent of the context, where conflicting subtree makes
an Alternative-group (P) at the top level in a multilevel service feature diagram.
Now, let us discuss the second category.

**Case-III(b):** A subtree represents a group (P) and is replaced by an And-group of optional (P) features. The matching rule \( m \) for this category will become

\[
m :: A \otimes (a \rightarrow \bigoplus_{i=1}^{m} B_i) \vdash A \otimes (a \rightarrow \bigotimes_{j=1}^{k} (b_j' \oplus b_j'^{\perp}))
\]

by replacing \( \ast \) with \( \oplus \) and \( \otimes \) on the left and the right side of matching rule, respectively, \( m > 1 < k \).

Please note that we move one step further on the right hand side of the matching rule. Informally speaking, the descendants get multiplied to their parents if selected. So we move on to the level of literals in the right. The decomposition of the matching rule will be taken to the level of literals, because the descendants can then be considered separately.

The linear formula of a service feature diagram, where a subtree forms a group (P) with no ancestor, but subfeatures, is \( A \otimes (a \rightarrow \bigoplus_{i=1}^{k} (B_i \otimes D_i)) \), where \( D_i \) represents the descendants of \( B_i \). If the application of the matching rule \( m \) over this feature diagram gives \( A \otimes (a \rightarrow \bigotimes_{j=1}^{k} ((b_j' \otimes D_j) \oplus b_j'^{\perp})) \), where \( D_j \) is the linear encoding of descendants of \( b_j \), we say that the application of \( m \) is independent of the context.

Again, it can be judged if we can derive:

\[
A \otimes (a \rightarrow \bigoplus_{i=1}^{k} (B_i \otimes D_i)) \vdash A \otimes (a \rightarrow \bigotimes_{j=1}^{k} ((b_j' \otimes D_j) \oplus b_j'^{\perp}))
\]

The parent of the replacing subtree implies \( D \), so we need to decompose the rule \( m \). We know that the \( m :: A \otimes (a \rightarrow \bigoplus_{i=1}^{m} B_i) \vdash A \otimes (a \rightarrow \bigotimes_{j=1}^{k} (b_j' \oplus b_j'^{\perp})) \) can be decomposed into

\[
A \vdash A, a \vdash a, \text{ and } \bigoplus_{i=1}^{m} B_i \vdash \bigotimes_{j=1}^{k} (b_j' \oplus b_j'^{\perp})
\]
the third sequent stated above, is further decomposed to

\[ B_1 \vdash \bigotimes_{j=1}^{k} (b'_j \oplus b_j^{\perp}) \]
\[ B_2 \vdash \bigotimes_{j=1}^{k} (b'_j \oplus b_j^{\perp}) \]
\[ \ldots \]
\[ B_m \vdash \bigotimes_{j=1}^{k} (b'_j \oplus b_j^{\perp}) \]

because combining all of these sequents by the inference rule \( L \oplus \) gives us back the original sequent. As each \( B_i = \bigotimes_{j=1}^{k} (b_j \otimes (b_x^v)) \)

above sequents can be written as:

\[ \bigotimes_{j=1}^{k} (b_j \otimes (b_x^v)) \vdash \bigotimes_{j=1}^{k} (b'_j \oplus b_j^{\perp}) \]
\[ \ldots \]
\[ \bigotimes_{j=1}^{k} (b_j \otimes (b_x^v)) \vdash \bigotimes_{j=1}^{k} (b'_j \oplus b_j^{\perp}) \]  \( (7.1) \)

The first sequent from the above list may be further decomposed to the following two levels:

\[ b_1 \otimes \bigotimes_{i=2}^{k} b_i^{\perp} \vdash b'_1 \oplus b_1^{\perp} \]
\[ \ldots \]
\[ (\bigotimes_{i=2}^{k} b_i^{\perp}) \otimes b_k \vdash b'_1 \oplus b_1^{\perp} \]

because the use of inference rule \( R \oplus \) and the iterative use of \( R \otimes \) and then \( L \otimes \) gives us back the previous sequents. Let us use the sequents which we obtain at the last level to generate the required proof:

\[ Ax. \]
\[ \begin{array}{c}
  b_1 \otimes \bigotimes_{i=2}^{k} b_i^{\perp} \vdash b'_1 & id \\\n  D_1 \vdash D_1
\end{array} \]

\[ R \otimes \]
\[ b_1 \otimes \bigotimes_{i=2}^{k} b_i^{\perp}, D_1 \vdash b'_1 \otimes D_1 \]
\[ \ldots \]

\[ L \otimes \]
\[ b_1 \otimes \bigotimes_{i=2}^{k} b_i^{\perp}, D_1 \vdash (b'_1 \otimes D_1) \oplus b_1^{\perp} \]
\[ R \oplus \]
\[ (b_1 \otimes \bigotimes_{i=2}^{k} b_i^{\perp} \otimes D_1), \ldots, (\bigotimes_{i=2}^{k} b_i^{\perp}) \otimes b_k \otimes D_k \vdash ((b'_1 \otimes D_1) \oplus b_1^{\perp}) \otimes \cdots \otimes ((b'_k \otimes D_k) \oplus b_k^{\perp})) \]
\[ \ldots \]
\[ L \otimes \]
\[ (b_1 \otimes \bigotimes_{i=2}^{k} b_i^{\perp} \otimes D_1) \oplus \cdots \oplus ((\bigotimes_{i=2}^{k} b_i^{\perp}) \otimes b_k \otimes D_k) \vdash ((b'_1 \otimes D_1) \oplus b_1^{\perp}) \oplus \cdots \oplus ((b'_k \otimes D_k) \oplus b_k^{\perp})) \]

\[ B_1 \otimes D_1 \vdash \bigotimes_{j=1}^{k} (b'_j \otimes D_j) \oplus b_j^{\perp} \]

\( ^{11} \)where \( s = \{1 \leq x \leq k \land x \neq j\} \). In case of group \( (P) \) representing an Alternative-group \( (P) \)
then \( v = \bot \), otherwise \( v \in \{0,1\}, b^v = b^\perp, b^1 = b. \)
The above deduction shows the result of using the first sequent from the list of sequents shown above in 7.1. In the same way, we get the above conclusion for each of \( B_i \otimes D_i \), where each \( D_i = \bigotimes_{j=1}^{k} D_j \) while considering each \( B_i \) from the list shown in 7.1. We do not show the proof of each sequent to avoid duplication. Using \( L\oplus \) on

\[
B_1 \otimes D_1 \vdash \bigotimes_{j=1}^{k} ((b'_j \otimes D_j) \oplus b_j^{\perp'}) , \ldots , B_m \otimes D_m \vdash \bigotimes_{j=1}^{k} ((b'_j \otimes D_j) \oplus b_j^{\perp'})
\]

triggers the following proof:

\[
\begin{array}{c}
\text{id} & A \vdash A \\
L \otimes & a \vdash a \\
& \bigoplus_{i=1}^{m} (B_i \otimes D_i) \vdash \bigotimes_{j=1}^{k} ((b'_j \otimes D_j) \oplus b_j^{\perp'}) \\
L \oplus & a, a \vdash \bigoplus_{i=1}^{m} (B_i \otimes D_i) \vdash \bigotimes_{j=1}^{k} ((b'_j \otimes D_j) \oplus b_j^{\perp'}) \\
\text{id} & A \vdash A \\
& a \vdash a \\
R \otimes & A, a \vdash \bigoplus_{i=1}^{m} (B_i \otimes D_i) \vdash A \otimes (a \vdash \bigotimes_{j=1}^{k} ((b'_j \otimes D_j) \oplus b_j^{\perp'})) \\
L \otimes & A \otimes (a \vdash \bigoplus_{i=1}^{m} (B_i \otimes D_i)) \vdash A \otimes (a \vdash \bigotimes_{j=1}^{k} ((b'_j \otimes D_j) \oplus b_j^{\perp'}))
\end{array}
\]

As it is clear from the final step in the proof, each \( D_j \) gets multiplied with the \( b'_j \). That is why we have used a different matching rule \( m \) in this proof.

While providing the verification of matching process, we either considered the group features or the And-groups of solitary features. We believe that the And-groups of solitary and group features can also be deduced in the same way by composing more than one cases. For example, a matching rule where Or-group (R) satisfies the And-group of optional (P) features and Or-group (R) can also be considered in the Case-II of the leaf-extension. We have considered these cases individually (please see list shown in the Case-II), so those cases can be combined if required.
7.4 Summary

In this chapter, we have discussed rule-based and instance-based matching, we also discussed that rule-based matching is more demanding than the instance-based matching. Then we move on by showing the verification of graphical matching rules by proving their correctness using the inference system of Linear Logic. The we discusses that the application of graphical rules over a service feature diagram is independent of the context, i.e., application of a rule over a service feature diagram does not depend on the position of the replacing subtree. A replacing subtree is the part of the provider feature diagram which deviates from the corresponding part in the requestor feature diagram. We considered the rule to be applied at top, bottom, or at the middle of the service feature diagram.
Chapter 8

Evaluation

Feature diagrams are used to represent all possible instances of software product lines in a hierarchical form. Kang et al proposed feature diagrams back in 1990 and they are recognized as one of the most important contributions in the field of software product line engineering [KCH+90]. We believe that the use of feature diagrams also provides flexibility in service specification and matching.

As discussed in Chapter 3, matching of classical feature diagrams has not received a lot of attention by the software product line community. In this thesis, we have proposed service feature diagrams which address the limitations encountered when matching classical feature diagrams. The semantics of service feature diagrams are based on Linear Logic.

8.1 Contributions

We categorize the contributions of the thesis into:

1. Extension in Feature Diagrams

   1.1 Service feature diagrams

   1.2 Mapping service feature diagrams into Linear Logic

   1.3 Derivation of instance formulas

2. Matching of Feature Diagrams
2.1 Graphical matching and merging rules

2.2 Verification of graphical rules by linear deduction

Let us discuss the above mentioned contributions.

8.1.1 Extension in Classical Feature Diagrams

To address the challenges raised by matching feature diagrams with propositional logic semantics, we propose service feature diagrams.

**Service Feature Diagrams:** Unlike classical feature diagrams, service feature diagrams respect the hierarchy of features. For example, traditionally a diagram with $a$ as root and $b$ as mandatory subfeature is equivalent to a diagram with $b$ as root and $a$ as mandatory subfeature. This is not the case in service feature diagrams. Apart from the added flexibility, explained further below, service feature diagrams suit scenarios where the hierarchy of features is important. Service feature diagrams add the following concepts to classical feature diagrams:

Types of features: Service feature diagram distinguish resource and shareable features. Each feature in a service feature diagram must reside in one of these categories. If a feature is required to be used only once, it is marked as a resource-feature, whereas a feature that may be selected/used multiple times is marked as a shareable feature. At the level of diagrams, resource and shareable features are represented by a box, and a box with a grey background, respectively.

At the level of logic, a resource feature is encoded by a proposition, whereas propositions encoding shareable features are preceded by $!$-modalities. For example, $A$ and $!A$ are propositions representing a resource and a shareable feature $A$, respectively.

Selection rights for variable features: Often in the case of services, one party (requestor or provider) wants to keep the choice of variable features with them.
Classical feature diagrams do not make any explicit declaration on who will select variable features. Service feature diagrams offer two categories of variable features based on selection rights: by the requestor or the provider. At the level of diagrams, we use solid edges to represent requestor’s choice, whereas dashed edges are used to give selection rights to the provider.

At the level of logic, we use additive conjunction $\&$ and additive disjunction $\oplus$ to encode variable features left for the requestor and the provider to choose from, respectively. So, we are able to model variable features to be chosen by either a requestor or a provider in a service feature diagram.

Notion of instance diagram: Service feature diagrams are customer-friendly in nature, designed keeping in mind the customer of a service. An instance diagram shows a permissible selection of features made by the requestor. Instance diagrams differ from the traditional notion of instance: They, not only differentiate requestor’s and provider’s choices for variable features, but also resource from shareable features. In Table 4.2, we have presented the rules to derive instance diagrams from service feature diagrams.

**Mapping Service Feature Diagrams into Linear Logic:** To formalise the notion of instance diagram, we encode the semantics of service feature diagrams into Linear Logic.

Encoding of service feature diagrams: We have provided the rules to encode the feature types of service feature diagrams in Table 5.1. The recursive application of those rules encodes service feature diagrams.

Encoding of instance diagrams: Table 5.3 states the rules to encode instance diagrams into Linear Logic. The linear encoding of a service feature diagram reflects the semantics captured by instance diagrams. We have also shown that the set of instance diagrams of a service feature diagram is equivalent, via the linear encoding, to the set of instance formulas of a service feature diagram.
Derivation of instance formulas: The linear encoding derives the instance formulas of a service feature diagram. For example, if $LF(fd)$ and $if$ are the linear and instance formulas of a service feature diagram $fd$, $LF(fd) \vdash_if$, where $\vdash_if$ is a restricted deduction using only a subset of inference rules of Linear Logic.

### 8.1.2 Matching Rules of Service Feature Diagrams

To describe the matching process diagrammatically, we propose two categories of graphical rules: Rules for matching and rules for merging. The graphical matching rules are used to check whether a set of requirements are realised by the offer. In case of more than one offer satisfying the requirements, we use merging rules to merge the provider feature diagrams, so that a single offer could be shown to the requestor.

**Graphical Matching Rules:** We have provided 58 graphical matching rules capturing all possible combinations which may occur during the match of service feature diagrams. Each matching rule is of the form $PFD \vdash RFD$ where $PFD$ is the provider and $RFD$ the requestor service feature diagram. For example,

\[
\begin{array}{c}
\frac{x}{x}
\end{array}
\]

means that a provided shareable feature $x$ satisfies a requirement for a resource feature $x$.

To provide the analysis of the matching rules, we categorise them into: Analysis of And-groups of solitary features with Group features and vice versa, and analysis of And-groups of Solitary and Group features with Group features and vice versa. We summarised the analysis in Tables 6.2 and 6.3, respectively.

**Graphical Merging Rules:** We have provided 37 rules, covering all cases of merging in service feature diagrams. We presented one rule that merges $n$ service feature diagrams into one, while the remaining rules are used to normalize the
resulting diagram. A normalized feature diagram is the one which does not violate any of the constraints captured in the metamodel.

Consider a case where a requestor is interested to book a hotel and a flight and wants to pay by credit card. If two offers jointly satisfy this requirement, we merge the offers, so that the requestor can be shown a combined offer. This merged feature diagram has two features with credit card as a payment mechanism. The metamodel of service feature diagrams does not allow more than one feature with the same name. We use rules which merge the multiple credit card features.

We categorise the merging rules into: Merging of solitary features, merging of solitary and group features, and merging of group features. The merging rules cover all possible cases. For example, reconsider the same example, where there is a possibility that both offers mark the credit card at different levels or in different groups. The graphical rules for merging can be used to normalized the resulting service feature diagram.

**Verification:** We have verified the graphical matching rules using the sequent calculus of Linear Logic. That means all are backed up by valid deductions. This shows that they are correct with respect to the inference system of Linear Logic. The deduction always ends with a sequent of the form \( LF(L) \vdash LF(R) \), where \( LF(L) \) and \( LF(R) \) are the linear encodings of the left and right hand sides of a graphical rule.

We have also provided a verification of the matching process of service feature diagrams using sequent calculus of Linear Logic in Chapter 7. We have provided two definitions of matching: One is based on instances, while the other on the rules and deduction. Instance-based matching means, that the set of instance diagrams of \( RFD \) is a subset of the set of instance diagrams of \( PFD \) (all instance formulas of \( RFD \) are derivable from the set of instance formulas of \( PFD \)), rules-based deduction matching means, that we get a sequence of transformations leading \( PFD \) to \( RFD \) (the linear encoding of \( PFD \) derives linear encoding of \( RFD \)).
8.2 Summary

In this chapter, we have covered the contributions presented in the dissertation. We have provided two main contributions: service feature diagrams and their matching. Service feature diagrams provide resource and shareable feature types, and give rights to requestor and provider. The traditional notion of instance is unable to capture the semantics proposed by service feature diagrams, so we introduced instance diagrams. Corresponding instance formulas are derivable from the linear encodings of service feature diagrams. We also discussed the graphical rules for matching and merging of feature diagrams. Then we move to discussing the formal support provided by Linear Logic to the contributions made in the thesis.
Chapter 9

Future Work

In this chapter, we will highlight future work, divided into two categories: aligning service feature diagrams with our work on visual contracts in Section 9.1, and tool support for service feature diagram in Section 9.2. Section 9.3 concludes the chapter.

9.1 Visual Contracts and Service Feature Diagrams

For the first phase of our future work, we are interested to connect service feature diagrams with visual contracts. Visual contracts are used for functional description and matching of services, but are not flexible in nature. On the other hand, feature diagrams provide variability at a high level of abstraction. To combine variability with detailed specification of services, we propose to connect these.

Visual contracts provide a modelling language where graphs model (data) states and rules specify state changing operations. The class of admissible states is specified by a type graph [Roz97]. Fig. 9.1 shows a type graph of a Travel Agent offering Guide, Hotel, Flight, and Transport bookings (please ignore the dashed boxed, we will explain those later). The transformation rules for the requestor and the provider are shown at the top and bottom in Fig. 9.2, respectively. To check if the description can satisfy the requirements, we need to match their visual contracts.

Formally, a (provider) rule satisfies another (requester) rule if the preconditions of the first entail the preconditions of the second and the postconditions / effects of
9.1 Visual Contracts and Service Feature Diagrams

Figure 9.1: Type Graph labelled by features: $BT$ for Bank Transfer, $CC$ for Credit Card, $Gd$ for Guide, $F$ for Flight, $H$ for Hotel, $Tra$ for Transport

the second entail those of the first. Entailment in this case boils down to subgraph matching, allowing for the specialisation of types, i.e., the right-hand side of the provider rule entails the right-hand side of the requester rule because the former is a supergraph of the latter with more general types.

Figure 9.2: Visual contracts specifying booking operations from requester (top) and provider (bottom) point of view

The rules and type graph describe a scenario where requirements are completely satisfied by the existing offer. We have proposed an approach based on feature models, where an offer not providing the complete requirements can also satisfy the requestor’s requirements. To use feature models with visual contracts, we label
9.1 Visual Contracts and Service Feature Diagrams

Figure 9.3: Feature Diagrams of Requestor, Provider, and their Common Features

The elements of the type graph and service models by the features to which they contribute. We have shown a labelled type graph in Fig. 9.1.

We match the feature diagrams by implication, and choose a common instance. For example, if $PFD$ and $RFD$ show the feature diagrams of provider and requestor, respectively, satisfiability of $PFD \Rightarrow RFD$ tells us whether $PFD$ satisfies $RFD$. Based on the common instance we obtain the variant of the underlying models. For example, consider the feature models of requestor and provider in Fig. 9.3, where the provider does not offer the guide, whereas the requestor has marked it as an optional feature. The feature diagram of the common instance is shown in Fig. 9.3(c), it does not have a guide in the provider specifications. For a detailed study of the approach, please see [NH09].

The work of [NH09], needs to be aligned with service feature diagrams to avail the flexibility provided by them. While connecting them, we may have to come up with criteria to link new variable features proposed by the service feature diagrams, at the level of visual contracts.

Our work in [NHO09, NHOH10], provides matching of requestors’ requirements with multiple offers, where visual contract are used to specify requirements and offers. The motivation behind that work was by scenarios, where either one offer
was not enough for the requirements. We plan to use service feature diagrams with [NHOH10] to provide flexibility in the partial matching of visual contracts.

On the other hand, we will also consider the development of tool support for our framework, which we discuss in the following section.

## 9.2 Tool Support

Existing tools only provide facility to classical feature diagrams. By and large, tools supporting classical feature diagrams support an interface for designing/editing feature diagrams, generating instances, and operations such as checking the validity of a feature diagram, detection of dead features etc. Please see [Wik12] for a comprehensive list of tools, supporting classical feature diagrams.

As already stated, there is no effort in the literature to connect feature diagrams with Linear Logic, so no existing tool supports service feature diagrams. We are planning to develop a modelling environment for service feature diagrams, providing the following features:

1. An environment to design service feature diagrams

2. Automatic generation of instance diagrams

3. Matching of service feature diagrams

Using the inference system of Linear Logic, the tool will be able to derive instance formulas, and theorem proving will be used to match multiple service feature diagrams. We do not require our tool to perform basic operations like checking validity of service feature diagrams or detection of dead features, we already handled it at metamodel level of service feature diagrams. For example, a normalized diagram will never consist dead features and always generate at least one instance diagram from a service feature diagram.
9.3 Summary

In this chapter, we have provided an overview of our future work at two levels: integrating service feature diagram with visual contracts and providing tool support for the service feature diagrams.
Appendix: A

A.1 Deduction of First Axiom

\[
\begin{align*}
\text{id} & \vdash \beta \\
\beta \vdash \beta & \uplus \beta & \uplus \beta^\perp \\
L \& & \beta & \& \beta^\perp & \uplus \beta & \uplus \beta^\perp \\
\end{align*}
\]

As we already said, \( \beta \) can either be \( \beta \), or \( \beta \uplus \beta^\perp \), or \( \beta \& \beta^\perp \) and from the above deductions, we say that \( \beta \vdash \beta \uplus \beta^\perp \). We use it as an axiom in the deduction for the merging of solitary features.

A.2 Deduction of Second Axiom

\[
\begin{align*}
\text{id} & \vdash \beta \\
\delta \vdash \delta & \uplus \beta & \uplus \beta^\perp \\
\delta \& \beta & \& \beta^\perp & \uplus \beta & \uplus \beta^\perp \\
\end{align*}
\]
As we already said, $\mathcal{B}$ can either be $\beta$, or $\beta \oplus \beta^\bot$, or $\beta \& \beta^\bot$ and from the above deductions, we say that $\delta \otimes (\delta \otimes (\delta \rightarrow \mathcal{B})) \vdash \delta \otimes (\delta \rightarrow (\beta \rightarrow (\beta \& \beta^\bot)))$. We use it as an axiom in the deduction for the merging of solitary features (where common features exist at different levels in a service feature diagram).
Appendix: B

Verification of Matching Rules

In this Appendix, we provide the validation of the graphical matching rules shown in Chapter 6. As already stated the the result of the graphical rule verification will be a proof of a sequent of the form:

\[ A \otimes (a \rightarrow (B_1 \star B_2 \star \cdots \star B_m)) \vdash A \otimes (a \rightarrow (B_1' \star B_2' \star \cdots \star B_n')) \]

where \( \star \in \{\otimes, \&, \oplus\} \) and \( m \geq 1 \leq n \). In formulas, lower-case letters \( a, b, \ldots \) are literals referring to the features of a feature diagram, capital letters \( A, B, \ldots \) refer to the conjuncts of literals, and the Greek letters \( \alpha, \beta, \ldots \) are meta-propositions referring to the features modelled by diamond symbols in diagrams. We will also use mathematical letters \( A, B, \ldots \) which will refer to arbitrary linear logical formulas.

Again the overall strategy of deduction will remain the same for each derivation, i.e., we start by the identity sequent or an inference figure and move on by applying applicable rules or inference figures. Vertical dots in the deduction mean that the rule mentioned in the deduction step has been applied multiple times (\(< n \) times), whereas horizontal dots show the sequence of sequents.

While providing the verification of graphical rules, we try to derive the multi-rules by using meta-propositions, wherever possible. Although, its not easy to identify the cases, where a multi-rule can be derived by using meta-propositions and where one needs to unfold the multi-rule into its variants and derive those rules independently.
Let us now verify the graphical matching rules provided in Chapter 6.

### B.1 Rules for And-groups of Optional (R) Features

If the parent of an optional (R) feature is selected, the requestor may select or reject an optional subfeature. In the following, we provide the correctness of the rules where the And-group of optional (R) features satisfies the requirements.

**And-group of Optional Features to And-group of Mandatory Features:**

If a provider allows a requestor to choose from an And-group of optional features, this satisfies an And-group of mandatory features, as shown below:

\[
\begin{align*}
&\text{Satisfies} \\
&\begin{array}{c}
a \\
\downarrow \\
\cdots \\
\downarrow \\
b_1 & b_2 & \cdots & b_n
\end{array}
\end{align*}
\]

The verification of above matching rule should end with

\[
a \otimes (a \otimes (a \rightarrow \bigotimes_{i=1}^n (b_i \& b_i^\perp))) \vdash a \otimes (a \rightarrow \bigotimes_{i=1}^n b_i)
\]

which we have obtained by encoding the diagrams shown above by using the rules shown in Table 5.1. Let us now check the derivation of this resulting sequent.

\[
\begin{align*}
\text{id} & \quad b_1, b_2, \ldots, b_n \vdash \bigotimes_{i=1}^n b_i \\
L\& & \vdash \ldots \\
& \quad b_1 \& b_1^\perp, b_2 \& b_2^\perp, \ldots, b_n \& b_n^\perp \vdash \bigotimes_{i=1}^n b_i \\
L\otimes & \vdash \ldots \\
& \quad \bigotimes_{i=1}^n (b_i \& b_i^\perp) \vdash \bigotimes_{i=1}^n b_i \\
IF2 & \quad a \otimes (a \otimes (a \rightarrow \bigotimes_{i=1}^n (b_i \& b_i^\perp))) \vdash a \otimes (a \rightarrow \bigotimes_{i=1}^n b_i)
\end{align*}
\]

**And-group of Optional Features to And-group of Optional (P) Features:**

An And-group of optional (R) features satisfies the requirements of an And-group of optional (P) features, as shown below:
And-group of Optional Features to Or-groups: An And-group of optional (R) features satisfies the Or-groups with requestor’s or provider’s choice, as shown below:

\[
\begin{align*}
&\text{IF1} & b_1, b_2, \ldots, b_n \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp) \\
&L\& & b_1 \& b_1^\perp, b_2 \& b_2^\perp, \ldots, b_n \& b_n^\perp \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp) \\
&L\otimes & \bigotimes_{i=1}^{n} (b_i \& b_i^\perp) \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp) \\
&\text{IF2} & a \otimes (a \otimes (a \to \bigotimes_{i=1}^{n} (b_i \& b_i^\perp))) \vdash a \otimes (a \to (\bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp)))
\end{align*}
\]
Appendix: Verification of Matching Rules

\[
\begin{align*}
L \& & \vdash & b_1 \land b_1^+, b_2 \land b_2^+, \ldots, b_n \land b_n^+ & = \bigotimes_{i=1}^n b_i \\
R \oplus & \vdash & b_1 \land b_1^+, b_2 \land b_2^+, \ldots, b_n \land b_n^+ & \vdash \bigoplus_{i=1}^n (b_i \otimes b_i^+) \\
L \otimes & \vdash & \bigotimes_{i=1}^n (b_i \land b_i^+) & = \bigoplus_{i=1}^n (b_i \otimes b_i^+) \\
IF2 & & a \otimes (a \otimes (\bigoplus_{i=1}^n (b_i \land b_i^+))) & \vdash a \otimes (a \rightarrow (\bigoplus_{i=1}^n (b_i \otimes b_i^+)))
\end{align*}
\]

**And-group of Optional Features to Alternative-groups**: An And-group of optional (R) features satisfies the Alternative-groups with requestor’s or provider’s choice, as shown below:

\[
\begin{align*}
& \text{id} \vdash b_1, b_2^+, \ldots, b_n^+ \vdash b_1 \land \bigotimes_{j=2}^n b_j^+ & \text{id} \vdash b_1^+, \ldots, b_{n-1}^+, b_n \vdash (\bigotimes_{j=2}^n b_j^+) \otimes b_n \\
& L \& \vdash b_1 \land b_1^+, \ldots, b_n \land b_n^+ & L \& \vdash b_1 \land b_1^+, \ldots, b_n \land b_n^+ \vdash (\bigotimes_{i=1}^n b_i \land b_i^+) \\
& R \& \vdash b_1 \land b_1^+, b_2 \land b_2^+, \ldots, b_n \land b_n^+ & R \& \vdash b_1 \land b_1^+, \ldots, b_n \land b_n^+ \vdash \bigotimes_{i=1}^n (b_i \land b_i^+) \\
& L \otimes & \vdash \bigotimes_{i=1}^n (b_i \land b_i^+) & \vdash \bigotimes_{i=1}^n (b_i \land b_i^+) \\
& R \oplus & \vdash \bigotimes_{i=1}^n (b_i \land b_i^+) & \vdash \bigotimes_{i=1}^n (b_i \land b_i^+) \\
& IF2 & a \otimes (a \otimes (\bigotimes_{i=1}^n (b_i \land b_i^+))) & a \otimes (a \rightarrow (\bigotimes_{i=1}^n (b_i \otimes b_i^+))) \\
\end{align*}
\]

\[
\begin{align*}
& \text{id} \vdash b_1, b_2^+, \ldots, b_n^+ \vdash b_1 \land \bigotimes_{i=2}^n b_i^+ \\
& Satisfies \quad & Satisfies
\end{align*}
\]

\[
\begin{align*}
& \text{id} \vdash b_1, b_2^+, \ldots, b_n^+ \vdash b_1 \land \bigotimes_{i=2}^n b_i^+ \\
& Satisfies \quad & Satisfies
\end{align*}
\]
Appendix: Verification of Matching Rules

B.2 Rules for Mixed And-groups (R)

An And-group of solitary features can have a combination of mandatory and optional features, which we call mixed And-group. If the optional features in the mixed And-group are selected by the requestor then we call it a Mixed And-group (R). Let us check which requirements can be satisfied by the mixed And-group (R) in descriptions.

Mixed And-group to Alternative-group (P): A mixed And-group (R) satisfies the requirements of an Alternative-group (P), as shown below:

\[
\begin{align*}
L \& 
\vdash b_1 &\& b_1^+, \ldots, b_n &\& b_n^+ \vdash b_1 \otimes \bigotimes_{i=2}^{n} b_i^+ \\
R \oplus 
\vdash b_1 &\& b_1^+, \ldots, b_n &\& b_n^+ \vdash \bigoplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in \mathcal{S}} b_j^+) \\
L \otimes 
\vdash \bigotimes_{i=1}^{n} (b_i &\& b_i^+) \vdash \bigoplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in \mathcal{S}} b_j^+) \\
IF2 
\vdash a \otimes (a \& (a \rightarrow (\bigotimes_{i=2}^{n} (b_i &\& b_i^+)))) \vdash a \otimes (a \& (a \rightarrow (\bigoplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in \mathcal{S}} b_j^+))))
\end{align*}
\]
Appendix: Verification of Matching Rules

Mixed And-group to Or-group (P): A mixed And-group (R) satisfies the requirements of an Or-group (P), as shown below. From here onwards \( i < n \), such that \( n \) does not violate the condition described earlier, i.e., \( n \geq 1 \).

\[
\begin{align*}
L & \& b_1, \ldots, b_i, b_{i+1}, \ldots, b_n \vdash \bigotimes_{i=1}^n b_i \\
R & \& (\bigotimes_{i=1}^l b_i) \otimes \bigotimes_{m=i+1}^n (b_m \& b_m^\perp) \vdash \bigotimes_{i=1}^n b_i \\
\end{align*}
\]

B.3 Rule for Mixed And-group (P)

A mixed And-group (P) is an And-group having a combination of mandatory and optional (P) features. Let us check what types of requirements can be satisfied by a mixed And-group (P).

Mixed And-group to Or-group (P): A mixed And-group (P) satisfies the requirements of an Or-group (P), as shown below:

\[
\begin{align*}
L & \oplus b_1, \ldots, b_i, b_{i+1}, \ldots, b_n \oplus b_{i}^\perp, \ldots, b_n^\perp \vdash \bigoplus_{i=1}^n b_i \\
R & \oplus (\bigotimes_{i=1}^l b_i) \otimes (\bigotimes_{m=i+1}^n (b_m \& b_m^\perp)) \vdash \bigoplus_{i=1}^n (\bigotimes_{j \in s} (b_i \otimes b_j^\perp))) \\
\end{align*}
\]
### Rules for Mixed And-group with Mixed Choices

A mixed And-group with mixed choices means that some of the optional features are selected by the requestor and some by the provider. Let us check which requirements are satisfied by a mixed And-group with mixed choices.

**Mixed And-group to Or-group (P):** A mixed And-group with mixed choices satisfies the requirements of an Or-group (P), as shown below:

\[
L \otimes \bigotimes_{l=1}^{i} b_l \otimes \bigotimes_{m=i+1}^{n} (b_m \oplus b_{m}^\perp) \vdash (\bigotimes_{i=1}^{n} b_i) \oplus (\bigotimes_{l=1}^{i} b_l \otimes \bigotimes_{m=i+1}^{n} b_{m}^\perp)
\]

\[
R \oplus \bigoplus_{l=1}^{i} b_l \otimes \bigoplus_{m=i+1}^{n} (b_m \oplus b_{m}^\perp) \vdash \bigoplus_{i=1}^{n} (b_i \otimes b_{i}^\perp)
\]

**IF2**

\[
a \otimes (a \rightarrow (\bigotimes_{l=1}^{i} b_l \otimes \bigotimes_{m=i+1}^{n} (b_m \oplus b_{m}^\perp)))) \vdash a \otimes (a \rightarrow (\bigoplus_{i=1}^{n} (b_i \otimes b_{i}^\perp)))
\]
B.5 Rules for Alternative-groups (R)

If a feature is chosen, exactly one subfeature from its Alternative-group (R) must be selected by the requestor. Let us discuss the matching rules for an Alternative-group (R).

Alternative-group to And-group of Optional (P) Features: The Alternative-group (R) satisfies an And-group of optional (P) features, as shown below:

\[
\begin{align*}
&IF_1 \quad b_1, b_2^+, \ldots, b_n^+ \vdash \bigotimes_{i=1}^n (b_i \oplus b_i^-) \\
&L\& \quad \vdots \\
&L\otimes \quad \vdots \\
&L\& \quad \vdots \\
&IF_2 \quad a \otimes (a \otimes (a \rightarrow (\bigotimes_{i=1}^n (b_i \otimes b_j^-)))) \vdash a \otimes (a \otimes (a \rightarrow (\bigotimes_{i=1}^n (b_i \otimes b_j^-))))
\end{align*}
\]

Alternative-group to Or-group (P): An Alternative-group (R) satisfies an Or-group (P), as shown below:

\[
\begin{align*}
&id \quad b_1 \otimes \bigotimes_{i=2}^n b_i^+ \vdash b_1 \otimes \bigotimes_{i=2}^n b_i^+ \\
&L\& \quad \vdots \\
&R\oplus \quad \vdots \\
&R\oplus \quad b_1 \otimes \bigotimes_{j \in S} b_j^+ \vdash \bigoplus_{i=1}^n (\bigotimes_{j \in S} (b_i \otimes b_j^-))
\end{align*}
\]
Appendix: Verification of Matching Rules

IF2
\[ a \otimes (a \otimes (a \rightarrow (\bigwedge_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j)))) \vdash a \otimes (a \rightarrow (\bigoplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j)))) \]

Alternative-group to Alternative-group (P): An Alternative-group (R) satisfies an Alternative-group (P), as shown below:

\[
\begin{array}{c}
\text{id} \\
\vdash b_1 \otimes \bigotimes_{i=2}^{n} b_i \vdash b_1 \otimes \bigotimes_{i=2}^{n} b_i \\
L & \vdash \bigwedge_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j) \vdash b_1 \otimes \bigotimes_{i=2}^{n} b_i \\
R & \vdash \bigwedge_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j) \vdash \bigoplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j) \\
L & \vdash a \otimes (a \otimes (a \rightarrow (\bigwedge_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j)))) \vdash a \otimes (a \rightarrow (\bigoplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j))))
\end{array}
\]

Alternative-group to And-group of Optional (P) Features and an Alternative-group (P): An Alternative-group (R) satisfies the requirements of an And-group of optional (P) features and an Alternative-group (P), as shown below:

\[
\begin{array}{c}
\text{id} \\
\vdash b_{i+1}^+, \ldots, b_{n-1}^+, b_n \vdash \bigoplus_{l=1}^{i} (b_l \oplus b_l^+) \\
R & \vdash b_{i+1}^+, \ldots, b_{n-1}^+, b_n \vdash \bigoplus_{m=i+1}^{n} (b_m \otimes \bigotimes_{j \in s} b_j^+) \\
L & \vdash \bigwedge_{i=1}^{n} (b_i) \otimes b_n \vdash \bigoplus_{l=1}^{i} (b_l \oplus b_l^+) \otimes \bigoplus_{m=i+1}^{n} (b_m \otimes \bigotimes_{j \in s} b_j^+) \\
L & \vdash \bigwedge_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j^+) \vdash \bigoplus_{l=1}^{i} (b_l \oplus b_l^+) \otimes \bigoplus_{m=i+1}^{n} (b_m \otimes \bigotimes_{j \in s} b_j^+)
\end{array}
\]
Alternative-group to And-group of Optional (P) Features and an Or-group (P): An Alternative-group (R) satisfies the requirements of an And-group of optional (P) features and an Or-group (P), as shown below:

\[
R \oplus \left( \bigwedge_{i=1}^{n} (b_i \otimes b^+_j) \right) \vdash \bigotimes_{i=1}^{n} (b_i \oplus b^+_l) \otimes \bigoplus_{m=i+1}^{n} (b_m \otimes b^+_j) \]

\[
a \otimes (a \otimes (a \rightarrow \left( \bigwedge_{i=1}^{n} (b_i \otimes b^+_j) \right))) \vdash a \otimes (a \rightarrow \left( \bigotimes_{i=1}^{n} (b_i \oplus b^+_l) \otimes \bigoplus_{m=i+1}^{n} (b_m \otimes b^+_j) \right)))
\]

B.6 Rules for Alternative-groups (P)

When the provider wants to keep the choice of an Alternative-group, it satisfies the following cases.

Alternative-group to And-group of Optional (P) Features: An Alternative-group (P) satisfies an And-group of optional (P) features, as shown below:
Appendix: Verification of Matching Rules

IF1:  
\[ b_1, b_2, \ldots, b_n \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^+) \]

\[ L \otimes \vdash b_1 \otimes \bigotimes_{i=2}^{n} (b_i \oplus b_i^+) \]

\[ L \oplus \vdash (b_1 \otimes \bigotimes_{i=2}^{n} b_i^+) \oplus (b_n \otimes \bigotimes_{i=1}^{n-1} b_i^+) \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^+) \]

IF1:  
\[ b_1^+, \ldots, b_{n-1}^+, b_n \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^+) \]

\[ L \otimes \vdash b_n \otimes \bigotimes_{i=1}^{n-1} (b_i \oplus b_i^+) \]

\[ \bigoplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j^+) \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^+) \]

IF2:  
\[ a \otimes (a \rightarrow \bigoplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j^+)) \vdash a \otimes (a \rightarrow (a \rightarrow \bigoplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j^+))) \]

Alternative-group to Or-group (P):  
An Alternative-group (P) satisfies an Or-group (P), as shown below:

\[ \begin{array}{c}
\text{Satisfies} \\
\begin{array}{c}
\text{id} \\
\oplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j^+) \vdash \bigoplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j^+)
\end{array}
\end{array} \]

B.7 Rules for Or-groups (R)

At least one subfeature of an Or-group (R) must be selected by the requestor, if the parent of this group is chosen. Let us discuss the requirements that can be satisfied by an Or-group (R).

Or-group to Or-group (P):  
An Or-group (P) of the requirements can be satisfied by an Or-group (R) of the description, as shown below:

\[ \begin{array}{c}
\text{Satisfies} \\
\begin{array}{c}
\text{id} \\
\oplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j^+) \vdash \bigoplus_{i=1}^{n} (b_i \otimes \bigotimes_{j \in s} b_j^+)
\end{array}
\end{array} \]
Appendix: Verification of Matching Rules

\[
\begin{align*}
\text{id: } & \quad b_i \vdash b_i \\
\text{L\&: } & \quad \bigwedge_{i=1}^n (b_i \otimes \bigotimes_{j \in s} b_j^i) \vdash \bigotimes_{i=1}^n b_i \\
\text{R\oplus: } & \quad \bigwedge_{i=1}^n (b_i \otimes \bigotimes_{j \in s} b_j^i) \vdash \bigoplus_{i=1}^n (b_i \otimes \bigotimes_{j \in s} b_j^i) \\
\text{IF2: } & \quad a \otimes (a \otimes (a \rightarrow (\bigwedge_{i=1}^n (\bigwedge_{j \in s} b_j^i)))) \vdash a \otimes (a \rightarrow (\bigoplus_{i=1}^n (\bigoplus_{j \in s} b_j^i))))
\end{align*}
\]

**Or-group to Alternative-group (R):** An Or-group (R) satisfies an Alternative-group with requestor’s choice, as shown below:

\[
\begin{align*}
\text{id: } & \quad b_i \vdash b_i \\
\text{L\&: } & \quad \bigwedge_{i=1}^n (b_i \otimes \bigotimes_{j \in s} b_j^i) \vdash \bigotimes_{i=1}^n b_i \\
\text{R\oplus: } & \quad \bigwedge_{i=1}^n (b_i \otimes \bigotimes_{j \in s} b_j^i) \vdash \bigoplus_{i=1}^n (b_i \otimes \bigotimes_{j \in s} b_j^i) \\
\text{IF2: } & \quad a \otimes (a \otimes (a \rightarrow (\bigwedge_{i=1}^n (\bigwedge_{j \in s} b_j^i)))) \vdash a \otimes (a \rightarrow (\bigoplus_{i=1}^n (\bigoplus_{j \in s} b_j^i))))
\end{align*}
\]

**Or-group to Alternative-group (P):** An Or-group (R) satisfies an Alternative-group with provider’s choice, as shown below:

\[
\begin{align*}
\text{id: } & \quad b_i \vdash b_i \\
\text{L\&: } & \quad \bigwedge_{i=1}^n (b_i \otimes \bigotimes_{j \in s} b_j^i) \vdash \bigotimes_{i=1}^n b_i \\
\text{R\oplus: } & \quad \bigwedge_{i=1}^n (b_i \otimes \bigotimes_{j \in s} b_j^i) \vdash \bigoplus_{i=1}^n (b_i \otimes \bigotimes_{j \in s} b_j^i) \\
\text{IF2: } & \quad a \otimes (a \otimes (a \rightarrow (\bigwedge_{i=1}^n (\bigwedge_{j \in s} b_j^i)))) \vdash a \otimes (a \rightarrow (\bigoplus_{i=1}^n (\bigoplus_{j \in s} b_j^i))))
\end{align*}
\]
Or-group to And-group of Mandatory Features: An Or-group (R) satisfies an And-group of mandatory features, as shown below:

\[
\begin{align*}
\text{Satisfies} & \quad \text{Satisfies} \\
\begin{array}{c}
\ast \\
b_1, \ldots, b_n
\end{array} & \quad \begin{array}{c}
\ast \\
b_1, \ldots, b_n
\end{array} \\
\begin{array}{c}
\otimes b_i \vdash \otimes b_i \\
\vdots \\
\otimes b_i \vdash \otimes (b_i \oplus b_i^+)
\end{array} & \quad \begin{array}{c}
\otimes b_i \vdash \otimes (b_i \oplus b_i^+) \\
\vdots \\
\otimes (b_i \oplus b_i^+) \vdash \otimes (b_i \oplus b_i^+)
\end{array} \\
L\& & \quad L\& \\
\begin{array}{c}
\& \land (\& (b_i \otimes \otimes b_i^+)) \vdash \otimes b_i \\
\vdots \\
\& \land (\& (b_i \otimes \otimes b_i^+)) \vdash \otimes (b_i \oplus b_i^+)
\end{array} & \quad \begin{array}{c}
\& \land (\& (b_i \otimes \otimes b_i^+)) \vdash \otimes (b_i \oplus b_i^+) \\
\vdots \\
\& \land (\& (b_i \otimes \otimes b_i^+)) \vdash \otimes (b_i \oplus b_i^+)
\end{array} \\
IF1 & \quad IF2 \\
\begin{array}{c}
b_1, \ldots, b_n \vdash \otimes (b_i \oplus b_i^+) \\
\vdots \\
b_n \vdash \otimes (b_i \oplus b_i^+)
\end{array} & \quad \begin{array}{c}
a \otimes (a \otimes (a \rightarrow (\& (b_i \otimes \otimes b_i^+))))) \vdash a \otimes (a \rightarrow (\& (b_i \oplus b_i^+)))
\end{array}
\end{align*}
\]

Or-group to And-group of Optional (P) Features: An Or-group (R) satisfies an And-group of optional (P) features, as shown below:

\[
\begin{align*}
\text{Satisfies} & \quad \text{Satisfies} \\
\begin{array}{c}
\ast \\
b_1, \ldots, b_n
\end{array} & \quad \begin{array}{c}
\ast \\
b_1, \ldots, b_n
\end{array} \\
\begin{array}{c}
\otimes b_i \vdash \otimes (b_i \oplus b_i^+) \\
\vdots \\
\otimes (b_i \oplus b_i^+) \vdash \otimes (b_i \oplus b_i^+)
\end{array} & \quad \begin{array}{c}
\otimes (b_i \oplus b_i^+) \vdash \otimes (b_i \oplus b_i^+) \\
\vdots \\
\otimes (b_i \oplus b_i^+) \vdash \otimes (b_i \oplus b_i^+)
\end{array} \\
L\& & \quad L\& \\
\begin{array}{c}
\& \land (\& (b_i \otimes \otimes b_i^+)) \vdash \otimes b_i \\
\vdots \\
\& \land (\& (b_i \otimes \otimes b_i^+)) \vdash \otimes (b_i \oplus b_i^+)
\end{array} & \quad \begin{array}{c}
\& \land (\& (b_i \otimes \otimes b_i^+)) \vdash \otimes (b_i \oplus b_i^+) \\
\vdots \\
\& \land (\& (b_i \otimes \otimes b_i^+)) \vdash \otimes (b_i \oplus b_i^+)
\end{array} \\
IF1 & \quad IF2 \\
\begin{array}{c}
b_1, \ldots, b_n \vdash \otimes (b_i \oplus b_i^+) \\
\vdots \\
b_n \vdash \otimes (b_i \oplus b_i^+)
\end{array} & \quad \begin{array}{c}
a \otimes (a \otimes (a \rightarrow (\& (b_i \otimes \otimes b_i^+))))) \vdash a \otimes (a \rightarrow (\& (b_i \oplus b_i^+)))
\end{array}
\end{align*}
\]

Or-group to Mixed And-group (P): An Or-group (R) satisfies a mixed And-group (P), as shown below:

\[
\begin{align*}
\text{Satisfies} & \quad \text{Satisfies} \\
\begin{array}{c}
\ast \\
b_1, \ldots, b_n
\end{array} & \quad \begin{array}{c}
\ast \\
b_1, \ldots, b_n
\end{array} \\
\begin{array}{c}
\otimes b_i \vdash \otimes (b_i \oplus b_i^+) \\
\vdots \\
\otimes (b_i \oplus b_i^+) \vdash \otimes (b_i \oplus b_i^+)
\end{array} & \quad \begin{array}{c}
\otimes (b_i \oplus b_i^+) \vdash \otimes (b_i \oplus b_i^+) \\
\vdots \\
\otimes (b_i \oplus b_i^+) \vdash \otimes (b_i \oplus b_i^+)
\end{array} \\
L\& & \quad L\& \\
\begin{array}{c}
\& \land (\& (b_i \otimes \otimes b_i^+)) \vdash \otimes b_i \\
\vdots \\
\& \land (\& (b_i \otimes \otimes b_i^+)) \vdash \otimes (b_i \oplus b_i^+)
\end{array} & \quad \begin{array}{c}
\& \land (\& (b_i \otimes \otimes b_i^+)) \vdash \otimes (b_i \oplus b_i^+) \\
\vdots \\
\& \land (\& (b_i \otimes \otimes b_i^+)) \vdash \otimes (b_i \oplus b_i^+)
\end{array} \\
IF1 & \quad IF2 \\
\begin{array}{c}
b_1, \ldots, b_i \vdash \otimes (b_i \oplus b_i^+) \\
\vdots \\
b_i \vdash \otimes (b_i \oplus b_i^+)
\end{array} & \quad \begin{array}{c}
b_{i+1}, \ldots, b_n \vdash \otimes (b_i \oplus b_i^+) \\
\vdots \\
b_m \vdash \otimes (b_i \oplus b_i^+)
\end{array}
\end{align*}
\]
Appendix: Verification of Matching Rules

\[
R \otimes \vdash b_1, \ldots, b_n \vdash \bigotimes_{l=1}^{i} (b_l \oplus b_l^\perp) \otimes \bigotimes_{m=i+1}^{n} b_m
\]

\[
L \otimes \vdash \bigotimes_{i=1}^{n} b_i \vdash \bigotimes_{l=1}^{i} (b_l \oplus b_l^\perp) \otimes \bigotimes_{m=i+1}^{n} b_m
\]

\[
L \& \vdash \bigotimes_{i=1}^{n} (\& (b_i \otimes b_j^\uparrow)) \vdash \bigotimes_{l=1}^{i} (b_l \oplus b_l^\perp) \otimes \bigotimes_{m=i+1}^{n} b_m
\]

\[
IF2 \quad a \otimes (a \otimes (a \rightarrow \bigotimes_{i=1}^{n} (\& (b_i \otimes b_j^\uparrow)))) \vdash a \otimes (a \rightarrow \bigotimes_{l=1}^{i} (b_l \oplus b_l^\perp) \otimes \bigotimes_{m=i+1}^{n} b_m)
\]

**Or-group to And-group of Solitary and Alternative-group (P):** An Or-group (R) satisfies the And-groups of solitary features and an Alternative-group (P), as shown below:

\[
IF1 \quad b_1^\perp, \ldots, b_i^\perp \vdash \bigotimes_{l=1}^{i} (b_l \oplus b_l^\perp)
\]

\[
IF2 \quad \bigotimes_{i=1}^{n-1} b_i^\perp, \ldots, b_n^\perp \vdash \bigotimes_{l=1}^{i} (b_l \oplus b_l^\perp) \otimes \bigotimes_{m=i+1}^{n} b_m \otimes \bigotimes_{j=s}^{r} b_j^\perp
\]

\[
R \oplus \vdash b_1^\perp, \ldots, b_i^\perp \vdash \bigotimes_{l=1}^{i} (b_l \oplus b_l^\perp)
\]

\[
L \otimes \vdash \bigotimes_{i=1}^{n} b_i^\perp \otimes b_n \vdash \bigotimes_{l=1}^{i} (b_l \oplus b_l^\perp) \otimes \bigotimes_{m=i+1}^{n} (b_m \otimes \bigotimes_{j=s}^{r} b_j^\perp)
\]

\[
L \& \vdash \bigotimes_{i=1}^{n} (\& (b_i \otimes b_j^\uparrow)) \vdash \bigotimes_{l=1}^{i} (b_l \oplus b_l^\perp) \otimes \bigotimes_{m=i+1}^{n} (b_m \otimes \bigotimes_{j=s}^{r} b_j^\perp)
\]

\[
IF2 \quad a \otimes (a \otimes (a \rightarrow \bigotimes_{i=1}^{n} (\& (b_i \otimes b_j^\uparrow)))) \vdash a \otimes (a \rightarrow \bigotimes_{l=1}^{i} (b_l \oplus b_l^\perp) \otimes \bigotimes_{m=i+1}^{n} (b_m \otimes \bigotimes_{j=s}^{r} b_j^\perp))
\]
Satisfies the And-groups of solitary features and an Or-group (P), as shown below:

\[
\begin{align*}
\text{id} & \quad b_{i+1}^{\perp}, \ldots, b_{n-1}^{\perp}, b_n \vdash (\bigotimes_{m=i+1}^{n-1} b_m^{\perp}) \otimes b_n \\
\text{id} & \quad b_1, \ldots, b_i \vdash \bigotimes_{l=1}^i b_l \\
R \oplus & \quad b_{i+1}^{\perp}, \ldots, b_{n-1}^{\perp}, b_n \vdash \bigoplus_{m=i+1}^n (b_m \otimes b_j^{\perp}) \\
R \otimes & \quad b_1, \ldots, b_i, b_{i+1}^{\perp}, \ldots, b_{n-1}^{\perp}, b_n \vdash (\bigotimes b_l) \otimes \bigoplus_{m=i+1}^n (b_m \otimes b_j^{\perp}) \\
L \otimes & \quad \bigotimes_{l=1}^i (\bigotimes b_l) \otimes (\bigotimes_{m=i+1}^{n-1} b_m^{\perp}) \otimes b_n \vdash (\bigotimes b_l) \otimes \bigoplus_{m=i+1}^n (b_m \otimes b_j^{\perp}) \\
L \& & \quad \bigotimes_{l=1}^i (\bigotimes b_l) \otimes (\bigotimes_{m=i+1}^{n-1} b_m^{\perp}) \otimes b_n \vdash (\bigotimes b_l) \otimes \bigoplus_{m=i+1}^n (b_m \otimes b_j^{\perp}) \\
\text{IF2} & \quad a \otimes (a \otimes (a \rightarrow ((\bigotimes_{l=1}^i b_l) \otimes \bigoplus_{m=i+1}^n (b_m \otimes b_j^{\perp})))) \\
\end{align*}
\]

**Or-group to And-group of Solitary and Or-group (P):** An Or-group (R) satisfies the And-groups of solitary features and an Or-group (P), as shown below:

\[
\begin{align*}
\text{id} & \quad b_{i+1}^{\perp}, \ldots, b_{n-1}^{\perp}, b_n \vdash (\bigotimes_{m=i+1}^{n-1} b_m^{\perp}) \otimes b_n \\
\text{id} & \quad b_1, \ldots, b_i \vdash \bigotimes_{l=1}^i b_l \\
R \oplus & \quad b_{i+1}^{\perp}, \ldots, b_{n-1}^{\perp}, b_n \vdash \bigoplus_{m=i+1}^n (b_m \otimes b_j^{\perp}) \\
R \otimes & \quad b_1, \ldots, b_i, b_{i+1}^{\perp}, \ldots, b_{n-1}^{\perp}, b_n \vdash (\bigotimes b_l) \otimes \bigoplus_{m=i+1}^n (b_m \otimes b_j^{\perp}) \\
L \otimes & \quad (\bigotimes b_l) \otimes b_n \vdash (\bigotimes b_l) \otimes \bigoplus_{m=i+1}^n (b_m \otimes b_j^{\perp}) \\
L \& & \quad (\bigotimes b_l) \otimes b_n \vdash (\bigotimes b_l) \otimes \bigoplus_{m=i+1}^n (b_m \otimes b_j^{\perp}) \\
\text{IF2} & \quad a \otimes (a \otimes (a \rightarrow ((\bigotimes_{l=1}^i b_l) \otimes \bigoplus_{m=i+1}^n (b_m \otimes b_j^{\perp})))) \\
\end{align*}
\]
An And-group of optional (P) features, as shown below:

If a provider wants to keep the choice of the Or-group, it satisfies the following set of requirements.

**Or-group to And-group of Optional (P) Features:** An Or-group (P) satisfies an And-group of optional (P) features, as shown below:

\[
\begin{align*}
\text{id} & \quad b_{1}^{\perp}, \ldots, b_{n}^{\perp}, b_{n} \vdash ( \bigotimes_{m=1}^{n-1} b_{m}^{\perp}) \otimes b_{n} \\
\text{id} & \quad b_{1}, \ldots, b_{i} \vdash \bigotimes_{l=1}^{i} b_{l} \\
R_{\bigotimes} & \quad b_{1}^{\perp}, \ldots, b_{n}^{\perp}, b_{n} \vdash \bigoplus_{m=1}^{n} \bigotimes_{j \in s} (b_{m} \otimes b_{j}^{\perp}) \\
R_{\bigotimes} & \quad b_{1}, \ldots, b_{i}^{\perp}, \ldots, b_{n-1}^{\perp}, b_{n} \vdash \bigotimes_{l=1}^{i} (b_{l} \otimes \bigoplus_{m=1}^{n} \bigotimes_{j \in s} (b_{m} \otimes b_{j}^{\perp})) \\
L_{\bigotimes} & \quad \bigotimes_{l=1}^{i} (b_{l} \otimes \bigoplus_{m=1}^{n} \bigotimes_{j \in s} (b_{m} \otimes b_{j}^{\perp})) \vdash \bigotimes_{l=1}^{i} (b_{l} \otimes \bigoplus_{m=1}^{n} \bigotimes_{j \in s} (b_{m} \otimes b_{j}^{\perp})) \\
L_{\&} & \quad \bigotimes_{l=1}^{i} (b_{l} \otimes \bigoplus_{m=1}^{n} \bigotimes_{j \in s} (b_{m} \otimes b_{j}^{\perp})) \vdash \bigotimes_{l=1}^{i} (b_{l} \otimes \bigoplus_{m=1}^{n} \bigotimes_{j \in s} (b_{m} \otimes b_{j}^{\perp})) \\
IF_{2} & \quad a \otimes (a \otimes (a \rightarrow \bigoplus_{i=1}^{n} \bigotimes_{j \in s} (b_{i} \otimes b_{j}^{\perp}))) \vdash a \otimes (a \otimes (a \rightarrow \bigoplus_{i=1}^{n} \bigotimes_{j \in s} (b_{i} \otimes b_{j}^{\perp})))) \\
\end{align*}
\]

**B.8 Rule for Or-group (P)**

If a provider wants to keep the choice of the Or-group, it satisfies the following set of requirements.
B.9 Rules for And-groups of an Alternative-group (R) and Solitary Features

A requestor-based choice of an Alternative-group along with solitary features satisfies the following set of requirements.

And-group of Optional (R) Features and an Alternative-group to Alternative-group (P): An And-group of optional (R) features and an Alternative-group (R) satisfies an Alternative-group of provider-based choice, as shown below:

\[
\begin{align*}
&\vdash (\bigotimes b_{i_1}^1) \otimes b_n \\
&\vdash (\bigotimes b_{i_1}^1) \otimes (\bigotimes b_{j_1}^1) \\
&\vdash (\bigotimes (b_{i_1} & b_{j_1}^1)) \otimes (\bigotimes b_{j_1}^1) \\
&\vdash (\bigotimes (b_{i_1} & (b_{i_1} \otimes b_{j_1}^1))) \otimes (\bigotimes b_{j_1}^1) \\
&\vdash (\bigotimes (b_{i_1} & b_{j_1}^1) \otimes (b_{i_1} \otimes (b_{i_1} \otimes b_{j_1}^1))) \otimes (\bigotimes b_{j_1}^1) \\
&\vdash (\bigotimes (b_{i_1} & b_{j_1}^1) \otimes (b_{i_1} \otimes (b_{i_1} \otimes b_{j_1}^1))) \\
&\vdash (\bigotimes (b_{i_1} & (b_{i_1} \otimes b_{j_1}^1))) \otimes (\bigotimes b_{j_1}^1) \\
&\vdash (\bigotimes (b_{i_1} & b_{j_1}^1) \otimes (\bigotimes (b_{i_1} \otimes b_{j_1}^1))) \otimes (\bigotimes b_{j_1}^1) \\
\end{align*}
\]

And-group of Mandatory Features and an Alternative-group to Or-group (P): An And-group of mandatory features and an Alternative-group (R) satisfies an Or-group (P), as shown below:
\[
\begin{align*}
\text{id} & \quad b_1^\perp, \ldots, b_{n-1}^\perp, b_n \vdash (\bigotimes_{i=1}^{n-1} b_i^\perp) \otimes b_n \\
L \otimes & \quad \bigotimes_{i=1}^{n} b_i, (\bigotimes_{m=i+1}^{n-1} b_m^\perp) \otimes b_n \vdash (\bigotimes_{i=1}^{n-1} b_i^\perp) \otimes b_n \\
L \& & \quad \bigotimes_{i=1}^{n} b_i, \bigotimes_{m=i+1}^{n} (b_m \otimes b_j^\perp) \vdash (\bigotimes_{i=1}^{n-1} b_i^\perp) \otimes b_n \\
R \oplus & \quad \bigotimes_{i=1}^{n} b_i, \bigotimes_{m=i+1}^{n} (b_m \otimes b_j^\perp) \vdash \bigoplus_{i=1}^{n} (b_i \otimes b_j) \\
L \otimes & \quad \bigotimes_{i=1}^{n} (b_i \otimes b_j), (\bigotimes_{m=i+1}^{n-1} b_m) \otimes b_n \vdash (\bigotimes_{i=1}^{n-1} b_i^\perp) \otimes b_n \\
L \& & \quad \bigotimes_{i=1}^{n} (b_i \otimes b_j), \bigotimes_{m=i+1}^{n} (b_m \otimes b_j^\perp) \vdash (\bigotimes_{i=1}^{n-1} b_i^\perp) \otimes b_n \\
R \oplus & \quad \bigotimes_{i=1}^{n} (b_i \otimes b_j), \bigotimes_{m=i+1}^{n} (b_m \otimes b_j^\perp) \vdash \bigoplus_{i=1}^{n} (b_i \otimes b_j) \\
L \otimes & \quad \bigotimes_{i=1}^{n} (b_i \otimes b_j), \bigotimes_{m=i+1}^{n} (b_m \otimes b_j^\perp) \vdash \bigoplus_{i=1}^{n} (b_i \otimes b_j) \\
\text{IF2} & \quad a \otimes (a \otimes (a \rightarrow ((\bigotimes_{i=1}^{n} b_i) \otimes (\bigotimes_{m=i+1}^{n} (b_m \otimes b_j^\perp)))))) \vdash a \otimes (a \otimes (a \rightarrow (\bigoplus_{i=1}^{n} (b_i \otimes b_j)))))
\end{align*}
\]
Appendix: Verification of Matching Rules

And-groups of Solitary and Alternative-group to And-group of Optional (P) Features: An And-group of solitary features and an Alternative-group (R) satisfies an And-group of optional (P) features, as shown below:
Appendix: Verification of Matching Rules

\[ IF_2 \]
\[ a \otimes (a \otimes (a \rightarrow (\bigotimes_{i=1}^{l} (b_i \oplus b_i^\perp) \otimes \bigotimes_{m=i+1}^{n} (b_m \otimes \bigotimes_{j \in s} b_j^\perp)))) \vdash a \otimes (a \rightarrow (\bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp))) \]

\[ IF_1 \]
\[ b_1, \ldots, b_{i+1}, b_{i+2}^\perp, \ldots, b_n^\perp \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp) \]

\[ Satisfies \]

\[ IF_1 \]
\[ b_1, \ldots, b_{i+1}, b_{i+2}^\perp, \ldots, b_n^\perp \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp) \]

\[ IF_2 \]
\[ a \otimes (a \otimes (a \rightarrow (\bigotimes_{i=1}^{l} (b_i \oplus b_i^\perp) \otimes \bigotimes_{m=i+1}^{n} (b_m \otimes \bigotimes_{j \in s} b_j^\perp)))) \vdash a \otimes (a \rightarrow (\bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp))) \]

\[ IF_1 \]
\[ b_1, \ldots, b_{i+1}, b_{i+2}^\perp, \ldots, b_n^\perp \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp) \]

\[ Satisfies \]

\[ IF_1 \]
\[ b_1, \ldots, b_{i+1}, b_{i+2}^\perp, \ldots, b_n^\perp \vdash \bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp) \]

\[ IF_2 \]
\[ a \otimes (a \otimes (a \rightarrow (\bigotimes_{i=1}^{l} (b_i \oplus b_i^\perp) \otimes \bigotimes_{m=i+1}^{n} (b_m \otimes \bigotimes_{j \in s} b_j^\perp)))) \vdash a \otimes (a \rightarrow (\bigotimes_{i=1}^{n} (b_i \oplus b_i^\perp))) \]
B.10  Rules for And-groups of an Alternative-group (P) and Solitary Features

Provider-based choice of an Alternative-group along with solitary features satisfies the following set of requirements.

And-groups of Solitary and Alternative-group to Or-group (P): An And-group of solitary and Alternative-group (P) satisfies an Or-group (P), as shown below:

\[
\begin{align*}
\text{id} & \cdot \mathcal{b} \cdot \mathcal{a} \cdot \mathcal{Satisfies} \cdot \mathcal{b} \\
\mathcal{b}_1, \ldots, \mathcal{b}_i & \vdash \bigotimes_{l=1}^{i} \mathcal{b}_l \\
\mathcal{b}_1, \ldots, \mathcal{b}_i & \vdash \bigoplus_{m=i+1}^{n} (b_m \bigoplus igoplus_{j \in s} b_j^+) \vdash \bigoplus_{m=i+1}^{n} (b_m \bigoplus igoplus_{j \in s} b_j^+)
\end{align*}
\]

\[\mathcal{R} \bigotimes \]

\[
\begin{align*}
\mathcal{b}_1, \ldots, \mathcal{b}_i, \bigoplus_{m=i+1}^{n} (b_m \bigoplus igoplus_{j \in s} b_j^+) & \vdash \bigotimes_{l=1}^{i} \mathcal{b}_l \\
\end{align*}
\]

\[\mathcal{L} \bigotimes \]

\[
\begin{align*}
\mathcal{b}_1, \ldots, \mathcal{b}_i, \bigoplus_{m=i+1}^{n} (b_m \bigoplus igoplus_{j \in s} b_j^+) & \vdash \bigotimes_{l=1}^{i} \mathcal{b}_l \\
\end{align*}
\]

\[\mathcal{I} F 2 \]

\[
\begin{align*}
a \otimes (a \otimes (a \rightarrow ((\bigotimes_{l=1}^{i} \mathcal{b}_l) \bigoplus_{m=i+1}^{n} (b_m \bigotimes \bigotimes_{j \in s} b_j^+))) \vdash a \otimes (a \rightarrow ((\bigotimes_{l=1}^{i} \mathcal{b}_l) \bigoplus_{m=i+1}^{n} (b_m \bigotimes \bigotimes_{j \in s} b_j^+))
\end{align*}
\]

\[
\begin{align*}
\text{id} & \cdot \mathcal{b} \cdot \mathcal{a} \cdot \mathcal{Satisfies} \cdot \mathcal{b} \\
\mathcal{b}_1, \ldots, \mathcal{b}_i & \vdash \bigoplus_{m=i+1}^{n} (b_m \bigoplus igoplus_{j \in s} b_j^+) \vdash \bigoplus_{m=i+1}^{n} (b_m \bigoplus igoplus_{j \in s} b_j^+)
\end{align*}
\]

\[\mathcal{R} \bigoplus \]

\[
\begin{align*}
\mathcal{b}_1, \ldots, \mathcal{b}_i, \bigoplus_{m=i+1}^{n} (b_m \bigoplus igoplus_{j \in s} b_j^+) & \vdash \bigotimes_{l=1}^{i} \mathcal{b}_l \\
\end{align*}
\]

\[\mathcal{L} \bigoplus \]

\[
\begin{align*}
\mathcal{b}_1, \ldots, \mathcal{b}_i, \bigoplus_{m=i+1}^{n} (b_m \bigoplus igoplus_{j \in s} b_j^+) & \vdash \bigotimes_{l=1}^{i} \mathcal{b}_l \\
\end{align*}
\]

\[\mathcal{L} \& \]

\[
\begin{align*}
\mathcal{b}_1 & \& \mathcal{b}_1^+, \ldots, \mathcal{b}_i & \& \mathcal{b}_i^+, \bigoplus_{m=i+1}^{n} (b_m \bigoplus igoplus_{j \in s} b_j^+) \vdash \bigoplus_{m=i+1}^{n} (b_m \bigotimes \bigotimes_{j \in s} b_j^+)
\end{align*}
\]
An And-group of solitary features and an Alternative-group (P) satisfies an Alternative-group (P), as shown below:

\[
\begin{align*}
& \text{id} \quad b_1^+, \ldots, b_n^+, b_0 \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes b_n \quad \text{id} \quad b_1^+, \ldots, b_n^+, b_{n-2}, b_{n-1}, b_n \vdash (\bigotimes_{i=1}^{n-2} b_i^+) \otimes b_{n-1} \otimes b_n \\
& R\otimes \quad b_1^+, \ldots, b_1^-, b_0 \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes (\bigotimes_{i=1}^{n-2} b_i^+) \otimes b_{n-1} \otimes b_n \\
& L\otimes \quad b_1^+, \ldots, b_1^+, b_0 \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes (\bigotimes_{i=1}^{n-2} b_i^+) \otimes b_{n-1} \otimes b_n \\
& R\oplus \quad b_1^+, \ldots, b_1^+, (\bigotimes_{i=1}^{n-1} b_i^+) \otimes b_n \vdash (\bigotimes_{i=1}^{n-2} b_i^+) \otimes b_{n-1} \otimes b_n \\
& L\oplus \quad b_1^+, \ldots, b_1^+, b_0 \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes (\bigotimes_{i=1}^{n-2} b_i^+) \otimes b_{n-1} \otimes b_n \\
& L\& \quad b_1 & b_1^+, \ldots, b_1 & b_1^+, b_0 \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes (\bigotimes_{i=1}^{n-2} b_i^+) \otimes b_{n-1} \otimes b_n \\
& L\otimes \quad b_1 & b_1^+, \ldots, b_1 & b_1^+, b_0 \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes (\bigotimes_{i=1}^{n-2} b_i^+) \otimes b_{n-1} \otimes b_n \\
& IF2 \quad a \otimes (\bigotimes_{i=1}^{n} (b_i & b_i^+) \otimes (\bigotimes_{i=1}^{n} b_i^+)) \vdash a \otimes (\bigotimes_{i=1}^{n} (b_i & b_i^+))
\end{align*}
\]

And-group of Solitary and Alternative-group to And-group of Optional (P) Features: An And-group of solitary features and an Alternative-group (P) satisfies an And-group of optional (P) features, as shown below:
Appendix: Verification of Matching Rules

IF1
\[ b_1, \ldots, b_{i+1}, b_{i+2}^{\perp}, \ldots, b_n^{\perp} \vdash \bigotimes_{i=1}^n (b_i \oplus b_i^{\perp}) \]

IF2
\[ a \otimes (a \rightarrow (\bigotimes_{i=1}^n b_i^{\perp} \bigotimes_{j=1}^n (b_m \otimes b_j^{\perp}))) \vdash a \otimes (a \rightarrow (\bigotimes_{i=1}^n (b_i \oplus b_i^{\perp}))) \]

Satisfies

IF1
\[ b_1, \ldots, b_i, b_{i+1} \bigodot \bigodot_{m=i+2} b_m^{\perp} \vdash \bigotimes_{i=1}^n (b_i \oplus b_i^{\perp}) \]

IF2
\[ a \otimes (a \rightarrow (\bigotimes_{i=1}^n b_i^{\perp} \bigotimes_{j=1}^n (b_m \otimes b_j^{\perp}))) \vdash a \otimes (a \rightarrow (\bigotimes_{i=1}^n (b_i \oplus b_i^{\perp}))) \]

Satisfies
### B.11 Rules for And-groups of an Or-group (R) and Solitary Features

A requestor-based choice of an Or-group along with solitary features satisfies the following set of requirements.

**And-groups of Solitary and Or-group to Or-group (P):** An And-group of solitary features and an Or-group (R) satisfies Or-group chosen by the provider, as shown below:

\[
\begin{align*}
L \otimes & \quad n \\
\quad b_{i+1} \otimes \bigoplus_{l=i+2}^{n} b_l^i & \vdash \bigotimes_{m=i+1}^{n} \left( b_m \oplus b_m^i \right) \\
L \oplus & \quad n-1 \\
\quad (b_{i+1} \otimes \bigotimes_{l=i+2}^{n} b_l^i) \oplus \left( \bigotimes_{m=i+1}^{n-1} \left( b_m \oplus b_m^i \right) \right) & \vdash \bigotimes_{m=i+1}^{n} \left( b_m \oplus b_m^i \right)
\end{align*}
\]

\[
\begin{align*}
id & \quad i \\
\quad \bigotimes_{l=1}^{i} \left( b_l \oplus b_l^i \right) & \vdash \bigotimes_{l=1}^{i} \left( b_l \oplus b_l^i \right) \\
\quad \bigoplus_{m=i+1}^{n} \left( b_m \oplus b_m^i \right) & \vdash \bigotimes_{i=1}^{n-1} \left( b_i \oplus b_i^i \right)
\end{align*}
\]

\[
\begin{align*}
R \otimes & \quad n \\
\quad \bigotimes_{l=1}^{i} \left( b_l \oplus b_l^i \right), \bigoplus_{m=i+1}^{n} \left( b_m \oplus b_m^i \right) & \vdash \bigotimes_{i=1}^{n-1} \left( b_i \oplus b_i^i \right)
\end{align*}
\]

\[
\begin{align*}
IF2 & \quad a \otimes \left( a \otimes \left( \bigotimes_{l=1}^{i} \left( b_l \oplus b_l^i \right) \right) \right) \vdash \bigotimes_{i=1}^{n} \left( b_i \oplus b_i^i \right)
\end{align*}
\]
Appendix: Verification of Matching Rules

\[
L \otimes \left( \bigotimes_{i} (b_i \& b_i^\perp) \otimes \bigotimes_{m=i+1}^{n} (\otimes_{j \in s} (b_m \otimes b_j^s)) \right) \vdash \bigotimes_{i=1}^{n} b_i
\]

\[
R \oplus \left( \bigotimes_{i} (b_i \& b_i^\perp) \otimes \bigotimes_{m=i+1}^{n} (\otimes_{j \in s} (b_m \otimes b_j^s)) \right) \vdash \bigoplus_{i=1}^{n} (\bigotimes_{j \in s} (b_i \otimes b_j^s))
\]

\[a \otimes (a \otimes (a \rightarrow \left( \bigotimes_{i} (b_i \& b_i^\perp) \otimes \bigotimes_{m=i+1}^{n} (\otimes_{j \in s} (b_m \otimes b_j^s))) \right) \vdash a \otimes (a \rightarrow \left( \bigoplus_{i=1}^{n} (\bigotimes_{j \in s} (b_i \otimes b_j^s))) \right)
\]

\[
Satisfies
\]

\[
\text{id} \quad b_1, \ldots, b_n \vdash \bigotimes_{i=1}^{n} b_i
\]

\[
L \otimes \left( b_1, \ldots, b_n, \bigotimes_{i=1}^{n} b_i \right) \vdash \bigotimes_{i=1}^{n} b_i
\]

\[
L & \left( b_1, \ldots, b_i, \bigotimes_{m=i+1}^{n} (\otimes_{j \in s} (b_m \otimes b_j^s)) \right) \vdash \bigotimes_{i=1}^{n} b_i
\]

\[
R \oplus \left( \bigotimes_{i} (b_i \& b_i^\perp) \otimes \bigotimes_{m=i+1}^{n} (\otimes_{j \in s} (b_m \otimes b_j^s)) \right) \vdash \bigoplus_{i=1}^{n} (\bigotimes_{j \in s} (b_i \otimes b_j^s))
\]

\[a \otimes (a \otimes (a \rightarrow \left( \bigotimes_{i} (b_i \& b_i^\perp) \otimes \bigotimes_{m=i+1}^{n} (\otimes_{j \in s} (b_m \otimes b_j^s))) \right) \vdash a \otimes (a \rightarrow \left( \bigoplus_{i=1}^{n} (\bigotimes_{j \in s} (b_i \otimes b_j^s))) \right)
\]

\[
Satisfies
\]

\[
\text{id} \quad b_1, \ldots, b_n \vdash \bigotimes_{i=1}^{n} b_i
\]

\[
L \otimes \left( b_1^\perp, \ldots, b_i^\perp, b_{i+1}, \ldots, b_n \right) \vdash \bigotimes_{i=1}^{n} b_i \otimes \bigotimes_{m=i+1}^{n} b_m
\]

\[
L \otimes \left( b_1, \ldots, b_i, \bigotimes_{m=i+1}^{n} b_m \right) \vdash \bigotimes_{i=1}^{n} b_i \otimes \bigotimes_{m=i+1}^{n} b_m
\]

\[
R \oplus \left( b_1, \ldots, b_i, \bigotimes_{m=i+1}^{n} b_m \right) \vdash \bigotimes_{i=1}^{n} b_i \otimes \bigotimes_{m=i+1}^{n} b_m
\]

\[
L \oplus \left( \bigotimes_{i} (b_i^\perp, \ldots, b_i^\perp, \bigotimes_{m=i+1}^{n} b_m) \right) \vdash \bigotimes_{i=1}^{n} b_i \otimes \bigotimes_{m=i+1}^{n} b_m
\]
An And-group of solitary features and an Or-group (R) satisfies Alternative-groups:

\[ R \oplus \]

\[ b_1 \oplus b_1^+, \ldots, b_i \oplus b_i^+, (\bigotimes_{m=i+1}^{n} b_m^+) \otimes b_n \vdash (\bigoplus_{j \in s} (b_i \otimes b_j^+)) \]

\[ L \& \]

\[ b_1 \oplus b_1^+, \ldots, b_i \oplus b_i^+, (\bigotimes_{m=i+1}^{n} b_m^+) \otimes b_n \vdash (\bigoplus_{j \in s} (b_i \otimes b_j^+)) \]

\[ L \otimes \]

\[ \bigotimes_{i=1}^{n} (b_i \oplus b_i^+) \otimes (\bigotimes_{m=i+1}^{n} (\bigotimes_{j \in s} (b_m \otimes b_j^+))) \vdash (\bigoplus_{j \in s} (b_i \otimes b_j^-)) \]

\[ IF2 \]

\[ a \otimes (a \otimes (a \leftarrow (\bigotimes_{i=1}^{n} (b_i \oplus b_i^+) \otimes (\bigotimes_{m=i+1}^{n} (\bigotimes_{j \in s} (b_m \otimes b_j^+))))) \vdash a \otimes (a \otimes (a \leftarrow (\bigotimes_{i=1}^{n} (b_i \otimes b_j^-)))) \]

And-group of Optional (R) Features and Or-group to Alternative-groups:

An And-group of solitary features and an Or-group (R) satisfies Alternative-groups of requestor’s choice and provider’s choice, respectively, as shown below:

\[ id \]

\[ b_1^+, \ldots, b_{n-1}^+, b_n \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes b_n \]

\[ L \otimes \]

\[ b_1^+, \ldots, b_{n-1}^+, (\bigotimes_{m=i+1}^{n} b_m^+) \otimes b_n \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes b_n \]

\[ L \& \]

\[ b_1^+, \ldots, b_{n-1}^+, \& (\bigotimes_{m=i+1}^{n} b_m^+) \otimes b_n \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes b_n \]

\[ R \& \]

\[ b_1^+, \ldots, b_{n-1}^+, \& (\bigotimes_{m=i+1}^{n} b_m^+) \otimes (\bigotimes_{j \in s} (b_j^-)) \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes (\bigotimes_{j \in s} (b_j^-)) \]

\[ L \& \]

\[ b_1 \& b_1^+, \ldots, b_i \& b_i^+, \& (\bigotimes_{m=i+1}^{n} b_m^+) \otimes (\bigotimes_{j \in s} (b_j^-)) \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes (\bigotimes_{j \in s} (b_j^-)) \]

\[ L \otimes \]

\[ \bigotimes_{i=1}^{n} (b_i \& b_i^+) \otimes (\bigotimes_{m=i+1}^{n} (\bigotimes_{j \in s} (b_m \otimes b_j^+))) \vdash (\bigotimes_{i=1}^{n} (b_i \& b_i^+)) \]

\[ IF2 \]

\[ a \otimes (a \otimes (a \leftarrow (\bigotimes_{i=1}^{n} (b_i \& b_i^+) \otimes (\bigotimes_{m=i+1}^{n} (\bigotimes_{j \in s} (b_m \otimes b_j^+))))) \vdash a \otimes (a \otimes (a \leftarrow (\bigotimes_{i=1}^{n} (b_i \& b_i^+)))) \]
Appendix: Verification of Matching Rules

185

solitary features, as shown below:

An And-group of solitary features and an Or-group (R) satisfies an And-group of solitary features:

\[ b_1^+, \ldots, b_{n-1}^+, b_n \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes b_n \]

\[ b_1^+, \ldots, b_i^+ \otimes b_n \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes b_n \]

\[ b_1^+, \ldots, b_i^+ \otimes (\bigotimes_{m=i+1}^{n} \bigotimes_{j \in s} b_j^+) \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes b_n \]

\[ b_1^+ \otimes b_i^+ \otimes b_n \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes b_n \]

\[ (\bigotimes_{i=1}^{n-1} b_i^+) \otimes \bigotimes_{m=i+1}^{n} \bigotimes_{j \in s} b_j^+ \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes b_n \]

And-groups of Solitary and Or-group to And-group of Solitary Features:

An And-group of solitary features and an Or-group (R) satisfies an And-group of solitary features, as shown below:

\[ (b_1^+ \otimes b_i^+) \otimes \bigotimes_{m=i+1}^{n} \bigotimes_{j \in s} b_j^+ \vdash (\bigotimes_{i=1}^{n-1} b_i^+) \otimes b_n \]

\[ a \otimes (a \otimes (a \rightarrow (\bigotimes_{i=1}^{n-1} b_i^+) \otimes \bigotimes_{m=i+1}^{n} \bigotimes_{j \in s} b_j^+))) \vdash a \otimes (a \otimes (a \rightarrow (\bigotimes_{i=1}^{n-1} b_i^+) \otimes b_n)) \]
Satisfies

\[
\begin{align*}
\text{id} & : b_1, \ldots, b_n \vdash b_i \\
L\& & : b_1 & b_1^+, \ldots, b_i & b_i^+, b_{i+1}, \ldots, b_n \vdash b_i \\
L\otimes & : \bigotimes_{i=1}^n (b_i & b_i^+) & \bigotimes_{m=i+1}^n b_m \vdash b_i \\
L\& & : \bigotimes_{i=1}^n (b_i & b_i^+) & \bigotimes_{m=i+1}^n (\bigotimes_{j\in s}^n b_j^+) \vdash b_i \\
L\otimes & : \bigotimes_{i=1}^n (b_i & b_i^+) & \bigotimes_{m=i+1}^n (\bigotimes_{j\in s}^n b_j^+) & \bigotimes_{m=i+1}^n b_m \vdash b_i \\
IF^2 & : a \otimes (a \otimes (a \rightarrow a \otimes \bigotimes_{i=1}^n (b_i & b_i^+) & \bigotimes_{m=i+1}^n (\bigotimes_{j\in s}^n b_j^+) \bigotimes_{m=i+1}^n b_m \vdash b_i \\
R\oplus & : b_1^+, \ldots, b_n^+, b_n \vdash b_1^+ & b_n^+ \bigotimes_{i=1}^{n-1} b_i^+ \\
id & : b_1^+, \ldots, b_n^+, b_n \vdash b_1^+ & b_n^+ \bigotimes_{i=1}^{n-2} b_i^+ \\
R\oplus & : b_1^+, \ldots, b_n^+, b_n \vdash b_1^+ & b_n^+ \bigotimes_{i=1}^{n-1} b_i^+ \\
\end{align*}
\]

B.12 Rules for And-groups of an Or-group (P) and Solitary Features

And-groups of Solitary and Or-group to Or-group (P): An And-group of solitary features and an Or-group (P) satisfies an Or-group (P), as shown below:
And-groups of Solitary and Or-group to And-group of Optional (P) Features: An And-group of solitary features and an Or-group (P) satisfies an And-
group of optional (P) features, as shown below:

\[
\begin{align*}
\text{IF1} & \quad b_{i+1}, b_{i+2}, \ldots, b_n \vdash \bigoplus_{m=i+1}^n (b_m \oplus b_m^\perp) \\
L & \quad b_{i+1} \bigotimes \bigotimes_{l=i+1}^n b_l \vdash \bigotimes_{m=i+1}^n (b_m \oplus b_m^\perp) \\
\text{IF1} & \quad b_{i+1}, \ldots, b_{n-1}, b_n \vdash \bigoplus_{m=i+1}^n (b_m \oplus b_m^\perp) \\
L & \quad (\bigotimes_{m=i+1}^{n-1} b_m^\perp) \otimes b_n \vdash \bigotimes_{m=i+1}^n (b_m \oplus b_m^\perp)
\end{align*}
\]

\[
\begin{align*}
\text{R} & \quad \bigoplus_{i=1}^n (b_i \oplus b_i^\perp) \vdash \bigotimes_{m=i+1}^n (b_m \oplus b_m^\perp) \\
L & \quad \bigotimes_{i=1}^n (b_i \oplus b_i^\perp) \otimes \bigoplus_{m=i+1}^n (b_m \oplus b_m^\perp) \\
\text{IF2} & \quad a \otimes (a \otimes (a \rightarrow ((\bigotimes_{i=1}^n (b_i \oplus b_i^\perp)) \oplus (\bigotimes_{i=1}^n (b_i \oplus b_i^\perp))))) \\
\end{align*}
\]
IF2

\[ a \otimes (a \otimes (a \to (\bigotimes_{i=1}^{l} b_i) \otimes \bigoplus_{m=1+1}^{n} (\bigoplus_{j \in s} (b_m \otimes \bigoplus_{i} b_{j}^\perp)))) \vdash a \otimes (a \otimes (a \to (\bigotimes_{i=1}^{l} (b_i \otimes b_{j}^\perp)))) \]

### B.13 Rule for Shareable Feature

**Shareable to Resource Feature:** A shareable feature satisfies a requirement of a resource feature as shown below:

\[
\begin{array}{ccc}
\text{a} & \text{Satisfies} & \text{a} \\
\end{array}
\]

\[
\begin{array}{ccc}
\text{id} & \vdash & \text{a} \\
\text{D!} & \vdash & \text{a} \\
\end{array}
\]
Appendix: C

Verification of Merging Rules

C.1 Merging Rule for Service Feature Diagrams

Merging of Service Feature Diagrams: For the merging of feature diagrams, all the diagrams must have at least one common feature, which can be a resource in at most one of the diagrams. A common feature may be non-root in at most one of the diagrams being merged. Let us check the proof of the rule:

\[
\begin{align*}
\alpha & \quad sf_1 & \quad \ldots & \quad sf_n & \quad \text{Merge} & \quad \alpha \\
\text{id} & \vdash sf_1 & \quad \ldots & \quad \text{id} & \vdash sf_n \\
\quad & \quad R \otimes & \quad \vdash sf_1, \ldots, sf_n & \quad \vdash sf_1 \otimes \cdots \otimes sf_n \\
\quad & \quad !a \vdash !a & \quad 1F2 & \quad \alpha \otimes (\alpha \otimes (\alpha \rightarrow sf_1), \ldots, sf_n) \vdash \alpha \otimes (\alpha \otimes (\alpha \rightarrow (sf_1 \otimes \cdots \otimes sf_n))) \\
\quad & \quad L \rightarrow & \quad \alpha \otimes (\alpha \otimes (\alpha \rightarrow sf_1), \ldots, !a, !a \rightarrow sf_n) \vdash \alpha \otimes (\alpha \otimes (\alpha \rightarrow (sf_1 \otimes \cdots \otimes sf_n))) \\
\quad & \quad L \otimes & \quad \alpha \otimes (\alpha \otimes (\alpha \rightarrow sf_1), \ldots, !a \otimes (!a \rightarrow sf_n)) \vdash \alpha \otimes (\alpha \otimes (\alpha \rightarrow (sf_1 \otimes \cdots \otimes sf_n))) \\
\quad & \quad W! & \quad \alpha \otimes (\alpha \otimes (\alpha \rightarrow sf_1), \ldots, !a \otimes (!a \otimes (!a \rightarrow sf_n)) \vdash \alpha \otimes (\alpha \otimes (\alpha \rightarrow (sf_1 \otimes \cdots \otimes sf_n))) \\
\quad & \quad L \otimes & \quad \alpha \otimes (\alpha \otimes (\alpha \rightarrow sf_1), \ldots, !a \otimes (!a \otimes (a \rightarrow sf_n)) \vdash \alpha \otimes (\alpha \otimes (\alpha \rightarrow (sf_1 \otimes \cdots \otimes sf_n)))
\end{align*}
\]

Although the diamond symbol can either be a resource or a shareable, the diamond symbol of the feature diagram being merged and the resulting feature diagram should have the same type, i.e., either a resource or a shareable feature. For example,
if the diamond symbol in the feature diagram being merged is a resource feature, then the root of the resulting feature diagram will be a resource feature.

C.2 Rules for Merging of Shareable Features

As already stated in the previous chapter, we consider three categories for merging shareable features: 1) merging solitary features, 2) merging solitary and group features, 3) merging group features.

Merging Solitary Features: As stated in the Chapter 6, we use generalized feature diagrams for this rule, as shown below. The combination of solid and dashed lines means that the rule gives the same result, if the feature is selected by the requestor or by the provider. The relevance-less features state that the rule gives the same result for $b$ being mandatory or optional.

![Feature Diagram](image)

The above-mentioned generic rule can produce 9 concrete merging rules. Instead of providing the correctness for each rule, we present three meta-proofs for the deduction, where a meta proposition $B$ is used to represent the diamond symbol $b$. If the diamond symbol $b$ represents a mandatory feature then $B$ will be replaced by $\beta$, if the diamond symbol $b$ represents an optional (R) then $B$ is replaced by $\beta \& \beta^\perp$, in case of a diamond symbol $b$ being an optional (P) feature, $B$ is replaced by $\beta \oplus \beta^\perp$.

A meta-proof can not be generated for all the generic-rules presented in Chapter 6. It is out of the scope to explore the techniques for identify the reasons why a meta-proof can be generated for a generic-rule. Here, we only generate the meta-proofs wherever possible based on our observation.

So, each meta-proof of the above rule represents three concrete deductions, which we get using the corresponding values for $B$. The shareable feature $b$ can either be
a mandatory, or an optional (R), an optional (P) feature. First let us consider it as a mandatory, using the meta proposition for the rest:

$$\frac{id}{\exists B} \quad \frac{B \vdash B}{W! \quad \exists B, !b \vdash B} \quad \frac{B \otimes !b \vdash B}{L\otimes \quad \exists \alpha \otimes (\alpha \otimes (\alpha \to (B \otimes !b)))) \vdash \alpha \otimes (\alpha \otimes (\alpha \to B)))}$$

Now, let us consider shareable feature $b$ as an optional (R) feature, using the meta propositions for the rest:

$$\frac{id}{\exists B} \quad \frac{B \vdash B}{W! \quad \exists B, !b \vdash B} \quad \frac{B \otimes !b \otimes !b \vdash B}{L\otimes \quad \exists \alpha \otimes (\alpha \otimes (\alpha \to B))) \vdash \alpha \otimes (\alpha \otimes (\alpha \to B)))}$$

There may exists one more variation of above rule, where a shareable feature $b$ is an optional (P) feature:

$$\frac{Axiom}{\exists B \vdash \beta \oplus \beta^+} \quad \frac{W!}{\exists B, !b \vdash \beta \oplus \beta^+} \quad \frac{L\oplus / \exists B, !b \vdash \beta \oplus \beta^+}{L\otimes / \exists \alpha \otimes (\alpha \to (B \otimes (\beta \oplus \beta^+))) \vdash \alpha \otimes (\alpha \to (\beta \oplus \beta^+)))}$$

The derivation of the Axiom $\exists B \vdash \beta \oplus \beta^+$, used in the beginning of the above deduction is explained in Appendix A. The above deductions state that if any of the features being merged is chosen by the provider then the selection rights of the resulting feature should also be given to the provider.

Let us now consider a case, where common features (being merged) exist at different levels in a service feature diagram, as shown below:

The above rule gives 9 different combinations. We will use the same technique which we have used in the previous rule, we will consider the shareable feature
b for being mandatory, optional (R) and optional (P) feature, respectively, while deducing the correctness for the rule above. Let us first consider a shareable feature b as mandatory using meta-proposition for the rest:

\[
\begin{align*}
\frac{id}{\delta \otimes (\delta \otimes (\delta \otimes \rightarrow B)) \vdash \delta \otimes (\delta \otimes \rightarrow B)} \\
\frac{W!}{yb, \delta \otimes (\delta \otimes \rightarrow B) \vdash \delta \otimes (\delta \otimes \rightarrow B)} \\
\frac{L\otimes}{yb \otimes \delta \otimes (\delta \otimes \rightarrow B) \vdash \delta \otimes (\delta \otimes \rightarrow B)} \\
\frac{IF2}{\alpha \otimes (\alpha \rightarrow (yb \otimes \delta \otimes (\delta \otimes \rightarrow B))) \vdash \alpha \otimes (\alpha \rightarrow (\delta \otimes (\delta \otimes \rightarrow B))))}
\end{align*}
\]

Let us now consider the shareable feature b as an optional (R) feature:

\[
\begin{align*}
\frac{id}{\delta \otimes (\delta \otimes \rightarrow B) \vdash \delta \otimes (\delta \otimes \rightarrow B)} \\
\frac{W!}{yb, \delta \otimes (\delta \otimes \rightarrow B) \vdash \delta \otimes (\delta \otimes \rightarrow B)} \\
\frac{Lk}{yb \otimes \delta \otimes (\delta \otimes \rightarrow B) \vdash \delta \otimes (\delta \otimes \rightarrow B)} \\
\frac{L\otimes}{(yb \otimes \delta \otimes (\delta \otimes \rightarrow B)) \vdash \delta \otimes (\delta \otimes \rightarrow B)} \\
\frac{IF2}{\alpha \otimes (\alpha \rightarrow ((yb \otimes \delta \otimes (\delta \otimes \rightarrow B)))) \vdash \alpha \otimes (\alpha \rightarrow (\delta \otimes (\delta \otimes \rightarrow B))))}
\end{align*}
\]

Let us now consider a variation of this rule, where the shareable feature b is an optional (P) feature:

\[
\begin{align*}
\frac{Ax}{\delta \otimes (\delta \otimes (\delta \otimes \rightarrow B)) \vdash \delta \otimes (\delta \otimes \rightarrow (\beta \oplus \beta^+))} \\
\frac{W1}{yb, \delta \otimes (\delta \otimes \rightarrow B) \vdash \delta \otimes (\delta \otimes \rightarrow (\beta \oplus \beta^+))} \\
\frac{W1}{y_b^+, \delta \otimes (\delta \otimes \rightarrow B) \vdash \delta \otimes (\delta \otimes \rightarrow (\beta \oplus \beta^+))} \\
\frac{Ax}{\delta \otimes (\delta \otimes \rightarrow B) \vdash \delta \otimes (\delta \otimes \rightarrow (\beta \oplus \beta^+))} \\
\frac{L\otimes}{(yb \otimes y_b^+, \delta \otimes (\delta \otimes \rightarrow B)) \vdash \delta \otimes (\delta \otimes \rightarrow (\beta \oplus \beta^+))} \\
\frac{L\otimes}{(yb \otimes y_b^+, \delta \otimes (\delta \otimes \rightarrow B)) \vdash \delta \otimes (\delta \otimes \rightarrow (\beta \oplus \beta^+))} \\
\frac{IF2}{\alpha \otimes (\alpha \rightarrow ((yb \otimes y_b^+, \delta \otimes (\delta \otimes \rightarrow B)))) \vdash \alpha \otimes (\alpha \rightarrow (\delta \otimes (\delta \otimes \rightarrow (\beta \oplus \beta^+))))}
\end{align*}
\]

C.2.1 Merging of Solitary and Group Features:

We consider the following types for the merging of solitary and group features:

Merging of an Alternative-group (R) and a Mandatory Feature:

\[
\begin{align*}
id \quad \frac{\&\varepsilon (\beta_i \otimes (\delta^+ \otimes \delta^1)) \& \varepsilon \delta \otimes (\beta_i^1 \otimes \delta^1) \vdash \&\varepsilon (\beta_i \otimes (\delta^+ \otimes \delta^1)) \& \varepsilon \delta \otimes (\beta_i^1 \otimes \delta^1)}{\&\varepsilon (\beta_i \otimes (\delta^+ \otimes \delta^1)) \& \varepsilon \delta \otimes (\beta_i^1 \otimes \delta^1)}
\end{align*}
\]
\[
\begin{align*}
W! & \\
L \otimes & \left( \bigotimes_{i=1}^{n} (\beta_i \otimes (\beta_j^i \otimes \delta^i)) \& (\delta \otimes \bigotimes_{i=1}^{n} \beta_i^i) \right) & !b & \vdash \\
L \otimes & \left( \bigotimes_{i=1}^{n} (\beta_i \otimes (\beta_j^i \otimes \delta^i)) \& (\delta \otimes \bigotimes_{i=1}^{n} \beta_i^i) \right) & !b & \vdash \\
IF & \alpha \otimes (\alpha \rightarrow (\bigotimes_{i=1}^{n} (\beta_i \otimes (\beta_j^i \otimes \delta^i)) \& (\delta \otimes \bigotimes_{i=1}^{n} \beta_i^i))) & \vdash \\
& \alpha \otimes (\alpha \rightarrow (\bigotimes_{i=1}^{n} (\beta_i \otimes (\beta_j^i \otimes \delta^i)) \& (\delta \otimes \bigotimes_{i=1}^{n} \beta_i^i))) & \vdash \\
\end{align*}
\]
Merging of an Alternative-group (R) and Optional (P) Features:

<table>
<thead>
<tr>
<th></th>
<th>id</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\delta \otimes \bigwedge_{i=1}^{n} \beta_{i}^{1} \vdash \delta \otimes \bigwedge_{i=1}^{n} \beta_{i}^{1}$</td>
<td>$\delta \otimes \bigwedge_{i=1}^{n} \beta_{i}^{1} \vdash \delta \otimes \bigwedge_{i=1}^{n} \beta_{i}^{1}$</td>
</tr>
<tr>
<td></td>
<td>$W! \vdash \delta \otimes \bigwedge_{i=1}^{n} \beta_{i}^{1}, !b_{1}, \ldots, !b_{n} \vdash \delta \otimes \bigwedge_{i=1}^{n} \beta_{i}^{1}$</td>
<td>$W! \vdash \delta \otimes \bigwedge_{i=1}^{n} \beta_{i}^{1}, !b_{1}^{+}, \ldots, !b_{n}^{+} \vdash \delta \otimes \bigwedge_{i=1}^{n} \beta_{i}^{1}$</td>
</tr>
<tr>
<td></td>
<td>$L \oplus \vdash \delta \otimes \bigwedge_{i=1}^{n} \beta_{i}^{1}, !b_{1} + !b_{1}^{+}, \ldots, !b_{n} + !b_{n}^{+} \vdash \delta \otimes \bigwedge_{i=1}^{n} \beta_{i}^{1}$</td>
<td>$R \oplus \vdash \delta \otimes \bigwedge_{i=1}^{n} \beta_{i}^{1}, !b_{1} + !b_{1}^{+}, \ldots, !b_{n} + !b_{n}^{+} \vdash \bigoplus (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1})) + (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1})$</td>
</tr>
<tr>
<td></td>
<td>$L &amp; \vdash \bigwedge_{i=1}^{n} (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1})) &amp; (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1}), !b_{1} + !b_{1}^{+}, \ldots, !b_{n} + !b_{n}^{+} \vdash \bigoplus (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1})) + (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1})$</td>
<td>$\bigwedge_{i=1}^{n} (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1})) &amp; (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1}), !b_{1} + !b_{1}^{+}, \ldots, !b_{n} + !b_{n}^{+} \vdash \bigoplus (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1})) + (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1})$</td>
</tr>
<tr>
<td></td>
<td>$L \otimes \vdash (\bigwedge_{i=1}^{n} (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1})) &amp; (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1})), !b_{1} + !b_{1}^{+}, \ldots, !b_{n} + !b_{n}^{+} \vdash (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1})) + (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1})$</td>
<td>$\bigwedge_{i=1}^{n} (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1})) &amp; (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1}), !b_{1} + !b_{1}^{+}, \ldots, !b_{n} + !b_{n}^{+} \vdash (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1})) + (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1})$</td>
</tr>
<tr>
<td>IF2</td>
<td>$\alpha \otimes (\alpha \otimes (\alpha \rightarrow ((\bigwedge_{i=1}^{n} (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1}))) &amp; (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1})), !b_{1} + !b_{1}^{+}, \ldots, !b_{n} + !b_{n}^{+})) \vdash \alpha \otimes (\alpha \otimes (\alpha \rightarrow (\bigwedge_{i=1}^{n} (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1}))) &amp; (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1}))))$</td>
<td>$\alpha \otimes (\alpha \rightarrow ((\bigwedge_{i=1}^{n} (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1}))) &amp; (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1})), !b_{1} + !b_{1}^{+}, \ldots, !b_{n} + !b_{n}^{+}) \vdash (\bigwedge_{i=1}^{n} (\delta \otimes \bigotimes_{j \in s} (\beta_{j}^{+} \otimes \delta^{1}))) &amp; (\delta \otimes \bigotimes_{i=1}^{n} \beta_{i}^{1})$</td>
</tr>
</tbody>
</table>
Merging of an Alternative-group (R) and Optional (R) Features:

\[
\begin{align*}
\text{id} & \quad \land \left( \forall_{i=1}^{n} (\beta_i \otimes \bigotimes_{j \in s}(\beta_j^\perp \otimes \delta^\perp)) \land (\delta \otimes \bigotimes_{i=1}^{n} \beta_i^\perp) \right) \iff \left( \forall_{i=1}^{n} (\beta_i \otimes \bigotimes_{j \in s}(\beta_j^\perp \otimes \delta^\perp)) \land (\delta \otimes \bigotimes_{i=1}^{n} \beta_i^\perp) \right) \\
W! & \quad \vdash \left( \forall_{i=1}^{n} (\beta_i \otimes \bigotimes_{j \in s}(\beta_j^\perp \otimes \delta^\perp)) \land (\delta \otimes \bigotimes_{i=1}^{n} \beta_i^\perp), \forall_{i=1}^{n} \beta_i^\perp \right) \iff \left( \forall_{i=1}^{n} (\beta_i \otimes \bigotimes_{j \in s}(\beta_j^\perp \otimes \delta^\perp)) \land (\delta \otimes \bigotimes_{i=1}^{n} \beta_i^\perp) \right)
\end{align*}
\]

IF2

\[
\alpha \otimes (\alpha \otimes (\alpha \rightarrow \left( \land \left( \forall_{i=1}^{n} (\beta_i \otimes \bigotimes_{j \in s}(\beta_j^\perp \otimes \delta^\perp)) \land (\delta \otimes \bigotimes_{i=1}^{n} \beta_i^\perp), \forall_{i=1}^{n} \beta_i^\perp) \right) \right)) + \alpha \rightarrow \left( \land \left( \forall_{i=1}^{n} (\beta_i \otimes \bigotimes_{j \in s}(\beta_j^\perp \otimes \delta^\perp)) \land (\delta \otimes \bigotimes_{i=1}^{n} \beta_i^\perp) \right) \right)
\]

Merging of an Alternative-group (P) and a Mandatory Feature:
Merging of an Alternative-group (P) and Optional (P) Features:
Merging of Alternative-group (P) and Optional (R) Features:

In the above rule, we have used a single mandatory feature with an Alternative-group, because an Alternative-group does not allow the selection of more than one subfeature. Merging more than one mandatory feature with an Alternative-group may not be acceptable for one of the offers that have been merged.
We can use an And-group of mandatory features while merging it with an Or-group, as shown below:

**Merging of Or-group (R) and Mandatory Features:**

\[
\begin{align*}
\alpha & \quad \text{Merge} \\
\{b_1, \ldots, b_n, d\} & \quad \{b_1, \ldots, b_n\}
\end{align*}
\]

\[
\begin{align*}
id & \quad \frac{n}{i=1} (\& \beta_i \otimes (\& \beta_j \otimes \delta^v)) & \quad (\delta \otimes \bigotimes_{i=1}^{n} \beta_i^\perp) \\
& \quad \vdash (\& \beta_i \otimes (\& \beta_j \otimes \delta^v)) & \quad (\delta \otimes \bigotimes_{i=1}^{n} \beta_i^\perp)
\end{align*}
\]

**Merging of Or-group (R) and Optional (P) Features:**

\[
\begin{align*}
\alpha & \quad \text{Merge} \\
\{b_1, \ldots, b_n, d\} & \quad \{b_1, \ldots, b_n\}
\end{align*}
\]

\[
\begin{align*}
id & \quad \frac{n}{i=1} (\& \beta_i \otimes (\& \beta_j \otimes \delta^v)) & \quad (\delta \otimes \bigotimes_{i=1}^{n} \beta_i) \\
& \quad \vdash (\& \beta_i \otimes (\& \beta_j \otimes \delta^v)) & \quad (\delta \otimes \bigotimes_{i=1}^{n} \beta_i)
\end{align*}
\]
Merging of an Or-group (R) and Optional (R) Features:
Merging of an Or-group (P) and Mandatory Features:

\[
\begin{align*}
W! & \quad \bigoplus_{i=1}^{\infty} \left( \mathcal{L}(\beta_i \otimes \bigotimes_{j \in s} (\beta_j^v \otimes \delta^v)) \right) \oplus \left( \delta \otimes \bigotimes_{i=1}^{n} \beta_i^\bot \right), !b_1, \ldots, !b_n \vdash \bigoplus_{i=1}^{\infty} \left( \mathcal{L}(\beta_i \otimes \bigotimes_{j \in s} (\beta_j^v \otimes \delta^v)) \right) \oplus \left( \delta \otimes \bigotimes_{i=1}^{n} \beta_i^\bot \right) \\
L\& & \quad \bigoplus_{i=1}^{\infty} \left( \mathcal{L}(\beta_i \otimes \bigotimes_{j \in s} (\beta_j^v \otimes \delta^v)) \right) \& \left( \delta \otimes \bigotimes_{i=1}^{n} \beta_i^\bot \right), !b_1, \ldots, !b_n \vdash \bigoplus_{i=1}^{\infty} \left( \mathcal{L}(\beta_i \otimes \bigotimes_{j \in s} (\beta_j^v \otimes \delta^v)) \right) \& \left( \delta \otimes \bigotimes_{i=1}^{n} \beta_i^\bot \right) \\
L\otimes & \quad \bigoplus_{i=1}^{\infty} \left( \mathcal{L}(\beta_i \otimes \bigotimes_{j \in s} (\beta_j^v \otimes \delta^v)) \right) \otimes \left( \delta \otimes \bigotimes_{i=1}^{n} \beta_i^\bot \right), !b_1, \ldots, !b_n \vdash \bigoplus_{i=1}^{\infty} \left( \mathcal{L}(\beta_i \otimes \bigotimes_{j \in s} (\beta_j^v \otimes \delta^v)) \right) \otimes \left( \delta \otimes \bigotimes_{i=1}^{n} \beta_i^\bot \right) \\
IF2 & \quad \alpha \otimes \left( \alpha \otimes \left( \alpha \rightarrow \bigoplus_{i=1}^{\infty} \left( \mathcal{L}(\beta_i \otimes \bigotimes_{j \in s} (\beta_j^v \otimes \delta^v)) \right) \otimes \left( \delta \otimes \bigotimes_{i=1}^{n} \beta_i^\bot \right) \right) \right) \vdash \alpha \otimes \left( \alpha \otimes \left( \alpha \rightarrow \bigoplus_{i=1}^{\infty} \left( \mathcal{L}(\beta_i \otimes \bigotimes_{j \in s} (\beta_j^v \otimes \delta^v)) \right) \otimes \left( \delta \otimes \bigotimes_{i=1}^{n} \beta_i^\bot \right) \right) \right)
\end{align*}
\]
Appendix: Verification of Merging Rules

\[
L \otimes \bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j \otimes \delta^v)) \oplus (\delta \otimes \beta_i^+) = \bigoplus_{i=1}^{n} \beta_i^+ \vdash \bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j \otimes \delta^v)) \oplus (\delta \otimes \beta_i^+) \\
I F^2
\]

\[
\alpha \otimes (\alpha \otimes (\alpha \rightarrow ((\bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j \otimes \delta^v)) \oplus (\delta \otimes \beta_i^+)) \oplus (\bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j \otimes \delta^v)) \oplus (\delta \otimes \beta_i^+))) + \bigoplus_{i=1}^{n} \beta_i^+) \vdash \alpha \otimes (\alpha \otimes (\alpha \rightarrow ((\bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j \otimes \delta^v)) \oplus (\delta \otimes \beta_i^+)) \oplus (\bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j \otimes \delta^v)) \oplus (\delta \otimes \beta_i^+)))
\]

Merging of an Or-group (P) and Optional (P) Features:

\[
\begin{array}{c}
\alpha \\
\hline
\vdash \bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j \otimes \delta^v)) \oplus (\delta \otimes \beta_i^+) + \bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j \otimes \delta^v)) + (\delta \otimes \beta_i^+)
\end{array}
\]

IF^2

\[
\alpha \otimes (\alpha \otimes (\alpha \rightarrow ((\bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j \otimes \delta^v)) \oplus (\delta \otimes \beta_i^+)) \oplus (\bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j \otimes \delta^v)) \oplus (\delta \otimes \beta_i^+))) + \bigoplus_{i=1}^{n} \beta_i^+) \vdash \alpha \otimes (\alpha \otimes (\alpha \rightarrow ((\bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j \otimes \delta^v)) \oplus (\delta \otimes \beta_i^+)) \oplus (\bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j \otimes \delta^v)) \oplus (\delta \otimes \beta_i^+)))
\]
Merging of an Or-group (P) and Optional (R) Features:

Let us discuss a slight variation of the rule shown above, where we merge group features as shown below:

Merging of Alternative-group (R) and Alternative-group (R):
Appendix: Verification of Merging Rules

\[
\begin{align*}
\text{id} & \frac{n}{i=1} (\beta_i \otimes (\bigotimes_{j \in s} (\beta_j^\perp \otimes \delta^\perp))) \& (\delta \otimes \bigotimes_{i=1}^n \beta_i^\perp) \vdash \\
\frac{n}{i=1} (\beta_i \otimes (\bigotimes_{j \in s} (\beta_j^\perp \otimes \delta^\perp))) \& (\delta \otimes \bigotimes_{i=1}^n \beta_i^\perp) & \vdash \\
\vdots & \\
W! & b_1, b_2^\perp, \ldots, b_n^\perp, \frac{n}{i=1} (\beta_i \otimes (\bigotimes_{j \in s} (\beta_j^\perp \otimes \delta^\perp))) \& (\delta \otimes \bigotimes_{i=1}^n \beta_i^\perp) \vdash \\
\vdots & \\
L \otimes & b_1 \otimes \frac{n}{i=2} b_i^\perp, \frac{n}{i=1} (\beta_i \otimes (\bigotimes_{j \in s} (\beta_j^\perp \otimes \delta^\perp))) \& (\delta \otimes \bigotimes_{i=1}^n \beta_i^\perp) \vdash \\
\vdots & \\
L \& & \frac{n}{i=1} (\bigotimes_{j \in s} (b_i \otimes b_j^\perp)), \frac{n}{i=1} (\beta_i \otimes (\bigotimes_{j \in s} (\beta_j^\perp \otimes \delta^\perp))) \& (\delta \otimes \bigotimes_{i=1}^n \beta_i^\perp) \vdash \\
\vdots & \\
L \otimes & \frac{n}{i=1} (\bigotimes_{j \in s} (b_i \otimes b_j^\perp)) \otimes (\bigotimes_{i=1}^n (\frac{n}{i=1} (\beta_i \otimes (\bigotimes_{j \in s} (\beta_j^\perp \otimes \delta^\perp))) & (\delta \otimes \bigotimes_{i=1}^n \beta_i^\perp)) \vdash \\
\vdots & \\
IF2 & \alpha \otimes (\alpha \otimes (\alpha \longrightarrow (\bigotimes_{i=1}^n (\beta_i \otimes (\bigotimes_{j \in s} (\beta_j^\perp \otimes \delta^\perp))) \& (\delta \otimes \bigotimes_{i=1}^n \beta_i^\perp)))) \longrightarrow \\
\alpha \otimes (\alpha \otimes (\alpha \longrightarrow ((\bigotimes_{i=1}^n (\beta_i \otimes (\bigotimes_{j \in s} (\beta_j^\perp \otimes \delta^\perp))) \& (\delta \otimes \bigotimes_{i=1}^n \beta_i^\perp))))
\end{align*}
\]

Merging of Alternative-group (R) and Alternative-group (P):
Merging of Alternative-group (P) and Alternative-group (P):
Appendix: Verification of Merging Rules

\[
\begin{align*}
\text{id} & \quad \bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j^\perp \otimes \delta^\perp)) \vdash \bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j^\perp \otimes \delta^\perp)) \\
W! & \quad \vdash \bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j^\perp \otimes \delta^\perp)) \\
L \otimes & \quad \vdash \bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j^\perp \otimes \delta^\perp)) \\
R \oplus & \quad \vdash \bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j^\perp \otimes \delta^\perp)) \\
\text{IF}2 & \quad \alpha \otimes (\alpha \rightarrow (((\bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j^\perp \otimes \delta^\perp)) \otimes (\bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j^\perp \otimes \delta^\perp)))))) \vdash \alpha \otimes (\alpha \rightarrow (((\bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j^\perp \otimes \delta^\perp)) \otimes (\bigoplus_{i=1}^{n} (\beta_i \otimes (\beta_j^\perp \otimes \delta^\perp)))))))
\end{align*}
\]
Merging of Or-group (R) and Or-group (R):

\[ id \vdash \gamma \]
\[ R \otimes \]
\[ \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \vdash \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ \gamma, \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ W! \]
\[ \vdash \gamma, \ldots, \gamma, \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ L \otimes \]
\[ \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ L \& \]
\[ \left( \bigotimes_{j \in s} (\beta_j \otimes (\bigotimes_{i=1}^{n} \beta_i \otimes \gamma)) \right) \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ \left( \bigotimes_{j \in s} (\beta_j \otimes (\bigotimes_{i=1}^{n} \beta_i \otimes \gamma)) \right) \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ \vdash \left( \bigotimes_{j \in s} (\beta_j \otimes (\bigotimes_{i=1}^{n} \beta_i \otimes \gamma)) \right) \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ R \& \]
\[ \left( \bigotimes_{j \in s} (\beta_j \otimes (\bigotimes_{i=1}^{n} \beta_i \otimes \gamma)) \right) \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ \vdash \left( \bigotimes_{j \in s} (\beta_j \otimes (\bigotimes_{i=1}^{n} \beta_i \otimes \gamma)) \right) \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ \vdash \left( \bigotimes_{j \in s} (\beta_j \otimes (\bigotimes_{i=1}^{n} \beta_i \otimes \gamma)) \right) \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ R \& \]
\[ \left( \bigotimes_{j \in s} (\beta_j \otimes (\bigotimes_{i=1}^{n} \beta_i \otimes \gamma)) \right) \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ \vdash \left( \bigotimes_{j \in s} (\beta_j \otimes (\bigotimes_{i=1}^{n} \beta_i \otimes \gamma)) \right) \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]

\[ \vdash \left( \bigotimes_{j \in s} (\beta_j \otimes (\bigotimes_{i=1}^{n} \beta_i \otimes \gamma)) \right) \vdash \gamma \otimes \left( \bigotimes_{i=1}^{n} \beta_i \right) \otimes \delta \]
Appendix: Verification of Merging Rules

Merging of Or-group (P) and Or-group (P):
Merging of Or-group (R) and Or-group (P):

\[
\begin{align*}
\text{id} & \quad (\bigotimes_{i=1}^{n} b_i) \otimes \gamma \vdash (\bigotimes_{i=1}^{n} b_i) \otimes \gamma \\
R \otimes & \quad (\bigotimes_{i=1}^{n} b_i) \otimes \gamma, (\bigotimes_{i=1}^{n} \beta_i) \otimes \delta \vdash ((\bigotimes_{i=1}^{n} b_i) \otimes \gamma) \otimes ((\bigotimes_{i=1}^{n} \beta_i) \otimes \delta) \\
L \oplus & \quad (\bigotimes_{i=1}^{n} b_i) \otimes \gamma, (\bigotimes_{i=1}^{n} \beta_i) \otimes \delta \vdash ((\bigotimes_{i=1}^{n} b_i) \otimes \gamma) \oplus (\bigotimes_{i=1}^{n} \beta_i) \otimes \delta \\
L \otimes & \quad (\bigotimes_{i=1}^{n} b_i) \otimes \gamma, (\bigotimes_{i=1}^{n} \beta_i) \otimes \delta \vdash (\bigotimes_{i=1}^{n} b_i) \otimes \gamma \oplus (\bigotimes_{i=1}^{n} \beta_i) \otimes \delta \\
L \& & \quad (\bigotimes_{i=1}^{n} b_i) \otimes \gamma, (\bigotimes_{i=1}^{n} \beta_i) \otimes \delta \vdash (\bigotimes_{i=1}^{n} b_i) \otimes \gamma \& (\bigotimes_{i=1}^{n} \beta_i) \otimes \delta \\
IF & \quad (\bigotimes_{i=1}^{n} b_i) \otimes \gamma, (\bigotimes_{i=1}^{n} \beta_i) \otimes \delta \vdash (\bigotimes_{i=1}^{n} b_i) \otimes \gamma \& (\bigotimes_{i=1}^{n} \beta_i) \otimes \delta
\end{align*}
\]
Merging of Alternative-groups can only be done if one group is the subset of the other group. Otherwise, these groups can not be merged due to the semantics of an Alternative-group, where exactly one feature must be chosen from this group. In the case of Or-groups we can merge any two (and indeed more Or-groups) with common parents.
References


References


of Seventeenth International Conference on Software Engineering and Knowledge Engineering (SEKE’05), pages 677–682, Taipei, Taiwan, Republic of China, 2005.


References


[EPAH09] Abdelrahman Osman Elfaki, S. Phon-Amnuaisuk, and Chin Kuan Ho. Using First Order Logic to Validate Feature Model. In Proceedings of


References

Proceedings of the AOM Workshop at MoDELS’05, page 2, Montego Bay, Jamaica, October 2005.


[SBRCT08] Sergio Segura, David Benavides, Antonio Ruiz-Cortés, and Pablo Trinidad. Generative and Transformational Techniques in Software


