Discovery and Validation for Composite Services on the Semantic Web

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Nalaka Dilshan Gooneratne B.App.Sc.(Hons.),
School of Computer Science and Information Technology,
Science, Engineering and Technology Portfolio,
RMIT University,
Melbourne, Victoria, Australia.

October 2008
Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Nalaka Dilshan Gooneratne
October 2008
Acknowledgments

I would like to thank my supervisors, Professor Zahir Tari and Associate Professor James Harland for their constant encouragement and guidance, and being excellent mentors. My sincere gratitude to Dr. Gregory Craske, who along with Professor Zahir Tari supervised the project for my undergraduate minor thesis. I would not have been able to start this degree without his support.

I am thankful to the Distributed Systems & Networking Discipline and the School of Computer Science & Information Technology at RMIT, and the Commonwealth Department of Education, Science and Training for their financial support throughout my candidature.

I would also like to thank the research program coordinators and administrators, Associate Professor Michael Winikoff, Associate Professor James Thom, Dr. Saied Tahaghoghi, Ms. Chin Scott, Mrs. Nyree Freeman and Ms. Beti Dimitrievska, for helping with all the administration issues related to my research program.

Sincere thanks go to my fellow Ph.D. students, Kazi Sakib, Abhinav Vora, James Broberg, Sarvanan Dayalan, Peter Dimopoulos, Anh Phan, Panu Phinjaroenphan, Vidura Abhaya, Sunidhi Bhalla, Islam Elgedawy, Sandy Citro, Anshuman Mukherjee, Fariborz Azimi, Malith Jayasinghe, Bin Rong, Mikhail Perepletchikov and Hendrik Gani for many interesting discussions, and creating a friendly and enjoyable work environment.

My gratitude to all my relatives, both here in Melbourne and in Sri Lanka for their continuous support. Finally, I would like to thank my wife, my parents, my brother and my grand father. It would have been impossible for me to pursue higher studies and take up a Ph.D. candidature, if not for their hard work, sacrifices and generosity.
To my parents
Credits

Portions of the material in this thesis have previously appeared in the following publications:


This work was supported by the Commonwealth Department of Education, Science and Training through an Australian Postgraduate Award, and the Distributed Systems & Networking Discipline and the School of Computer Science & Information Technology at RMIT University.

The thesis was written in TexMaker on Ubuntu, and typeset using the \LaTeX document preparation system.

All trademarks are the property of their respective owners.
## Contents

Abstract 1

1 Why Service Discovery and Validation are needed? 3
   1.1 Research Issues with Discovery and Validation 5
   1.2 Limitations of Existing Approaches 7
   1.3 Thesis Contributions 9

2 Composite Services on the Semantic Web 14
   2.1 Web Services 14
   2.2 Service Descriptions 16
      2.2.1 Description Logic (DL) 16
      2.2.2 $\pi$-calculus 17
      2.2.3 Control Flow Patterns 19
   2.3 The Semantic Web and Ontologies 23
      2.3.1 Meta-Ontology 24
      2.3.2 Context Matching Technique 28
   2.4 Summary 29

3 Creating Accurate Functional and Behavioural Descriptions 30
   3.1 Motivation 30
   3.2 Interval Time Logic (ITL) 35
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3 Proposed Formalisms</td>
<td>36</td>
</tr>
<tr>
<td>3.3.1 WS-ALUE</td>
<td>36</td>
</tr>
<tr>
<td>3.3.2 WS-π-calculus</td>
<td>40</td>
</tr>
<tr>
<td>3.4 Verification Technique</td>
<td>43</td>
</tr>
<tr>
<td>3.5 Case Study</td>
<td>51</td>
</tr>
<tr>
<td>3.6 Related Work</td>
<td>55</td>
</tr>
<tr>
<td>3.6.1 Functional Description Frameworks</td>
<td>55</td>
</tr>
<tr>
<td>3.6.2 Behavioural Description Frameworks</td>
<td>58</td>
</tr>
<tr>
<td>3.7 Discussion</td>
<td>60</td>
</tr>
<tr>
<td>3.8 Summary</td>
<td>62</td>
</tr>
<tr>
<td>4 S-Match: Matching Strictly Dependent Global Constraints</td>
<td>64</td>
</tr>
<tr>
<td>4.1 Motivation</td>
<td>64</td>
</tr>
<tr>
<td>4.2 Constraint Satisfaction</td>
<td>68</td>
</tr>
<tr>
<td>4.3 User Requests</td>
<td>71</td>
</tr>
<tr>
<td>4.4 S-Match</td>
<td>73</td>
</tr>
<tr>
<td>4.4.1 Composite Service Acquisition</td>
<td>75</td>
</tr>
<tr>
<td>4.5 Discussion</td>
<td>87</td>
</tr>
<tr>
<td>4.6 Theoretical Analysis</td>
<td>88</td>
</tr>
<tr>
<td>4.7 Evaluation</td>
<td>90</td>
</tr>
<tr>
<td>4.7.1 Experiments</td>
<td>90</td>
</tr>
<tr>
<td>4.7.2 Case Study</td>
<td>92</td>
</tr>
<tr>
<td>4.8 Related Work</td>
<td>101</td>
</tr>
<tr>
<td>4.9 Summary</td>
<td>105</td>
</tr>
<tr>
<td>5 I-MATCH: Matching Independent Global Constraints</td>
<td>107</td>
</tr>
<tr>
<td>5.1 Motivation</td>
<td>107</td>
</tr>
<tr>
<td>5.2 User Requests</td>
<td>109</td>
</tr>
<tr>
<td>5.3 I-Match</td>
<td>112</td>
</tr>
<tr>
<td>5.3.1 Service Indexing</td>
<td>113</td>
</tr>
<tr>
<td>5.3.2 Composite Service Acquisition</td>
<td>118</td>
</tr>
<tr>
<td>Functional Constraints</td>
<td>118</td>
</tr>
<tr>
<td>Relational Constraints</td>
<td>121</td>
</tr>
<tr>
<td>(A) Greedy Approach</td>
<td>122</td>
</tr>
</tbody>
</table>
## CONTENTS

(B) Derivation-based Approach ........................................... 127

5.4 Theoretical Analysis .................................................. 132

5.5 Evaluation .................................................................. 135
  5.5.1 Experiments ......................................................... 135
  5.5.2 Case Study .......................................................... 144

5.6 Summary .................................................................... 149

6 VGC: Generating Valid Global Communication Models 150

6.1 Motivation .................................................................. 150

6.2 Interval Time Logic-based Transitive Temporal Reasoning .... 155

6.3 Specification of Global Communication Models ................. 157

6.4 Generating Global Communication Models ....................... 158
  6.4.1 Generating Conversation Lists ................................. 160
  6.4.2 Complete Conversation Sets (CCSs) ......................... 164
  6.4.3 Detecting Deadlocks and Synchronisation Conflicts .... 166
  6.4.4 Deriving Specifications of Global Communication Models 169

6.5 Theoretical Analysis .................................................. 173

6.6 Related Work ........................................................... 178

6.7 Discussion .................................................................. 181

6.8 Summary .................................................................... 181

7 Conclusion 183

7.1 Overview of Solution .................................................. 184

7.2 Research Contributions ................................................ 185

7.3 Future Work ............................................................. 186
  7.3.1 Service Descriptions .............................................. 186
  7.3.2 Service Matching .................................................. 187
  7.3.3 Service Validation .................................................. 187

Bibliography 188
List of Figures

1.1 Service Oriented Architecture (SOA) ........................................ 4
1.2 Sample Composite Service .................................................. 5
1.3 Current Matching Techniques .............................................. 9

2.1 Types of Composite Services ............................................... 15
2.2 Language Constructs ...................................................... 18
2.3 Basic Control Flow Patterns ............................................. 19
2.4 Control Flows for Creating Complex Patterns ....................... 21
2.5 Workflow Types ............................................................ 22
2.6 Rules in a Meta-Ontology .................................................. 26
2.7 Substitution Graph ......................................................... 27
2.8 Transformation Graph ..................................................... 27

3.1 Sample Interaction Protocol ................................................ 33
3.2 Untraversable Interaction Protocols ..................................... 44
3.3 Execution Paths ............................................................ 45
3.4 Interaction Protocols in WS-$\pi$-calculus ............................... 46
3.5 Completion Index .......................................................... 48
3.6 Sample WS-$ALUE$ Description ......................................... 52
3.7 Interaction Protocol of a Computer Sales Service .................. 53
3.8 Sample WS-$\pi$-calculus Description ................................... 54

4.1 Constraint Types ............................................................ 69
4.2 Candidate Service .......................................................... 74
4.3 Restricted Attribute-based Service Indexing ......................... 78
4.4 Value-based Service Indexing ............................................ 79
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>Dimensions of an Attribute-Leveling Cube</td>
<td>80</td>
</tr>
<tr>
<td>4.6</td>
<td>Complexity - Available Services</td>
<td>85</td>
</tr>
<tr>
<td>4.7</td>
<td>Recall - Strictly-dependent Constraints - Varying no. of Available Services</td>
<td>93</td>
</tr>
<tr>
<td>4.8</td>
<td>Time Taken - Strictly-dependent Constraints - Varying no. of Available Services</td>
<td>94</td>
</tr>
<tr>
<td>4.9</td>
<td>Recall - Strictly-dependent Constraints - Varying no. of Service Types</td>
<td>95</td>
</tr>
<tr>
<td>4.10</td>
<td>Time Taken - Strictly-dependent Constraints - Varying no. of Service Types</td>
<td>96</td>
</tr>
<tr>
<td>4.11</td>
<td>Case Study - Available Services</td>
<td>97</td>
</tr>
<tr>
<td>4.12</td>
<td>Case Study - Two Dimensional View of the Attribute-Leveling Cube</td>
<td>98</td>
</tr>
<tr>
<td>4.13</td>
<td>Case Study - Leveled Attribute-Leveling Cube</td>
<td>99</td>
</tr>
<tr>
<td>4.14</td>
<td>Syntactic Approach - SHOP2 Method and Operator Instances</td>
<td>102</td>
</tr>
<tr>
<td>5.1</td>
<td>Sample Specification of a Functional Constraint</td>
<td>111</td>
</tr>
<tr>
<td>5.2</td>
<td>Sample Specification of a Relational Constraint</td>
<td>112</td>
</tr>
<tr>
<td>5.3</td>
<td>Indexed Services</td>
<td>114</td>
</tr>
<tr>
<td>5.4</td>
<td>Set of Slot Lists</td>
<td>116</td>
</tr>
<tr>
<td>5.5</td>
<td>Directed Acyclic Graph modeling a Relational Constraint</td>
<td>121</td>
</tr>
<tr>
<td>5.6</td>
<td>Sample Slot Lists</td>
<td>123</td>
</tr>
<tr>
<td>5.7</td>
<td>The Proposed Approach - Greedy Selection</td>
<td>123</td>
</tr>
<tr>
<td>5.8</td>
<td>Alternative Value Selections</td>
<td>126</td>
</tr>
<tr>
<td>5.9</td>
<td>Value Selection with Backtracking</td>
<td>126</td>
</tr>
<tr>
<td>5.10</td>
<td>Approximable Constraint</td>
<td>128</td>
</tr>
<tr>
<td>5.11</td>
<td>Sample Approximable Constraint</td>
<td>128</td>
</tr>
<tr>
<td>5.12</td>
<td>Recall - Functional Constraints - Varying no. of Available Services</td>
<td>138</td>
</tr>
<tr>
<td>5.13</td>
<td>Time Taken - Functional Constraints - Varying no. of Available Services</td>
<td>139</td>
</tr>
<tr>
<td>5.14</td>
<td>Recall - Functional Constraints - Varying no. of Service Types</td>
<td>140</td>
</tr>
<tr>
<td>5.15</td>
<td>Time Taken - Functional Constraints - Varying no. of Service Types</td>
<td>141</td>
</tr>
<tr>
<td>5.16</td>
<td>Recall - Relational Constraints - Varying no. of Available Services</td>
<td>142</td>
</tr>
<tr>
<td>5.17</td>
<td>Time Taken - Relational Constraints - Varying no. of Available Services</td>
<td>143</td>
</tr>
<tr>
<td>5.18</td>
<td>Recall - Relational Constraints - Varying no. of Service Types</td>
<td>144</td>
</tr>
<tr>
<td>5.19</td>
<td>Time Taken - Relational Constraints - Varying no. of Service Types</td>
<td>145</td>
</tr>
<tr>
<td>5.20</td>
<td>Available services</td>
<td>146</td>
</tr>
<tr>
<td>5.21</td>
<td>Slot Lists</td>
<td>147</td>
</tr>
<tr>
<td>6.1</td>
<td>Interaction Protocols</td>
<td>152</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

6.2 Global Communication Models ........................................ 154
6.3 Graphs modeling a Conversations List ................................. 164
6.4 Errors in Global Communication Models .............................. 167
6.5 Alternative Interaction Protocol of Insurance-I ...................... 168
6.6 Deriving a Specification of a Global Communication Model ....... 172
List of Tables

2.1 Constructs of $\mathcal{AL}$ ................................................. 17
2.2 Basic Control Flow Patterns ........................................... 20
2.3 Complex Patterns in Workflows ...................................... 22

3.1 Effects of Interactions .................................................. 34
3.2 Temporal Relationships between Concurrent Interactions ........ 36
3.3 WS-$\mathcal{ALUE}$ Constructs ............................................. 39
3.4 Evolving Execution State of a Sample Service ...................... 56
3.5 Current Functional Description Frameworks .......................... 57
3.6 Current (Specific) Behavioural Description Frameworks .......... 59

4.1 Solving an Incrementally Optimisable Non-binary Constraint ....... 71
4.2 Table of Notations ....................................................... 88
4.3 Ontological Relationships .............................................. 97
4.4 Case Study - Candidate Services ...................................... 98
4.5 Case Study - Conforming Composite Services ....................... 99
4.6 Naive Approach - Combinations of Attributes ................. 100
4.7 Naive Approach - Combinations of Values .......................... 101

5.1 Sample Services ......................................................... 114
5.2 Experiments and Results ............................................... 137
5.3 Ontological Relationships .............................................. 147
5.4 Candidate Services .................................................... 147

6.1 Transitivity Table for Temporal Reasoning .......................... 156
6.2 Grouping Relationships .................................................. 171
6.3 Descriptions of Communicative Actions ................................................. 175
6.4 Valid Global Communication Model ....................................................... 176
6.5 Invalid Global Communication Models .................................................... 177
Abstract

The availability of business functions on the Internet is increasing. These business functions referred to as web services are widely adopted because they increase software reusability and business agility. When a stand-alone service with the ability to service a complex request is not available, a composite service that is able to handle the request can be formed by composing multiple services. However, current technologies are not sufficient to form composite services that accurately provide a requested functionality for the following reasons.

- Current frameworks do not have the capacity to create complete service descriptions since they do not model all the functional aspects together (i.e. the purpose of a service, state transitions, data transformations). Those that deal with behavioural descriptions are not able to model the ordering constraints between concurrent interactions completely since they do not consider the time taken by interactions. Furthermore, there is no mechanism to assess the correctness of a functional description.

- Existing semantic-based matching techniques cannot locate services that conform to global constraints. Semantic-based techniques use ontological relationships to perform mappings between the terms in service descriptions and user requests. Therefore, unlike techniques that perform either direct string matching or schema matching, semantic-based approaches can match descriptions created with different terminologies and achieve a higher recall. Global constraints relate to restrictions on values of two or more attributes of multiple constituent services.

- Current techniques that generate and validate global communication models of composite services yield inaccurate results (i.e. detect phantom deadlocks or ignore actual deadlocks) since they either (i) do not support all types of interactions (i.e. only send and receive, not service and invoke) or (ii) do not consider the time taken by interactions.
This thesis presents novel ideas to deal with the stated limitations. First, we propose two formalisms (namely, WS-\textit{ALUE} and WS-\pi-calculus) for creating functional and behavioural descriptions respectively. WS-\textit{ALUE} extends the Description Logic language \textit{ALUE} with some new predicates and allows the creation of descriptions that model all the functional aspects together. These new predicates allow various attributes of descriptions to be represented in a differentiable manner. WS-\pi-calculus extends \pi-calculus with Interval Time Logic (ITL) axioms. The ITL axioms accurately model the temporal relationships between concurrent interactions. A technique that compares a WS-\pi-calculus description of a service against its WS-\textit{ALUE} description is introduced to detect any errors that are not equally reflected in both descriptions.

We propose novel semantic-based matching techniques to locate composite services that conform to global constraints. These constraints are of two types: \textit{strictly dependent} or \textit{independent}. A constraint is of the former type if the values that should be assigned to all the remaining restricted attributes can be uniquely determined once a value is assigned to one. A complete and correct technique that locates services that conform to strictly dependent constraints is provided. This technique uses a three-dimensional data structure called an Attribute Leveling Cube and locates conforming services in polynomial time. Any global constraint that is not strictly dependent is independent. The proposed approach that deals with independent constraints is correct, but not complete, and is a heuristic approach. This technique incorporates user defined objective functions, greedy algorithms and domain rules to locate conforming services, since solving independent constraints is NP-hard.

We propose a new approach to generate global communication models (of composite services) that are free of deadlocks and synchronisation conflicts. This technique requires WS-\pi-calculus descriptions of services. First, conversations are formed by composing interactions of constituent services. Then the temporal relationships between the conversations are found using an ITL-based transitive temporal reasoning mechanism. While doing so, deadlocks and synchronisation conflicts are detected by comparing the derived temporal relationships against those specified in WS-\pi-calculus descriptions. Finally, the temporal relationships between conversations are grouped and any duplicate information is eliminated, to derive a concise specification of a global communication model.
Chapter 1

Why Service Discovery and Validation are needed?

The web, which originally made information available through static documents, is evolving to a stage where it provides functionality which can be programmatically accessed through web services. These services rely on a Service Oriented Architecture (SOA) to facilitate their discovery and usage [Papazoglou, 2003; Sycara et al., 1999; 2002]. SOA consists of three main entities: providers, brokers and requesters. Service providers implement and host services, and publish their interfaces as descriptions in service registries. A service broker hosts such a registry and matches requests against the entries in its registry. Service requesters issue requests on behalf of users to locate descriptions in registries and then bind them to appropriate providers to invoke relevant services.

As the number of available brokers increases, they have to compete to acquire requests from requesters. In order to do so, brokers will need to improve the quality of registered services [Zeng et al., 2003; 2004] (e.g. response time, execution cost, reputation, reliability, availability) and provide value added functionalities (e.g. A service not only selling computers, but also organising shipping and insuring according to user constraints). A complex service (that provides a collection of related functionalities) can be easily made available by composing many existing services into one composite service. A broker can make such services available to requesters by incorporating an Aggregator to automatically compose services, thus, increasing the usage of services that are registered [Huhns, 2002; Papazoglou, 2003; Papazoglou and Georgakopoulos, 2003]. Figure 1.1 provides an overview of how various elements are integrated in a SOA.
CHAPTER 1. WHY SERVICE DISCOVERY AND VALIDATION ARE NEEDED?

There are two challenges that arise when implementing an aggregator: service discovery and validation. Service discovery is the process in which web services are located. A key issue that needs to be dealt when locating services is the service matching problem. Service matching is the process that compares service descriptions against user requests to select constituent services that are capable of providing the required functionalities [Elgedawy et al., 2004a; Benatallah et al., 2005; Paolucci et al., 2002]. Service validation is a process which checks if the selected services can provide the required functionality [Woodman et al., 2004]. This is performed by composing interaction protocols of constituent services to form global communication models, and then verifying that these models are well-formed (i.e. free of errors such as deadlocks, unspecified receptions and synchronisation conflicts).

The following is a scenario that further illustrates the importance of service discovery and validation.

**Scenario 1.1 (Composite Service)** A user wishes to purchase a computer online, and wants it insured and delivered, and sends a request to a service broker via a requester. The registry of the broker does not contain a single service selling computers, and organising shipping and insurance. However, this registry contains descriptions of services that perform these tasks separately. In such a situation, a broker can form a service as in Figure 1.2, if it contains an aggregator that implements a technique to locate composite services.

Let us assume that the located Shipping service requires an insurance policy number for a pickup date to be confirmed, and the Insurance service requires the pickup date to be confirmed...
before a policy number is issued. In such a situation, this composite service cannot provide the requested functionality since a deadlock occurs when the Shipping and Insurance services are composed. Therefore, the aggregator of the service broker needs to implement a validation technique to detect this deadlock.

1.1 Research Issues with Discovery and Validation

Discovery and validation of composite services is a relatively new area of research. Our focus in this thesis is on creating composite services that accurately provide the requested functionality. More specifically, the objective is to ensure that descriptions of services are equivalent reflections of their implementations, locate constituent services that conform to the specifications in a user request, and determine whether the composite service can successfully complete its execution. Hence, the following research questions are pursued.

A) How to create accurate service descriptions?

A description of a particular service is accurate if it correctly describes all the implemented features. That means, the accuracy of a service description is dependent on two factors: completeness and correctness. Completeness relates to the ability of a description to model all the required details (e.g., a description of a Computer Sales service that issues an invoice after each successful transaction is not complete if the fact that it issues an invoice is not mentioned). A description is not correct if it contains any errors (e.g., a service that rents computers being described as one that sells).
Service discovery and validation techniques rely on descriptions to obtain information regarding services. If the descriptions stored in a given registry are not accurate, aggregators may provide services that are not capable of providing requested functionalities. For example, the description of the renting service mentioned above may be located by a discovery technique as the *Computer Sales* service in Scenario 1.1. However, none of the composite services that are formed with this *Computer Sales* service would be able to satisfy the request, because that does not sell computers. Therefore, techniques to assess the completeness and correctness of service descriptions need to be developed.

**B) Which services conform to the specifications in a user request?**

The requirements specified in a user request can be categorised into two types: service or system. The former relates to the functionality or the behaviour of a requested service (e.g. the purchased computer should be a *Macintosh*, shipping details should be confirmed before insurance is organised) [Elgedawy et al., 2004a]. The latter describes the constraints that apply to execution environments of services (e.g. a selected service should have a minimum availability of 90% [Menascé, 2002; Zeng et al., 2004] or all communications should be encrypted using a X509 certificate [Tang et al., 2006]). Hence, discovery techniques that consider both service and system requirements need to be developed. Otherwise, services located by such techniques may not satisfy requests.

**C) Can a composite service successfully complete its execution?**

While a discovery technique locates composite services that conform to the specifications in a user request, this does not guarantee that they will successfully complete their execution in the intended manner. When multiple services are composed to form a composite service, a global communication model is generated by the interaction protocols of the constituent services. However, these models are susceptible to errors such as deadlocks, unspecified receptions and synchronisation conflicts [Woodman et al., 2004]. An example of such a case is provided in Scenario 1.1. The *Shipping* service first accepts an insurance policy number and then dispatches a pickup date, whereas the *Insurance* service first accepts a pickup date and then dispatches an insurance policy number. The global communication model of the composite service used to purchase a computer contains a deadlock because the ordering of interactions in the *Shipping* and *Insurance* services conflict. Therefore, techniques that are able to form error-free global communication models of composite services (when it is feasible...
to do so according to the interaction protocols of constituent services) need to be defined.

1.2 Limitations of Existing Approaches

This section briefly highlights the limitations of existing solutions that deal with the issues outlined in Section 1.1.

Creating Accurate Service Descriptions

Service descriptions used by discovery and validation techniques are of four types: functional, behavioural, technical and QoS (Quality of Service). A functional description specifies the functionality or the goal of a service [Elgedawy, 2003; de Bruijn et al., 2005]. This provides a description of what is performed. A behavioural description specifies the interaction protocol of a service [Banerji et al., 2002; Shen et al., 2005], and models how it provides the functionality that is specified in its functional description. A technical description provides the details required to bind to and execute a service (e.g. data formats, authentication and encryption mechanisms) [Curbera et al., 2002; Tang et al., 2006]. A QoS description provides details of the level of service that can be provided (e.g. execution cost, response time, reliability, availability, reputation) [Menascé, 2002; Zeng et al., 2004].

Correctness of a service description can be automatically evaluated by comparing it against a relevant source. The details in QoS descriptions are compared against details in execution logs [Menascé, 2002]. Technical descriptions are compared against implementations. Behavioural descriptions are compared against implementations using verification techniques [Foster et al., 2003]. However, to our knowledge, there is no technique that can be used to assert the correctness of functional descriptions.

Assessing the completeness of QoS and technical descriptions is a fairly subjective matter and is dependent on various issues such as the task for which they are intended (e.g. discovery, validation, execution, monitoring) and the domain in which they are used. The functionality of a service is modeled by describing its goal, and the state transition and data transformation caused by its execution. All three aspects have to be described in the functional description of a service. If not, a service that actually provides a particular functionality may be missed by a discovery technique. For example, assume that the functional description of a *Computer Sales* service only describes its purpose and the state transition. A user issues a request for a service that sells computers and provides a sales invoice. Even though the above *Computer
Sales service conforms to this request, it might not be located if the fact that it produces a sales invoice as an output is not specified in its functional description. Existing functional description frameworks do not represent all three functional aspects together [de Bruijn et al., 2005; Elgedawy, 2003; Martin et al., 2004; Sycara et al., 2002].

A behavioural description models an interaction protocol by specifying the interactions of a service and the ordering constraints between those interactions. Interactions of web services are of four types: send, receive, invoke and service. Send is an interaction in which a parameter is dispatched and receive is one in which a parameter is accepted. Invoke dispatches a parameter and then accepts another. Service on the other hand first dispatches a parameter and then accepts another. Of these types, send and receive take place at time points, whereas invoke and service take place during time intervals. Most existing frameworks support all four interaction types. However, none of them consider the time taken by interactions when specifying the ordering constraints between them. Hence, they are not able to represent the ordering constraints completely.

Service Matching

Matching techniques defined in existing service discovery approaches are of two types: syntactic and semantic. The former analyses a description (a user request or a service description) as a set of strings or parameter definitions and perform either syntax driven string matchings [Wu et al., 2003a; Zeng et al., 2004] or schema mappings [Medjahed et al., 2003; Medjahed and Bouguettaya, 2005]. They assume that descriptions are created using a single synonym-free vocabulary and eliminate the requirement for any mediation mechanism that derives alternative interpretations. The latter utilises ontological relationships to perform mappings between the terms in service descriptions and user requests [Akkiraju et al., 2006b; Elgedawy et al., 2004a]. Therefore, unlike the syntactic approaches, these techniques are able match descriptions created with different terminologies and achieve a higher recall [Elgedawy et al., 2004a].

Constraints are included in user requests to accurately specify a required service [Elgedawy et al., 2004a]. These constraints could be either local or global. Local constraints restrict the values of a particular attribute of a single service, whereas global constraints simultaneously restrict the values of two or more attributes of multiple constituent services. For example, type.Computer = MACINTOSH is a local constraint that can be applied to the composite service in Scenario 1.1, whereas location.Dispatch = location.Pickup ∈ validRegion.Insurance is a
CHAPTER 1. WHY SERVICE DISCOVERY AND VALIDATION ARE NEEDED?

global constraint. Current syntactic matching techniques consider both types of constraints. However, none of the existing semantic approaches can locate services that conform to global constraints. Only local constraints are considered by these approaches (see Figure 1.2).

![Figure 1.3: Current Matching Techniques](image)

Generating Well-formed Global Communication Models

Current validation techniques (for composite services) are of two types: latent and active. A latent approach first forms global communication models and then use model checkers to verify its well-formedness [Bultan et al., 2006; Kang et al., 2007]. An active approach maps behavioural descriptions of services to theoretical formalisms and uses reasoning mechanisms to check if an error-free global communication model can be formed [Woodman et al., 2004]. However, current techniques are not sufficient since they either (i) do not consider all the different types of temporal relationships that exist between interactions, or (ii) do not support all types of interactions (i.e. only send and receive, not service and invoke). Thus, they return false-negatives (e.g. detect phantom deadlocks) and false positives (e.g. ignore actual deadlocks).

1.3 Thesis Contributions

The research contributions we made in response to the issues mentioned in Section 1.1 are highlighted here.

Service Descriptions

1. We introduce two formalisms for creating functional and behavioural descriptions respectively: WS-\textit{ALUE} and WS-\textit{π}-calculus. The former extends the Description Logic...
(DL) language $\mathcal{ALUE}$ with some new axioms (e.g. functions that differentiate concepts, roles, inputs, outputs). WS-$\mathcal{ALUE}$ is able to specify all three functional aspects of a service together (i.e. goals, state transitions and data transformations). The latter extends $\pi$-calculus with (i) two new communicative actions to support service and invoke interactions (ii) Interval Time Logic axioms to represent all the different types of temporal relationships between interactions and (iii) DL constructors to specify the effects of interactions. Functional descriptions can be compared against behavioural descriptions when the effects of interactions are included.

2. A new technique that checks the correctness of functional descriptions is proposed. This technique checks if a valid terminal state of a service can be derived from its initial execution state according to its behavioural description. This means the devised approach checks if the aggregated effect that is caused by performing interactions according to a given protocol is equivalent to the state transition in a functional description. The technique traverses through the interaction protocol of a service using a pre-order depth-first search algorithm to derive the aggregated effect of its interactions.

Service Discovery

We also introduce novel semantic matching techniques that locate composite services that conform to global constraints. Such techniques utilise substitution graphs and transformation graphs of meta-ontologies, and the context matching technique in [Elgedawy et al., 2004a] to match descriptions created with syntactically heterogeneous terminologies. Global constraints are classified as either strictly dependent or independent based on the complexity of solving them (i.e. determining the values that should be assigned to the attributes). A constraint is strictly dependent if the values that should be assigned to all the remaining restricted attributes can be uniquely determined once a value is assigned to one. $\text{location.Dispatch} = \text{location.Pickup} \in \text{validRegion.Insurance}$ is an example of a strictly dependent global constraint. The values that should be assigned to $\text{location.Pickup}$ and $\text{validRegion.Insurance}$ can be uniquely determined once a value is assigned to $\text{location.Dispatch}$. Such constraints can be solved in polynomial time. Any global constraint that is not strictly dependent is independent (e.g. $\text{productionTime.Computer} + \text{approvalPhase.Insurance} + \text{timeTaken.Delivery} < 7_{\text{DAYS}}$ and $\text{availableDate.Computer} \leq \text{approvalDate.Insurance} \leq \text{date.Pickup}$). Separate matching approaches are defined to deal with the two types of con-
CHAPTER 1. WHY SERVICE DISCOVERY AND VALIDATION ARE NEEDED?

1. S-Match: It locates services that conform to strictly dependent constraints. This approach is correct and complete (i.e. retrieves a particular composite service if and only if it conforms to the given constraints). Leveling is performed to ensure that values assigned to multiple attributes are comparable. A three-dimensional data structure called an Attribute Leveling Cube is used since a naive approach takes exponential time to (i) level attributes and (ii) find conforming values.

2. I-Match: It locates services that conform to independent constraints. This approach is correct, but it is not complete. That means, it may not locate all the services that conform to a given constraint. Services are indexed in a two dimensional data structure called a Slot List to efficiently locate those that assign a certain value to a particular attribute. Optimisation techniques are used to find values that conform to a given constraint in polynomial time (because an exhaustive approach takes exponential-time). The optimisation technique used, is dependent on the structure of the considered constraint. Pre-defined (user defined) objective functions [Nareyek, 2001; Voudouris and Tsang, 2001] are used to identify values that conform to non-binary locally optimisable constraints (e.g. productionTime.Computer + approvalPhase.Insurance + timeTaken.Delivery < 7 DAYS). If a constraint is binary (e.g. availableDate.Computer ≤ approvalDate.Insurance ≤ date.Pickup), conforming values are either (i) found using a greedy approach or (ii) derived using domain rules.

Service Validation

We propose a new approach (called VGC) to generate well-formed global communication models of composite services. This technique constructs a specification of a communication model if and only if one can be formed according to the interaction protocols of constituent services. First, conversations are formed by composing the interactions of constituent services. Interactions are composed based on a composability model. This model matches the communication channels and the parameters used to perform interactions, and checks if they are of compatible types. Then, sets of conversations that consist of all the interactions of constituent services are formed to ensure that a derived global communication model is free of unspecified receptions or transmissions [Peng and Purushothaman, 1989]. Such sets of conversations are referred to as Complete Conversation Sets (CCS). Next, the
CHAPTER 1. WHY SERVICE DISCOVERY AND VALIDATION ARE NEEDED?

temporal relationships between the interactions of a CCS are found using an Interval Time Logic (ITL)-based transitive temporal reasoning mechanism [Allen, 1983]. While doing so, deadlocks and synchronisation conflicts are detected by identifying inconsistencies in the temporal relationships. An inconsistency occurs if the temporal relationships between any two interactions (of a global communication model) conflict with those of an interaction protocol (of a constituent service). Finally, a concise specification of a global communication model is obtained by grouping the temporal relationships between interactions and eliminating any duplicate information.

1.4 Thesis Structure

The rest of this thesis is structured as follows.

Chapter 2 (Composite Services for the Semantic Web) Background of some concepts used in this thesis are provided in this chapter. This includes an analysis of Description Logic [Baader et al., 2003], $\pi$-calculus [Sangiorgi and Walker, 2001], the Meta-Ontology [Elgedawy, 2003], control flow patterns in interaction protocols [van der Aalst et al., 2003a] and the context matching technique [Elgedawy et al., 2004a].

Chapter 3 (Creating Accurate Functional and Behavioural Descriptions) This chapter introduces two formalisms: WS-$ALUE$ and WS-$\pi$-calculus for creating functional and behavioural descriptions of services. Also, we propose a technique that verifies the correctness of WS-$ALUE$ descriptions and WS-$\pi$-calculus descriptions. A case study in which we detect errors in service descriptions by verifying them using the devised technique, is provided.

Chapter 4 (S-Match) This chapter introduces a semantic matching technique for locating composite services. Only strictly dependent constraints are considered by this approach. Details of a case study and simulation experiments that compare the proposed approach against some existing techniques are provided.

Chapter 5 (I-Match) Semantic matching techniques that locate services that conform to non-binary locally optimisable independent and binary independent constraints are proposed here. The devised approaches are evaluated using a case study and simulation experiments.
Chapter 6 (VGC) This chapter presents a module that generates well-formed global communication models of composite services. The proposed approach consists of (i) a composability model to form conversations from interactions of constituent services, (ii) a technique that extends a transitive reasoning mechanism to detect deadlocks and synchronisation conflicts in global communication models, and (iii) a process that derives concise specifications by grouping the temporal relationships between interactions (of a global communication model) and eliminating any duplicate information. This chapter also provides details of a case study performed to evaluate the devised approach.

Chapter 7 (Conclusion) This chapter concludes the thesis by indicating how the proposed solutions can be utilised in real-world scenarios and highlighting some areas of future work.
Chapter 2

Composite Services on the Semantic Web

Background material required to understand the work in this thesis is presented in this chapter. This includes a brief description of web services, some concepts related to service description languages and ontologies.

2.1 Web Services

The World Wide Web Consortium (W3C) describes a web service as: (i) “A software system designed to support inter-operable Machine to Machine interaction over a network.” [Haas and Brown, 2004] and (ii) “A software application identified by a URI (Uniform Resource Identifier), whose interfaces and bindings are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols.” [Schlimmer, 2002]. Based on these two complementary descriptions, we define web services as applications that are (i) programatically accessible (i.e. invoked by other web services and applications), (ii) loosely coupled (i.e. use document-based communication. Commonly used object-based communication mechanisms tend to form highly coupled systems [Vogels, 2003]), and (iii) accessed and invoked using standard protocols over the Internet (e.g. HTTP, SOAP, XML)

Web services are of two types: simple and composite. A simple service is one atomic service, whereas a composite service is an aggregation of multiple services. Composite services are executed in a collaborative manner to provide “value added services”. These services can
be categorised into three types as either vertical, horizontal or hybrid, based on the visibility of constituent services. A composite service is vertical if a requester is only allowed to see one constituent service (see Figure 2.1a). The visible service is both a provider and a consumer. This service delegates tasks to other constituent services, coordinates their execution and returns an aggregated result to a requester\(^1\) (e.g. in the composite service shown in Figure 2.1a, only the Insurance service is visible to the user. He/she would not be aware that the Insurance Bureau service is invoked by the Insurance service to check a client’s claims records). A horizontal composite service is created when all the constituent services are visible to a user (see Figure 2.1b). The hybrid form is created by combining both horizontal and vertical composite services (see Figure 2.1c).

---

\(^1\)Note that a scenario in which a service is invoked through a broker does not form a vertical composite service. The visible service of a vertical composite service is not a broker.
The work described in this thesis assumes that all constituent services are visible. It is difficult to assess the ability of a particular composite service to accurately provide a requested functionality, if its constituent services are not visible. Hence, only horizontal composite services are considered in this thesis.

2.2 Service Descriptions

The proposed formalisms for creating functional and behavioural descriptions, WS-ALUE and WS-π-calculus extend the Description Logic language ALUE and π-calculus. This section briefly describes DL and π-calculus. The following also provides an analysis of control flow patterns. An understanding of these patterns is required to identify bounded and structural conflict-free (i.e. deadlock and synchronisation conflict free) interaction protocols and global communication models.

2.2.1 Description Logic (DL)

Description Logic [Baader et al., 2003] is a set of knowledge represent languages. They have been widely used in various application domains for the following reasons.

1. The ability to represent knowledge in a structured and formally understood manner.

2. The ability to define several reasoning mechanisms to use represented knowledge (i.e. check consistency, satisfiability and entailment).

DL-based languages consist of concepts, properties, individuals and complex statements. A concept denotes a set of individuals (e.g. Computer, Laptop). A property describes a relationship between multiple concepts (e.g. is a, consists of). An individual is an element of a concept (e.g. DELL, MACINTOSH). A complex statement defines a concept by combining other concepts and properties (e.g. Laptop ≡ Computer ⊓ Battery).

Knowledge is represented in DL languages using terminologies and assertions. An assertion is a specification that explicitly relates an individual to a particular concept (e.g. Laptop(DELL) denotes that DELL is an instance of Laptop). A terminology is a set of complex statements which define the concepts of a domain. DL languages are differentiated based on the constructs that are used to create complex statements. These constructs determine the expressiveness of a particular DL language. \(\mathcal{AL}\) (Attributive Language) is the minimal
CHAPTER 2. COMPOSITE SERVICES ON THE SEMANTIC WEB

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept</td>
<td>$C$</td>
</tr>
<tr>
<td>Property</td>
<td>$P$</td>
</tr>
<tr>
<td>Top</td>
<td>$\top$</td>
</tr>
<tr>
<td>Bottom</td>
<td>$\bot$</td>
</tr>
<tr>
<td>Negation</td>
<td>$\neg C$</td>
</tr>
<tr>
<td>Conjunction</td>
<td>$C \sqcap C'$</td>
</tr>
<tr>
<td>Value restriction</td>
<td>$\forall P.C$</td>
</tr>
</tbody>
</table>

Table 2.1: Constructs of $\mathcal{AL}$

DL language (of practical use) [Baader et al., 2003]. The constructs of $\mathcal{AL}$ are described in Table 2.1.

Basic extensions made to $\mathcal{AL}$ include union ($\sqcup$), full existential quantification ($\exists P.C$), number restrictions ($\geq n \, R$, $\leq n \, R$) and complex negations ($\neg(C \sqcap D)$). WS-$\mathcal{ALUE}$, the proposed formalism for creating functional descriptions is based on the extension of $\mathcal{AL}$ that supports union and full existential quantification ($\mathcal{ALUE}$). A DL language is selected because of the ability to create well structured descriptions (i.e. easily differentiable elements). $\mathcal{ALUE}$ is the minimal DL language that defines most of the constructs that are required to represent the three functional aspects of a service: purpose, data transformations and state transitions. They are described with complex statements, value restriction constructs and existential quantification constructs respectively. Further details on how $\mathcal{ALUE}$ is used to model functionalities of services will be described in the next chapter.

2.2.2 $\pi$-calculus

A specification of an interaction protocol describes the interactions of a service and the ordering constraints between them. Many formalisms that model interaction protocols have been proposed (e.g. $\pi$-calculus [Sangiorgi and Walker, 2001; Woodman et al., 2004], Mealy Finite State Machines (MFSMs) [Berardi et al., 2003; Bultan et al., 2006; Deutsch et al., 2007] and Petri-Nets [van der Aalst and Hofstede, 2000; Murata, 1989; Ratzer et al., 2003]). These formalisms model interactions as either events or primitive actions. Unlike MFSMs and

\footnote{Negation of complex statements.}
Petri-Nets, π-calculus is able to model the entities involved in a particular interaction with its representation of channels. Modeling such entities in descriptions of interactions is important when there are 3 or more communicating entities, especially when a service performs multiple interactions that either send or receive a common parameter to/from multiple services. For example, if the Computer Sales service in Scenario 1.1 sends a specification of a computer to the Insurance and Shipping services using two separate interactions and the entities involved in these interactions are not specified, then it would be difficult to differentiate between the two interactions. For this reason, π-calculus is selected as the basis to develop the proposed formalism for creating behavioural descriptions (WS-π-calculus).

An interaction \( a \) in π-calculus is represented as follows.

\[
a = \overline{chp} \mid ch(p) \mid \tau \mid [x = y]a
\]

where \( \overline{chp} \) represents an interaction in which \( p \) is sent through a channel \( ch \) and \( ch(p) \) represents an interaction in which \( p \) is received through \( ch \). Interactions of web services include those of types service and invoke, which use two parameters (i.e. accept one and dispatch another or vise versa). In the monadic form of π-calculus, \( p \) can only be used to represent one token, whereas in the polyadic form \( p \) can be more than one token. That means in the monadic form only one parameter can be sent or received in an interaction, whereas in the polyadic form more than one parameter can be either sent or received. \( \tau \) represents an unobservable action, where a process evolves from a state \( S \) to a state \( S' \) without performing an interaction. \([x=y]a\) represents an interaction in which \( a \) is performed if the condition \([x=y]\) is true.

In π-calculus, ordering constraints are specified using four basic language constructs: sequence, choice, parallel and iteration. These are graphically depicted in Figure 2.2.

![Figure 2.2: Language Constructs](image)

π-calculus models an interaction protocol as a process \( P \) and uses the following syntax to
generate descriptions.

\[ P = a.P \mid P|P \mid P + P \mid <n>!P \mid 0 \]

\( a.P \) represents a sequence construct where interaction \( a \) is followed by process \( P \). \( P+P \) represents a choice construct where either process has to be performed. \( P|P \) represents a parallel construct where both processes have to be performed concurrently. \( <n>!P \) represents an iteration construct where \( P \) has to be repeated \( n \) number of times. \( <n> \) is optional and can be omitted if \( P \) is performed an infinite number of times. \( \theta \) is used to specify explicit termination. For example, \( ch(p).\theta \) means receive \( p \) along \( ch \) and stop. \( ch(p).\theta \) is often abbreviated to \( ch(p) \).

### 2.2.3 Control Flow Patterns

A composite service that provides a requested functionality cannot be formed with constituent services that do not complete their execution [van der Aalst et al., 2003a]. Services with interaction protocols that are unbounded or contain structural conflicts (such as deadlocks and synchronisation conflicts) cannot terminate. Hence, it is important to identify those ordering constraints that do not cause such errors.

Workflows model ordering constraints using control flow patterns [van der Aalst et al., 2003b; Kiepuszewski et al., 2000]. A vast amount of research has been done in the area of workflows to identify control flows that are bounded and free of structural conflicts [van der Aalst et al., 2003b; Kiepuszewski et al., 2000]. The following provides a brief description of control flow patterns, and indicates those that are bounded and free of structural conflicts.

Control flow patterns [van der Aalst et al., 2003b] can be either basic or complex. The basic control flow patterns and the situations in which they occur are described in Table 2.2. The notations used to graphically represent these patterns are shown in Figure 2.3.

![Basic Control Flow Patterns](image-url)

*Figure 2.3: Basic Control Flow Patterns*
### Table 2.2: Basic Control Flow Patterns

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequence</td>
<td>When an action has one in-flow and one out-flow</td>
</tr>
<tr>
<td>and-split</td>
<td>When an action has multiple out-flows that are executed simultaneously.</td>
</tr>
<tr>
<td>or-split</td>
<td>When an action has multiple out-flows and only one is executed.</td>
</tr>
<tr>
<td>and-join</td>
<td>When multiple parallel in-flows merge together as one flow in a synchronized manner.</td>
</tr>
<tr>
<td>or-join</td>
<td>When multiple in-flows merge together as one flow in an unsynchronized manner.</td>
</tr>
</tbody>
</table>

Complex patterns are created by combining one or more simple patterns [van der Aalst et al., 2003a; Kiepuszewski et al., 2000]. The control flows that form these complex patterns are graphically depicted in Figure 2.4. Four types of complex patterns can be defined based on the ways in which these control flows occur in a workflow.

**Structured Cycle**: A loop with one entry point and one exit point, where the entry is made with an or-join and the exit is made with an or-split.

**Arbitrary Cycle**: A loop with no restriction on the number of entry points and exit points, where the entry is made with an or-join or an and-join and the exit is made with an or-split or an and-split (see Figures 2.4a-f).

**Normal Split and Join**: A control flow with a split and a join which is non-conflicting and corresponding.

**Definition 2.1 (Corresponding Splits and Joins)** [Kiepuszewski et al., 2000] A split $S$ has a corresponding join $J$ if the out-flows $f_1, \ldots, f_n$ of $S$, eventually become the only in-flows of $J$ (see Figure 2.4g).

**Definition 2.2 (Conflicting Splits and Joins)** [Kiepuszewski et al., 2000] Given a split $S$ and a join $J$, where the out-flows $f_1, \ldots, f_n$ of $S$ are joined by $J$, $S$ and $J$ conflict if $S$ is an or-split and $J$ is an and-join or if $S$ is an and-split and $J$ is an or-join (see Figure 2.4h).

**Arbitrary Split and Join**: A control flow with a split and a join, where no restrictions apply on the split and join.
CHAPTER 2. COMPOSITE SERVICES ON THE SEMANTIC WEB

Figure 2.4: Control Flows for Creating Complex Patterns
We categorize workflows into three types based on these complex patterns as either Arbitrary Workflows, Restricted Loop Workflows or Normal Workflows. The complex patterns that are supported by each workflow type are described in Table 2.3. These three workflow types vary in expressiveness since they apply different restrictions on the type of complex patterns that are supported. The relationship between the three workflow types based on expressiveness, can be graphically depicted as in Figure 2.5.

<table>
<thead>
<tr>
<th></th>
<th>Arbitrary Cycles</th>
<th>Structured Cycles</th>
<th>Arbitrary Split and Joins</th>
<th>Normal Split and Joins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Restricted Loop</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Normal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 2.3: Complex Patterns in Workflows*

Figure 2.5: Workflow Types

Arbitrary workflows are the most expressive since they support all types of complex patterns, and normal workflows are the least expressive since they only support normal split and joins. However, there is a trade-off between the expressiveness and certain properties that can be guaranteed by a workflow.

From the three types of workflows, only the normal workflows can guarantee freedom from deadlocks and synchronization conflicts, and boundedness [Kiepuszewski et al., 2000]. Such properties cannot be guaranteed for the arbitrary workflows and restricted loop workflows, unless they are constrained in such a way that they have an equivalent normal workflow.

Arbitrary workflows and restricted loop workflows can be converted to equivalent normal workflows if they do not contain any and-splits, and-joins or infinite loops. And-splits and and-joins introduce synchronization conflicts and deadlocks to workflows. A synchronization
conflict occurs if the out-flows of an and-split are joined by an or-join, and a deadlock occurs if the out-flows of an or-split are joined by an and-join [Kiepuszewski et al., 2000].

**Proposition 2.1** A restricted loop workflow that does not contain and-splits, and-joins or infinite loops can be converted to an equivalent normal workflow [van der Aalst et al., 2003b].

Proof

1. A structured cycle can be converted to an equivalent acyclic sequence by duplicating the actions [Oulsnam, 1982].

2. An arbitrary split and join with out parallelism can be converted an equivalent normal split and join [Kiepuszewski et al., 2000]. ♦

**Proposition 2.2** An arbitrary workflow that does not contain and-splits, and-joins or infinite loops can be converted to an equivalent normal workflow [van der Aalst et al., 2003b].

Proof

1. An arbitrary loop can be converted to a structured loop by duplicating the actions and using auxiliary variables. Hence, an arbitrary workflow can be converted to equivalent restricted loop workflow [Oulsnam, 1982].

2. A restricted loop workflow which does not contain and-split, and-joins or infinite loops can be converted to an equivalent normal workflow according to Proposition 2.1. ♦

### 2.3 The Semantic Web and Ontologies

The World Wide Web was primarily intended for human interpretation and use. The Semantic Web is an extension of the Web where content is annotated with semantic information to facilitate automated interpretation [McIlraith et al., 2001; McIlraith and Martin, 2003]. These annotations are linked to Ontological descriptions that define taxonomic and logical relationships. This enables automated computer programs to identify and classify the relationships between described web entities.

The composite matching techniques proposed in this thesis are extensions of the Context Matching Technique in [Elgedawy et al., 2004a]. These techniques are semantic-based approaches (i.e. they use ontological relationships to match terms in service descriptions to
CHAPTER 2. COMPOSITE SERVICES ON THE SEMANTIC WEB

user requests.) and assumes that ontological descriptions are structured according to the Meta-Ontology [Elgedawy, 2003; Elgedawy et al., 2004a; 2008]. This section provides a brief description of the Meta-Ontology and the Context Matching Technique.

2.3.1 Meta-Ontology

An ontology is an abstraction of a domain of interest into elements and relationships. They provide definitive interpretations of elements in a particular domain [Staab and Studer, 2004]. Therefore, linking the terms used in service descriptions to ontological descriptions offers two benefits.

1. Service descriptions can be specified in an unambiguous manner. For example, the term Memory can be interpreted as the function of the brain that provides the ability to retain information, or an electrical storage device. An erroneous interpretation of a service description that contains the term Memory can be prevented if it is linked to an appropriate ontological description.

2. Facilitates the usage of semantic-based matching techniques. For example, an ontological relationships that exists between the terms Laptop and Computer can be used to match one to the other.

Current ontologies can be categorised as either graph-based [Doan et al., 2003; Elgedawy et al., 2004a; Sycara et al., 1999; 2002; Yeh et al., 2003] or Description Logic (DL)-based [Horrocks et al., 2003] approaches.

OWL [Arroyo et al., 2004; Bechhofer et al., 2004; Horrocks et al., 2003; Staab and Studer, 2004] is one of the most prominent and recently developed DL-based ontology specification languages. Like most of its predecessors such as DAML, DAML-ONT, SHOE, OIL and DAML-OIL, OWL is based on Description Logic [Baader et al., 2003]. It has been widely utilised in semantic web based architectures and techniques due to the following reasons [Arroyo et al., 2004; Bechhofer et al., 2004; Horrocks et al., 2003; Staab and Studer, 2004]:

- It is backward compatible and includes all the features of ontology specification languages such as RDF-S, DAML, SHOE, DAML-ONT, OIL and DAML-OIL
- It has a clearly defined syntax and semantics for representing knowledge
- It can be used to reason about properties such as membership, equivalence, consistency and classification
• It is developed and backed by the W3C\(^3\)

However, OWL ontologies cannot be used for semantic-based matching because they are either too complex or not expressive enough to accurately model a domain. The full functionality of OWL is in the OWL-Full ontology. However, determining if a service description is consistent with the descriptions (inferring entailment) of an OWL-Full ontology is undecidable [Baader et al., 2003; Staab and Studer, 2004]. A restricted version, OWL-DL, is a decidable subsystem. However, OWL-DL ontologies require explicit typing of elements. This restriction prevents OWL-DL ontologies from accurately modeling a domain [Baader et al., 2003; Staab and Studer, 2004]. For example, there is a domain where a Computer contains Processing\_Components such as CPU, SoundCard, VideoCard, etc., and Intel and Dell are individuals of the classes CPU and Computer. OWL-DL cannot describe the relationship between Dell and Intel because this requires CPU to be both a class and an individual\(^4\). In addition, the use of DL-based ontologies and generic graph-based ontologies can lead to inaccurate matching results as they cannot model the contextual aspects of relationships [Elgedawy et al., 2004a]. For example, matching techniques that use these ontologies would check the matching of Computer with Laptop, by checking either the taxonomic distance [Mokhtar et al., 2006; Sycara et al., 1999: 2002] or existence of subsumption relations [Benatallah et al., 2005; Doan et al., 2003; Li and Horrocks, 2004; Paolucci et al., 2003; Rodriguez and Egenhofer, 2003]. However, this is not correct since a computer cannot be used instead of a laptop in every scenario. It can only be used when the laptop is kept in one fixed location (i.e. when direct current is used instead of a battery).

The matching techniques proposed in this thesis use the meta-ontology [Elgedawy, 2003; Elgedawy et al., 2004a; 2008]. To the best of our knowledge it is the only approach that consists of constructs that accurately model a domain and allow ontological relationships to be reasoned with a “simple mechanism”.

A Meta-Ontology consists of four types of elements: concepts, operation, roles and rules. Concepts are basic entities of a domain. Each concept is described with a set of attributes. Domain elements with common attributes are abstracted into concepts (e.g. Computer, Finance, Insurance). Transactions that are permitted in a domain are specified with operations (e.g. Sales(), Insure(), Ship()). Roles are used to specify the actors such as Sales\_Assistant,
**Telemarketer, Auctioneer.** Rules define the legitimate derivations of a domain (e.g. a customer is categorised as VIP if more than 100 purchases are made within a week, insurance should be approved on the next day according to standard procedures, a computer should be picked up a day after it is made available according to standard procedures). However, an exact description of how these rules are represented is not provided in [Elgedawy, 2003; Elgedawy et al., 2004a; 2008]. We assume that a rule is specified as a function, which derives a value that should be assigned to a “derived” attribute based on a value that is assigned to a “determinant” attribute. For example, the value **VIP** is assigned to the derived attribute **type.Customer** if a value greater than **100** is assigned to the determinant attribute **noOfPurchases.Customer**. The context in which these assignments should be made to the attributes is specified with an operation, an affected concept and a role. Therefore, the three sample rules described above are modeled in a meta-ontology as in Figure 2.6.

![Rules in a Meta-Ontology](image)

**Figure 2.6: Rules in a Meta-Ontology**

The elements in WS-ALUE descriptions refer to operations, concepts, roles and attributes to ensure that unambiguous interpretations are derived by reasoning mechanisms (in matching techniques). Rules are used by one of the proposed discovery techniques that deals with global constraints to locate conforming values in polynomial time⁵.

---

⁵Solving global constraints is NP-hard [Dechter, 1992; Freuder, 1982; 1985]. Further details regarding these constraints and how rules are used, are provided Chapter 5.
The relationships between the elements of an ontology are specified with substitution graphs and transformation graphs. In a substitution graph, the contextual aspect of a relationship is specified with a substitution condition. Each (substitution) graph defines a set of attribute mappings. Figure 2.7 depicts an example of a substitution graph which specifies the relationship between the terms Computer and Laptop.

Figure 2.7: Substitution Graph

Figure 2.8: Transformation Graph

27 (October 2008)
A transformation graph models a relationship that requires the meta-model of an element to be changed. It is specified with one or more consecutive operations. The above example, which attempts to state the relationship between Dell and Intel can be modeled with a transformation graph as in Figure 2.8. It is specified with the operations Component:listComponents(Computer) and ComponentType:typesOfComponents(Computer).

These two graph-based data structures are specified for the concepts, operations, roles and rules of an ontology. The meta-ontology requires any transitive relationship to be explicitly specified with a substitution graph or a transformation graph. For example, if the concept Calculator can be substituted with Laptop, and Laptop can be substituted with Computer, there is a transitive substitution relationship between Calculator and Computer. Then, a substitution graph which allows the concept Calculator to be substituted with Computer is explicitly included in the ontological descriptions. This eliminates the requirement for a complicated reasoning mechanism.

The matching techniques proposed in this thesis use substitution and transformation graphs to match service descriptions to user requests. An existing technique that uses these graphs to match constraints is briefly described next.

2.3.2 Context Matching Technique

The technique proposed in [Elgedawy et al., 2004a] matches the contexts of two similar goals. The matching techniques proposed in this thesis use Elgedawy et al.’s notion of “semantically related attributes” to identify the list of attributes related to a given attribute according to ontological relationships. The following provides a definition of a semantically related attribute and shows how it is used to match constraints.

Definition 2.3 (Semantically Related Attributes)  [Elgedawy et al., 2004a] Given two attributes $a_i$ and $a_y$ where their scopes are defined with the concepts $c_i$ and $c_y$ respectively, $a_i$ is semantically related to $a_y$ if $c_y$ can be substituted with $c_i$ using either a substitution graph or a transformation graph, and the substitution results in $a_i$ being mapped to $a_y$.

For example, the attribute Computer.display can be semantically related to Laptop.size according to the substitution graph given in Figure 2.7 because it enables Laptop to be substituted with Computer and the substitution results in display being mapped to size.

In the context matching technique of [Elgedawy et al., 2004a], a condition $cnd_i$ is matched to another condition $cnd_j$ if $cnd_j$ can be substituted with $cnd_i$. Let $cnd_i$ and $cnd_j$ be two
conditions, where $cnd_i$ and $cnd_j$ are specified with the concepts $c_i$ and $c_j$, attributes $a_i$ and $a_j$, comparative operators $o_i$ and $o_j$, and values $v_i$ and $v_j$. Condition $cnd_j$ can be substituted with $cnd_i$ if

1. $a_i$ can be semantically related to $a_j$, and

2. the condition $d_i \cap d_j \neq \emptyset$ is true, where $d_i$ is the domain defined by $o_i$ and $x_i$, and $d_j$ is the domain defined by $o_j$ and $x_j$.

For example, the condition $Laptop.size < 17 \text{ INCHES}$ can be substituted with $Computer.display > 15 \text{ INCHES}$ because (i) $Computer.display$ can be semantically related to $Laptop.size$ with the substitution graph in Figure 2.7, and (ii) $p \cap q \neq \emptyset$, where $p$ is the set that consists of every value less than 17 inches and $q$ is the set that consists of every value that is greater than 15 inches.

2.4 Summary

This chapter provides descriptions on Description Logic [Baader et al., 2003], $\pi$-calculus [San-giorgi and Walker, 2001], control-flow patterns [van der Aalst et al., 2003a;b; Kiepuszewski et al., 2000], Meta-Ontology [Elgedawy, 2003; Elgedawy et al., 2004a; 2008] and the Context Matching Technique [Elgedawy et al., 2004a]. The proposed formalisms for creating service descriptions are extensions of Description Logic and $\pi$-calculus. The devised composite matching techniques incorporate the concept of semantically related attributes extracted from the Context Matching Technique, and assume that terms used in descriptions (i.e. service descriptions and user requests) are defined in Meta-Ontologies. The proposed technique that verifies the correctness of service descriptions and the one that checks whether composite services are free of deadlocks and synchronisation conflicts described in Chapters 3 and 6 respectively, assumes that the reader has an understanding of control flow patterns.
Chapter 3

Creating Accurate Functional and Behavioural Descriptions

Discovery and validation techniques rely on descriptions for information on the functionality of a service and the way in which it is provided. Therefore, the correctness of these techniques are dependent on the accuracy of such descriptions. This chapter introduces two novel formalisms, namely WS-ALUE and WS-π-calculus [Gooneratne et al., 2006; 2007a], for creating functional and behavioural descriptions of services, and a technique that compares a WS-π-calculus description of a service against its WS-π-calculus description to detect any errors that are not equally reflected in both descriptions [Gooneratne et al., 2006; 2007a]. The two formalisms are extensions of the Description Logic language ALUE and π-calculus respectively. Unlike existing frameworks, WS-ALUE enables the purpose, the data transformations and state transitions to be modelled together, and WS-π-calculus accurately represents the temporal relationships between interactions and their effects. The verification technique extends a depth-first graph traversal algorithm.

3.1 Motivation

Service discovery and validation techniques require a careful and accurate specification of functional and behavioural descriptions. Otherwise, located services may not provide the requested functionality and it would be difficult to correctly detect errors such as deadlocks, synchronisation conflicts and unspecified receptions in global communication models of composite services.
A description of a service is accurate if it correctly describes all the implemented features. Behavioural descriptions of services model their interaction protocols. These protocols specify the interactions and the ordering constraints between them. Interactions performed by web services can be categorized into four types. They are send, receive, invoke and service. Send is an interaction in which a parameter is dispatched and receive is one in which a parameter is accepted. Invoke dispatches a parameter and then accepts another. Service on the other hand first accepts a parameter and then dispatches another. Ordering constraints are usually specified using language constructs such as sequence, choice, parallel and iteration. A sequence construct states that interactions have to be performed in a contiguous manner. When one out of multiple alternative interactions has to be selected, it can be specified using a choice construct. An iteration is used when the same set of interactions has to be performed multiple times. The number of times they are performed can be bounded or unbounded. A parallel construct specifies that multiple interactions have to be performed concurrently. The above interaction types and language constructs are supported by most existing behavioural description frameworks (e.g. WSCL [Banerji et al., 2002], BPEL4WS [Foster et al., 2003; Shen et al., 2005] and WS-choreography [Burdett and Kavantzas, 2004]).

The functionality of a service can be modelled by describing its purpose, the data transformations or the state transitions. The purpose of a Computer Sales service is to sell computers. It transitions from an initial state “Awaiting Request for Computer” to a valid terminal state “Consigned Computer for Shipping”. It performs a transformation in which specifications for a computer and a customer’s identification details are taken as inputs and a sales invoice is produced as output. Service discovery techniques may yield inaccurate results if these three aspects are not described in functional descriptions. Let us assume that the functional description of a Computer Sales service only describes its purpose and state transitions. This service issues a sales invoice as an output. A user issues a request for a service that sells computers and provides a sales invoice. A discovery technique based on a closed-world assumption would miss this service since it presumes that any feature that is not described is not present [Elgedawy et al., 2004a; 2008]. On the other hand, a technique based on an open-world assumption would not be able to make a conclusive decision [Sycara et al., 1999; 2002]. Existing functional description frameworks such as PILLAR [Elgedawy, 2003; Elgedawy et al., 2004a], OWL-S [Martin et al., 2004; McIlraith and Martin, 2003; Sabou et al., 2003], WSDL [Christensen et al., 2001], WSML [de Bruijn et al., 2005; Keller et al., 2004], SCDL [Gao et al., 2002], and LARKS [Sycara et al., 1999; 2002] do not describe all
Assessing the correctness of functional descriptions is a difficult task since the details required to define the purpose of a service or the state transitions cannot be derived from an implementation directly. However, errors that are not equally reflected in both the functional and behavioural descriptions of a service can be detected by verifying them for consistency. Techniques that perform this are of two types: transformation-based and transition-based. Approaches of the former type detect errors by comparing the inputs and outputs in functional descriptions against the parameters of the interactions in the behavioural descriptions. For example, a Computer Sales service has an interaction that dispatches invoice.Computer and is specified in its behavioural description. However, invoice.Computer is not mentioned as an output in the functional description of the Computer Sales service. This error (occurring in the functional description) can be detected easily using a transformation-based technique. This technique detects that the functional description should contain invoice.Computer as an output and indicates that the functional and behavioural descriptions are inconsistent. Transition-based approaches require the effects of interactions to be specified in behavioural descriptions. These approaches traverse through interaction protocols and check if the state transitions in a functional description can be achieved according to the effects of interactions. For example, a valid terminal state of a Computer Sales service includes an assertion indicating that a computer is consigned for shipping. This is specified in the state transitions of its functional description. The Computer Sales service has an interaction that obtains a consignment note and updates the execution state to indicate that a computer is consigned. However, this interaction is omitted from the behavioural description due to an error. This can be detected easily using a transition-based approach. It identifies that the above assertion should not be included in the functional description of the Computer Sales service according to its behavioural descriptions, and indicates that the descriptions are inconsistent. To our knowledge, there is no technique that can verify consistency between the functional and behavioural descriptions of a single service.

Current behavioural descriptions do not specify the effects of interactions. Additionally, they cannot model the ordering constraints completely when dealing with concurrent interactions, since they do not consider the time taken by them. Interactions of web services are of two classes: (i) those that take place at time points (send and receive) and (ii) those that take place during time intervals (invoke and service). If the temporal aspects of interactions are not considered, then the effects of concurrent interactions could be considered in an
correct order when comparing behavioural descriptions against functional descriptions. Let us assume that the interactions of a *Computer Sales* service are ordered as in Figure 3.1a. A solid circle represents a “send” (interaction), an empty circle a “receive” and an empty square an “invoke”. The effects caused by the interactions are described in Table 3.1. If the time taken by the interactions are not considered, the protocol in Figure 3.1a would be modelled as in Figure 3.1b. A trace through the former protocol (which considers the time taken by the interactions) would result in the status of the computer being changed to *INVOICED*. However, this would not be the case with the latter protocol. The final status of the computer would be changed to *CONSIGNED*.

![Figure 3.1: Sample Interaction Protocol](image)

This chapter proposes two new formalisms, namely WS-*ALUE* and WS-π-calculus, for creating the functional and behavioural descriptions respectively. WS-*ALUE* is developed by extending the Description Logic language *ALUE* [Baader et al., 2003] to create descriptions that specify all three functional aspects together. Description Logic has been used for knowledge representation in various application domains because of its ability to represent...
CHAPTER 3. CREATING ACCURATE FUNCTIONAL AND BEHAVIOURAL DESCRIPTIONS

<table>
<thead>
<tr>
<th>Interaction</th>
<th>Description of effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awaiting request from customer</td>
<td></td>
</tr>
</tbody>
</table>
| 1 | Status of computer is changed to **REQUESTED**  
Status of customer is changed to **IDENTIFIED** |
| 2 | Status of insurance is changed to **QUOTED** |
| 3 | Status of computer is changed to **INSURED**  
Status of insurance is changed to **ONGOING** |
| 4 | Status of computer is changed to **CONSIGNED**  
Pickup date for the computer is **CONFIRMED** |
| 5 | Status of customer is changed to **INVOICED** |

*Table 3.1: Effects of Interactions*

knowledge in a structured and formally understood manner [Baader et al., 2003]. **ACLEE** is the minimal language that defines most of the constructs required to represent the functional aspects of a service. We extend **ACLEE** with some new constructs. They allow the attributes (that describe the functional aspects) to be specified in an easily differentiable manner.

**WS-π-calculus** extends **π-calculus** [Sangiorgi and Walker, 2001] with Interval Time Logic (ITL) axioms [Allen, 1991; Allen and Ferguson, 1994] and **WS-ACLEE** constructs. ITL axioms model temporal relationships between time intervals. In **WS-π-calculus**, they accurately specify the ordering constraints between concurrent interactions. The effects of interactions are specified with **WS-ACLEE** constructs. **WS-choreography** [Burdett and Kavantzas, 2004] and **π-calculus** [Sangiorgi and Walker, 2001] are the only current behavioural description frameworks that model the communicating entities in interactions. Modeling the communicating entities is important when differentiating between multiple interactions of similar types that exchange the same parameters. **π-calculus** has a simple structure since it was only designed to represent and analyse interaction protocols of communicating processes. **WS-choreography** is an extension of **π-calculus**. Unlike **π-calculus**, **WS-choreography** is able to represent all four types of interactions, and was mainly designed to model global communication models of composite services. We selected **π-calculus** as the basis to develop the proposed formalism for creating behavioural descriptions because of its simple structure and the features that it already possessed (such as support for all four language constructs and the ability to represent the type, parameters and communication channels of interactions).
We also propose a transition-based technique to compare a behavioural description of a service against its functional description. This technique checks if a valid terminal state of a service can be derived from its initial execution state by updating the execution state with the effects of its interactions. These updates are performed in an incremental manner by traversing through an interaction protocol using a pre-order depth-first search algorithm. An interaction protocol cannot be traversed if it contains any structural conflicts [Kiepuszewski et al., 2000; Sadiq and Orlowska, 2000] and cannot be traversed in finite time if it contains any unbounded loops. Therefore, we identify the class of interaction protocols that are finite and free of structural conflicts, and characterize WS-π-calculus descriptions that represent such interaction protocols. We also propose a polynomial algorithm that verifies such WS-π-calculus descriptions against WS-ALUE descriptions.

A case study in which the consistency of the functional and behavioural descriptions of a Computer Sales service is verified is provided. The descriptions are specified with WS-ALUE and WS-π-calculus and verified using the proposed technique. An error is deliberately incorporated to the WS-ALUE description to show how the verification technique can be used to detect errors in descriptions.

The rest of the chapter is organised as follows. The next section gives a brief description of Interval Time Logic, which is used to extend π-calculus. Section 3.3 describes the proposed formalisms WS-ALUE and WS-π-calculus. Section 3.4 presents the proposed verification technique which compares a WS-π-calculus description against a WS-ALUE description. Section 3.5 provides the case study. Section 3.6 provides an analysis of some current functional and behavioural description frameworks. Then in Section 3.7, we place the proposed approach in the context of other verification techniques that are used with web services. Finally, a summary of the contributions described in this chapter, is provided in Section 3.8.

### 3.2 Interval Time Logic (ITL)

Interval Time Logic (ITL) axioms [Allen, 1991; Allen and Ferguson, 1994; Bellini et al., 2000] provide a simple approach for representing the temporal relationships between time interval based events. The relationships are represented with axioms of the form $r(e_1, e_2)$. These axioms are used in WS-π-calculus to specify the temporal relationships between concurrent interactions. There are 13 different types of relationships [Allen, 1991; Allen and Ferguson, 1994; Bellini et al., 2000]. The relationship types, the notations used to represent them and their graphical representations are provided in Table 3.2.
<table>
<thead>
<tr>
<th>Type</th>
<th>Notation</th>
<th>Graphical representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overlaps</td>
<td>$o(x, y)$</td>
<td><img src="image" alt="Overlap" /></td>
</tr>
<tr>
<td>Overlapped-by</td>
<td>$oi(y, x)$</td>
<td><img src="image" alt="Overlap" /></td>
</tr>
<tr>
<td>Starts</td>
<td>$s(x, y)$</td>
<td><img src="image" alt="Start" /></td>
</tr>
<tr>
<td>Started-by</td>
<td>$si(y, x)$</td>
<td><img src="image" alt="Start" /></td>
</tr>
<tr>
<td>During</td>
<td>$d(x, y)$</td>
<td><img src="image" alt="During" /></td>
</tr>
<tr>
<td>Contains</td>
<td>$di(y, x)$</td>
<td><img src="image" alt="Contains" /></td>
</tr>
<tr>
<td>Finishes</td>
<td>$f(x, y)$</td>
<td><img src="image" alt="Finishes" /></td>
</tr>
<tr>
<td>Finished-by</td>
<td>$fi(y, x)$</td>
<td><img src="image" alt="Finished" /></td>
</tr>
<tr>
<td>Meets</td>
<td>$m(x, y)$</td>
<td><img src="image" alt="Meets" /></td>
</tr>
<tr>
<td>Met-by</td>
<td>$mi(y, x)$</td>
<td><img src="image" alt="Met-by" /></td>
</tr>
<tr>
<td>Before</td>
<td>$b(x, y)$</td>
<td><img src="image" alt="Before" /></td>
</tr>
<tr>
<td>After</td>
<td>$bi(y, x)$</td>
<td><img src="image" alt="After" /></td>
</tr>
<tr>
<td>Equals</td>
<td>$(=)(x, y)$</td>
<td><img src="image" alt="Equals" /></td>
</tr>
</tbody>
</table>

*Table 3.2: Temporal Relationships between Concurrent Interactions*

### 3.3 Proposed Formalisms

This section describes the extensions made to ALUE and $\pi$-calculus to include the proposed formalisms for specifying functional and behavioural descriptions.

#### 3.3.1 WS-ALUE

WS-ALUE models all three functional aspects of a service. The purpose is described using an operation. This operation is constrained by the role it performs in a domain and an affected concept. For example, a *Computer Sales* service would perform the operation *Sales()*.
It would affect the concept \textit{Computer} and perform the role of an \textit{Sales Assistant}. The data transformations are described with inputs and outputs. \textquotedblleft In-constraints	extquotedblright{} and \textquotedblleft out-constraints	extquotedblright{} specify restrictions that apply to the values used as inputs and outputs. State transitions are described with pre-conditions and post-conditions. They describe conditions that need to hold for a service to execute and the effects of an execution respectively.

The conditions (used to specify the in-constraints, out-constraints, pre-conditions and post-conditions) model a relationship between an attribute and a value. A condition explicitly reduces the scope of an attribute to a particular domain. The scope of an attribute is the set of values that can be assigned to the attribute and it is defined by associating an attribute with a concept. For example, the scope of the attribute \textit{dispatchDate} cannot be determined unless it is specified as \textit{dispatchDate}.\textit{Computer}. Then any value (specifying the time taken to dispatch a computer) can be assigned to the attribute \textit{dispatchDate}. When an attribute is used in a condition, its scope is reduced to a particular domain. For example, in a condition of the form \textit{dispatchDate}.\textit{Computer} < 4 DAYS, the scope of \textit{dispatchDate}.\textit{Computer} is reduced to a value that is less than 4 days. The set of values that can be assigned to the attribute of a condition are referred to as a condition domain.

\textbf{Definition 3.1 (Condition)} A condition \textit{cnd} is specified with a concept \textit{c}, an attribute \textit{a}, a comparative operator \textit{o} and a value \textit{v}, where operator \textit{o} defines a relationship that must hold between attribute \textit{a} and value \textit{v}, \textit{o} can be any of the comparative operators =, <, >, \neq, \geq, \leq, \in, \ni, \exists, or \notin, and the scope of \textit{a} is defined with the concept \textit{c}.

Examples of conditions are \textit{dispatchDate}.\textit{Computer} < 4 DAYS, \textit{validRegion}.\textit{Insurance} \in \textit{AUSTRALIA} and \textit{status}.\textit{Computer} = \textit{INVOICED}.

WS-\textit{ALUE} extends the DL language \textit{ALUE} with some new constructs. \textit{ALUE} is not powerful enough for specifying functional descriptions for the following reasons.

1. The value restriction constructs provide an inelegant and unscalable approach for modeling conditions. For example, the condition \textit{dispatchDate}.\textit{Computer} < 4 DAYS would be specified in a DL description as\footnote{We assume that the condition domain contains the values \textit{1.DAY}, \textit{2.DAYS}, \textit{3.DAYS} and \textit{4.DAYS}.}:

$$\forall \textit{dispatchDate}.\textit{Computer} \land (\textit{dispatchDate}(\textit{1.DAY}) \lor \textit{dispatchDate}(\textit{2.DAYS}) \lor \textit{dispatchDate}(\textit{3.DAYS}) \lor \textit{dispatchDate}(\textit{4.DAYS}))$$
CHAPTER 3. CREATING ACCURATE FUNCTIONAL AND BEHAVIOURAL DESCRIPTIONS

This representation of a condition is not elegant and results in very long descriptions when conditions have large domains.

WS-\textit{ALUE} replaces value restriction constructs with open value restriction constructs. This new construct provides a simple and elegant approach for modeling a constraint. An open value restriction construct is a binary predicate that describes a relationship between a value and an attribute. Details regarding the structure of this construct are provided in Table 3.3.

2. The constructs of \textit{ALUE} are not capable of differentiating between:

(a) an operation, a role and an effected concept. For example, the purpose of an \textit{Insurance} service would be represented in \textit{ALUE} as:

\[ \text{Insure} \sqcap \text{Computer} \sqcap \text{Insurance\_Representative} \]

There is no distinction between the operation \text{Insure()}, the concept \text{Computer} and the role \text{Insurance\_Representative}.

(b) inputs and outputs. For example, the parameters of a \textit{Computer Sales} service can be specified as \( \exists \text{specification}\_\text{Computer}, \exists \text{details}\_\text{Customer}, \) and \( \exists \text{invoice}\_\text{Computer} \) using full existential quantification constructs. However, there is no way to identify whether a certain expression is an input or an output.

(c) pre-conditions, post-conditions, in-constraints, and out-constraints. The state transitions and the constraints applied to the parameters can be specified using open value restriction constructs. (e.g. \( = \text{WAITING\_REQUEST status}\_\text{Computer} \), \( = \text{INVOICED status}\_\text{Computer} \), \( \geq 18 \text{\_YEARS age}\_\text{Customer} \) and \( < 4 \text{\_DAYS dispatchDate}\_\text{Computer} \)). However, there is no way to differentiate between the different types of conditions.

The proposed formalism, WS-\textit{ALUE}, introduces new axioms to differentiate between the various elements of a description. The syntax of its constructs are defined in Table 3.3\textsuperscript{2}.

In WS-\textit{ALUE}, a functional description is represented with an operation definition. An operation \( O \) is created using the constructs of WS-\textit{ALUE} as follows.

\textsuperscript{2} The comparison operator in open value restriction constructs is represented with \( \geq \) for simplicity. It can be replaced with either \( =, <, >, \neq, \leq, \subset, \subseteq, \supset, \supseteq, \in, \exists, \) or \( \notin \).
CHAPTER 3. CREATING ACCURATE FUNCTIONAL AND BEHAVIOURAL DESCRIPTIONS

<table>
<thead>
<tr>
<th>Construct</th>
<th>Syntax</th>
<th>Extensions in WS-ALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>$P$</td>
<td></td>
</tr>
<tr>
<td>Atomic Concept</td>
<td>$A$</td>
<td>✓</td>
</tr>
<tr>
<td>Atomic Role</td>
<td>$T$</td>
<td>✓</td>
</tr>
<tr>
<td>Atomic Operation</td>
<td>$O$</td>
<td>✓</td>
</tr>
<tr>
<td>Complex Concept</td>
<td>$C$</td>
<td></td>
</tr>
<tr>
<td>Top Concept</td>
<td>$\top$</td>
<td></td>
</tr>
<tr>
<td>Bottom Concept</td>
<td>$\bot$</td>
<td></td>
</tr>
<tr>
<td>Concept Conjunction</td>
<td>$C \cap C'$</td>
<td></td>
</tr>
<tr>
<td>Concept Disjunction</td>
<td>$C \cup C'$</td>
<td></td>
</tr>
<tr>
<td>Concept Assertion</td>
<td>$C(a)$</td>
<td></td>
</tr>
<tr>
<td>Full existential quantification</td>
<td>$\exists P.C$</td>
<td></td>
</tr>
<tr>
<td>Open value restriction</td>
<td>$\geq n \ P.C$</td>
<td>✓</td>
</tr>
<tr>
<td>Role predicate</td>
<td>$Role()$</td>
<td>✓</td>
</tr>
<tr>
<td>Input predicate</td>
<td>$Input()$</td>
<td>✓</td>
</tr>
<tr>
<td>Output predicate</td>
<td>$Output()$</td>
<td>✓</td>
</tr>
<tr>
<td>In-constraint predicate</td>
<td>$InCon()$</td>
<td>✓</td>
</tr>
<tr>
<td>Out-constraint predicate</td>
<td>$OutCon()$</td>
<td>✓</td>
</tr>
<tr>
<td>Pre-condition predicate</td>
<td>$PreCon()$</td>
<td>✓</td>
</tr>
<tr>
<td>Post-condition predicate</td>
<td>$PostCon()$</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 3.3: WS-ALUE Constructs

$$O = C$$  
Affected concept

- $\cap Role(T)$  
Role
- $\cap < Input(\exists P.C) >$  
Set of inputs
- $\cap < InCon(\geq n \ P.C) >$  
Set of in-constraints
- $\cap < Output(\exists P.C) >$  
Set of outputs
- $\cap < OutCon(\geq n \ P.C) >$  
Set of out-constraints
- $\cap < PreCon(\geq n \ P.C) >$  
Set of pre-conditions
- $\cap < PostCon(\geq n \ P.C) >$  
Set of post-conditions

An operation definition consists of an affected concept $C$, a role $T$ and sets of inputs,
outputs, in-constraints, out-constraints, pre-conditions and post-conditions.

\[ C = A \mid \top \mid \bot \mid C \cap C' \mid C \cup C' \mid C(a) \]

\( C(a) \) models a specific instance of a concept. For example, if a web service sells DELL computers, the affected concept is specified as \( \text{Computer}(DELL) \). The following is a WS-\( \mathcal{ALUE} \) description of a \( \text{Computer Sales} \) service.

\[
\text{Sales}() = \text{Computer} \cap \text{Role(Sales Assistant)} \\
\cap \text{PreCon}(= \text{AWAITING REQUEST status.Computer}) \\
\cap \text{PostCon}(= \text{CONSIGNED status.Computer}) \\
\cap \text{PostCon}(= \text{INVOICED status.Customer}) \\
\cap \text{PostCon}(= \text{CONFIRMED pickupDate.Consignment}) \\
\cap \text{PostCon}(= \text{ONGOING status.Insurance}) \\
\cap \text{Input}(\exists \text{specification.Order}) \cap \text{Input}(\exists \text{quote.Insurance}) \\
\cap \text{Input}(\exists \text{note.Consignment}) \cap \text{Input}(\exists \text{policy.Insurance}) \\
\cap \text{Output}(\exists \text{noteRequest.Consignment}) \cap \text{Output}(\exists \text{invoice.Customer}) \\
\cap \text{Output}(\exists \text{quoteRequest.Insurance}) \cap \text{Output}(\exists \text{policyRequest.Insurance}) \\
\cap \text{InCon}(= \text{date.System policyDate.Insurance}) \\
\cap \text{InCon}(> 18 \text{ YEARS customerAge.Order}) \\
\cap \text{OutCon}(\geq \text{date.System invoiceDate.Computer})
\]

### 3.3.2 WS-\( \pi \)-calculus

WS-\( \pi \)-calculus is introduced to allow the creation of behavioural descriptions for web services. This formalism is developed by extending \( \pi \)-calculus with two new types of interactions (i.e. invoke and service), ITL axioms [Allen, 1991; Allen and Ferguson, 1994] and WS-\( \mathcal{ALUE} \) constructs. \( \pi \)-calculus is not sufficient for specifying behavioural descriptions of web services due to the following reasons.

1. Invoke and service interactions are not supported. Even though multiple parameters can be sent or received with a single interaction in the polyadic form, one in which a single parameter is sent and another is received or vise versa cannot be specified. All the parameters have to be sent in one direction.
2. The time taken by the interactions are not considered when specifying the ordering constraints between concurrent interactions. This results in an inaccurate representation of ordering constraints. The temporal aspects are not considered in π-calculus because time interval-based interactions (i.e. invoke and service) are not supported.

3. The effects of an interaction cannot be represented. This hinders the ability to compare a behavioural description against a functional description using a transition-based approach.

WS-π-calculus rectifies these issues and extends π-calculus as follows.

1. Defines notations for specifying invoke and service interactions.

2. Represents temporal relationships between concurrent interactions using Interval Time Logic (ITL) axioms.

3. Specifies the effects of interactions with a list of “add-conditions” and a list of “delete-conditions”.

In WS-π-calculus, an interaction $i$ can be of any of the four types send, receive, invoke or service and is represented as follows.

$$i = \text{ent}[\text{send}](p) \mid \text{ent}[\text{receive}](p) \mid \text{ent}[\text{invoke}](p_d p_a) \mid \text{ent}[\text{service}](p_a p_d)$$

The entity involved in an interaction is represented with $\text{ent}$. The type is specified in $[\ ]$. Parameters are specified in $( )$, with $p_d$ representing a dispatched parameter and $p_a$ representing one that is accepted. This distinction is not required with send and receive interactions, since only one parameter is either dispatched or accepted. The parameters are represented with full existential quantification DL constructs. Effects of interactions are specified with a list of add-conditions and delete-conditions. These conditions are specified with WS-$\text{ALUE}$ open value restriction constructs. The two types of conditions are differentiated with $\text{Add}()$ and $\text{Delete}()$ predicates. A relationship between two concurrent interactions $i_1$ and $i_2$ is represented with an ITL axiom $r(i_1, i_2)$, where $r$ is a temporal relationship type\(^3\). For example, the temporal relationships between the three concurrent interactions $\text{Get Insurance Quote}$, $\text{Get Insurance Policy}$ and $\text{Get Consignment Note}$ of the $\text{Computer Sales}$ service in Figure 3.1a are specified in WS-π-calculus as follows\(^4\).

\(^3\)The different types of ITL axioms are given in Table 3.2.

\(^4\)cs$_i$ refers to the $i^{th}$ interaction of the $\text{Computer Sales}$ service.
\[ s(cs_2, cs_4), b(cs_2, cs_3), d(cs_3, cs_4) \]

**Definition 3.2 (WS-π-calculus Interaction Protocol)** An interaction protocol of a web service is described as a process \( P \) in WS-π-calculus and is defined as follows.

\[
P = i.P \mid < r(P, P) > \mid P + P \mid < n >!P \mid 0
\]

As with \( \pi \)-calculus, \( \theta \) denotes an inaction. This allows the termination of an interaction protocol to be specified explicitly.

A WS-π-calculus specification of a *Computer Sales* service is provided in the following.

\[
P = cs_1.s(cs_2, cs_4), b(cs_2, cs_3), d(cs_3, cs_4).cs_5
\]

\[
cs_1 = \text{Customer}[\text{receive}]((\exists \text{specification}.\text{Computer}, \exists \text{details}.\text{Customer})
\]

\[
\text{Add}(= \text{REQUESTED status}.\text{Computer}),
\text{Add}(= \text{RECEIVED specification}.\text{Order})
\]

\[
\text{Delete}(= \text{AWAITING_REQUEST status}.\text{Computer})
\]

\[
\cs_2 = \text{Insurance}[\text{invoke}]((\exists \text{quoteRequest}.\text{Insurance})
\]

\[
\text{Add}(= \text{QUOTED status}.\text{Insurance})
\]

\[
\cs_3 = \text{Insurance}[\text{invoke}]((\exists \text{policyRequest}.\text{Insurance})
\]

\[
\text{Add}(= \text{ONGOING status}.\text{Insurance}),
\text{Add}(= \text{INSURED status}.\text{Computer}),
\text{Delete}(= \text{QUOTED status}.\text{Computer}),
\text{Delete}(= \text{REQUESTED status}.\text{Computer})
\]

\[
\cs_4 = \text{Shipping}[\text{invoke}]((\exists \text{noteRequest}.\text{Consignment})
\]

\[
\text{Add}(= \text{CONFIRMED pickupDate}.\text{Consignment}),
\text{Add}(= \text{CONSIGNED status}.\text{Computer})
\]

\[
\text{Delete}(= \text{INSURED status}.\text{Computer})
\]

\[
\cs_5 = \text{Customer}[\text{send}]((\exists \text{invoice}.\text{Customer})
\]

\[
\text{Add}(= \text{INVOICED status}.\text{Customer})
\]

\[
\text{Delete}(= \text{RECEIVED specification}.\text{Order})
\]
3.4 Verification Technique

Here we propose a technique to detect errors in functional and behavioural descriptions of services. This technique detects these errors as long as they are not equally reflected in both descriptions. Errors are detected by checking if the state transitions specified in a WS-
ALUE description can be achieved according to the effects of interactions in a WS-\(\pi\)-calculus description. That means this technique determines that there is an error in at least one description of a web service \(WS\) if the condition
\[
post(WS) = pre(WS) \cup effects(P)
\]
is not achievable, where \(pre(WS)\) is the set of pre-conditions, \(post(WS)\) is the set of post-conditions, and \(P\) is the interaction protocol of \(WS\).

This technique traverses through an interaction protocol and incrementally updates the execution state of a service according to the effects of its interactions. Finally, the updated execution state is compared against the valid terminal states to determine whether the descriptions are error-free. In WS-
ALUE descriptions, the initial execution state and the valid terminal states are defined with a set of pre-conditions and a set of post-conditions respectively.

The proposed technique cannot be used when the interaction protocol specified in a behavioural description contains any structural conflicts (i.e. deadlocks or synchronisation conflicts), or is unbounded. An interaction protocol that contains a structural conflict cannot be traversed. The final execution state that is compared against the valid terminal state cannot be derived if an interaction protocol is unbounded. For example, one may notice that the interaction \(D\) of the interaction protocol in Figure 3.2a is unreachable since it contains an unbounded loop. A final execution state to compare against a valid terminal state cannot be derived since the point at which the iterations terminate and the execution state is updated with the effects of \(D\) cannot be established. The interaction protocols shown in Figure 3.2b and 3.2c cannot be traversed since the first contains a deadlock and the latter contains a synchronisation conflict. In Figure 3.2b, interaction \(D\) can only be executed if interactions \(B\) and \(C\) are executed in a synchronized manner. However, interactions \(B\) and \(C\) cannot be executed together since an or-split occurs after interaction \(A\). Similarly, the interactions \(B\) and \(C\) in Figure 3.2c have to be executed in a synchronized manner because an and-split occurs after \(A\). However, this conflicts with the or-join that occurs at interaction \(D\) since only one of the interactions, either \(B\) or \(C\) can be executed. This is a typical example of a
CHAPTER 3. CREATING ACCURATE FUNCTIONAL AND BEHAVIOURAL DESCRIPTIONS

synchronisation conflict.

![Diagrams showing unbounded, deadlock, and synchronisation conflict interactions.](image)

Figure 3.2: Untraversable Interaction Protocols

We introduce the concept of a Valid Interaction Protocol to identify those interaction protocols that can be traversed with a finite number of computational operations.

**Definition 3.3 (Valid Interaction Protocol (VIP))** An interaction protocol $P$ is a VIP if it consists of a finite number of interactions, and is free of deadlocks and synchronisation conflicts.

Next, we demonstrate that a VIP can be traversed with a finite number of operations. This requires an understanding of the notion of an execution path.

**Definition 3.4 (Execution Path)** An execution path $EP$ of a WS-$\pi$-calculus interaction protocol $P$ is a sub-process of $P$, where $EP$ contains the first and the final interactions of $P$, and it does not contain any choice constructs.

As a behavioural description may contain choice constructs, it is important to ensure that interactions are not taken from different execution paths when traversing through protocols. This is because interactions from different paths cannot be performed together in a single execution of a service (as they correspond to different choices made during execution). For example, the interaction protocol depicted in Figure 3.3a would have the execution paths in Figure 3.3b and Figure 3.3c. A single execution of this service cannot include either $B$ and $D$ or $C$ and $D$.

The number of execution paths in an interaction protocol is determined by the number of or-splits that it contains. The set of execution paths of an interaction protocol can be obtained by generating its coverage. The coverage of an interaction protocol is a set of finite paths beginning at the initial interaction $i_0$ where each interaction belongs to at least one path. A coverage can be generated if an interaction protocol is reachable, deterministic and...
finite [Bourdonov et al., 2003]. We assume that the interactions protocols considered here are data independent [Aggarwal et al., 1990]. Therefore, they are deterministic. The (deterministic or non-deterministic) process that selects a particular path at a choice construct and the variables considered by this process are invisible. All the interactions of a protocol are reachable if it does not contain any deadlocks or synchronisation conflicts [Sadiq and Orlowska, 1999; 2000]. Thus, all the execution paths of a VIP can be traversed.

**Proposition 3.1**  The execution paths of an interaction protocol $P$ can be traversed if $P$ is a VIP.

**Proof**

1. Since $P$ is a VIP

   (a) Every interaction $i_j$ of $P$ is reachable from its initial interaction $i_0$ (VIPs do not contain any deadlocks or synchronisation conflicts).

   (b) $P$ contains a finite number of interactions.

2. The coverage graph $C(P)$ of $P$ can be generated since it is finite and reachable.

3. The execution paths of $P$ can be traversed since $C(P)$ can be generated. ♦

**Characterization of VIPs**

Here we provide a characterization of the class of WS-$\pi$-calculus descriptions that represent VIPs. This enables us to clearly identify the WS-$\pi$-calculus descriptions that can be verified with the proposed technique.
An interaction protocol is finite and free of deadlocks and synchronisation conflicts if it can be mapped to a normal workflow which only contains bounded structured cycles\(^5\). That means, an interaction protocol is a VIP if it does not contain any arbitrary cycles, arbitrary split and joins or unbounded structured cycles.

The choice and parallel constructs of WS-\(\pi\)-calculus only support non-conflicting and corresponding split and joins. For example, the interaction protocol depicted in Figure 3.4a cannot be specified in WS-\(\pi\)-calculus. It would be specified as either \(A.(= (B, C).E) + D).F\) or \(A.((B+C).E) + D).F\), which would be equivalent to the interaction protocols depicted in Figures 3.4b and 3.4c respectively. Hence, WS-\(\pi\)-calculus descriptions that use the sequence, choice and parallel constructs model VIPs. The iteration construct of WS-\(\pi\)-calculus only supports structured cycles. Thus, a WS-\(\pi\)-calculus description with a bounded number of iterations represents a VIP.

\[ P = \quad i.P \quad | \quad < r(P, P) > \quad | \quad P + P \quad | \quad n!P \quad | \quad 0 \]

Definition 3.5 (WS-\(\pi\)-calculus Valid Interaction Protocol) A valid interaction protocol \(P\) in WS-\(\pi\)-calculus is defined as follows\(^6\).

Join Calculus [Sangiorgi and Walker, 2001; Smith and Fingar, 2004] which extends \(\pi\)-calculus with a new join construct is able to represent control patterns that have conflicting and non-corresponding splits and joins. However, it is undesirable to have web services with interaction protocols that have conflicting and non-corresponding splits and joins because

---

\(^5\)See Section 2.2.3 for a description on different types of cycles, split and joins, and workflows.

\(^6\)Note that the \(n\) in iteration constructs \(n!P\) is not optional here.
it allows deadlocks and synchronisation conflicts to occur [Sadiq and Orlowska, 1999; 2000]. Hence, WS-\(\pi\)-calculus extends the basic \(\pi\)-calculus, instead of Join Calculus since such a level of expressiveness is not required.

The Proposed Algorithm

The following describes an algorithm to detect errors in functional and behavioural descriptions of services. We assume that either the functional description or the behavioural description is specified accurately. Hence, we do not consider situations where errors are equally reflected in both descriptions of a service. Such errors cannot be detected with the proposed algorithm. Also, we assume that a given behavioural description models a VIP.

This algorithm requires a WS-\(A\Re\Le\E\) description which specifies an initial execution state \(S\) and a valid terminal state \(VTS\), and a WS-\(\pi\)-calculus description which specifies an interaction protocol \(P\), where \(<EP>\) is the set of execution paths in \(P\), \(EP \in <EP>\) and \(EP = [i_1, \ldots, i_n]\). This algorithm checks if the following condition can be achieved.

\[
\exists EP \in <EP> \quad \text{where } VTS = S \cup \text{effect}(i_1) \cup \ldots \cup \text{effect}(i_n) \tag{3.1}
\]

The function \(\text{effect}(i_x)\) specifies the way in which the execution state is modified when interaction \(i_x\) is performed. There is an error in at least one description (i.e. either the WS-\(A\Re\Le\E\) description, the WS-\(\pi\)-calculus description or both) if the condition given in Equation 3.1 cannot be achieved.

The proposed algorithm traverses through an interaction protocol of a service to incrementally update its execution state with the effects of the interactions. The algorithm uses a pre-order depth-first search technique to perform the traversal [Ghosh and Bhattacharjee, 1984]. When updating the execution state of a service with the effects of each interaction, changes have to be performed in the correct order (as when the service is actually executed). This can be accomplished easily by updating the execution state according to the ordering constraints of an interaction protocol. However, the order in which effects of concurrent interactions should be updated cannot be determined directly because ordering constraints are represented with multiple ITL axioms. For example, the ordering constraints between the concurrent interactions of the Computer Sales service in Figure 3.1a are modelled with \(s(cs_2, cs_4), b(cs_2, cs_3), \text{ and } d(cs_3, cs_4)\). The first interaction to complete its execution cannot be identified directly from these ITL axioms. Hence, the proposed algorithm converts the parallel constructs to a series of sequence constructs according to the order in which inter-
actions complete their execution. By sequencing the parallel constructs we guarantee that the execution state of a service is modified with the effects of concurrent interactions in the correct order. Concurrent interactions are sequenced using a completion index. Each element of this index corresponds to an interaction $i$ and contains the list of concurrent interactions that complete their execution before $i$. Figure 3.5 depicts the way in which a completion index is generated for the above concurrent interactions. The order in which the concurrent interactions $cs_2$, $cs_3$ and $cs_4$ complete their execution can be determined easily with this index.

![Completion Index Diagram]

The algorithm used to sequence the parallel constructs is given below (see Algorithm 3.1).

The following are brief descriptions of the functions used by this algorithm.

- **firstToComplete(a):** Returns the interaction that completes its execution first from the two mentioned in an ITL axiom $a$ (line 8).
- **secondToComplete(a):** Returns the interaction that completes its execution last from the two mentioned in an ITL axiom $a$ (line 8).
- **hasMoreInteractions(I):** Checks if a completion index $I$ contains any information (line 10).
- **next(I):** Returns the interactions that completes its execution next in a completion index $I$, and removes the row that corresponds to the returned interaction from the index (line 11).

Algorithm 3.1 takes an interaction protocol (process) $P$ and converts its parallel constructs to a series of sequence constructs. A loop is used to traverse through each sub-process $p$ in
$P$. If $p$ is not an atomic interaction and not a parallel construct it is recursively passed to the sequencing function (line 15). This enables nested parallel constructs to be sequenced. Parallel constructs are sequenced using a Completion Index $I$. First, this index is generated from the details in the ITL axioms (line 7-9). Then, a sequence construct is generated by retrieving interactions from $I$ based on the order in which they complete their execution (lines 10-12). Finally, the parallel construct is replaced with the generated sequence construct (line 13). The complexity of this algorithm is $mn$, where $m$ is the number of sub-process in a given interaction protocol $P$ and $n$ is the number of concurrent interactions in each parallel construct.

1. **Process sequencing($P$)**
2. Initialise Completion Index $I$;
3. **for each process** $p \in P$ **do**
   4. **if** $p$ is not an atomic interaction **then**
      5. **if** $p$ is a parallel construct **then**
         6. $P' \leftarrow \emptyset$;
         7. **for each ITL axiom** $a \in p_i$ **do**
            8. $I[\text{secondToComplete}(a)].\text{add}(\text{firstToComplete}(a))$;
         9. **end**
     10. **while** hasMoreInteractions($I$) **do**
        11. $P' \leftarrow P'.\text{next}(I)$;
        12. **end**
    13. **else**
        14. sequencing($p$);
    15. **end**
4. **end**

**Algorithm 3.1**: Algorithm for Sequencing Parallel Constructs

Algorithm 3.2 provides the evaluate function used to traverse through an interaction protocol, update the execution state according to the effects of interactions and check if the
valid terminal state can be derived from the initial execution state. This algorithm only has to consider the sequence, choice and iteration constructs when traversing through an interaction protocol (since the sequencing function converts parallel constructs to sequence constructs).

The task performed by Algorithm 3.2 can be summarised as follows. Assume a web service $WS$ has an initial state $S_0$ and an interaction protocol that contains the execution paths $<EP>$, where each $EP = [i_1, ..., i_n]$. The evaluate function traverses through an execution path $EP$, updates every state $S_{(x-1)}$ according to the effects of interaction $i_x$ and obtains state $S_x$, where $1 \leq x \leq n$. Once a state $S_n$ is derived it is compared against the valid terminal state $VTS$. Then, $VTS$ is returned if $S_n$ conforms to $VTS$. Otherwise, this algorithm moves to the next execution path of $P$. If all the execution paths have been traversed, then $S_n$ is returned. The evaluate function requires an initial execution state $S$, a valid terminal state $VTS$ and an interaction protocol $P$. This function returns $VTS$ if it can be derived from $S$ according to the effects of interactions of one execution path. Otherwise, an arbitrary terminal state is returned.

The corresponding algorithm for the evaluate function is given below (Algorithm 3.2). The first sub-process $P' \in P$ is assigned to $M$ and the remaining sub-processes are assigned to $N$ (lines 5 and 6). We assume that a web service will always have at least one interaction. That means the minimal WS-$\pi$-calculus description would be $i.0$, where $i$ is assigned to $M$ and $0$ to $N$. If $M$ is an interaction, the effect of executing $M$ is applied to current execution state $S$ (line 22). Then, a recursive call is made to the evaluate function to apply the effects of the remaining interactions in $N$. If $N$ is $\emptyset$, the current execution state $S$ would be returned (line 3). If $M$ is a choice construct (line 7), then each sub-process $P' \in M$ denotes an alternative execution path. Hence, we generate each alternative path $Q$ by merging $P'$ with $N$ (line 9). The states derived by traversing the separate paths are compared against the valid terminal state. If the derived state conforms to the valid terminal state the valid terminal state is returned (line 12). If $M$ is an iteration construct (line 16), where $M$ is equal to $n!P'$, then the execution state is updated $n$ number of times with the effects of $P'$. The complexity of the evaluate function is the same as that of a simple pre-order depth first search algorithm, which is $mn$ where $m$ is the number of choice constructs and $n$ is the number of interactions in an execution path [Ghosh and Bhattacharjee, 1984].
Algorithm 3.2: Algorithm for Comparing WS-ALUE and WS-π-calculus Descriptions

3.5 Case Study

This section shows how the proposed verification technique is used to detect errors in service descriptions. This is performed by comparing a WS-π-calculus description of a sample Computer Sales service against its WS-ALUE description. An error is deliberately incor-
Porated into the WS-\textsc{ALUE} description. According to the WS-\(\pi\)-calculus description, \textit{status.Customer} should be either \textit{PAYMENT\_RECEIVED} or \textit{PROCESSING\_PAYMENT} once an execution of this service is completed. However, the WS-\textsc{ALUE} description states that \textit{status.Customer} should be \textit{INVOICED} in a valid terminal state. The WS-\textsc{ALUE} description is given in Figure 3.6.

\begin{align*}
Sales & = \text{Computer} \sqcap \text{Role}(Sales\_Assistant) \\
& \quad \sqcap \text{PreCon}(= \text{AWAITING\_REQUEST} \text{status.Computer}) \\
& \quad \sqcap \text{PostCon}(= \text{SHIPPED} \text{status.Computer}) \\
& \quad \sqcap \text{PostCon}(= \text{ONGOING} \text{status.Insurance}) \\
& \quad \sqcap \text{PostCon}(= \text{INVOICED} \text{status.Customer}) \\
& \quad \sqcap \text{PostCon}(= \text{CONFIRMED} \text{pickupDate.Consignment}) \\
& \quad \sqcap \text{Input}(\exists \text{specification.Computer}) \sqcap \text{Input}(\exists \text{details.Customer}) \\
& \quad \sqcap \text{Input}(\exists \text{note.Consignment}) \sqcap \text{Input}(\exists \text{quote.Insurance}) \\
& \quad \sqcap \text{Input}(\exists \text{policy.Insurance}) \sqcap \text{Input}(\exists \text{details.DirectDebit}) \\
& \quad \sqcap \text{Input}(\exists \text{details.CreditCard}) \\
& \quad \sqcap \text{Output}(\exists \text{quoteRequest.Consignment}) \\
& \quad \sqcap \text{Output}(\exists \text{noteRequest.Consignment}) \\
& \quad \sqcap \text{Output}(\exists \text{quoteRequest.Insurance}) \\
& \quad \sqcap \text{Output}(\exists \text{invoice.Customer}) \sqcap \text{Input}(\exists \text{receipt.Customer}) \\
& \quad \sqcap \text{InCon}(= \text{date.System policyDate.Insurance}) \\
& \quad \sqcap \text{InCon}(> 18\_YEARS \text{age.Customer}) \\
& \quad \sqcap \text{OutCon}(\geq \text{date.System invoiceDate.Computer})
\end{align*}

\textit{Figure 3.6: Sample WS-\textsc{ALUE} Description}

The following describes the interaction protocol of the \textit{Computer Sales} service (see Figure 3.5a for a graphical representation). The WS-\(\pi\)-calculus description of the interaction protocol \(P\) of the \textit{Computer Sales} service is given in Figure 3.8.

1. Receives a specification of a computer from a customer.
2. Receives a customer’s identification details.
3. Invokes an *Insurance* service to obtain a quote.
4. Invokes an *Insurance* service to obtain a policy.
5. Invokes a *Shipping* service to organise the delivery of a computer.
6. Sends an invoice to a customer.
9. Sends a receipt to a customer.

![Diagram](image)

(a) With Parallel Constructs

(b) With Parallel Constructs converted to Sequences

*Figure 3.7: Interaction Protocol of a Computer Sales Service*

First, the parallel constructs of $P$ are converted to a sequence. The ITL axioms $b(cs_3,cs_4)$, $s(cs_3,cs_5)$ and $d(cs_4,cs_5)$ are converted to $cs_3,cs_4,cs_5$. The WS-$\pi$-calculus specification of the sequenced interaction protocol is given below and is graphically depicted in Figure 3.5b.

$$cs_1,cs_2, b(cs_3,cs_4), cs_3,cs_4,cs_5,cs_6,(cs_7+cs_8),cs_9,0$$

Then, the sequenced interaction protocol is compared against the WS-$ALUE$ description in Figure 3.6 using the *evaluation* function (in Algorithm 3.2). The pre-conditions are assigned...
\[ P = cs_1 \cdot cs_2 \cdot b(cs_3, cs_4) \cdot s(cs_3, cs_5) \cdot d(cs_4, cs_5) \cdot cs_6 \cdot (cs_7 + cs_8) \cdot cs_9 \cdot 0 \]

\[ cs_1 = \text{Customer}[\text{receive}](\exists \text{specification}.\text{Computer}, \exists \text{details}.\text{Customer}) \]
\[ \text{Add}(= \text{REQUESTED} \text{ status}.\text{Computer}), \]
\[ \text{Delete}(= \text{AWAITING REQUEST} \text{ status}.\text{Computer}) \]

\[ cs_2 = \text{Customer}[\text{receive}](\exists \text{specification}.\text{Computer}, \exists \text{details}.\text{Customer}) \]
\[ \text{Add}(= \text{IDENTIFIED} \text{ status}.\text{Customer}) \]

\[ cs_3 = \text{Insurance}[\text{invoke}](\exists \text{quoteRequest}.\text{Insurance}) \]
\[ \text{Add}(= \text{QUOTED} \text{ status}.\text{Insurance}) \]

\[ cs_4 = \text{Insurance}[\text{invoke}](\exists \text{policyRequest}.\text{Insurance}) \]
\[ \text{Add}(= \text{ONGOING} \text{ status}.\text{Insurance}), \]
\[ \text{Add}(= \text{INSURED} \text{ status}.\text{Computer}), \]
\[ \text{Delete}(= \text{QUOTED} \text{ status}.\text{Insurance}), \]
\[ \text{Delete}(= \text{REQUESTED} \text{ status}.\text{Computer}) \]

\[ cs_5 = \text{Shipping}[\text{invoke}](\exists \text{noteRequest}.\text{Consignment}) \]
\[ \text{Add}(= \text{CONFIRMED} \text{ pickupDate}.\text{Consignment}), \]
\[ \text{Add}(= \text{CONSIGNED} \text{ status}.\text{Computer}) \]
\[ \text{Delete}(= \text{INSURED} \text{ status}.\text{Computer}) \]

\[ cs_6 = \text{Customer}[\text{send}](\exists \text{invoice}.\text{Customer}) \]
\[ \text{Add}(= \text{INVOICED} \text{ status}.\text{Customer}) \]
\[ \text{Delete}(= \text{IDENTIFIED} \text{ status}.\text{Customer}) \]

\[ cs_7 = \text{Customer}[\text{receive}](\exists \text{details}.\text{CreditCard}) \]
\[ \text{Add}(= \text{PAYMENT RECEIVED} \text{ status}.\text{Customer}) \]
\[ \text{Delete}(= \text{INVOICED} \text{ status}.\text{Computer}) \]

\[ cs_8 = \text{Customer}[\text{receive}](\exists \text{details}.\text{CreditCard}) \]
\[ \text{Add}(= \text{PROCESSING PAYMENT} \text{ status}.\text{Customer}) \]
\[ \text{Delete}(= \text{INVOICED} \text{ status}.\text{Computer}) \]

\[ cs_9 = \text{Customer}[\text{send}](\exists \text{receipt}.\text{Customer}) \]
\[ \text{Add}(= \text{SHIPPED} \text{ status}.\text{Computer}) \]
\[ \text{Delete}(= \text{CONSIGNED} \text{ status}.\text{Computer}) \]

Figure 3.8: Sample WS-\pi-calculus Description
to $S$, the post-conditions are assigned to $VTS$ and the sequenced interaction protocol is assigned to $P$ of this function (line 1). The execution state $s$ is updated according to the effects of the interactions. The way in which the execution state of the *Computer Sales* service evolves as the effects of the relevant interactions are applied is depicted in Table 3.4. (Interaction $cs_x$ causes the execution state to transition from state $S_{x-1}$ to $S_x$. $S_0$ is the initial execution state, and $S_8$ and $S_9$ are terminal states). The updates are performed in an incremental manner for the sequence $cs_1.ccs_2.ccs_3.ccs_4.ccs_5.ccs_6$. At the choice construct $cs_7 + cs_8$ the first alternative path $cs_7.ccs_9.0$ is considered. As this path is not able to derive the state $VTS$, the second alternative path is $ccs_8.ccs_9.0$ is considered next. However, the second path is also not able to derive $VTS$ and there are no more alternatives to consider. Therefore, the technique returns state $S_9$. There is an error in at least one description since $S_9 \neq VTS$.

3.6 Related Work

The aim of this section is to analyse some of the existing functional and behavioural description frameworks that are relevant to our work and their limitations.

3.6.1 Functional Description Frameworks

A summary of the aspects described through current functional description frameworks is presented in Table 3.5.

All the frameworks, except WSDL [Christensen et al., 2001; Chinnici et al., 2007], LARKS [Sycara et al., 1999; 2002] and SCDL [Gao et al., 2002], describe two functional aspects of a service. SCDL and LARKS are very similar frameworks. They use input, outputs, in-constraints and out-constraints to describe the data transformations performed by services. In-constraints and out-constraints are used to apply constraints on the input and outputs. WSDL is an XML based formalism for describing the interfaces of a service. Each interface consists of a set of operations. An operation definition contains a set of input messages and output messages. The purpose of a service and the state transitions cannot be specified with WSDL, LARKS and SCDL.

WSDL-S [Akkiraju et al., 2005; Paolucci and Wagner, 2006] is an extension of WSDL, which includes descriptions of state transitions. These state transitions are described with pre-conditions and effects. WSML [de Bruijn et al., 2005; Keller et al., 2004] models a web service as a software entity that consumes and delivers certain objects. The consumed objects
### Chapte 3. Creating Accurate Functional and Behavioural Descriptions

<table>
<thead>
<tr>
<th>Interac-tion</th>
<th>Exec. State</th>
<th>State Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$S_0$</td>
<td>(= AWAITING_REQUEST status.Computer)</td>
</tr>
<tr>
<td></td>
<td>$S_1$</td>
<td>(= REQUESTED status.Computer)</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$S_2$</td>
<td>(= REQUESTED status.Computer)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\cup$ (= IDENTIFIED status.Customer)</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$S_3$</td>
<td>(= REQUESTED status.Computer) $\cup$ (= QUOTED status.Insurance)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\cup$ (= IDENTIFIED status.Customer)</td>
</tr>
<tr>
<td>$c_4$</td>
<td>$S_4$</td>
<td>(= INSURED status.Computer) $\cup$ (= ONGOING status.Insurance)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\cup$ (= IDENTIFIED status.Customer)</td>
</tr>
<tr>
<td>$c_5$</td>
<td>$S_5$</td>
<td>(= CONSIGNED status.Computer) $\cup$ (= ONGOING status.Insurance)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\cup$ (= IDENTIFIED status.Customer) $\cup$ (= CONFIRMED pickupDate.Consignment)</td>
</tr>
<tr>
<td>$c_6$</td>
<td>$S_6$</td>
<td>(= CONSIGNED status.Computer) $\cup$ (= ONGOING status.Insurance)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\cup$ (= IDENTIFIED status.Customer) $\cup$ (= CONFIRMED pickupDate.Consignment)</td>
</tr>
</tbody>
</table>

First Alternative

| $c_7$        | $S_7$       | (= CONSIGNED status.Computer) $\cup$ (= ONGOING status.Insurance) |
|              |             | $\cup$ (= PAYMENT_RECEIVED status.Customer) |
| $c_9$        | $S_9$       | (= SHIPPED status.Computer) $\cup$ (= ONGOING status.Insurance) |
|              |             | $\cup$ (= PAYMENT_RECEIVED status.Customer) $\cup$ (= CONFIRMED pickupDate.Consignment) |

Second Alternative

| $c_8$        | $S_8$       | (= CONSIGNED status.Computer) $\cup$ (= ONGOING status.Insurance) |
|              |             | $\cup$ (= PROCESSING_PAYMENT status.Customer) |
| $c_9$        | $S_9$       | (= SHIPPED status.Computer) $\cup$ (= ONGOING status.Insurance) |
|              |             | $\cup$ (= PROCESSING_PAYMENT status.Customer) $\cup$ (= CONFIRMED pickupDate.Consignment) |

Table 3.4: Evolving Execution State of a Sample Service

---

56 (October 2008)
are the inputs required by a service, and the context in which they can be used is described with a set of assumptions. The delivered objects are described with outputs and situations in which they can be delivered are described with a set of effects. OWL-S (formerly known as DAML-S) [Ankolekar et al., 2001; Martin et al., 2004; McIlraith and Martin, 2003] is a three layered service description framework. These layers are the service profile which describes the functional aspects of a service, the process model which describes the behavioural aspects and the service grounding which provides the information required to invoke a service such as protocols and authentication mechanisms. A service profile of OWL-S models the data transformations and the state transitions using inputs, outputs, pre-conditions and effects. Pre-conditions, post-conditions, assumptions and effects in WSDL-S, WSML and OWL-S are described with constraints. These three frameworks are not able to model the purpose of a service.

PILLAR [Elgedawy, 2003; Elgedawy et al., 2004a] describes a service from two separate views: the service active view (SAV) and the service passive view (SPV). The SPV describes the information required to execute a service, such as its interfaces and the protocols used to exchange information. The functional and behavioural aspects are described in the SAV. The $G^+$ goal model of the SAV is used to specify the functional aspects of a service. In this goal model, the purpose of a service is described using an operation and the state transitions are specified using a set of context conditions. These conditions are of two types: pre-conditions and post-conditions. They describe the initial execution state and the valid terminal states respectively. The data transformations cannot be represented in PILLAR.

<table>
<thead>
<tr>
<th>Framework</th>
<th>Purpose</th>
<th>State transition</th>
<th>Data transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PILLAR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>WSML</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OWL-S</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SCDL</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LARKS</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>WSDL</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>WSDL-S</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

*Table 3.5: Current Functional Description Frameworks*
None of the existing frameworks describe all three functional aspects (the purpose, data transformations and state transitions) of a service together.

### 3.6.2 Behavioural Description Frameworks

Behavioural description frameworks can be categorized as either specific or fuzzy frameworks. The former provides guidelines for describing interactions. The latter represents the behaviour of a service as a sequence of actions. Specific guidelines for describing attributes of interactions such as the type, parameters, the communicating entity and effects are not provided in fuzzy description frameworks. Parameters are used to identify the information exchanged in an interaction. There are four types of interactions (e.g. send, receive, invoke and service) and they are used to identify the direction of an interaction. The communicating entity is used to differentiate between multiple interactions that exchange the same parameter. For example, an Insurance service has two interactions that dispatches policy. Insurance: one to a Shipping service and another to a Computer Sales service. These two interactions can be differentiated easily if the descriptions of interactions (of the Insurance service) model the communicating entities. We use the term entity since the other party involved in an interaction can be either a service or a user. The effects of interactions are required to assess the way in which an execution state of a service evolves as it is executed. That means, the behavioural description cannot be compared against functional descriptions (using a transition-based approach) unless the effects of interactions are specified.

The process model of OWL-S [Ankolekar et al., 2001; Martin et al., 2004; McIlraith and Martin, 2003] and the scenarios of PILLAR [Elgedawy et al., 2004b; 2005; 2008] are examples of existing fuzzy frameworks. The former models ordering constraints that exist between interactions using the four basic language constructs (sequence, choice, parallel and iteration). However, it does not consider the time taken by the interactions when specifying the ordering constraints. This hinders the ability to represent the ordering constraint accurately. The latter only supports sequence constructs.

A summary of existing specific description frameworks is provided in Table 3.6.

WS-choreography [Burdett and Kavantzas, 2004] is a framework primarily used to define global communication models. A global communication model is a shared common behavioural description of two or more interacting services. The sequence in which the interactions take place between the interacting services are described in a global communication model. However, it can also be used to describe the interaction protocol of a single service.
In WS-choreography, each interacting entity is assigned a role. An interaction that takes place between two roles is performed via a channel. Hence, the channel variables clearly outline the entities involved in an interaction. All four types of interactions are supported in WS-choreography.

WSCL [Banerji et al., 2002] represents each interaction as an XML document and an interaction protocol is specified as a sequence of exchanged XML documents. It is primarily designed for describing an interaction protocol of a single service. WSCL descriptions are made of three element types; interactions, transitions and conversations. The interaction element specifies the type of interaction (send, receive, send-receive, receive-send or empty). Empty interactions denote the start and end of an interaction protocol. The transition elements are used to specify the ordering constraints. Each transition has a source interaction, a destination interaction and a condition which specifies an assertion that has to hold for the transition to occur. The conversation element specifies the list of interactions and transitions that make up an interaction protocol.

BPEL4WS [Foster et al., 2003; Shen et al., 2005] descriptions consist of four major sections: variables, partner links, fault handlers and process definition. A variable section defines
a message type. A partner link defines the interacting entities. A fault handler describes the activities that have to be performed to respond to failures that occur when executing a service. An actual interaction protocol is described in the process section. The ordering constraints that exist between the interactions are specified with the sequence, pick, switch and flow elements. They are similar to the sequence, iteration, choice and parallel constructs.

Like the fuzzy frameworks, none of the existing specific frameworks consider the time taken by the interactions when specifying the ordering constraints. Hence, they are not able to model the ordering constraints accurately. Also, the effects of the interactions could not be specified in any of them.

3.7 Discussion

Here we provide a brief discussion of existing techniques that verify various properties of web services. Techniques that detect errors in functional and behavioural descriptions of a service by comparing them against each other are of two types: transition-based and transformation-based. The former checks if the valid terminal state (post-conditions) of a service specified in a functional description can be derived from the initial execution state (initial execution state) by tracing through all possible executions (according to its interaction protocol). The latter compares the parameters of interactions against the inputs and outputs specified in a functional description.

The verification technique proposed in this chapter is a transition-based approach. To our knowledge this is the only approach that is specifically designed to detects errors in functional and behavioural descriptions. We are aware of two other verification techniques that could be used in similar circumstances [Elgedawy et al., 2004b; 2005; 2008; Woodman et al., 2004]. A brief description of these two techniques and their limitations are provided in the following.

The technique in [Woodman et al., 2004] verifies the ability of a composite service to produce required outputs from a set of inputs. This technique models interaction protocols with π-calculus and uses reduction rules to perform the verification [Sangiorgi and Walker, 2001]. Woodman et al.’s approach can be used as a transition-based technique. However, this technique can only be used if a protocol consists of send and receive interactions. It does not support invoke and service interactions. The aggregate matching technique introduced in [Elgedawy et al., 2004b; 2005; 2008] matches a sequence of web services with a Goal Achievement Procedure (GAP). A GAP models a sequence of actions that derives a set of
CHAPTER 3. CREATING ACCURATE FUNCTIONAL AND BEHAVIOURAL DESCRIPTIONS

post-conditions from a set of pre-conditions. That means this matching technique checks if a sequence of web services are able to derive the post-conditions of a GAP from its pre-conditions. Hence, it can be used as a transition-based approach by substituting the web services with the effects of interactions. However, this technique can only be used when the ordering constraints of an interaction protocol are specified with a sequence construct. GAPs are not able to model choice, parallel and iteration constructs.

In addition to the two approaches introduced in [Elgedawy et al., 2004b; 2005; 2008] and [Woodman et al., 2004], there are other techniques [Bultan et al., 2003; Foster et al., 2003] which are used for various verification tasks in web services. However, these cannot be used to detect errors in functional and behavioural descriptions of services under any circumstance because they are used for entirely unrelated verification tasks. A technique that checks whether an implementation of a composite service conforms to its design specifications is proposed in [Foster et al., 2003]. The specifications are described in UML using Message Sequence Charts (MSC) and the implementations are described in BPEL4WS. First, both a MSC and a BPEL4WS description are converted to Finite State Processes (FSPs). Then, the two FSPs are compared using a Labelled Transition System Analyzer [Magee, 1999].

The approach in [Bultan et al., 2003; 2006; Fu et al., 2005] determines the synchronisability of conversations and detects deadlocks in global communication models of composite services. Synchronisability ensures that the same set of conversations are generated for a composite service that supports both synchronous and asynchronous communication. Deadlocks are detected by comparing the temporal constraints in interaction protocols against those between conversations. Interaction protocols of constituent services and a global communication model are converted to BPEL4WS specifications and compared using WSAT [Fu et al., 2004a]. WSAT is a model checker that verifies BPEL4WS specifications. It converts BPEL4WS specifications to Promela\(^7\) and invokes a SPIN [Holzmann, 1997] model checker to perform the verification. Similar approaches that detect deadlocks and synchronisation conflicts in global communication models have been introduced in [Bao et al., 2006], [Berardi et al., 2003], [Beyer et al., 2005], [Deutsch et al., 2006], [Kang et al., 2007], [Koshkina and van Breugel, 2004] and [Yi and Kochut, 2005]. Detailed descriptions regarding these approaches are provided in Chapter 6.

\(^7\)Promela is the input language of SPIN.
CHAPTER 3. CREATING ACCURATE FUNCTIONAL AND BEHAVIOURAL DESCRIPTIONS

3.8 Summary

This chapter proposes two formalisms, WS-ALUE and WS-π-calculus, for creating functional and behavioural descriptions of web services, and a verification technique that detects errors in these descriptions. Only errors that are not equally reflected in both the WS-ALUE description and the WS-π-calculus description of a service are detected.

Service descriptions have to model all three functional aspects (the purpose, the state transitions and the data transformations) together. If not, a discovery technique may overlook a service, since it is difficult to match a request against a description that does not contain sufficient information. Current functional description frameworks only consider either one or two of the functional aspects (see Table 3.5). WS-ALUE extends the DL language ALUE with some new constructs, and is able to model all three functional aspects.

Behavioural descriptions model the interaction protocols of services. This is performed by specifying the interactions and the ordering constraints between them. Interactions of services are of four types: send, receive, invoke and service; and the first two types take place at a particular time point and the latter types take a time interval to complete. This temporal aspect of interactions has to be considered when specifying the ordering constraints. If not, the ordering constraints between concurrent interactions cannot be modelled completely. Additionally, the effects caused by executing an interaction have to be specified in behavioural descriptions. If not, a behavioural description cannot be compared against a functional description using a transition-based approach. Performing such a comparison is important to detect certain errors in functional and behavioural descriptions of services. Current behavioural description frameworks model the ordering constraints between interactions using four language constructs: sequence, choice, parallel and iteration. They do not consider the time taken by the interactions and are not able to model the ordering constraints accurately. Furthermore, they do not specify the effects of interactions (see Table 3.6). WS-π-calculus extends π-calculus with service and invoke interactions, ITL axioms and WS-ALUE constructs. The ITL axioms enable the temporal aspects of the interactions to be considered when specifying the ordering constraints. WS-ALUE constructs are used to specify the effects of interactions.

Techniques that detect errors in functional and behavioural descriptions by comparing them against each other are of two types: transition-based and transformation-based. The former checks if the post-conditions of service can be derived from its pre-conditions accord-
ing to the effects caused by its interactions. The latter compares the parameters used in interactions against the inputs and outputs of a functional description. To our knowledge, there is no technique that verifies functional and behavioural descriptions for any specification errors. The technique proposed in this chapter is a transition-based approach. The proposed technique traverses through an interaction protocol and modifies the execution state of a service according to the effects of the interactions to perform the above verification. It is developed by extending a pre-order depth-first search algorithm. A case study showing how errors are detected in a WS-ALUE description of a Computer Sales service using the proposed techniques is provided. A comprehensive analysis which compares the proposed technique against existing verification techniques [Bultan et al., 2006; Elgedawy et al., 2008; Foster et al., 2003; Woodman et al., 2004] and illustrates their limitations is also given.
Chapter 4

S-Match: Matching Strictly Dependent Global Constraints

Web service discovery requires matching techniques for comparing and selecting service descriptions based on user constraints. Semantic-based approaches achieve higher recall than other approaches (such as syntax-based approaches), because they employ ontological reasoning mechanisms to match syntactically heterogeneous descriptions. However, existing semantic-based approaches are not scalable as they perform an exhaustive search to locate composite services that conform to global constraints. Such constraints simultaneously restrict the values assigned to attributes of multiple constituent services. This chapter proposes a semantic-based matching technique that locates composite services that conform to strictly dependent global constraints [Gooneratne et al., 2007b]. A given constraint is strictly dependent if it can be solved once a value is assigned to one attribute. The proposed technique relates attributes of services to a common attribute to ensure that they have the same scope. This enables the assigned values to be compared and evaluated against a given global constraint. Conforming composite services are located in polynomial time with a three-dimensional data structure that indexes services based on their types, attributes and values they assign to these attributes.

4.1 Motivation

Service discovery is a mechanism enabling services to be located based on functional requirements [Benatallah et al., 2005; Elgedawy et al., 2004a; Mokhtar et al., 2006], non-functional
requirements (e.g. quality of service [Zeng et al., 2003; 2004] and geo-spatial aspects [Fileto et al., 2003; Kuhn, 2005; Tryfona and Pfoser, 2005]) or all [Sycara et al., 1999; 2002]. These mechanisms employ matching techniques to compare user requests against service descriptions and select appropriate services. Matching techniques can be categorised as either syntactic-based or semantic-based. The former analyses a service description as a set of strings or parameter definitions and perform either syntax driven string matchings [Hosoya and Pierce, 2001; Li and Danzig, 1997] or schema mappings [Rahm and Bernstein, 2001; Wang and Strouila, 2003; Zaremski and Wing, 1995; 1997]. The latter utilises ontological relationships to perform mappings between the terms in service descriptions and user requests [Elgedawy et al., 2004a; Gao et al., 2002; Li and Horrocks, 2004]. Unlike syntactic-based approaches, semantic-based techniques are able to match descriptions created with different terminologies and achieve higher recall [Elgedawy et al., 2004a].

Services retrieved by discovery techniques are of two types: simple and composite [Medjahed et al., 2003; Medjahed and Bouguettaya, 2005]. A simple service is one atomic service, whereas a composite service is an aggregation of multiple “heterogeneous” services.

This chapter proposes a semantic-based matching technique that locates composite services based on functional requirements. Let us extend Scenario 1.1 (which describes a situation where a user purchases a computer using online services), to illustrate the issues that have to be addressed by such a technique. The user indicates that the shipper should be able to pickup the purchased computer from the seller’s dispatch location. The insurance for the computer should commence from the time at which it is picked up. The time taken to assemble the computer, get the insurance approved and deliver should be less than 7 days. The computer must not be insured until it is assembled (so that the insured period can be maximised); and it should not be picked up by the shipper until it is insured. The composite service shown in Figure 1.2 has to be formed by locating, co-ordinating and collaborating the indicated constituent services, and executed to satisfy this request.

A service discovery technique locating the composite service in Figure 1.2 would require accurate specifications of both service descriptions and user requests. Efforts to create accurate service descriptions include frameworks such as PILLAR [Elgedawy, 2003; Elgedawy et al., 2004a], WSMO [de Bruijn et al., 2006; Keller et al., 2004], OWL-S [Ankolekar et al., 2001; Martin et al., 2004; McIlraith et al., 2001], SCDL [Gao et al., 2002] and LARKS [Sycara et al., 1999; 2002] and WS-ALUE. Constraints are included in user requests to accurately describe the services that need to be located [Ankolekar et al., 2001; Elgedawy, 2003] and
these can be categorised as either local or global constraints. Local constraints restrict the values of a particular attribute of a single service. Global constraints simultaneously restrict the values of two or more attributes of multiple constituent services. For example, \(\text{type.Computer} = \text{MACINTOSH}\) is a local constraint, whereas \(\text{productionTime.Computer} + \text{approvalPhase.Insurance} + \text{timeTaken.Delivery} < 7_{\text{DAYS}}\) and \(\text{location.Dispatch} = \text{location.Pickup} \in \text{validRegion.Insurance}\) are global constraints.

Locating a composite service that conforms to a global constraints is known to be NP-hard [Zeng et al., 2003; 2004]. For example, if a given global constraint restricts \(q\) attributes, and available service descriptions assign \(p\) values to each attribute, then a service discovery technique that locates a conforming service may have to consider \(p^q\) combinations of values. As a consequence most of the existing matching techniques (for locating composite services) do not consider global constraints [Agarwal et al., 2005; Akkiraju et al., 2006a;b; Lin et al., 2006; McIlraith and Son, 2002; Paolucci et al., 2003; Rao and Su, 2004]. Nonetheless, there are some that consider them and use integer programming solutions focusing on local optimisations [Zeng et al., 2003; 2004] and AI planners [Sirin and Parsia, 2004; Wu et al., 2003a] to efficiently locate conforming composite services. However, all the techniques of the latter type are syntactic-based approaches. None of the current semantic-based composite matching approaches consider global constraints.

This chapter proposes a semantic-based matching technique which locates services that conform to global constraints. Since locating services that conform to global constraints is NP-hard, our approach considers a restricted class called strictly dependent global constraints. Services that conform to such constraints can be located in polynomial time. A global constraint is strictly dependent if the values that should be assigned to all the remaining restricted attributes can be uniquely determined once a value is assigned to one. \(\text{location.Dispatch} = \text{location.Pickup} \in \text{validRegion.Insurance}\) and \(\text{date.Dispatch} < 1_{\text{DAY}}\) \(\text{date.Pickup} = \text{commencementDate.Insurance}\)\(^4\) are examples of strictly dependent global constraints. In the first constraint, if the value \(\text{VICTORIA}\) has been assigned to \(\text{location.Dispatch}\), then the same value has to be assigned to \(\text{location.Pickup}\) and this should be included in the set of values assigned to \(\text{validRegion.Insurance}\). In the second one, if the value \(1_{\text{ST\_MARCH\_2007}}\) is assigned to \(\text{date.Dispatch}\) then \(2_{\text{ND\_MARCH\_2007}}\) has to be assigned

\(^4\)Standard knowledge representation formalisms would represent \(\text{date.Dispatch} < 1_{\text{DAY}}\) \(\text{date.Pickup}\) as \((\text{date.Dispatch} + 1_{\text{DAY}}) = \text{date.Pickup}\). However, the latter representation is not used since it conflicts with the notations that model certain non-strictly dependent constraints in Chapter 5.
to \textit{date.Pickup} and \textit{commencementDate.Insurance}.

When matching service descriptions, it may not be possible to compare the values assigned to attributes if their scopes are different. For example, in a situation where \textit{VICTORIA} is assigned to \textit{location.Dispatch} and \textit{MELBOURNE} is assigned to \textit{location.Pickup}, a comparison which checks if \textit{location.Dispatch} = \textit{location.Pickup} would determine that the values are not equal, even though they are semantically similar. The proposed approach ensures that the assigned values have similar scopes by relating them to a common attribute using ontological relationships. This process is referred to as attribute leveling. A naive approach is not sufficient to locate conforming composite services, since it would take exponential time to identify all the combinations of services that:

(i) consist of restricted attributes which level and

(ii) assign conforming values to these attributes.

Hence, the proposed approach uses a three-dimensional data structure called an attribute leveling cube to perform this task in polynomial time. An attribute leveling cube is generated by indexing services based on their type, their restricted attributes and the values that they assign to those attributes. Once a cube is generated, the details associated with attributes which do not level are eliminated. Then, combinations of values that conform to a given strictly dependent global constraint are determined and placed in value vectors. Finally, services which assign these values to restricted attributes are extracted from the cube to form conforming composite services.

Simulation experiments are used to compare the performance and the recall levels achieved by implementations of the proposed technique, a relevant semantic-based matching technique [Elgedawy et al., 2004a; Wu and Wu, 2005] and a syntactic-based composite matching technique [Rao and Su, 2004; Wu et al., 2003a;b]. Also, a case study which evaluates the three techniques using a sample scenario is provided. Our results indicate that the proposed technique is more scalable than the semantic-based technique and achieves a higher recall level than the syntactic-based technique.

The rest of the chapter is organised as follows. Section 4.2 provides a brief analysis of the way in which constraints are categorised by current techniques that deal with Constraint Satisfaction Problems. Section 4.3 provides concise guidelines on how requests for composite services should be specified. Section 4.4 describes the proposed composite matching technique. Section 4.5 provides a brief discussion about the advantages and drawbacks
of the proposed approach. A theoretical analysis of the proposed approach for soundness and completeness is provided in Section 4.6. Details of the case study and the simulation experiments are provided in Section 4.7. Section 4.8 reviews existing composite matching techniques. Finally, we summarise the contributions of this chapter in Section 4.9.

4.2 Constraint Satisfaction

Locating a composite service that conforms to a global constraint is an extension of a typical constraints satisfaction problem (CSP) [Zeng et al., 2003; 2004]. The way in which constraints are categorised by techniques that deal with CSPs is described here. This categorisation is important to clearly outline the type of constraint considered by the proposed matching technique.

A CSP is a 3-tuple \([V, L, C]\), where \(V\) is a set of variables \(\{v_m, \ldots, v_n\}\), \(L\) is a list of value sets \([L_m, \ldots, L_n]\), and \(C\) is a set of constraints \(\{c_x, \ldots, c_y\}\). Each \(L_i \in L\) contains a set of values that can be assigned to a variable \(v_i\). A constraint is a relation between multiple variables, and such relations restrict the possible value assignments (e.g. \(v_a < v_b\), \(v_a + v_b \geq v_c\), \(v_c \in \{v_d, \ldots, v_e\}\)). A tuple of values \([l_m, \ldots, l_n]\) where each \(l_i\) is assigned to a variable \(v_i\) is a solution if it conforms to all the constraints in \(C\). The objective of a technique that solves a CSP is to find such solutions. However, finding such solutions is known to be NP-Hard [Dechter, 1992; Freuder, 1982; Kirousis, 1993; Nareyek, 2001; Voudouris and Tsang, 2001]. For example, if \(|V| = n\) and each \(|L_i| = m\) (of the above CSP), then \(m^n\) combination of values have to be considered to locate all the solutions.

The primal graph [Dechter, 1992] of a CSP is a directed graph where the nodes correspond to the variables and an edge exists between two nodes if there is a constraint between the corresponding variables. CSPs are categorised into two types as either cyclic or acyclic [Dechter, 1992]. A CSP is cyclic if its primal graph contains a loop [Dechter, 1992]. The matching techniques proposed in this thesis assume that considered global constraints are extensions of acyclic CSPs.

Constraints can be categorised into different types based on various aspects such as the relationships between variables, the techniques used to identify solutions and complexity. An overview of the way in which constraints are categorised is provided in Figure 4.1.

Constraints can be categorised as either strictly dependent or independent based on the ability to find complete solutions in polynomial time [Dechter, 1992; Freuder, 1982; 1985; Mackworth, 1977; Mackworth and Freuder, 1985]. A constraint is strictly dependent
if the values that should be assigned to all the variables are uniquely determined when an assignment is made to one variable. Any constraint that is not strictly dependent is independent. For example, $v_a < 5$ $v_b = v_c$ and $v_a \in v_d$ are strictly dependent constraints whereas $v_x \leq v_y$ is independent. If 10 is assigned to $v_a$, then 15 has to be assigned to $v_b$ and $v_c$, and any set assigned to $v_e$ has to include 10 (i.e. the set can be identified by checking for just one value). When 10 is assigned to $v_x$, 15 or any value which is greater can be assigned $v_y$ (i.e. the value that should be assigned to $v_y$ cannot be uniquely determined). All available solutions can be found for a strictly dependent constraint in polynomial time whereas such a property cannot be associated with an independent constraint [Cooper et al., 1994; Freuder, 1982; 1985]³.

Independent constraints are divided into two types based on the number of variables involved as either binary or non-binary [Ladkin and Maddux, 1994]. Non-binary constraints can be classified into two types based on whether a technique can be used to optimise the

---

²$v_a < 5$ $v_b$ is not specified as $v_a < (v_b + 5)$ because it conflicts with the notations that model locally optimisable non-binary independent constraints in Chapter 5.

³Note that it is assumed that bounded sets of values can be assigned to each variable.
process that locates solutions. Optimisable non-binary constraints are of two types: locally optimisable and incremental. This categorisation is based on the types of functions used by the optimisation techniques. A constraint is locally optimisable if a function that selects the most optimised value can be defined for each of its variables [Karloff, 1991; Zeng et al., 2003; 2004]. For example, consider a constraint $a + b + c < 10$, where $\{3, 8, 9\}$, $\{4, 1, 2\}$ and $\{7, 4, 9\}$ are the values that can be assigned to $a$, $b$ and $c$ respectively. This constraint is locally optimisable if a function that returns the minimum value is associated with each variable. These functions that select suitable values are referred to as “Objective Functions”.

Locally optimisable constraints are categorised into two types as either linear or non-linear based on the complexity of the objective functions [Zeng et al., 2003; 2004]. A constraint is incremental if a neighbourhood function that maps one candidate solution to another set of candidate solutions, and an evaluation function that returns a numeric value for each candidate solution can be defined [Nareyek, 2001; Voudouris and Tsang, 2001]. Techniques that solve incremental constraints using these functions start by randomly generating a candidate solution. If the candidate solution does not conform to the given constraint it moves to another solution. This is performed by using a neighbourhood function to map the candidate solution to a set and selecting one based on the values returned by an evaluation function. The search terminates when a solution is located or a candidate solution which is better than the current one (according to the evaluation function) cannot be located. Let us assume that a constraint $a + b + c = 13$ and the values $\{3, 5, 7\}$, $\{2, 8, 9\}$ and $\{4, 6, 1\}$ that can be assigned to them are given. The evaluation function (used to solve this constraint) returns the difference between the sum of values of a candidate solution and 13. The minimum value returned by this function is 0 and indicates that a candidate solution is an actual solution. The neighbourhood function generates a set of candidate solutions from a given one by changing one of the assigned values. The way in which a solution for $a + b + c = 13$ is found using these functions is shown in Table 4.1.

Techniques that solve locally optimisable constraints using linear objective functions and those that deal with incremental constraints using the functions described above are able to complete their execution in polynomial time [Karloff, 1991; Nareyek, 2001; Voudouris and Tsang, 2001; Zeng et al., 2003; 2004]. However, unlike those of the former type, the techniques of the latter type are not complete. The incremental search for a solution could lead to a dead-end. For example, a solution cannot be found if $\{3, 8, 4\}$ is selected instead of $\{5, 2, 4\}$ from the first set of candidate solutions generated by the neighbourhood function in

70
CHAPTER 4. S-MATCH: MATCHING STRICTLY DEPENDENT GLOBAL CONSTRAINTS

<table>
<thead>
<tr>
<th>Selected Candidate Solution</th>
<th>Candidate Solutions returned by the Neighbourhood function</th>
<th>Value assigned by the Evaluation Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3, 2, 4]</td>
<td>[5, 2, 4]</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>[3, 8, 4]</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>[3, 2, 1]</td>
<td>7</td>
</tr>
<tr>
<td>[5, 2, 4]</td>
<td>[7, 2, 4]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>[5, 8, 4]</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>[5, 2, 1]</td>
<td>5</td>
</tr>
<tr>
<td>[7, 2, 4]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 4.1: Solving an Incrementally Optimisable Non-binary Constraint*

Table 4.1. A solution which assigns either 8 or 9 to b cannot be formed with the given values. To overcome this, heuristic-based approaches that guide neighbourhood functions have been proposed [Nareyek, 2001; Voudouris and Tsang, 2001].

Techniques solving unoptimisable and binary constraints have an exponential worst-case time complexity. They either consider each variable individually (local consistency/local optimum) [Minton et al., 1990] or incorporate heuristic based approaches. However, like the basic approaches that deal with incremental constraints, those that only consider a local optimum to solve binary constraints are not complete [Minton et al., 1990].

4.3 User Requests

Concise guidelines for creating requests for composite services are provided here. The proposed matching technique assumes that a given request is structured according the guidelines given here. A request consists of a composite service template and a constraint model. The former specifies the types of services that need to be collaborated to form a composite service. It is defined as a collection of service types. A service type is described with the triplet [O, C, R], where O is an operation, C is an affected concept and R is a role.

**Definition 4.1 (Composite Service Template)** Given a composite service cs which consists of the constituent services s_x, s_y, where each s_i is of type S_i, the collection of service types S_x, ..., S_y form the composite service template CST of cs.
A description of the composite service template of the service given in Figure 1.2 follows.

\{[Sales()], Computer, Sales_Assistant],
 [Insure()], Computer, Sales_Representative],
 [Ship()], Computer, Shipping_Agent] \}

Constraints are included in a request to provide an accurate description of the required services. For example, the constraint “type.Computer = Macintosh” (which is applied on services that sell computers) states that only those that sell macintosh computers are required. The constraint model described here contains a strictly dependent global constraint.

**Definition 4.2 (Strictly Dependent Global Constraint)** Let gc be a global constraint restricting the values assigned to the attributes \(a_x, \ldots, a_y\) of services of types \(S_x, \ldots, S_y\), where each \(a_i \in \{a_x, \ldots, a_y\}\) is an attribute used to describe services of type \(S_i \in \{S_x, \ldots, S_y\}\). gc is strictly dependent if the values \(v_{x+1}, \ldots, v_y\) that should be assigned to \(a_{x+1}, \ldots, a_y\) can be uniquely determined once a value \(v_x\) is assigned to \(a_x\).

Pragmatically, a set of local constraints, strictly dependent global constraints and independent global constraints\(^4\) would be used to accurately specify a request. However, the technique proposed in this chapter only locates services that conform to strictly dependent global constraints. Therefore, a strictly dependent global constraint is included in a user request. The following is an outline of how a request for a composite service should be structured.

\[
\begin{align*}
\text{User Request} & : \text{Composite Service Template,} \\
& \text{Strictly Dependent Global Constraint} \\
\text{Composite Service Template} & : \{\text{Service Type}\} \\
\text{Strictly Dependent Global Constraint} & : \{\text{Binary Attribute Comparison}, \text{Type}\} \\
\text{Binary Attribute Comparison} & : \text{Attribute, Comparison Operator, Attribute} \\
\text{Type} & : \text{PRE} \mid \text{POST} \mid \text{ANY} \\
\end{align*}
\]

A strictly dependent global constraint is described with a type and a set of binary attribute comparisons. The type explicitly specifies whether a constraint restricts attributes that describe either pre-conditions or post-conditions. A binary attribute comparison describes a relationship that must exist between values assigned to two attributes. It is specified with a pair of attributes and a comparison operator. A comparison operator can be

\(^4\)Global constraints which are not strictly dependent.
either of $=, <_{n}, >_{n}$ or $\in$, where $<_{n}$ and $>_{n}$ mean exactly $n$ lesser and exactly $n$ greater respectively. A non-empty set of these binary attribute comparisons is required to describe a strictly dependent global constraint. For example, the strictly dependent global constraint $\text{location.Dispatch} = \text{location.Pickup} \in \text{validRegion.Insurance}$ would be specified in a user request as follows.

$$\{\text{location.Dispatch} = \text{location.Pickup}, \text{location.Pickup} \in \text{validRegion.Insurance}\}, \text{PRE}$$

A binary attribute comparison describing a strictly dependent global constraint cannot include comparison operators such as $<, >, \leq, \geq, \neq, \subset, \subseteq \text{ and } /$. Once a value is assigned to one attribute, the value that should be assigned to the other attribute cannot be uniquely determined with these operators. For example, in $\text{date.Dispatch} < \text{date.Pickup}$, once the date $1\text{ST}\_\text{MAY}\_2007$ is assigned to $\text{date.Dispatch}$, any date that is after $1\text{ST}\_\text{MAY}\_2007$ can be assigned to $\text{date.Pickup}$. Therefore, a global constraint described with such a binary attribute comparison would be independent.

### 4.4 S-Match

A matching approach to locate composite services, is described in this section. This approach requires: (i) WS-ALUE service descriptions, (ii) user requests structured according to the guidelines given in Section 4.3, and (iii) the terms in the descriptions (service descriptions and user requests) to be defined in an ontology that is structured according to the Meta-Ontology [Elgedawy, 2003; Elgedawy et al., 2004a]. There are two main phases in this approach: candidate acquisition and composite service acquisition. The first phase locates services of the types that are included in a composite service template. The second phase identifies tuples of services that conform to a strictly dependent global constraint. These two phases are described in the following.

**Candidate Acquisition**

This phase describes a way of locating candidate services. A candidate service is a service of a particular type, and the proposed approach locates such services for all the types in a composite service template. By locating these services, it ensures that the constituent services included in a located composite service are of appropriate types. The following is a formal definition of a candidate service.
Definition 4.3 (Candidate Service) Let $S_m$ be a service type and $s_n$ a service, where $S_m$ and the purpose$^5$ of $s_n$ are described with the operations $o_m$ and $o_n$, affected concepts $c_m$ and $c_n$, and roles $r_m$ and $r_n$ respectively. Service $s_n$ is a candidate of type $S_m$ if

1. $o_m$ can be substituted with $o_n$,
2. $c_m$ can be substituted with $c_n$ and
3. $r_m$ can be substituted with $r_n$.

For example, the service Computer Sales-I in Figure 4.2 is a candidate of type Computer Sales if Sales(), Laptop and Telemarketer can substitute Sales(), Computer and Sales_Assistant respectively.

![Figure 4.2: Candidate Service](image)

Algorithm 4.1 shows how candidate services are located. This algorithm requires the available services $W$ and a composite service template which consists of the service types $S_1, \ldots, S_n$. For each service type, it iterates through the services in $W$ (line 6) and identifies those that are candidates (line 11). Once it is determined that a service $s_i$ is a candidate of type $S_j$, it is placed in the corresponding candidates list $candidates(S_j)$ (line 11). Note that a candidates list $candidates(S_i)$ is generated for each service type $S_i$ in $\{S_1, \ldots, S_n\}$. A set of candidates lists $Candidates$, where $Candidates = \{candidates(S_1), \ldots, candidates(S_n)\}$ is returned at the end (line 14). The functions used by this algorithm and details returned by them are as follows.

- $operation(X)$: Operation used to describe the purpose of a service or a service type (lines 3 and 7).

$^5$The purpose of a service models the performed transaction (see Chapter 3).
• affectedConcept(X): Affected concept used to describe the purpose of a service or a service type (lines 4 and 8).

• role(X): Role used to describe the purpose of a service or a service type (lines 5 and 9).

• substitutes_o(o_i): Set of operations substituted with an operation o_i (line 10).

• substitutes_c(c_i): Set of concepts substituted with a concept c_i (line 10).

• substitutes_r(r_i): Set of roles substituted with a role r_i (line 10).

1 Candidates : candidateAcquisition({S_1, \ldots, S_n}, W)
2 for each S_i in {S_1, \ldots, S_n} do
3 \hspace{1em} o_m \leftarrow operation(S_i);
4 \hspace{1em} c_m \leftarrow affectedConcept(S_i);
5 \hspace{1em} r_m \leftarrow role(S_i);
6 \hspace{1em} for each s_j in W do
7 \hspace{2em} o_n \leftarrow operation(s_j);
8 \hspace{2em} c_n \leftarrow affectedConcept(s_j);
9 \hspace{2em} r_n \leftarrow role(s_j);
10 \hspace{2em} if (o_m \in substitutes_o(o_n)) and (c_m \in substitutes_c(c_n)) and
11 \hspace{3em} (r_m \in substitutes_r(r_n)) then
12 \hspace{4em} candidates(S_i) \leftarrow candidates(S_i) \cup s_j;
13 \hspace{2em} end
14 \hspace{1em} end
15 return \{candidates(S_1), \ldots, candidates(S_n)\};

Algorithm 4.1: Locating Candidate Services

Algorithm 4.1 has a polynomial time complexity (mn, where m is the number of types in a given composite service template and n is the number of available services).

4.4.1 Composite Service Acquisition

This phase locates tuples of candidate services that form conforming composite services and it is divided into three steps.
1. First, the restricted attributes of the candidate services are identified. An attribute (used to specify a pre-condition or a post-condition of a service) is restricted, if the values assigned to it are restricted by a user constraint\(^6\). Such attributes need to be identified as the values assigned to them need to be checked to determine whether a service conforms to a given constraint. Since this technique is semantic-based, an attribute of a service is considered as a restricted attribute if it is semantically related to one that is used to specify a strictly dependent global constraint. Let us consider the constraint \(\text{location.Dispatch} = \text{location.Pickup} \in \text{validRegion.Insurance}\) described in Section 4.3. This restricts the attributes \(\text{location.Dispatch} \), \(\text{location.Pickup} \) and \(\text{validRegion.Insurance} \) of services of types \([\text{Sales()}], \text{Computer}, \text{Sales_Assistant}]\), \([\text{Ship()}], \text{Computer}, \text{Shipping_Agent}]\) and \([\text{Insure()}], \text{Computer}, \text{Sales_Representative}]\). Let us assume that \(\text{ComputerSales-I} \), which is a service of type \([\text{Sales()}], \text{Computer}, \text{Sales_Assistant}]\) is available, and that \(\text{location.Dispatch} \) can be substituted with \(\text{pickUpLocation.Item} \) according to the ontological relationships. The description of \(\text{ComputerSales-I} \) includes the attribute \(\text{pickUpLocation.Item} \), but not \(\text{location.Dispatch} \). In such a situation, \(\text{location.Dispatch} \) can be substituted with \(\text{pickUpLocation.Item} \) and it can be determined that \(\text{pickUpLocation.Item} \) is the restricted attribute of \(\text{ComputerSales-I} \). A formal definition of a restricted attribute is given below.

**Definition 4.4 (Restricted Attribute)** Let \(gc\) be a strictly dependent global constraint that restricts the attributes \(\{a_x, \ldots, a_y\}\) of services of types \(\{S_x, \ldots, S_y\}\). An attribute \(a'_i\) of a candidate service \(s_i\) of type \(S_i\) is a restricted attribute if \(a'_i\) can be semantically related to \(a_i\) where \(S_i \in \{S_x, \ldots, S_y\}\) and \(a_i \in \{a_x, \ldots, a_y\}\).

2. In the second step, tuples of candidate services consisting of restricted attributes that semantically relate to a common attribute are identified. Let us assume that \(\text{pickUpLocation.Item} \), \(\text{collectionPoint.Cargo} \) and \(\text{region.Coverage} \) are restricted attributes of a tuple of services. These attributes have different scopes. The scope of \(\text{pickUpLocation.Item} \) is a city, \(\text{collectionPoint.Cargo} \) is an intercontinental region and \(\text{region.Coverage} \) is a country. In such a situation, the values assigned to the three attributes cannot be compared since their scopes are different. For example, the values cannot be compared if \(\text{Melbourne}, \text{Oceania}\) and \(\text{Australia}\) are assigned to the three attributes. However, if the

\(^6\)A user constraint has a type, and a condition of a service which describe a restricted attribute has to be of the same type.
three attributes can be semantically related to name.State, the values that are assigned to them can be mapped to those in the scope of name.State. Then, the mapped values can be compared since they have the same scope. This process of semantically relating restricted attributes to a common attribute is referred to as attribute leveling.

**Definition 4.5 (Leveled Attributes)** The restricted attributes \{a_x, \ldots, a_y\} of a tuple of services \{s_x, \ldots, s_y\}, where each \(a_i \in \{a_x, \ldots, a_y\}\) is from \(s_i \in \{s_x, \ldots, s_y\}\), level if

\[
\exists a_i \in \text{semanticallyRelated}(a_x) \land \ldots \land a_i \in \text{semanticallyRelated}(a_y)
\]

3. In the final step, the values (assigned to each leveled tuple of restricted attributes) are evaluated to determine whether they conform to the given constraint. If they do, the corresponding candidate services (from which the attributes were extracted) are combined to form a conforming composite service. A formal definition of a conforming composite service follows.

**Definition 4.6 (Conforming Composite Service)** Let gc be a strictly dependent global constraint specified with a set of binary attribute comparisons \(R\), and \(a_x, \ldots, a_y\) be a tuple of leveled restricted attributes of the services \(s_x, \ldots, s_y\), where each \(a_i \in \{a_x, \ldots, a_y\}\) is an attribute of \(s_i \in \{s_x, \ldots, s_y\}\). The services \(s_x, \ldots, s_y\) form a conforming composite service if a tuple of values \(v_x, \ldots, v_y\) can be assigned to \(a_x, \ldots, a_y\), in the following way: (i) each \(v_i\) in \(\{v_x, \ldots, v_y\}\) is assigned to \(a_i\) in \(\{a_x, \ldots, a_y\}\), and (ii) \(v_x, \ldots, v_y\) conform to each comparison \(r_j\) in \(R\).

However, it is difficult to locate such composite services with a “naive” technique. If a given constraint restricts attributes of \(p\) service types, each type has \(q\) candidate services, and the restricted attribute of each service is semantically related to \(n\) attributes, \((p^q)^n\) tuples of attributes might have to be considered to identify those that level. A similar problem occurs if an exhaustive search is performed by a process which evaluates the values assigned to a tuple of restricted attributes. A condition (a pre-condition or a post-condition) in a service description specifies the set of values that can be assigned to a certain attribute of the service. Given a tuple of \(n\) attributes, if \(m\) values can be assigned to each attribute according to a service description, then \(m^n\) combinations of values may have to be considered to determine if the services conform to a constraint.
The complexity of an algorithm that identifies leveled attributes could be reduced, if services are indexed based on their restricted attributes. Let us consider the above scenario where the restricted attributes `pickupLocation.Item`, `collectionPoint.Cargo` and `region.Coverage` of the services `ComputerSales-I`, `Shipping-I` and `Insurance-I` are semantically related to `name.State`. A restricted attribute based indexing mechanism would store the services in a two-dimensional array as in Figure 4.3.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>ComputerSales-I</th>
<th>Shipping-I</th>
<th>Insurance-I</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pickupLocation.Item</code></td>
<td>[Sales(), Computer, Sales_Assistant]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>collectionPoint.Cargo</code></td>
<td></td>
<td>[Ship(), Computer, Shipping Agent]</td>
<td></td>
</tr>
<tr>
<td><code>region.Coverage</code></td>
<td>[ComputerSales-I]</td>
<td>[Shipping-I]</td>
<td>Insurance-I</td>
</tr>
<tr>
<td><code>name.State</code></td>
<td>[ComputerSales-I]</td>
<td>[Shipping-I]</td>
<td>Insurance-I</td>
</tr>
</tbody>
</table>

*Figure 4.3: Restricted Attribute-based Service Indexing*

Such an array can be used to easily determine (i) whether the restricted attributes of a tuple of services can be leveled and (ii) the attributes at which they level. For example, in the above two-dimensional array, the rows with empty slots can be eliminated to determine that the restricted attributes of `ComputerSales-I`, `Shipping-I` and `Insurance-I` level at `name.State`.

On the other hand, the process (that evaluates the values assigned to the restricted attributes) does not have to perform an exhaustive search since the considered global constraints are strictly dependent. Given a constraint that restricts attributes $a_x, \ldots, a_y$ of services of types $S_x, \ldots, S_y$, the values that have to be assigned to $a_{x+1}, \ldots, a_y$ can be established for each value that is assigned to $a_x$ by services of type $S_x$. Then, these values can be stored in a list of vectors. We will refer to these vectors as Value Vectors. Finally, the services that assign the values in the value vectors to their restricted attributes can be located and combined to form conforming composite services. Let us consider a scenario which includes the constraint `pickupLocation.Item = collectionPoint.Cargo ∈ region.Coverage`, where `pickupLocation.Item`, `collectionPoint.Cargo` and `region.Coverage` are semantically related to `name.State`. Also, assume that the services `ComputerSales-I`, `Shipping-I` and `Insurance-I` assign the values \{QLD, VIC, NT, ACT\}, \{NSW, NZ, SA, VIC\} and \{NSW, SA, TAS, VIC\} to the three attributes. First, this algorithm would generate the value vectors [QLD, QLD, QLD], [VIC, VIC, VIC], [NT, NT, NT] and [ACT, ACT, ACT]. Then, for each value vector,
it would iterate through the lists of values that are assigned to the restricted attributes. Once the value vector \([\text{VIC}, \text{VIC}, \text{VIC}]\) is considered, this approach would retrieve the services \text{ComputerSales-I}, \text{Shipping-I} and \text{Insurance-I} since all three of them assign the value \text{VIC} to their restricted attribute. Finally, these services would be combined to form the service tuple \([\text{ComputerSales-I}, \text{Shipping-I}, \text{Insurance-I}]\), which is a conforming composite service. Therefore, in a situation where \(m\) values can be assigned to each attribute \(a_i\) in a tuple \(a_x, \ldots, a_y\) and \(|a_x, \ldots, a_y| = n\), the complexity of the process that evaluates the values can be reduced to \(m^2 n\) (from \(m^n\)).

The algorithm described earlier can be optimised further, if the services are indexed based on the values they assign to their restricted attributes. By doing so, it can be directly identified whether a service assigns a certain value to its restricted attribute (i.e. there is no need to iterate through a list of assigned values). The two dimensional array that would be generated if the services are stored using a value based indexing mechanism is shown in Figure 4.4. This data structure can be used to directly determine whether a service assigns a particular value to its restricted attribute. Therefore, the complexity of the algorithm that evaluates the values can be reduced to \(mn\).

![Figure 4.4: Value-based Service Indexing](image)

In the proposed technique, services are indexed based on their restricted attributes and the values that they assign to these attributes. This is performed with a three dimensional data structure called an Attribute Leveling Cube (ALC). A global view of this data structure is shown in Figure 4.5. It depicts the dimensions of an attribute leveling generated for a
strictly dependent global constraint \( gc \), which restricts the attributes \( a_x, \ldots, a_y \) of services
of types \( S_1, \ldots, S_n \), where the set \( \{ v_i, \ldots, v_j \} \) contains the values assigned to the attributes
in \( \{ a_x, \ldots, a_y \} \).

\[ \text{Attributes} \]

\[ \text{Service Types} \]

\[ \text{Values} \]

\[ \text{Restricted by } gc \]

\[ \text{Assigned values} \]

\[ \text{Figure 4.5: Dimensions of an Attribute Leveling Cube} \]

A location in an ALC is identified with three coordinates: a service type, an attribute and a value. For example, the triplet \([ S_i, a_j, v_k ]\) represents the location that corresponds to the service type \( S_i \), attribute \( a_j \) and value \( v_k \). Each location holds a reference to a list candidate services. A list at a location \([ S_i, a_j, v_k ]\) contains services of type \( S_i \) which assign the value \( v_k \) to the attribute \( a_j \).

The algorithm used to identify restricted attributes and generate an attribute leveling cube is given below (see Algorithm 4.2). This algorithm requires a strictly dependent global constraint and the candidate lists located for the service types in a composite service template. First, it iterates through the candidates of each type (lines 4 and 5) and locates the restricted attributes (lines 6-9). Then, it obtains the domains\(^7\) of the conditions that describe the restricted attributes (line 10 and 16). Finally, this algorithm iterates through the values in these condition domains (lines 11-13 and 17-19) and stores references to the candidate services at the appropriate locations of the Attribute Leveling Cube (line 12 and 18). Algorithm 4.2 has a polynomial time complexity \( abcd e \) where \( a \) is the number of service types, \( b \) is the number of candidates, \( c \) is the number of conditions used to describe each candidate service, \( d \) is the number attributes that are semantically related to each restricted attribute, and \( e \) is the

\[ ^7\text{A condition domain describes the values that can be assigned to an attribute.} \]
number of values in each condition domain). This algorithm assumes: (i) that each attribute is semantically related to a bounded number of attributes, and (ii) that each condition domain contains a bounded number of values. The following provides brief descriptions of the details returned by the functions used in this algorithm.

- `restrictedAttributes(gc)`: Attributes restricted by constraint `gc` (line 3).
- `candidates(S)`: Candidate services of type `S` (line 5).
- `conditions(s, t)`: Conditions of type `t` (either pre-conditions, post-conditions or both) in the description of service `s` (line 6).
- `conditionDomain(cnd)`: Domain of a condition `cnd` (line 10).
- `semanticallyRelated(a)`: Set of attributes semantically related to attribute `a` (lines 9 and 14).
- `mappedValue(a, v, b)`: The value mapped to `b` (according to the ontological descriptions) when `v` is assigned to `a`, and `a` is semantically related to `b` (line 15).

Following this, details associated with attributes that do not level (according to Definition 4.5) are eliminated from the data structure. Let `ALC` be an attribute leveling cube generated with the service types `{S_m, \ldots, S_n}`, attributes `{a_x, \ldots, a_y}` and values `{v_p, \ldots, v_q}`. For a particular service type `S_j` and attribute `a_k`, if \( \notin \{v_{i}\} \) such that location \([S_j, a_k, v_i] \neq \emptyset\), where \(S_j\in\{S_m, \ldots, S_n\}\) and \(a_k\in\{a_x, \ldots, a_y\}\), then all the details associated with attribute `a_k` are removed from `ALC`.

Algorithm 4.3 ensures that only the details associated with leveled attributes are kept in an `ALC`. First, this algorithm traverses through an `ALC` (lines 5 and 6) and identifies attributes which do not level (lines 7-12). Then, all the entries associated with such attributes are removed (lines 13-17). The complexity of the algorithm is polynomial.

Next, this technique generates value vectors. These value vectors are used to identify the candidate services (in the attribute leveling cube), that need to be combined to form conforming composite services. Given the attribute leveling cube `ALC` which is described above, a value vector is generated for each value `v_i` assigned to a restricted attribute `a_j` by a service of type `S_m`, where \(v_i\in\{v'_p, \ldots, v'_q\}\), \(a_j\in\{a'_x, \ldots, a'_y\}\), \(\{v'_p, \ldots, v'_q\}\subseteq\{v_p, \ldots, v_q\}\) and \(\{a'_x, \ldots, a'_y\}\subseteq\{a_x, \ldots, a_y\}\). Note that \(\{a'_x, \ldots, a'_y\}\subseteq\{a_x, \ldots, a_y\}\) because details associated with some attributes would be eliminated when a cube is leveled, and \(\{v'_p, \ldots, v'_q\}\subseteq\{v_p, \ldots, v_q\}\)
cubeGeneration(candidates(S_m), ..., candidates(S_n), gc)

initialise array ALC;

\{ c_m.a_m, ..., c_n.a_n \} ← restrictedAttributes(gc);

for each candidates(S_i) in (candidates(S_m), ..., candidates(S_n)) do

for each s_x in candidates(S_i) do

for each c_y in conditions(s_x, category(gc)) do

\( c_y ← concept(cnd_y); \)
\( a_y ← attribute(cnd_y); \)

if \( c_i.a_i ∈ \text{semanticallyRelated}(c_y.a_y) \) then

\( d_y ← \text{conditionDomain}(cnd_y); \)

for each value v_k in domain d_y do

\( \text{ALC}[S_i, c_y.a_y, v_k] ← \text{ALC}[S_i, c_y.a_y, v_k] \cup s_x; \)

end

for each c_j.a_j in \text{semanticallyRelated}(c_i.a_i) do

\( d_j ← \text{mappedValues}(c_y.a_y, c_j.a_j, d_y); \)

for each value v_k in domain d_j do

\( \text{ALC}[S_i, c_j.a_j, v_k] ← \text{ALC}[S_i, c_j.a_j, v_k] \cup s_x; \)

end

end

end

Algorithm 4.2: Generating an Attribute Leveling Cube

because services of type \( S_m \) may not assign every value in \( \{ v_p, ..., v_q \} \) to the restricted attributes.

The way in which value vectors are generated is shown in Algorithm 4.4. This algorithm requires a strictly dependent global constraint and a leveled attribute leveling cube. First, one service type (affected by the constraint) is selected. Then, the algorithm identifies the values that are assigned to the restricted attributes of candidate services of the selected type (lines 5-7). For each these values, a value vector is generated (lines 6-16) and placed a list
1 leveling(ALC)
2 for i←0; i<|ALC[0]|; i++ do
3     for k←0; k<|ALC|; k++ do
4         check = false;
5         for j←0; j<|ALC[k, i]|; j++ do
6             if ALC[k, i, j] is not empty then
7                 check = true;
8             end
9         end
10        if check = false then
11           for m←0; m<|ALC|; m++ do
12             ALC[m, i, j]←∅;
13          end
14      end
15  end

Algorithm 4.3: Leveling an Attribute Leveling Cube

(line 17). This algorithm also has a polynomial time complexity. Brief descriptions of the
details returned by the functions used in this algorithm are provided in the following.

- binaryAttributeComparison(gc): The set of binary attribute comparisons used to specify gc (line 7).

- attributes(ac): The two attributes used to specify a binary attribute comparison ac (line 8).

- comparisonOperator(ac): The comparison operator in a binary attribute comparison ac (line 9).

- determineValue(a₂, a₁, v, c): The value that should be assigned a₂ according to the
  relation c between a₁ and a₂, and the value v assigned to a₁ (line 12).

Finally, a set of service tuples are retrieved from an attribute leveling cube. Each service
tuple in this set is a conforming composite service. The service tuples are formed by combi-
Algorithm 4.4: Generating Value Vectors

The candidate service lists that have to be combined are indicated by the value vectors. Let us consider an attribute leveling cube \( ALC \) and a value vector \([v_x, \ldots, v_y]\) to illustrate the way in which these candidate lists are combined. \( ALC \) is generated for a constraint that is applied on services of types \( S_x, \ldots, S_y \) and \([v_x, \ldots, v_y]\) contains values that can be assigned to the attribute \( a_j \). A set of service tuples \( ST \) is generated with \( ALC \) and \([v_x, \ldots, v_y]\) as follows.

\[
ST = \forall s_x \in \text{candidates}([S_x, a_j, v_x]) \times \cdots \times \forall s_y \in \text{candidates}([S_x, a_j, v_x])
\]

\(^8\)Note that each slot in an attribute leveling cube contains a list of candidate services.
Such a set of service tuples is generated from each value vector. \( ST \) is only generated if all the retrieved candidate service lists are not empty. If not, composite services that do not include all the required constituent services would be generated. The algorithm that generates \( ST \) for each value vector is given below (see Algorithm 4.5). This algorithm requires a leveled attribute leveling cube and a list of value vectors. It iterates through the value vectors, extracts the candidate service lists at the indicated locations, and places them in a service list tuple (line 11). If a candidate service list is empty it continues with the next value vector (lines 8-9). Then, the Cartesian product of the candidate service lists in each service list tuple is obtained to generate a set of service tuples \( ST \) (lines 15-23). These \( ST \)s that are generated for all the value vectors are included in the set service_tuples (line 24). All the service tuples in this set are conforming composite services.

The worst-case time complexity of Algorithm 4.5 is exponential. If we assume that a value vector contains \( n \) values and each extracted candidate service list contains \( m \) services, then \( m^n \) service tuples would be generated by this algorithm. However, experimental evidence (in Figures 4.8 and 4.10) indicates that this technique (for locating composite services) is “generally” polynomial since Algorithm 4.5 does not consider non-conforming composite services.

Let us assume that the services described in Figure 4.6 are available. When every combination generated with these services forms a conforming composite service, Algorithm 4.5 and a naive approach would generate an exponential number \((2^3)\) of service tuples. However, if the service tuple \([\text{ComputerSales-II, Shipping-II, Insurance-II}]\) is the only one that forms a conforming composite service, the other services (\( \text{ComputerSales-I, Shipping-I} \) and \( \text{Insurance-I} \)) would not be considered by Algorithm 4.5. These remaining services would be eliminated from \( ALC \) when the attributes are leveled, or they would be avoided by the process that retrieves the candidate service lists according to the value vectors. Algorithm 4.5 directly generates the service tuple \([\text{ComputerSales-II, Shipping-II, Insurance-II}]\), whereas a naive approach may still consider an exponential number of service tuples before it retrieves.
CHAPTER 4. S-MATCH: MATCHING STRICTLY DEPENDENT GLOBAL CONSTRAINTS

locateCompositeServices(ALC, VEP)

for each value vector vep in VEP do
  service_list_tuple ← NULL;
  attribute ← vep.getAttribute();
  for i←0; i<|ALC|; i++ do
    value ← vep.getValue();
    if ALC[i, attribute, value] is empty then
      service_list_tuple ← NULL;
      break;
    else
      service_list_tuple.add(ALC[i, attribute, value]);
    end
  end
  if service_list_tuple ≠ NULL then
    candidates_p, ..., candidates_q ← service_list_tuple;
    ST ← NULL;
    for each service s_p in candidates_p do
      for each service s___ in candidates___ do
        for each service s_q in candidates_q do
          ST.add(s_p, ..., s_q);
        end
      end
    end
    service_tuples ← service_tuples ∪ ST;
  end
end

Algorithm 4.5: Locating Composite Services that Conform to Strictly Dependent Global Constraints

the conforming composite services.
4.5 Discussion

The proposed approach performs better than existing solutions because:

(i) it locates composite services that conform to strictly dependent global constraints in situations where user requests and service descriptions are syntactically heterogeneous, and

(ii) it does so in polynomial time.

The main drawback of the proposed approach is that it retrieves duplicate entries. A duplicate entry occurs if the same service tuple is retrieved more than once. This happens for two reasons.

1. Restricted attributes of constituent services are semantically related to multiple attributes that level\(^9\). Let \(gc\) be a strictly dependent global constraint. If there is a service tuple \(s_x, \ldots, s_y\), where their restricted attributes can be leveled at more than one attribute and the values assigned to them conform to \(gc\), then that service tuple would be retrieved more than once by this technique. If the restricted attributes can be leveled at \(n\) attributes, where \(n > 1\), then there would be \(n-1\) duplicate entries.

2. Multiple conforming values are assigned to restricted attributes. Let \(gc\) be a strictly dependent global constraint that restricts the attributes \(a_x, \ldots, a_y\), where the services \(s_x, \ldots, s_y\) assign the values \([v_{x1}, \ldots, v_{y1}]\) and \([v_{x2}, \ldots, v_{y2}]\) to the restricted attributes. If both tuples of values conform to \(gc\), one duplicate entry would be retrieved by Algorithm 4.5. In a situation in which \(n\) conforming value tuples can be assigned to the restricted attributes by a tuple of services, where \(n > 1\), \(n-1\) duplicate entries would be retrieved by this technique.

However, the number of these duplicate entries increases at a linear rate. If there are \(p\) conforming composite services, the number of duplicate entries that can be generated would be \(p(q-1)(r-1)\), where \(q\) is the number of attributes at which each tuple of restricted attributes level, and \(r\) is the number of conforming value tuples that can be assigned to each tuple of restricted attributes.

\(^9\)See Definition 4.5 for a description on attribute leveling.
4.6 Theoretical Analysis

This section provides an analysis of the soundness and completeness of S-Match. The following describes the semantics of these properties when they are applied to service discovery techniques.

- **Soundness** - Retrieved services always conforms to given constraints.
- **Completeness** - Locates all the services that conform to a given constraint (from those available).

We assume that the given descriptions (service descriptions, user requests and ontological descriptions) and their interpretations are accurate. Interpretations are obtained by substituting and semantically relating terms in these descriptions using ontological relationships. Table 4.2 describes the notations used in this section.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$gc$</td>
<td>A given global constraint which restricts the attributes ${a_x, \ldots, a_y}$.</td>
</tr>
<tr>
<td>$A$</td>
<td>Available services.</td>
</tr>
<tr>
<td>$VL$</td>
<td>A set ${L_x, \ldots, L_y}$ where each $L_i$ contains the values assigned to an attribute $a_i$ in ${a_x, \ldots, a_y}$ by the services in $A$.</td>
</tr>
<tr>
<td>$K$</td>
<td>$\prod_{i=1}^{n} L_i, \mid a_x, \ldots, a_y \mid = n.$</td>
</tr>
<tr>
<td>$Q$</td>
<td>Tuples of values in $K$ that conform to $gc$.</td>
</tr>
<tr>
<td>$P$</td>
<td>$\prod_{i=1}^{n} A_i, \mid a_x, \ldots, a_y \mid = n.$</td>
</tr>
<tr>
<td>$R$</td>
<td>The set of composite services in $P$ that conform to $gc$ (That means, the constituent services of each service $p \in P$ assign the values in a tuple $q$ to their restricted attributes).</td>
</tr>
</tbody>
</table>

*Table 4.2: Table of Notations*

A composite service conforms to a request if the constituent services are of appropriate types and the values assigned to their restricted attributes adhere to the binary attribute comparisons of the given constraint (see Definition 4.6). The proposed technique ensures that constituent services are of appropriate types by generating candidate service lists. Services that assign conforming values to their restricted attributes are selected by indexing them based on the assigned values and generating value vectors. These value vectors consist
of values that should be assigned to restricted attributes of conforming services. Therefore, whether a certain property (soundness or completeness) can be associated with this technique is dependent on the process that generates these value vectors.

**Soundness**

A service discovery technique $T$ is sound if all located services conform to a given constraint $gc$. That means, $T$ is correct if

$$T(P) \subseteq R$$

(4.1)

$T$ achieves this property if the process $T^*$ used to generate value vectors adheres to the following.

$$T^*(K) \subseteq Q$$

(4.2)

S-Match (the proposed approach), generates a set of value vectors $M$, where $\forall m \in M$, $m = [v_x, \ldots, v_y]$, and each $v_i \in \{v_x, \ldots, v_y\}$ can be assigned to $a_i \in \{a_x, \ldots, a_y\}$. Each value vector $m$ conforms to $gc$. The values $v_{x+1}, \ldots, v_y$ assigned to $a_{x+1}, \ldots, a_y$ are derived based on a value $v_x$ assigned to $a_x$ and the binary attribute comparisons of $gc$. $M \cap Q \neq \emptyset$ since some of these derived values may not be actually assigned to restricted attributes. However, the value vectors in $(M \cap Q)$ are disregarded when services are extracted from an attribute leveling cube to form conforming composite services. Therefore, S-Match is a sound approach.

**Completeness**

A service discovery technique $T$ is complete if it locates all the available conforming services. That means, $T$ is complete if

$$R \subseteq T(P)$$

(4.3)

$T$ achieves this property if the process $T^*$ used to generate value vectors adheres to the following.

$$Q \subseteq T^*(K)$$

(4.4)

S-Match, generates a value vector for each value in $L_x$. That means, the proposed technique (i) considers all the values assigned to the restricted attribute $a_x$ (by the available services), (ii) obtains all the relevant interpretations $V_x$ of each assigned value $v_x$ using ontological descriptions, and (iii) generates a value vector for each value $v'_x \in V_x$. Any tuple
\( \forall p \in Q \implies p \in T^*(K) \) since \( p \) has to include a value assigned \( a_x \). Thus, \( Q \subseteq T^*(K) \) and the proposed approach is complete.

4.7 Evaluation

The effectiveness of the proposed approach was evaluated by comparing it against a syntactic-based composite matching technique [Wu et al., 2003a] and a naive approach defined by extending the semantic-based approach in [Elgedawy et al., 2004a]. They were compared using simulation experiments and a case study. Details of these evaluations are provided in the following.

4.7.1 Experiments

This evaluation compares implementations of the three techniques (proposed approach, and those defines in [Wu et al., 2003a] and [Elgedawy et al., 2004a]) for performance and recall. Recall was calculated based on the number of retrieved unique conforming composite services.

The technique proposed in [Wu et al., 2003a] is implemented with JSHOP\(^{10}\) and others were done in Java (JDK 1.5.0)\(^{11}\). A MySQL 5.0 Community Server\(^{12}\) database is used as a registry to store the service descriptions and the ontological descriptions. Experiments were conducted on Pentium IV 3.0 GHz machines with 1 Gigabyte of memory.

To our knowledge, there is no standard data set that can be used to evaluate semantic-based service discovery techniques. We were unable to obtain a real-world data set of semantic-based web service descriptions. None of the composite matching techniques that we came across [Agarwal et al., 2005; Akkiraju et al., 2006b; McIlraith and Son, 2002; Paolucci et al., 2003; Sirin and Parsia, 2004; Wu et al., 2003a; Zeng et al., 2004] defined an approach that could be used to generate test data. Therefore, the following steps were performed to generate a test data set.

1. Defined an ontology with information extracted from the following sources.
   (a) http://www.isima.fr/rey/demoBCover.html.
   (b) http://www.daml.org/ontologies/
   (c) http://www.wsmd.org/WSMO_ontologies.html
   (d) http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl

\(^{10}\)https://sourceforge.net/projects/shop - Java version of SHOP.
\(^{11}\)http://java.sun.com
\(^{12}\)http://www.mysql.com

(October 2008)
The ontology contained 157 concepts, 19 roles and 27 operations, with each one being described with a set of attributes.

2. Randomly generated 2000 graphs (substitution graphs and transformations graphs) to define ontological relationships.

3. Service descriptions were created with elements that were randomly selected from the ontology. Each description had an operation, an affected concept, a role and between 20 to 30 pre-conditions and post-conditions.

A similar approach was used in [Elgedawy et al., 2004a] and [Mokhtar et al., 2006] to generate test data to evaluate implementations of their semantic-based matching techniques. Three types of experiments were conducted in this evaluation:

1. with forced requests
2. with random requests
3. in a restricted environment

A random request includes a constraint specified with randomly selected attributes, whereas a forced request includes one specified with attributes that have the same scopes. If a constraint of a user request is specified with attributes that have different scopes, a syntactic-based technique would not be able to retrieve any conforming composite services. Let us consider the constraint \( \text{location.Dispatch} = \text{location.Pickup} \in \text{validRegion.Insurance} \). These attributes \( \text{location.Dispatch}, \text{location.Pickup} \) and \( \text{validRegion.Insurance} \) have different scopes (e.g. a name of suburb, a name of a country and a name of a state has to be assigned to the three attributes respectively). In such a situation a syntactic-based approach would not be able to compare the values assigned to these attributes. Experiments were conducted with forced requests and random requests to evaluate the proposed technique in situations where a syntactic-based approach is capable and incapable of retrieving services.

Experiments were conducted by varying the number of available service descriptions and service types in composite service templates. Those conducted with random and forced requests considered between 200 to 2000 service descriptions and composite service templates that include between 5 to 25 service types. It was not feasible to execute (the naive approach implemented with) the technique in [Elgedawy et al., 2004a] in such an environment since it employed an exponential algorithm. Hence, the experiments that considered the naive
approach were conducted in a restricted environment, which contained between 20 to 60 service descriptions and composite service templates that include between 2 to 8 service types.

The results from the experiments that varied the number of available services are shown in Figures 4.7 and 4.8 (The Semantic label refers to the naive approach). The composite service templates contained 10 service types when these experiments were conducted with random and forced requests and 4 service types when they were conducted in a restricted environment.

The results from the experiments that varied the number of service types in a composite service template are shown in Figures 4.9 and 4.10. 500 service descriptions were considered when these experiments were conducted with random and forced requests whereas 50 were considered when they were conducted in a restricted environment.

The syntactic-based technique was the most efficient. Unlike the two semantic-based approaches, it did not perform any ontological reasoning. The naive semantic-based technique was the least efficient in the experiments that were conducted in the restricted environment. In the two worst cases (with 60 services and a composite service template that includes 8 service types) it took 22.2 minutes and 45.7 minutes respectively. The proposed technique completed its execution in 46 seconds and 14 seconds.

On the other hand, the two semantic-based approaches were able to achieve higher recall than the syntactic-based approach (in the experiments that were conducted in the restricted environment). The syntactic-based technique achieved the lowest recall in all three experiments. It was unable to locate any conforming composite services with the random requests. The syntactic-based technique was unable to compare syntactically heterogeneous attributes because it does not perform any ontological reasoning.

From these results it is evident that the proposed approach: (i) is more efficient and scalable than a semantic-based approach which does not incorporate a facility to handle global constraints, and (ii) achieves higher recall than a syntactic-based approach.

4.7.2 Case Study

A case study which analyses the three service discovery techniques (the proposed approach, syntactic-based approach in [Rao and Su, 2004; Wu et al., 2003a], and a naive approach defined with the technique in [Elgedawy et al., 2004a]) is described here. A scenario where a user attempts to purchase a computer using web services is considered for this case study. It
Figure 4.7: Recall - Strictly-dependent Constraints - Varying no. of Available Services

is assumed that the sample request described in Section 4.3 (which contains \textit{location.Dispatch} = \textit{location.Pickup} ∈ \textit{validRegion.Insurance}) is given by the user and the services described in Figure 4.11 are available (only the conditions that describe the restricted attributes are
Figure 4.8: Time Taken - Strictly-dependent Constraints - Varying no. of Available Services

specified in the descriptions).

The ontological relationships and some relevant attribute mappings are described in Table 4.3 ($C \rightarrow D$ indicates that $C$ substitutes $D$). The steps taken by these techniques to locate
Figure 4.9: Recall - Strictly-dependent Constraints - Varying no. of Service Types

conforming composite services are described in the following.

**Proposed Technique** First, the candidate services are located and they are given in Table 4.4.
Next, the restricted attributes are identified and an attribute leveling cube is generated. Figure 4.12 provides a two-dimensional view of the generated attribute leveling cube. The two-dimensions correspond to the service types in the composite service template.

Figure 4.10: Time Taken - Strictly-dependent Constraints - Varying no. of Service Types

---

(October 2008)
CHAPTER 4. S-MATCH: MATCHING STRICTLY DEPENDENT GLOBAL CONSTRAINTS

![Diagram](image)

**Figure 4.11: Case Study - Available Services**

<table>
<thead>
<tr>
<th>Concept substitutions</th>
<th>Source attribute</th>
<th>Source value</th>
<th>Target attribute</th>
<th>Target value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item (\rightarrow) Computer</td>
<td>pickupLocation</td>
<td>MELBOURNE</td>
<td>name</td>
<td>VIC</td>
</tr>
<tr>
<td>Desktop (\rightarrow) Computer</td>
<td>collectionPoint</td>
<td>VIC, NSW, SA</td>
<td>name</td>
<td>VIC</td>
</tr>
<tr>
<td>Laptop (\rightarrow) Computer</td>
<td>collectionPoint</td>
<td>WA</td>
<td>name</td>
<td>WA</td>
</tr>
<tr>
<td>Cargo (\rightarrow) State</td>
<td>collectionPoint</td>
<td>VIC, NSW, SA</td>
<td>name</td>
<td>AUSTRALIA</td>
</tr>
<tr>
<td>Cargo (\rightarrow) Country</td>
<td>collectionPoint</td>
<td>WA</td>
<td>name</td>
<td>AUSTRALIA</td>
</tr>
<tr>
<td>Coverage (\rightarrow) Country</td>
<td>region</td>
<td>VIC</td>
<td>name</td>
<td>VIC</td>
</tr>
<tr>
<td>Dispatch (\rightarrow) State</td>
<td>location</td>
<td>SYDNEY</td>
<td>name</td>
<td>NSW</td>
</tr>
<tr>
<td>Pickup (\rightarrow) State</td>
<td>location</td>
<td>AUSTRALIA</td>
<td>name</td>
<td>{VIC, NSW, SA, TAS, QLD, WA}</td>
</tr>
<tr>
<td>Pickup (\rightarrow) Country</td>
<td>validRegion</td>
<td>NSW</td>
<td>name</td>
<td>NSW</td>
</tr>
<tr>
<td>Insurance (\rightarrow) State</td>
<td>location</td>
<td>BRISBANE</td>
<td>name</td>
<td>AUSTRALIA</td>
</tr>
<tr>
<td>Shop (\rightarrow) Country</td>
<td>validLocation</td>
<td>JAPAN</td>
<td>name</td>
<td>JAPAN</td>
</tr>
</tbody>
</table>

**Table 4.3: Ontological Relationships**

97 (October 2008)
and the restricted attributes of the candidate services. Each X mark indicates that at least one element in the array at the corresponding location is not empty. Then, this attribute leveling cube is leveled. \textit{name.State} and \textit{name.Country} are the only attributes that level and details of all the other attributes are eliminated from the attribute leveling cube. A sliced view of the attribute leveling cube after it has been leveled is provided in Figure 4.13. The numbers in the slots refer to candidate services.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
Sales(Computer) & Shipping(Computer) & Insure(Computer) \\
\hline
ComputerSales-I & Shipping-I & Insurance-I \\
ComputerSales-II & Shipping-II & Insurance-II \\
ComputerSales-III & Shipping-III & Insurance-III \\
\hline
\end{tabular}
\caption{Case Study - Candidate Services}
\end{table}

Once the attributes are leveled, value vectors are generated. The vector \{\textit{AUSTRALIA}, \textit{AUSTRALIA}, \textit{AUSTRALIA}\} is generated from the details in the slice in Figure 4.13a, and \{\textit{VIC, VIC, VIC}\} and \{\textit{NSW, NSW, NSW}\} are generated from Figure 4.13b. Then, the candidate service lists at the locations indicated by the value vectors are extracted. This results in the tuples \[[\text{ComputerSales-I, \{Shipping-I, Shipping-II\}, Insurance-I}]\] and \[[\text{ComputerSales-II, \{Shipping-I, Shipping-II\}, \{Insurance-I, Insurance-II\}}]\]. Finally, the Cartesian product of the candidate service lists in each tuple is obtained to form a set of conforming composite services. The located composite services are given in Table 4.5.
Naive Technique First the candidates of the service types in the composite service template are located. Then, an exhaustive approach is employed to locate conforming composite services.

- It generates all possible combinations of candidate services ($m^n$ combinations, where $n$ is the number of service types and $m$ is the number of candidates of each type). Since there are 3 service types in the given composite service template and 3 candidates of each type, $3^3$ combinations services are considered for this scenario.

- While generating the combinations, those that consist of restricted attributes that level are identified. An exponential number of attribute combinations may need to be considered to identify those that level ($(p+q)^n$ combinations where $p$ is the number attributes used to describe a service and $q$ is the number of substitu-
tions that can be performed with these attributes). For example, Table 4.6 gives the tuples of attribute considered to determine if the tuple of candidate services *ComputerSales-I*, *Shipping-I*, *Insurance-I* contain restricted attributes that level.

<table>
<thead>
<tr>
<th>ComputerSales-I</th>
<th>Shipping-I</th>
<th>Insurance-I</th>
</tr>
</thead>
<tbody>
<tr>
<td>pickupLocation.Item</td>
<td>collectioPoint.Cargo</td>
<td>region.Coverage</td>
</tr>
<tr>
<td>pickupLocation.Item</td>
<td>collectioPoint.Cargo</td>
<td>name.State</td>
</tr>
<tr>
<td>pickupLocation.Item</td>
<td>name.Country</td>
<td>region.Coverage</td>
</tr>
<tr>
<td>pickupLocation.Item</td>
<td>name.Country</td>
<td>name.State</td>
</tr>
<tr>
<td>pickupLocation.Item</td>
<td>name.State</td>
<td>region.Coverage</td>
</tr>
<tr>
<td>name.State</td>
<td>collectioPoint.Cargo</td>
<td>name.State</td>
</tr>
<tr>
<td>name.State</td>
<td>collectioPoint.Cargo</td>
<td>name.State</td>
</tr>
<tr>
<td>name.State</td>
<td>name.Country</td>
<td>region.Coverage</td>
</tr>
<tr>
<td>name.State</td>
<td>name.Country</td>
<td>name.State</td>
</tr>
<tr>
<td>name.State</td>
<td>name.State</td>
<td>region.Coverage</td>
</tr>
<tr>
<td>name.State</td>
<td>name.State</td>
<td>name.State</td>
</tr>
</tbody>
</table>

*Table 4.6: Naive Approach - Combinations of Attributes*

- Once a tuple of restricted attributes (extracted from a tuple of candidate services) level, the values assigned to them are checked to determine whether they conform to the given constraint. If they do, the tuple of candidate services from which the attributes were extracted is selected as a conforming composite service. This step may require an exponential number of values to be considered \(x^n\) where \(x\) is the number of values assigned to each attribute). The combinations of value assignments that have to be considered to determine whether the values assigned to the restricted attributes of *name.State* of *ComputerSales-I*, *Shipping-I*, *Insurance-I* conform to the given constraint, is given in Table 4.7.

This technique locates the same conforming composite services that were located by the proposed technique. However, unlike the proposed approach which is “generally” polynomial, this approach has an exponential time-complexity.

**Syntactic-based Technique** First, a SHOP-2 method is generated according to the com-
posite service template in the given request. Then, the available service descriptions and the user request are converted to operator instances. The method and the operator instances generated for this scenario are given in Figure 4.14\textsuperscript{13}. Finally, the operator instances are matched (unified) to the method with SHOP-2 to locate composite services. Since this technique is not capable of performing any ontological reasoning, the operation, affected concept and role that describe the services need to be syntactically similar to those that describe the service types of a composite service template. Also, the restricted attributes in the candidate services need to be described with the same attributes that specify the given constraint. In the above scenario, only \textit{ComputerSale-II}, \textit{Shipping-II} and \textit{Insurance-I} meet these requirements. The same value has to be assigned to the restricted attributes of these services according to the given constraint. However, \textit{ComputerSale-II}, \textit{Shipping-II} and \textit{Insurance-I} assign SYDNEY, AUSTRALIA and VIC to their restricted attributes respectively. Therefore, the syntactic approach does not locate any conforming composite services.

Both semantic-based approaches achieve higher recall levels than the syntactic-based technique. This is because they use ontological relationships to match descriptions created with different terminologies. However, the naive technique is inefficient and not scalable because it follows an exhaustive approach. The proposed technique is both scalable and capable of achieving a higher recall level.

### 4.8 Related Work

Current discovery techniques for composite services are of two types: composability-projected and service template-based. The former models the service discovery problem as a planning problem, which attempts to locate an orchestration of services that cause the required state-

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
ComputerSales-I & Shipping-I & Insurance-I \\
VIC & NSW & VIC \\
VIC & SA & VIC \\
VIC & VIC & VIC \\
\hline
\end{tabular}
\caption{Naive Approach - Combinations of Values}
\end{table}

\textsuperscript{13}The ? indicates that it is a variable that needs to be unified.
CHAPTER 4. S-MATCH: MATCHING STRICTLY DEPENDENT GLOBAL CONSTRAINTS

SHOP2 Method

(:method (user_request)
  ((g.constraint ?operation0 ?c.concept0 ?role0 ?concept0 ?attribute0)
   (?service0 ?operation0 ?c.concept0 ?role0 ?concept0 ?attribute0 ?value)
   (g.constraint ?operation1 ?c.concept1 ?role1 ?concept1 ?attribute1)
   (?service1 ?operation1 ?c.concept1 ?role1 ?concept1 ?attribute1 ?value)
   (g.constraint ?operation2 ?c.concept2 ?role2 ?concept2 ?attribute2)

Operator Instances

(ComputerSales-I Sales Desktop Sales_Assistant Item pickupLocation MELBOURNE)
(ComputerSales-II Sales Computer Sales_Assistant Dispatch location SYDNEY)
(ComputerSales-III Sales Laptop Sales_Assistant Shop location BRISBANE)
(Shipping-I Shipping Item Shipping_Agent Cargo collectionPoint VIC)
(Shipping-I Shipping Item Shipping_Agent Cargo collectionPoint NSW)
(Shipping-I Shipping Item Shipping_Agent Cargo collectionPoint SA)
(Shipping-II Shipping Computer Shipping_Agent Pickup location AUSTRALIA)
(Shipping-III Shipping Computer Shipping_Agent Cargo collectionPoint WA)
(Insurance-I Insure Computer Insurance_Representative Insurance validRegion NSW)
(Insurance-II Insure Computer Insurance_Representative Coverage region NSW)
(Insurance-III Insure Laptop Insurance_Representative Service validLocation NSW)
(g.constraint Sales Computer Sales_Assistant Dispatch location)
(g.constraint Shipping Computer Shipping_Agent Pickup location)
(g.constraint Insure Computer Insurance_Representative Insurance validRegion)

Figure 4.14: Syntactic Approach - SHOP2 Method and Operator Instances

transitions (producing the required post-conditions based on some defined pre-conditions). The latter provides an exact specification of the types of component services that should exist in a composite service. The main features and drawbacks of these techniques are described here.
CHAPTER 4. S-MATCH: MATCHING STRICTLY DEPENDENT GLOBAL CONSTRAINTS

Composability-projected Techniques

These techniques employ planners to perform service discovery. They can be divided into two groups as those that use either backward chaining or forward chaining algorithms, based on the reasoning mechanism of their planners. Techniques of the first type are proposed in [Agarwal et al., 2005; Lin et al., 2006; McIlraith and Son, 2002].

The technique introduced in [Agarwal et al., 2005] focuses on both service discovery and execution. It requires OWL ontological descriptions [Arroyo et al., 2004; Staab and Studer, 2004] and OWL-S service descriptions [Ankolekar et al., 2001]. First, a set of composite services that produce the required post-conditions are located. If a service that produces the required post-conditions cannot be located, the one that produces the most number of these post-conditions is selected. If multiple services are retrieved, then the most appropriate one is selected based on non-functional (quality of service) attributes such as response time and cost. Then, an executable BPEL4WS [Foster et al., 2003; Shen et al., 2005] specification is generated for the located service. However, details of the mechanism used to reason about the OWL ontologies are not provided.

A technique that uses Golog (a first-order logic-based planner) [Levesque et al., 1997] to retrieve composite services, is proposed in [McIlraith and Son, 2002]. This technique requires each available service to be described with a procedure. These procedures are selected from a hierarchy of “generic procedures” called a web procedures ontology. A sequence of procedures that produce the required state transition is located using a Golog planner. The approach introduced in [Lin et al., 2006] requires services to be defined with a service name, a set of pre-conditions and a set of post-conditions. The terms used to specify the pre-conditions and post-conditions are defined with Description Logic (DL) terminologies [Baader et al., 2003]. They define four types of relationships to match services: identical, conditionally identical, substitutes and conditionally substitutes. Two services $s_x$ and $s_y$ are identical if they always provide the same functionality. Service $s_x$ substitutes $s_y$ if the state transition caused by $s_y$ is always caused by $s_x$. If these relationships only occur in some situations then they are conditional. Akkiraju et al. suggested to locate composite services using a forward chaining algorithm [Akkiraju et al., 2006a;b;c]. Their technique requires services to be described with WSDL-S [Akkiraju et al., 2005] and uses relationships in OWL ontologies [Arroyo et al., 2004; Staab and Studer, 2004] to match terms in those description. The forward chaining algorithm calculates a semantic score for each service that is selected as a constituent service. A score of 1.0 is given if two terms are equivalent, and a score of 0.5 is given if one term
subsumes the other. An overall score is calculated for each composite service by aggregating the semantic scores of the constituent services. These overall scores are used to rank retrieved services. The algorithm optimises the selection process by only considering a service if the number of required post-conditions that it achieves is more than a pre-defined threshold.

Unlike the technique described in [McIlraith and Son, 2002], expressive descriptions that describe the purpose, state transitions and data transformations were considered in [Agarwal et al., 2005; Akkiraju et al., 2006a;b; Lin et al., 2006]. However, none of these techniques represent global constraints nor locate composite services that conform to such constraints. Therefore, services located by such approaches may not satisfy a user request.

**Service Template-based Techniques**

Service template-based techniques can be categorised as those that use planners and those that do not use planners. All the planner-based techniques used variations of Hierarchical Task Network (HTN) planners [Erol et al., 1994]. The technique introduced in [Paolucci et al., 2003] used a planner called RETSINA [Paolucci et al., 1999; Sycara et al., 2003]. This approach requires OWL-S service descriptions and performs subsumption checking to match terms used in them. However, specific details of as to how RETSINA is used in the service discovery process is not given. Techniques that use SHOP-2 (Simple Hierarchical Ordered Planner-2) [Nau et al., 1999; Nau, 2003] are proposed in [Rao and Su, 2004; Wu et al., 2003a] and [Sirin and Parsia, 2004; Sirin et al., 2004; Sirin and Hendler, 2005]. Both local and global constraints are considered in these two service discovery techniques. They require service descriptions and user requests to be specified with OWL-S. First, a “SHOP method” is generated according to the OWL-S process model of a request. Then, the request and available service descriptions are converted to operator instances. Finally, the operator instances are matched (unified) to the method with SHOP-2 to locate composite services. The approach in [Wu et al., 2003a;b] directly used the unification functions of SHOP-2 to perform the matching. That means basic string matching functions are considered to match service descriptions to a user request. The approach in [Wu et al., 2003a] is extended in [Sirin and Parsia, 2004] by incorporating an Description Logic reasoner [Baader et al., 2003]. However, this reasoner is only able to handle OWL-DL ontologies. OWL-DL is a very restrictive ontology description language which cannot represent most of the relationships that exist between the elements of a domain [Arroyo et al., 2004; Bechhofer et al., 2004; Horrocks et al., 2003; Staab and Studer, 2004]. Also, the reasoner is only used to check
whether the pre-conditions of a given operator (in the hierarchy) are entailed by the state of the planner. Therefore, the approach in [Sirin and Parsia, 2004] is able to locate conforming composite services when a given global constraint is specified with syntactically heterogeneous attributes, but not when requests and service descriptions are also specified with syntactically heterogeneous methods. Therefore, it is similar to a syntactic-based technique. Several experiments that evaluate implementations of this technique have been provided. However, these experiments were performed on test data that is used to evaluate AI planners and Sirin et al.’s technique was not required to perform any ontological reasoning to match requests against service descriptions.

The technique introduced in [Zeng et al., 2003; 2004] requires a request for a composite service to be specified with a state chart diagram. Each state represents a service community, which is a collection of services of a certain type. Services were described with a quality of service model that consists of attributes such as execution price, execution duration, reliability, availability, and reputation. Both local and global constraints were considered in their approach. However, these constraints were limited to those that can be specified with an attribute of the quality of service model. A syntactic-based integer programming technique that focuses on local optimisation is used to match the user requests to the service descriptions.

Except for the technique proposed in [Paolucci et al., 2003], all the other service template-based techniques define models that represent global constraints and they are able to locate conforming composite services. However, all of them are syntactic-based. They are not sufficient to locate suitable composite services when service descriptions and user requests are created with different terminologies. To our knowledge there is no semantic-based matching technique that locates composite services that conform to global constraints.

4.9 Summary

The constraints used to specify functional requirements in user requests are of two types: local and global. Those of the latter type can be divided into two categories as either strictly dependent or independent, based on the complexity finding complete solutions. A global constraint is strictly dependent if the values that should be assigned to all the remaining attributes are uniquely determined once a value is assigned to one attribute. This chapter proposes a sound and complete semantic-based matching technique which locates composite services that conform to such constraints.
First, candidate services are located to ensure that the constituent services of a composite service are of appropriate types. Then, the attributes of the candidate services that are restricted by a given constraint are identified. These attributes need to be identified because the values assigned to them need to be checked to determine whether a service can be included in a composite service. Next, the restricted attributes of candidate services are related to a common attribute to ensure that their scopes are similar and the values assigned to them are comparable. This process which ensures that values assigned to restricted attributes are comparable is referred to as attribute leveling. Finally, services that assign conforming values to their restricted attributes are combined to form conforming composite services. A three-dimensional data structure (called an Attribute Leveling Cube) is used to improve the performance of the technique. An attribute leveling cube is generated by indexing candidate services based on their type, their restricted attributes and the values assigned to these attributes, and is used to optimise the processes that perform attribute leveling and combine candidate services to form composite services. Results obtained from simulation experiments and a case study indicate that the devised approach achieves higher recall than a syntactic-based approach and that it performs better than a naive semantic-based approach.
Chapter 5

I-MATCH: Matching Independent Global Constraints

This chapter proposes a semantic-based matching approach to locate services that conform to independent global constraints in polynomial time [Gooneratne and Tari, 2008]. Two types of constraints are considered: functional and relational. The latter is a collection of binary constraints whereas the former is a locally optimisable non-binary constraint. First, values that conform to a given constraint are identified with either local optimisation or approximation techniques. Then, services that assign these values to attributes restricted by global constraints are located and later combined to form composite services. The service selection process is optimised by indexing the available services based on the values they assign to their restricted attributes. Experimental results show that the proposed approach performs better than existing semantic-based approaches and achieves higher recall than a syntactic-based approach.

5.1 Motivation

Constraints are included in user requests to accurately model requirements. Those used when specifying functional requirements are of two types: local and global. The former restricts the values of a particular attribute of a single service, whereas the latter simultaneously restricts the values of two or more attributes of multiple constituent services. Global constraints can be classified based on the complexity of solving them (i.e. determining the values that should be assigned to their attributes) as either strictly dependent or independent. A global con-
constraint is of the former type if the values that should be assigned to all the remaining restricted attributes can be uniquely determined once a value is assigned to one. \( \text{location.Dispatch} = \text{location.Pickup} \in \text{validRegion.Insurance} \) is an example of a strictly dependent global constraint. Once \textit{MELBOURNE} is assigned to \textit{location.Dispatch}, the same value has to be assigned to \textit{location.Pickup} and included in the set of values assigned to \textit{validRegion.Insurance} (the set can identified by checking for one value). Any global constraint that is not strictly dependent is \textit{independent}. For example, \textit{productionTime.Computer} + \textit{approvalPhase.Insurance} + \textit{timeTaken.Delivery} < 7 DAYS and \textit{availableDate.Computer} ≤ \textit{approvalDate.Insurance} ≤ \textit{date.Pickup} are independent global constraints. A semantic-based matching technique for locating composite services that conform to strictly dependent global constraints is introduced in Chapter 4. None of the current semantic-based matching approaches consider independent global constraints since solving such constraints is NP-hard [Dechter, 1992; Nareyek, 2001; Voudouris and Tsang, 2001].

This chapter proposes a semantic-based matching approach to locate services that conform to independent global constraints. Such constraints can be categorised into various types based on aspects such as arity, mechanisms used to solve them and the complexity of those mechanisms. The technique proposed in this chapter considers two types of independent constraints: functional (locally optimisable non-binary) and relational (binary). Constraints of the former type allow the values assigned to the attributes to be optimised individually. A constraint is of the latter type if its specified with relational operators and attributes. For example, \textit{productionTime.Computer} + \textit{approvalPhase.Insurance} + \textit{timeTaken.Delivery} < 7 DAYS is a functional constraint and \textit{availableDate.Computer} ≤ \textit{approvalDate.Insurance} ≤ \textit{date.Pickup} is a relational constraint.

The proposed technique first indexes services based on the values they assign to the attributes restricted by a global constraint. This enables the quick retrieval of services that assign a particular value to their restricted attribute. Then optimisation techniques are used to find combinations of values that conform to a given constraint in polynomial time (since an exhaustive approach would be exponential [Zeng et al., 2003; 2004]). The optimisation used (to locate conforming values) is determined by the type of a given constraint. If the constraint is functional, pre-defined (user defined) objective functions [Nareyek, 2001; Voudouris and Tsang, 2001] are used to identify conforming values. If the constraint is relational, it is

---

1 A review of existing techniques is given in Section 4.8.
2 Detailed descriptions of ways in which constraint are categorised are given in Section 4.2.

108 (October 2008)
either approximated to a strictly dependent constraint using domain rules [Elgedawy, 2003; Srivastava et al., 2000] (since such constraints can be solved in polynomial time) or conforming values are found using a greedy approach [Minton et al., 1990]. Finally, services that assign these values to their restricted attributes are combined to form conforming composite services.

Extensive experiments are conducted to compare the proposed techniques against several other techniques, such as a syntactic-based composite matching technique [Rao and Su, 2004; Wu et al., 2003a] and a relevant semantic-based matching technique [Elgedawy et al., 2004a] (which incorporates an exhaustive approach). Our results indicate that the proposed techniques are more efficient than existing approaches when dealing with a large number of service descriptions. They achieve a higher recall than the syntactic-based approach in all experiments. Additionally, the complexity and the number of services retrieved by these techniques were analysed using a case study.

The rest of the chapter is organised as follows. Section 5.2 provides concise guidelines on how requests that include either functional or relational constraints should be specified. Section 5.3 describes the proposed composite matching techniques. The proposed techniques are analysed for soundness and completeness in Section 5.4. Details of the simulation experiments and the case study are given in Section 5.5. Finally, the contributions of this chapter are summarised in Section 5.6.

5.2 User Requests

Concise guidelines for creating requests for composite services are provided here. The proposed approach, I-Match assumes that a given request is structured according these guidelines. A request consists of a composite service template and a constraint model. The former specifies the types of services that need to be collaborated to form a composite service. It is defined as a collection of service types. A service type is described with the triplet \([O, C, R]\), where \(O\) is an operation, \(C\) is an affected concept and \(R\) is a role. A formal definition of a Composite Service Template is provided in Section 4.3.

The constraint model contains either a functional or a relational constraint. Next, we describe how these constraints are specified in requests. Typically, a global constraint is specified with a set of attributes. When locating conforming composite services, an exhaustive approach has to be employed to consider all the possible combinations that can be generated from the values assigned to these attributes. However, as described in Section 4.2, this requires an exponential combination of values to be considered. The proposed approach limits
the number of combinations by either approximating the constraint using domain rules or finding values using local optimisation techniques. Nonetheless, if the process that limits the considered combinations is too restrictive and only attempts to locate one conforming combination of values, the number of located composite services would be minimal. Therefore, the proposed approach allows all the values that can be assigned to one attribute (from the set of attributes) to be considered during the matching process. This attribute is referred to as a “free attribute”. By doing so, the number of considered combinations is increased at a linear rate. Let us consider a constraint $gc$ that restricts the attributes $\{a_p, \ldots, a_q\}$ where $n$ values can be assigned to $a_p$ according to the service descriptions. An optimisation technique that considers $m$ combinations of values to locate those that conform, can be extended to consider $mn$ combinations if $a_p$ is considered as a free attribute. Therefore, in the proposed approach, a specification of a constraint includes a non-empty set of attributes (where the values assigned to them are restricted) and a free attribute. The type, which indicates whether a constraint restricts attributes that describe either pre-conditions (of services), post-conditions or both, is also included.

Additionally, specifications of functional constraints should include descriptions of functions that locally optimise the value selection process and aggregate those values. Aggregate functions are required because the relations between attributes of functional constraints are non-binary. Hence, a process that checks if the assigned values conform to a given constraint may require them to be aggregated. The proposed approach assumes that these functions are performed by web services, and refers to them as “Aggregation Services” and “Optimisation Services”. An optimisation service is specified for every restricted attribute except the free attribute. An overview of the structure of such constraints is given below.

$$\text{Functional Constraint} : \{\text{Attribute}\}, \text{Free Attribute}, \text{Type}, \text{Comparison Operator}$$

$$\{\text{Optimisation Service, Attribute}\}, \text{Aggregation Service}$$

A sample specification of the functional constraint $productionTime.Computer + approvalPhase.Insurance + timeTaken.Delivery < 7 DAYS$ is given in Figure 5.1.

A relational constraint is specified with a set of binary attribute comparisons. Each one of them describe a relationship between two attributes. The operator used to specify this relationship can be one of the following: $=, <, >, \leq, \geq, \neq, \in, \subset, \subseteq$ or $\notin$. An overview of
CHAPTER 5. I-MATCH: MATCHING INDEPENDENT GLOBAL CONSTRAINTS

\{productionTime.Computer, approvalPhase.Insurance, timeTaken.Delivery, 7\_DAYS\},
productionTime.Computer, PRE, <
\{\langle min()\rangle, approvalPhase.Insurance\}, \{\langle min()\rangle, timeTaken.Delivery\}\},
\{sum(), \{productionTime.Computer, approvalPhase.Insurance, timeTaken.Delivery\}\}  
\[
\begin{align*}
\text{min} & = 0 \quad \cap \text{Role}(\emptyset) \\
& \quad \cap \text{PreCon}(= \text{status.Selector AWAITING VALUES}) \\
& \quad \cap \text{PostCon}(= \text{status.Selector SELECTED}) \\
& \quad \cap \text{Input}(\exists dateList.Utility) \quad \cap \text{Output}(\exists date.Utility) \\
& \quad \cap \text{OutCon}(\geq date.Utility 0\_DAYS) \\
\text{sum} & = 0 \quad \cap \text{Role}(\emptyset) \\
& \quad \cap \text{PreCon}(= \text{status.Values AWAITING}) \\
& \quad \cap \text{PostCon}(= \text{status.Values PROCESSED}) \\
& \quad \cap \text{Input}(\exists productionTime.Computer) \quad \cap \text{Input}(\exists approvalPhase.Insurance) \\
& \quad \cap \text{Input}(\exists timeTaken.Delivery) \\
& \quad \cap \text{InCon}(\geq productionTime.Computer 0\_DAYS) \\
& \quad \cap \text{InCon}(\geq approvalPhase.Insurance 0\_DAYS) \\
& \quad \cap \text{InCon}(\geq timeTaken.Delivery 0\_DAYS) \\
& \quad \cap \text{Output}(\exists time.Total) \quad \cap \text{OutCon}(\geq time.Total 0\_DAYS)
\end{align*}
\]

**Figure 5.1: Sample Specification of a Functional Constraint**

The structure of a relational constraint is given below.

Relational Constraint : \{Attributes\}, Free Attribute,
\{Binary Attribute Comparison\}, Type

Binary Attribute Comparison : Attribute, Relational Operator, Attribute

A specification of the relational constraint *availableDate.Computer* \leq *approvalDate.Insurance* \leq *date.Pickup* is given in Figure 5.2.
5.3 I-Match

Details of the proposed matching technique, I-Match, are provided in this section. This technique requires: (i) WS-ALUE service descriptions, (ii) user requests structured according to the guidelines given in Section 5.2, and (iii) the terms in descriptions (service descriptions and user requests) to be defined in an ontology structured according to the Meta-Ontology [Elgedawy, 2003; Elgedawy et al., 2004a]. It locates services that conform to independent global constraints and they are formally defined in the following.

**Definition 5.1 (Conforming Composite Service)** Let $gc$ be an independent global constraint that restricts the attributes $[a_1, \ldots, a_n]$ of services of types $[S_1, \ldots, S_m]$. A service tuple $[s_1, \ldots, s_m]$ is a composite service that conforms to $gc$ if:

1. $\forall s_i, s_i$ is of type $S_i$,
2. $\forall s_i, s_i$ is described using an attribute $a'_i$, where $a'_i \in \{a_1, \ldots, a_n\}$, and
3. $[v_1, \ldots, v_n]$ assigned to $[a'_1, \ldots, a'_n]$ of $[s_x, \ldots, s_y]$ conform to $gc$.

There are three main phases in this approach, namely: candidate acquisition, service indexing and composite service acquisition. The first phase locates services of the types in a composite service template\(^3\). This ensures that the constituent services in a composite service are of appropriate types, and reduces the number of services that need to be considered in the following phases. A service of a particular type is referred to as a candidate service. The way in which these candidate services are located, is described in Section 4.4. The second phase indexes services based on the values assigned to restricted attributes, so that those which assign a particular value can be retrieved quickly. The final phase identifies combinations of

---

\(^3\)A composite service template is defined with a collection of service types. (See Section 4.3).
values that conform to a given constraint and combines services that assign those values to their restricted attributes (to form conforming composite services).

The techniques described here do not consider arbitrary interpretations of user constraints. That means, the attributes specified in the constraints are not semantically related to other attributes used to obtain alternative specifications of user requests. Alternative interpretations of user constraint are considered by the technique introduced in Chapter 4, which locates services that conform to strictly dependent global constraints. However, the aggregation services used when locating services that conform to functional constraints require values that can be assigned to specific attributes, as inputs. Hence, the values assigned to the attributes of a functional constraint must be mapped to those acceptable by the relevant aggregation service. For example, a user request includes the constraint \( \text{productionTime}.\text{Computer} + \text{approvalPhase}.\text{Insurance} + \text{timeTaken}.\text{Delivery} < 7 \text{ DAYS} \), and an aggregate service that requires the inputs \( \text{productionTime}.\text{Computer}, \text{approvalPhase}.\text{Insurance} \) and \( \text{duration}.\text{Shipping} \), \( \text{timeTaken}.\text{Delivery} \) is semantically related to \( \text{duration}.\text{Shipping} \) and \( \text{deliveryTime}.\text{Consignment} \) according to the ontological descriptions. In such a situation the interpretation of the constraint obtained by substituting \( \text{timeTaken}.\text{Delivery} \) with \( \text{deliveryTime}.\text{Consignment} \) is ineffectual since values assigned to \( \text{deliveryTime}.\text{Consignment} \) are not accepted by the aggregation service.

5.3.1 Service Indexing

This phase identifies the restricted attributes of candidate services and later indexes them in a two dimensional structure (based on the values assigned to these attributes). An attribute is said to be “restricted” if its values are restricted by a user constraint. Such attributes need to be identified because the values assigned to them determine whether a particular candidate service can be included in a conforming composite service. Since this technique is semantic-based, an attribute of a service is a restricted attribute if it is semantically related to one used to specify a global constraint. A formal description of restricted attributes is given in Definition 4.4.

Candidate services need to be indexed based on the values assigned to their restricted attributes because of the following reasons.

1. To compile a list of assigned values for each restricted attribute. The optimisation techniques used in the next phase require these lists to find conforming values. They can be compiled easily by indexing services based on the assigned values. Let us assume that
the services described in Table 5.1 are available. This table describes their restricted attributes and the assigned values. The attributes *time.Assemble*, *timeTaken.Cover* and *duration.Shipping* are semantically related to *productionTime.Computer*, *approvalPhase.Insurance* and *timeTaken.Delivery* respectively, according to the ontological relationships. If the services in Table 5.1 are indexed as in Figure 5.3, then the lists of values assigned to the restricted attributes can be compiled easily.

<table>
<thead>
<tr>
<th>Service</th>
<th>Attribute</th>
<th>Assigned Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComputerSales-I</td>
<td><em>time.Assemble</em></td>
<td>5_DAYS, 4_DAYS</td>
</tr>
<tr>
<td>ComputerSales-II</td>
<td><em>productionTime.Computer</em></td>
<td>5_DAYS, 3_DAYS</td>
</tr>
<tr>
<td>Insurance-I</td>
<td><em>approvalPhase.Insurance</em></td>
<td>5_DAYS, 3_DAYS</td>
</tr>
<tr>
<td>Insurance-II</td>
<td><em>timeTaken.Cover</em></td>
<td>48_HOURS, 24_HOURS</td>
</tr>
<tr>
<td>Shipping-I</td>
<td><em>timeTaken.Delivery</em></td>
<td>5_DAYS, 3_DAYS</td>
</tr>
<tr>
<td>Shipping-II</td>
<td><em>duration.Shipping</em></td>
<td>72_HOURS, 48_HOURS</td>
</tr>
</tbody>
</table>

*Table 5.1: Sample Services*

![Figure 5.3: Indexed Services](image)

2. Once the conforming values are found, services that assign them to their restricted attributes will be combined to form composite services. When doing so, services that assign these conforming values to their restricted attributes can be retrieved quickly if they are indexed based on the assigned values.

Let us consider a global constraint \( gc \) which restricts the attributes \( a_x, \ldots, a_y \) of services

(October 2008)
of types \( S_x, \ldots, S_y \), where \(|x, \ldots, y| = p \). Each service type has \( q \) candidates and each one of them assigns \( m \) values to a restricted attribute. Each attribute is semantically related to \( n \) attributes. When a conforming tuple of values \([v_x, \ldots, v_y]\) is found, a naive approach would perform \( pqmn \) operations (i.e. \( O(n^4) \)) to locate the services that conform to \( gc \). However, if the services are indexed based on the values assigned to their restricted attributes this can be performed in \( mn \) operations (i.e. \( O(n^2) \)).

Let us assume that the constraint \( \text{productionTime.Computer} + \text{approvalPhase.Insurance} + \text{timeTaken.Delivery} < 7 \text{DAYS} \), the services in Table 5.1 and the tuple of conforming values \([5 \text{DAYS}, 1 \text{DAY}, 2 \text{DAYS}]\), are given. In this situation, if the services are not indexed based on the values they assign to their restricted attributes, 30 operations (which includes 6 operations to semantically relate and map values according to the ontological relationships and \( 3 \times 2 \times 2 \) operations to iterate through the values) need to be performed to locate those that assign the conforming values. However, it would only take 5 operations if the services are indexed as in Figure 5.3.

I-Match generates a set of slot lists to index candidate services based on the values assigned to restricted attributes. Figure 5.4 provides a global view of this data structure. It depicts a set of slot lists \( \{L_1, L_2, L_3, L_4\} \) generated for a global constraint that restricts the attributes \([a_1, a_2, a_3, a_4]\). Note that \( L_i \) is generated for each \( a_i \). A slot \( l_{ij} \) is included in a list \( L_i \) for each value \( v_j \) that can be assigned to attribute \( a_i \). Each slot contains a set of services. A service \( s_k \) is included in a slot \( l_{ij} \) if it is able to assign the value \( v_j \) to \( a_i \). That means, the service description of \( s_k \) is specified using either

- an attribute \( a_i \) to which \( v_j \) is assigned or
- an attribute \( a'_i \) to which a value \( v'_j \) is assigned: \( a'_i \) is semantically related to \( a_i \) and the relationship causes \( v'_j \) to be mapped to \( v_j \).

For example, the values assigned to the restricted attribute \( \text{duration.Shipping} \) of \( \text{Shipping-II} \) describes the time taken to ship an item in “hours”. However, if the ontological relationships semantically relate \( \text{duration.Shipping} \) to \( \text{timeTaken.Delivery} \), and map the assigned values to \( 3 \text{DAYS} \) and \( 2 \text{DAYS} \), then \( \text{Shipping-II} \) is placed in the slots that correspond to \( \langle \text{timeTaken.Delivery}, 3 \text{DAYS} \rangle \) and \( \langle \text{timeTaken.Delivery}, 2 \text{DAYS} \rangle \).

The algorithm that generates a set of slot lists is given below (see Algorithm 5.1). This algorithm requires a global constraint and a set of candidate service lists. First, it iterates
through the candidate services (lines 4 and 5) and identifies the restricted attributes (line 6-9 and 14). Then, the values that can be assigned to these attributes are determined (lines 10 and 18) and services are stored in the appropriate slots (lines 11-13 and 19-21). The following provides descriptions of the details returned by the functions of Algorithm 5.1.

- **restrictedAttributes(gc)**: Attributes restricted by constraint gc (line 3).
- **candidates(S)**: Candidate services of type S (line 5).
- **conditions(s, t)**: Conditions of type t (either pre-conditions, post-conditions or both) included in the description of a service s (line 6).
- **conditionDomain(cnd)**: Domain of a condition cnd (line 10).
- **semanticallyRelated(a)**: Set of attributes semantically related to a (line 14).
- **mappedValue(a, v, b)**: The value mapped to b (according to the ontological descriptions) when v is assigned to a, and a is semantically related to b (line 17).
- **conditionDomain(a, o, v)**: Domain of a condition described with attribute a, operator o and value v (line 18).

The time complexity of Algorithm 5.1 is $O(n^2)$. This is determined by the number of restricted attributes, candidate services, conditions service descriptions and the values that can be assigned to a restricted attribute. This algorithm assumes that the domain of values
1. GenerateSlotLists(candidates(S_m), ..., candidates(S_n), gc)
2. initialise Slot Lists SL;
3. \{c_m.a_m, ..., c_n.a_n\} ← restrictedAttributes(gc);
4. for each candidates(S_i) in \{candidates(S_m), ..., candidates(S_n)\} do
   5. for each s_x in candidates(S_i) do
      6. for each cnd_y in conditions(s_x, category(gc)) do
         7. c_y ← concept(cnd_y);
         8. a_y ← attribute(cnd_y);
         9. if c_i.a_i = c_y.a_y then
            10. d_y ← conditionDomain(cnd_y);
            11. for each value v_k in domain d_y do
               12. SL[S_i, v_k] ← SL[S_i, v_k] ∪ s_x;
            end
         else if c_i.a_i ∈ semanticallyRelated(c_y.a_y) then
            14. o_y ← comparisonOperator(cnd_y);
            15. v_y ← value(cnd_y);
            16. v_y' ← mappedValue(c_y.a_y, v_y, c_i.a_i);
            17. d_y ← conditionDomain(c_y.a_y, o_y, v_y');
            18. for each value v_k in domain d_y do
               19. SL[S_i, v_k] ← SL[S_i, v_k] ∪ s_x;
            end
      end
   end
end

Algorithm 5.1: Generating Slot Lists

that can be assigned to a particular attribute and the number of attributes to which a particular attribute is semantically related are bounded. The number of restricted attributes used to describe a constraint and the number of conditions in a service description are quite small, and their impact on the complexity on this algorithm is insignificant.
5.3.2 Composite Service Acquisition

The task performed in this phase is divided into two steps. First, values that conform to a given constraint are identified from those assigned to the restricted attributes. Then, services that assign these conforming values are combined to form composite services. Identifying the values that conform to a given constraint is an NP-hard problem [Dechter, 1992; Nareyek, 2001; Voudouris and Tsang, 2001]. Given a constraint $gc$ which restricts the attributes $\{a_x, \ldots, a_y\}$, where $|x, \ldots, y| = n$ and $m$ values are assigned to each attribute $a_i$ by the candidate services, $m^n$ combination of values have to be considered to locate those that conform to $gc$. Hence, the proposed approach uses local optimisation and approximation techniques to find such values. The way in which these techniques are extended to locate services that conform to functional constraints and relational constraints are described in the following.

**Functional Constraints**

I-Match performs two steps to locate services that conform to functional constraints. The first forms tuples of values by combining those represented in a set of slot lists. This is performed by removing empty slots from slot lists, selecting a value for each non-free attribute and combining them with those assigned to the free attribute. Values are selected using the optimisation services specified in a user request. The second forms composite services by finding tuples that consist of conforming values, and combining the services at the slots (in the slot lists) that correspond to these values.

In the previous phase, a slot list was generated for each restricted attribute. An empty slot in a slot list indicates that the corresponding value is not assigned to the relevant restricted attribute by any of the candidate services. Therefore, such values do not need to be considered when generating tuples and are removed from the slot lists. The proposed approach selects one value for each non-free attribute to reduce the number of generated value tuples. Let us consider a functional constraint $gc$ which restricts the attributes $\{a_x, \ldots, a_y\}$, where $|x, \ldots, y| = n$, $a_x$ is the free attribute, and $m$ values are assigned to each attribute $a_i \in \{a_x, \ldots, a_y\}$. In such a situation, I-Match would combine the values assigned to $a_x$ with the values selected for the remaining attributes and generate $m^n$ tuples. An exhaustive approach which considers all the values would generate $m^n$ tuples. For example, consider the constraint $productionTime.Computer + approvalPhase.Insurance$...
+ timeTaken.Delivery < 7 DAYS given in Section 5.2 and the services in Table 5.1. The lists
{3 DAYS, 4 DAYS, 5 DAYS}, {1 DAY, 2 DAYS, 3 DAYS, 5 DAYS} and {2 DAYS, 3 DAYS, 5 DAYS} are generated from the values assigned to the attributes productionTime.Computer, approvalPhase.Insurance and timeTaken.Delivery respectively. In this scenario, I-Match would select the values 1 DAY and 2 DAYS from those assigned to approvalPhase.Insurance and timeTaken.Delivery using the optimisation services. Then they would be combined with 3 DAYS, 4 DAYS and 5 DAYS (which are assigned to productionTime.Computer) to form [3 DAYS, 1 DAY, 2 DAYS], [4 DAYS, 1 DAY, 2 DAYS] and [5 DAYS, 1 DAY, 2 DAYS].

The proposed approach generates 3 tuples, whereas one that considers all the values would generate 36 tuples (3 × 4 × 3 - all possible combinations).

Once the tuples are generated, the values in each of them are combined using the aggregation service specified in a request, and evaluated. Then, if the values in a tuple conform to the constraint, the services at the corresponding slots (of the slot lists) are combined to form composite services. Let us consider a functional constraint gc which restricts the attributes \(a_x, \ldots, a_y\) of services of types \(S_x, \ldots, S_y\). A set of slot lists \(SL\) is generated for \(gc\). A slot at a location \(\langle a_i, v_i \rangle\) which corresponds to an attribute \(a_i\) and a value \(v_j\) contains services of type \(S_i\) which assign the value \(v_j\) to attribute \(a_i\). Therefore, if \([v_x, \ldots, v_y]\) is a tuple of values that conforms to \(gc\), then the services at the slots \(\langle a_x, v_x \rangle, \ldots, \langle a_y, v_y \rangle\) would form conforming composite services. For example, in the above scenario the values in [3 DAYS, 1 DAY, 2 DAYS] conform to the constraint productionTime.Computer + approvalPhase.Insurance + timeTaken.Delivery < 7 DAYS. Therefore, the services ComputerSales-II, Insurance-II and Shipping-II at the slots \(\langle productionTime.Computer, 3 DAYS \rangle, \langle approvalPhase.Insurance, 1 DAY \rangle\) and \(\langle timeTaken.Delivery, 2 DAYS \rangle\) are combined to form a composite service.

Algorithm 5.2 locates services that conform to functional constraints. This algorithm requires a functional constraint and a set of slot lists. First, it removes the empty slots (line 3), iterates through the slot lists (lines 4-7) and uses the optimisation services to select a value for each non-free attribute (line 5). Then, these values are combined with those assigned to the free attribute to form value tuples (line 9). Next, these value tuples are evaluated to determine whether they conform to the given constraint (line 10). If they do, services are retrieved from the relevant slots and combined to form composite services (lines 11-17). The following provides descriptions of the details returned by the functions used in this algorithm.

- \(aggregationService(gc)\): Aggregation service specified in the description of a constraint
CHAPTER 5. I-MATCH: MATCHING INDEPENDENT GLOBAL CONSTRAINTS

$gc$ (line 2).

- $\text{optimisationService}(a, gc)$: Optimisation service used to select a value for a restricted attribute $a$ of $gc$ (line 5).

- $\text{optimisedValue}(L, s)$: A value selected by optimisation service $s$ from a list $L$ (line 6).

```
1 locateCompositeServicesforFunctionalConstraints(SL, gc)
2 $s_a() \leftarrow \text{aggregationService}(gc)$;
3 removeEmptySlots(SL);
4 for $i = 1$ to $|SL|; i++$ do
5     $s_o \leftarrow \text{optimisationService}(gc, a_i);$;
6     $\text{values}[i] \leftarrow \text{optimisedValue}(SL[i], s_o);$;
7 end
8 for $i = 0$ to $|SL[0]|; i++$ do
9     $\text{values}[0] \leftarrow SL[0, i]$;
10    if $s_a(\text{values})$ conforms to $gc$ then
11       for each service $s_0$ in $SL[0, \text{values}[0]].\text{getServices()}$ do
12         for each service $s_{...}$ in $SL[..., \text{values} [...]].\text{getServices()}$ do
13             for each service $s_n$ in $SL[n, \text{values}[n]].\text{getServices()}$ do
14                 conforming_services.add($s_0, ..., s_n$);
15             end
16         end
17     end
18 end
```

Algorithm 5.2: Locating Composite Services that conform to Functional Constraints

The time complexity of Algorithm 5.2 is exponential. If a given constraint restricts $p$ attributes, there are $m$ conforming value tuples and each slot (in the set of slot lists) contains $q$ services, then $m(p^q)$ composite services would be generated. However, like the technique introduced in Chapter 4, this algorithm identifies tuples of conforming values in polynomial time and only considers services that assign those values to their restrict attributes. The complexity of this algorithm is “generally” polynomial ($mn$ where $n$ is the
number of conforming composite services). This is evident from the results (in Figures 5.13 and 5.15) obtained from the simulated experiments.

Relational Constraints

The proposed approach which locates services that conform to relational constraints consists of two steps. The first determines the values that conform to a given constraint. The second retrieves services that assign these conforming values to their restricted attributes and combines them to form composite services. I-Match defines two separate techniques to locate services that conform to relational constraints: an optimised approach and an approximate approach. The first extends a basic greedy algorithm (backtrack-free search algorithm [Freuder, 1982; 1985; Minton et al., 1990]) to determine the values that conform to a given constraint. However, since this approach is not complete, a second approach that approximates a given constraint using domain rules is developed. This approach converts a relational constraint to a strictly dependent global constraint, and guarantees that conforming values can be determined in polynomial time. The located composite services are returned with the domain rules that are used for the approximations. Unlike the proposed optimised approach, this one informs a user about the restrictions used to reduce the number of value combinations that are considered.

These approaches model a relational constraint as a directed acyclic graph \( G = (V, E) \), where \( V \) is a set of nodes and \( E \) is a set of arcs. The nodes of such a graph represent the attributes and the arcs model the relations between the attributes. That means, each arc and the two connected attributes model a binary attribute comparison of a constraint. Figure 5.5 depicts a graph modeling a constraint that restricts the attributes \( a_1 - a_6 \), where the binary attribute comparison are specified with the operators \( \{o_{12}, o_{13}, o_{24}, o_{34}, o_{35}, o_{46}\} \).

![Figure 5.5: Directed Acyclic Graph modeling a Relational Constraint](image)
Values that conform to a constraint are identified by generating instances of such graphs. A tuple of values assigned to the nodes of a graph is an instance. For example, the values \([v_1, v_6]\), where each value \(v_i\) is assigned to an attribute \(a_i\), forms an instance of the graph depicted in Figure 5.5. Such an instance conforms to a given constraint if each of its values adhere to the relations of the binary attribute comparisons (represented by the arcs). A value that conforms to a binary attribute comparison is referred to as an Arc Consistent Value.

**Definition 5.2 (Arc Consistent Value)** Let \(gc\) be a relational constraint that restricts the values assigned to attributes \(\{a_x, \ldots, a_y\}\), where \(a_i\) is a non-free attribute and \(a_i \in \{a_x, \ldots, a_y\}\). \(\{a_j, \ldots, a_k\}\) is the set of attributes connected to \(a_i\) with the arcs \(\{o_{ij}, \ldots, o_{ik}\}\), where \(\{a_j, \ldots, a_k\} \subseteq \{a_x, \ldots, a_y\}\). \([v_j, \ldots, v_k]\) is a tuple of values assigned to \(\{a_j, \ldots, a_k\}\). A value \(v_i\) assigned to \(a_i\) is arc consistent, if the relations represented by the arcs \(o_{ij}, \ldots, o_{ik}\) exist between \(v_i\) and the values in \([v_j, \ldots, v_k]\).

An instance of a graph that consists of arc consistent values is a tuple of conforming values. Such tuples are referred to as arc consistent instances.

**Definition 5.3 (Arc Consistent Instance)** Let \(gc\) be a relational constraint that restricts the values assigned to attributes \(\{a_x, \ldots, a_y\}\), where \(a_i\) is a non-free attribute and \(a_i \in \{a_x, \ldots, a_y\}\). A tuple of values \([v_x, \ldots, v_y]\) assigned to \(\{a_j, \ldots, a_k\}\) is an arc consistent instance if each value \(v_j\) in \(\{a_j, \ldots, a_k\}\) is locally arc consistent.

(A) Greedy Approach

Identifying an arc consistent instance is NP-hard [Freuder, 1982; 1985]. If a given relational constraint restricts \(n\) attributes and \(m\) arc consistent values can be assigned to each attribute, then \(m^n\) combinations of values may have to be considered to identify an arc consistent instance. Therefore, the proposed optimised approach (to locate services conforming to relational constraints) incorporates a greedy algorithm to complete this task in polynomial time \((m^2n)\). When generating an arc consistent instance, this algorithm first assigns a value to the free attribute and then, iterates through the remaining attributes assigning arc consistent values. This process is performed for each value that is assigned to the free attribute. If such a value cannot be located for a particular attribute, it does not backtrack and consider alternative (arc consistent value) assignments for other attributes. For example, consider the constraint \(availableDate.Computer \leq approvalDate.Insurance \leq date.Pickup\) described in Section 5.2 and the set of slot lists in Figure 5.6. Figure 5.7 shows the way in
which values are selected by the proposed optimised approach. A solid line indicates an arc consistent instance, whereas a dashed line indicates that the assignments have led to a dead-end (i.e. an arc consistent instance is not located for the corresponding value that is assigned to the free attribute). When \(4\text{TH.JULY}\) and \(5\text{TH.JULY}\) are assigned to \(\text{availableDate.Computer}\) and \(\text{approvalDate.Insurance}\) respectively, an arc consistent value cannot be located for \(\text{date.Pickup}\). In this situation the alternative arc consistent value \(4\text{TH.JULY}\) that can be assigned to \(\text{approvalDate.Insurance}\) is not considered by this algorithm.

![Sample Slot Lists](image)

**Figure 5.6: Sample Slot Lists**

![The Proposed Approach - Greedy Selection](image)

**Figure 5.7: The Proposed Approach - Greedy Selection**

Like the technique that deals with functional constraints, the optimised approach combines services in the slots (of the slot lists) that correspond to the values in an arc consistent instance, once such an instance is located. The algorithm for this approach is given below (in
Algorithm 5.3). This algorithm requires a set of slot lists and a relational global constraint \( gc \). First, the empty slots are removed from the slot lists (line 2). These slots need to be removed because the corresponding values are not assigned to the relevant restricted attributes by any candidate service. Then, for each value that is assigned to the free attribute (line 4) arc consistent values are located for the remaining attributes and placed in a value tuple (lines 5-13). Finally, if the value tuple is an arc consistent instance (i.e. an arc consistent value has been located for all the non-free attributes), services at the indicated slots are retrieved and combined to conforming composite services (lines 14-22). The following provides descriptions of the details returned by the functions of Algorithm 5.3.

- \( \textit{attributeComparisons}(gc) \): Attribute comparisons used to describe \( gc \) (lines 6 and 14).
- \( \textit{includes}(x, Y) \): Set of attribute comparisons in \( Y \) specified using attribute \( x \) (line 6).
- \( \textit{isArcConsistent}(X, Y) \): Checks if the values in a tuple \( X \) conform to the binary attribute comparisons in \( Y \) (line 9).
- \( \textit{isArcConsistentInstance}(X, Y) \): Checks if the values in a tuple \( X \) form an arc consistent instance according to the binary attribute comparisons in \( Y \) (line 14).

The time complexity of Algorithm 5.3 is exponential. If a given constraint restricts \( p \) attributes, there are \( m \) conforming value tuples, and each slot (in the set of slot lists) contains \( q \) services, then \( m(p^q) \) composite services would be generated. However, like the proposed technique which locates services that conform to functional constraints, Algorithm 5.3 identifies tuples of conforming values in polynomial time and only considers services that assign those values to their restrict attributes. The complexity of this algorithm is “generally” polynomial (\( mn \) where \( n \) is the number of conforming composite services). This is evident from the results (in Figures 5.17 and 5.19) obtained from the evaluation.

The main drawback of this approach is that it may not return a conforming composite service when one is available. The values considered by the greedy algorithm differ according to the ordering of slots in the slot lists. Let us consider the slot lists given in Figure 5.6. When the values are ordered as in Figure 5.8a it would identify two arc consistent instances, whereas if it is ordered as in Figure 5.8b none of them would be identified. This drawback can be invalidated by backtracking and considering alternative arc consistent values. For example, when the values are ordered as in Figure 5.8b, if alternative arc consistent values are considered, then two locally arc consistent paths can be identified as in Figure 5.9.
Algorithm 5.3: Locating Services that conform to Relational Constraints: Greedy Approach

However, the proposed approach does not consider alternative values because it increases the time-complexity of the process that locates arc consistent instances [Freuder, 1982; 1985].
CHAPTER 5. I-MATCH: MATCHING INDEPENDENT GLOBAL CONSTRAINTS

![Alternative Value Selections](image_url)

(a)

![Value Selection with Backtracking](image_url)

(b)

Figure 5.8: Alternative Value Selections

Figure 5.9: Value Selection with Backtracking
Next, we propose a different matching technique, which locates services that conform to relational constraints (the approximate approach). Since it is difficult to define a sound approach with polynomial time, this technique approximates a given relational constraint to a strictly dependent global constraint. Composite services that conform to such constraints can be located in polynomial time (if any are available). Relational constraints are approximated using domain rules and these rules are returned to users so that the context in which services are located can be understood. This technique consists of two steps. The first determines if a given relational global constraint is approximable. That means, it checks if a relational constraint can be mapped to a strictly dependent global constraint using domain rules. If it is approximable, the second locates services that conform to the approximated constraint.

A global constraint is strictly dependent if the values that should be assigned to all the restricted attributes can be uniquely determined based on a value that is assigned to one attribute. A relational global constraint is approximable if:

1. there is a set of domain rules which derive unique values for the non-free attributes based on a value that is assigned to the free attribute, and
2. the derived values form an arc consistent instance.

**Definition 5.4 (Approximable Relational Constraint)** Let $g_c$ be a relational global constraint which restricts the values assigned to attributes $\{a_x, \ldots, a_y\}$, where $a_x$ is its free attribute and $V_x$ is the set of values assigned to $a_x$. $\{r_m, \ldots, r_n\}$ is the set of available rules. $g_c$ is approximable if $\exists v_x \in V_x$ and a set of rules $\exists \{r_p, \ldots, r_q\} \subseteq \{r_m, \ldots, r_n\}$, which can derive the values $v_{x+1}, \ldots, v_y$ based on $v_x$, such that $[v_x, v_{x+1}, \ldots, v_y]$ forms an arc consistent instance.

A scenario in which the relational constraint represented in Figure 5.5 could be considered as approximable is shown in Figure 5.10. The constraint is approximable if the values $[v_2 \ldots v_6]$ assigned to the attributes $[a_2 \ldots a_6]$ are derived using the rules $\{r_{12}, r_{13}, r_{14}, r_{35}, r_{46}\}$, based on a value $v_1$ assigned to $a_1$, and $[v_1 \ldots v_6]$ form an arc consistent instance.

Let us consider the constraint $\text{availableDate.Computer} \leq \text{approvalDate.Insurance} \leq \text{date.-Pickup}$, the set of slot lists given in Figure 5.6 and the rules described in Figure 2.6. Figure 5.11 depicts an example of how domain rules can be used to derive arc consistent instances. The values that should be assigned to $\text{approvalDate.Insurance}$ and $\text{date.Pickup}$ are...
uniquely determined based on the values 3RD_JULY, 4TH_JULY and 5TH_JULY assigned to the free attribute availableDate.Computer. This results in three arc consistent instances being generated.

Figure 5.10: Approximable Constraint

Figure 5.11: Sample Approximable Constraint
A complete solution that approximates a relational constraint using domain rules takes exponential-time. Let us consider a constraint $gc$ that restricts the attributes $\{a_x, \ldots, a_y\}$, where $a_x$ is its free attribute and $|a_x, \ldots, a_y| = m$. For each attribute $a_i$ there is a set of rules $R_i$ where each $r_i$ is able to derive a value $v_i$ for $a_i$ based on a value $v_j$ assigned to $a_j$. $a_j \in \{a_x, \ldots, a_z\}$, $a_i \in \{a_{z+1}, \ldots, a_y\}$, $x \leq z < y$ and $|R_i| = n$. In such a situation $(m-1)^n$ combinations of rules may have to be considered to establish if $gc$ is approximable. Hence, the proposed approach uses a greedy approach to select rules. A particular rule $r_i$ is selected if it is able to derive a value $v_i$ for an attribute $a_i$ in $\{a_{z+1}, \ldots, a_y\}$ based on a value $v_j$ assigned to $a_j$, such that $v_i$ is arc consistent. Then, the technique proceeds to select a rule that derives a value for $a_{i+1}$. Any other rule that derives a value for $a_i$ is not considered.

Like the previous technique that locates services that conform to relational constraints, once an arc consistent instance is generated, the services in the slots (of the slot lists) that correspond to the values in the instance are combined to form composite services. The algorithm that determines whether a given constraint is approximable is given below (in Algorithm 5.4). This algorithm requires a relational constraint, the available domain rules and a set of slot lists. A given constraint is considered to be approximable if at-least one tuple that forms an arc consistent instance can be derived based on a value that is assigned to its free attribute. This algorithm attempts to generate such an instance for each value assigned to the free attribute. When generating an instance, it iterates through the available rules (line 6) and attempts to derive an arc consistent value for each non-free attribute (lines 7-15). If a derived value is arc consistent, the value and the rule used to derive it are stored in a temporary array (lines 13 and 14). Finally, each arc consistent instance is placed in a list along with the rules used for the derivations (lines 20-22). The following provides descriptions of the details returned by the functions used in this algorithm.

- $determinantAttribute(r)$: Determinant attribute of a rule $r$ (line 7).
- $derivedAttribute(r)$: Derived attribute of $r$ (line 8).
- $isSemanticallyRelated(a, b)$: Checks if attribute $a$ is semantically related to $b$ (line 9).

The complexity of Algorithm 5.4 is $m^2nq$, where $m$ is the number of values assigned to an attribute, $n$ is the number of restricted attributes and $q$ is the number of available domain rules that can be used to derive a value for a particular non-free attribute. This algorithm assumes that a finite number of domain rules are available. The number of restricted attributes
used to describe a constraint are small and its impact on the complexity of this algorithm is insignificant.

Algorithm 5.4: Approximating Relational Constraints

Algorithm 5.5 locates services that conform to an approximated relational constraint. A given relational constraint is approximable if the derived values list in Algorithm 5.4 is not empty (contains at-least one arc consistent instance). Algorithm 5.5 requires the de-
derived_values_list and the slot lists generated for the given relational constraint. An arc consistent instance (in the derived_values_list) indicates the values that should be assigned to the restricted attributes according to the approximated constraint. This algorithm iterates through the derived instances (line 3) and checks if slots that correspond to the values in them exist (line 7). If they do, the services in the slots are combined to form composite services (lines 11-17). If not, the instance is ignored (line 8). Each composite service is placed in a list along with the domain rules used to derive the corresponding arc consistent instance (line 14).

```
1 locateCompositeServicesforRelationalConstraints2(SL, derived_values_list)
2 removeEmptySlots(SL);
3 for each element e in derived_values_list do
4     values—e.getValues();
5     rules—e.getRules();
6     for j—0; j<|values|; j++ do
7         if ?SL[j, values[j]] then
8             go to line 3;
9         end
10     end
11     for each service s0 in SL[0, values[0]].getServices() do
12         for each service sn in SL[. . . , values[. . . ]].getServices() do
13             for each service sn in SL[n, values[n]].getServices() do
14                 conforming_services.add(⟨s0, . . . , sn⟩, rules);
15             end
16         end
17     end
18 end
```

Algorithm 5.5: Locating Services that conform to Approximated Relational Constraints

The time complexity of Algorithm 5.5 is exponential. If a given constraint restricts $p$ attributes, there are $m$ arc consistent instances in the derived_values_list, and each slot (in the set of $s$) contains $q$ services, then $m(p^q)$ composite services would be generated. However, like the technique that locates services that conform to functional constraints, this technique
is “generally” polynomial (\(mn\) where \(n\) is the number of conforming composite services). Algorithm 5.5 only considers a service if it can be combined with other services to form a conforming composite service (i.e. A service is only considered if it assigns a value in an arc consistent instance to its restricted attribute). This is evident from the results (in Figures 5.17 and 5.19) obtained from the evaluation.

The proposed techniques are better than existing approaches that consider independent global constraints because they are both semantic-based and able to locate conforming services in polynomial time. Hence, they are able to locate conforming composite services efficiently in situations where user requests and service descriptions are syntactically heterogeneous. However, like the technique introduced in Chapter 4, which retrieves services that conform to strictly dependent global constraints, the main drawback of these approaches is that they retrieve duplicate entries. A duplicate entry occurs if a composite service that includes a particular sequence of constituent services is retrieved more than once. They are retrieved because the constituent services assign multiple conforming values to their restricted attributes. Detailed descriptions of situations in which duplicate entries occur are given in Section 4.5.

5.4 Theoretical Analysis

This section provides an analysis of the soundness and completeness of the proposed approaches. We assume that the given descriptions (service descriptions and user requests) are accurate. The analysis performed here disregards any alternative interpretations of user requests. They are obtained by substituting and semantically relating terms in these descriptions using ontological relationships. Let us consider a request \(r\) specified with \(n\) elements (concepts, roles, operations and attributes). If each of these can be semantically related or substituted with \(m\) elements, then there are \(m^n\) possible interpretations of \(r\). Hence, the proposed approaches do not consider alternative interpretations. Descriptions of the notations used during this analysis are given in Table 4.2.

A composite service conforms to a request that includes an independent global constraint, if the constituent services are of appropriate types and the values assigned to their restricted attributes are conforming. The three proposed algorithms ensure that constituent services are of appropriate types by generating candidate service lists. In order to select constituent

---

4The semantics of soundness and completeness (when they are applied to service discovery techniques) are described in Section 4.6
services for composite services, first candidate services are indexed based on the values they assign to their restricted attributes. Then, tuples of values that conform to a given constraint are identified and services that assign those values to their restricted attributes are combined to form composite services. Therefore, whether a certain property (soundness or completeness) can be associated with a particular technique is determined by the process that identifies conforming value tuples.

**Soundness**

A service discovery technique $T$ is sound if all the located services conform to a given constraint $gc$. That means, $T$ is sound if

$$T(P) \subseteq R$$  \hspace{1cm} (5.1)

where $P$ is all possible combinations of composite services that can be generated with the given candidate services and $R \subseteq P$ that conforms to $gc$. $T$ achieves this property if the process $T^*$ used to identify conforming value tuples adheres to the following.

$$T^*(K) \subseteq Q$$  \hspace{1cm} (5.2)

where $K$ contains the set of tuples generated by combining the values assigned to the restricted attributes of candidate services, and $Q \subseteq K$ which contains tuples that conform to $gc$.

- **I-Match for Functional Constraints.** Let $gc$ be a constraint that restricts the attributes $a_x, \ldots, a_y$ and denotes $a_x$ as its free attribute. First, “optimisation services” (objective functions) are used to select a value for each non-free attribute of $gc$. That means a tuple $[v_{i+1}, \ldots, v_y]$ where each $v_i$ is assigned to $a_i$ is selected. Then, these values are combined with those assigned to the free attribute $a_x$ to generate a set of tuples $K' \subseteq K$. Then, the values in each tuple $k' \in K'$ is aggregated using an “aggregation services” and evaluated to check if they conform to $gc$ (i.e. $\exists k' \in Q$).

- **I-Match for Relational Constraints (Optimised Approach).** This technique uses a greedy algorithm to generate a set of arc consistent instances $K'$. An arc consistent instance is a tuple of values that conforms to $gc$. Hence, $K' \subseteq Q$.

- **I-Match for Relational Constraints (Approximate Approach).** This approach uses domain rules to derive a set of arc consistent instances $K'$. The set $K' \cap K \neq \emptyset$ since
values which are not assigned to the restricted attributes could be derived by the domain rules. However, any tuple (arc consistent instance) \( k' \in K' \) is discarded, if a service that assigns a value \( v \in k' \) does not exist. That means, every \( k' \in K' \), where \( \exists k' \in Q \) is discarded.

Therefore, the three proposed approaches are sound.

**Completeness**

A service discovery technique \( T \) is complete if it is able to locate all the available conforming services. That means, \( T \) is complete if

\[
R \subseteq T(P) \tag{5.3}
\]

\( T \) achieves this property if the process \( T^* \) used to identify conforming value tuples adheres to the following.

\[
Q \subseteq T^*(K) \tag{5.4}
\]

- **I-Match for Functional Constraints.** Let \( a_x \) be the free attribute of \( gc, [v_{x+1}, \ldots, v_y] \) the tuple that contains the values selected by the optimisation services and \( K' \) the set of conforming value tuples located by this approach. \( K' \subseteq X \) and \( X \subseteq K \), where \( X = \{ L_i \times [v_{x+1}, \ldots, v_y] \} \) because an optimisation service only selects one value for a non-free attribute. Thus, \( K' \subseteq Q \) and the completeness of this approach cannot be guaranteed.

- **I-Match for Relational Constraints (Optimised Approach).** This approach uses a greedy algorithm to generate a set \( K' \) of conforming value tuples. Hence, \( K' \subseteq Q \) and the reason for this is described in Section 5.3 (See Figures 5.8 and 5.9).

- **I-Match for Relational Constraints (Approximate Approach).** When \( Q \neq \emptyset \), this technique could derive a set of value tuples \( K' \), where \( K' = \emptyset \) because of any of the following reasons.

  1. All the arc consistent instances (generated by deriving values with domain rules) could be discarded because at least one value in each of them is not assigned to a restricted attribute by any service.
2. The algorithm that approximates a given relational constraint is not able to derive an arc consistent instance. Since this algorithm employs a greedy approach to select rules, those suitable may be overlooked.

3. The available domain rules are not sufficient to derive an arc consistent instance.

All three approaches are not complete.

5.5 Evaluation

This section focuses on a simulation based and a case study based evaluation of the proposed techniques. They are compared against a syntactic-based composite matching technique [Wu et al., 2003a;b] and an exhaustive approach (defined by extending the semantic-based approach in [Elgedawy et al., 2004a]). The reason for performing two separate evaluations will be described later.

5.5.1 Experiments

This evaluation compares implementations of the above mentioned techniques (proposed techniques, [Wu et al., 2003a;b] and [Elgedawy et al., 2004a]) for performance and recall. Recall is calculated based on the number of unique conforming composite services retrieved.

The technique proposed in [Wu et al., 2003a;b] is implemented with JSHOP\textsuperscript{5} and others were done in Java (JDK 1.5.0)\textsuperscript{6}. A MySQL 5.0 Community Server\textsuperscript{7} database is used as a registry to store service descriptions and ontological descriptions. Experiments were conducted on Pentium IV 3.0 GHz machines with 1 Gigabyte of memory. A data set which consists of sample service descriptions and ontological descriptions were generated using the method described in Section 4.7.

The experiments were separated based on the type of constraint (either functional or relational) included in a request. They were conducted in three environments.

1. Forced requests
2. Random requests
3. Restricted environments

\textsuperscript{5}https://sourceforge.net/projects/shop - Java version of SHOP.
\textsuperscript{6}http://java.sun.com
\textsuperscript{7}http://www.mysql.com
Experiments were conducted with forced requests and random requests to separately evaluate the proposed techniques in situations where syntactic-based approaches are capable and incapable of retrieving services. A syntactic-based technique would not be able to retrieve conforming composite services if the values assigned to an attribute used to specify a functional constraint and an attribute used to model the corresponding input of an optimisation or an aggregation service have different scopes. Let us consider the functional constraint \( \text{production-Time.Computer} + \text{approvalPhase.Insurance} + \text{duration.Shipping} < 7\text{.DAYS} \). The aggregation service included in the user request requires values assigned to \( \text{productionTime.Computer} \), \( \text{approvalPhase.Insurance} \) and \( \text{timeTaken.Delivery} \) as inputs. The values assigned to \( \text{productionTime.Computer} \), \( \text{approvalPhase.Insurance} \) and \( \text{timeTaken.Delivery} \) are specified in days and those assigned to \( \text{duration.Shipping} \) are specified in hours. In such a situation, a syntactic-based approach would not be able to retrieve any conforming composite services because the aggregation service cannot be used. The values assigned to \( \text{timeTaken.Delivery} \) and \( \text{duration.Shipping} \) have different scopes.

In a forced request generated with a functional constraint, the attributes that model the inputs of aggregation and optimisation services correspond to those that are used to specify the constraint. In a random request, these attributes are selected randomly. However, random requests were not considered for the experiments that consider relational constraints. Such a constraint cannot be evaluated if it is specified with attributes that have different scopes\(^8\). Let us consider the constraint \( \text{availableDate.Computer} \leq \text{approvalDate.Insurance} \leq \text{day.Available} \), where the values assigned to \( \text{availableDate.Computer} \) and \( \text{approvalDate.Insurance} \) are calendar dates\(^9\), whereas those assigned to \( \text{day.Available} \) are days of the week\(^10\). This constraint cannot be evaluated because days of the week cannot be compared against calendar dates directly. A forced request with a relational constraint is specified with attributes that have the same scope.

Experiments were conducted by varying the number of available service descriptions and service types in composite service templates. Those conducted with random and forced requests considered between 200 to 2000 service descriptions and composite service templates that include between 5 to 25 service types. Since the (exhaustive approach implemented with

\(^8\) Note that attributes that have different scopes can be compared by leveling them. A formal description of attribute leveling is given in Definition 4.5. However, attribute leveling is not considered when generating slot lists because aggregation and optimisation services required values that can be assigned to specific attributes.

\(^9\) e.g. 24/07/2007, 31/12/2010, 27/04/2008

\(^10\) e.g. Monday, Tuesday, Friday
the technique in [Elgedawy et al., 2004a] employed an exponential algorithm, it was not feasible to execute it in such an environment. Hence, the experiments with the exhaustive approach were conducted in a restricted environment, which contained between 20 to 100 service descriptions and composite service templates that include between 2 to 10 service types. They were conducted with forced requests.

Details of the experiments and the graphs that depict their results are indicated in Table 5.2.

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Environment</th>
<th>Services</th>
<th>Service Types</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional</td>
<td>Forced Requests</td>
<td>200 - 2000</td>
<td>5</td>
<td>Figure 5.12a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>5-25</td>
<td>Figure 5.14a</td>
</tr>
<tr>
<td></td>
<td>Random Requests</td>
<td>200 - 2000</td>
<td>5</td>
<td>Figure 5.12b</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>5-25</td>
<td>Figure 5.14b</td>
</tr>
<tr>
<td></td>
<td>Restricted</td>
<td>20 - 100</td>
<td>4</td>
<td>Figure 5.12c</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>2-10</td>
<td>Figure 5.14c</td>
</tr>
<tr>
<td>Relational</td>
<td>Forced Requests</td>
<td>200 - 2000</td>
<td>5</td>
<td>Figure 5.16a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>5-25</td>
<td>Figure 5.18a</td>
</tr>
<tr>
<td></td>
<td>Restricted</td>
<td>20 - 100</td>
<td>4</td>
<td>Figure 5.16b</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>2-10</td>
<td>Figure 5.18b</td>
</tr>
</tbody>
</table>

Table 5.2: Experiments and Results

**Functional Constraints**

In the experiments conducted with functional constraints, the exhaustive approach and the syntactic approach achieved the highest and lowest recall levels respectively. Unlike the syntactic approach, the semantic-based approaches matched syntactically heterogeneous service descriptions and user requests using ontological relationships. The exhaustive approach was able to retrieve more services than the proposed approach. The former approach considered all possible combinations of services and for each combination employed an exhaustive algorithm to evaluate the values assigned to the restricted attributes (i.e. check if the values assigned to the restricted attributes conform to a given constraint). The proposed approach only considered a limited number of values when locating services.
Figure 5.12: Recall - Functional Constraints - Varying no. of Available Services

On average, the proposed approach performed 88% better than the syntactic approach in the experiments that varied the number of available services. The syntactic approach used JSHOP which performs a combinatorial depth-first search to locate conforming composite
services. On the other hand, the proposed technique determined the values that conform to a given constraint in polynomial time and only combined services that assign those values. The syntactic technique performed better than the proposed approach when the number of available services varies. 

Figure 5.13: Time Taken - Functional Constraints - Varying no. of Available Services
of services types in composite service templates increased. In the worst-case (25 service types), it performed 48% better with forced requests and 25% better with random requests. The proposed approach retrieved services that assign conforming values to their restricted

Figure 5.14: Recall - Functional Constraints - Varying no. of Service Types
attributes, and then obtained their Cartesian Product to form composite services. Hence, the time taken increased at a rapid rate when the number of service types was increased. On the other hand, the syntactic was not able identify many candidate services. Therefore,
the number of services that had to be combined were limited. The exhaustive approach was the least efficient in the restricted environment. In the two worst cases (with 100 available services and 10 service types) it took around 21 minutes and 28 minutes respectively, whereas the proposed approach only took 7 seconds and 12 seconds.

**Relational Constraints**

The recall levels achieved by the exhaustive approach and the proposed optimised approach were higher than that achieved by the syntactic approach in the experiments conducted with relational constraints. The proposed approximate approach did not retrieve many services because the rules required to approximate a given constraint were not available (i.e. the used random process did not generate the required rules). However, unlike the experiments that considered functional constraints, the number of conforming services retrieved by all the techniques decreased as the number of service types increased. In these experiment, the number of binary attribute comparisons in requests were correlated to the number of service
types. Hence, the values assigned to the restricted attributes of candidate services had to conform to an increasing number of binary attribute comparisons.
Like in the experiments conducted with functional constraints, the following trends were present in those conducted with relational constraints for the same reasons. The performance of the proposed optimised technique was 76% better than that of the syntactic approach in the experiments that varied the number of available services. The performance of the syntactic approach was 32% better in the experiments that increased the number of service types in a composite service template. The exhaustive approach was the least efficient in the restricted environment.

5.5.2 Case Study

Here we describe a case study to evaluate the proposed techniques that deal with relational constraints. The advantages of using the proposed derivation-based algorithm to deal with relational constraints were not shown through the simulation experiments because of the
inability to generate the required rules. This case study clearly shows these advantages.

This case study analyses the proposed techniques, a syntactic-based composite match-
CHAPTER 5. I-MATCH: MATCHING INDEPENDENT GLOBAL CONSTRAINTS

ing technique [Wu et al., 2003a;b] and an exhaustive approach (defined by extending the semantic-based approach in [Elgedawy et al., 2004a]). It uses a scenario where a user attempts to purchase a computer with web services. The request described in Section 5.2 (which contains the constraint \( \text{availableDate.Computer} \leq \text{approvalDate.Insurance} \leq \text{date.Pickup} \)) is issued by the user, and the service descriptions in Figure 5.20 and the ontological descriptions in Table 5.3 are available. Additionally, the rules in Figure 2.6 are given (The three rules are referred to as \( r_1 \), \( r_2 \) and \( r_3 \) as indicated).

![Figure 5.20: Available services](image)

The semantic-based techniques first locate the candidate services and they are given in Table 5.4. Next, the steps performed by the three semantic-based techniques are described separately.

- **I-Match for Relational Constraints - Optimised Approach.** This technique generates the set of slot lists given in Figure 5.21. Then, the arc consistent instances are determined with a greedy algorithm. However, such instances cannot be identified in this scenario because of the way in which the slots are ordered in the slot lists. Hence, this technique is not able to locate any conforming composite services.

- **I-Match for Relational Constraints - Approximate Approach.** Like the previous approach, this technique generates the slot lists in Figure 5.21. Then, the given con-
strait is approximated and arc consistent instances are generated. The domain rules $r_2$ and $r_3$ are used to derive the instances $\langle 03-07-2007, 04-07-2007, 05-07-2007 \rangle$, $\langle 07-07-2007, 08-07-2007, 09-07-2007 \rangle$, and $\langle 04-07-2007, 05-07-2007, 06-07-2007 \rangle$. Finally,
the services that assign these values to their restricted attributes are combined to form the services \([\text{ComputerSales-I}, \text{Insurance-III}, \text{Shipping-III}]\) and \([\text{ComputerSales-III}, \text{Insurance-II}, \text{Shipping-I}]\). Each of these composite services are returned with rules \(r_2\) and \(r_3\) used to approximate the given constraint.

- **Exhaustive technique.** This technique generates all the possible combinations of candidate services. Since there are 3 service types in the given composite service template and 3 candidates of each type \(3^3\) combination services would be considered for this scenario. In each combination of services, the restricted attributes and the sets of values assigned to them are identified. These sets of values are combined to check if any of the combination conform to the given constraint. If they do, the corresponding combination of services is returned as a conforming composite service. This technique retrieves 10 conforming composite services \(([\text{ComputerSales-I}, \text{Insurance-II}, \text{Shipping-I}], [\text{ComputerSales-I}, \text{Insurance-II}, \text{Shipping-III}], [\text{ComputerSales-I}, \text{Insurance-III}, \text{Shipping-I}], [\text{ComputerSales-I}, \text{Insurance-III}, \text{Shipping-III}], [\text{ComputerSales-III}, \text{Insurance-II}, \text{Shipping-I}], [\text{ComputerSales-III}, \text{Insurance-II}, \text{Shipping-III}], [\text{ComputerSales-III}, \text{Insurance-III}, \text{Shipping-I}], [\text{ComputerSales-III}, \text{Insurance-III}, \text{Shipping-III}], and [\text{ComputerSales-III}, \text{Insurance-III}, \text{Shipping-III}])

Next, the steps performed by the syntactic technique are described. First, this technique generates a SHOP-2 method according to the structure of a given relational constraint. Then, the available service descriptions and the user request are converted to operator instances. Finally, the operator instances are matched (unified) to the method to locate composite services. Since this technique is not capable of performing any ontological reasoning, the operation, affected concept and role that describe the services need to be syntactically similar to those specified in the composite service template. Also, the restricted attributes of candidate services need to be described with the same attributes that specify the given constraint. Only \(\text{ComputerSale-I}, \text{Shipping-I}\) and \(\text{Insurance-I}\) satisfy these requirements. However, they do not form a conforming composite service because the values assigned to their restricted attributes do not conform to the given constraint.

In this scenario, the proposed optimistic approach and the syntactic approach were not able to retrieve any composite services. The former was not able to generate any arc consistent instances and the latter was not able to match syntactically heterogeneous descriptions. The remaining techniques were able to locate conforming services. The exhaustive approach
achieved the highest recall. However, unlike this approach, the proposed approximate technique located services in polynomial-time. Additionally, the rules used to approximate the given constraint were returned along with the service. Hence, the user would be able to understand the context in which the services were located.

5.6 Summary

This chapter proposes sound semantic-based matching algorithms that locate composite services that conform to independent global constraints. Such constraints can be categorised into various types based on aspects such as arity, the techniques used to solve them and the complexity of those techniques. Two types of independent global constraints are considered in this chapter. They are relational (binary) and functional (non-binary locally optimisable).

First, candidate services are located to ensure that the constituent services of a composite service are of appropriate types. Then, tuples of values that conform to a given constraint are located from those assigned to the restricted attributes of candidate services. Finally, services that assign these conforming values are combined to form composite services.

Optimisation techniques are used to locate tuples of values that conform to given constraint since this is a typical constraint satisfaction problem. Such techniques require the lists of values assigned to each restricted attribute. These lists are generated by indexing services based on the assigned values in a two dimensional data structure (a set of slot lists). Such a data structure also enables the quick retrieval of services when combining those that assign conforming values to form composite services. The technique that deals with functional constraints uses optimisation services (i.e. objective functions) to find conforming values in polynomial time. Two approaches that locate services that conform to relational constraints are developed. The first uses a greedy algorithm to identify conforming values. However, since this approach is not complete, a second approach that approximates a given constraint is proposed. This approach uses domain rules to derive values that conform to a given constraint. The domain rules used to derive the conforming values are returned with each service so that the context in which it is located can be understood by a user.

Results obtained from simulation experiments and a case study indicate that the devised approaches achieve higher recall than a syntactic-based approach. They are more scalable than any existing semantic-based matching technique considering independent global constraints.
Chapter 6

VGC: Generating Valid Global Communication Models

As the range of services available on the Web increase, new value added services can be created by composing existing ones. It is then vital to ensure that compositions of web services are free from errors such as deadlocks and synchronisation conflicts. Current techniques are lacking in this regard because they either (i) do not consider all the different types of temporal relationships that exist between interactions, or (ii) do not support all types of interactions (i.e. only send and receive, not service and invoke). This chapter introduces an approach that overcomes these problems [Gooneratne et al., 2008]. First, a communication model is generated by composing interactions of constituent services. A composability model checks if interactions are of compatible types, and matches the parameters and the channels used to perform them. Then, the temporal relationships between all the interactions of the communication model are found using a reasoning mechanism. While doing so, these relationships are compared against those specified in descriptions of interaction protocols, to detect any deadlocks or synchronisation conflicts.

6.1 Motivation

Since a service can be programmatically invoked by another, new value added services can be created by combining existing ones. These value added services are referred to as “Composite Services”. Unlike atomic services, composite ones cannot be executed immediately after they are located. The interactions of constituent services need to be composed [Carman et al.,
CHAPTER 6. VGC: GENERATING VALID GLOBAL COMMUNICATION MODELS

2003; Fileto et al., 2003; Gao et al., 2006; Liu et al., 2004; Medjahed and Bouguettaya, 2005; Yang et al., 2005], and verified [Foster et al., 2003; Narayanan and McIlraith, 2002; Woodman et al., 2004] in order to determine whether they can be executed without errors (e.g. unspecified receptions, deadlocks, synchronisation conflicts).

Techniques used to discover, compose and verify composite services require “accurate” service descriptions [Elgedawy et al., 2004a]. Such descriptions are of two types: functional and behavioural. The transaction performed by a service is specified in a functional description, and its interaction protocols are specified in a behavioural description. An interaction protocol models both the interactions as well as the temporal relationships that exist between interactions. Interactions of web services are of four types: send, receive, invoke and service [Woodman et al., 2004]. These can be divided into two classes: (i) interactions that take place at time points (send and receive) and (ii) interactions that take place during time intervals (invoke and service). All four types of interactions and the time taken by them have to be considered when specifying the temporal relationships of an interaction protocol. If not, descriptions of interaction protocols, global communication models (formed by composing interactions of consistent services) and the results yielded by techniques that verify these models would be inaccurate. Let us extend Scenario 1.1 to illustrate these issues. This scenario depicts a user purchasing a computer using online services. The interaction protocols of located ComputerSales-I, Shipping-I and Insurance-I services are shown in Figures 6.1a, 6.1b and 6.1c respectively. A communicative action that dispatches a parameter is indicated with a solid circle and one that accepts a parameter is indicated with an empty circle.

Modeling either an invoke or a service interaction with a send and a receive can be problematic when the same parameter is dispatched or accepted more than once [Woodman et al., 2004]. Insurance-I contains two interactions providing insurance quotes to ComputerSales-I and Shipping-I. Both interactions accept an Insurance Quote Request and dispatch an Insurance Quote. It is difficult to associate an Insurance Quote with the corresponding Insurance Quote Request if these interactions are described with send and receive interactions.

Phantom deadlocks can be detected and (real) deadlocks may be missed when temporal aspects of interactions are ignored. The model shown in Figure 6.2a would be formed by composing the interaction protocols of Figure 6.1 when the temporal aspects of interactions are not considered. A solid circle represents a “send” (interaction), an empty circle a “receive”, and a solid square a “service” and an empty square an “invoke”\(^1\). Get Consignment

\(^1\)Note that in Figure 6.2 we model Provide Insurance Quote in Insurance-I as a service rather than a receive

151 (October 2008)
Note of Computer Sales-I is composed with receive Shipping Request and send Consignment Note of Shipping-I. A deadlock would occur if Get Consignment Note in ComputerSales-I is performed before Get Insurance Policy since the one that sends a Consignment Note in Shipping-I can only be performed after Get Insurance Policy. A similar error occurs even if Get Consignment Note is performed after Get Insurance Policy since the interaction that receives a Shipping Request in Shipping-I needs to be performed before Get Insurance Policy. On the other hand, the model depicted in Figure 6.2b can be generated if the temporal aspects of interactions are considered. This model is free of deadlocks and synchronisation conflicts because both the order in which the interactions are performed as well as the order-followed by a send, and similarly for Get Insurance Quote, Get Insurance Policy and Get Consignment Note.
ing constraints in the interaction protocols of the constituent services do not conflict. Such models are referred to as Valid Global Communication Models.

Current techniques cannot compose constituent services and form valid global communication models because they use inaccurate descriptions that either do not consider all four types of interactions, or do not consider the time taken by them when modeling the temporal relationships (e.g. behavioural descriptions created with frameworks such as the process model of OWL-S [Ankolekar et al., 2001; Martin et al., 2004; McIlraith and Martin, 2003], the scenarios of PILLAR [Elgedawy, 2003; Elgedawy et al., 2005; 2004b] BPEL4WS [König et al., 2008; Shen et al., 2005], WSCL [Banerji et al., 2002] and WS-choreography [Burdett and Kavantzas, 2004] or formalisms such as finite state machines [Berardi et al., 2005; 2003], \( \pi \)-calculus [Sangiorgi and Walker, 2001], Calculus of Communicating Systems (CCS) [Sangiorgi and Walker, 2001] and Communicating Sequential Processes (CSP) [Sangiorgi and Walker, 2001]). Hence, existing verification techniques for deadlock and synchronisation conflicts detection return false positives and false negatives. A false positive occurs if a valid global communication model is detected as one that contains a deadlock or synchronisation conflict and a false negative occurs in a converse scenario.

This chapter describes the VGC approach to deal with the issues discussed above and generates valid global communication models of composite services. Global communication models are produced by composing interactions protocols described with WS-\( \pi \)-calculus. Unlike existing behavioural description frameworks, WS-\( \pi \)-calculus can create descriptions that consist of all four types of interaction types and accurately models the temporal relationships between interactions.

First, conversations are formed by composing the interactions of constituent services. The temporal relationships specified in interaction protocols are not considered when forming conversations. Interactions are composed based on a composability model. This model checks (i) the type compatibility of interactions and (ii) the matching of communication channels and parameters. Then, sets of conversations (that consist of all the interactions of constituent services) are formed to ensure that a derived global communication model is free of unspecified receptions. Such sets of conversations are referred to as Complete Conversation Sets (CCS). An unspecified reception occurs when a parameter dispatched by an interaction is not accepted by another. Such errors cause deadlocks to occur in global communication models. Next, temporal relationships between the interactions of a CCS are found using an Interval Time Logic (ITL)-based transitive temporal reasoning mechanism [Allen,
CHAPTER 6. VGC: GENERATING VALID GLOBAL COMMUNICATION MODELS

(a) Temporal aspects of Interactions not considered

(b) Temporal aspects of Interactions considered

Figure 6.2: Global Communication Models
1983]. While doing so, deadlocks and synchronisation conflicts that could occur are detected by identifying any inconsistency in the temporal relationships. Inconsistencies occur when the temporal relationships between any two interactions of a global communication model conflicts with those specified in an interaction protocol of a constituent service. Finally, a concise specification of a global communication model is obtained by grouping (generalising) the temporal relationships between the interactions. The correctness of VGC is theoretically analysed using a sample scenario. We map specifications of global communication models to \( \pi \)-calculus terms and show how the proposed approach correctly determines situations where deadlocks and synchronisation conflicts do and do not occur using reaction rules.

The rest of the chapter is organised as follows. The ITL-based transitive temporal reasoning mechanism used to identify the relationships between interactions of global communication models is described in 6.2. Section 6.3 provides guidelines on how specifications of global communication models should be structured. Section 6.4 describes the proposed technique that generates such specifications. Section 6.5 provides details of the theoretical analysis. Existing techniques that generate and verify global communication models of composite services are reviewed in Section 6.6. Section 6.7 outlines the advantages and limitations of the proposed approach. Finally, we summarise the contributions of this chapter in Section 6.8.

6.2 Interval Time Logic-based Transitive Temporal Reasoning

Here, we provide a brief description of the technique in [Allen, 1983] that computes the transitive relationship between two time interval based interactions. This technique is used by the proposed approach VGC to find the temporal relationship between the interactions of a global communication model. VGC detects deadlocks or synchronisation conflicts by comparing temporal relationships between interactions of a global communication model against those specified in the interaction protocols of constituent services.

Given three time interval-based interactions \( i_a, i_b \) and \( i_c \), where the temporal relationships between \( i_a \) and \( i_b \), and \( i_b \) and \( i_c \) are known, Allen in [Allen, 1983] determines the transitive relationship between \( i_a \) and \( i_c \). This transitive relationship is determined using the Transitivity Table (Table 6.1)\(^2\). For example, if \( s(i_a, i_b) \) and \( oi(i_b, i_c) \), then the transitive relationship between \( i_a \) and \( i_c \) is either \( oi \), \( d \) or \( f \). In situations where there is uncertainty about the relationships the union of all the possible outcomes is considered as the transitive relationship.

\(^2\)The notations used to model different types of temporal relationships are given in Table 3.2.
For example, if $s(i_a, i_b) \cup m(i_a, i_b)$ holds, then the transitive relationship between $i_a$ and $i_c$ would be either $oi$, $d$, $f$, $o$ or $s$.

<table>
<thead>
<tr>
<th>$i_a, r_1, i_b$</th>
<th>b</th>
<th>bi</th>
<th>d</th>
<th>di</th>
<th>o</th>
<th>oi</th>
<th>m</th>
<th>mi</th>
<th>s</th>
<th>si</th>
<th>f</th>
<th>fi</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>b</td>
<td>-</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>bi</td>
<td>b</td>
<td>bi</td>
<td>mi</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
<td>bi</td>
<td>d</td>
<td>-</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>di</td>
<td>b</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>o</td>
<td>b</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>oi</td>
<td>b</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>m</td>
<td>b</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>mi</td>
<td>b</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>s</td>
<td>b</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>si</td>
<td>b</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>f</td>
<td>b</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>fi</td>
<td>b</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
</tr>
</tbody>
</table>

Table 6.1: Transitivity Table for Temporal Reasoning
6.3 Specification of Global Communication Models

VGC generates a global communication model of a composite service by composing interactions of the constituent services. Concise guidelines on how such models should be structured are provided here. A global communication model of a composite service specifies both the conversations that take place between constituent services and the temporal relationships that exist between these conversations.

A conversation takes place between two services if an interaction of one (service) can be composed with one or two interaction(s) of the other. Note that in certain situations a single invoke or service interaction could be composed with two other interactions (a send and a receive) since it performs two communicative interactions. So in other words, the conversation is between one interaction and one other unless the first is a service or invoke and the second is not, meaning that it has to be send followed by receive (if the first one is service) or receive followed by send (if the first one is invoke). Hence, if the first interaction is service (respectively invoke), then the second interaction must be either invoke (respectively service) or a combination of send and receive.

Definition 6.1 (Conversation) Given two distinct services $s_a$ and $s_b$ with interactions $i_a$ (of $s_a$), $i_b$ and $i'_b$ (of $s_b$):  

1. $i_a$ and $i_b$ form a conversation if $i_a$ can be composed with $i_b$

2. $i_a$, $i_b$ and $i'_b$ form a conversation if $i_a$ can be composed with $i_b$ and $i'_b$

For example, the conversations in the global communication model depicted in Figure 6.2b are specified as follows:

$$c_1 = [cs_2, in_1]; c_2 = [cs_3, in_3, in_5]; c_3 = [cs_4, sh_1, sh_4]; c_4 = [sh_2, in_2]; c_5 = [sh_3, in_4]$$

The temporal relationships between the conversations of a global communication model are specified using the language constructs of WS-$\pi$-calculus (given in Definition 3.5).

Definition 6.2 (Global Communication Model) A global communication model $C$ is defined as follows, where $c$ is a conversation, and $r$ defines a temporal relationship between
CHAPTER 6. VGC: GENERATING VALID GLOBAL COMMUNICATION MODELS

\[ C \quad r(C, c) \mid c \]
\[ r \quad b \mid bi \mid d \mid di \mid o \mid oi \mid m \mid mi \mid s \mid si \mid f \mid fi \]

\( s(b(fi(c_1, c_4), d(c_2, c_5), c_3) \) is a specification of the global communication model in Figure 6.2b.

6.4 Generating Global Communication Models

Here we show how a global communication model of a composite service is generated. VGC makes the following assumptions: (i) the constituent services of a composite service are located using a service discovery technique, (ii) the interaction protocols of services are specified using WS-π-calculus, (iii) the ontological descriptions are structured according to the Meta-Ontology and (iv) the interaction protocols are valid\(^6\). Given a set of interaction protocols \{ \( p_x, \ldots, p_y \) \} of constituent services \( s_x, \ldots, s_y \) of a composite service \( cs \), the proposed technique generates a specification of a valid global communication model \( gc \) of \( cs \).

VGC is divided into four steps.

**STEP 1.** First, all the conversations that could take place between the constituent services are determined and stored in a set of conversation lists. A conversation indicates how two or more interactions of constituent services should be composed and ensures that parameters can be exchanged between composed interactions.

As service definitions may contain choice constructs, it is important to ensure that interactions are not taken from different execution paths. This is because interactions from different paths cannot be performed together in a single execution of a service (as they correspond to different choices made during execution). An execution path denotes a sequence of interactions that can be used to achieve a functionality provided by a web service (see Definition 3.4). Each web service could consist of multiple execution paths. For this reason we need to work with combinations of execution paths, rather than the direct definitions of each service.

**Definition 6.3 (Combination of Execution Paths (CEP))** Let \( cs \) be a composite service formed with the constituent services \( \{ s_1, \ldots, s_n \} \), where each \( s_i \) contains a set

---

\(^6\)WS-π-calculus descriptions that model valid interaction protocols are formally specified in Definition 3.5.
of execution paths $E_i$. A CEP is a tuple $[e_1, \ldots, e_n]$, where each execution path $e_i$ belongs to $E_i$.

Given the execution paths $\{E_1, \ldots, E_n\}$ of a composite service $[s_1, \ldots, s_n]$, where set $E_i$ contains the executions paths of a service $s_i$ and $|E_i| = n$, VGC combines the executions paths (i.e. $E_1 \times \cdots \times E_n$) and generates $m^n$ combinations of execution paths.

Next, a conversation list is generated for each CEP. Such a list stores all the conversations that can take place between interactions of the execution paths in a CEP.

**STEP 2.** The sets of conversations that form Complete Conversation Sets (CCSs) are located from each conversation list. A CCS is a set of conversations containing all the interactions of a CEP. Let $f()$ be a function that returns the set of interactions included in an execution path or a conversation (which we will call an interaction function). Given a CEP $P$ and a set of conversations $C$, where $P = \{e_1, \ldots, e_n\}$ and $C = \{c_1, \ldots, c_m\}$, $C$ is a CCS of $P$ if

\[
\begin{align*}
    f(e_1) \cup \cdots \cup f(e_n) &\equiv f(c_1) \cup \cdots \cup f(c_m) \\
    1 \leq i, j \leq m, f(c_i) \cap f(c_j) &\equiv \emptyset
\end{align*}
\]

A CCS ensures that a global communication model does not have any misses or overlaps. A miss occurs in a global communication model if this does not contain all the interactions of a CEP. An overlap occurs if either multiple interactions accept a single dispatched parameter or parameters dispatched by multiple interactions are accepted by a single interaction. Deadlocks and unspecified receptions occur in global communication models if there are overlaps or misses [Woodman et al., 2004].

**STEP 3.** In the third step, those CCSs forming global communication models with errors (i.e. deadlocks and synchronisation) are located. This is performed by comparing the temporal relationships between the interactions of a global communication model against those specified in interaction protocols of constituent services. The relationships between interactions are found using the reasoning mechanism described in Section 6.2, and these are stored in a Relations List.
CHAPTER 6. VGC: GENERATING VALID GLOBAL COMMUNICATION MODELS

**STEP 4.** A specification of a global communication model is derived from a CCS. Concise and accurate specifications are required when executing composite services [Nanda et al., 2004]. These specifications are derived from the details in Relations Lists.

We describe each of these steps in more detail in the subsections that follow.

### 6.4.1 Generating Conversation Lists

Given a set of interaction protocols (of the constituent services), here we will show how to construct a conversation list for each CEP. A conversation list contains all the conversations that can take place between the interactions of a CEP. Composed interactions form a conversation if they conform to all of the following conditions:

1. their parameters match
2. they are of compatible types
3. they are performed through entailed channels

**Parameter Matching** : Parameters of interactions need to be matched to ensure that corresponding parameters are dispatched and accepted by the communicating entities. A dispatched parameter matches an accepted parameter or vice versa if one can be substituted with the other. That means, given two parameters $p$ and $p'$, $p$ matches $p'$ if either $p \leadsto p'$ or $p' \leadsto p$. For example, the parameter $\text{noteRequest.Consignment}$ of the 4th interaction of $\text{ComputerSales-I}$ can be matched to $\text{request.Shipping}$ of the 1st interaction of $\text{Shipping-I}$ if either $\text{noteRequest.Consignment} \leadsto \text{request.Shipping}$ or $\text{request.Shipping} \leadsto \text{noteRequest.Consignment}$.

**Compatibility Checking** : An interaction dispatching a parameter indicates that a service produces one, and an interaction accepting a parameter indicates that a service expects one. Composed interactions need to be of compatible types in order to ensure that a dispatched parameter is accepted by another, and one expected by a service is produced by another. Otherwise, unspecified receptions or trivial deadlocks would occur [Peng and Purushothaman, 1989]. For example, if the 4th interaction of $\text{Insurance-I}$ in Figure 6.2a

\[ p \leadsto p' \] denotes that $p$ substitutes $p'$ according to the descriptions in a Meta-Ontology [Elgedawy et al., 2004a; 2008].
sends a *Shipping Quote* instead of receiving one, and this interaction is composed with the one that sends a *Shipping Quote* in *Shipping-I*, then unspecified receptions would occur at both *Insurance-I* and *Shipping-I*, because both interactions produce parameters and neither expects one. Similarly, if the 3rd interaction of *Shipping-I* receives a *Shipping Quote* instead of sending one, and this interaction is composed with the one that receives a *Shipping Quote* in *Insurance-I*, there would be a deadlock since both interactions expect a *Shipping Quote*, but neither produces one. Composed interactions are compatible if the communicative actions that they perform allow them to co-exist. That means, two interactions (say $i_a$ and $i_b$) are compatible if one of the following statements hold:

(a) $i_a$ is of type *send* and $i_b$ is of type *receive*,
(b) $i_a$ is of type *receive* and $i_b$ is of type *send*,
(c) $i_a$ is of type *invoke* and $i_b$ is of type *service*, or
(d) $i_a$ is of type *service* and $i_b$ is of type *invoke*.

*Invoke* and *Service* interactions perform two communicative actions: one that dispatches a parameter and another that accepts a parameter. Hence, an interaction of either type can be composed with a *send* and a *receive*. Three interactions $i_a$, $i_b$ and $i_c$ are compatible if one of the following holds:

(e) $i_a$ is of type *invoke*, and $i_b$ and $i_c$ are of types *receive* and *send* respectively, or
(f) $i_a$ is of type *service*, and $i_b$ and $i_c$ are of types *send* and *receive* respectively.

**Entailed Channels:** When performing an interaction, a channel is formed to ensure that parameters are dispatched or/and accepted between the correct entities. In WS-$\pi$-calculus, these channels are described by specifying the entities involved in an interaction. Let us consider a scenario where there is an interaction $i$ performed by a service $s_a$ and $i'$ performed by $s_b$. The WS-$\pi$-calculus description of $s_a$ indicates that Interaction $i$ is performed with a service of type $S_x$ and the description of $s_b$ indicates that $i'$ is performed with a service of type $S_y$. In such a situation, the channel $[s_a, S_x]$ used to perform $i$ is entailed by the channel $[s_b, S_y]$ used to perform $i'$, if $s_a$ is of type $S_y$, and $s_b$ is of type $S_x$. Given a service $s$ and a type $S$ where the purpose\(^8\) of $s$, and $S$ are

\[^8\]The purpose of a service models the performed transaction (see Chapter 3).
described with the operations $O$ and $O'$, roles $R$ and $R'$ and affected concepts $C$ and $C'$, $s$ is of type $s'$ if (i) $O \xrightarrow{s} O'$, (ii) $R \xrightarrow{s} R'$, and (iii) $C \xrightarrow{s} C'$.

For example, the purpose of Insurance-I and ComputerSales-I is specified as [Insure(), Computer, Insurance_Agent] and [Sales(), Computer, Sales_Assistant] respectively. The WS-$\pi$-calculus description of ComputerSales-I indicates that its 2nd interaction is performed with a service of type [Process(), Item, Insurance_Representative] and the description of Insurance-I states that the 1st interaction is performed with a service of type [Retail(), Laptop, Sales_Agent]. The channel used to perform the 2nd interaction of ComputerSales-I is entailed by the one used to perform the 1st interaction of Insurance-I, if (i) Sales() $\xrightarrow{s}$ Retail(), Computer $\xrightarrow{s}$ Laptop, and Sales_Assistant $\xrightarrow{s}$ Sales_Agent, and (ii) Insure() $\xrightarrow{s}$ Process(), Computer $\xrightarrow{s}$ Item, and Insurance_Agent $\xrightarrow{s}$ Insurance_Representative.

Once a conversation is formed this is stored in a conversation list. The following example shows a conversation list generated from the interaction protocols depicted in Figure 6.1:

$[cs_2, in_1], [cs_2, in_2], [cs_3, in_3, in_5], [cs_3, sh_1, sh_4], [sh_2, in_2], [sh_2, in_1][sh_3, in_4]$

Algorithm 6.1 (see below) generates a Conversations List. This algorithm requires a CEP and returns a Conversations List. It iterates through the interactions of each execution path and extracts those which are compatible in other paths (lines 2-7). Then, the interactions are compared to see whether they are composable. Those which are composable, are placed in a Conversations List (line 8). When dealing with invoke and services interactions of a selected execution path, it attempts to compose them with the send and receive interactions (lines 10-26). This algorithm has a polynomial time complexity ($mn$, where $m$ is the number of constituent services in a composite service and $n$ is the number of interactions in each execution path). The following is a brief description of the functions used in this algorithm.

- $\text{retrieveInteractions}(e, t)$: Returns interactions of type $t$ in execution path $e$ (lines 6, 11 and 12).
- $\text{isComposable}(i, i')$: Checks if interactions $i$ and $i'$ are composable (line 8).
- $\text{isComposable}(i, a, i')$: Checks if interaction $i'$ is composable with communicative action $a$ of $i$ (lines 14 and 17).

\footnote{Note that there is only one CEP to consider since none of the interaction protocols contain choice constructs.}
Algorithm 6.1: Generating a Conversations List
6.4.2 Complete Conversation Sets (CCSs)

This section describes how Complete Conversation Sets (CCSs) are located from a conversation list. We model a conversation list \( L \) as a graph \( G=(V,E) \), where \( V \) is a set of conversations and \( E \) is a set of edges that connects conversations which have common interactions. A graph generated for the sample Conversations List given above is provided in Figure 6.3a. Then, sets of vertices that model CCSs are derived from such graphs.

Definition 6.4 (Set of Vertices Modeling a CCS)  
Given a graph \( G \) generated for a Conversations List \( L \), the corresponding CEP \( P \) and an interaction function \( f \), a set of vertices \( V' = \{v_1, \ldots, v_n\} \) of a sub-graph \( G'=(V',E') \) of \( G \) model a CCS if \( V' \subseteq V \), \( E'=\emptyset \) and \( f(v_1) \cup \cdots \cup f(v_n) \equiv f(P) \).

Figure 6.3b depicts the set of vertices derived from the graph in Figure 6.3a. The conversations in this set of vertices form a CSS.

Algorithm 6.2 locates the CCSs in a Conversations List. This algorithm requires a Conversations List, the corresponding CEP, a variable to store the traversed paths and one to store the located CCSs. It extracts a conversation from the Conversations List, adds it to the traversed path and makes a recursive call. Then, all the conversations that have an interac-
tion that is common with one in the traversed path are removed from the Conversations List. Afterwards, if the Conversations List is empty and the conversations in the path form a CCS it is placed in the variable that stores the sets of CCSs. Otherwise, the algorithm extracts another conversation from the Conversations List and recursively continues the process until the Conversations List is empty. This algorithm has an exponential time complexity \( m^n \), where \( n \) is the number of conversations in a CCS and \( m \) is the number of conversations in a graph \( G \) that contain an interaction \( i \). A brief description of the functions used in this algorithm follow.

*removeOverlaps\((G, p)\): Removes each conversation \( c_i \) in graph \( G \) that conforms to the following condition, where \( f() \) is a function that returns the interactions used in a conversation.

\[ f(c_i) \cap f(c'_i) \] such that \( \exists c'_i, c_i \in p \)

*contains\((p, P)\): Checks if the conversations in \( p \) conform to the condition specified in Equation 6.1 according to a CEP \( P \) (line 4).

\begin{verbatim}
1 generateCCS(conversations_list, CEP, path, SCCS)
2 removeOverlaps(conversations_list, path);
3 if conversations_list.isEmpty() then
4     if contains(path, CEP) then
5         SCCS.add(path);
6     end
7 else
8     for each conversation \( c_i \) in the conversations_list do
9         path.add(c_i);
10        generateCCS(conversations_list, CEP, path, SCCS);
11     end
12 return SCCS;
13 end
\end{verbatim}

*Algorithm 6.2: Locating Complete Conversation Sets*
6.4.3 Detecting Deadlocks and Synchronisation Conflicts

A novel technique is proposed to check whether or not a valid global communication can be derived from a CCS. A conversation between interactions $i_a$ and $i_b$, or $i_a$, $i_b$ and $i_c$ dictates that the temporal relationship(s) between them should be $= (i_a, i_b)$ for the former case, and $si(i_a, i_b)$ and $fi(i_a, i_c)$ for the latter. Our approach takes these temporal relationships that exist between interactions because of conversations and the relationships defined in the descriptions of constituent services, and reasons about the relationships between all the interactions of a CEP using the transitive temporal reasoning mechanism described in Section 6.2. A deadlock or a synchronisation conflict occurs in a global communication model if a relationship derived by this mechanism conflicts with another.

Let us consider the following CCS to illustrate the use of the proposed approach: $[cs_2, in_2], [cs_3, in_3, i_5], [cs_3, sh_1, sh_4], [sh_2, in_1][sh_3, in_4]$. This is formed with the interaction protocols given in Figure 6.1. Figure 6.4a depicts a global communication model derived from this CCS. This model contains a synchronisation conflict because the reasoning mechanism determines that the relationship between $sh_1$ and $cs_4$ should be $b(sh_1, cs_4)$\(^{10}\) and the conversation between $cs_4$, $sh_1$ and $sh_4$ dictates that $s(sh_1, cs_4)$. If the 3\(^{rd}\) and 5\(^{th}\) interactions of the Insurance-I in Figure 6.1 are swapped, then the global communication model given in Figure 6.4b would be derived from the following CCS: $[cs_2, in_1], [cs_3, in_3, i_5], [cs_3, sh_1, sh_4], [sh_2, in_2][sh_3, in_4]$. A deadlock is detected because the relationships between $in_3$ and $cs_3$ conflict. The conversation between $cs_3$, $in_5$ and $in_3$ dictates that $f(in_3, cs_3)$ should hold. However, the reasoning mechanism derives that $bi(in_3, cs_3)$ according to the relationship $bi(in_3, in_5)$ in the interaction protocol of Insurance-I and the relationship $s(in_5, cs_3)$ that occurs because of the conversation.

Given a CEP $P$ and a CCS $C$ derived from $P$, VGC derives the temporal relationships between each pair of interactions $i$ and $i'$ based on those that exist between the conversations in $C$, where $i \in p_i$, $p_i \in P$, $i' \in I$ and $I = \{P \setminus p_i\}$. These relationships are derived using the proposed transitive temporal reasoning mechanism (see Section 6.2). While doing so, VGC checks if each derived temporal relationship conflicts with those specified in each $p_i$ of $P$. If they conflict, a global communication model that is free of deadlocks or synchronisation conflicts cannot be derived from the given CCS.

VGC uses a Relations List to record the temporal relationships and later checks if they

---

\(^{10}\) $b(sh_1, cs_4)$ as a result of the conversations between $cs_2$ and $in_2 (= (cs_2, in_2))$ and $in_1$ and $sh_2 (= (in_1, sh_2))$ and interaction protocols of Shipping-I and ComputerSales-I ($b(sh_1, sh_2)$ and $s(cs_2, cs_4)$)
(a) A Global Communication Model with a Synchronisation Conflict

(b) A Global Communication Model with a Deadlock

Figure 6.4: Errors in Global Communication Models
conflict. At the start, this list contains specifications of temporal relationships between the interactions of constituent services as well as those that implicitly exist between the interactions that are included in conversations of a CCS. Each relation in this list is specified as an Interval Time Logic (ITL) axiom modeling a relationship between two interactions. VGC then determines all the two-hop transitive temporal relationships that are extracted from the execution paths. Given two interactions $i_a$ and $i_b$, a two-hop transitive temporal relationship exists between $i_a$ and $i_b$ if there is an interaction $i_c$, where the relationships between $i_a$ and $i_c$, and $i_c$ and $i_b$ are known. Once each of these two-hop transitive temporal relationships are determined, they are compared against those in the Relations List. If the relationships do not conflict, those derived are included in the list. Otherwise, the algorithm will not be able to compute a global communication model which is free of errors.

Algorithm 6.3 (shown below) checks a given CCS. This algorithm requires a CEP, a CSS derived from the CEP, and a Relations List. It iterates through the interactions of a CEP and combines each interaction of a particular execution path with those of other execution paths (Lines 2-6). While doing so, the two hop transitive relationships between each pair of interactions are obtained, and each of them is compared against those in the Relations List. If a transitive relationship does not conflict with the rest, it is added to the list. Otherwise, the algorithm stops (i.e. there is no global communication model free of deadlocks and synchronisation conflicts for the CCS). The complexity of this algorithm is $mn^2$, where $m$ is the number of execution paths in a CEP and $n$ is the number of interactions in each path.

The functions used in this algorithm are:

- $include(i, R)$: Returns the list of relations in $R$ that describe a relationship between $i$ and another interaction (lines 4 and 7).
- $twoHopTransitivePairs(L, L')$: Returns pairs of transitive relationships in $L$ and $L'$ (line 8).
- $relationships(T, i, i')$: Returns the transitive relationships between interactions $i$ and


CHAPTER 6. VGC: GENERATING VALID GLOBAL COMMUNICATION MODELS

\(i'\) according to the pairs in \(T\) (line 9).

- \(\text{isConsistent}(r, R)\): Checks whether a relationship \(r\) conflicts with one in \(R\) (line 11).

\begin{algorithm}
\begin{algorithmic}
\State \textbf{verifyCCS}(CEP, CCS, R)
\For {each execution path \(p\) in CEP}
\For {each interaction \(i\) in path \(p\)}
\State \(L \leftarrow \text{includes}(i, R)\);
\For {each execution path \(p'\) in \(\{CEP \setminus \{p\}\}\)}
\For {each interaction \(i'\) in path \(p'\)}
\State \(L' \leftarrow \text{includes}(i', R)\);
\State \(T \leftarrow \text{twoHopTransitivePairs}(L, L')\);
\State \(Q \leftarrow \text{relationships}(T, i, i')\);
\For {each relationship \(r\) in \(Q\)}
\If {\(\neg \text{isConsistent}(r, R)\)}
\State \(\text{return FALSE}\);
\EndIf
\State \(R\).add\(r\);
\EndFor
\EndFor
\EndFor
\EndFor
\State \(\text{return TRUE}\);
\EndFor
\end{algorithmic}
\end{algorithm}

\textit{Algorithm 6.3: Verifying a CCS}

6.4.4 Deriving Specifications of Global Communication Models

Concise and accurate specifications of global communication models are important for efficient execution of composite services [Nanda et al., 2004]. Here we specify the temporal relationships between conversations and group them to eliminate duplicate information. The temporal relationships between conversations are found based on those that exist between interactions. Descriptions of the relationships between interactions are obtained from a Rela-
tions List (generated when verifying a CCS). VGC derives concise and accurate specifications in two steps.

**STEP 1.** First, the temporal relationships between conversations are determined and placed in a Grouping List. A grouping is a specification that models relationships between conversations in a hierarchical manner (e.g. $b(c_1, c_2)$, $s(c_1, o(c_2, c_3))$, $s(c_1, b(o(c_2, c_3), c_4))$). A grouping that consists of $n$ conversations is referred to as an $n$-grouping.

The relationships between conversations are the same as those between dominating interactions. A dominating interaction is one that is executed throughout a conversation. Such interactions represent the time taken by the conversations. Hence, in those formed by composing a time interval-based interaction (either a service or an invoke) with two time point based interactions (a send and a receive), the dominating one should be either the service or the invoke interaction. In all other conversations, either interaction could be the dominating one. For example, the dominating interactions of conversations in the CCS in Figure 6.3b are underlined in the following specification:

$$c_1 - [cs_2, in_1]; c_2 - [cs_3, in_3, i_5]; c_3 - [cs_4, sh_1, sh_4]; c_4 - [sh_2, in_2]; c_5 - [sh_3, in_4].$$

The relationships between these conversations ($c_1$-$c_5$) are those that exist between corresponding dominating interactions $cs_2$, $cs_3$, $cs_4$, $sh_2$ and $sh_4$.

**STEP 2.** The temporal relationships between conversations are grouped to eliminate any duplicate information. Service execution engines are able to easily establish the order in which conversations should be performed, when relationships are grouped. For example, the order in which conversations $c_1$, $c_2$ and $c_3$ should be performed can be specified using a set of ITL axioms as $s(c_1, c_3)$, $b(c_1, c_2)$ and $d(c_2, c_3)$. However, this can be specified in a grouped manner as $s(b(c_1, c_2), c_3)$. Unlike the former, the order in which the conversations need to be performed can be easily identified (without any reasoning) with the latter specification. Such specifications are derived by forming clusters. A cluster is formed by grouping the relationships between three conversations.

**Definition 6.5 (Cluster)** Given the relationships $r_{12}(c_1, c_2)$, $r_{13}(c_1, c_3)$ and $r_{23}(c_2, c_3)$ between three conversations $c_1$, $c_2$ and $c_3$,

1. a grouping $r_{12}(c_1, r_{23}(c_2, c_3))$ is a cluster if $r_{12} = r_{13}$
2. a grouping $r_{13}(r_{12}(c_1, c_2), c_3)$ is a cluster if $r_{13} = r_{23}$
3. a grouping $r'(r_{12}(c_1, c_2), c_3)$ generated according to the rules in Table 6.2 is a cluster.

For example, the clusters $b(c_1, s(c_2, c_4)), b(o(c_1, c_5), c_2)$ and $s(b(c_1, c_2), c_3)$ can be formed with the sets of relationships \{\(b(c_1, c_2), b(c_1, c_4), s(c_2, c_4)\}\}, \{\(b(c_1, c_2), b(c_5, c_2), o(c_1, c_3)\)\} and \{\(s(c_1, c_3), d(c_2, c_3), b(c_1, c_2)\)\} respectively.

The proposed approach generates $n$-groupings by forming clusters. While doing so, VGC checks whether a grouping contains all the conversations of a CCS. If it contains all the conversations, then this grouping is returned as a specification of a global communication model. Otherwise, the generated cluster is placed in the Grouping List. That means, VGC derives a specification by forming all $n$-groupings for $2 \leq n \leq m$, where $m$ is the number of conversations the given CCS. Figure 6.6 shows how the relationships between conversations given above are grouped to obtain a specification of a global communication model.

Algorithm 6.4 (given below) derives a specification of a global communication model from a verified CCS and a Relations List $L$ (that is generated during the verification process). First, the relationships between the conversations are retrieved and placed in a Cluster Queue $Q$ (lines 2 and 3). Then, this algorithm iterates through $Q$ to form triplets of clusters (lines 4-6). While iterating it checks whether the triplets of clusters are groupable (lines 7 and 8).

Note that the relationship types $si$, $fi$, $mi$, $di$, $oi$ and $bi$ are not considered since such relations can be transposed to their inverse types (i.e. $s$, $f$, $m$, $d$, $o$ and $b$ respectively). For example, $si(c_1, c_2)$ can be transposed to $s(c_2, c_1)$.
\[ \begin{align*}
  b(c_1, c_2), s(c_1, c_3), d(c_2, c_3), f_i(c_1, c_4), b_i(c_2, c_4), d_i(c_3, c_4), b(c_1, c_5), d_i(c_2, c_5), d_i(c_3, c_5), b(c_4, c_5) \\
  2\text{-grouping} & \quad f(c_4, c_1), b(c_1, c_2), b(c_4, c_2), d(c_5, c_2), s(c_1, c_3), \\
  & \quad d(c_2, c_3), d(c_4, c_3), d(c_5, c_3), b(c_1, c_3), b(c_4, c_5) \\
  & \quad \downarrow \\
  & \quad b(c_1, c_2), b(c_4, c_2), f(c_4, c_1) \Rightarrow b(f(c_4, c_1), c_2) \\
  & \quad b(c_1, c_2), b(c_5, c_2), b(c_1, c_3) \Rightarrow b(b(c_1, c_3), c_2) \\
  & \quad b(c_4, c_2), b(c_5, c_2), b(c_4, c_3) \Rightarrow b(b(c_4, c_3), c_2) \\
  & \quad s(c_1, c_3), d(c_2, c_3), b(c_1, c_2) \Rightarrow s(b(c_1, c_2), c_3) \\
  & \quad s(c_1, c_3), d(c_4, c_3), f(c_4, c_1) \Rightarrow s(f(c_4, c_1), c_3) \\
  3\text{-grouping} & \quad s(c_1, c_3), d(c_5, c_3), b(c_1, c_5) \Rightarrow s(b(c_1, c_5), c_3) \\
  & \quad d(c_2, c_3), d(c_4, c_3), b(c_4, c_2) \Rightarrow d(b(c_4, c_2), c_3) \\
  & \quad d(c_2, c_3), d(c_5, c_3), d(c_5, c_2) \Rightarrow d(d(c_5, c_2), c_3) \\
  & \quad d(c_4, c_3), d(c_5, c_3), b(c_4, c_5) \Rightarrow d(b(c_4, c_5), c_3) \\
  & \quad b(c_1, c_5), b(c_4, c_3), f(c_4, c_1) \Rightarrow b(f(c_4, c_1), c_3) \\
  & \quad b(c_4, c_2), b(c_4, c_5), d(c_5, c_2) \Rightarrow b(c_4, d(c_5, c_2)) \\
  & \quad b(c_1, c_2), b(c_1, c_3), d(c_5, c_2) \Rightarrow b(c_1, d(c_5, c_2)) \\
  & \quad \downarrow \\
  & \quad b(b(c_1, c_3), c_2), b(b(c_4, c_5), c_2), f(c_4, c_1) \Rightarrow b(b(f(c_4, c_1), c_5), c_2) \\
  & \quad s(b(c_1, c_2), c_3), d(b(c_4, c_2), c_3), f(c_4, c_1) \Rightarrow s(b(f(c_4, c_1), c_2), c_3) \\
  & \quad s(b(c_1, c_2), c_3), d(d(c_5, c_2), c_3), b(c_1, c_3) \Rightarrow s(b(b(c_1, c_3), c_2), c_3) \\
  & \quad d(b(c_4, c_2), c_3), d(d(c_5, c_2), c_3), b(c_4, c_5) \Rightarrow d(b(b(c_4, c_3), c_2), c_3) \\
  & \quad s(b(c_1, c_5), c_3), d(b(c_4, c_5), c_3), f(c_4, c_1) \Rightarrow s(b(f(c_4, c_1), c_3), c_3) \\
  & \quad b(c_4, d(c_5, c_2)), b(c_1, d(c_5, c_2)), f(c_4, c_1) \Rightarrow b(b(f(c_4, c_1), d(c_5, c_2))) \\
  & \quad \downarrow \\
  5\text{-grouping} & \quad s(b(f(c_4, c_1), c_2), c_3), s(b(f(c_4, c_1), c_5), c_3), d(c_5, c_2) \\
  & \quad \downarrow \\
  & \quad s(b(f(c_4, c_1), d(c_5, c_2)), c_3)
\end{align*}\]

Figure 6.6: Deriving a Specification of a Global Communication Model
CHAPTER 6. VGC: GENERATING VALID GLOBAL COMMUNICATION MODELS

If they are, the grouped cluster is checked to see if it contains all the conversations of a CCS (line 9). Such a grouping of clusters forms a specification of a composite service. Otherwise, the grouping is placed in $Q$ (line 12) and the algorithm continues generating triplets of cluster. The time complexity of this algorithm is $3mn^2$, where $m = (p - 1) / 2$ ($p$ is the number of conversations in a given CCS) and $n = p - 1$. Following is a brief description of the functions used in this algorithm.

- $dominatingInteractions(CSS)$: Returns the dominating interactions of the conversations in a CCS (line 2).
- $relationships(I,L)$: Returns a Cluster Queue that contains the relationships between the conversations. These relationships are determined based on those between the interactions in $I$. The relationships between the interactions in $I$ are retrieved from the Relations List $L$ (line 3).
- $grouping(u_a, u_b, u_c)$: Returns a cluster $u'$ according to the details given in Table 6.2 if the clusters $u_a, u_b, u_c$ are groupable. Otherwise, returns NULL (line 7).
- $containsAll(u, CSS)$. Checks if a cluster $u$ contains all the conversations of a CCS (line 9).

6.5 Theoretical Analysis

This section provides an analysis of the soundness and completeness of VGC. The following describes the semantics of these properties when they are applied to techniques that form global communication models of composite services.

- **Sound** - If only valid models (i.e. free of deadlocks and synchronisation conflicts) are derived.

- **Complete** - Forms a valid model if at least one can be formed.

**Soundness**

The soundness of VGC is proven using the sample scenarios described with Figures 6.2b, 6.4a and 6.4b. We will show that a global communication model is free of deadlocks by mapping it to a $\pi$-calculus [Sangiorgi and Walker, 2001] process definition, and then reducing it to an
Algorithm 6.4: Deriving a Specification of a Global Communication Model

empty set using reaction rules [Sangiorgi and Walker, 2001]. Both situations, namely when specifications (of valid global communication models) are derived or when they not derived, are considered to establish the soundness of our approach.

Valid Global Communication Models

Here, we map the sample specification \( s(b(f(c_4, c_1), d(c_5, c_2)), c_3) \) derived in the previous section (for the scenario described with Figure 6.2b) to a π-calculus process definition, and later reduce it to an empty set. The conversations in \( s(b(f(c_4, c_1), d(c_5, c_2)), c_3) \) can be mapped to communicative actions of π-calculus as in Table 6.3.

The following is the π-calculus process \( P \) of the global communication model in Figure 6.2b, where \( D, S \) and \( I \) are interaction protocols of ComputerSales-I, Shipping-I and Insurance-I services respectively.

\[
\begin{align*}
1 & \text{ deriveGlobalModel}(CCS, L) \\
2 & \text{ } \rightarrow \text{ dominatingInteractions}(CCS); \\
3 & \text{ } Q \rightarrow \text{ relationships}(I, L); \\
4 & \text{ } \textbf{for each relationship } u_a \text{ in } Q \text{ do} \\
5 & \quad \text{ } \textbf{for each relationship } u_b \text{ in } \{Q \setminus u_a\} \text{ do} \\
6 & \quad \quad \text{ } \textbf{for each relationship } u_c \text{ in } \{Q \setminus \{u_a \cup u_b\}\} \text{ do} \\
7 & \quad \quad \quad \ u' \leftarrow \text{ grouping}(u_a, u_b, u_c); \\
8 & \quad \quad \quad \text{ } \textbf{if } u' \neq \text{ NULL } \text{ then} \\
9 & \quad \quad \quad \quad \text{ } \textbf{if } \text{ containsAll}(u', CSS) \text{ then} \\
10 & \quad \quad \quad \quad \quad \text{ } \textbf{return } u'; \\
11 & \quad \quad \quad \quad \text{ } \textbf{else} \\
12 & \quad \quad \quad \quad \quad \text{ } R.\text{add}(u'); \\
13 & \quad \quad \quad \quad \text{ } \textbf{end} \\
14 & \quad \quad \text{ } \textbf{end} \\
15 & \quad \text{ } \textbf{end} \\
16 & \text{ } \textbf{end} \\
\end{align*}
\]
Table 6.3: Descriptions of Communicative Actions

<table>
<thead>
<tr>
<th>Conversation</th>
<th>Interactions</th>
<th>Action(s)</th>
<th>Description(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>cs2</td>
<td>d2, d3</td>
<td>ci(quote.Insurance), ci(quote.Insurance), ci &lt; quoteRequest.Insurance &gt;</td>
</tr>
<tr>
<td></td>
<td>in2</td>
<td>i1, i2</td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td>cs3</td>
<td>d4, d5</td>
<td>ci(policy.Insurance), ci(policy.Insurance), ci &lt; policyRequest.Insurance &gt;</td>
</tr>
<tr>
<td></td>
<td>in3</td>
<td>i5</td>
<td>ci(activationRequest.Policy)</td>
</tr>
<tr>
<td></td>
<td>in5</td>
<td>i7</td>
<td>ci &lt; referenceNo.Policy &gt;</td>
</tr>
<tr>
<td>c3</td>
<td>cs4</td>
<td>d6, d7</td>
<td>si(quote.Shipping), si &lt; noteRequest.Consignment &gt;, si (quote.Shipping)</td>
</tr>
<tr>
<td></td>
<td>sh1</td>
<td>s1</td>
<td>cs(request.Shipping)</td>
</tr>
<tr>
<td></td>
<td>sh4</td>
<td>s3</td>
<td>cs(note.Consignment)</td>
</tr>
<tr>
<td>c4</td>
<td>sh2</td>
<td>s2, s3</td>
<td>si(quote.Insurance), si &lt; quoteRequest.Insurance &gt;</td>
</tr>
<tr>
<td></td>
<td>in2</td>
<td>i3, i4</td>
<td>si(quoteRequest.Insurance), si &lt; quoteRequest.Insurance &gt;</td>
</tr>
<tr>
<td>c5</td>
<td>sh3</td>
<td>s4</td>
<td>si &lt; quote.Shipping &gt;</td>
</tr>
<tr>
<td></td>
<td>in2</td>
<td>i6</td>
<td>cs(quote.Shipping)</td>
</tr>
</tbody>
</table>

The communication channels between ComputerSales-I and Insurance-I, ComputerSales-I and Shipping-I, and Shipping-I and Insurance-I are represented with ci, cs and si respectively. The way in which this π-calculus process transforms (as communications are performed) is shown in Table 6.4. The global communication model in Figure 6.2b is free of deadlocks and synchronisation conflicts since this process (s(b(f(c1, c4), d(c5, c2)), c3)) is reduced to an empty set.

Invalid Global Communication Models

Here we use the reductions of π-calculus to check whether the models in Figures 6.4a and 6.4b contain any structural errors. Note that VGC was not able to derive a global communication
model for these scenarios. The process definitions of these two models\textsuperscript{12} are as follows:

1. The scenario described using Figure 6.4a.

\[
\begin{align*}
P_1 &= D \mid S \mid I \\
D &= (d_2|d_6).d_3.d_4.d_5.d_7.0 \\
S &= s_1.s_2.s_3.s_4.s_5.0 \\
I &= i_1.i_3.(i_2|i_4).i_5.i_6.i_7.0
\end{align*}
\]

2. The scenario described using Figure 6.4b.

\[
\begin{align*}
P_2 &= D \mid S \mid I \\
D &= (d_2|d_6).d_5.d_4.d_3.d_7.0 \\
S &= s_1.s_2.s_3.s_4.s_5.0 \\
I &= i_1.i_3.(i_2|i_4).i_7.i_6.i_5.0
\end{align*}
\]

The way in which these two process definitions transform as communicative actions are performed are shown in Table 6.5. Both models do not reduce to an empty set. In the first scenario, the communicative action \([d_2, i_3]\) can be performed since \(i_3\) is guarded by \(i_1\). That means, a communicative action that involves \(i_1\) needs to be performed before one that involves \(i_3\) can be performed. In the second scenario, neither \([d_4, i_3]\), \([s_4, i_6]\) nor \([i_7, d_5]\) cannot be performed since \(i_7\) is guarded by \(i_7\), \(i_6\) is guarded by \(i_7\) and \(d_5\) is guarded by \(d_4\). Hence, there is at-least one deadlock or synchronisation conflict in both process definitions.

\textsuperscript{12}Descriptions of communicative actions in the process definitions are available in Table 6.3.
We conclude that VGC is sound since it only derives a specification of a global communication model if it is free of deadlocks and synchronisation conflicts.

Completeness

A technique (say $V$) that forms a global communication model of a composite service (say $cs$) is complete if it is able to form a valid model when one is available. Let $M$ be the set of valid global communication models that can be formed with the set of interaction protocols $P$ of the constituent services of $cs$. $V$ is sound if we have

$$\exists m \in V(P), m \in M, M \neq \emptyset \quad (6.2)$$

Therefore $M \neq \emptyset$ implies that

- $C \neq \emptyset$ where $C$ is the set of CCSs\(^{13}\) that can be formed with $P$, and
- $\exists c \in C$ such that the temporal relationships $R$ between the conversations in $c$ do not cause deadlocks or synchronisation conflicts and $c \equiv conversationsOf(m)$

Hence, a technique $V$ is complete if the process $V^*$ used to form CCSs generates $c$. Algorithm 6.2 uses an exhaustive approach to locate CCSs. Thus, the proposed technique VGC is complete since it considers all CCSs that can be generated with $P$.

---

\(^{13}\)See Equation 6.1 for a description of CCSs (Complete Conversation Sets).

177 (October 2008)
6.6 Related Work

Verification techniques used on composite services can be categorised into types as those used to identify valid global communication models and other approaches.

Current techniques that derive valid global communication models are not sufficient since they either do not consider all four types of interactions or they do not consider the different temporal relationships between concurrent interactions. We briefly review these techniques in the following. They are categorised based on the formalisms or frameworks used to specify behavioural descriptions and global communication models.

The techniques proposed in [Bultan et al., 2003; 2006; Fu et al., 2004b;c; 2005], [Berardi et al., 2003; 2004; 2006] and [Deutsch et al., 2004; 2006; 2007] model interaction protocols of services as Mealy Finite State Machines (MFSM). All three approaches only consider send and receive interactions. The first approach composes interactions of constituent services to form conversations. The temporal relationships between conversations are determined based on those that exist between send interactions and specified as Linear Temporal Logic (LTL) axioms. Fu et al.’s approach determines the synchronisability of conversations and detects deadlocks in global communication models. Synchronisability ensures that the same set of conversations are generated for a composite service that supports both synchronous and asynchronous communication. Deadlocks are detected by comparing the temporal constraints in interaction protocols against those between conversations. Interaction protocols of constituent services and a global communication model are converted to BPEL4WS specifications and compared using WSAT [Fu et al., 2004a]. WSAT is a model checker that verifies BPEL4WS specifications. It converts BPEL4WS specifications to Promela\(^{14}\) and invokes a SPIN [Holzmann, 1997] model checker to perform the verification. The approach proposed in [Berardi et al., 2003; 2004; 2006] models the problem of forming a global communication model as a Deterministic Propositional Dynamic Logic (DPDL) formula. The DPDL formula is derived from MFSMs that describe the interaction protocols of services. Their work analyses the complexity of identifying the conversations that form a valid global communication model. The technique proposed by Deutsch et al. models a service as a state-based entity that reacts to an input message. They define the conditions under which analysis of global communication models are decidable and support lossy communication channels. A web service is modelled as an entity that has a sequence of state transitions. Each transition

\(^{14}\)Promela is the input language of SPIN.
is a reaction to a received messages. They define a state of a global communication model as a snapshot and show how the properties of the snapshots can be analysed. Deutsch et al.’s approach requires communication channels, alphabets used to define the input messages of a domain, and the input queues of each service to be bounded.

Techniques suggested in [Bordeaux et al., 2004] and [Beyer et al., 2005] model the interaction protocols of constituent services as Labelled Transition Systems (LTS). Both approaches form conversations by assessing the compatibility of interactions. The latter approach assesses the compatibility of interactions by checking their types and the parameters. The former approach provides a compatibility model which outlines situations where interactions either depict opposite behaviours (are compatible), or allow unspecified receptions and deadlocks to occur. Bordeaux et al.’s approach does not determine the ordering constraints between conversations, and hence does not provide a specification of a global communication model. Beyer et al.’s technique models global communication models as LTSs, but does not verify whether they are free of errors such as deadlocks and synchronisation conflicts.

The approach in [Woodman et al., 2004] represents the interaction protocols of constituent services and global communication models using \( \pi \)-calculus. The reaction rules of \( \pi \)-calculus are used to detect deadlocks or livelocks in a global communication model. A similar approach that describes web services using Calculus of Communicating Systems (CCS), describes the desired properties in a global communication model using Linear Temporal Logic axioms and verifies whether the CCS descriptions conform to the axioms using The Concurrency Workbench of the New Century (CWB), is given in [Bao et al., 2006]. CWB is a verification tool that supports model checking, pre-order checking and equivalence checking. However, both approaches do not support invoke and service interactions since \( \pi \)-calculus and CCS are directly used to model interaction protocols.

Approaches introduced in [Koshkina and van Breugel, 2004], [Kang et al., 2007] and [Yi and Kochut, 2004] use BPEL4WS to represent global communication models. Koshkina and van Breugel convert BPEL4WS specifications to BPE-calculus, and verify them using CWB. Verifications are performed to detect deadlocks, compare implementations against specifications, and identify bi-similarity and behavioural equivalence of different specifications. The other approaches define transformation rules that convert BPEL4WS descriptions to Coloured Petri-Nets (CPNs). Kang et al. detect deadlocks, synchronisation conflicts and livelocks using CPN Tools [Ratzer et al., 2003], whereas Yi and Kochut use JCPNet [Yi and Kochut, 2005]. The approach proposed in [Deng et al., 2007] converts Web Service Choreo-
ography Interface (WSCI) descriptions of global communication models to CPNs and then verifies them using CPN Tools. However, all four approaches do not specify how the interaction protocols of constituent services are composed to form global communication models.

The verification techniques mentioned above are used when composing interaction protocols (of constituent services of composite services) and are used to identify valid global communication models. In addition to these techniques there are those that are used when locating, implementing and executing composite services. When performing service discovery they are used to determine whether a located composite service causes the required state transitions, when implementing an implementation’s conformance to its specification is verified and when executing they are used to check if the constituent services have the required privileges and rights to access data flows. The following provides brief descriptions of some verification techniques [Chafle et al., 2005; Foster et al., 2003; Narayanan and McIlraith, 2002; 2003] that perform these tasks.

Narayanan and McIlraith introduce an approach that locates a composite service that causes the required state transition (achieves the required effects). First, parameters, pre-conditions and effects specified in DAML-S descriptions are converted to situation calculus axioms and composed using a reasoner (Golog) [McIlraith and Son, 2002] to form composite services. Then, situation calculus specifications of composite services are mapped to Petri-Nets and verified (using KarmaSIM, which is a simulation tool used to verify Petri-Nets). The verification process detects deadlocks and livelocks that occur in these specifications because of conflicting pre-conditions and effects.

Foster et al. propose a technique that checks if an implementation of a service conforms to its specification. A specification is described in UML using Message Sequence Charts (MSC) and an implementation is described in BPEL4WS. Both, MSC and BPEL4WS specifications are converted to finite state processes and compared using a Labelled Transition System Analyzer [Magee, 1999].

Chafle et al. propose a technique that provides a decentralised execution topology for a composite service based on a BPEL4WS specification of a global communication model and a set of dataflow rules that specify the access privileges of services and security policies of organisations. First, a topology filter groups interactions in a BPEL4WS specification into partitions based on the involved services, and generates all the possible combinations of partitions. Then, a constraint reinforcer locates a combination of partitions where the services in each partition is able to perform its interactions without violating the security
6.7 Discussion

This section presents the advantages and the limitations of the devised approach, which forms global communication models of composite services.

Advantages

• VGC is the only approach that considers all four types of interactions of services (send, receive, invoke and service) and the temporal relationships that exist between them. As pointed out in the previous section, current techniques do not consider the temporal aspects of interactions. The scenarios described with Figure 6.2a and 6.2b show how this can lead to inaccurate results being returned by techniques that form global communication models.

• VGC is correct and sound. Proofs of these properties of the devised approach are given in Section 6.7.

Limitations

• The functions used to form CCSs, and detect deadlocks and synchronisation conflicts have an exponential time-complexity. However, it is shown in [Berardi et al., 2003] that sound approaches that form global communication models are exponential.

• VGC was not evaluated using an implementation since a dataset is not available (no WS-π-calculus descriptions of real-world web services).

6.8 Summary

This chapter describes a theoretically correct technique that forms a valid global communication model of a composite service. A valid communication model is one that is free of deadlocks and synchronisation conflicts. Global communication models derived using existing techniques are not valid since they use inaccurate behavioural descriptions. They either do not consider all four types of interactions (send, receive, invoke and service) of services or they do not accurately model the temporal relationships between concurrent interactions.
The devised approach requires interaction protocols of services to be described with WS-$\pi$-calculus. As pointed out through the sample scenarios in Section 6.1, WS-$\pi$-calculus is the only formalism that accurately models the behavioural aspects of services.

Specifications of global communication models are derived using a four step process. First, the Conversations are formed by composing interactions of constituent services. Two interactions are composable if the parameters and the channels used to perform them match and they are of compatible types. In the second step, combinations of conversations that form global communication models are located. Then, deadlocks and synchronisation conflicts are detected by reasoning about the temporal relationships between interactions (of a global communication model). Finally, a specification of a global communication model is derived by grouping the relationships between the interactions. A case study that analyses the correctness and soundness of the proposed approach using a sample scenario is provided.
Chapter 7

Conclusion

This thesis describes novel approaches for creating service descriptions, and locating and validating composite services.

We state that functional and behavioural descriptions utilised by discovery and validation techniques should be correct and complete. That means, service descriptions should be equivalent reflections of implementations. The attempt made in this thesis (to define formalisms for creating such descriptions) is based on the following arguments.

1. All three functional aspects (i.e. the purpose, state transitions and data transformations) should be represented in service descriptions.

2. Descriptions that model the behaviour of a service by describing interaction protocols should support all four types of interactions (i.e. send, receive, invoke and service). They should consider the temporal aspects of interactions when describing the ordering constraints.

3. The correctness of service descriptions should be assessed, once they are created.

We accurately describe functional requirements in user requests using constraints and demonstrate the importance of considering global constraints when locating composite services. Also, we show how deadlocks could occur in global communication models of composite services when ordering constraints between their interactions do not correspond to those specified in behavioural descriptions. We argue that the temporal aspects of interactions should be considered when forming global communication models of composite services and comparing ordering constraints in global communication models against those in specifications of interaction protocols.
CHAPTER 7. CONCLUSION

The remainder of this Chapter contains the following: i) Demonstrates how the techniques proposed in this thesis can be used to locate a composite service that conforms to a user request, ii) Outlines the key contributes of this thesis, and iii) Indicates directions of future work.

7.1 Overview of Solution

In this section, we show how the contributions of this thesis can be utilised to locate a composite service for the request given in Scenario 1.1. This example assumes the following.


2. These services are described using WS-$\mathcal{ALUE}$ and WS-$\pi$-calculus [Gooneratne et al., 2006; 2007a].

3. WS-$\mathcal{ALUE}$ descriptions of the services contain the details in Figures 4.11 and 5.6, and Table 5.1.

4. The WS-$\pi$-calculus descriptions of ComputerSales-II, Shipping-II and Insurance-II contain the details given in Figure 6.1.

5. Available ontologies are structured as Meta-Ontologies [Elgedawy, 2003; Elgedawy et al., 2004a; 2008] and they model the details in Figure 2.6 and Table 4.3.

6. The user request contains the following constraints;

   (a) Strictly dependent global constraint $\text{location.Dispatch} = \text{location.Pickup} \in \text{validRegion.Insurance}$ given in Section 4.3.

   (b) Functional constraint $\text{productionTime.Computer} + \text{approvalPhase.Insurance} + \text{timeTaken.Delivery} < 7\_\text{DAYS}$ given in Section 5.2.

   (c) Relational constraint $\text{availableDate.Computer} \leq \text{approvalDate.Insurance} \leq \text{date.Pickup}$ described in Figure 5.2.

First, the WS-$\pi$-calculus description of each service is compared against its WS-$\mathcal{ALUE}$ description. This enables any specification error that is not equally reflected in both descriptions to be detected easily. A technique that performs such a comparison is proposed.
in Chapter 3. Section 3.5 shows how an error in either the WS-π-calculus description or the WS-ALUE description of ComputerSales-I is detected using this technique.

Next, composite services that conform to the given global constraints are located.

1. A case study that illustrates how services that conform to the above strictly dependent global constraint are located using the technique proposed in Chapter 4 (S-Match), is given in Section 4.7.

2. A case study that shows how services that conform to the functional constraint \( \text{productionTime.}\text{Computer} + \text{approvalPhase.}\text{Insurance} + \text{timeTaken.}\text{Delivery} < 7\text{.DAYS} \), and the relational constraint \( \text{availableDate.}\text{Computer} \leq \text{approvalDate.}\text{Insurance} \leq \text{date.}\text{Pick-up} \) are located using the proposed matching technique, I-Match, is given in Section 5.5.

Finally, the global communication model of each locate composite service is formed, and verified. The verification checks whether such a model is valid\(^1\). Let us assume that ComputerSales-I, Shipping-I and Insurance-I form a composite service that conforms to the global constraints specified in a user request. A case study showing how the global communication model of this composite service is formed and verified, is given in Section 6.5.

### 7.2 Research Contributions

The solutions described in this thesis are summarized in the following.

- We propose two formalisms for creating functional and behavioural descriptions: WS-ALUE and WS-π-calculus. The former extends the Description Logic language ALUE and represents all three functional aspects of a service. The latter extends the process definitions of π-calculus to include invoke and service interactions, and incorporates Interval Time Logic (ITL) axioms to devise an approach that considers the temporal aspects of interactions when modeling ordering constraints. Even though functional and behavioural descriptions should be compared against the implementation of a service to guarantee their correctness, it is difficult to perform this pragmatically. Hence, this thesis defines an approach that compares the two descriptions of a service. Although this technique is unable to ensure their correctness, it can be used to detect errors which are not equally reflected in both descriptions.

\(^1\)A global communication model is valid if the control flows between the interactions of the constituent services are free of synchronisation conflicts or deadlocks.
• We categorise global constraints as either strictly dependent or independent based on the complexity of finding complete solutions (i.e. locating all the conforming services). Constraints of the former type can be solved in polynomial time whereas such a property cannot be associated with those of the latter. The semantic-based matching approaches S-Match and I-Match are proposed for locating services that conform to strictly dependent and independent constraints respectively. The latter is a group of techniques that considers two types of independent global constraints: functional (locally optimised non-binary) and relational (binary). Services that conform to functional constraints are located in polynomial time using objective functions that locally optimise the values assigned to attributes. The technique that deals with relational constraints either uses a greedy approach or incorporates a domain rule-based approximation technique to locate conforming services.

• We propose a technique that incorporates a composability model and an ITL-based transitive reasoning mechanism to generate global communication models of composite services. The former assembles conversations by composing the interactions of constituent services. The latter is used to find the temporal relationships between conversations of a global communication model, compare them against the specifications of interaction protocols and detect any deadlocks or synchronisation conflicts.

7.3 Future Work

The following discusses issues to be considered to improve the work proposed in this thesis.

7.3.1 Service Descriptions

The impact that descriptions created with the proposed formalisms have on service discovery and validation techniques needs to be analysed. A sample that consists of services from various industries needs to be selected and described using the proposed and existing frameworks. Then the results obtained by discovery and validation techniques when these services are described with the various frameworks needs to be compared. Functional and behavioural descriptions can be compared using two types of techniques. The first checks the possibility of deriving the post-conditions of a web service from its pre-conditions by incrementally modifying the initial state (defined by the pre-conditions) with the effects of the interactions. The second compares the data transformations specified in functional descriptions against
the parameters of interactions. The proposed technique is of the initial type, whereas one of \( \pi \)-calculus [Sangiorgi and Walker, 2001]. A study needs to be conducted to identify the different types of errors that occur when creating service descriptions. Then the range of errors that are detected and missed by the proposed verification technique can be identified. Furthermore, tools that support the creation of WS-ALUE and WS-\( \pi \)-calculus descriptions need to be developed.

### 7.3.2 Service Matching

As the proposed matching techniques (that locate services that conform to global constraints) only consider strictly dependent, locally optimisable non-binary, and binary constraints, those that consider unoptimisable and incrementally optimisable non-binary constraints need to be developed\(^2\). User requests that describe the functional aspects of required composite services could contain multiple local and global constraints. Furthermore, the service selection process could be refined by categorising these constraints as either mandatory or optional. Hence, the proposed techniques need to be extended to develop one that considers all the above mentioned constraints types. The techniques proposed in this thesis need to be evaluated using a real-world dataset that consists of service descriptions, ontological descriptions and user requests. The experiments described in Chapters 4 and 5 were performed with randomly generated data sets.

### 7.3.3 Service Validation

Future work should focus on analysing the devised validation approach using real-world scenarios. This would facilitate an analysis of the impact of the various exponential algorithms used by this approach (i.e. - the algorithm used to identify combinations of conversations that form global communication models, and the algorithm that detects deadlocks and synchronisation conflicts). Additionally, methods of optimising these algorithms using domain knowledge-based heuristic mechanisms should be investigated.

\(^2\)An overview of the different constraint types are given in Figure 4.1.


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


197 (October 2008)
BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


200 (October 2008)


