Coordination mechanisms for self-interested multi-agent systems

Quan Bai
University of Wollongong
Coordination Mechanisms for Self-interested Multi-agent Systems

A thesis submitted in fulfillment of the requirements for the award of the degree

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Quan Bai
© Copyright 2007

by

Quan Bai

All Rights Reserved
Dedicated to

my parents and wife
Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

Quan Bai
April 30, 2007
A multi-agent system (MAS) is a collection of agents that interact with each other. Multi-agent systems (MASs) can be classified as self-interested MASs and cooperative MASs according to the features of agent goals. Coordination is one of the major issues of MAS research. It plays a central role in MASs to ensure agents achieve interactions properly. Today, the remarkable growth of MAS applications brings higher requirements and more challenges to agent coordinations. Many complex applications require a MAS to include various agents to work together under an open and dynamic environment.

Toward some challenges in current agent coordination research, this thesis deeply investigates agent coordination problems in self-interested MASs, and proposes three coordination mechanisms based on three different methodologies. Firstly, this thesis investigates the use of Coloured Petri Net techniques in agent coordinations. As one of the best modelling tools, Coloured Petri Nets can express a great range of concurrent systems and interactions in graphical representations and well-defined semantics, and allow formal analysis and transformations. In the thesis, a coordination mechanism that uses Coloured Petri Nets to model interaction protocols of agents is proposed and implemented. This mechanism allows interaction protocols to be separated from hard-coded agents. The separation of agents and interaction protocols makes it possible to allow agents to evaluate and select protocols before they operate interactions. Furthermore, a Coloured Petri
Net based approach that allows agents to propose, exchange and evaluate interaction protocols is also introduced in this thesis. By citing Coloured Petri Net techniques in agent coordinations, agents have more flexible and rational interactions. Secondly, this thesis proposes a flexible team formation mechanism for self-interested agents. In this mechanism, agents can evaluate the performance and importance of other agents in the system, and select team members with reasonable terms and costs according to the evaluation result automatically. Comparing with some traditional team formation mechanisms, the flexible team formation mechanism makes agent team compositions more reasonable, and avoids some potential benefit conflicts among self-interested team members (more suitable for self-interested agents). Finally, a knowledge level coordination mechanism that uses of ontologies to describe and manage MAS knowledge is introduced in this thesis. By using ontologies, MAS knowledge can be described and organised in hierarchies, and the dynamic features of independent knowledge sources are captured.
Acknowledgements

Ph.D study is a long journey that could never have been completed without the help of many people.

First and foremost, I would like to thank my supervisor, Associate Professor Minjie Zhang, for her continuous encouragement, patience, and support throughout my Ph.D studies. She is the person who opened the door to academic research for me. Her research knowledge and technical advice have been essential in the completion of this thesis.

I am very grateful to Professor John Fulcher for giving me many helpful suggestions and providing me a chance to apply my knowledge into real applications. My thanks also go to many people in my school for always giving me helps over the years.

My thanks also go to my friends at University of Wollongong, especially the members of my group, Hui Yang, Kaiyu Song, Jia Tang and Fenghui Ren. They make my life in Wollongong so memorable.

On the personal front, I would express my deepest gratitude and love to my parents. I owned them too much in the years of my Ph.D study. I also want to thank my sister and her daughter. If they have not been taking care of my parents, I would never concentrate on my studies. Most important of all, I would like to thank my wife, Wenwei Yan, for her unquestionable support and understanding throughout the years. Her unreserved love makes it possible for me to finish this thesis.
I would like to thank the University of Wollongong for providing me a University Postgraduate Award. I also want to thank the School of IT and Computer Science for the financial supports towards attending research conferences.

Finally, my thanks also go to my examiners, for their valuable suggestions and comments which have been most helpful for improving the quality of this thesis.
The followings are a list of my research papers that have been published or accepted during my Ph.D study that is to end by the completion of this thesis.


• Quan Bai and Minjie Zhang, Agent Coordination Through Ontology Managements. In *Proceedings of International Conference on Artificial Intelligence and Their Applications*, pages 141-146, Innsbruck, Austria, 2004.


## Contents

Abstract \hspace{1cm} vi

Acknowledgements \hspace{1cm} viii

Publications \hspace{1cm} x

1 Introduction \hspace{1cm} 1
   1.1 Agents and Agent Interactions \hspace{1cm} 3
   1.2 Multi-agent Systems \hspace{1cm} 5
   1.3 Self-interested and Cooperative Multi-agent Systems \hspace{1cm} 7
      1.3.1 Traditional Classification \hspace{1cm} 7
      1.3.2 The Blurring Boundary \hspace{1cm} 8
   1.4 Agent Coordination in Multi-agent Systems \hspace{1cm} 9
      1.4.1 Classification of Coordination Techniques \hspace{1cm} 10
      1.4.2 Objective Coordination Mechanisms \hspace{1cm} 11
   1.5 Challenges in Multi-agent Coordination \hspace{1cm} 11
   1.6 Research Objectives and Major Contributions of the Thesis \hspace{1cm} 13
   1.7 Thesis Organisation \hspace{1cm} 15

2 Related Research and Literature Review \hspace{1cm} 18
   2.1 Distinctions between Objective and Subjective Coordination \hspace{1cm} 19
   2.2 Agent Communication in Multi-Agent Systems \hspace{1cm} 20
2.2.1 Agent Communication Language ............... 21
2.2.2 Communication Infrastructure ................. 23
2.2.3 Summary ..................................... 27

2.3 Knowledge and Ontology of Multi-Agent Systems .................................. 27
2.3.1 Ontology in Multi-Agent Systems ............... 28
2.3.2 Ontology Languages .......................... 28
2.3.3 Knowledge and Ontology Services ............... 29
2.3.4 Summary ..................................... 32

2.4 Coordination through Contracting .................. 33
2.4.1 The Contract Net Protocol ..................... 33
2.4.2 The Contingency Contract ...................... 35
2.4.3 The Leveled Commitment Contract ............... 36

2.5 Organisational Coordination ........................ 37
2.5.1 Organisations with Centralised Controls ......... 38
2.5.2 Inclusion of Middle-Agents in Agent Organisations ............. 39
2.5.3 Self-Organising Mechanisms ................... 40
2.5.4 Summary ..................................... 41

2.6 A Summary of Literature Review .................... 41

3 Coloured Petri Net Based Approaches for Coordinating Agent Interactions and Negotiations 43
3.1 Petri Net and Coloured Petri Net ................. 46
  3.1.1 Petri Net .................................. 46
  3.1.2 Coloured Petri Net ......................... 48

3.2 Related Work .................................. 52

3.3 A CPN Based Framework for Agent Interactions ....... 55
  3.3.1 Framework Architecture ..................... 56
  3.3.2 Using CPN Models to Represent Interaction Protocols .... 58
  3.3.3 Communication Interface between Protocols and Agents .... 62
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4</td>
<td>A CPN Based Approach for Flexible Agent Negotiations</td>
<td>66</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Using CPN Tokens to Conduct Negotiation Proposals</td>
<td>67</td>
</tr>
<tr>
<td>3.4.2</td>
<td>CPN Model for Individual Agents</td>
<td>68</td>
</tr>
<tr>
<td>3.4.3</td>
<td>CPN Model for Simple One-to-One Negotiation</td>
<td>69</td>
</tr>
<tr>
<td>3.4.4</td>
<td>The Generic CPN Model for Negotiation</td>
<td>70</td>
</tr>
<tr>
<td>3.4.5</td>
<td>A CPN Based Approach for Flexible Negotiations</td>
<td>75</td>
</tr>
<tr>
<td>3.4.6</td>
<td>Interface between CPN Model and Agents</td>
<td>77</td>
</tr>
<tr>
<td>3.4.7</td>
<td>Negotiation Processes</td>
<td>78</td>
</tr>
<tr>
<td>3.5</td>
<td>Advantages of CPN Based Interactions</td>
<td>97</td>
</tr>
<tr>
<td>3.6</td>
<td>Summary</td>
<td>98</td>
</tr>
<tr>
<td>4</td>
<td>Coloured Petri Net Based Interaction Analysis</td>
<td>100</td>
</tr>
<tr>
<td>4.1</td>
<td>Related Work</td>
<td>101</td>
</tr>
<tr>
<td>4.2</td>
<td>Analysis Techniques of PNs and CPNs</td>
<td>103</td>
</tr>
<tr>
<td>4.2.1</td>
<td>PN Analysis</td>
<td>103</td>
</tr>
<tr>
<td>4.2.2</td>
<td>CPN Analysis</td>
<td>106</td>
</tr>
<tr>
<td>4.3</td>
<td>A CPN Based Approach for Flexible Interaction Formation</td>
<td>110</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Inclusion of Two Kinds of Interactions in the Approach</td>
<td>111</td>
</tr>
<tr>
<td>4.3.2</td>
<td><em>Call</em> and <em>Receive</em> Interaction Proposals in Default Interactions</td>
<td>113</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Protocol Analysis</td>
<td>117</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Advantages of the Protocol Analysis</td>
<td>123</td>
</tr>
<tr>
<td>4.4</td>
<td>Potential Application</td>
<td>124</td>
</tr>
<tr>
<td>4.5</td>
<td>Summary</td>
<td>125</td>
</tr>
<tr>
<td>5</td>
<td>Flexible Team Formation in Self-Interested Multi-Agent Systems</td>
<td>127</td>
</tr>
<tr>
<td>5.1</td>
<td>System Architecture and Problem Definition</td>
<td>130</td>
</tr>
<tr>
<td>5.1.1</td>
<td>The System Architecture</td>
<td>131</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Definitions and Assumptions</td>
<td>131</td>
</tr>
<tr>
<td>5.2</td>
<td>One-Shot and Long-Term Team Formation</td>
<td>134</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>5.2.1</td>
<td>One-Shot Teams</td>
<td>134</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Long-Term Teams</td>
<td>136</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Advantages and Disadvantages of Long-term and One-shot Team Formation</td>
<td>137</td>
</tr>
<tr>
<td>5.3</td>
<td>Flexible Team Formation Mechanism</td>
<td>138</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Team Member Performance Evaluations</td>
<td>138</td>
</tr>
<tr>
<td>5.3.2</td>
<td>System Agent Resource Evaluations</td>
<td>140</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Flexible Member Selection by Using Fuzzy Rules</td>
<td>140</td>
</tr>
<tr>
<td>5.4</td>
<td>Experiments</td>
<td>146</td>
</tr>
<tr>
<td>5.5</td>
<td>Related Work</td>
<td>149</td>
</tr>
<tr>
<td>5.6</td>
<td>Summary</td>
<td>150</td>
</tr>
<tr>
<td>6</td>
<td>An Ontology-Based Mechanism for Agent Coordination through Knowledge Management</td>
<td>152</td>
</tr>
<tr>
<td>6.1</td>
<td>Related Work</td>
<td>154</td>
</tr>
<tr>
<td>6.2</td>
<td>Formal Expressions of Ontologies</td>
<td>156</td>
</tr>
<tr>
<td>6.3</td>
<td>Classification of MAS Ontologies</td>
<td>157</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Common Ontologies of MASs</td>
<td>159</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Special Ontologies of Agents</td>
<td>161</td>
</tr>
<tr>
<td>6.4</td>
<td>An Ontology Based Framework for MAS Knowledge Management</td>
<td>163</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Framework Structure</td>
<td>164</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Knowledge Management Services</td>
<td>165</td>
</tr>
<tr>
<td>6.5</td>
<td>Summary</td>
<td>171</td>
</tr>
<tr>
<td>7</td>
<td>Conclusions</td>
<td>172</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary of Major Contributions</td>
<td>172</td>
</tr>
<tr>
<td>7.2</td>
<td>Future Works</td>
<td>175</td>
</tr>
<tr>
<td>A</td>
<td>Glossary of Terms</td>
<td>176</td>
</tr>
</tbody>
</table>
# List of Tables

1.1 Multi-agent Systems vs Centralised Systems .................................. 7  
2.1 Comparison of Objective and Subjective Coordination ............... 20  
2.2 Comparison of Communication Techniques ............................. 27  
3.1 Description and Comparison of CPN Development Tools .......... 53  
3.2 The Message Colour Set .............................................. 59  
3.3 Formal Description for the CPN Model of FIPA Inform .......... 62  
3.4 The PRO Colour Set .................................................. 67  
3.5 Parameter Meanings of the PRO Colour Set ....................... 68  
3.6 CPN Negotiation Protocol Formal Description ..................... 72  
4.1 Formal Description for CPN $M$ .................................... 109  
4.2 Forward and Backward Incidence Matrices of CPN $M$ ........ 109  
4.3 The IntPro Colour Set ............................................... 114  
4.4 Forward and Backward Incidence Matrices of PP ................ 122  
5.1 Status of An Agent ..................................................... 133  
5.2 Features of One-Shot Teams and Long-Term Teams .......... 137  
5.3 Fuzzy Rule Base Matrix .............................................. 146  
6.1 Formal Expression of Printer Product Ontology .................. 158  
6.2 Formal Expression of IT Product Ontology ........................ 162  
6.3 Formal Expression of A Knowledge Source Ontology ............ 164
# List of Figures

1.1 Intelligent Agent ................................................. 4
1.2 Cooperative Degree .............................................. 10

2.1 Example of a KQML Message ................................. 22
2.2 FIPA Agent Reference Model ................................. 25
2.3 The Ontolingua Server ................................. 31
2.4 The UML Model for the Contract Net Protocol .......... 34
2.5 Organisations in a Multi-agent System ................. 37
2.6 The Blackboard Architecture ................................. 38

3.1 An Example of Petri Nets .................................... 47
3.2 Change of Marking after Transition is Fired .......... 49
3.3 An Example of Coloured Petri Nets ...................... 51
3.4 An Example of XML-Based CPN Format ................. 52
3.5 CPN Based Framework for Agent Interactions ......... 56
3.6 A CPN Place in Interaction Model ...................... 59
3.7 A CPN Transitions in Interaction Models .............. 60
3.8 UML Model for FIPA Inform Protocol .................... 61
3.9 CPN Model for FIPA Inform Protocol (Ready to Inform) 63
3.10 CPN Model for FIPA Inform Protocol (Informed) ...... 63
3.11 CPN Model for FIPA Inform Protocol (Processed) ..... 63
3.12 Structure of Comms/CPN ..................................... 64
3.13 Java/CPN Interface ............................................. 65
3.14 CPN Model for Individual Agents ................................. 69
3.15 CPN Model for One-to-One Negotiation ......................... 70
3.16 CPN Model for Many to Many Negotiation ..................... 71
3.17 The CPN Model for Rubinstein’s Negotiation Protocol ......... 74
3.18 Layered Architecture .............................................. 76
3.19 The Start Transition .............................................. 77
3.20 The Consider & Modify Transition ................................. 78
3.21 One-to-One Negotiation: the seller and the buyer place Proposal Tokens to Proposal Places ......................... 81
3.22 One-to-One Negotiation: the seller sends Proposal Token to buyer 82
3.23 One-to-One Negotiation: the buyer considers and modifies the received proposal ................................. 83
3.24 One-to-One Negotiation: the buyer sends modified Proposal Token to the seller ................................. 84
3.25 One-to-One Negotiation: the seller considers and modifies the proposal of the buyer ................................. 85
3.26 One-to-One Negotiation: the seller agrees on the proposal of the buyer .............................................. 86
3.27 One-to-One Negotiation: the negotiation meets a deal .............. 87
3.28 One-to-One Negotiation: the Proposal Token of buyer is removed by the Remove Dealt Transition, the negotiation is accomplished successfully ................................. 88
3.29 One-to-One Negotiation: no agreement is met before the deadline, i.e. the negotiation fails ................................. 89
3.30 One-to-One Negotiation Process: the negotiation terminates unsuccessfully .............................................. 90
3.31 One-to-Two Negotiation: the seller and buyers place their Proposal Tokens in Proposal Places ................................. 93

xx
3.32 One-to-Two Negotiation: the seller sends its *Proposal Tokens* . . . 94
3.33 One-to-Two Negotiation: buyers modify proposals according to
the received proposal from the seller . . . . . . . . . . . . . . . . . . . 95
3.34 One-to-Two Negotiation: negotiations terminates successfully/unsuccessfully 96
Chapter 1

Introduction

Nowadays, multi-agent system (MAS) is one of the most important concepts in both artificial intelligence (AI) and the mainstream of computer science (CS). It promises a new paradigm for conceptualising, designing, and implementing software systems. This promise is particularly attractive for creating software that operates in open and dynamic environments, such as the Internet [167].

The history of MAS research is not very long. In the past, it was regarded as a subbranch of distributed artificial intelligence (DAI) [19] [100]. However, with the rapid growth of MAS techniques and applications, nowadays MAS has become one of the most important and representative AI techniques, and has been widely applied in many fields including E-commerce, decision support systems, grid computing, etc.

Expanding the range of applications brings higher requirements to multi-agent systems (MASs). Most complex applications require a MAS to include a number of agents, with homogeneous or heterogenous structures, to work together under an open and dynamic environment. The heterogeneity of agents and the dynamic nature of working environments bring many challenging issues to MAS research.

Research on MAS mainly focuses on the development of principles and models which are related with agent interactions and coordinations. It mainly involves the following aspects:

1. Design of MAS architecture and construction of agent-societies/MASs;
2. Design of methods for description and representation of agent knowledge and MAS domain knowledge;

3. Design of languages and protocols for agent communications;

4. Design of mechanisms for agent coordinations and cooperations; and

5. Analysis of agent interaction patterns.

The purpose of this PhD study is to investigate the challenging issues in agent coordination and to develop coordination mechanisms among self-interested agent systems. In MAS research, agent coordination is one of the key research issues. Coordination is the process of managing interdependencies among agent activities. It plays a central role to MASs. Without reasonable coordination mechanisms, agents of a MAS are unable to achieve their individual goals through interactions, and the MAS might quickly degenerate into a collection of agents with chaotic behaviours [62] [167].

In MASs, agents can be characterised as self-interested agents or cooperative agents [100] [159]. In most cases, coordination mechanisms for cooperative and self-interested agents are different. Normally, coordinations for agents with selfish features are more complex because these agents possess higher self-control and sophistication levels. In this thesis, we especially focus on agents with self-interested properties and propose coordination mechanisms targeting current challenging issues in MAS research. However, some of these mechanisms are also generic and suitable for both cooperative and self-interested MASs.

In this chapter, the related concepts of agents and agent interactions are introduced in Section 1.1. Then, characteristics of MASs are presented in Section 1.2. In Section 1.3, definitions of self-interested and cooperative MAS are presented. The boundary between self-interested and cooperative MASs in both traditional and modern MAS research is also introduced in Section 1.3. The problems that should be handled by coordinations are addressed in Section 1.4. In addition,
Section 1.4 also introduces the classification of coordination mechanisms. Section 1.5 particularly focuses on introducing challenging issues in MAS coordination. The major contributions of this thesis are presented in Section 1.6. Finally, the organisation of this thesis is outlined in Section 1.7.

1.1 Agents and Agent Interactions

An agent can be defined as an intelligent entity, which performs given tasks by using its knowledge and information gleaned from the working environment [182]. It can act in a suitable manner toward achieving the given tasks successfully. An intelligent agent can be a software agent (softbot) or physical agent (robot). Some researchers even suggest considering human actors as a kind of agent [148]. An intelligent agent can also be fixed in a computer or travels among different servers (i.e. a mobile agents [117]). However, most intelligent agents possess the following common properties [62] [183] (also see Figure 1.1):

- **Autonomy**: an agent has some level of self-control ability. It can exist and execute tasks in an environment without human directions;

- **Adaptivity**: an agent has the ability to learn and improve its performance with experience;

- **Reactivity**: an agent can perceive its environment and respond in a timely fashion to changes that occur in the environment;

- **Pro-activity**: an agent not only can simply act in response to its environments but also are able to exhibit goal-directed behaviours by taking the initiative;

- **Mobility**: many agents, especially mobile agents, have the ability to migrate in a self-directed way between host platforms;
1.1. Agents and Agent Interactions

- **Social ability**: an agent has the ability to interact, communicate and work with other agents.

![Figure 1.1: Intelligent Agent](image)

Being distinct with other intelligent systems, agents have needs of social activities. This kind of needs are engendered by the limitation of individual (agent) ability and interdependencies among tasks. Agent interactions are necessary and unavoidable for the following reasons:

- Some tasks that agents need to solve are decomposed from a complex problem, which cannot be solved by a single agent. In this case, agents need to communicate with others to check processes or outputs of other agents in order to meet global constraints. For instance, in the application of agent based system design where each agent is responsible for the design of a component of a complex system, agents have to exchange information to adhere local constraints toward overall requirements [83].

- Many tasks, especially in the E-commerce applications, must be executed through interactions. For example, in agent based market places, a seller agent’s task (such as selling a product) is achieved through spreading product information to other agents and negotiating prices with buyer agents [5] [68] [107].
In some agent based monitoring and scheduling applications, agents are allocated to geographically or temporally distributed terminals to execute tasks, and these agents need to collaborate together to achieve global and local monitoring/scheduling. Taking agent based grid scheduling as an example, a scheduler (agent), which is assigned to a computing grid, needs to not only manage its local resources but also communicate with other grid schedulers to exchange their resource availabilities to achieve task scheduling together [37] [130].

1.2 Multi-agent Systems

The social activities of an agent are executed in a MAS. A MAS can be considered as a society of agents that live and work together. Compared with a centralised AI system, a MAS has the following characteristics (also see Table 1.1):

- **Distributed resources**
  
  Computational resources and capabilities are distributed to agents of a MAS. This feature reduces the risk and damage caused by single point failures, which could be critical or even fatal for a centralised system [167] [178].

- **Loosely coupled net-structure**
  
  Many MASs possess loosely coupled net structures [167] [178]. Such MASs may allow their agents to modify their interconnections, and include or eliminate agents during running time. Compared with centralised systems, this characteristic enhances the extendability and flexibility of a MAS and makes it more suitable for open working domains.

- **Decentralised control**
  
  Although many MAS applications are still using centralised controlling
mechanisms, decentralised control is a trend in MAS development. Contrary to centralised systems, there is no global control in a decentralised controlling MAS. Such a MAS may include a number of facilitators to assist agents’ decision makings [131] [169], but there is not a central process that decides each agent’s action. The decentralised control results in asynchronous computations in a MAS. Hence, with decentralised control, MASs require appropriate coordination mechanisms to ensure that individual decisions of agents result in good global performances.

- **Dynamic environment**

Most single agents and centralised AI systems are built for static environments. Whereas agents in MASs need to face open working environments where the configuration and capabilities of other agents and network resources are changeable and dynamic [134]. This feature requires agents in MASs to have a higher adaptivity than single agent systems, and enforces agents to cooperate with appropriate colleagues to perform tasks collectively in order to share unstable system resources.

- **Heterogeneities in MAS**

In many MASs, agents are designed in different ways (i.e. agents are written in different languages, have different structures, etc). This means a MAS may include both homogeneous and heterogeneous agents in one system. The heterogeneity of agents brings difficulties for communications, information sharing and inter-operations among agents [97] [99] [100]. To solve problems brought by heterogeneities and to coordinate heterogeneous agents to work together, it is necessary to have a suitable information/knowledge sharing mechanism embedded inside a MAS. However heterogeneities do not exist in single agent systems.
1.3. Self-interested and Cooperative Multi-agent Systems

1.3.1 Traditional Classification

Activities of an agent are driven by its goal(s). According to the property of goal(s), agents can be generally classified as self-interested (competitive) or cooperative (benevolent) agents [99] [100].

**Definition 1.1.** A MAS that contains agents with distinct or even competitive individual goals is defined as a self-interested MAS. Generally, an agent of a self-interested MAS collaborates with other agents to realise or maximise their local utilities.

**Definition 1.2.** A MAS that contains agents with common goals is defined as a cooperative MAS. Normally, agents of a cooperative MAS work together toward maximising the achievement of their common goal(s).

An example of cooperative MAS applications is the RoboCup [24] [28] [88]. In a robot soccer team, all robot players (agents) collaborate together to achieve their common goal – win the game. A typical example of self-interested MASs is the electronic market place [102] [176] [186]. In an electronic market place,

<table>
<thead>
<tr>
<th>System Resource</th>
<th>Multi-agent Systems</th>
<th>Centralised Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distributed to agents</td>
<td>Centralised occupancy</td>
</tr>
<tr>
<td>System Structure</td>
<td>Centralised structure or loosely coupled network</td>
<td>Centralised structure</td>
</tr>
<tr>
<td>Control</td>
<td>Centralised control or decentralised control</td>
<td>Centralised control</td>
</tr>
<tr>
<td>Component</td>
<td>Heterogeneous &amp; Homogeneous agents</td>
<td>Homogeneous components</td>
</tr>
<tr>
<td>Working Environment</td>
<td>Open &amp; dynamic</td>
<td>Mostly stable</td>
</tr>
</tbody>
</table>

Table 1.1: Multi-agent Systems vs Centralised Systems
different agents work in the same environment toward non-cooperative individual goals. However, agents still need to collaborate with others in order to maximise their rewards, i.e. purchase/sell items collaboratively in order to get best price.

1.3.2 The Blurring Boundary

As the sophistication of MASs increases, the traditional classification of self-interested and cooperative MASs becomes unpractical and unreasonable for many domains [190]. In many MAS applications, a MAS can neither be a simple market system nor an agent colony. The boundary between self-interested and cooperative MASs is blurring [99] [190]. This is mainly caused by the following reasons:

- Firstly, in many current MAS applications, agents of a MAS are from different organisational entities. These agents come together because the organisations they belong to have some cooperations [124]. Therefore the term and conditions of the collaboration between individual agents mainly depends on the higher level relationship between the organisations. This kind of MASs is not purely self-interested because of the existence of common goal(s) among agents. However such MASs can neither be classified as typically cooperative agents because collaborations between agent members are frangible and depend on not only the system’s overall utility but also many outside factors.

- Secondly, in many MAS applications, self-interested agents are also required to take care of the global utility of the system in order to maintain and improve their working environments. As the social welfare of the system increases, all system members including self-interested agents will benefit.

- Thirdly, a MAS can include agents from different organisation entities in the system. This leads an agent of a MAS to have different attitudes toward different targets. An agent can be cooperative with agents from the same
organisation as itself, but act self-interested with agents of other organisations. Therefore, a MAS could be a system, mixed with self-interested and cooperative agents. In this situation, it is difficult to identify whether the MAS is cooperative or self-interested.

- Finally, even from the same organisation, cooperative agents may also behave in a self-interested way due to the limitation of its local view [87] [190].

In order to identify whether an agent is self-interested or cooperative, a potential method which is used in current literature is to use a Cooperative Degree to describe an agent instead of simply concluding whether an agent is cooperative or self-interested [158] [159]. The Cooperative Degree of an agent can be obtained through using statistical models to compare the agent’s degree of concern for local utilities and other agents’ utilities. For example, the Dual Concern Model [101], which is shown in Figure 1.2, can be used to evaluate the cooperative degree of an agent [190]. In this model, two factors, i.e. Degree of Concern for Own Utilities and Degree of Concern for Other’s Utilities, are used to evaluate the Cooperative Degree of an agent.

1.4 Agent Coordination in Multi-agent Systems

Distributed characteristics of MASs, which are introduced in Section 1.2, enhance the overall performance of MASs, especially along the dimensions of computational efficiency, reliability, extensibility, robustness, maintainability, responsiveness, flexibility, and reusability. However, the enhancement is based on efficient coordinations. Only with an appropriate coordination mechanism can loosely coupled heterogeneous agents behave properly in open environments.
1.4.1 Classification of Coordination Techniques

Coordination techniques can be classified as subjective or objective coordinations [39] [126] [128]. In subjective coordinations, agents are considered as active entities that perform psychological actions toward achieving their subjective goals. Objective coordinations are mostly performed by some part of a MAS on behalf of the system’s designer. For example, the system’s designer can build middle agents [79] to perform coordination activities, such as matchmaking and broking services, in a MAS.

Researchers who work on subjective coordinations mainly study developing agents with higher autonomy and make agents become eligible coordination actors [80]. In contrast, researchers who work on objective coordinations mostly focus on the improvement of social facilities and policies in order to construct better environments for individual agents. This thesis does not focus on improving agent coordination abilities (subjective coordination), but mainly focuses on objective coordinations.
1.4.2 Objective Coordination Mechanisms

Obviously, coordination mechanisms developed under different views could be different and even contrasting. Generally, to achieve objective agent coordinations, the coordination mechanisms may cover the following aspects:

1. **Avoid action conflicts without centralised control:**
   A coordination mechanism needs to assist agents to avoid action conflicts and make rational individual decisions that can result in good outcomes for the group [178]. For example, in a cooperative robot soccer team, a coordination mechanism of the team can facilitate individual players to make joint plans to realise common goals and avoid possible action conflicts between them (such as obstructing each other) [178].

2. **Facilitate agents to manage collaborations:**
   Especially for self-interested MASs, the coordination mechanism needs to facilitate agents to discover “colleagues” that have common interests with them. In addition, the mechanism can also help agents to form and keep collaboration relationships or terminate collaborations when necessary.

3. **Facilitate agent communication and knowledge sharing:**
   Especially for heterogeneous agents, different agents could possess different knowledge representation formats and communication language. A related knowledge mapping mechanism is required in a MAS to merge interaction/communication gaps and facilitate communications and interactions among agents.

1.5 Challenges in Multi-agent Coordination

In recent years, a number of researchers have devoted much time to the study of multi-agent coordination and have made brilliant outcomes. However, there are
still some challenges in the field of multi-agent coordination.

- **Agent coordination in massive MASs and complex agent societies:**
  
  Some outstanding research results on coordination for MASs with a large number of agents have been produced in [78] [89] [153]. However, this work can involve only agents with some simple actions, cooperative behaviours and uncomplicated interactions. To build large scale agent societies or MASs that can enable sophisticated agents (cooperative and self-interested) to operate complex activities and interactions is a major research issue in the area of MAS coordination.

  This issue involves research on many aspects including: to develop suitable system/society structures that can enable massive agent interactions; to develop agents with appropriate complexity and autonomy, which can perform efficient operations and interactions in the massive agent society; and to build efficient communication/interaction protocols that can enable massive agent interactions; etc.

- **Agent coordination in complex MASs and dynamic environments:**

  The complex and dynamic nature of MASs requires agents to have high level adaptability and learning ability [180]. A MAS may include various agents from different organisations. Hence an agent of the MAS needs to not only find out potential opportunities and allies that have joint interests with it, but also to identify reliability of its potential allies and information gleaned from the environment. Furthermore, an agent needs to face the open and dynamic working environment where the system constraints, resource availability, and collaboration relationships are changeable. The complex and dynamic nature of complex MASs brings additional requirements and challenges to agents and MAS researchers.

- **Coordination in self-interested MASs:**
In comparison with simple cooperative MASs, coordinations in self-interested MASs are more difficult. The selfish features and sophistication of self-interested agents make collaboration relationships more complicated and fragile. For example, a self-interested agent may want to terminate its collaboration with some agents if its individual goal has changed. It is because the outdated collaborations among agents may no longer be suitable and even conflict with the new goal of the agent. However, breaking the collaborations with others unilaterally is a selfish behaviour and could harm other agents’ benefits. In this case, a social level coordination mechanism is required to resolve benefit conflicts between self-interested agents. How to build such mechanisms to efficiently coordinate behaviours of self-interested agents’ and avoid serious benefit conflicts among self-interested agents is also a challenging issue in MAS research.

In summary, at present, research in multi-agent coordination faces a number of challenges. It is beyond the scope of any particular field of MASs and agent research to solve these challenges. The task ahead is to develop coordination mechanisms that involve multiple technologies of different research areas, including agent learning, knowledge representation, software engineering, etc, to overcome limitations of current approaches in agent coordination.

1.6 Research Objectives and Major Contributions of the Thesis

In targeting the challenging issues of MAS coordination, in this thesis, I focus on the following objectives to ameliorate agent coordination.

- Considering the dynamic features of open environments, one major objective of my research is to develop a coordination mechanism which can enhance the flexibility and extensibility of MASs. Ideally, the mechanism
should allow an unfixed number of agents to join a MAS and form interactions with other agents according to their requirements;

- To allow heterogeneous agents, from different origins, to achieve interactions and cooperations in a MAS, my research also focuses on developing a mechanism to allow agents with different knowledge representation formats to achieve interactions;

- Toward the dynamic features of open environments and sophisticated self-interested agents, another major objective of my research is to develop a mechanism that can allow self-interested agents to flexibly form reasonable interaction relationships when environment constraints and agent goals are changed.

This thesis explores and uses multiple techniques, including software engineering techniques, knowledge management techniques, etc, in agent coordinations. The major contributions of this thesis are outlined as follows.

1. A Coloured Petri Net (CPN) [92] based model for agent interactions is proposed and implemented. The purpose of this work is to improve agent coordination through agent modelling. In the CPN based model, agent interaction protocol can be represented in both mathematical and graphical ways. A generic model for coordinating activities in agent negotiation is developed to demonstrate the promise of this method. Through this demonstration, it can be seen that a generic CPN model can handle agent interactions that involve different numbers of agents. The CPN model can also separate interaction protocols from agents and enable interactions between agents which are built in different languages.

2. A CPN approach to handle agent interaction analysis is proposed. In this approach, CPN techniques are used to analyse robustness and rationality of agent interactions. Since CPNs can be represented in a mathematical format, agents can analyse and even predict the result of collaborations by
1.7. Thesis Organisation

The rest of this thesis is organised as follows.

Chapter 2 is the detail literature review of major approaches and related

evaluating related CPN models. In this way, agents can form interactions according to their needs.

3. A flexible team formation mechanism is developed for self-interested MASs. In this mechanism, the selfish features of self-interested agents are considered, and a method to facilitate agents to choose collaboration objects and terms through reputation based estimations is proposed. The term and cost of keeping collaboration relations between agents depends on the agent performance, task requirements and environment status. Hence, the flexible team formation mechanism can allow agents to form more flexible and rational teams.

4. An ontology based mechanism for agent knowledge management is developed. In this mechanism, ontologies are hired to describe agent knowledge and relationships of different knowledge. Common and special ontologies are classified and defined, and a common ontology is defined and specified for the common vocabulary of agent interactions. As such, we try to handle some problems caused by different knowledge representations among agents.

5. A knowledge management framework for MASs is developed. In this framework, knowledge management processes are defined and several system facilitators are hired to assist agents to share, publish and manage their local knowledge. This framework improves knowledge sharing among agents at a system level.
works in agent coordination. In this chapter, distinctions between objective coordinations and subjective coordinations are discussed; a number of significant coordination technologies of current MAS research are reviewed; and challenges in the research on agent coordination are analysed and addressed in detail.

In Chapter 3 and Chapter 4, CPN based methodologies are explored to be applied in agent coordinations. Chapter 3 presents a mechanism that uses Coloured Petri Net (CPN) techniques to coordinate agent interactions and negotiations. This mechanism utilises the modelling power of CPN techniques, and separates the interaction/negotiation protocols from agents. In the mechanism, CPN models are used to represent interaction/negotiation protocols of agents. Then, agents can operate interactions/negotiations through sending/receiving messages to/from the CPN model of a protocol. By using CPNs to mediate agent interactions, agents can select protocols they prefer to operate interactions/negotiations. These features make agent interactions more flexible, and enhance the extensibility of MASs. Chapter 4 introduces a CPN based approach for interaction analysis. This approach defines two kinds of interactions, i.e. default interaction and proposed interaction, in MASs. Default interactions are basic interactions for agents to exchange and “negotiate” on their preferred interaction protocols. Agents can analyse incoming protocols, which are represented in CPN models, by using CPN analysis algorithms. Through adding analysis processes in agent interactions, agents can evaluate potential interactions and propose interactions according to their demands. These features are very suitable for self-interested agents, and make agent interactions more rational and robust.

Chapter 5 discusses team formation problems toward features of self-interested MASs. In this chapter, two widely applied team formation mechanisms, i.e. one-shot team formation mechanisms and long-term team formation mechanisms, are discussed and compared. To cover some limitations of these two team formation mechanisms, a flexible team formation mechanism that allows agent to select
collaboration terms and rewards is developed. The flexible team formation mechanism allows agents automatically to evaluate the performance and importance of team members, then, to select reasonable collaboration terms and rewards according to the evaluation result. This mechanism makes team compositions in self-interested MASs more reasonable, and avoids some potential benefit conflicts among agents.

Chapter 6 proposes a mechanism that handles agent coordinations in knowledge level through managing MAS knowledge. Ontologies are hired to specify knowledge of MASs, which enables MAS knowledge to be described and organised in hierarchies.

Finally, the contribution of this thesis and future works of this research are presented in Chapter 7.
Chapter 2

Related Research and Literature Review

Multi-agent coordination is an integrated topic that involves multiple techniques of a number of research areas. In recent years, researchers from various fields, including DAI, software engineering (SE), agent-based E-Commerce, etc, developed many exciting techniques for agent coordination. With different knowledge backgrounds, different researchers may study agent coordinations from different viewpoints.

In this chapter, some related research work and important techniques of agent coordination are discussed and reviewed. As mentioned in Subsection 1.4.1, MAS researchers may consider agent coordination problems from objective or subjective viewpoints. In Section 2.1, we discuss the main features and the main differences of/between objective coordination and subjective coordination. Then, from Section 2.2 to Section 2.5, we review a number of important techniques that deal with agent coordinations. Section 2.2 reviews several significant techniques that enable and ameliorate agent communications in MASs. A number of agent communication languages (ACLs) and agent communication infrastructures, which are widely applied in current MAS applications, are presented and compared in this section. Section 2.3 focuses on knowledge management problems of MASs, and introduces several important knowledge management techniques of current MAS research. Then, the concepts and techniques of achieving agent coordinations by using contracting mechanisms are reviewed in Section 2.4. Section 2.5 introduces the concept of organisational coordination and discusses some significant MAS organisational coordination techniques. Finally, a summery of the
chapter is given in Section 2.6.

2.1 Distinctions between Objective and Subjective Coordination

Malone defined coordination as the task of “managing dependencies between activities” \[109\]. Based on this definition, Shumacher \[154\] \[155\] classified coordination techniques into two groups: the subjective coordination and the objective coordination. Shumacher clarified that the separation of coordination techniques is because of the existence of the two agent dependencies \[156\], i.e. objective (inter-agent) and subjective (intra-agent) dependencies. Generally, the inter-agent and intra-agent dependencies of agents need to be coordinated from the objective view and the subjective view, respectively.

Omicini \[127\] \[129\] confirmed Shumacher’s point of view. In addition, Omicini declared that most researchers with DAI backgrounds hold the subjective viewpoint \[125\]. These researchers develop coordination mechanisms based on improving agent activities, which are driven by the subjective goals of agents. The objective viewpoint is normally held by researchers with software engineering (SE) backgrounds. Most SE researchers consider coordinations as a top-down process. They build system components (not agents that operate tasks) to perform coordinations, on behalf of MAS designers. In brief, in subjective coordination, agents of a MAS are the coordinating entities, whereas they are coordinated entities in objective coordinations \[127\] \[128\].

Schumacher and Krone also pointed out that the focus of objective coordination and that of subjective coordination are different. In \[154\] \[95\] they introduced that objective coordination mainly focused on organisational related techniques including describing of a MAS environment and the handling of agent interactions; and subjective coordination emphasised on agent reasoning techniques, such
as multi-agent planning and negotiation.

Obviously, the research on objective and subjective coordination has different theoretical supports and research focuses. However, both Shumacher and Omicini [129] [155] pointed out that a MAS needed to include both objective and subjective coordination in the system. Especially for a complex MAS, it is necessary to identify both subjective and objective dependencies between agents, and to establish an appropriate integrating mechanism to achieve agent coordination.

In summary, most current agent coordination techniques can be ranged into objective or subjective coordination. The two kinds of coordination techniques are different but both of them are important for MASs. The differences between objective and subjective coordination are listed in Table 2.1.

<table>
<thead>
<tr>
<th>Role of Agent</th>
<th>Objective Coordination</th>
<th>Subjective Coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordination Objective</td>
<td>Inter-agent dependencies</td>
<td>Intra-agent dependencies</td>
</tr>
<tr>
<td>Direction</td>
<td>Top-down</td>
<td>Bottom-up</td>
</tr>
<tr>
<td>Techniques</td>
<td>Organisation of the environment, handling agent interactions</td>
<td>Agent reasoning, negotiation, agent team coalition, etc.</td>
</tr>
</tbody>
</table>

Table 2.1: Comparison of Objective and Subjective Coordination

2.2 Agent Communication in Multi-Agent Systems

Communication plays a central role for agent interactions. Without communication, no global behaviour can be achieved in a MAS. To make agents achieve communications, it is necessary to develop an appropriate communication mechanism in MAS. A communication mechanism should cover all essential elements
for agent communications. In general, a communication mechanism includes the following aspects [163]:

- An agent communication language (ACL): the formal language for agents to describe agents’ intentions and knowledge;

- Communication protocols: the rules that agents need to follow in the communication [4];

- A communication infrastructure: the infrastructure that relates to the type of communication medium and the type of connection.

Thus, agents of a MAS can be able to compose information by using the ACL, and exchange information with others by following the communication protocol. In addition, the whole communication process can be facilitated by the mediums, which are defined in the communication infrastructure of the MAS. In this subsection, we introduce several important communication techniques that are widely applied in current MAS research. Especially, some representatives of agent communication languages (ACLs) and communication infrastructures are previewed.

### 2.2.1 Agent Communication Language

“An ACL provides agents with a means of exchanging information and knowledge” [96]. Wooldridge [182] and Cranefield et al [36] classified ACLs as two groups, i.e. the message-based ACLs and the knowledge-based ACL. The main difference between these two kinds of ACLs is whether they can support expressions of knowledge-level messages. In more detail, knowledge-based ACLs can include ontology definitions and knowledge references in messages, but message-based ACLs can only be used to compose simple messages without knowledge-level references.
Message-Based ACLs

Message-based ACLs define a common message format and a set of frequently used performatives [163]. KQML (The Knowledge Query and Manipulation Language) [50] [114] and FIPA-ACL (FIPA Agent Communication Language) [51] are two typical message-based languages, which are widely applied in many MAS applications. Take KQML as an example. It defines a common message format which is shown in Figure 2.1. From Figure 2.1, it can be seen that each KQML message has a performative (“inform”) and a number of parameters (content, receiver, etc). The performative expresses the intended action of the agent, and the parameters express the content and objective of the communication.

```plaintext
(inform
 :content  (PRICE iBookG4 ?price)
 :receiver buyer_agent
 :language sl
 :ontology computer-auction
)
```

Figure 2.1: Example of a KQML Message

As a traditional ACL, message-based ACL has been adopted by many MAS applications and agent platforms, such as JACK [23], ZEUS [122] and FIPA-OS [52]. However, the message-based ACL also have some weaknesses. Firstly, agents that use the message-based ACL to communicate with each other must fully understand all vocabularies of the language. Otherwise communication will be interrupted due to unexpected syntaxes [74]. The reason for this problem is that message descriptions in message-based ACL have a lack of knowledge level specifications [99]. This shortcoming restricts agents to limited vocabularies, and increases the risk of the communication. In addition, Odell et al [123] pointed
out that most message-based ACLs, such as FIPA-ACL [51], must be operated within a particular communication infrastructure. This may bring obstacles for connecting agents with other computational abstractions, and limit the flexibility and extensibility of MASs.

**Knowledge-Based ACLs**

The knowledge-based ACLs are different from the message-based ACLs. Generally, knowledge-based ACLs enable agents to express properties of objects and relationships between objects within their information. One typical knowledge-based ACL is the Knowledge Interchange Format (KIF) [59]. KIF is a computer-oriented language for the interchange of knowledge among disparate programs [59]. It was originally developed to be a common language for expressing properties of a particular domain, then was adapted as a ACL in many MAS applications. Since KIF can provide definition descriptions of objects, functions, and relations, agents that use KIF as the ACL can express properties of objects, relationships between objects and general properties of a domain [182] in their messages.

Knowledge-based ACLs are widely applied in web-based applications. However, most knowledge-based ACLs were not originally developed for agent applications. Therefore many agent actions can not be described in these languages. This feature greatly restricts the use of these languages, especially in non-web based domains [118].

### 2.2.2 Communication Infrastructure

Agent communications need to be operated in the communication infrastructure of a MAS. In this subsection, several representative communication infrastructures in current MAS research are reviewed.
The FIPA Communication Infrastructure

Since 1995, the Foundation for Intelligent Physical Agents (FIPA) [51] has set standards for agent and multi-agent systems. These standards include an ACL (i.e., FIPA-ACL), a number of interaction protocols, a content language (i.e., FIPA-CL), etc. On the aspect of communication infrastructure, FIPA developed the FIPA agent reference model as the standard infrastructure. The structure of the FIPA agent reference model is shown in Figure 2.2. It includes the following four medium components:

- **Directory Facilitator (DF)**: the DF provides a yellow page service to agents;
- **Agent Management System (AMS)**: the AMS enables agents to register on the platform;
- **Agent Communication Channel (ACC)**: the ACC supports inter-operability both within and across different platforms;
- **Message Transport System (MTS)**: the MTS provides the communication service for local and inter-platform message exchange.

Although the FIPA agent reference model is widely applied in many MAS applications, it still has some shortcomings:

- Firstly, Poslad [67] indicated that current FIPA standards do not prescribe how to manage error handling and discrete message processing in agent interactions. The same shortcoming was also found by some other researchers. Li et al. [103] and Hutchison et al. [76] stated that the current interaction standards defined in FIPA lack error handling mechanisms to handle unexpected situations such as missing messages and deviation in the message order. Such a situation may lead to communication interruptions. This shortcoming increases the risk of the communication and reduces the robustness of the MAS.
In addition, another shortcoming of FIPA communication infrastructure is the limited flexibility. Li et al [103] and Bai et al [15] pointed out that the current communication infrastructure of FIPA requires agents to be hard-coded with interaction protocols. This limitation influences the extensibility of agents and reduces the flexibility and robustness of the agents.

Finally, the FIPA does not define an ontology language to express knowledge [180], and does not include semantic mapping components in the communication infrastructure. This shortcoming restricts agents from obtaining
ontology and knowledge related services, such as ontology editing, from a pure FIPA based platform.

Some Extensions of FIPA Interaction Infrastructure

The weaknesses of the FIPA interaction infrastructure limit its applications. To build more robust interaction infrastructures, several researchers have introduced improved FIPA interaction infrastructures. Here, two extended FIPA infrastructures are presented as examples, i.e. JADE and RETSINA.

- **The Java Agent Development Environment (JADE):**
  The JADE platform [23] [30] is an agent platform that complies with the FIPA standard. The interaction infrastructure of JADE is very similar to FIPA. However, JADE includes Java Virtual Machine (JVM) to enable agent migrations among hosts. Using of JVM enhances the flexibility of agent interactions in JADE. In addition, as a Java-based platform, JADE can be connected easily with many Java applications. This feature makes JADE more extendable and flexible.

- **Reusable Environment for Task-Structured Intelligent Networked Agents (RETSINA):**
  RETSINA [172] is a MAS infrastructure that was developed by Carnegie Mellon University recently. One significant innovation of RETSINA is that it has a layered structure. As an independent layer, the communication infrastructure of RETSINA provides agent discovery and message transfer services. The layered structure of RETSINA greatly enhances the reusability and domain independency of MASs.

Indirect Communication Infrastructures

The communication infrastructures previously introduced are for direct agent interactions. However, there is another kind of communication infrastructures that
specifically supports indirect agent interactions, between mobile agents. Most current indirect interaction infrastructures adopt tuple-based interaction models [26] [58]. By using this model, agents can communicate with each other by putting and removing tuples from a shared space. The indirect interaction infrastructures are applied in many computational and multi-agent systems, such as MARS [55], Sun’s JavaSpaces [53] and LIME [119].

2.2.3 Summary

The structure of the communication infrastructures and properties of the ACLs mainly determine the communication activities that agents can perform in a MAS. Table 2.2 is a summarised comparison of the interaction infrastructures introduced in this section.

<table>
<thead>
<tr>
<th>Infrastructure</th>
<th>Bound ACL</th>
<th>Main Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIPA Standard</td>
<td>FIPA-ACL</td>
<td>Lack of error handling mechanisms; hard-code protocols with agents</td>
</tr>
<tr>
<td>JADE</td>
<td>FIPA-ACL</td>
<td>Mainly follows FIPA standards; has a Java-based infrastructure</td>
</tr>
<tr>
<td>RETSINA</td>
<td>KQML</td>
<td>Has a layered structure</td>
</tr>
<tr>
<td>LIME</td>
<td>LINDA</td>
<td>Supports tuple-based interactions between mobile agents</td>
</tr>
</tbody>
</table>

Table 2.2: Comparison of Communication Techniques

2.3 Knowledge and Ontology of Multi-Agent Systems

In most MASs, agents usually use ACLs to compose messages to express their intensions. However, if two agents need to communicate with each other in a
certain application domain, it is necessary for them to agree on the related terminol-
ogy that they will use to describe the domain. In addition, since agents of a MAS might use different knowledge representation formats, the same object could be represented in different ways by different agents. Hence, a knowledge sharing mechanism is necessary for agents to understand each other especially in heterogenous MASs. In current MAS research, many MASs use ontology as a key concept to manage and share knowledge in MASs.

2.3.1 Ontology in Multi-Agent Systems

Ontology is a popular research topic in various research areas because of its promise of a shared and common understanding of domain knowledge that can be communicated across humans and computer programs [177]. The term “ontology” is borrowed from philosophy, and refers to the subject of existence. In the context of MASs, an ontology is a formal description of the concepts related to agents and MAS domains [64]. It explicitly defines the type of concepts used to describe MAS domains and the constraints on these concepts [166].

The most typical type of ontology used in building MASs involves a structural component, which is a taxonomy of knowledge and knowledge relationships [69]. Agents can use ontologies defined in a domain to describe their views of the domain. Since the ontologies have taxonomic relationships, agents are able to map and acquire knowledge through the ontology.

2.3.2 Ontology Languages

Ontologies need to be expressed in a knowledge-based language. KIF (recall section 2.2) is a language that can describe ontologies. However, KIF was originally designed as a language for computers but not for humans. Therefore, translation interfaces are required to allow human editors to write KIF ontologies indirectly.
In recent years, ontologies have been widely applied in many web-based applications. These web-based applications stimulate the development of semantic-web languages, which can be used to compose web ontologies [70] [116]. Many semantic-web languages can also be applied in application areas of MASs. Languages such as Darpa Agent Markup Language (DAML) [38], the Web Ontology Language (OWL) [6], and Xtensible Markup Language (XML) [32], are used to compose MAS ontologies in many MAS applications. The primary intention of a semantic-web language is to develop an accessible language for both humans and computers [17]. However, there is still a long way to go to realise this target. In addition, the semantic-web language is for web-based applications. This restricts many other MAS application domains to adopt the semantic-web language as the knowledge representation language or an ACL.

In summary, most current ontology languages are not robust and cannot be effectively applied in MASs. Malucelli and Oliveira summarised the following problems of current ontology languages.

- Currently, there are many ontology languages. However, none of them has become the standard;
- Some ontology languages do not have formally defined semantics, and agents cannot use these languages;
- Most current ontology languages cannot handle semantic and knowledge mappings effectively.

### 2.3.3 Knowledge and Ontology Services

Many applications require agents with different knowledge representation formats to share their knowledge. Inclusion of knowledge and ontology services into MAS infrastructures can facilitate agents with heterogeneous knowledge representations to achieve interactions easier [41] [42]. Generally, ontology services have the
following functionalities [110]:

1. Describe definitions and relationships between concepts;
2. Maintain ontology definitions and expressions;
3. Provide translation services for different ontologies;
4. Provide ontology searching services to agents.

The Ontolingua Server

The Ontolingua server [9] [48] is a web-based knowledge and ontology services, which provides a common platform for ontology developments and sharing. It is not originally developed as a MAS component, whereas there are many web-based MASs including Ontolingua server which acts as an external knowledge source.

The architecture of the Ontolingua server is shown in Figure 2.3. The central component of the Ontolingua server is the ontology library, which is an ontology-base that stores ontologies of various domains. Remote or standard applications (agents) can obtain ontology services from the ontology library through the server and translator. Human or agent editors can edit and add ontologies into the library through a Hypertext Markup Language (HTML) interface.

The Ontolingua server can provide knowledge and ontology services for some MASs. However, the Ontolingua server has two limitations that restrict it from being widely applied in MAS applications:

- Firstly, the Ontolingua server uses HTML as the knowledge editing interface. However, HTML is a human readable markup language that cannot be understood by many agents. This limitation prevents agents from editing ontologies in the Ontolingua server directly. To overcome this shortcoming, an markup language that can be readable by both agents and humans is required. Currently, many researchers work on this issue and have developed
2.3. Knowledge and Ontology of Multi-Agent Systems

Figure 2.3: The Ontolingua Server

some agent-readable markup languages such as the Darpa Agent Markup Language (DAML) [38] and the Web Ontology Language (OWL) [6]. At present neither of these languages has been recognised as a standard agent markup language.

• Secondly, as with so that other knowledge servers, the Ontolingua server is a web-based server, and most applications that are connected with it should also be web-based. This feature restricts the Ontolingua server from being applied in many non-web based domains [118].

Ontology Services for MASs

The Ontolingua is an independent server that provides knowledge and ontology services for general web-based applications (not particular for MASs). It still has a limited compatibility with many MASs. Therefore, some researchers have built
ontology services in the MAS infrastructure.

Malucelli et al [111] [112] developed ontology agents to provide ontology services for E-commerce negotiation applications. Malucelli chose the Virtual Enterprisers Formation (VEF) as the testing platform. The major service provided by these ontology agents is to map ontologies of different E-commerce organisations (such as different providers).

Malucelli’s ontology service is mainly for ontology mapping. Whereas, Hozo [91] [8] is an ontology service for building and maintaining ontologies of MASs. Hozo has three components: Onto-Studio, Ontology Editor and Ontology Server. Ontology Editor and Ontology Server are two user interfaces for ontology editing and the ontology base, respectively. Onto-Studio is the key component that assists human users to design ontologies from existing documents. Although the ontology editing on Hozo still needs human participation, agents on Hozo can analyse concept relations automatically. This promises the potential of agent (only) ontology editing.

RETSINA [172] also includes ontology services in its infrastructure. In the ACL infrastructure layer of RETSINA (RETSINA has a layered structure), a public ontology server is embedded. Beside the ontology server, several middle agents are included in RETSINA and provide various ontology services such as ontology translating, ontology searching, etc.

2.3.4 Summary

In summary, as the complexity of applications increases, more and more MAS researchers realise the need to include knowledge management and ontology services in the MAS infrastructure. Some exiting techniques, such as the semantic web, promise the potential of automatic ontology/knowledge management. However, most of the existing ontology techniques have some limitations or are not mature enough. Hence, a more extensive study is needed to develop improved
2.4 Coordination through Contracting

To coordinate agent activities and form agent cooperations, some MAS researchers suggest including contracting mechanisms in MASs. In general, contracting is to bind contracts with agents to conduct agent commitments. Once a contract is made, all contract parties (agents) must manage their activities by fulfilling the obligations of the contract. The contracting mechanism is originally perceived by Smith [164] in the Contract Net Protocol. Many MAS researchers have applied and extend Smith’s work and have developed a number of contract based coordination mechanisms. In this section, we introduce and review some significant contracting mechanisms in the current MAS research.

2.4.1 The Contract Net Protocol

The Contract Net protocol is a high-level protocol for achieving efficient cooperation and coordination in MASs. In the Contract Net protocol, agents are classified as two groups, i.e. Initiators and Participants. An Initiator (agent) acts as a manager that spreads task information to agents of a MAS. Participants (agents) are contractors that give responses on whether they are willing to commit on a contract. The UML model of the Contract Net protocol is shown in Figure 2.4. Generally, the Contract Net protocol is composed of the following steps:

1. An Initiator sends out a Call-for-Proposal (CFP) to Participant agents of a MAS;

2. Each Participant evaluates the feasibility of the received CFP and bids on it by sending a proposal if the evaluation result is “feasible”;

3. The Initiator evaluates received bids, then chooses the best proposal and
awards the contract to the sender of the proposal. Also, the Initiator rejects bids from the other Participant(s);

4. The winning Participant processes the task.

The Contract Net protocol can be used to assign tasks dynamically and allow agents entering/leaving a system at will. However, it has several major weaknesses:

1. The Contract Net protocol assumes that all agents of the system are benevolent. As a result, there is no mechanism to check the radiabilities and
conflicts of proposals of *Participants*;

2. In the (traditional) Contract Net protocol, the *Initiator* can cancel a contract even when the *participant(s)* has/have already worked on the task. This is possible if the participants are not self-interested, i.e. the participants do not mind the loss of part of their effort without rewards;

3. As the number of agents increases, the communication consumptions in a Contract Net protocol may increase dramatically. This may cause communication bottlenecks in MASs.

To overcome the limitations of the Contract Net protocol, a number of MAS researchers have developed some improved contracting mechanisms. Here we introduce two contracting mechanisms which are more suitable for self-interested MASs, i.e. the *Contingency Contract* [141] and the *Leveled Commitment Contract* [151].

### 2.4.2 The Contingency Contract

Raiffa [141] suggested to use *Contingency Contracts* in self-interested MAS beside full commitment contracts. The obligations in a *Contingency Contract* are contingent on future events. Agents can make deals according to *Contingency Contracts* without fulfilling any full commitments.

*Contingency Contracts* can enable more flexible commitments among agents. However, this mechanism has the following drawbacks.

1. In many open working domains, it is very hard to predict future events which may occur in *Contingency Contracts*.

2. Future events in a *Contingency Contract* are normally predicted based on individual views of agents. In most cases, it is difficult for an agent to evaluate the reliability and accuracy of future events which are predicted by another agent.
2.4.3 The Leveled Commitment Contract

To avoid the drawbacks of Contingency Contract, Sandholm suggested the use of Leveled Commitment Contracts in MASs [151] [152]. Leveled Commitment Contracts is different from other contracting mechanisms as they allow an agent (either contractor or a contractee) to de-commit from a contract. Agents need to specify de-commitment penalties in a Leveled Commitment Contract. Then, an agent can be freed from the obligations of the contract by paying de-commitment penalties to other parties of the contract.

Compared with other contracting mechanisms, the Leveled Commitment Contract has the following major advantages [152]:

1. With a quitting (de-committing) mechanism, the Leveled Commitment Contract allows an agent (specifically a self-interested agent) to accept new tasks when the agent is committing to a contract. This feature enable agents to purchase higher profits, so that Leveled Commitment Contracts are more suitable for self-interested agents;

2. The Leveled Commitment Contract can reduce the costs resulting from inaccurate predictions. An agent could miscalculate the future events when it evaluates a task. As the task is processed, the agent may recognise these mistakes. In this case, the agent can terminate/de-commit the contract, and reduce additional costs;

3. The Leveled Commitment Contract saves computation resources and time. The quitting mechanism of the Leveled Commitment Contract allows agents to make rough evaluations on tasks. Agents can use less computation resources and time to achieve task evaluations.

The Leveled Commitment Contract is a flexible contracting mechanism which is suitable for self-interested agents. However, the de-committing penalties of contracts are sometimes difficult to be set by agents automatically. This is also
the major problem that limits the implementation of the *Leveled Commitment Contract*.

### 2.5 Organisational Coordination

In MASs and agent societies, an efficient way of ensuring coherent behaviour and resolving conflicts of agents is to organise agents in reasonable structures. From the organisational perspective, the architecture of a MAS can be depicted as Figure 2.5 [82], where agents of a MAS are grouped into different organisations, and agents of an organisation need to manage their activities by following the rules of the organisation. In this section, we introduce and review some important organisational coordination mechanisms in MAS research.

![Figure 2.5: Organisations in a Multi-agent System](image-url)

---

Figure 2.5: Organisations in a Multi-agent System
2.5. Organisational Coordination

2.5.1 Organisations with Centralised Controls

Many MASs include a centralised controller in agent organisations in order to achieve coordination. A classic centralised architecture is the Master/Slave architecture. In a Master/Slave organisation, the master controller is in charge of gathering information from the agents, allocating resources to agents, and assigning tasks to agents of the group in order to ensure global coherence. The Master/Slave architecture requires the master controller (agent) to have higher abilities than slave agents. In addition, agents of a Master/Slave organisation must be cooperative agents.

Another classic centralised architecture is the Blackboard architecture, which is shown in Figure 2.6. In the blackboard architecture, agents have some specialised knowledge and share their knowledge via a public artifact, known as a blackboard. The concept of blackboard system was first developed in the early 1970’s to solve signal-interpretation problems [145]. Then, the blackboard architecture was widely applied in MASs as a communication and knowledge sharing architecture [33].

![Figure 2.6: The Blackboard Architecture](image)

The centralised organisation architectures can efficiently spread information...
and allocate tasks/resources among agents, but they are no longer a major area of MAS research activity for the following major reason. In many current applications, it is very difficult to build a central controller to coordinate all agents of a MAS. In [43], Durfee et al pointed out that centralised controls, such as Master/Slave and Blackboard, are contrary to some basic assumptions of MAS.

### 2.5.2 Inclusion of Middle-Agents in Agent Organisations

There is a need for mechanisms for advertising, finding, managing, and updating agent services and information in MASs. To address these issues, the notion of a middle-agent was brought to the context of MAS research. In [168] [181], Wong and Sycara classified agents of a MAS as two types, i.e. (1) end-agents: service providers and requesters of the MAS; and (2) middle-agents: entities/agents that provide interaction and information services for end-agents. In general, middle agents can be classified, according to different functionalities, as the following types [1] [181]:

- **Matchmakers** and **yellow page** agents: Agents that process the advertisements of service provider agents and assist service requester agents to find service provider agents based on advertised capabilities [169] [170];

- **Blackboard-agents**: Agents that collect and hold requests from end-agents of the MAS [40];

- **Brokers**: Agents that not only receive requests from end-agents but also provide services to end-agents in conjunction with resources of some other agents.

Middle-agents are very useful in large scale, distributed and open MAS application domains. They can provide various decentralised services to agents of a MAS. However, as with any other coordination mechanisms, middle-agents also have several limitations. Firstly, middle-agents such as brokers require agents to
have a static knowledge of the middle agents of the system. This feature limits the application of *middle-agents* in many dynamic domains. Secondly, *middle-agents* could be a communication bottleneck of a MAS due to receiving too many requests from *end-agents*. Finally, middle-agents may cause single point failures in MASs.

### 2.5.3 Self-Organising Mechanisms

In many dynamic application domains, there is no unique organisational structure that is suitable for all situations. Namely, a particular organisational structure can be only suitable for a particular domain [25] [72] [77]. Thus it is not practical to predefine agent organisations in the MAS development stage, and a self-organising mechanism is needed to facilitate agents to find common interests and form organisations.

Self-organising mechanisms for self-interested and cooperative MASs are different. Generally, cooperative agents can form more stable organisations due to less goal conflicts among organisation members. In contrast, it is very hard to keep stable organisations arising from self-interested agents due to possible conflicts among selfish goals. Self-interested agents form organisations for purchasing some temporary common benefits. Therefore, organisations among self-interested agents need to be modified when agent goals and resource availability change.

Self-organising among cooperative agents can be achieved via evaluating local utilities and joint intensions of individual agents [157] [162]. Whereas, such evaluations sometimes are hard to be operated in self-interested MASs. Toward features of self-interested agents, some researchers suggested to cite human organisational theories into multi-agent organising mechanisms. For example, market-base methods, such as auction [150] and voting [134] have been applied in many MAS applications. These methods are suitable for self-interested MASs.
2.5.4 Summary

Agent organisations of the MAS are the collection of roles, relationships and authority structures that govern agent behaviours. Organisational controls provide general long-term guidelines for each agent of the MAS. These guidelines can reduce the complexity of each agent’s operational decision making, decrease unnecessary coordination costs, avoid agent behaviour conflicts, and reduce communication consumptions. As the complexity of MAS applications increases, there is a trend that requires MASs to include self-organising mechanisms to facilitate agents to form flexible organisations.

2.6 A Summary of Literature Review

Multi-agent coordination is an integrated topic that involves the multiple techniques of a number of research areas. In this chapter, a number of important agent coordination techniques were introduced and reviewed.

In Section 2.1, the classification of agent coordination techniques were reviewed. According to Shumacher’s classification, agent coordination techniques were classified as subjective coordination techniques and objective coordination techniques [154] [155]. Omicini agreed on Sumacher’s classification, and pointed out that MAS researchers with different knowledge backgrounds might study agent coordinations from different viewpoints (i.e. the subjective viewpoint or the objective viewpoint) [125]. Shumacher and Omicini [127] [129] [154] [155] illustrated how objective coordinations and subjective coordinations are used to coordinate the inter-agent and intra-agent dependencies of agents, respectively. Both of these techniques are important for MASs.

From Section 2.2 to Section 2.5, a number of important techniques that handle agent coordinations were introduced and reviewed. In Section 2.2, several significant techniques that could enable and ameliorate agent communications in MASs were reviewed. Two kinds of ACLs, i.e. message-based ACLs and
knowledge-based ACLs, were compared in Subsection 2.2.1. In addition, several agent communication infrastructures, which are widely applied in current MAS, were introduced in Subsection 2.2.2. Section 2.3 focused on knowledge management problems of MASs, and introduced concepts of using ontologies to represent and manage MAS knowledge. This section reviewed several important knowledge/ontology editing and management techniques, including ontology languages and ontology services, and pointed out the limitations of current MAS knowledge management techniques. Contracting coordination mechanisms were introduced in Section 2.4. Three important contracting mechanisms, the Contract Net Protocol, the Contingency Contract, and the Leveled Commitment Contract, were also reviewed in Section 2.4. Finally, Section 2.5 introduced concepts of organisational coordination. Advantages and disadvantages of some classic organisational coordination mechanisms, such as the Blackboard System and middle-agent coordinations, were discussed, and the trend of including self-organising mechanisms in MASs was shown in this section. Furthermore, differences between the self-organising mechanisms for self-interested MASs and cooperative MASs were also reviewed in Section 2.5.
Chapter 3

Coloured Petri Net Based Approaches for Coordinating Agent Interactions and Negotiations

As described in Section 2.2 of Chapter 2, agents in MASs compose their messages in an ACL, then exchange messages with others by following some interaction protocols. Interaction protocols constrain the possible sequences of messages that may occur in agent interactions. An agent needs to indicate the interaction protocol, which it wishes to follow, to other agents before it starts the interactions. Then the protocol recipients will follow the pattern described in the protocol to start the conversation if they accept the protocol. In traditional interaction mechanisms, interaction protocols are governed by standard bodies such as FIPA and KQML, or written in immutable documents that can be distributed among agents.

As the application domains of MASs are getting more and more complex, it is increasingly necessary for agents to be able to flexibly and robustly communicate with each other in a changing and uncertain environment known as an open environment. At the same time, the limitations of current MAS interaction mechanisms are rising up, especially in the following aspects [65] [76]:

- **Lack of flexibility and extensibility:**

  Most current interaction mechanisms require agents to be hard-coded with
interaction protocols. Being hard-coded with a particular interaction protocol, an agent cannot achieve interactions with agents with different interaction protocols. In addition, hard-coding protocols actually merge agents into a part of MAS interaction infrastructure. These mechanisms have conflicts with the dynamic feature of open environments, and both the flexibility and extensibility of the agents are greatly reduced.

• **Rigid interaction**

In an open environment, interactions between agents can be influenced by some unexpected factors, such as unexpected messages, loss of messages or deviation in the message order. However, most current MASs lack error handling mechanisms to deal with these unexpected factors. This increases the interaction risk in MASs.

• **Lack of interaction diagnose**

Ideally, interaction protocols should be represented in a format that allows performance analysis, validation, verification, monitoring, and debugging. However, most current MASs do not have such mechanisms. Therefore, even if an agent interaction is interdicted in some unexpected scenarios, it is still hard to find out the reason (or an agent) that causes the interdiction.

In this thesis, an interaction mechanism to make MASs more flexible and robust is developed. Generally, this work focuses on improving agent interactions from the following aspects:

• The interaction mechanisms should allow agents to work in MASs with different interaction protocols. To achieve this target, it is necessary to separate hard-coded interaction protocols from agents;

• The interaction protocol should have a formal description that can be readable by agents from different organisations (which might be developed in different tools and languages);
• The interaction mechanism can handle some unpredictable scenarios, which are unforeseen at design time;

• A generic interaction model should be developed to handle one to one, one to many and many to many interactions;

• Extensible interaction models should be included to allow agent to modify interaction protocols;

To achieve the above objectives, in this research, Coloured Petri Net (CPN) techniques [84] [85] [86] are applied in agent interaction modelling. The CPN is a high-level extension of the Petri Net (PN) [61] [132] [146]. Petri Nets (PNs) and Coloured Petri Nets (CPNs) are system modelling tools that can provide an appropriate mathematical formalism for the description, construction and analysis of distributed and concurrent systems. CPNs can express a great range of interactions in graphical representations and well-defined semantics, and allow formal analysis and transformations. PN and CPN are considered as one of the best modelling tools for concurrent systems and interactions [93] [94]. By using CPNs to model agent interactions, interaction protocols can be separated from agents. In addition, heterogeneous agents can be linked together and generate interaction relationships flexibly. Also, the formal representations of CPNs can enable agents to analyse and acquire (CPN-based) interaction protocols before they setup the interaction relationships with others. Therefore, through hiring CPN techniques into coordinations of agent interaction, the flexibility, extensibility and robustness of MASs can be improved.

In this work, we use CPN techniques to handle agent interaction and negotiation activities. In this chapter, a CPN based framework for general agent interaction and a CPN based approach for agent negotiation are introduced. This chapter is arranged as follows. In Section 3.1, some PN and CPN concepts that relate to this research are introduced. In Section 3.2, some related works in the current research of MASs that use CPNs to handle agent interaction problems are
introduced. Section 3.3 proposes a CPN based framework for agent interactions. The concepts of using CPNs to model agent interaction protocols are introduced in Subsection 3.3.2. In Section 3.4, a CPN based approach for flexible agent negotiation is presented. The advantages of using CPN techniques to handle agent interactions are introduced in Section 3.5. Finally, this chapter is summarised in Section 3.6.

### 3.1 Petri Net and Coloured Petri Net

The concept of PNs has its origin in Carl Adam Petri’s thesis [133], where PNs are used to model communication systems. PNs and CPNs are different from other event-based process modelling notations as they can model both states and events of a system. This feature is ideal for the description of interaction states and transitions between interaction states of concurrent systems. In this section, we briefly introduce some basic concepts of PNs and CPNs and present some PN and CPN techniques that are related to our work.

#### 3.1.1 Petri Net

**Definition of the Petri Net**

A PN can be formally defined by the four-tuple [61] [132] [146]:

\[
PN = (P, T, A, \mu)
\]  

(3.1)

The meanings of the four parameters in the tuple are:

1. **Place set** \( P = (p_1, p_2, ..., p_n) \): \( P \) is a set of places of a PN. A place \( p_i \) can contain a number of tokens. The token availability of a place represents whether the resource/condition represented by the place is available.
2. Transition set $T = (t_1, t_2, ..., t_n)$: $T$ is a set of transitions of a PN. A transition $t_i$ normally represents an action or event of the system.

3. Arc set $A$: $A$ is a set of directed arcs that link places and transitions together. Note: an arc can only link a transition and a place and cannot link transitions together nor places together.

4. Marking $\mu$: A marking $\mu$ is an assignment of tokens to the places of a PN. Tokens are assigned to and can be transferred between the places of a PN. The number and position of tokens are changed during the execution of a PN, which means $\mu$ will be changed after each transition firing.

Figure 3.1 shows a simple example of PNs. In this example, $P = (p_1, p_2, p_3, p_4)$, $T = (t_1, t_2, t_3)$. Since $p_1$ and $p_3$ have one token and other places have no tokens inside, the current marking of the PN is $\mu = (1, 0, 1, 0)$.

**Figure 3.1: An Example of Petri Nets**

**Transition Firing Rules of PNs**

There are a number of transition firing rules associated with different types of PNs. Generally, the allocation status of tokens determines which transition(s) can be fired/executed. In addition, all kinds of PNs share the following common rules [132] [146]:

- **Rule 1:** A transition is enabled if it has enough tokens on its input places.
- **Rule 2:** A transition is enabled if its input places are not empty.
- **Rule 3:** A transition is enabled if its output places are not fully filled.
- **Rule 4:** A transition is enabled if its input places are empty and its output places are fully filled.
- **Rule 5:** A transition is enabled if its input places are half filled and its output places are half empty.
1. A transition is enabled (or fired/executed) only if the token number of all input places of the transition is equal to or greater than their arcs’ weights. Take the PN in Figure 3.1 as an example. According to the current marking of the PN, only $t_3$ can be fired because $p_3$, which is the only input place of $t_3$, has a token inside.

2. After a transition is fired, the tokens at its input places will be moved to its output places.

3. After a transition is fired, the number of tokens that are moved from/to each input/output place equals to the weight of the linking arc. Take the PN in Figure 3.1 as an example again. $t_3$ has one input arc and one output arc that link $t_3$ with $p_3$ and $p_4$, respectively. Hence, after $t_3$ is fired, a token will be transferred from $p_3$ to $p_4$. Therefore, after $t_3$ is fired, one token will be removed from $p_3$ and one token will be moved to $p_4$. Also, the marking of the PN will be changed from $\mu = (1, 0, 1, 0)$ to $\mu' = (1, 0, 0, 1)$ (see Figure 3.2).

### 3.1.2 Coloured Petri Net

Tokens of a PN can only have two values: true or false, which means that the token exists or does not exist in the place, respectively. Tokens of CPNs are different from tokens in PNs. PN tokens are simply blank markers, whereas CPN tokens can conduct data and represent more meaningful information. Another improvement of CPNs is that arcs and transitions of CPNs can conduct arc functions and guard functions. These functions can control token transferring and transformations during transition firings.

**Definition of the Coloured Petri Net**

Formally, a CPN can be defined by a 9-tuple [120]:
3.1. Petri Net and Coloured Petri Net

\[ CPN = (\Sigma, P, T, A, F, C, G, E, \mu) \] (3.2)

The nine parameters in the tuple have the following meanings:

1. *The aggregation of coloured sets* \( \Sigma \): is a set of non-empty data-types, where each coloured class is a token data-type of a CPN;

2. *The place set* \( P \): is a set of places of the CPN. Each place is defined to contain tokens that belong to a particular colour set;

3. *The transition set* \( T \): is a set of transitions of the CPN;

4. *The arc set* \( A \): is a set of arcs that link transitions and places of the CPN;

---

Figure 3.2: Change of Marking after Transition is Fired

(a) Before \( t_3 \) is fired, \( \mu = (1, 0, 1, 0) \)

(b) After \( t_3 \) is fired, \( \mu = (1, 0, 0, 1) \)
5. The colour domain mapping function set $F$: is a set of mapping functions that define from $A$ into $P \times T \cup T \times P$;

6. The colour function set $C$: is a set of the colour functions that define $P$ into $\Sigma$;

7. The guard function set $G$: is a set of guard functions of transitions that define token transferring conditions;

8. The arc function set $E$: is a set of arc functions that are conducted on arcs of the CPN;

9. The initialisation function $\mu$: defines the initial marking of the CPN.

Figure 3.3 gives a simple example of CPNs. In the CPN of this example, $P = \{\text{Sender, Responder}\}; \ T = \{\text{Send, Reply}\}; \Sigma = \{\text{MESSAGE}\}$, where MESSAGE is the only colour set of the CPN; $G = \{\text{checkTure}(m), \text{checkFalse}(m)\}$, where checkTure$(m)$ and checkFalse$(m)$ are two flag-checking functions; $E = \{\text{sent}(m), \text{reply}(m)\}$, where sent$(m)$ and reply$(m)$ are two arc functions that can modify token values; Responder and Sender places are defined to contain tokens with a type of MESSAGE. From this example, it can be seen that the data-types of tokens are defined in colour sets, which can be complex data types. Through defining appropriate colour sets, CPN tokens can be used to express significant information, such as schemas or specifications. Functions on the arcs of a CPN specify the token(s) that they can carry, and functions on output arcs that can modify tokens’ value. In addition, transitions of CPNs are associated with guard functions that enforce some constraints on token values/colours.

**Represent CPNs in Formal Languages**

PN and CPN models can be expressed in XML-based languages, such as Petri Net Markup Language (PNML) [18] and CPN XML [108]. By using these languages, CPN models can be represented in a formal interchange format. Therefore, the
formal interchange format makes it possible to move CPN models between different platforms including MASs. In addition, as a formal language, XML can be read and understood by agents. This feature allows agents to read and generate CPN models according to the formal XML descriptions. Figure 3.4 shows an example of XML-based CPN format.

**CPN development tools**

Many CPN development tools have been successfully developed and widely applied in various areas. Table 3.1 illustrates and compares some of today’s most significant CPN development tools [13] [165]. These tools have different special and distinctive features and use different inscription languages for colour set, net structure and function descriptions.

In this thesis, we use CPN/Tools [11] [16] [144], a tool developed by Aarhus University (Denmark), as the CPN model development tool. CPN/Tools is a tool for editing, simulating and analysing CPNs. It uses an extended Standard
3.2 Related Work

The PNs and CPNs are originally designed as modelling tools for concurrent systems. Since PNs and CPNs can model both states and events of a system, some MAS researchers use CPNs to represent interaction states and transitions between interaction states in a MAS.

Cost [34] [35] proposed the use of CPNs as a model underlying a language for protocol specification by taking the advantages of CPNs’ great expressive power as support for concurrency. In addition, Cost used CPN ML as the language for declarations and net inscriptions. By using CPN/Tools, CPN models can have a XML format, which allows them to be interchanged among different platforms. Furthermore, CPN/Tools supports TCP/IP (Transmission Control Protocol/Internet Protocol) connections and makes it possible to link CPN models with other programs (agents). For this thesis, all CPN models and diagrams will be those developed by using CPN/Tools.

3.2 Related Work

Figure 3.4: An Example of XML-Based CPN Format

```xml
<net id="test" type="CPN"
    <name>
        <value>Example CPN</value>
    </name>
    <place id="p1">
        ..........
    </place>
    <place id="pg1">
        ..........
    </pg1>
</net>
```

Meta Language (SML), CPN ML [10] [98], as the language for declarations and net inscriptions. By using CPN/Tools, CPN models can have a XML format, which allows them to be interchanged among different platforms. Furthermore, CPN/Tools supports TCP/IP (Transmission Control Protocol/Internet Protocol) connections and makes it possible to link CPN models with other programs (agents). For this thesis, all CPN models and diagrams will be those developed by using CPN/Tools.
### Table 3.1: Description and Comparison of CPN Development Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Provider</th>
<th>Inscription Language</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHA/Sim</td>
<td>AlphaTech</td>
<td>Menu based predefined language</td>
<td>Easy to use</td>
</tr>
<tr>
<td>Artifex</td>
<td>Artis S.A.</td>
<td>C++</td>
<td>Generation of distributed code, GUI</td>
</tr>
<tr>
<td>CPN/Tools</td>
<td>Uni Aarhus</td>
<td>CPN ML and SML</td>
<td>Providing access to full SML interactive compiler</td>
</tr>
<tr>
<td>HiQPN</td>
<td>Uni Dortmund</td>
<td>Predefined language</td>
<td>Analysis can exploit hierarchical structure</td>
</tr>
<tr>
<td>INCOME</td>
<td>Promatis</td>
<td>Prolog</td>
<td>Data and organisation models</td>
</tr>
<tr>
<td>PEP</td>
<td>Uni Hildesheim</td>
<td>Boolean and arithmetic operators</td>
<td>Generation of nets form PBC, $B(PN^2)$ or PFA</td>
</tr>
<tr>
<td>PNTalk</td>
<td>Uni Brno</td>
<td>Smalltalk</td>
<td>Object-oriented nets</td>
</tr>
<tr>
<td>Thorn/DE</td>
<td>OFFIS</td>
<td>C++</td>
<td>Fast distributed and sequential simulation, hybrid nets</td>
</tr>
</tbody>
</table>

He developed CPN interpreters to execute interchanged CPN interaction protocols of the MAS. The use of CPN models in Cost’s work greatly facilitates the development of systems of interacting agents and proves the potential for interpreting CPN models from CPN ML expressions. However, Cost did not describe how to interpret CPN models or how to make agents interact by using the interpreted protocols in detail.

Freire [54] also investigated methods to represent interaction protocols. Freire used Agent Unified Modelling Language (AUML) to describe interaction protocols and hired XML as the interchange representation format. Comparing the works of Freire and Cost, it can be seen that CPNs have more advantages than AUML in two aspects: (1) AUML cannot be converted into a formal language directly, but a CPN can be formally described in XML or other formal languages; (2)
3.2. Related Work

A CPN can be directly used as a simulation tool for agent interactions but this function cannot be achieved by using AUML.

Nowostawski [121] described a layered approach based on CPNs that could be used for modelling complex, concurrent conversations among agents in a multi-agent system. Agent interactions were divided into three layers and represented by corresponding CPN models, respectively. Nowostawski also compared features of different interaction modelling tools, including UML, Deterministic State Automaton, Enhanced Dooley Graph and CPNs. Nowostawski claimed that the CPN has more advantages in modelling complex concurrent interactions than other tools. Nowostawski’s work gives an outline of using CPNs in agent interaction modelling. Purvis et al [136] [137] [138] extended the work of Nowostawski and explored the potential application domains of CPN based interactions models, including environmental emergency systems, E-business, tourism systems and work-flow management systems. However, both Nowostawski and Purvis failed to describe how to dynamically construct interactions according to CPN models.

Some researchers used CPNs to monitor agent interactions. Poutakidis [135] used the PNs to monitor agent interactions and generate precise and informative error messages when protocols are not correctly followed by the agents. Gutnik [65] introduced a scalable PN representation of interaction protocols for overhearing agent interactions. These two works demonstrated how to use CPNs to diagnose weaknesses in agent interactions. However, none of these works included a CPN analysis.

There are also some works on the investigation of protocols’ flexibility, robustness and extensibility. Hutchison [76] presented an example protocol, called Merchant-Customer protocol, to describe flexibility and robustness of agent interaction protocols in open systems. Ahn [2] suggested a handshaking mechanism for conversation policy agreements that enables agents to exchange and agree to new conversation policies at runtime.

Some researchers used PNs and CPNs to model agent negotiation activities.
Hung and Mao [75] presented a PN based framework for modelling agent negotiations. This framework formalised agent negotiation activities into formal PN models. However, due to limitations of PNs, it is impossible to represent complex negotiation activities in PNs. Chen et al [29] proposed a semi-automated negotiation processes between agents by using CPNs. However, in this work, Chen et al suggested to embed CPN models into individual agents. As a result, agents in a MAS must have the same architecture. This will limit the flexibility of MASs.

In summary, all of the works presented show the advantages of CPNs in developing more flexible and robust interaction mechanisms. However, most of them only focus on using CPNs to represent agent interactions. However, they have left out the aspect of CPNs which can be applied to enhance agent interactions in many other areas, including interaction simulation, interaction analysis and for providing generic interaction models. Considering this omission, we focus on discovering more advantages that can be provided by CPNs to improve agent interactions.

### 3.3 A CPN Based Framework for Agent Interactions

In our research, we focus on using CPNs to improve agent interactions from the following aspects:

1. Developing CPN based generic interaction models that can be adopted by heterogeneous agents from different origins;

2. Separating interaction protocols from agents and enabling agents to select and execute CPN based protocols by their demands;

3. Using CPNs to visualise agent interactions for monitoring and debugging purposes.
In this section, a CPN based framework for agent interactions is proposed. Within this framework, interaction protocols are represented in CPN models, and separated from agents. Agents operate interactions through sending/receiving CPN tokens to/from corresponding CPN models.

### 3.3.1 Framework Architecture

To achieve the proposed improvements, we put forward a CPN based framework for agent interactions, which is shown in Figure 3.5. This framework has three layers: the protocol layer, the media layer and the agent layer. The detailed descriptions of the three layers are as follows:

![Figure 3.5: CPN Based Framework for Agent Interactions](image)

This layer is the top layer of the framework. It conducts interaction protocols that are used in agent interactions. The interaction protocols are described in CPN models, and are independent from agents. The CPN models have communication ports to accept/send data from/to outside programs. These ports allow the communication between agents and interaction protocols.

2. *The Agent Layer*:

Agents are located in the bottom layer of the framework. Since interaction protocols are conducted in another layer (i.e. the protocol layer), agents in this framework do not need to be hard-coded with interaction protocols. To achieve interactions, agents only need to compose their messages in appropriate formats and send messages to the protocol layer through communication ports.

3. *The Media Layer*:

The media layer is between the protocol layer and the agent layer. It contains a number of facilitators that assist agent interactions. For different applications, the media layer can include various facilitators.

The CPN based framework splits interaction protocols from agents to an individual layer. To interact with others in this framework, agents only need to create messages and send messages to the appropriate communication port that links with the CPN model. Then, the CPN model will be responsible to transfer messages in the direction and order that are defined in the interaction protocol. The protocol layer is the primary part of the framework. It contains interaction protocols that might be adopted by agents. In the next section, we introduce how to express interaction protocols in CPN models.
3.3.2 Using CPN Models to Represent Interaction Protocols

Generally, it takes three steps to create CPN models for interaction protocols:

1. Define appropriate token colour sets, places and transitions according to the properties of the interaction protocol;

2. Use directed arcs to link places and transitions together according to the message flow that is defined in the interaction protocol;

3. Define guard and arc functions on related arcs and transitions according to the message transforming rules and the message transferring conditions, which are described in the interaction protocol.

In this subsection, we define CPN components for interaction protocol modelling, and introduce how to construct CPN models for interaction protocols.

Use CPN Tokens to Represent Messages

We use CPN tokens to represent messages that are exchanged between agents. To achieve this, a specific data-type (i.e. colour set) is needed for message tokens. Since different applications have different requirements for message structures, here, we define a generic colour set for message tokens, which is shown in Table 3.3.2. In this colour set, Sender and Receiver(s) indicate the IDs of agents that take part in the interaction. MessageContent contains the message that the agent wants to express. Flags are some boolean or integer parameters that facilitate transition firings.

CPN Places

The states of an interaction are represented by CPN places. For CPNs, each place has an associated type determining what kind of data that the place can
colour set \( Message = \text{Sender}: \text{STRING} \times \text{Receiver}: \text{STRING} \times \text{MessageContent}: \text{TEXT} \times \text{Flag1}: \text{BOOL} \times \text{Flag2}: \text{BOOL} \cdots \times \text{FlagN}: \text{BOOL}; \)

Table 3.2: The \( Message \) Colour Set

contain. Here, the type of a place can determine which kind of message can be exchanged in the interaction. In addition, the status of token allocation in places directly determines which transition can be executed at that moment (see Figure 3.6).

Figure 3.6: A CPN Place in Interaction Model

**CPN Transitions and CPN Arcs**

In a CPN, transitions are linked with a number of input and output places by directed arcs. In our framework, CPN transitions and arcs are used to carry interaction policies of a protocol.

For a CPN model, directed arcs between transitions and places determine token transferring directions. Guard functions and the arc functions determine which tokens can be transferred through the transition (i.e. fire the transition)
and how the token value be changed, respectively (see Figure 3.7). To use a CPN model to represent an interaction protocol, transitions in the model are used to denote interaction related actions. Guard functions and arc functions are used to carry execution conditions of actions.

Furthermore, the firing of a transition consumes the input token(s) from the input places as computing parameter(s), and generates new token(s) in the output place(s). This leads to the state (marking) of a CPN model to change. In our framework, we use transition firings to represent changing of states and transferring of message in agent interactions.

![Figure 3.7: A CPN Transitions in Interaction Models](image)

**CPN Models for Interaction Protocols**

After introducing the meanings of CPN components, in this subsection, we introduce how to use CPNs to model agent interaction protocols. Here we take the FIPA Inform protocol as an example to demonstrate how to use CPNs to model agent interaction protocols.
Figure 3.8 shows the UML model for the FIPA Inform protocol. From this model, it can be seen that there are two executions in this protocol, which are inform and process. These two executions may lead to the interaction transformation among three states, i.e. before inform, informed and processed.

![Figure 3.8: UML Model for FIPA Inform Protocol](image)

To describe the FIPA Inform protocol in Figure 3.8 into a CPN model, firstly, we can define four places corresponding to the three possible states of the interaction: a Start Place for the start state, a Received Place and a Terminated1 Place for the informed state, and a Terminated2 Place for the processed state. Secondly, two transitions, i.e. the Inform Transition and the Process Inform Transition, are defined to represent the informing and processing message activities in the FIPA Inform protocol, respectively. Finally, some related guard functions and arc functions are defined to check tokens’ formats and modify related flags of tokens. The formal description of the CPN model is shown in Table 3.3. The graphical representations of the CPN model are shown from Figure 3.9 to Figure 3.11.

The change of states in an interaction can be described as follows:

1. To start the interaction, at least one message token should be assigned
3.3. A CPN Based Framework for Agent Interactions

\[
CPN = (\Sigma, P, T, A, N, C, G, E, I)
\]

\[
P = \{\text{Start}, \text{Received}, \text{Terminated1}, \text{Terminated2}\}
\]

\[
T = \{\text{Inform}, \text{Process Inform}\}
\]

\[
\Sigma = \{\text{MESSAGE}\}
\]

\[
G = \{\text{messageGood}(\text{message})\}
\]

\[
E = \{\text{send}(\text{message}), (n = n+1)\}
\]

Table 3.3: Formal Description for the CPN Model of FIPA Inform

to the \textit{Start Place}. If this token satisfies the constraints specified in the
\textit{guard function} \textit{MessageGood()}, the \textit{Inform Transition} will be enabled. This
means that the agent is ready to Inform another agent (see Figure 3.9).

2. After the \textit{Inform Transition} is fired, a message token will be removed from
the \textit{Start Place}, and a new token will be generated in the \textit{Received Place} and
the \textit{Terminated1 Place}, respectively. At this moment, the \textit{Process Inform}
\textit{Transition} is enabled (see Figure 3.10).

3. Finally, after the \textit{Process Inform Transition} is fired, both \textit{Terminated1 Place}
and \textit{Terminated2 Place} will have a token (see Figure 3.11). This denotes
that a FIPA inform interaction is terminated.

3.3.3 Communication Interface between Protocols and Agents

By dividing agents and interaction protocols into two layers, agent interactions
can be mediated by the protocol layer. As a result, an agent interaction be-
comes a “token playing game”. Agents can achieve and control interactions
through placing/removing tokens to/from corresponding CPN places in the pro-
tocol layer. Most CPN development tools provide communication infrastructures
to allow communications between CPN models and external programs/processes.
3.3. A CPN Based Framework for Agent Interactions

Figure 3.9: CPN Model for FIPA Inform Protocol (Ready to Inform)

Figure 3.10: CPN Model for FIPA Inform Protocol (Informed)

Figure 3.11: CPN Model for FIPA Inform Protocol (Processed)
In this research, Comms/CPN [12], the communication infrastructure provided by CPN/Tools, is used to establish the communication between agents and the protocol layer.

Comms/CPN is designed to act as an interface between CPN models, which are developed by CPN/Tools, and external applications. In this research, we use Comms/CPN as the communication interface between agents and the protocol layer. The structure of Comms/CPN is shown in Figure 3.12 [56]. It consists of three modules, which are organised as layers. The Communication Layer contains an interface to TCP/IP and all TCP/IP and socket related functions. The Messaging Layer is responsible for transferring data between CPN models and agents. The Connection Management Layer manages multiple connections between external applications/agents and CPN models.

Figure 3.12: Structure of Comms/CPN
3.3. A CPN Based Framework for Agent Interactions

Comms/CPN can accept connections from various applications that are composed in different languages, such as Java, C++ applications, etc. These applications communicate with Comms/CPN through TCP/IP sockets. Agents are developed by using Java, and a Java/CPN [56] interface is integrated with agents to realise communications between agents and Comms/CPN. The Java/CPN interface is shown Figure 3.13. To communicate with the protocol layer, an agent uses the \textit{connection} method to establish the connection with the Comms/CPN. The connection method takes a host name and port number as arguments. The host name and port number are corresponding to the CPN model (i.e. interaction protocol) that an agent wants to adopt. Once a connection has been established (i.e. the socket is opened), input and output streams are extracted from the socket to allow (the agent) sending/receiving messages to/from the CPN model by using the \textit{accept} and \textit{send} methods. Finally, the agent uses the \textit{disconnect} method to close the input/output stream and the socket, so that the connection between the agent and the protocol layer is terminated.

```java
public interface JavaCPNInterface {
    public void connect(String hostName, int port) throws IOException;
    public void accept(int port) throws IOException;
    public void send(ByteArrayInputStream sendBytes) throws SocketException;
    public ByteArrayInputStream receive() throws SocketException;
    public void disconnect() throws IOException;
}
```

Figure 3.13: Java/CPN Interface
3.4 A CPN Based Approach for Flexible Agent Negotiations

In Section 3.3, we introduced the method of how to use CPN techniques to coordinate flexible agent interactions. In this section, we propose a CPN based approach to coordinate flexible agent negotiations.

Agent negotiation is an important type of interaction, especially for self-interested MASs such as agent-based E-commerce [107] [149] systems, and agent-based Grid systems [160] [161]. In open environments, multiple negotiations can happen concurrently. Also, factors including negotiation protocols, negotiation participants and market statuses are not fixed. How to capture and model these dynamic factors is one of the most important issues for agent negotiations.

In general, activities of an agent negotiation are proceeded in a series of rounds and can be depicted as follows.

1. Negotiations can happen when there are at least two agents of the opposite types (e.g. one buyer and one seller).
2. Agents propose their preferred deal conditions (such as price) in their proposals.
3. Agents send proposals to their negotiation partners by following some protocols, such as Rubinstein’s alternating offers protocol [147].
4. A negotiation proceeds to the next round if no agreement is reached.
5. A negotiation is terminated when an agreement or some predefined terms (such as deadline) are reached.

In this section, we introduce a CPN based approach to handle and model activities in agent negotiations. In this approach, CPN tokens are used to carry
negotiation proposals of agents and represent negotiation protocols in CPN models. Agents then can operate negotiations by exchanging CPN tokens via CPN models.

### 3.4.1 Using CPN Tokens to Conduct Negotiation Proposals

To use CPN tokens to conduct negotiation proposals, we define a PRO colour set to prescribe the data format that agents need to follow to compose their proposals. The PRO colour set is shown in Table 3.4. This colour set contains seven parameters, where the seller indicates the unique ID of a seller agent; the buyer indicates the unique ID of a buyer agent; \( p \) is the price that the agent offers in the current round; deadline \((T)\) is the total round of the negotiation; round \((t)\) is the number of processed rounds; dealFlag \((df)\) is the flag that indicates whether the negotiation has successfully proceeded; and sendFlag \((sf)\) is the controlling flag that indicates whether the Proposal Token is ready for sending. The meanings of parameters in the PRO colour set are described in Table 3.5.

|---|---|

#### Table 3.4: The PRO Colour Set

To conduct a proposal, agents need to compose Proposal Tokens. Agent IDs of negotiation parties and preferred prices \( p \) of agents are described in Proposal Tokens. Agents then send their Proposal Tokens to the corresponding CPN model that represents a particular negotiation protocol. Through firing transitions of the CPN model, negotiation proposals of agents can be transferred among agents (of different negotiation parties) automatically.
3.4. A CPN Based Approach for Flexible Agent Negotiations

Parameter & Values & Meanings
\begin{tabular}{|l|l|}
\hline
seller: STRING & Seller agent’s ID \\
\hline
buyer: STRING & Buyer agent’s ID \\
\hline
$p$: INT & Preferred price \\
\hline
$\text{round} (t)$: INT & Number of processed rounds \\
\hline
$\text{deadLine} (T)$: INT & Number of total rounds \\
\hline
$\text{sendFlag} (sf)$: BOOL & $sf = true$: Token is ready to send \\
& $sf = false$: Token is sent \\
\hline
$\text{dealFlag} (df)$: INT & $df = 0$: Negotiation is processing \\
& $df = 1$: Agreement meets \\
& $df = 2$: Negotiation is failed \\
\hline
\end{tabular}

Table 3.5: Parameter Meanings of the PRO Colour Set

3.4.2 CPN Model for Individual Agents

After defining the PRO colour set for negotiation proposals, we can construct corresponding CPN models to express negotiation protocols. Negotiation activities of an individual agent (buyer or seller) can be simply described by Figure 3.14. In this CPN model, five CPN transitions and two CPN places are used to represent negotiation activities of an individual agent. The Proposal Place is used to conduct Proposal Tokens which are composed by an agent. This is the input place of the Send Proposal Transition. The Send Proposal Transition denotes the sending proposal activity of an agent. Then the Send Proposal Transition fires and a Proposal Token(s) is transferred to the other party of a negotiation. The Received Place is used to contain incoming Proposal Tokens from other agents. Received Place is an output place of the Receive Transition, which denotes the receiving proposal activity of an agent. When an agent receives a Proposal Token from others, it will evaluate the incoming proposal, and modify the incoming proposal if the agent cannot accept the proposal. These activities are represented in the Consider&Modify Transition. In addition, the Deal Transition and Fail
3.4. A CPN Based Approach for Flexible Agent Negotiations

Transition are used to represent the success and failure of a negotiation, respectively.

3.4.3 CPN Model for Simple One-to-One Negotiation

The CPN model for an one-to-one negotiation can be constructed through linking the CPN models of two individual agents. Figure 3.15 shows the CPN model for a simple one-to-one negotiation. In this model, we define two places, i.e. Sellers’ Place and Buyers’ Place, to conjunct the CPN models of two individual agents. As the output place of the Send Proposal Transition of a seller/buyer agent, the Sellers’ Place/Buyers’ Place conducts Proposal Token(s) which is/are sent by a seller/buyer. Sellers’ Place/Buyers’ Place is also the input place of the Receive Transition, and the firing of Receive Transition will import the Proposal Token(s) which is/are sent by the buyer/seller agent from the Sellers’ Place/Buyers’ Place to the Received Place of the seller/buyer agent.
3.4.4 The Generic CPN Model for Negotiation

In the previous subsection, we introduced the method of constructing a simple one-to-one negotiation. Similarly, we can also compose the CPN model for a n-to-n \((n \geq 1)\) negotiation by linking CPN models of \(n\) individual agents (see Figure 3.16). However, if the negotiation type (one-to-one, one-to-many or many-to-many) and the number of negotiation agents are not fixed, the links between individual CPN models will be an unfixed factor. This unfixed factor makes it impractical and too complicated to construct a CPN model without specifying the negotiation type and the number of negotiation agents. In order to capture these two uncertain features, we develop a generic CPN model for handling agent negotiations with different negotiation types and multiple negotiation partners. The generic CPN model identifies the composing and receiving agents of a Proposal Token according to colours of Proposal Tokens. In addition, we define a set of guard functions on corresponding transitions to control the transferring of Proposal Tokens.

Here, we take Rubinstein’s alternating offers negotiation protocol [147] as an
3.4. A CPN Based Approach for Flexible Agent Negotiations

Figure 3.16: CPN Model for Many to Many Negotiation

example (other negotiation protocols can also be modelled in CPNs). According to Rubinstein’s protocol, in each round of a negotiation, each agent has a desired space of possible deals. This space consists of the most desirable price and the least desirable (reserved) price of the agent. If the agent is the proposal sender in the current round, it will propose a deal within its desired space in the proposal. On the other hand, as a proposal receiver, the agent will consider the received proposal by checking whether the received price is within its desired space. After each round, agents will modify their desired spaces by using their concession functions or schemes if no agreement is reached (different agents may use different concession functions or schemes).

The CPN model of Rubinstein’s protocol is shown in Figure 3.17, and the formal description of this CPN model is shown in Table 2. In this model, we define five “public” places and one transition to link buyers and sellers together and control the negotiation process. The Start Transition and the Start Place are used to control the start of negotiations. The Seller’s Proposal Place and the Buyer’s Proposal Place contain the Proposal Tokens from sellers and buyers,
respectively. The *Dealt Place* contains *Proposal Tokens* that are dealt by agents. The *Failed Place* contains *Proposal Tokens* that have reached deadlines.

\[
CPN = (\Sigma, P, T, A, F, C, G, E, I)
\]

\[
P = \{\text{Start, Received, Proposal, Buyer’s Proposal, Seller’s Proposal, Dealt, Failed}\}
\]

\[
T = \{\text{Start, Send Proposal, Receive, Consider\&Modify, Deal, Remove Dealt, Fail, Remove Failed}\}
\]

\[
A = P \times T \cup T \times P
\]

\[
\Sigma = \{\text{PRO}\}
\]

\[
G = \{\text{checkName()}, \text{checkDeal()}, \text{checkSend()}, \text{acceptConnection()}, \text{CommAgent()}\}
\]

\[
E = \{\text{sent()}, \text{pro2:PRO}\}
\]

\[
C = P \rightarrow \Sigma
\]

\[
F = A \rightarrow P \times T \cup T \times P
\]

\[
I = 1'(9999) \text{ in the Start Place}
\]

Table 3.6: CPN Negotiation Protocol Formal Description

For each negotiation party (i.e. seller or buyer), we define seven transitions and two places to represent its negotiation activities. The outgoing *Proposal Tokens* (refer to Subsection 3.4.1) of an agent are contained in the *Proposal Place* of the agent. Tokens will be sent to corresponding negotiation partners after the *Send Proposal Transition* is fired. The incoming *Proposal Tokens* from negotiation partners are transferred into the *Received Place* of the agent through firing the *Receive Transition* of the agent. If the *Received Place* contains token(s) (i.e the agent receive proposal(s) from its negotiation parter(s)), the *Consider\&Modify Transition* will be enabled. The *Consider\&Modify Transition* is an interface that links agents and the CPN negotiation model (see Subsection 3.4.6). *Proposal Tokens* will be sent to negotiation agents after the *Consider\&Modify Transition* fires.
After reading the received proposal, the agent will decide whether to accept or reject it. If the agent accepts the proposal, it will modify the \textit{dealFlag} (refer to Table 3.4 and Table 3.5) of the \textit{Proposal Token} to 1 and send the token back to the \textit{Proposal Place}. If the agent cannot accept the incoming proposal, it will generate a new proposal according to the market status and its demands. Then, it will create a new \textit{Proposal Token} and send the token back to the \textit{Proposal Place} of the CPN model.

The value of \textit{dealFlag} of a \textit{Proposal Token} controls the firing of the \textit{Deal Transition} and \textit{Fail Transition}. If there is a \textit{Proposal Token} where \textit{dealFlag} = 1, the \textit{Deal Transition} will be enabled. Then, the token will be removed from the \textit{Proposal Place} after the \textit{Deal Transition} fires. This denotes that the corresponding agent agrees to the incoming proposal. If there is a \textit{Proposal Token} with \textit{dealFlag} = 2, the \textit{Fail Transition} will be enabled. After the \textit{Fail Transition} fires, the corresponding token will be removed from the \textit{Proposal Place}. \textit{dealFlag}'s value changes to 2 only if \textit{round} > \textit{deadline}. The value of \textit{round} (refer to Table 3.4) of a \textit{Proposal Token} increases one after each time the \textit{Send Proposal Transition} fires. This means that the negotiation proceeds to the next round.

The CPN model of Figure 3.17 is a generic model for agent negotiation. Any agents can operate negotiations (by using Rubinstein’s protocol) by sending/receiving \textit{Proposal Tokens} to/from this CPN model. In this model, \textit{checkName()} is a guard function (see Figure 3.17) that verifies the values of \textit{buyer} and \textit{seller} in \textit{Proposal Tokens} (refer to Table 3.4). Bound with \textit{checkName()}, a transition (such as the \textit{Receive Transition}) is enabled only if tokens of its input places are from the same pair of agents (with same \textit{buyer} and \textit{seller} IDs). Hence we can use token colours to distinguish different pairs of negotiation agents. Therefore, this CPN model is a generic model that can handle any negotiation cases, i.e. one-to-one, one-to-many or many-to-many.
Figure 3.17: The CPN Model for Rubinstein’s Negotiation Protocol
3.4.5 A CPN Based Approach for Flexible Negotiations

In the Subsection 3.4.4, we have introduced using CPN models to represent negotiation protocols. In this subsection, we introduce a CPN based approach that can enable heterogeneous agents to use different protocols to operate negotiations.

Layered Architecture

In many current MASs, negotiation protocols are hard-coded within agents. For such MASs, it is very hard to allow agents to select different negotiation protocols flexibly. Therefore, hard-coding protocols within agents actually conflicts with the dynamic feature of open environments, and greatly reduces the flexibility and extensibility of MASs.

In our CPN based approach, we develop a layered architecture which separates negotiation protocols from agents. The layered architecture of the approach is shown in Figure 3.18. The architecture contains three layers, which are the Protocol Layer, the Media Layer, and the Agent Layer. The Protocol Layer conducts CPN models of negotiation protocols which may be adopted by agents. All CPN models in the Protocol Layer have distinguished I/O ports which can allow them to be connected with agents. The Media Layer contains two facilitators, i.e. the Agent Register and the Port Manager. The Agent Register records and manages the port information of agents. The Port Manager manages the ports of CPN models in the protocol layer. In this architecture, negotiation agents are in the Agent Layer.

Flexible Negotiations

By using the layered architecture, we separate negotiation protocols from agents. This enables agents the flexibility to choose different negotiation protocols. In addition, heterogeneous agents can interact with each other to join a negotiation.

In this approach, agents need to satisfy two conditions to be included in
the system and join negotiations with others. Firstly, agents do not need to be homogeneous, but they must be able to understand the data structure of the \textit{PRO} colour set (refer to Table 3.4 and Table 3.5) and compose \textit{Proposal Tokens} to express their negotiation proposals. Secondly, to join negotiations, agents must register to the \textit{Agent Register} of the \textit{Media Layer} by providing their related port information.

To join a negotiation, an agent specifies its preferred negotiation protocol to the \textit{Port Manager}. Then the \textit{Port Manager} will respond the I/O port number of
the corresponding CPN model. With the port number, the agent will be able to send/receive Proposal Tokens to/from the CPN negotiation protocol.

3.4.6 Interface between CPN Model and Agents

Comms/CPN contains an interface to TCP/IP and all TCP/IP and socket related functions. By defining socket ports in CPN net structures, CPN models can receive/send data from/to external agents. For example, in the CPN model of Figure 3.17, two transitions are combined with socket ports, i.e. the Start Transition and the Consider&Modify Transition (see Figure 3.19 and Figure 3.20). The Start Transition is combined with an guard function called acceptConnection(). This function waits for connection requests from a particular socket (in Figure 3.19 the host name and port number of the socket are Conn1 and 9000, respectively). If an agent opens a TCP/IP socket with the same host name and port number as defined in acceptConnection(), the Start Transition will be enabled. This means a negotiation will start.

![start action acceptConnection("Conn 1", 9000);](image)

Figure 3.19: The Start Transition

During a negotiation, the Consider&Modify Transition is the transition that sends and receives tokens from agents. The Consider&Modify Transition uses a guard function called CommAgent() to send old Proposal Tokens and to receive new Proposal Tokens from agents.

Through the above two transitions, the CPN negotiation model will be able to communicate with agents. In addition, agents will use methods defined in the Java/CPN interface to send and receive data with the CPN model.
3.4. A CPN Based Approach for Flexible Agent Negotiations

3.4.7 Negotiation Processes

In the previous subsections, concepts and architecture of the CPN based approach for coordinating agent negotiation have been proposed. In this subsection, processes of a one-to-one negotiation and a one-to-two negotiation are demonstrated to show how negotiations are modelled in the CPN based approach.

Example One: One-to-One Negotiation Processes

From Figure 3.21 to 3.30, an example of one-to-one negotiation is given. In this example, there are a seller agent $s01$ and buyer agent $b01$, involved in a negotiation by using Rubinstein’s protocol. The deadline of the negotiation is 10 rounds. At the beginning of the negotiation (round 0), the price in the proposal of $s01$ and $b01$ are 500 and 300, respectively. The negotiation is shown in the following process.

1. At round 0, $s01$ and $b01$ generate Proposal Tokens which conduct their preferred prices ($p = 500$ and $p = 300$), and place the Proposed Tokens in seller’s Proposal Place and buyer’s Proposal Place, respectively (see Figure 3.21);
2. At round 1, $s01$ sends its token to $b01$ by firing the Send Proposal Transition (of the seller side). Then, $s01$’s token is received by $b01$ after the Receive Transition (of the buyer side) fires. Then, the Consider&Modify Transition of the buyer side is enabled because its only input place, i.e. the Received Place, contains a token (see Figure 3.22);

3. After the Consider&Modify Transition fires, the received token from $s01$ is transferred to the buyer agent $b01$. After receiving the token, $b01$ modifies its proposal and increases the value of price from 300 to 320 (by using the concession function of $b01$). This process is described in Figure 3.23;

4. At round 2, $b01$ sends its token to $s01$ by firing the Send Proposal Transition on the buyer side. Then, $b01$’s token can be received by $s01$ after the Receive Transition (of the seller side) fires. Then, the Consider&Modify Transition of the seller’s side is enabled (see Figure 3.24);

5. $s01$ receives the Proposal Token from $b01$ after the firing of the Consider&Modify Transition. Then, $s01$ modifies its proposal and decreases the value of price in the proposal (see Figure 3.25);

6. After several rounds, $s01$ agrees on the received proposal from $b01$ (the round number depends on the concession function used by agents). $s01$ modifies the value of dealFlag of the token to 2, which denotes the negotiation meets a deal. Then, the Deal Transition on the seller side is enabled (see Figure 3.26);

7. After the Deal Transition fires, the Proposal Token is transferred from seller’s Proposal Place to the Dealt Place. Also, the token in the Deal Place will enable the Remove Dealt Transition (see Figure 3.27);

8. After the Remove Dealt Transition fires, $b01$’s token is removed from the Proposal Place on the buyer’s side to the Dealt Place. The negotiation is
accomplished successfully. This process is shown in Figure 3.28. (Note: \( b01 \) can also accept the proposal of \( s01 \) first. In this case, \( b01 \)'s token will be removed by the \textit{Deal Transition} on the buyer's side, and \( s01 \)'s token will be removed by the \textit{Remove Deal Transition} on the seller's side);

9. If the negotiation between \( s01 \) and \( b01 \) cannot meet a deal when reaching the deadline, the \textit{Fail Transition} and \textit{Remove Failed Transition} are enabled when \( \text{round} \geq \text{deadline} \). These two transitions will remove the \textit{Proposal Tokens} of \( s01 \) and \( b01 \) to the \textit{Failed Place}. This denotes that the negotiation is terminated unsuccessfully. This case is demonstrated in Figure 3.29 and 3.30.
3.4. A CPN Based Approach for Flexible Agent Negotiations

Figure 3.21: One-to-One Negotiation: the seller and the buyer place Proposal Tokens to Proposal Places
Figure 3.22: One-to-One Negotiation: the seller sends Proposal Token to buyer
3.4. A CPN Based Approach for Flexible Agent Negotiations

Figure 3.23: One-to-One Negotiation: the buyer considers and modifies the received proposal
Figure 3.24: One-to-One Negotiation: the buyer sends modified Proposal Token to the seller
3.4. A CPN Based Approach for Flexible Agent Negotiations

Figure 3.25: One-to-One Negotiation: the seller considers and modifies the proposal of the buyer
Figure 3.26: One-to-One Negotiation: the seller agrees on the proposal of the buyer
3.4. A CPN Based Approach for Flexible Agent Negotiations

Figure 3.27: One-to-One Negotiation: the negotiation meets a deal
Figure 3.28: One-to-One Negotiation: the Proposal Token of buyer is removed by the Remove Deal Transition, the negotiation is accomplished successfully.
3.4. A CPN Based Approach for Flexible Agent Negotiations

Figure 3.29: One-to-One Negotiation: no agreement is met before the deadline, i.e. the negotiation fails
Figure 3.30: One-to-One Negotiation Process: the negotiation terminates unsuccessfully
Example Two: a One-to-Two Negotiation Processes

From Figure 3.31 to 3.34, another example is given to demonstrate a one-to-two negotiation. In this example, seller agent $s01$ operates negotiations with buyer agent $b01$ and $b02$ concurrently (by using Rubinstein’s protocol). The deadline of the negotiation is 10 rounds. At the beginning of the negotiation (*round* 0), the price in the proposal of $s01$, $b01$ and $b02$ are 500, 380 and 100, respectively. The negotiation is shown in the following process.

1. At *round* 0, $s01$, $b01$ and $b02$ place their *Proposal Tokens* in corresponding *Proposal Places* (see Figure 3.31);

2. At *round* 1, $s01$ sends its tokens to the buyers by firing the *Send Proposal Transition* (of the seller side). Then, $s01$’s tokens are transferred to the *Received Place* on the buyer’s side (see Figure 3.32);

3. The *checkName()* and *CommAgent()* functions on the *Consider&Modify Transition* allocate received *Proposal Tokens* to $b01$ and $b02$ according to values of $Buser$’s $ID$ in the tokens;

4. $b01/b02$ modifies its proposal by increasing the *price* in the proposal (see Figure 3.33), then sends the revised proposal to $s01$. After receiving the revised tokens from $b01$ and $b02$, $s01$ also uses similar processes to modify and send the corresponding *Proposal Tokens* to $b01$ and $b02$;

5. If the negotiation can meet a deal before the deadline, corresponding tokens will be moved from *Proposal Places* to the *Dealt Place* after the firing of the *Deal Transition* and *Remove Dealt Transition*. Otherwise *Fail Transition* and the *Remove Failed Transition* will move the *Proposal Tokens* to the *Failed Place* 3.34.
The previous two examples demonstrate the general processes of one-to-one/one-to-many negotiations. It can be seen that in the CPN based approach, agent negotiations are operated through communications between agents and CPN models. Any numbers of agents can participate in a negotiation by sending/receiving Proposal Tokens to/from a CPN model. CommAgent() and checkName(), which are guard functions of the Consider&Modify Transition, identify agent IDs on Proposal Tokens and allocate tokens to corresponding agents via sockets. Hence, by using the proposed CPN model, many-to-many negotiations can also be handled.
3.4. A CPN Based Approach for Flexible Agent Negotiations

Figure 3.31: One-to-Two Negotiation: the seller and buyers place their *Proposal Tokens* in *Proposal Places*
Figure 3.32: One-to-Two Negotiation: the seller sends its Proposal Tokens
Figure 3.33: One-to-Two Negotiation: buyers modify proposals according to the received proposal from the seller
3.4. A CPN Based Approach for Flexible Agent Negotiations

Figure 3.34: One-to-Two Negotiation: negotiations terminates successfully/unsuccessfully
3.5 Advantages of CPN Based Interactions

Many current agent interaction coordination methods are based on static state-based diagrams, and normally cannot be executed directly [66]. By applying CPN techniques, our CPN based approaches can overcome these shortcomings and have the following advantages:

1. *Generic interaction models*

   CPNs can be used to construct generic interaction models in MASs. Any number of agents can start interactions through communicating with the CPN protocol layer. In addition, these interactions can be one-to-one, one-to-many and many-to-many. However, most other interaction mechanisms do not have a generic model that can handle interactions with different agent numbers.

2. *Separating interaction protocols from agents*

   By describing interaction protocols as CPN models, agents can avoid being hard-coded with interaction protocols. This feature actually separates agents from interaction infrastructures and makes agents more independent. This feature also greatly enhances the flexibility of agents in MASs.

3. *Handling interactions among heterogeneous agents*

   By using CPNs to model agent interactions, agent interactions will be mediated by CPNs. Agents achieve interactions through sending and receiving data with CPN interaction models. Hence, there is less requirement for agent structures and origins. Agents only need to follow the message data structure which is described in the related CPN colour set, and have TCP/IP socket.

4. *Direct simulations and executions*
CPNs can include functions and some operations in transitions and arcs, so that rules and operations can be embedded into CPN models directly. Furthermore, CPN models can be directly simulated and executed. This feature brings much convenience for evaluations and analysis of agent interactions.

5. Linking with formal language support

CPN models can be represented in formal languages such as XML and CPN ML. The formal CPN descriptions can be conducted in some text-based files, such as XML files. These files can be transferred among different MASs, and extracted in a MAS if a MAS includes a CPN interpreter. This greatly enhances the reusability and extensibility of MASs.

6. Graphical and mathematical representation

PNs and CPNs have both graphical and mathematical representations. The mathematical representation of the CPN allows automatic analysis and evaluation of agent interactions. In Chapter 4, we will introduce some related coordination mechanisms that utilise CPN mathematical representations for analysis purposes.

3.6 Summary

In summary, with the increasing needs of robust and flexible agent interactions, the shortcomings of traditional agent interaction mechanisms are rised up. Most current interaction mechanisms require agents to be hard-coded with interaction protocols, and have limitations for interactions between heterogeneous agents. These features greatly reduce the flexibility, extensibility and robustness of MASs. To cover some of these shortcomings, we use CPN techniques to coordinate agent interactions. By using CPNs, a generic interaction model can be created. The generic model can be used to handle interactions with various agent numbers.
In this chapter, we proposed a CPN based framework for agent interactions. This framework separates agents and interaction protocols into two different layers and hires CPN communication infrastructure to link agents and interaction protocols (modelled in CPNs). Agent interactions are changed to a token playing game, which can be operated by various agents. Therefore, heterogeneous agents can achieve interactions within the CPN based framework. To demonstrate the CPN based interaction coordination mechanisms, we implement the CPN based framework for coordinating agent negotiation activities. Through the demonstration, it can be seen that CPNs can greatly enhance the flexibility of agent interactions.
Chapter 4

Coloured Petri Net Based Interaction Analysis

In the previous chapter, we introduced a CPN based approach which uses CPN models to represent agent interaction protocols and separate interaction protocols from agents. This framework mainly utilises the modelling and representation power of CPNs to coordinate agent interactions. Beside modelling and representation, PNs and CPNs can also provide techniques to analyse the behaviour of modelled systems.

Firstly, the PN theory provides a set of net analysis techniques and algorithms that allow developers to evaluate various net properties of PN models, including safeness, boundedness, conservation, liveness, reachability, etc. Through analysing these properties, we can evaluate whether the structure of a PN/CPN model is robust and reasonable.

Secondly, benefitting from the executable feature of the PN/CPN, we can analyse the transition firing conditions of a PN/CPN, and predict the firing results of transitions. Through these predictions and analysis, an agent can know whether it can achieve an interaction, and what it can obtain from the interaction. This will allow agents to make more reasonable decisions before involving an interaction.

In this chapter, we introduce a CPN based approach that utilises mathematical representations of CPNs to analyse and predict agent interactions. In this approach, an agent composes its intentions in a CPN model and sends a message
that conducts the CPN model to its interaction objective (another agent). The agent that receives the message can analyse and predict the CPN model and give responses according to the analysis/prediction results.

This chapter is arranged as follows. In Section 4.1, some related work that uses CPN techniques to analyse agent interaction models is introduced. In Section 4.2, some related analysis techniques provided by CPNs are described. In Section 4.3, a CPN based approach that enables agents to analyse interaction protocols and form flexible interactions is proposed. Then, a potential application of this approach is given in Section 4.4. Finally, this chapter is summarised in Section 4.5.

4.1 Related Work

CPN techniques have been applied for agent coordinations by many MAS researchers. In Section 3.2, some significant CPN based coordination methods in current MAS research were introduced and reviewed. However, most of these methods focus on using CPNs as a tool to model, represent and simulate agent interactions. This section introduces some related work that adopts PN and CPN analysis techniques in agent coordinations.

In [106], Ling used PNs as a system engineering tool for developing MASs. In this work, well-established analysis methods of PNs are used to detect properties of agent interaction. In addition, these MAS PN models can be used as a basis for synthesising skeleton codes of interacting agents.

PNs and CPNs are used to model and analyse mobile agent systems by many researchers. In [105], Ling developed a PN based work-flow net to model the behaviour of mobile agents as a business process, and applied PN analysis methods to avoid dead-lock problems. Ling also applied the work-flow net to analyse itineraries of mobile agents [104]. Xu [184] [185] developed a PN based approach to model mobile agents and their mobilities. In this approach, high-level PNs are
hired to control and exam migration procedures of mobile agents. Rana developed a PN based approach to model interactions among stationary-mobile agents. In this approach, Rana [142] provided several PN blocks to allow the automatic construction of mobile agent systems. Such blocks can also be used to obtain visual feedbacks on design parameters involved in an agent based system. Koriem [90] also used PNs to facilitate the creation of mobile agents. In addition, Koriem developed PN models to evaluate the performance of mobile agent networks on the aspects of agent communication time, size of mobile agent, number of mobile agents and number of hops.

Some researchers have worked on agent interaction coordinations from a “protocol engineering” view [46] [47] [115]. El Fallah-Seghrouchni [46] [47] introduced the concepts of protocol engineering, and hired CPN formalism for interaction protocol modeling tool. Mazouzi [115] extended El Fallah-Seghrouchni’s work and proposed a generic approach for analysis, specification and verification of agent interaction protocols. This approach evaluates interaction protocols through analysing several CPN properties, including structural liveness, boundedness, etc.

In summary, many agent and MAS researchers have realised the advantages of PN/CPN techniques and applied PNs/CPNs not only as a system modelling and describing tool but also as a system analysis and verification tool. Some analysis and verification techniques provided by PNs/CPNs are especially suitable for evaluations of the performance and robustness of mobile agent systems and MASs. Some important properties of interaction models, such as consistency and deadlock, can be directly analysed by checking the net properties of corresponding PNs/CPNs. Compared with the related work introduced in this section, our work focuses more on utilising mathematical formats of PNs/CPNs to achieve interaction analysis and predictions in MASs. We develop a mechanism that allows an agent to not only understand incoming interaction proposals, but also to predict the effects of those interactions by analysing the formal PN/CPN
models. We demonstrate that agents can have more initiatives in interactions and make more reasonable decisions. This mechanism is especially suitable for self-interested MASs as agents can evaluate interactions and select interactions they prefer. In addition, the mechanism can make agents more adaptable to cope with dynamic environments.

4.2 Analysis Techniques of PNs and CPNs

The PN theory provides several analysis methods to evaluate properties of PN and CPN models. Most of these methods are based on the formal definitions of PNs and CPNs. In this section, we will introduce the mathematical representations of PNs and CPNs.

4.2.1 PN Analysis

As introduced in Subsection 3.1.1, a PN contains a set of transitions \( T \) and a set of places \( P \). According to the PN theory, the links between \( T \) and \( P \) of a PN can be expressed by an input function \( I \) and an output function \( O \).

**Definition 4.1.** For a PN \( N \) with a set of places \( P \) and a set of transitions \( T \), \( I : T \to P^\infty \) is the input function of \( N \), a mapping from \( T \) to the input places of \( N \); \( O : T \to P^\infty \) is the input function of \( N \), a mapping from \( T \) to the output places of \( N \).

According to the Matrix Equation Method in the PN theory [61] [132], the net structure and input/output function of a PN can be represented by a set of matrices, which are defined in the following definitions [61].

**Definition 4.2.** For a PN \( N \) with a set of places \( P \) and a set of transitions \( T \), \( Pre \in N^{\|P\|\times\|T\|} \) is defined as the backward incidence matrix of \( N \). \( Pre[p, t] = n(n > 0) \), where \( p \in P \) and \( t \in T \), denotes that there is \( n \) arcs from the place \( p \) to the transition \( t \).
Definition 4.3. For a PN $N$ with a set of places $P$ and a set of transitions $T$, $Post \in N^{[P \times \{T\}}$ is defined as the forward incidence matrix of $N$. $Post[p, t] = n(n > 0)$, where $p \in P$ and $t \in T$, denotes that there is $n$ arcs from the transition $t$ to the place $p$.

Definition 4.4. For a PN $N$ with a backward incidence matrix $Pre$ and a forward incidence $Post$, where $Pre, Post \in N^{[P \times \{T\}}$, $C = Post - Pre$ is defined as the incidence matrix of $N$.

For example, if we have a PN $N$ in Figure 4.1, $N$ can be represented as Equation 4.2.

![Image of Petri Net N](image-url)
According to the value of the Pre/Post matrix and the marking of a PN, we can analyse which transitions can be enabled so as to predict the future marking of the PN after a transition is fired.

**Definition 4.5.** Suppose \( N \) is a PN with \( T, P, \text{Pre}, \text{Post} \) and \( C \). \( \mu \) is the current marking \( N \). A transition of \( t_j \in T \) can be represented by a vector \( e[j] \) which is zero everywhere except in the \( j^{th} \) component. Transition \( t_j \) is enabled if

\[
\mu \geq \text{Pre} \cdot e[j] \quad (4.2)
\]

The result-marking \( \mu'_j \) of firing transition \( t_j \) can be calculated by using the following equation.

\[
\mu'_j = \mu + (\text{Post} - \text{Pre}) \cdot e[j] = \mu + C \cdot e[j] \quad (4.3)
\]
Let’s take the PN model in Figure 4.1 as an example. According to Definition 4.5, it can be seen that the firing condition of $t_1$ is to have a marking which is greater than

$$P_{re} \cdot e[j] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = (1, 0, 0, 0, 0, 0)$$ (4.4)

Since the current marking is $\mu = (2, 0, 0, 0, 0, 1) \geq (1, 0, 0, 0, 0, 0)$, hence $t_1$ is enabled. In addition, according to Equation 4.4, we can also find out the marking of $N$ after $t_1$ is fired through the calculation in Equation 4.5. The marking of the PN after firing of $t_1$ can be shown in Figure 4.2.

$$\mu'_1 = (2, 0, 0, 0, 0, 1) + \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & -2 \\ 0 & -1 & 2 \\ 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}
= (2, 0, 0, 0, 0, 1) + (-1, 1, 0, 1, 0, 0)
= (1, 1, 0, 1, 0, 1)$$ (4.5)

4.2.2 CPN Analysis

As introduced in Subsection 3.1.2, a CPN is more complex than a PN mainly in the following aspects:
4.2. Analysis Techniques of PNs and CPNs

1. A place of a CPN has a place type which denotes the tokens that the place can contain;

2. A transition of a CPN can be combined with guard functions, which denote the firing conditions of the transition.

3. An arc of a CPN can conduct arc functions that control transformations of tokens.

CPNs can also be converted as a matrix format. However, the converting method for CPNs is more complex than the converting method for PNs.
**Definition 4.6.** Suppose $M$ is a CPN with a set of places $P$ and a set of transitions $T$; $\Sigma$ is the set of colour classes of $M$. $F$ is the colour domain mapping of $M$. $Pre \in \mathbb{N}^{P \times T}$ is defined as the backward incidence matrix of $M$, where $Pre[p,t]$ denotes the mapping from $p \in P$ to $t \in T$.

**Definition 4.7.** Suppose $M$ is a CPN with a set of places $P$ and a set of transitions $T$; $\Sigma$ is the set of colour classes of $M$. $F$ is the colour domain mapping of $M$. $Post \in \mathbb{N}^{P \times T}$ is defined as the forward incidence matrix of $M$, where $Post[p,t]$ denotes the mapping from $t \in T$ to $p \in P$.

**Definition 4.8.** For a CPN $M$ with a backward incidence matrix $Pre$ and a forward incidence $Post$, where $Pre, Post \in \mathbb{N}^{P \times T}$, $C = Post \ominus Pre$ is defined as the incidence matrix of $M$, where $\ominus$ is the difference operator of two unions of colour sets.

For example, Figure 4.3 and Table 4.1 can be used to show the graphical and formal representations of a CPN $M$, respectively. According to Definition 4.6 and 4.7, we can have $Pre$ and $Post$ matrices of $M$, which are shown in Table 4.2.

![Figure 4.3: A Coloured Petri Net M](image-url)
4.2. Analysis Techniques of PNs and CPNs

\[ CPN = (\Sigma, P, T, A, N, C, G, E, I) \]

\[ P = \{ p_1, p_2, p_3, p_4, p_5, p_6 \} \]

\[ T = \{ t_1, t_2, t_3 \} \]

\[ \Sigma = \{ Col1, Col2 \} \]

\[ G = \{ gf_1, gf_2, gf_3 \} \]

\[ E = \{ af_{11}, af_{12}, af_{14}, af_{22}, af_{23}, af_{35}, af_{43}, af_{52}, af_{63} \} \]

Table 4.1: Formal Description for CPN M

<table>
<thead>
<tr>
<th>Backward Incidence Matrix Pre</th>
<th>Forward Incidence Matrix Post</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_1 : gf_1 )</td>
<td>( t_2 : gf_2 )</td>
</tr>
<tr>
<td>( p_1 ): Col1</td>
<td>( af_{11} )</td>
</tr>
<tr>
<td>( p_2 ): Col1</td>
<td>( af_{22} )</td>
</tr>
<tr>
<td>( p_3 ): Col1</td>
<td>( af_{43} )</td>
</tr>
<tr>
<td>( p_4 ): Col1</td>
<td>( af_{52} )</td>
</tr>
<tr>
<td>( p_5 ): Col2</td>
<td>( af_{63} )</td>
</tr>
</tbody>
</table>

Table 4.2: Forward and Backward Incidence Matrices of CPN M

We can also analyse whether a transition is enabled in a CPN. However, differed with analysis methods of PNs, elements related with the analysis are not sets of integers but contain coloured tokens and arc functions. In other words, to enable a transition, the current marking of the CPN must satisfy the conditions that are conducted by the arc and transition.

**Definition 4.9.** Suppose that \( M \) is a CPN with \( T, P, Pre, Post \) and \( C \). \( \mu \) is the current marking of \( M \). \( t_j \in T \) is a transition of \( M \). Vector \( e[j] \) contains tokens that satisfy the minimum requirements which are defined on the guard function of \( t_j \). \( t_j \) is enabled if

\[ \mu \succeq Pre \cdot e[j] \]  \hspace{1cm} (4.6)
where operator \( \succeq \) denotes satisfying of conditions (on the guard function).

If Equation 4.6 can be satisfied, \( t_j \) will be enabled. The result marking \( \mu' \) of firing \( t_j \) (i.e. the marking after \( t_j \) fires) can be calculated by using Equation 4.7, where \( \ominus \) is the difference operator of two colour set unions, \( \oplus \) is the combination operator of two colour set unions, and \( \bullet \) denotes the colour transformation of a token by using an arc function in Pre or Post.

\[
\mu' = \mu \oplus (Post \cdot e[j]) \ominus (Pre \cdot e[j])
\] (4.7)

### 4.3 A CPN Based Approach for Flexible Interaction Formation

As introduced in Chapter 3, an interaction protocol can be represented as a CPN model. Hence, through analysing the CPN model of a protocol (by using the CPN analysis techniques introduced in Section 4.2), an agent can evaluate whether the protocol is suitable for itself. In this section, we propose a CPN based approach that allows agents to evaluate interactions before they form interaction relationships, and form interactions based on their preferences dynamically. Therefore, agents do not need to stick on fixed and predefined interaction protocols but can dynamically generate interaction protocols according to their demands. This improvement is very suitable for the open environments and the features of self-interested agents.

In Chapter 3, we have proposed a CPN based framework that separates agent interaction protocols and agents in different layers (recall to Section 3.3). In the approach introduced in this section, we adapt a similar framework. Differently, this approach includes a Net Constructor in the media layer. The Net Constructor is for constructing interaction protocols, which are sent by agents (see Figure 4.4). In the rest of this section, we introduce the structure of the CPN based approach.
and the processes of agent operating interactions.

Figure 4.4: The Framework for Flexible Interactions

4.3.1 Inclusion of Two Kinds of Interactions in the Approach

In this CPN based approach, we define two kinds of interactions for agents in a MAS, which are the default interaction and the proposed interaction. The default interaction is operated in a simple and fixed protocol, i.e. the default protocol. This protocol is conducted by the protocol layer as a default. Every agent in
a MAS can use the default protocol to achieve (default) interactions. On the other hand, the protocol for a proposed interaction, i.e. the proposed protocol, is not predefined in the protocol layer. A proposed protocol is generated via a negotiation-like process, which is operated by agents. A proposed protocol will be generated in the protocol layer only if all agents involved in the interaction agree on having interactions by using that protocol.

Generally, agents form a proposed interaction in the following procedures.

1. Agent $A$ proposes an interaction proposal to describe its preferred interaction protocol and sends the proposal to agent $B$;

2. Agent $B$ evaluates the received interaction proposal from $A$ and makes a decision on whether to accept it;

3. If agent $B$ cannot accept the received proposal, it modifies the proposal according to its demands, then sends the modified proposal back to agent $A$. Agent $A$ and agent $B$ negotiate on the interaction protocol through exchanging proposals;

4. If both of agent $A$ and agent $B$ agree on a proposed interaction protocol, the protocol will be sent to the *Net Constructor*;

5. The *Net Constructor* generates the CPN model for the proposed interaction proposal, and the *Port Manager* allocates communication ports to the interaction; and

6. After receiving the interaction ports from the *Port Manager*, agent $A$ and $B$ begin their interaction by using the protocol which is previously agreed by them.

In this approach, agents use the default interaction protocol to send and receive interaction proposals which conduct proposed protocols. A proposed protocol actually expresses the intended interaction of an agent. Through a negotiation
process, agents that are involved in an interaction can exchange their preferred interaction protocols until these agents meet on an agreed protocol. Then, agents use their agreed protocol to achieve interactions.

### 4.3.2 Call and Receive Interaction Proposals in Default Interactions

Default interaction allows agents to call and receive proposed interaction proposals. The protocol for default interactions is described in the CPN model of Figure 4.5. The components and colour set of this CPN model is described as follows.

- The Interaction Proposal (IntPro) colour set: The IntPro colour set is the data-type which is used to define the format of interaction proposals. The structure of IntPro colour set is shown in Figure 4.3.2. This colour set is composed by six parts:

  1. Composer: A string that holds the ID of the agent that composes
the proposal;

(2) **Receiver**: A string that holds the ID(s) of the agent(s) which the proposal is sending to;

(3) **ProSpec**: A text file that contains the formal description for the CPN model of a proposed Protocol;

(4) **InitialMarking**: A string that holds the initial marking of the CPN model;

(5) **Status**: A string that indicates the status of the proposal. *Status* has three possible values: (1) “Accepted”, (2) “Rejected” and (3) “Modified”. “Accepted” indicates that the proposed protocol has been accepted by the receiver (agent). “Rejected” indicates that the Proposed Protocol has been rejected by the receiver (agent). “Modified” indicates that an agent modifies the proposed protocol and asks for the approval from the interaction partner(s) (agent(s));

(6) **Comments**: A string that holds the comments from an agent.

colour set \( \text{IntPro} = \text{Composer} : \text{STRING} \times \text{Receiver} : \text{STRING} \times \text{ProSpec} : \text{File} \times \text{InitialMarking} : \text{STRING} \times \text{Status} : \text{STRING} \times \text{Comments} : \text{STRING} \);

<table>
<thead>
<tr>
<th>Table 4.3: The <em>IntPro</em> Colour Set</th>
</tr>
</thead>
</table>

- **The Proposal Token**: *Proposal Tokens* belong to the IntPro colour set. A *Proposal Token* contains a specification of a proposed protocol, which expresses the desired interaction of an agent;

- **The Call Place**: A *Call Place* is the place that conducts *Proposal Tokens* which are composed by an agent. An agent put a *Proposal Token* into its
4.3. A CPN Based Approach for Flexible Interaction Formation

Figure 4.6: An Example of a Protocol Proposal

*Call Place* when the agent needs to send an interaction proposal to other agents.

- *The Receive Place*: A *Receive Place* is the place that conducts incoming *Proposal Tokens*.

- *The Send Proposal Transition*: A *Send Proposal Transition* is the transition that sends *Proposal Tokens* of an agent to a corresponding agent. The *Send Proposal Transition* can be enabled when there is/are *Proposal Token(s)* in the *Call Place*. A *Proposal Token* will be moved from the *Call Place* to the *Receive Place* of another agent after the *Send Proposal Transition* is fired.

An agent can indicate its intentions/requirements in a proposed protocol, and inform other agents of its intentions/requirements by sending them the protocol. In a proposed protocol, an agent can specify the datum/resources that it can provide, and its required datum/resources. For example, suppose that the proposed protocol in Figure 4.6 is proposed by *Agent1*, and *Agent2* is the receiver of the protocol. The formal specification of this protocol is shown in Figure 4.7. In this protocol, place *P1* is connected with *Agent1* via an input port; place *P2* and *P3*
"An Example Protocol Proposal"

(protocol
  (name "ExampleProposal");
  (agents "Agent1", "Agent2");

  (place (name "P1") (colset "TYPE1");
  (place (name "P2") (colset "TYPE2");
  (place (name "P3") (colset "TYPE1");
  (place (name "P4") (colset "TYPE2");

  (transition (name "T1")
    (guard "fun checkName(ty1:TYPE1, ty2:TYPE2) =
                 (#producer ty1)="Agent1") andalso
                 (#producer ty2)="Agent2");")

  (transition (name "T2") (guard "0");

  (arc (name "a1") (from "P1") (to "T1") (arcfun "ty1");
  (arc (name "a2") (from "P2") (to "T1") (arcfun "ty2");
  (arc (name "a3") (from "T1") (to "P3") (arcfun "ty1");
  (arc (name "a4") (from "T1") (to "P4") (arcfun "ty2");
  (arc (name "a5") (from "P4") (to "T2") (arcfun "ty2");

  (port (name "por1") (type "In") (adhere_with "P1")
       (link_with "Agent1");
  (port (name "por2") (type "In") (adhere_with "P1")
       (link_with "Agent2");
  (port (name "por3") (type "Out") (adhere_with "P3")
       (link_with "Agent2");

)

Figure 4.7: The Specification of a Proposed Protocol
are connected with Agent2 with an input port and an output port, respectively. In other words, in the interaction, Agent1 will send tokens to P1, Agent2 will send tokens to P2 and get tokens from P3. This indicates that Agent1 wants to obtain datum/resources of TYPE2 (i.e. the type of P2 and P4) from Agent2 and provides Agent2 TYPE1 (i.e. the type of P1 and P3) datum/resources as the requital.

Before agents form a proposed interaction, they evaluate and “negotiate” about the interaction protocol. This process is to make the proposed interaction suitable for realising individual goals of agents. In the following subsection, we introduce how agents analyse proposed protocols.

4.3.3 Protocol Analysis

After receiving a proposal, an agent will analyse the protocol from two aspects:

1. Whether the agent can achieve the interaction with its current status;
2. Whether the proposed interaction is in conflict with the goal(s) of the agent.

A protocol analysis is achieved by evaluating the CPN model of the protocol. Generally, it is processed in two steps: (1) Place Type Analysis, and (2) Gain Analysis.

Place Type Analysis

The first step of Protocol Analysis (i.e. Place Type Analysis) is to test whether the agent can understand the CPN model of a proposed protocol that it received. To judge whether a CPN model can be understood by an agent, we give the following related definitions.

**Definition 4.10.** Place Type Set (PTS): For a proposed protocol, the PTS is a set of place types that occur in the CPN model of the protocol. For example, the PTS of the protocol in Figure 4.6 is \{TYPE1, TYPE2\}. 
Definition 4.11. Understandable Type Set (UTS): The UTS of an agent is a set of data types that exist in the knowledge base of the agent.

Definition 4.12. Non-understandable Type Set (NTS): If an agent receives a CPN model of a proposed protocol, the NTS of the agent for the CPN model is defined as:

\[ NTS = PTS - (PTS \cap UTS) \] (4.8)

where \( PTS \) is the Place Type Set of the CPN model and \( UTS \) is the Understandable Type Set of the agent.

Definition 4.13. An agent can understand a proposed protocol when \( PTS \subseteq UTS \) or \( NTS = \emptyset \), where \( PTS \) is the Place Type Set of the proposed protocol, \( UTS \) is the Understandable Type Set of the agent and \( NTS \) is the Non-understandable Type Set of the agent.

After an agent receives a Proposal Token, it will process the Place Type Analysis for the proposed protocol which is conducted by the token. Firstly, the agent generates the \( PTS \) of the proposed protocol. Then, the agent can find out its \( NTS \) for the protocol by using Equation 4.8. If the agent can understand the proposed protocol, i.e. \( NTS = \emptyset \), the protocol analysis will go to the next step. Otherwise, the agent modifies the Status value in the token to “Reject” and indicates the non-understood place type(s) in Comments, then sends the token back. This means that the agent rejects the proposal because it cannot fully understand the proposed protocol.

In Figure 4.8, we give a simple example of Place Type Analysis. In this example, Agent3 receives two Proposal Tokens \( PT1 \) and \( PT2 \) from Agent1 and Agent2, respectively. The \( UTS \) of Agent3 is shown in its \( UTS \) Table (see Figure 4.8), i.e. \( UTS = \{ DataType1, DataType2, DataType3, DataType4 \} \).

From the CPN model conducted by \( PT1 \), we can find out its \( PTS \), i.e. \( PTS_{PT1} = \{ DataType2, DataType5 \} \). According to Definition 4.12, we can calculate the
4.3. A CPN Based Approach for Flexible Interaction Formation

\[ NTS_{PT1} = PTS_{PT1} - (PTS_{PT1} \cap UTS) \]
\[ = \{\text{DataType1, DataType2}\} - \{\text{DataType1, DataType2}\} \]
\[ = \emptyset \]

This result indicates that Agent3 can understand the proposed protocol which is proposed by Agent1. Hence, the place type analysis for PT1 is accomplished, and Agent3 will process other analysis to PT1.

By using the same method, we can calculate the NTS of the CPN model
which is conducted by $PT2$ as follows:

\[
NTS_{PT2} = PT_{PS_{PT2}} - (PT_{S_{PT2}} \cap UTS)
= \{\text{DataType1, DataType5}\} - \{\text{DataType1}\}
= \{\text{DataType5}\}
\]

This means that $\text{DataType5}$ is not in the $UTS$ of Agent3, i.e. Agent3 cannot fully understand the proposed protocol of Agent2. In this situation, Agent3 will modify the received Proposal Token, and send the token back to Agent2 by firing its Send Proposal Transition (see Figure 4.9). In the modified token, Agent3 will change the value of Status to “Reject”, and change the value of Comments to “$\text{DataType5}$ cannot be understood”. In this way, Agent3 rejects the proposed protocol of Agent2 and informs Agent2 that the reason for the rejection is that $\text{DataType5}$ is not understandable.

![Figure 4.9: To Reject a Proposed Protocol in Place Type Analysis](image-url)
4.3. A CPN Based Approach for Flexible Interaction Formation

Gain Analysis

If the proposed protocol which is conducted by a Proposal Token can be understood by an agent, the agent will process the second step of protocol analysis, i.e. Gain Analysis. In a Gain Analysis, an agent will use the CPN analysis methods which were introduced in Section 4.2 to analyse and simulate the CPN model of a received protocol. From this simulation, the agent will find out what it will gain/lose from the proposed interaction.

Before introducing the Gain Analysis, we give some related definitions first.

Definition 4.14. For a CPN model of a proposed protocol, the Gain Set ($GSet$) is defined as the set of tokens that an agent can gain from the firing of transitions. It can indicate the datum/resources that an agent can obtain from the proposed interaction.

Definition 4.15. For a CPN model of a proposed protocol, the Cost Set ($CSet$) is defined as the set of tokens that an agent needs to provide to enable transitions of the CPN model. It indicates the datum/resources that an agent needs to provide in the proposed interaction.

Normally, each colour set in the CPN model of a proposed protocol corresponds to a particular data/resource type that will occur in the interaction. Therefore, each element in the $GSet/CSet$ represents a unit of data/resource that the agent will obtain or provide in the proposed interaction. From the $GSet$ and $CSet$, an agent can know whether it can obtain benefits from the proposed interaction, and whether the proposed interaction conflicts with its goal.

In a Gain Analysis, an agent needs to find out the $GSet$ and the $CSet$ of a proposed protocol, by calculating the required marking and the result-marking of firing transitions. These calculations can be achieved by using the analysis algorithms and methods which are introduced in Section 4.2. In addition, in the specification of a proposed protocol, the port type also indicates the token provider/receiver of a place (refer to the example shown in Figure 4.6 and 4.7).
For example, suppose that Agent2 receives a proposed protocol PP from Agent1, where the CPN model and specification of the protocol are shown in Figure 4.6 and Figure 4.7, respectively. The Gain Analysis of PP can be operated as the following processes.

1. According to Definition 4.6 and Definition 4.7, Agent2 can calculate the Backward Incidence Matrix Pre and Forward Incidence Matrix Post of PP, which are shown in Table 4.4;

<table>
<thead>
<tr>
<th></th>
<th>Pre</th>
<th></th>
<th>Post</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T1</td>
<td>T2</td>
<td>T1</td>
</tr>
<tr>
<td>P1: TYPE1</td>
<td>ty1</td>
<td></td>
<td>P1: TYPE1</td>
</tr>
<tr>
<td>P2: TYPE2</td>
<td>ty2</td>
<td></td>
<td>P2: TYPE1</td>
</tr>
<tr>
<td>P3: TYPE1</td>
<td></td>
<td></td>
<td>P3: TYPE1 ty1</td>
</tr>
<tr>
<td>P4: TYPE2</td>
<td>ty2</td>
<td></td>
<td>P1: TYPE1 ty2</td>
</tr>
</tbody>
</table>

Table 4.4: Forward and Backward Incidence Matrices of PP

2. According to Equation 4.6, Agent2 can find that it needs a marking $\mu = (1' ty1 : TYPE1, 1' ty2 : TYPE2, \emptyset, \emptyset)$ to enable transition $T1$, where $ty1/ty2$ is a variable in type TYPE1/TYPE2;

3. From the port type indicated in the specification (Figure 4.7), it can be seen that Agent1 is the token provider of $P1$. That means Agent2 only needs to provide $\{1' ty2 : TYPE2\}$ to enable $T1$. Hence Agent2 adds $\{1' ty2 : TYPE2\}$ into its CSet, i.e. $CSet = \emptyset \cup \{1' ty2 : TYPE2\} = \{1' ty2 : TYPE2\}$;

4. By using Equation 4.7, Agent2 can calculate the result-marking of firing
4.3. A CPN Based Approach for Flexible Interaction Formation

\[ T1: \]
\[
\mu'_{T1} = \mu \oplus (Post \cdot e[j]) \ominus (Pre \cdot e[j])
\]
\[
= (1'\text{ty}1 : TYPE1, 1'\text{ty}2 : TYPE2, \emptyset, \emptyset) \oplus (\emptyset, \emptyset, 1'\text{ty}1 : TYPE1, 1'\text{ty}2 : TYPE2)
\]
\[
= (\emptyset, \emptyset, 1'\text{ty}1 : TYPE1, 1'\text{ty}2 : TYPE2)
\]

By using the same method, \textit{Agent2} can find out that the result-marking of firing \( T2 \) is \( \mu'_{T2} = (\emptyset, \emptyset, 1'\text{ty}1 : TYPE1, \emptyset); \)

5. \( PP \) only includes two transitions, which are \( T1 \) and \( T2 \). Hence \( \mu'_{T2} \) is the final marking after the interaction. \( \mu'_{T2} \) indicates that after an interaction a token, i.e. \( \{1'\text{ty}1 : TYPE1\} \), will be left in \( P3 \), and the token receiver of \( P3 \) is \textit{Agent2} (specified in the “port type” in Figure 4.7). Therefore, \textit{Agent2} can obtain \( \{1'\text{ty}1 : TYPE1\} \) from the proposed interaction, and \( GSet = \emptyset \cup \{1'\text{ty}1 : TYPE1\} = \{1'\text{ty}1 : TYPE1\}; \)

6. \textit{Agent2} evaluates the data/resource that it will obtain or provide according to the value of \( GSet \) and \( CSet \); then makes a decision on whether it can accept the proposed interaction.

After the \textit{Gain Analysis}, if an agent decides to accept a proposed interaction, it will change the Status value of the \textit{Proposal Token} to “Accept” and send the token back to the interaction partner(s) (agent(s)). Otherwise, it will modify the proposed protocol according to its expectation or simply reject the proposal. The decision of an agent mainly depends on whether the proposed interaction conflicts with its goal, and how many benefits the agent can obtain from the interaction.

4.3.4 Advantages of the Protocol Analysis

In previous subsections, we introduced a CPN based approach that allows agents to analyse interaction protocols before they form interactions. The Protocol
4.4 Potential Application

Analysis processes in this approach enhance the robustness and reasonability of agent interactions especially in the following aspects:

1. By including the Place Type Analysis process before agents form interactions, an agent can evaluate whether an incoming interaction protocol is understandable and executable, so that interaction risks can be reduced;

2. Through including the Gain Analysis process, agents can evaluate whether a potential interaction conflicts with its goal(s). This feature is especially matched with the selfish features of self-interested agents;

3. An agent can modify the analysis methods according to the status of its working environment. This allows agents to make more reasonable decisions and be more adaptable to open and dynamic environments.

4.4 Potential Application

In the previous sections, we introduced a CPN based approach to enable agents to have flexible interactions. A potential application of the approach is in supply-chain formation [179]. A supply-chain is a network of interrelated exchange relationships among multiple levels of production [31]. Supply-chain formation is the process of assembling complex production and exchange relationships between different organisations (i.e. companies, factories, etc). Many researchers use agent technologies in automotive supply-chain formations. In such applications, each agent is used to manage the finite resources of a particular organisation. Contracting and cooperations between various organisations are realised through agent interactions. Agents in a supply-chain system are self-interested. The goal of an agent is to maximise the profit of its represented organisation, and avoid penalties caused by un-reasonable contracts. Therefore, an agent has to analyse received interaction requests and give proper responses according to current resource availability.
The CPN based approach can help agents to analyse and dispose various interaction requests. For a supply-chain formation application, various kinds of resources and products can be defined as colour sets, and a CPN model can be used to describe supply-request (SR) relations between different organisations.

For example, the CPN model of Figure 4.10 shows the SR relation between Firm-1 and Firm-2. The CPN components of this model are described as follows:

- Places $P1$ and $P2$ represent two kinds of materials/resources of Firm-1 (light grey places in Figure 4.10);
- Places $P3$, $P4$ and $P5$ represent three kinds of materials/resources of Firm-2 (white grey places in Figure 4.10);
- Places $P5$ and $P7$ represent two kinds of products (dark grey places in Figure 4.10);
- Transitions $T1$ and $T2$ represent production of a production by using required materials.

By using the algorithms and methods introduced in previous sections, Firm-1 and Firm-2 can analyse the CPN model, and check whether they can accomplish the producing tasks which are described in this model. They can also modify the CPN model according to their statuses.

4.5 Summary

In this chapter, we presented a CPN based approach that utilises mathematical representations of CPNs to analyse and predict agent interactions. This approach includes two kinds of interactions among agents of a MAS. Agents first achieve basic interactions by using default interaction protocol. Through a default interactions, agents construct the future (complex) interactions by “negotiating” interaction proposals. Each interaction proposal conducts the CPN model of
a proposed protocol. Through analysing the CPN model, agents can confirm whether they can achieve the proposed interaction, and what it can gain/lose from the interaction. In this way, agents are able to form more reasonable interactions. The CPN based approach is very suitable for coordinating interactions among self-interested agents.

In this chapter, we also explored a potential application domain of the approach, which is the supply-chain formation. In such applications, we can use CPN colour sets to represent various products and resources of a supply-chain system, and use CPN models to express SR relations among different organisations. Then, through analysing the CPN model of a SR relation, an agent can find out whether its current status can allow it to form the SR relation.
Flexible Team Formation in Self-Interested Multi-Agent Systems

As social entities, intelligent agents need to collaborate with others regardless of whether they are cooperative or self-interested. Due to the distributed nature of the problem to be solved and because of the limitations of agents’ abilities, agents in many cases need to form teams to work on some tasks. In the current MAS research, MAS team formation is facing a number of challenges, especially with regard to the following two aspects:

- Many current multi-agent systems (MASs) are required to work in open and dynamic environments [7] [60] [174] [173]. Uncertainties of dynamic environments obstruct coherent teamwork and bring difficulties for agent team formation. In dynamic environments, system constraints, resource availabilities, agent goals, etc. are all changeable. The changing of these factors may directly require a MAS to deal with different situations. In a new situation, keeping an outdated team organisation may obstruct agents to achieve their individual goals. Therefore, “there is no single type of organisation that is suitable for all situations [71].”

- Compared with cooperative agents, collaborations among self-interested agents are more complicated and dynamic due to the selfish features. Self-interested agents are impelled to cooperate with others by their individual
goals. In an agent team composed of self-interested agents, temporary coop-
erations among agents might be conflict to selfish goals of individual agents
as the environment changes. In open and dynamic environments, if factors
such as agent goals, task requirements and resources are changed, a selfish
agent may need to modify or even relieve the collaboration relationships
with its “colleagues”. Otherwise the collaboration would be in conflict or
even harmful to the individual goal of the agent. Considering this point,
some researchers suggest to hire dynamic agent team formation strategies in
this kind of applications. However, how long collaboration should be kept
among particular agents is always a problem to be discussed. Focusing on
general self-interested multi-agent systems, this chapter proposes a flexible
team formation mechanism that can enable agents to select team members
with reasonable terms and objects. The flexibility of the mechanism can
allow agents to form more reasonable teams that avoid some potential profit
conflicts among self-interested team members.

In many MAS applications, a dynamic team formation mechanism is needed
to enable agents to automatically form and reform groups/teams to avoid profit
conflicts between agents according to the changing of the environment. Toward
this objective, a number of researchers try to find an optimal mechanism for dy-
namic team formation and member selection. Generally, in current MAS research,
there are two kinds of team formation mechanisms which are widely applied in
self-interested MASs, i.e. one-shot team formation and long-term team formation.
The concepts of these two kinds of team formation mechanisms are described as
follows.

- **One-shot team formation mechanism**

  In self-interested MASs, individual agents’ willingness and goals are impor-
tant factors that need to be considered during team forming. The research
on team formation for self-interested agents generally focuses on forming
one-shot teams, also called short-term teams, for individual tasks. In this kind of mechanism, agents come together when they need to handle some tasks, and their relationships will be terminated after tasks have been accomplished.

- **Long-term team formation mechanism**

  Obviously, one-shot teaming can arouse frequent grouping and regrouping among agents. Each grouping/regrouping consumes some communication and computation resources. To overcome the weakness of one-shot teaming, Rathod and desJardins proposed several stable-team formation strategies for self-interested MASs [143]. These strategies allow self-interested agents to form long-term relationships to cut team formation consumptions. However, for many self-interested MASs, agent goals or willingness are changeable and remain uncertain. A long-term relationship is very hard to be kept after the goals of team member agents are changed.

Both of one-shot team formation and long-term team formation mechanisms have some weaknesses. One-shot team formation may bring high communication and computation consumptions to a MAS. However, long-term team formation is not suitable for the dynamic features of open environments and the selfish feature of self-interested agents.

To cover some shortcomings of one-shot and long-term team formation, in this thesis, a mechanism that enables self-interested agents to flexibly choose team durations and members is developed. Factors, such as agent historical performances, task requirements and resource constraints, are considered in the mechanism. Especially for open environments, the flexible team formation and member selection mechanism are more suitable for self-interested agents applications. This flexible team formation mechanism enables more dynamic and reasonable collaboration between agents and reduces unnecessary consumptions and benefit conflicts brought by team formation. Due to the high uncertainties
of most open environments, analysis and evaluations of dynamic factors are not very easy. Especially, it is impossible to find a fixed standard for agent evaluations, e.g. how good an agent performance is. Regarding this point, fuzzy rules are used in our flexible team formation mechanism to evaluate the performance and importance of agents. This will enable an agent to dynamically select collaboration durations and objectives based on the result of fuzzy evaluations, and to choose collaboration manners more flexibly.

This chapter is arranged as follows. In the Section 5.1, the MAS structures and some important definitions and assumptions in this research are introduced. Section 5.2 presents the advantages, disadvantages and suitable areas of long-term and one-shot teams. The flexible team formation mechanism is introduced in Section 5.3. In Section 5.4, experiments that compare the flexible mechanism with one-shot and long-term team formation are presented. Some related works of this research are presented and compared in Section 5.5. Finally, a summary of this chapter is given in Section 5.6.

5.1 System Architecture and Problem Definition

Various MAS applications may have different system structures. In this research, the MAS environment is set up to demonstrate and analyse the team formation and member selection mechanisms. Hence, the system structure is set up toward assisting agent communication and task allocation. Some simplifying assumptions and definitions, which can avoid adding to the scheduling and task decomposing problems, are also made, and only elementary agents and task models are included in the MAS. However, these models are generic enough to be practical and applicable to a wide range of real applications.
5.1. System Architecture and Problem Definition

5.1.1 The System Architecture

The MAS architecture of this research is shown in Figure 5.1. In this research, all agents of the system are assumed to be self-interested. Their goals are to achieve awards through accomplishing tasks sent by outside users. New tasks are published on the Task Board of the system, and will be removed from Task Board after being taken by an agent or agent team. Published tasks are accessible to all individual agents and agent teams of the system. The agent number of the system can be dynamic. Agents can enter and leave the system according to their willingness. However, agents have to publish and remove their registration information on the Agent Board of the system before they enter and leave the system. The registration information records the skills and status of an agent (see Definition 5.2).

Agent abilities are limited. To perform tasks beyond its ability, an agent needs to collaborate with other agents through joining or forming a team. Each agent team is composed of one (and only one) Team Leader (TL) and several Team Members (TMs). After an agent joins an agent team, it can get payments from the agent team. At the same time it needs to work for the agent team for a certain period. The payment and serving term are described in the contract (see Definition 5.3) between the Team Member (TM) and the TL.

5.1.2 Definitions and Assumptions

Before presenting the team formation mechanism, some important definitions and assumptions are given in this subsection.

**Definition 5.1.** A task is defined formally as $t_i = \langle w_i, R_i' \rangle$, where $w_i$ is the reward gained by an agent/agent-team if task $t_i$ is accomplished by that agent/agent team; $R_i$ is the set of resources or skills, which are possessed by agents of the system, required by task $t_i$. A task can only be assigned to one agent or agent team.
Definition 5.2. An agent is formally defined as \( a_i = (g_i, R_i, s_i) \), where \( g_i \) is a set of individual goals of agent \( a_i \); \( R_i \) is the skills and resources possessed by agent \( a_i \); \( s_i \) is the status of \( a_i \), where \( s_i = (v_a, v_p, t) \). \( s_i \) represents whether agent \( a_i \) is performing a task and participating in an agent team. Meanings of different \( s_i \) values are listed in Table 5.1. The names and meanings of \( v_a \), \( v_p \) and \( t \) are listed as following:

Availability \( v_a \): \( v_a \) presents whether an agent is performing a task. \( v_a = 0 \) when the agent has no task (available); \( v_a = 1 \) when the agent is performing a task (not available);

Position Parameter \( v_p \): \( v_p \) presents whether an agent is an individual agent, TL
5.1. System Architecture and Problem Definition

or TM. $v_p = 0$ when the agent is individual; $v_p = 1$ when the agent is a TM of an agent team; $v_p = 2$ when the agent is a TL of an agent team.

Contract Ending Time $t$: $t$ is the contract ending time of an agent (also see Definition 5.3).

<table>
<thead>
<tr>
<th>$s_i$ value</th>
<th>Status of agent $a_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0, 0)</td>
<td>Performing no task; has no agent team.</td>
</tr>
<tr>
<td>(1, 0, 0)</td>
<td>Performing a task; has no agent team.</td>
</tr>
<tr>
<td>(0, 1, 0)</td>
<td>Has a one-shot contract as a TM; performing no task currently.</td>
</tr>
<tr>
<td>(1, 1, 0)</td>
<td>Has a one-shot contract as a TM; performing a task currently.</td>
</tr>
<tr>
<td>(0, 1, t)</td>
<td>The TM of an agent team for period $t$; performing no task currently.</td>
</tr>
<tr>
<td>(1, 1, t)</td>
<td>The TM of an agent team for period $t$, performing a task currently.</td>
</tr>
<tr>
<td>(0, 2, 0)</td>
<td>The TL of an agent team; performing no task currently. (It is assumed that the TL cannot quit from its agent team and let $t$ value of a TL equal to 0.)</td>
</tr>
<tr>
<td>(1, 2, 0)</td>
<td>The TL of an agent team; performing a task currently.</td>
</tr>
</tbody>
</table>

Table 5.1: Status of An Agent

Definition 5.3. A contract $c_{ij}$ is an agreement between TL $a_i$ and TM $a_j$. It can be defined as $c_{ij} = \langle t_{ij}, p_{ij}, S_{ij} \rangle$, where $t_{ij}$ is the contract ending time; $p_{ij}$ is the penalty that the TL or TM has to pay (to the other side of the contract) if it breaks the contract and terminates the cooperation relationship before $t_{ij}$; $S_{ij}$ is a set of payment that $a_j$ can gain through serving the agent team. $S_{ij}$ can be described as tuple $\langle sc_{ij}, sd_{ij} \rangle$. For contracts between the TL and TM of a one-shot team, $t_{ij}$, $p_{ij}$, and $sd_{ij}$ equal to 0. $sc_{ij}$ is the payment that TM $a_j$ can gain for each task completed by the agent team, when $a_j$ directly participates in the task. $sd_{ij}$ is the dividend (or award) that TM $a_j$ can share for each task completed by the agent team, when $a_j$ does not actually participate in that task.

Definition 5.4. An agent team is a set of agents. It can be formally defined as $AT_i = \langle MS_i, TR_i \rangle$, where $MS_i$ is the set of agents that currently are TMs
of $AT_i$; $TR_i$ is the capacity of the whole agent team. Here, it is assumed that

$$TR_i = \sum_{j|a_j \in MS_i} (R_j + R_i),$$

where $R_i$ and $R_j$ are resources possessed by the team leader and team members, respectively. In other words, the capability of an agent team is the sum of its TMs’ capabilities and TL’s capability. It is also defined that $\forall i \neq j : MS_i \cap MS_j = \emptyset$, which means an agent can only participate in one agent team.

**Definition 5.5.** A *Contributor Set* $CS_{ij}(CS_{ij} \subset MS_i)$ of Agent Team $AT_i$ is the set of agents that participate in performing task $t_j$, where $t_j$ is a task of agent team $AT_i$. For a one-shot team, the CS equals to $MS_i$ of the team (also refer to Definition 5.4).

**Definition 5.6.** For agent team $AT_i$, a *Member Contribution* $mc_{ijk}$ is the contribution of agent $a_k$, where $a_k \in CS_{ij}$, in performing task $t_j$ ($t_i = \langle w, R'_i \rangle$). $mc_{ijk}$ equals to $w/N$, where $N$ is the size of CS and $w$ is the task reward.

## 5.2 One-Shot and Long-Term Team Formation

After presenting the system architecture and some important definitions, concepts and comparisons of the one-shot and long-term team formation mechanisms are presented in this section.

### 5.2.1 One-Shot Teams

One-shot team strategy is widely applied in many MAS applications. In this strategy, agents of the system do not have a team initially. When a task $t_i$ is published in the *Task Board*, agents start to bid on the new task. The system facilitator will choose (or randomly select) a bidder to assign the task. After the agent bids the task successfully, it becomes a TL and starts to look for collaborators according to the task requirement $R'_i$. Finally, the agent team will disband after the task ($t_i$) is accomplished.
5.2. One-Shot and Long-Term Team Formation

Generally, the one-shot team strategy includes the following processes. (Here, it is assumed that the agents of the MAS cannot achieve the task individually.)

1. The system facilitator of the MAS publishes a new task $t_i = \langle w_i, R'_i \rangle$ on the Task Board, where $w_i$ and $R'_i$ are the award and required resources of the task;

2. Agents, whose $g < w_i$ and $s = (0,0,0)$ bid on $t_i$;

3. The system facilitator awards $t_i$ to agent $a_j(a_j =< g_j, R_j, s_j >)$. At the same time, $a_j$ becomes the TL of agent team $AT_j$ and modifies its $s_j$ to $(0, 2, 0)$. At this movement, $TR_j = R_j$;

4. $a_j$ searches the Agent Board to look for agents with status $(0, 0, 0)$, which can provide the lacking resources $R$, where $R \subseteq (R'_i - R'_i \cap TR_j)$;

5. $a_j$ finds a required agent $a_p$, where $R_p \subseteq (R'_i - R'_i \cap TR_j)$;

6. $a_j$ sends a contract $c_{jp}$ to $a_p$, where $sc_{jp} \leq (w_i-g_j) \cdot sizeOf(R_p)/sizeOf(R'_i - R_i)$;

7. $a_p$ accepts $c_{jp}$ if $sc_{jp} \geq g_p$ or rejects $c_{jp}$ if $sc_{jp} \leq g_p$;

8. If $c_{jp}$ is accepted by $a_p$, $TR_j = TR_j \cup R_p$, and $a_p$ modifies its status to $(0, 1, 0)$;

9. Goes to Process (4) until $TR_j = R'_i$;

10. $AT_j$ starts to perform $t_i$; the TL and the TMs of $AT_j$ modifies/modify its/their statuses to $(1, 1, 0)$ and $(1, 2, 0)$, respectively;

11. $AT_j$ accomplishes $t_i$; agents of $AT_j$ modify their statuses to $(0,0,0)$ and are released from the team.
5.2. One-Shot and Long-Term Team Formation

5.2.2 Long-Term Teams

In a long-term team formation strategy, the agent team will not be dissolved after performing tasks. On the contrary, the TL of the team pays TMs some payments to keep the cooperative relationship, even if the TM does not contribute on the task accomplishment.

The long-term team strategy normally includes the following processes.

1. TL $a_i$ finds several free agents, whose status values are $(0, 0, 0)$, from the Agent Board and sends them contracts in order to form a team with them. Agents modify their statuses to $(0, 1, t_{ij})$ if they accept the contracts. In this case, agent team $AT_i$ is formed successfully;

2. TL $a_i$ searches the Task Board for a suitable task and bids on task $t_k(t_k = \langle w_k, R'_k \rangle)$, where $R'_k \subseteq TR_i$ and $w_k \geq \sum_{j|a_j \in MS_i}(S_{ij} + g_i)$ (also refer to Definition 5.1 to 5.4).

3. If $t_k$ is bided successfully, TL $a_i$ assigns $t_k$ to TM $a_p, a_q...a_n$, where $R_p \cup R_q, ..., \cup R_n$ is the minimum set that satisfies $R'_k \subseteq R_p \cup R_q, ..., \cup R_n$. At the same time, $a_p, a_q, ..., a_n$ modify their statuses to $(1, 1, t_{ip}), (1, 1, t_{iq}), ..., (1, 1, t_{in})$. Also, for this task performance, the Contributor Set $CS_{ik}$ (refer to Definition 5.5) should be $\{a_p, a_q, ..., a_n\}$;

4. $a_p, a_q, ..., a_n$ modify their statuses to $(0, 1, t_{ip}), (0, 1, t_{iq}), ..., (0, 1, t_{in})$ after $t_k$ is accomplished;

5. TL $a_i$ awards TM $a_m \ (a_m \in AT_i)$ with $(sc_{im} + sd_{im})$ if $a_m \in CS_{ik}$, or $sd_{im}$ if $a_m$ is not in $CS_{ik}$;

In addition, if the TL $a_i$ or TM $a_p$ wants to terminate the contract before the contract ending time $t_{ip}$, they may process the following two steps.

1. $a_i/a_p$ terminates $c_{ip}$ with $a_p/a_i$, and pays $p_{ip}$ to $a_p/a_i$;

2. $a_p$ is released from $AT_i$, and its status is modified to $(0, 0, 0)$.
5.2.3 Advantages and Disadvantages of Long-term and One-shot Team Formation

One-shot teams are suitable for dynamic MAS application domains. It always keeps loosely coupled relationships among agents as default. However, many dynamic applications are not that dynamic. For example, the tasks may have some similarities, and their requirements might be similar (which means they may just need similar agent teams). In this case, frequent grouping and regrouping are not very necessary, since each time grouping consumes some system resources. As opposed to one-shot teams, long-term teams can greatly reduce the system consumption caused by grouping and regrouping. However, most current long-term team formation strategies cannot figure out when agents should form long-term teams, which agents should be included in, and how long the relationships should be kept. For self-interested MAS applications, keeping unnecessary long-term cooperative relationships could be very dangerous and harmful for the overall performance of the system. Features of one-shot teams and long-term team are summarised and compared in Table 5.2.

<table>
<thead>
<tr>
<th></th>
<th>One-Shot Teams</th>
<th>Long-Term Teams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication Consumption</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Suitable Domains</td>
<td>Suitable for highly dynamic environments</td>
<td>Suitable for stable environments</td>
</tr>
<tr>
<td>Suitable MASs</td>
<td>Suitable for self-interested MAS</td>
<td>Suitable for cooperative MAS</td>
</tr>
<tr>
<td>Relationships among Team Members</td>
<td>Loosely coupled</td>
<td>Tightly coupled</td>
</tr>
</tbody>
</table>

Table 5.2: Features of One-Shot Teams and Long-Term Teams
5.3 Flexible Team Formation Mechanism

From the description of short-term and long-term team formation in the previous section, it can be seen that both long-term and one-shot teams have some advantages and disadvantages. One-shot teams are suitable for dynamic tasks, i.e. requirements of various new tasks which are totally different. By contrast, long-term teams possess advantages when tasks are “stable” or similar. For most self-interested agents, the duration of teams should not be fixed on a certain term. Taking human society as an example, a company may sign different contracts (with different durations and conditions) with different employees. According to the performance of employees and changes of the job market, the company will want to make changes in these contracts in the future. For MASs, it is also necessary to have a flexible team formation mechanism, which can enable team leaders to choose different collaboration durations with agents according to the changing trend of task-requirements and the performance of agents. In this section, a flexible team formation mechanism is introduced. In the mechanism, value and availability of agents are evaluated. Then, team leaders will determine the required members and choose proper collaboration durations and cost according to the evaluation results.

5.3.1 Team Member Performance Evaluations

In general, agents that always contribute to performing tasks and can bring more benefits to the team are the most valuable members of an agent team. These agents should be kept on the team for a long term. In contrary, the agent team should not include agents that bring little contribution to the team. In this mechanism, two factors, which are Utilisation Ratio \((ur)\) and Contribution Ratio \((cr)\), are used to evaluate the value of a TM.

**Definition 5.7.** *Utilisation Ratio* \(ur_{Mk}\) (\(ur_{Mk} \in [0, 1]\)) is the frequency that a TM \(a_k\) has participated in the most recent \(M\) tasks of the agent team \(AT_i\). It
can be calculated by Formula 5.1. The value of parameter $M$ is chosen by TLs or assigned by users. TLs can also adjust $M$ values according to environment situations and team performance.

$$ur_{Mk} = \frac{1}{M} \sum_{j=1}^{M} \left( k | a_k \in CS_{ij} \right)$$  \hspace{1cm} (5.1)

**Definition 5.8.** *Contribution Ratio* $cr_{Mk}$ ($cr_{Mk} \in [0, 1]$) is the ratio that TM $a_k$ has contributed to the agent team $AT_i$ in the most recent $M$ tasks. It can be calculated by using Formula 5.2 (also refer to Definition 5.6).

$$cr_{Mk} = \frac{\sum_{j=1}^{M} mc_{ijk} \left( k | a_k \in CS_{ij} \right)}{\sum_{j=1}^{M} w_j}$$  \hspace{1cm} (5.2)

The following example shows how to evaluate TMs through $ur$ and $cr$. Suppose $t_1 = <40, R'_1 >$, $t_2 = <50, R'_2 >$ and $t_3 = <60, R'_3 >$ are the most recent three tasks accomplished by agent team $AT_i$. $a_p, a_q, a_r$ and $a_s$ are TMs of $AT_i$. TMs that participate in the three tasks are \{a_p, a_q\}, \{a_p, a_r\} and \{a_p, a_q\}, respectively. According to Equation 5.1 and 5.2, it can be found that the $ur$ and $cr$ values of $a_p, a_q, a_r$ and $a_s$ are:

- $a_p$: $ur_{3p} = 1$, $cr_{3p} = \frac{(40/2+50/2+60/3)}{(40+50+60)} = 0.5$
- $a_q$: $ur_{3q} = 0.67$, $cr_{3q} = \frac{(40/2+60/3)}{(40+50+60)} = 0.33$
- $a_r$: $ur_{3r} = 0.33$, $cr_{3r} = \frac{50/2}{(40+50+60)} = 0.17$
- $a_s$: $ur_{3s} = 0$, $cr_{3p} = 0$

Comparing $ur$ and $cr$ values of the four TMs of $AT_i$, it can be seen that $a_p$ is the most important member of $AT_i$ since it frequently participated in recent tasks and contributed the most benefit to the team. On the other hand, $a_s$ did not participate in recent tasks and has no contribution to $AT_i$. 
5.3. Flexible Team Formation Mechanism

5.3.2 System Agent Resource Evaluations

With \( wr \) and \( cr \), the TL can evaluate the contribution of a TM. However, to make reasonable contracts with a TM, the TL also needs to evaluate whether it is easy to find similar agents (which possess similar resources and skills) in the MAS. In this mechanism, Agent Resource Availability (\( ara \)) is the parameter defined to evaluate agent resource availability in the MAS.

Definition 5.9. Agent Resource Availability \( ara_k \): \( ara_k \) is the ratio of available agents (which do not have a team/task) that possess the same or more resources than TM \( a_k \). It can be calculated by using Formula 5.3 (Note: in Formula 5.3, \( N_{av} \) is the available agent number of the MAS).

\[
ara_k = \frac{\sum_{s_i=(0,0,0)} R_{k \subseteq R_i}}{N_{av}}
\]  

For example, suppose that \( a_k \) is a TM of \( AT_i \). Currently, there are ten out of twenty available agents in the MAS, which possess same or more resources than \( a_k \). Hence, the \( ara \) value of TM \( a_k \) is: \( ara_k = 0.5 \).

5.3.3 Flexible Member Selection by Using Fuzzy Rules

According to the values of \( cr \), \( ur \) and \( ara \), in this mechanism, TLs use a fuzzy method to determine collaboration durations and cost with their TMs.

Input and Output Parameters

In the fuzzy method, \( ur \), \( cr \) and \( ara \) are input parameters. The output parameters are Contract Term \( ct \) and Commission Amount \( ca \). They are defined in Definition 5.10 and 5.11.

Definition 5.10. Contract Term \( ct_k \) is the parameter to denote the duration that agent \( a_k \) should be kept in the agent team. It is an output parameter that
needs to be identified through the fuzzy method. The working range of \textit{Contract Term} is in \([0, \text{MAXTERM}]\). \text{MAXTERM}, which is a constant that is defined in the MAS, and denotes the maximum term that an agent can be kept in an agent team.

\textbf{Definition 5.11.} \textit{Commission Amount} \(ca_k\) is the parameter that denotes the maximum commission that the agent team should pay to agent \(a_k\) in order to keep \(a_k\) in the team. It is an output parameter that needs to be identified through the fuzzy method. The working range of Commission Amount is in \([0, \text{MAXPAY}]\). \text{MAXPAY} is a parameter that is decided by TLs. \text{MAXPAY} denotes the maximum payment that an agent team can afford to keep a single agent as a TM.

\textbf{Membership Functions for Input Parameters}

For \(ur\), four linguistic states are selected and expressed by appropriate fuzzy sets. They are \textit{Never} (N), \textit{Seldom} (S), and \textit{Medium} (M) and \textit{Frequent} (F). Another input parameter \(cr\) also has four linguistic states, which are \textit{None} (N), \textit{Little} (L), \textit{Medium} (M) and \textit{Huge} (H). The trapezoidal \cite{22} fuzzy membership function is adopted here to define fuzzy memberships of these four fuzzy sets. The membership functions for these four fuzzy sets are defined from Formulae 5.4 to 5.7, respectively. They are also depicted in Figure 5.2.

\[
F_{\text{Never}}(x) = \begin{cases} 
1 - 5x & x \in [0, 0.2] \\
0 & x \not\in [0, 0.2]
\end{cases} \tag{5.4}
\]

\[
F_{\text{Seldom}}(x) = \begin{cases} 
10x - 1 & x \in [0.1, 0.2] \\
1 & x \in (0.2, 0.3) \\
4 - 10x & x \in [0.3, 0.4] \\
0 & x \not\in [0.1, 0.4]
\end{cases} \tag{5.5}
\]
5.3. Flexible Team Formation Mechanism

\[ F_{\text{Medium}}(x) = \begin{cases} 
10x - 3 & x \in [0.3, 0.4] \\
1 & x \in (0.4, 0.6) \\
7 - 10x & x \in [0.6, 0.7] \\
0 & x \not\in [0.3, 0.7] 
\end{cases} \]  \hspace{1cm} (5.6)

\[ F_{\text{Frequent}}(x)/F_{\text{Huge}}(x) = \begin{cases} 
10x - 6 & x \in [0.6, 0.7] \\
1 & x \in (0.7, 1] \\
0 & x \not\in [0.6, 1] 
\end{cases} \]  \hspace{1cm} (5.7)

Figure 5.2: Fuzzy Membership Function for ur/cr

For ara, three linguistic states are selected, which are Rare (R), Some (S), Many (M). The membership functions for ara are defined from Formulae 5.8 to 5.10 and depicted in Figure 5.3.

\[ F_{\text{Rare}}(x) = \begin{cases} 
1 - 4x & x \in [0, 0.4] \\
0 & x \not\in [0, 0.4] 
\end{cases} \]  \hspace{1cm} (5.8)
5.3. Flexible Team Formation Mechanism

\[ F_{\text{Some}}(x) = \begin{cases} 
5x - 1 & x \in [0.2, 0.4] \\
3 - 5x & x \in (0.4, 0.6] \\
0 & x \not\in [0.2, 0.6] 
\end{cases} \]  \hspace{1cm} (5.9)

\[ F_{\text{Many}}(x) = \begin{cases} 
5x - 2 & x \in [0.4, 0.6] \\
1 & x \in (0.6, 1] \\
0 & x \not\in [0.4, 1] 
\end{cases} \]  \hspace{1cm} (5.10)

Figure 5.3: Fuzzy Membership Function for \( ara \)

Membership Functions for Output Parameters

There are two output parameters, which are Contract Term \((ct)\) and Commission Level \((cl)\) in the fuzzy method. For \( ct \), four linguistic states are selected, which are Long \((L)\), Medium \((M)\), Short \((S)\) and No \((N)\). For \( cl \), High \((H)\), Medium \((M)\), Low \((L)\) and No \((N)\) are chosen as the four linguistic states. Fuzzy membership functions of these fuzzy sets are defined from Formulae 5.11 to 5.14 and described in Figure 5.4.
5.3. Flexible Team Formation Mechanism

\[ F_{Na}(x) = \begin{cases} 
1 - 10x & x \in [0, 0.1] \\
0 & x \notin [0, 0.1] 
\end{cases} \]  \hfill (5.11)

\[ F_{Short}(x)/F_{Low}(x) = \begin{cases} 
10x & x \in [0, 0.1] \\
1 & x \in (0.1, 0.3) \\
4 - 10x & x \in [0.3, 0.4] \\
0 & x \notin [0, 0.4] 
\end{cases} \]  \hfill (5.12)

\[ F_{Medium}(x) = \begin{cases} 
10x - 3 & x \in [0.3, 0.4] \\
1 & x \in (0.4, 0.6) \\
4 - 10x & x \in [0.6, 0.7] \\
0 & x \notin [0.3, 0.7] 
\end{cases} \]  \hfill (5.13)

\[ F_{Long}(x)/F_{High}(x) = \begin{cases} 
10x - 6 & x \in [0.6, 0.7] \\
1 & x \in (0.7, 1] \\
0 & x \notin [0.6, 1] 
\end{cases} \]  \hfill (5.14)

**Fuzzy Rule Base**

A fuzzy rule base is a matrix of combinations of each of the input linguistic parameters and their corresponding output parameters. The rule base in this mechanism is described in Table 5.3.

**Determination of Output Membership Values and Defuzzification**

Each entry of the rule base is a rule, which is defined by ANDing two linguistic input parameters to produce an output combination, in the form of: \( IF(F(ur) = \alpha \text{ AND } F(cr) = \beta \text{ AND } F(ar) = \gamma) \text{ THEN } (F(ct) = \delta \text{ AND } F(cl) = \eta) \), where \( \alpha \in \{Never, Seldom, Medium, Frequent\} \), \( \beta \in \{None, Little, Medium, \),
5.3. Flexible Team Formation Mechanism

Figure 5.4: Fuzzy Membership Function for $ct/cl$

$\{\text{Large}\}$, $\gamma \in \{\text{Rare, Some, Many}\}$, $\delta \in \{\text{Long, Medium, Short, No}\}$, and $\eta \in \{\text{High, Medium, Low, No}\}$. In this mechanism, AND/MIN operator is used to combine the membership values together, i.e. the weakest membership determines the degree of membership in the intersection of fuzzy sets [44]. Hence, the output membership value $\mu_{\delta/\eta}(v)$ can be calculated by Formula 5.15.

$$\mu_{\delta/\eta}(v) = \text{MIN}(\mu_{\alpha}(ur), \mu_{\beta}(cr), \mu_{\gamma}(ara))$$ (5.15)

With the output membership, the output values can be determined by tracing the membership values for each rule back through the output membership functions. Finally, centroid defuzzification method [44] is hired to find out the output value. In centroid defuzzification, the output value is calculated by Formula 5.16, where $\mu(v_i)$ is the $i^{th}$ output value, $v_i$ is its corresponding output value, and $k$ is the number of fuzzy rules which are activated.

$$DF = \frac{\sum_{i=1}^{k} (v_i \cdot \mu(v_i))}{\sum_{i=1}^{k} \mu(v_i)}$$ (5.16)
5.4 Experiments

To analyse the performance of the flexible team formation mechanism, some experiments are conducted to compare it with one-shot and long-term team formation mechanisms. The experiment environment is set up to simulate the scenario introduced in Section 5.1. Each agent possesses one or more than one kind of resources and it needs to contribute its resources to achieve awards through accomplishing tasks of the system. However, in most cases, an agent cannot accomplish a task due to its resource limitation. Hence, agents have to cooperate with others to realise their goals. A set of tasks are sent to agents, and agents achieved these tasks by using one-shot, long-term and flexible team formation mechanisms, respectively. Two factors are compared in the experiment, which are Agent Searching Time (AST) and Award Distribution Situation (ADS).

AST represents the time that a team leader needs to search for required agents.
from the agent board to accomplish the tasks. In general, the higher AST, the more communication cost the team leader needs to spend on searching agents. Through this experiment, it can be seen that flexible team formation has the least Agent Searching Times (ASTs). On the contrary, the AST of one-shot team formation is much higher than both long-term and flexible team formation (See Figure 5.5). This result shows that the communication consumption of using one-shot team formation is the highest. The reason for this result is that in one-shot team formation, agent teams are disbanded when each task is accomplished, and a new team needs to regroup for the new task. With long-term and flexible team formation, the whole team or part of team is kept after a task is finished. So these two strategies can have less communication consumptions.
5.4. Experiments

Figure 5.6: Award Distribution Situation Comparison

*ADS* is the second compared factor. It represents the rationality of agent team organisation. Without considering communication consumptions, a one-shot team has an ideal organisation structure because all its team members contribute to task executions. Hence the *ADS* of one-shot teams can be considered as the benchmark of team organisation rationality. Through the experiment, it can be found that the *ADS* of flexible teams is closer to one-shot teams than long-term teams (See Figure 5.6). Therefore, the flexible teams have more reasonable organisations than long-term team.

From the results of the experiment, it can be seen that the flexible team formation mechanism is more suitable for self-interested agents and open environments. It can enable agent teams to keep valuable team members according to their performance and changing of environments. Furthermore, agent teams can adjust their long-term member selection standards through modifying the member evaluation parameters. This feature can make team formation more flexible and suitable for open environments. Therefore, comparing with one-shot
and long-term team formation mechanisms, the flexible team formation mechanism can enable self-interested agents to form more reasonable teams in an open environment with less communication consumptions.

5.5 Related Work

Team formation is an important issue in MAS research. It is a subbranch of agent coordination and organisation. In [71], Horling and Lesser reviewed and evaluated the most important agent organisations in current MAS applications. In [71], Horling and Lesser also introduced agent team as a kind of typical multi-agent organisation, and pointed out that team organisations have increased communication consumptions as a main drawback.

Communication consumptions and computational complexities of several classic team formation and coordination strategies [140] [81] [173] were evaluated by Pynadath through using the COMmunicative Multiagent Team Decision Problem (COM-MTDP) model [139]. COM-MTDP borrowed economic team theory [113] [188] and provided a generic framework that evaluated team formation and coordination strategies. Through the evaluation results obtained by Pynadath, it is obvious that the dynamics of joint goals, domain constraints and task requirements can greatly increase communications within a team.

Gaston used another approach to evaluate agent organisations. In [57], Gaston proposed an approach that evaluated multi-agent organisational performance by using social network theories. Through several experiments, Gaston demonstrated that MAS organisational performances were impacted by the underlying social network structure.

A common feature of Pynadath and Gaston’s methods is that they all cited sociological theories. Especially in recent years, more and more MAS researchers realised the benefit of citing human organisational theories into multi-agent team
formation. Market-base approaches, such as auction [150], voting [134] and contract nets [187], have been applied in many MAS applications. These approaches are especially suitable for self-interested MASs. However, the collaboration terms between agents in most current market-based approaches are normally short (even one-shot). This kind of collaboration aggravates communication consumption problems. Toward shortcomings of market-based approaches, Rathod proposed a stable team formation strategy for self-interested agents [143]. This idea was also borrowed from human behaviour. Rathod suggested to adopt different team strategies in different working domains or situations. However, how to select and automatically refresh different team strategies was not introduced in Rathod’s work.

Expanding on the previous research presented, the mechanism presented in this chapter focuses on the features of self-interested agents and takes agent/environment evaluations into team formation processes. Collaboration terms between agents are based on these evaluation results. This mechanism reduces communication consumptions and avoids unreasonable collaboration relationships in agent teams. The advantages of our team formation mechanism have been demonstrated in the experiments, which are presented in Section 5.4. From the experimental results, it can be seen that the mechanism presented in this chapter has lower communication cost than one-shot team formation, and the organisation reasonability is close to optimal.

5.6 Summary

As a social entity, an intelligent agent needs to collaborate with others in most multi-agent environments. At the same time, especially for self-interested agents, unreasonable team formation mechanisms could obstruct agents to purchase their local profits, or lead to unnecessary system consumptions. Focused on challenges brought by dynamic application domains, many AI researchers suggested using
long-term or one-shot team formation mechanisms in MASs. However, both of these mechanisms have some advantages and disadvantages.

Focused on the features of self-interested multi-agent systems, advantages and disadvantages of one-shot and long-term team formation mechanisms were evaluated in this chapter. Furthermore, a flexible team formation mechanism that can avoid some limitations of one-shot and long-term team formation mechanisms was proposed. The flexible team formation mechanism can enable agents to automatically evaluate the performance of other agents in the system, and select team members with reasonable terms and costs according to the evaluation result. In the flexible team formation mechanism, factors related with agent performance and task requirements are considered as evaluation factors. Through evaluating these factors, team compositions are more reasonable and can avoid some potential benefit conflicts among team members.
Chapter 6

An Ontology-Based Mechanism for Agent Coordination through Knowledge Management

Many complex problems are increasingly required to be solved by using diverse knowledge and multiple techniques. In order to solve these kinds of problems, a MAS needs to import domain knowledge from various knowledge sources. External knowledge sources of a MAS could be developed by different organisations for different purposes. These knowledge sources update their data independently, and may use different formats to represent knowledge. Hence, a MAS can be considered as a synthetical system, which is associated with heterogeneous and independent knowledge sources. To combine multiple techniques, a number of heterogeneous agents developed by different organisations may also be included in a MAS. These agents have different formats to store and represent their knowledge. To work together in the same MAS to achieve some common goals, these heterogeneous agents need to interact with each other to exchange their knowledge or acquire domain knowledge from the system.

In order to help agents obtain the latest domain knowledge from various knowledge sources, it is necessary to include knowledge management mechanisms in MASs. Many MAS researchers suggest the use of ontologies as knowledge description tools in MAS knowledge management mechanisms [20] [45]. Ontology is the taxonomy of objects, classes, properties, functions and their relations. It can be used as a tool for allowing communication and knowledge sharing among
distributed users and applications by providing a semantically rich description and a common understanding of a domain [3] [27]. In the context of MAS, ontologies provide agent-understandable descriptions of knowledge about concepts, relationships and constraints that can exist for an agent or a community of agents. Through using ontologies to describe MAS knowledge, interactions among heterogeneous agents can be bridged through their commitment to the common ontologies of a MAS.

The ontologies of a MAS can be considered as building blocks of knowledge that relate with agents and their working domains. Through using ontologies to describe knowledge of MASs, the knowledge management processes of MASs can be simplified by the following three aspects.

1. Ontologies can describe not only specifications of knowledge of agents but also relations between different knowledge. This feature brings convenience for developing mechanisms to facilitate agents to search and acquire knowledge, especially in information “mass” working environments.

2. Ontologies are usually described in formal languages, such as programming languages and semantic web languages. Although at present, none of these languages is recognised as a standard language for expressing ontologies, several exclusive ontology languages such as OIL [49] and DAML [38], have been widely and successfully applied in many application areas. On the other hand, many ontology-editing tools, which can automatically translate ontologies into different ontology languages, have already been successful developed by some knowledge engineering research institutes. These ontology languages and tools promise the potentials of using heterogeneous knowledge representation formats, and managing heterogeneous knowledge in a MAS.

3. Ontologies describe knowledge of a MAS into structural blocks. Hence, a MAS can adapt to changes of domain knowledge through modulating and
refreshing related ontologies, but not modifying the whole knowledge-base. This feature greatly enhances the adaptability and reusability of MASs.

In this chapter, we introduce an ontology based knowledge management mechanism to handle knowledge related activities of agents (i.e. knowledge sharing, knowledge refreshing, knowledge publication, etc). In this mechanism, ontologies are used to describe knowledge of agents and relationships between different knowledge. In addition, we propose a knowledge management framework, which includes a set of knowledge and ontology services, to facilitate agents to search, publish and refresh ontologies.

The rest of this chapter is arranged as follows. Firstly, some related works that use ontologies and knowledge engineering techniques to handle agent coordinations are introduced in Section 6.1. In Section 6.2, we introduce the formal expression of MAS ontologies. In Section 6.3, classification of MAS ontologies are introduced. Then, Section 6.4 proposes an ontology-based framework for MAS knowledge management. A set of facilitators are included in the framework to provide knowledge related services for agent. Finally, this chapter is summarised in Section 6.5.

### 6.1 Related Work

The heterogeneous and dynamic nature of MAS knowledge sources brings difficulties for agent interactions and knowledge sharing. Some MAS researchers use ontology based methods and some other knowledge engineering methods to handle agent coordinations. In this section, related works using ontology in agent coordinations are reviewed.

Guber [63] [64] proposed the possibility of hiring knowledge engineering methods in MAS coordinations and using ontologies to specify knowledge of MASs. Based on Guber’s work, more and more researchers realised the importance of
including knowledge and ontology related services in MAS infrastructures. Falasconi suggested including ontology and terminology servers in MAS frameworks to mediate semantic transactions among different agents. In addition, RETSINA (which was introduced in Chapter 2) also included ontology servers in the MAS infrastructure. These servers could provide services such as knowledge gathering and semantic mapping [171] [172].

Tamma and Bench-Capon [175] proposed a knowledge model to facilitate knowledge sharing in MASs. This model enriches the usual agent ontologies with some additional semantic information that provides a precise description of agent knowledge properties. These enriched agent ontologies can provide information to facilitate agents to solve problems which are brought by semantic inconsistency and heterogeneity in knowledge representation. However, an enriched agent ontology needs longer building time and more work loads.

Castano et al worked on knowledge/information resource discovery problems. In [27], Castano et al proposed an ontology-based collaboration model for supporting knowledge source discovery in MASs. In this model, ontologies were used to represent the structure and the semantic of information resources of MASs. Based on these ontologies, some knowledge source discovery services were developed by hiring some knowledge engineering methods (such as ontology mapping methods).

In summary, many current MAS researchers have used ontologies and some other knowledge engineering methods to handle coordination problems in MASs, including knowledge sharing, semantic mapping, knowledge gathering, etc. Most of these works treat agents as knowledge users only. However, nowadays, many agents possess strong learning abilities and a high degree of cognitive levels. These agents may not only act as a knowledge user but also have the ability to generate and modify the knowledge of a MAS. In addition, in an open environment, domain knowledge may be refreshed at any time. Many current MAS approaches
do not have a mechanism to handle problems brought by the dynamic refresh-
ment of MAS knowledge sources. In this research, we develop an ontology based
mechanism to handle knowledge related coordinations in MASs. Compared with
most current related works, our work improves agent ontology-based coordina-
tions from the following two aspects:

1. Some specific ontologies are defined to capture the dynamic features of
   knowledge sources of a MAS;

2. A set of services are included in the mechanism to facilitate knowledge
gathering, refreshing and sharing in a MAS.

6.2 Formal Expressions of Ontologies

As previously mentioned, ontologies are formal knowledge specifications that can
be understood by agents. Normally, they are described in formal languages, such
as programming languages and semantic web languages. In our research, we take
OIL languages as the ontology specification language.

OIL [21] [49] is an advanced ontology language that has rich representation
formalism and ontological modelling primitives. An ontology in OIL is repre-
sented via a set of ontology definitions. Four key words, which are class-def,
subclass-of, slot-def, and slot-constraint, are used in ontology definitions to de-
scribe relationships of different concepts. The meanings of these four key words
are described as follows:

- **class-def**: a class definition associated with a class name and a class de-
scription;

- **subclass-of**: a statement of a class’s parent class(es);

- **slot-def**: a slot definition associated with a slot name.
• slot-constraint: a list of global constraint(s) applied to a slot (a slot can also be called as a role or an attribute);

Here, we take the domain of online auction as an example to demonstrate using OIL to express ontologies. In Table 6.1, an ontology of printer products (a kind of auction item) is given. In this ontology, related concepts of printer products are described in a number of classes. For instance, printer is defined as a subclass of digitalProduct; laserJetPrinter is defined as a subclass of printer; and HPLaserJet1100se is defined as subclass of HPProduct and laserJetPrinter. In addition, slots related with printer products and the application domains of these slots are also defined in the printer ontology. For example, manufacturedBy is a slot for all items but printSpeed is only for printer products.

6.3 Classification of MAS Ontologies

Generally, ontologies of a MAS can be classified as common ontologies and special ontologies [14] [73] [189]. Both common ontologies and special ontologies are for specifying related knowledge of MASs and agents. The major difference between common ontologies and special ontologies is whether they are published and accessible for all members of a MAS.

Definition 6.1. A common ontology is a description of common knowledge that is related to a MAS’s working domain or the multi-agent society. Common ontologies are reachable for all agents of a MAS.

Definition 6.2. A special ontology is a description of some specific knowledge that is related to some particular agents or a single agent of a MAS. A special ontology is generated by an individual agent from the local view point of the agent. Special ontologies are not accessible for all agents of a MAS.
ontology-type: “domain ontology”

class-def item
slot-def HasItemID
domain item
class-def digitalProduct
subclass-of item
class-def printer
subclass-of item
slot-def manufacturedBy
domain item
slot-def printingSpeed
domain printer
slot-def printingTechnology
domain printer
class-def laserJetPrinter
subclass-of printer
slot-constraint printingTechnology
has-value “Laser Jet”
class-def HPPProduct
subclass-of item
slot-constraint manufacturedBy
has-value “HP”
class-def HPLaserJet1100se
subclass-of laserJetPrinter AND
subclass-of HPPProduct
slot-constraint printingSpeed
has-value “8ppm”

Table 6.1: Formal Expression of Printer Product Ontology
6.3. Classification of MAS Ontologies

6.3.1 Common Ontologies of MASs

In most current MAS applications, common ontologies are used to describe the domain knowledge of a MAS. The domain ontologies of a MAS are normally developed by the MAS designers. Recently, benefited from development of knowledge engineering studies, ontologies of many application domains, such as E-Commerce and E-Education, have already been defined in some ontology libraries. These ontology libraries are established by several distinguished knowledge engineering research institutes, and applied as external ontology sources by many applications, including MAS applications. However, some MASs may link with various external ontology sources, which are developed by different organisations. These ontology sources may have heterogeneous representation formats and semantic conflicts. This may bring difficulties for MAS knowledge management. To overcome this point, we include two kinds common ontologies in a MAS, i.e. Domain Knowledge Ontologies and Knowledge Source Ontologies.

Domain Knowledge Ontologies

Definition 6.3. A Domain Knowledge Ontology is a shared and common understanding of a particular domain. It includes a representational vocabulary of terms that are precisely defined, and specified with relationships between terms.

A Domain Knowledge Ontology specifies the conceptualisation of a domain in terms of concepts. Each concept represents a class for a specific set of entities of a MAS domain. In our framework, two kinds of relationships are defined to denote associations among different concepts:

- A Subclass-Of relationship organises concepts of a domain to a generalisation hierarchy.

- A Context relationship denotes a specific relationship which is semantically related to a certain context.
Figure 6.1 shows a simple example of *Domain Knowledge Ontologies*, which describes the domain knowledge that is related with IT products. In this ontology, concepts of the IT product domain, which are denoted by rectangles, are linked by lines with different shapes that denote the two kinds of relationships (i.e. the *Subclass-Of* relationships and *Context* relationships). *Subclass-Of* relationships organise related concepts of the ontology in a hierarchical structure. For example, *Hardware Producer* is a sub-class of *IT Company*, and *HP* is a sub-class of *Hardware Producer*. The only *Context* relationship in the *IT Product* ontology is *produce*. This relationship denotes the manufacture relationship between IT companies and IT products. For instance, *HP* (a *Hardware Producer*) manufactures *Printer* and *Computer*. The formal expression of *IT Product* ontology is shown in Table 6.3.1.

**Knowledge Source Ontologies**

The domain knowledge of a MAS could be provided by different external knowledge sources. The variety of knowledge sources brings some problems for agent coordinations. Firstly, various knowledge sources, which are developed by different organisations, may have heterogeneous knowledge representations and semantic conflicts. These heterogeneities and conflicts can make agents misunderstand their working domain. In addition, different knowledge sources may also have some associations. It is necessary to describe associations between different knowledge sources to ensure the accuracy of domain knowledge. Hence, we include *Knowledge Source Ontologies* to specify related information of knowledge sources.

**Definition 6.4.** Knowledge Source Ontologies are ontologies that describe relationships between different knowledge sources and knowledge categories. In addition, the knowledge representation format of a knowledge source is also included in its *Knowledge Source Ontology*. 
Figure 6.1: Ontology of IT Product

Figure 6.2 shows an example of Knowledge Source Ontologies that describe related information of knowledge source Canon Product Knowledge Base. From this ontology, we can see that Canon Product Knowledge Base is a knowledge source that belongs to the IT Product category. It provides knowledge that is related to Canon Product, which is described in KIF language (i.e. a knowledge representation language).

6.3.2 Special Ontologies of Agents

Special ontologies are used to specify knowledge that is related with individual agents of the MAS. An individual agent can also publish its special ontology to a MAS, i.e. convert a special ontology to a common ontology. However, to publish
6.3. Classification of MAS Ontologies

**ontology-type**: “domain knowledge ontology”

```
class-def ITPProduct
class-def ITCompany
slot-def Produce
  domain ITCompany
class-def Software
  subclass-of ITPProduct
class-def Hardware
  subclass-of ITPProduct
class-def Printer
  subclass-of Hardware
class-def Computer
  subclass-of Hardware
class-def SoftwareProducer
  subclass-of ITCompany
class-def Microsoft
  subclass-of SoftwareProducer
class-def HardwareProducer
  subclass-of ITCompany
class-def HP
  subclass-of HardwareProducer
  slot-constraint produce
    has-value Printer AND Computer
```

Table 6.2: Formal Expression of IT Product Ontology

...a special ontology, an agent has to use the common ontology language of the MAS to compose the special ontology, and request system facilitators to verify the ontology before publishing. For example, in the domain of E-Commerce, when a seller wants to spread a new product in the market, it needs to publish ontologies of the new product into the MAS and make those ontologies become common ontologies that can be obtained by all other agents of the MAS.
6.4 An Ontology Based Framework for MAS Knowledge Management

Using ontologies to describe knowledge of MASs brings convenience to refreshing domain knowledge and updating the knowledge of individual agents according to their requirements. However, for a MAS, it is necessary to include knowledge facilitators in the MAS structure to assist agents to acquire, refresh and update knowledge, and manage knowledge that related with the MAS. In this section, we propose a framework for MAS knowledge management.
6.4. An Ontology Based Framework for MAS Knowledge Management

| ontology-type: “knowledge source ontology” |
| class-def KnowledgeSource |
| slot-def Provides |
| domain KnowledgeSource |
| slot-def BelongsTo |
| domain KnowledgeSource |
| slot-def IsExpressedIn |
| domain KnowledgeSource |
| class-def ITProductKnowledge |
| class-def CanonProductKnowledge |
| subclass-of ITProductKnowledge |
| class-def CanonProductKnowledgeBase |
| subclass-of KnowledgeSource |
| slot-constraint IsExpressedIn |
| has-value KIF |
| slot-constraint BelongsTo |
| has-value ITProduct |
| slot-constraint Provide |
| has-value CanonProductInformation |

Table 6.3: Formal Expression of A Knowledge Source Ontology

6.4.1 Framework Structure

The structure of the framework is shown in Figure 6.3. In this framework, the ontology-base and knowledge-base are two databases that store domain knowledge and corresponding common ontologies (which describe domain knowledge of the MAS), respectively. The framework includes a set of facilitators to manage knowledge of the MAS and assists agents in the MAS to acquire and refresh knowledge. The functions of these knowledge facilitators are described as follows.

- **Knowledge Source Monitor**:

  A MAS may have multiple external knowledge sources. **Knowledge Source**
Monitors are facilitators that are used to monitor refreshment of external knowledge sources of a MAS. In addition, Knowledge Source Monitors gather refreshed knowledge from external knowledge sources.

- **Knowledge Source Register:**
  The Knowledge Source Register is the facilitator that registers and manages the information of knowledge sources.

- **Semantic Mapping Engine:**
  The Semantic Mapping Engine is the facilitator that detects and resolves semantic conflicts between knowledge from different knowledge sources.

- **Knowledge Searcher:**
  In this framework, the Knowledge Searcher is the facilitator that assists agents to search and acquire domain knowledge from the knowledge-base.

- **Ontology Board:**
  The Ontology Board is the facilitator that assists agents of a MAS to publish their special knowledge and ontologies. The Ontology Board provides a platform to conduct pre-published special knowledge and ontologies of agents.

### 6.4.2 Knowledge Management Services

Knowledge facilitators, which are introduced in Subsection 6.4.1, manage domain knowledge of the MAS and provide knowledge related services to agents. In this subsection, these services are introduced. Generally, facilitators of the frame work provide five major services:

- The Knowledge Source Registry Service (KSRS);
- The Semantic Mapping Service (SMS);
Figure 6.3: The Framework for MAS Knowledge Management
6.4. An Ontology Based Framework for MAS Knowledge Management

- The Knowledge Monitor and Update Service (KMUS);
- The Knowledge Searching Service (KSS);
- The Ontology Publication Service (OPS).

**The Knowledge Source Registry Service (KSRS)**

External knowledge sources may update their knowledge-base independently. To make domain knowledge of the MAS consistent with the latest knowledge of external knowledge sources, the KSRS is included in our knowledge management framework to register new knowledge sources. The processes of KSRS is described in Figure 6.4. There are two major processes in the KSRS to include a new knowledge source:

1. To establish the Knowledge Source Ontology:

   To include a new knowledge source, the Knowledge Source Register will import the Knowledge Source Ontology of the new knowledge source to the ontology-base. A Knowledge Source Ontology can be imported from the developer of the knowledge source, or established by human experts.

2. To allocate a Knowledge Source Monitor to the new external knowledge source.

   After importing the Knowledge Source Ontology of a new knowledge source, the Knowledge Source Register allocates a Knowledge Source Monitor to the new knowledge source. As introduced in Subsection 6.4.1, A Knowledge Source Monitor is the facilitator that links with a knowledge source and monitors knowledge refreshment in the knowledge source. Knowledge Source Monitors provide Knowledge Monitor and Update Services (KMUSs) in a MAS.
The Semantic Mapping Service (SMS)

Different external knowledge sources and agents of a MAS could represent knowledge in different formats. To make agents understand knowledge in different formats, the SMS is included in the knowledge management framework. This service uses semantic mapping techniques of knowledge engineering and converts knowledge in different representation formats in a standard format. The Semantic Mapping Engine is the facilitator that provides the SMS. The Semantic Mapping Engine imports knowledge of external knowledge sources from the Knowledge Source Monitors, and converts heterogenous knowledge into the standard format of the MAS. Then, the Semantic Mapping Engine stores converted knowledge into the knowledge-base of the system. Furthermore, the Semantic Mapping Engine also links with the Ontology Board. From the Ontology Board, the Semantic Mapping Engine receives special ontologies from individual agents, converts these ontologies into the standard format of the system, and stores them into the ontology-base.
The Knowledge Monitor and Update Service (KMUS)

In dynamic working environments, knowledge in external knowledge sources is changeable, and the MAS needs to keep its knowledge consistent with outside knowledge sources. The KMUS is the service that monitors the updating of knowledge sources. After a knowledge source is registered by the Knowledge Source Register, a Knowledge Source Monitor is allocated to the knowledge source and monitors knowledge refreshment in the knowledge sources. When a Knowledge Source Monitor finds that its monitored knowledge source has updated, the Knowledge Source Monitor will import the updated knowledge to the Semantic Mapping Engine of the system. Then, after operating semantic mapping processes to the knowledge, the Semantic Mapping Engine refreshes the knowledge-base with updated knowledge. Processes of KMUS are also described in Figure 6.5.

![Knowledge Monitor and Refreshing from External Knowledge Servers](image-url)
The Knowledge Searching Service (KSS)

Since it is impossible to store all domain knowledge and common ontologies into individual agents, it is necessary to include knowledge searching services in a MAS to facilitate agents to acquire knowledge. In our knowledge management mechanism, the knowledge and ontology searching services are accomplished by the Knowledge Searcher of a MAS. Generally, the Knowledge Searcher takes two procedures to process a searching query of an agent, i.e. (1) to search the related knowledge category from the ontology-base; and (2) to retrieve the related knowledge from the knowledge-base.

When an agent wants to acquire some domain knowledge, it will send a query to the Knowledge Searcher. Receiving a query, the Knowledge Searcher searches the ontology-base to obtain related knowledge categories of the query from the Domain Knowledge Ontology. Then, the Knowledge Searcher retrieves related knowledge from the knowledge-base according to the knowledge category, which is obtained from the ontology-base. Finally, the Knowledge Searcher sends the searching results to the agent.

The Ontology Publication Service (OPS)

Individual agents of a MAS may possess some special ontologies which are not published in the system. Sometimes, agents may need to publish their special ontologies to make other agents understand them (i.e. make special ontologies become to common ontologies of the MAS). For example, in an E-Commerce domain, an agent may need to publish a special ontology that describes related knowledge of a new product, to make other agents understand the detailed information about the product.

In our framework, we include the OPS to publish special ontologies of individual agents. When an agent wants to publish a special ontology to the system, it sends the ontology to the Ontology Board. Receiving a special ontology, the
Ontology Board informs the Semantic Mapping Engine to operate SMS to the ontology. Then, the Semantic Mapping Engine adds the special ontology to the ontology base of the system. This denotes that the special ontology is transferred to a common ontology of the system (i.e. the special ontology is published).

6.5 Summary

In this chapter, we proposed an ontology based mechanism to handle knowledge related coordinations in MASs. In the mechanism, ontologies in MASs are classified as common ontologies and special ontologies. Common ontologies are further classified as Domain Knowledge Ontologies and Knowledge Source Ontologies. Related information of external knowledge sources of a MAS is described in its Knowledge Source Ontology. By using these ontologies, MAS knowledge is described and arranged in hierarchies, and the dynamic features of independent knowledge sources are captured. Furthermore, we proposed an ontology based framework for knowledge management. In this framework, a set of facilitators are included in the MAS infrastructure. These facilitators provide related services to assist agents to acquire the latest domain knowledge and to publish their knowledge.
Chapter 7

Conclusions

Coordination is one of the major issues of MAS research. It plays a central role in MASs. The remarkable growth of MAS applications brings higher requirements and more challenges to agent coordinations. In recognising these challenges, this thesis deeply investigated agent coordination problems in self-interested MASs, and proposed coordination mechanisms based on three different methodologies. In this chapter, the major contributions of this thesis are summarised and the future work of this research is outlined.

7.1 Summary of Major Contributions

The major contributions of this thesis was to develop agent coordination mechanisms through three different methodologies:

• *Developing CPN based mechanisms to coordinate agent interactions*

  This thesis first investigated the use of CPN techniques in agent coordinations. In Chapter 3, a CPN based mechanism for coordinating agent interactions was proposed. In this mechanism, interaction protocols were described as CPN models, and separated from agents. It was shown now, agents could achieve interactions through sending/receiving messages (i.e. CPN tokens) to/from CPN models. This mechanism was different to most
7.1. Summary of Major Contributions

traditional agent interaction mechanisms, which required agents to be hard-coded with interaction protocols. In comparison to traditional mechanisms, using CPN models to represent protocols could allow agents to select different interaction protocols to operate interactions, showing that the flexibility and extensibility of MASs were enhanced.

Chapter 4 explored some further advantages of CPN based techniques and proposed an approach that utilised CPN analysis techniques to evaluate and predict agent interactions. This approach defined two kinds of interactions (i.e. the default interaction and the proposed interaction) for agents of a MAS, and allowed agents to evaluate the interaction protocol before operating an interaction. By operating analysis processes, agents could select and propose more rational interactions according to their requirements.

• Developing a flexible team formation mechanism for self-interested agents

Another major contribution of this thesis was the development of a flexible team formation mechanism for self-interested agents. As two kinds of wildly used team formation mechanisms, both the one-shot team formation mechanism and the long-term team formation mechanism have two major limitations: (1) communication consumptions in the one-shot team formation mechanism are un-affordable for some MASs; (2) reward allocation in the long-term team formation mechanism are not reasonable enough, especially for self-interested MASs. In the flexible team formation mechanism developed in this thesis, agents could evaluate the performance and importance of other agents in the system, and automatically select team members with reasonable terms and costs according to the evaluation result. Compared to one-shot and long-term team formation mechanisms, the flexible team formation mechanism could make team compositions more reasonable, and avoid some potential benefit conflicts among self-interested team members.
7.1. Summary of Major Contributions

• **Proposing ontology based mechanisms to manage MAS knowledge**

The third contribution of this thesis was the use of ontologies to describe and manage MAS knowledge. Chapter 6 introduced the methods of describing concepts of MAS knowledge and relationships of different concepts in ontologies. MAS ontologies in this research were classified as common ontologies and special ontologies. Furthermore, *Knowledge Source Ontologies* were developed as a kind of MAS common ontologies to describe related information of MAS knowledge sources. Through this way, MAS knowledge was described and arranged in hierarchies, and the dynamic features of independent knowledge sources were captured.

Coordination mechanisms introduced in this thesis can be applied in following potential domains:

• CPN based mechanisms which are introduced in Chapter 3 and Chapter 4 can be applied in many open application domains. One potential application domain is agent based E-Business. In an E-Business domain, agents from different organisations are required to interact each other to realise their goals. In such applications, CPN models can easily link agents from different organisations together, and enable these agents achieve interactions.

• The team-formation mechanism which is introduced in Chapter 5 is suitable for applications that require agents have self-organisation abilities. In addition, this mechanism is also suitable for open environments where agent reputations and system resources are changeable. By using the team-formation mechanism introduced in Chapter 5, agents can automatically estimate resource availabilities and partner reputations, so as to choose suitable cooperation partners.
7.2 Future Works

This research can be extended by engaging in investigations focussing on the following suggestions.

• In Chapter 3 and Chapter 4, two CPN based mechanisms that utilised CPN techniques to represent and analyse interaction protocols of agents were proposed and developed. Future exploration of this research would be to use CPNs to not only model interaction activities, but also to establish rules for agents actions. Agents would be able to exchange/acquire their expertise and strategies with/from other agents. By using a formal language (such as XML) to describe the related CPN model, the expertise of an agent can be transferred to other agents. Then, through analysis of the CPN model, a learner (agent) can understand and learn the expertise from others (agents) automatically, so as to improve the learning ability of agents.

• In Chapter 5, a flexible team forming mechanism was proposed. However, a team in current research is only a kind of simple agent organisations. Future work of this research should focus on:

   (1) to extend self-organisation mechanisms to more complex organisations; and

   (2) to develop a mechanism that allows agents to select organisation style according to their requirements.
Appendix A

Glossary of Terms

ACL  Agent Communication Language
ADS  Award Distribution Situation
AI   Artificial Intelligence
AMS  Agent Management System
ara  Agent Resource Availability
AST  Agent Searching Time
AUML Agent Unified Modeling Language
B(PN^2) Parallel Programming Language
CPN  Colored Petri Net
CPN ML Extension of Standard Meta Language for Colored Petri Net
cr   Contribution Ratio
CSet  Cost Set
DAI  Distributed Artificial Intelligence
DF   Directory Facilitator
DPS  Distributed Problem Solving
FIPA  The Foundation for Intelligent Physical Agents
FIPA-ACL FIPA Agent Communication Language
FIPA-CL FIPA Content Language
GSet  Gain Set
HTML Hypertext Markup Language
IntPro Interaction Proposal
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JADE</td>
<td>Java Agent Development Framework</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>KIF</td>
<td>Knowledge Interchange Format</td>
</tr>
<tr>
<td>KMUS</td>
<td>The Knowledge Monitor and Update Service</td>
</tr>
<tr>
<td>KSS</td>
<td>The Knowledge Searching Service</td>
</tr>
<tr>
<td>KQML</td>
<td>The Knowledge Query and Manipulation Language</td>
</tr>
<tr>
<td>MAS</td>
<td>Multi-agent System</td>
</tr>
<tr>
<td>NGSet</td>
<td>Net Gain Set</td>
</tr>
<tr>
<td>NTS</td>
<td>The Non-understandable Type Set</td>
</tr>
<tr>
<td>OPS</td>
<td>The Ontology Publication Service</td>
</tr>
<tr>
<td>PBC</td>
<td>Petri Net Calculus</td>
</tr>
<tr>
<td>PFA</td>
<td>Parallel Finite Automata</td>
</tr>
<tr>
<td>PN</td>
<td>Petri Net</td>
</tr>
<tr>
<td>Post</td>
<td>The forward incidence matrix of a PN/CPN</td>
</tr>
<tr>
<td>Pre</td>
<td>The backward incidence matrix of a PN/CPN</td>
</tr>
<tr>
<td>PRO</td>
<td>The Proposal Colset</td>
</tr>
<tr>
<td>PTS</td>
<td>The Place Type Set</td>
</tr>
<tr>
<td>RETSINA</td>
<td>Reusable Environment for Task-Structured Intelligent Networked Agents</td>
</tr>
<tr>
<td>SE</td>
<td>Software Engineering</td>
</tr>
<tr>
<td>SML</td>
<td>Standard Meta Language</td>
</tr>
<tr>
<td>SMS</td>
<td>The Semantic Mapping Service</td>
</tr>
<tr>
<td>SR</td>
<td>Supply-Request</td>
</tr>
<tr>
<td>PNML</td>
<td>Petri Net Markup Language</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transmission Control Protocol/Internet Protocol</td>
</tr>
<tr>
<td>TL</td>
<td>Team Leader</td>
</tr>
<tr>
<td>TM</td>
<td>Team Member</td>
</tr>
<tr>
<td>ur</td>
<td>Utilisation Ratio</td>
</tr>
<tr>
<td>UTS</td>
<td>The Understandable Type Set</td>
</tr>
<tr>
<td>VEF</td>
<td>The Virtual Enterprisers Formation</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>XML</td>
<td>Xtensible Markup Language</td>
</tr>
</tbody>
</table>


