Extracting Concurrent Threads from Concurrent Objects

by

Mitsutaka Okazaki

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Prof. Takuya Katayama

School of Information Science
Japan Advanced Institute of Science and Technology

June, 2004

Abstract

This thesis proposes an approach for extracting concurrent execution sequences from concurrent objects. Recently, object-oriented development methods have played an important role in the domain of practical software engineering and have been adapted into developments of real-time systems. However, while object-oriented technologies have been maturing as a diagrammatical description language of systems, less technical tools and supports have been developed for analysing or verifying real-time properties of systems.

This thesis aims to clarify a logical foundation for analysing significant information for a real-time property: 'How many and what kind of threads are concurrently executed in a system'. A thread here means a execution sequence in a system; and since real systems usually have strict physical constraints on the number of CPUs, real-time performance of a system greatly depends on the number of concurrent threads are being executed in the system. Therefore, for analysing the real-time properties of a system it is important to obtain such information. However, existing object-oriented development methodologies do not give enough support to obtain threads from object-oriented models.

This thesis presents a solution of this problem; a transformation method from an object-oriented model into a thread-based model. In our approach, we clearly define two kinds of model. One is the *concurrent object model* which represents a typical object-oriented behaviour model based on concurrent state machines. The other is the *concurrent thread model* that is modelled as a set of explicit threads. It is easy to obtain information about the number of concurrent threads from the latter model. Then, we provide a method for transforming a concurrent object model into a concurrent thread model. This approach is formalised by using Basic Concurrent Regular Expressions (BCREs), which are an extension of regular expressions. There are two extending operators that represent concurrency and communication of concurrent state machines. As a logical base for the transformation method, we propose and use an axiom system for equivalent transformation of BCREs. It is then confirmed that this system is both sound and complete. We present our transformation procedure based on the equivalent transformation of BCREs. By using our axiom system, we also prove that our method is both sound and terminating.

Acknowledgements

I am grateful to many people who have contributed to this thesis. First of all, I would like to thank to Takuya Katayama for his guidance, encouragement and for giving me a chance to study overseas. His supervision has been invaluable for the creation of this thesis. Had he not been my supervisor, I could not have maintained my motivation for this work.

I am also grateful to all members of the judging committee: Katsuhiko Gondow, Kokichi Futatsugi, Xavier Defago and Shin Nakajima for their extensive and excellent comments on drafts of the thesis.

I also would like to say thank my friends and colleagues at JAIST. Especially, Toshiaki Aoki, Naohiro Hayashibara, Kenrou Yatake, Hayato Kawashima, and Tsuyoshi Takahashi for their comments on the work, everyday talking, sharing lunch, dinner, a large amount of drink, and encouragement.

Part of this work was completed while visiting the Imperial College, London. I wish to thank Jeff Kramer and the dse-group members at the Imperial College for their kindness and comments on the work and for many excellent experiences during my visiting. Especially, I would like to thank Gawesh Jawaheer and Sye Loong Keoh for discussions, and for sharing lunches and coffees, and for helping me have fun in London.

Finally, my love and thanks go to my parents and brother for their continual encouragement and support. During working on this thesis, I was financially supported by a Nihon Ekueikai scholarship, for which I am most grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

A number of software development methods have been proposed, with object-oriented development methods recently playing an important role in the domain of practical software engineering. Object-oriented technologies have been widely adopted, not only for large enterprise software but also for real-time embedded software because the complexity of recent embedded systems has dramatically advanced. There are a number of tools and environments that support object-oriented analysis, design and implementation. However, the current object-oriented methods still do not pay enough attention to the verification of real-time software. Little is known on how to guarantee the correctness of real-time software through object-oriented developments. The concern with this problem domain has been growing for a decade and a number of researchers have come to work on it.

Our research is concerned with object-oriented real-time system development. There are some earlier research that handles real-time constraints in object-oriented developments. For example, Real-Time UML[10], which is known as an extension of UML[13] can denote real-time constraints in object-oriented models. Some methods for real-time software developments have also been proposed[3, 4, 5]. However, these methods are roughly defined in a natural language. To verify real-time properties of systems, much knowledge and experience is needed. Because of the lack of a systematic way to check the properties, it is not possible to support verification efforts for computer systems; thus, there is not a way to avoid human errors and mistakes. This is a significant problem for the current object-oriented development methodologies, and it needs to be solved to acheive more dependable real-time software developments.

This thesis aims to clarify a logical foundation for analysing properties of object-oriented real-time software. We focus on the issue of *information of concurrency*: 'How many and

what kind of threads are concurrently executed in a system'. A thread here means an execution sequence in a system. Such information is essential because real-time properties of a system greatly depend on the number of simultaneous threads. A concurrent system is generally implemented as a set of concurrent threads. Logically speaking, all threads in a system are executed concurrently. However, real systems are often implemented in a different way; some threads are allocated to a single or a few CPUs, and then are executed under time-sliced context switching. In such a pseudo concurrent execution, CPU resources are shared by multiple threads. Therefore performance of a system slows down relative to the number of threads. It is thus important for analysing real-time properties to clarify how many and which threads are executed at the same time in a system.

1.2 Concurrent Thread Model

To obtain the information of concurrency, this thesis focuses on the *concurrent thread model*, which consists of a set of threads and a global automata shown in Figure 1.1. In this model, a global automata controls global execution timing of concurrent threads. In Figure 1.1, a dotted arrow denotes a thread. A set of threads is assigned to each of the states of the global automata. When the system is in a particular state, threads assigned to that state are executed concurrently. The model in Figure 1.1 denotes the following behaviour. The system begins with the state0 and a single thread named A is executed in this state. The system state is changed to the next state after the all assigned threads are terminated. When the thread A terminates in the state 0, the state is changed to states 1 or 2 (non-deterministic). We here assume that the state has been changed to state 2. Then, in state 2, threads D,E and F are executed concurrently. Then, the state is changed to state3 when all threads assigned to state2 are stopped, and then threads in the state3 wake up executed, and ..., the execution is then continued in a similar way. Using the concurrent thread model, it is clear that how many and which threads are executed concurrently with respect to states of a system.

In system design, using a concurrent thread model is a reasonable way when there is a need to analyse real-time properties because information of concurrency is clear in the model. However, the problem is that analysis models generally used in a object-oriented development have quite different architecture from the concurrent thread model. Unfortunately, it is very complicated to extract concurrent threads from an object-oriented model. Although an object is also considered as a concurrent entity like a thread, there generally is not a one-to-one relationship between an object and a thread. Let us consider that a system consists of many objects. Since there is a limit on the number of CPUs, performance may become very slow if all objects are

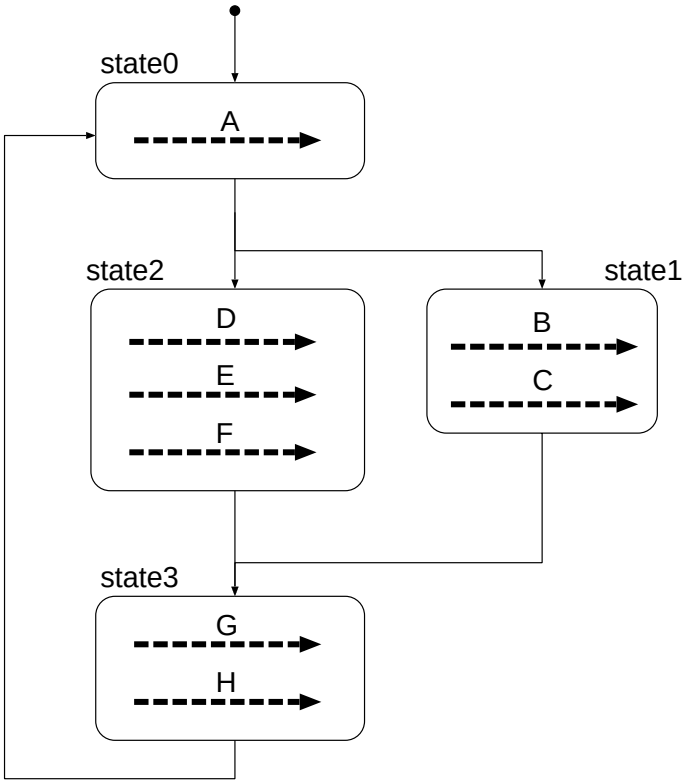


Figure 1.1: Concurrent thread model

mapped to concurrent threads. A typical design is to construct each thread according to a group of objects or part of objects that will never be executed concurrently.

A reasonable way to identify an object group for a thread is along a communication sequence between objects. Figure 1.2 depicts a system that has three objects: A, B and C. In this figure, a round square represents an object. The enclosing box around objects are the boundary between the inside and the outside of the system. A dotted arrow represents the communication between objects. Communication here means an abstraction of an event or a method invocation. A number next to an arrow line represents the order of occurrence of the communication. For example, once event 1 reaches from the outside of the system, event 1.1 occurs. Then, event 1.2 occurs and goes out of the system. In the same manner, the event sequence $2 \rightarrow 2.1 \rightarrow 2.2 \rightarrow 2.3$ occurs for the incoming event 2. For event 3, $3 \rightarrow 3.1 \rightarrow 3.2$ occurs. A group of objects is constructed through such a sequence. From the system depicted in Figure 1.2, three groups are obtained. The objects A and B are grouped from a sequence for event 1. The objects A, B and C are for event 2 and, the objects B and C are for event 3. Each thread is implemented sequentially as a procedure for executing objects of a group . For example, a thread for responding to event 1 is implemented as a procedure that invokes objects A then B. The whole system behaviour can be modelled as a set of threads.

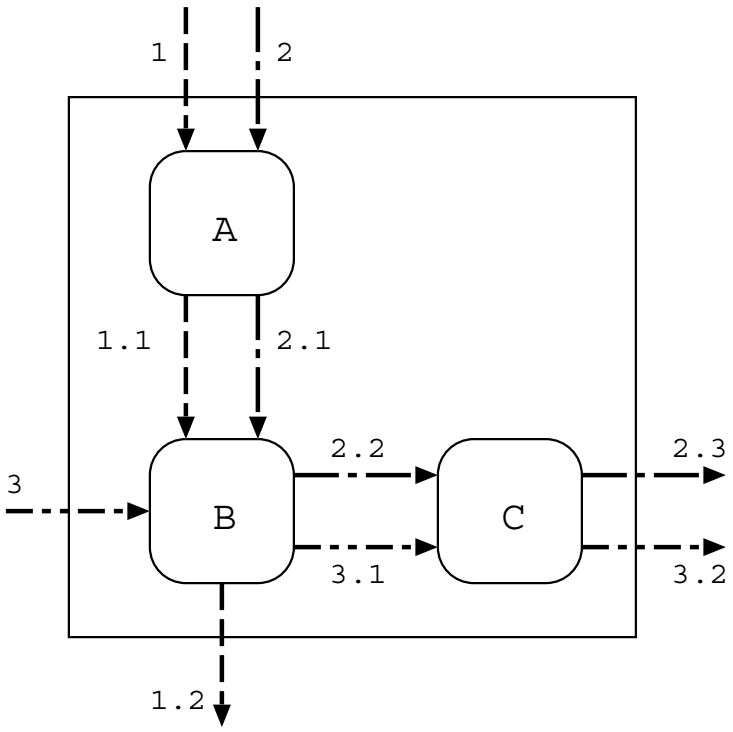


Figure 1.2: Object and Thread

Some object-oriented development methods for embedded systems, for example, SES approach[3] and OCTOPUS method[4] adopt a thread-based model as their design model for a system. Since an object-oriented model is adopted as their analysis model, it is necessary to obtain a concurrent thread model from an analysis model at the beginning of the design phase. However, these methods do not propose any systematic way of grouping objects and constructing threads from an object-oriented model. Therefore, developers need to extract threads by their heuristics; however, this is inefficient and there is also some risk of human mistakes. Moreover, if a large number of objects are given, it is almost impracticable to exhaust all possible threads by hand.

1.3 Approach

The objective of this work is to propose a systematic way to transform an object-oriented model called a *concurrent object model* to a concurrent thread model. This objective is accomplished according to the following approach.

1. We formalise the concurrent object model and the concurrent thread model using Basic Concurrent Regular Expressions(BCREs).

2. We propose a complete axiom system F_c for BCREs. This axiom system can prove the

equivalence of BCREs.

3. We present a method of how to transform a concurrent object model to a concurrent thread model. The axiom system F_c is used to prove that our method preserves the behaviour of a target system between before and after the transformation.

1.4 Thesis outline

This thesis is organised into 8 chapters as follows. In Chapter 2, the definition of Basic Concurrent Regular Expressions is introduced. In Chapter 3, a complete axiom system F_c is given. Equivalence of BCREs can be proven utilizing this system. Chapter 4 describes how to formalise the concurrent object and thread models using BCREs. In Chapter 5, a transformation method from a concurrent object model to a concurrent thread model is proposed. Chapter 6 describes some transformation examples using our transformation method. Chapter 7 compares our work to some related researches. Chapter 8 presents some conclusions and future directions for our work.

Chapter 2

Basic Concurrent Regular Expressions

In this chapter we define the syntax and semantics of Basic Concurrent Regular Expressions (BCREs), which is a behaviour description language for concurrent systems. It is based on Concurrent Regular Expressions (CREs)[2] that is an extension of Regular Expressions. There are four extending operators in CREs: interleaving, alpha-closure, synchronous composition and renaming. The interleaving operator represents concurrency between a finite number of state machines. The alpha-closure operator denotes concurrency between an infinite number of state machines. The synchronous composition operator represents communication between state machines. The renaming operator can rename transition labels in state machines. In this thesis, we adopt the interleaving and synchronous composition operators. We do not adopt the other two operators as they are not essential for modelling our software in the remaining chapters of the thesis.

Before giving the exact definition of BCREs, let us look at an example expression.

$$((a+b).c^*) \,[\,]\, (a.(c \| d).e)$$

$a, b, \cdots$ are symbols. The operator ., $+$ and $*$ are similar to the ones of regular expressions. The operator . represents a sequence of action and $+$ a choice of actions. The operator $*$, called closure, means zero or more iteration. The operators $\|$ and $[\,]$ are to describe concurrent behaviours. The expression $\alpha \| \beta$ means that α and β occur concurrently. On the other hand, the operator $[\,]$ represents not only concurrency but also communication. $\alpha \,[\,]\, \beta$ is interpreted as that α and β occur concurrently and communicate with each other.

2.1 Syntax

Let us define the syntax of BCREs. We assume that σ is a finite set of symbols and use $a, b, \cdots$ to range over σ. Then, the syntax of BCREs on σ is defined as follows.

DEFINITION **2.1 (Syntax)**

1. *$a \in \sigma \cup \bot$ is a BCRE.*
2. *Assume that α and β are BCREs. Then, $\alpha + \beta$, $\alpha.\beta$, α^*, $\alpha \parallel \beta$, $\alpha \,[\, s \,]\, \beta$ and (α) are also BCREs*

where s is a set of actions ($s \subseteq \sigma$). The symbol $\bot$ is a special character that means an empty word. The operator $\parallel$ is called interleaving and $[\, s \,]$ is called synchronous composition. The only difference between BCREs and the standard regular expressions is that these two operators exist. The other operators $+$,. and $*$ are similar to the ones in regular expressions.

2.2 Semantics

The semantics of BCREs are defined as a set of sequences on symbols. Such a set is also called the language of BCREs. L is used as a projection from a BCRE to its language. The definition of L, that is the semantics of BCREs, is as follows.

DEFINITION **2.2 (Language of BCREs)**

Let a be in σ, α and β are BCREs on σ and, ω, ω_1 and ω_2 be sequences on σ. Then,

1. $L(\bot) = \phi$, $L(\epsilon) = \{\epsilon\}$, $L(a) = \{a\}$
2. $L(\alpha.\beta) = \{\omega_1 \cdot \omega_2 \,|\, \omega_1 \in L(\alpha),\ \omega_2 \in L(\beta)\}$
3. $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
4. $L(\alpha^*) = \bigcup_{i=0,1,\cdots} L(\alpha^i)$
5. $L((\alpha)) = L(\alpha)$
6. $L(\alpha \parallel \beta) = \{\omega \,|\, \omega_1 \in L(\alpha), \omega_2 \in L(\beta), \omega \in intl(\omega_1, \omega_2)\}$
7. $L(\alpha \,[\, s \,]\, \beta) = \{\omega \,|\, \omega \in (\sigma(\alpha) \cup \sigma(\beta))^*,\ \omega/(\sigma(\alpha) \cup S) \in L(\alpha),\ \omega/(\sigma(\beta) \cup S) \in L(\beta)\}$

where $intl$ is defined as follows.

- $intl(a, \epsilon) = intl(\epsilon, a) = \{a\}$

- $intl(a \cdot \omega_1, b \cdot \omega_2) = \{a \cdot \omega \,|\, \omega \in intl(\omega_1, b \cdot \omega_2)\} \cup \{b \cdot \omega \,|\, \omega \in intl(a \cdot \omega_1, \omega_2)\}$

σ^* represents a set of all sequences on σ, $\sigma(\alpha)$ is a set of all the symbols that appear in $L(\alpha)$ and ϵ is an empty word (a zero length sequence).

$\epsilon \cdot \omega = \omega \cdot \epsilon = \omega$ follows for all ω. α^imeans a i times sequence of α:

$$\alpha^0 = \epsilon,\ \alpha^1 = \alpha,\ \alpha^2 = \alpha\alpha,\ \alpha^3 = \alpha\alpha\alpha, \cdots$$

For $s \subseteq \sigma$, ω/s means a restriction of ω over s. Any symbol not in s is removed from w. For instance,

$$a \cdot b \cdot c \cdot d/\{a, c\} = a \cdot c$$

$\omega/s = \epsilon$ if ω contains no symbol in s. In the remainder of this thesis, we omit some parenthesis around a restricting operation if it does not make the expression ambiguous. For example, we denote $\omega_1 \cdot \omega_2/\sigma(\alpha) \cup \sigma(\beta)$ instead of $(\omega_1 \cdot \omega_2)/(\sigma(\alpha) \cup \sigma(\beta))$.

2.3 Modelling behaviour

In this section we illustrate some brief examples of software behaviour and explain an intuitive meaning of BCREs, what behaviour is, and what systems can be modelled with BCREs.

2.3.1 Action

We use the term *action* as the atomic behaviour of a system. An action can be considered as an atomic behaviour of a system such as an event, a method invocation, a line of source code and so on. We do not discuss in depth what is an appropriate relationship between an action and its implementation. A real program instance corresponds to an action that can be settled according to a design decision that is different among various software developments. An action in this thesis is a kind of abstract concept

and is loosely defined as an atomic behaviour that is considered as a fragment of behaviour that can not be divided into any smaller actions.

Syntactically, an action is described as a string that is an identifier for a single action. We denote an action as a string with lower-case characters. Such a string sometimes has subscripts. For example, a, b, c, $a_0, a_1, ..., a_n, open, close, post$ and get are used as denotations of an action.

2.3.2 Action sequence

Every possible behaviour of a system is regarded as a sequence of actions. Let us consider a system whose behaviour is defined by a state machine shown in Figure 2.1.

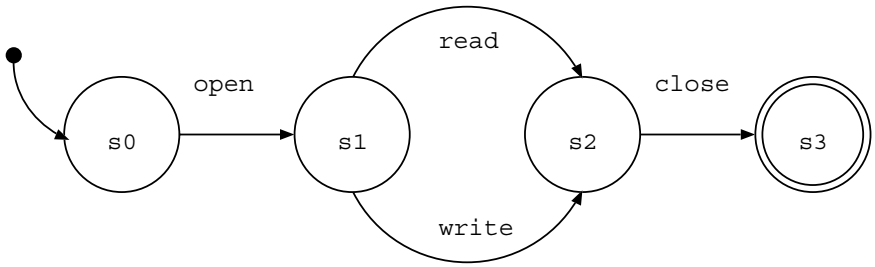


Figure 2.1: State Machine Example

This state machine begins with the initial state s_0. After the action $open$ occurs, its state changes to s_1. Then after the action $read$ or $write$, the system reaches the state s_2. Finally, the action $close$ occurs and the system reaches the final state s_3. Such behaviour can be denoted as action sequences $open \cdot read \cdot exit$ or $open \cdot write \cdot exit$.

2.3.3 Behaviour

Whole behaviours of a system can be defined as a set of action sequences. Figure 2.2 depicts a state machine with a loop of transition. The initial state s_0 is also a final state. The behaviour

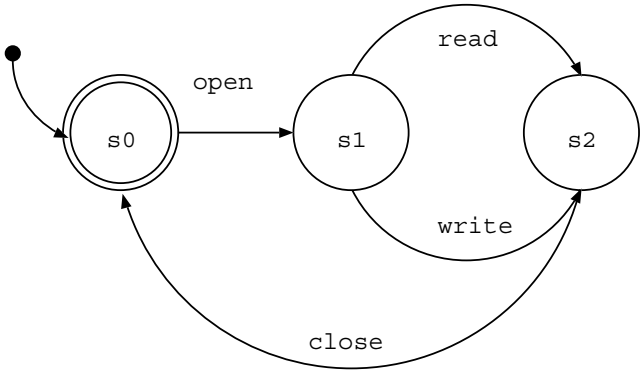


Figure 2.2: State Machine with loop

of state machine of Figure 2.2 is that

$$open \cdot read \cdot exit,$$
$$open \cdot write \cdot exit,$$
$$open \cdot read \cdot exit \cdot open \cdot read \cdot exit, \cdots$$
$$open \cdot write \cdot exit \cdot open \cdot read \cdot exit, \cdots$$
$$\cdots$$

If a system has infinite repetitions of behaviour, infinite action sequences may be observed. It is impossible to write such a set of infinite sequences directly as a set of sequences. To describe

such behaviour, a higher notation that can handle infinity is required. The closure operator (*) can be used as such notation. The expression α^* represents an arbitrary time repetition of α.

It is easy to define behaviour of a single state machine as a BCRE. In a similar way, to transform a state machine to a regular expression, we can obtain a BCRE that corresponds to a state machine. All actions are mapped to symbols of BCREs. For example, the behaviour of the state machine shown in Figure 2.2 can be described by the regular expression:

$$(open.(read + write).close)^*$$

The set of action sequences that this expression means is its language:

$$L((open.(read + write).close)^*)$$

2.3.4 Concurrency

Concurrent behaviour can be denoted by using the $||$ operator of BCREs. In the semantics of BCREs, concurrency is modelled by *interleaving semantics*. Interleaving means a serialization of concurrent sequences. With interleaving semantics, concurrent behaviour is not distinguished from a choice of their possible serialization. For example, $a||b$ is not distinguished from a choice of ab and ba. $ab||cd$ is not distinguished from a choice among $abcd$, $acbd$, $acdb$, $cabd$, $cadb$ and $cdab$. The following definition $intl$ is a projection from concurrent sequences to an interleaving sequence.

- $intl(a, \epsilon) = intl(\epsilon, a) = \{a\}$
- $intl(a \cdot \omega_1, b \cdot \omega_2) = \{a \cdot \omega \,|\, \omega \in intl(\omega_1, b \cdot \omega_2)\} \cup \{b \cdot \omega \,|\, \omega \in intl(a \cdot \omega_1, \omega_2)\}$

The following is an example of a state machine with internal concurrency. There are two sub state machines inside the round square. They are executed concurrently after the action $open$ is completed. Thus actions $read$ and $write$ are executed concurrently, then action $close$ is executed. This state machine can be denoted by BCREs as follows.

$$open.(read||write).close$$

The behaviour is

$$L(open.(read||write).close) = \{open \cdot read \cdot write \cdot close,\ open \cdot write \cdot read \cdot close\}$$

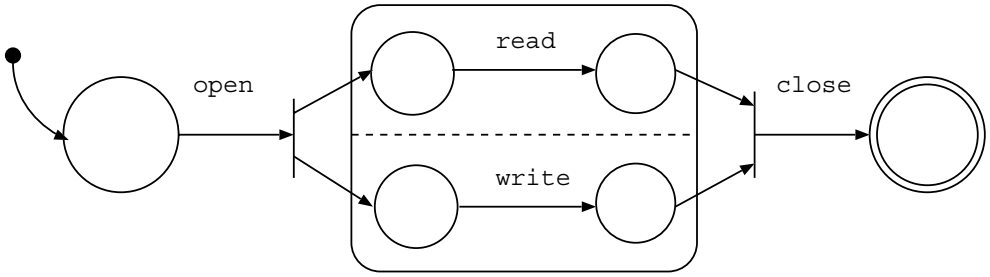


Figure 2.3: State Machine with Concurrency

2.3.5 Communication

Assume that there is a system that consists of two concurrent state machines, and α and β are defined as the behaviour of these machines, respectively. Suppose that these state machines are executed concurrently and communicate with each other. Then, the behaviour of this system can be defined as $\alpha [\ ] \beta$.

The operation $[\ ]$ represents both communication and concurrency. If there is no communication between α and β, $\alpha [\ ] \beta$ has the same language as $\alpha \| \beta$. In the expression $\alpha [\ ] \beta$, the same symbols appearing in both α and β are called *communication symbols*. These symbols mean actions for *synchronous communication* between α and β. Synchronous communication is a communication that satisfies the rule that all participants are blocked until the end of the communication and that communications never fail.

Figure 2.4 depicts two concurrent state machines. These two state machines can be defined

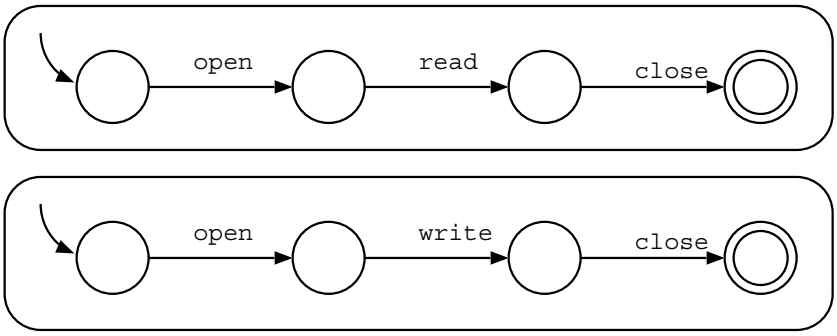


Figure 2.4: Concurrent State Machines with Communication

as $open \cdot read \cdot close$ and $open \cdot write \cdot close$. $open$ and $close$ are communication symbols between them. If these state machines are executed concurrently without communication, the behaviour of this system can be represented with a set of fully interleaved sequences as follows.

$$\{\ open \cdot read \cdot close \cdot open \cdot write \cdot close,\ open \cdot read \cdot open \cdot close \cdot write \cdot close,$$
$$open \cdot read \cdot open \cdot write \cdot close \cdot close,\ open \cdot write \cdot open \cdot read \cdot close \cdot close$$

$$open \cdot write \cdot open \cdot close \cdot read \cdot close,\ open \cdot write \cdot close \cdot open \cdot read \cdot close,$$
$$open \cdot open \cdot read \cdot close \cdot write \cdot close,\ open \cdot open \cdot read \cdot write \cdot close \cdot close,$$
$$open \cdot open \cdot write \cdot read \cdot close \cdot close,\ open \cdot open \cdot write \cdot close \cdot read \cdot close\ \}$$

Some of these sequences violate the rule of synchronous communication. If a sequence has a communication symbol that does not adjoin with the corresponding communication symbol, such a sequence is regarded as violating the rule of synchronous communication. For example,

$$open \cdot open \cdot read \cdot write \cdot close \cdot close$$

does not violate the rule but the following does.

$$open \cdot read \cdot open \cdot write \cdot close \cdot close$$

If communication succeeds, there must be no other symbols between two corresponding communication symbols in a sequence. The latter sequence represents the behaviour where the communication $open$ did not block until the end of the communication. There is an action $read$ between two communication symbols $open$. This sequence shows a behaviour where $open.read.close$ performs the action $read$ before $open.write.close$ finishes $open$. In other words, $open.read.close$ performs $read$ without waiting for $open$ of $open.write.close$. Such behaviour clearly violates synchronous communication.

The $[\ ]$ operation eliminates such sequences from its language. Only the following two sequences are left as behaviour of $open.read.close[\ ]open.write.close$

$$\{open \cdot open \cdot read \cdot write \cdot close \cdot close,\ open \cdot open \cdot write \cdot read \cdot close \cdot close\ \}$$

It is clear in the set above that once $open$ occurs, the neighbour symbol is also $open$. The $[\ ]$ operation eliminates such redundancy. $open{\cdot}open$ is replaced with a single $open$ and $close{\cdot}close$ is replaced with $close$. Thus, the exact language is as follows.

$$\{open \cdot read \cdot write \cdot close,\ open \cdot write \cdot read \cdot close\ \}$$

According to the rule of synchronous communication, communications never fail. However, we can write expressions that never satisfy this rule. For example, let us consider $a.b.a[\ ]a.b.c$. The second occurrence of a in $a.b.a$ fails to communicate with $a.b.c$ because $a.b.c$ has only one occurrence of a. There is no other sequence that satisfies the rule of communication, and so the whole behaviour of such a system becomes an empty set, that is, $L(a.b.a[\ ]a.b.c) = \phi$

2.4 Abbreviations

In the remainder of this thesis, we use some abbreviations to describe more precise expressions. Firstly, the sequential operator (.) is omitted unless this makes the expression ambiguous. For example, $\alpha.\beta$ will be simply denoted as $\alpha\beta$.

We will omit some redundant parenthesis according to the binding power of the operators. The following is a list of operators in order of their ascending binding power.

$$* \ . \ + \ \| \ [\, s \,]$$

So we can denote

$$a^*b\,[\, s \,]\, bc \| de$$

as

$$((a^*)b)\,[\, s \,]\, ((bc) \| (de))$$

According to the associative property of the operator (.), that is $L((\alpha\beta)\gamma) = L(\alpha(\beta\gamma))$, we will omit some parenthesis. For example, we simply write $\alpha\beta\gamma$ instead $(\alpha\beta)\gamma$ or $\alpha(\beta\gamma)$. Since $+$ and the $[\ ]$ operator are also associative, we will omit parenthesis in a similar way unless it makes the expression ambiguous.

Especially for a series of $+$ operations, the following summation symbol is used.

DEFINITION **2.3 (Summation)**
Let $\alpha_1, \cdots, \alpha_r$ be BCREs. Then,

$$\sum_{j=1}^{r} \alpha_j \equiv \alpha_1 + \alpha_2 + \cdots + \alpha_r$$

where $\sum_{j=1}^{r} \equiv \alpha_1$ for $r = 1$

2.5 Summary

In this chapter, we presented the concept of Basic Concurrent Regular Expressions and gave a concrete definition of the syntax and semantics of BCREs, which are an extension of the regular expressions known so far. There are two extending operators $\|$ and $[\, s \,]$. The former represents concurrency and the latter represents communication between state machines. In the semantics of BCREs, concurrency is modelled by interleaved semantics between, and can be represented by, BCREs. Synchronized communication is assumed for the semantics of the

$[\,s\,]$ operator. Such a definition of semantics for concurrent systems is almost similar to that in process algebra. We will discuss on the relationship between process algebra and our BCREs in Chapter 7.

We also illustrated how to define the behaviour of software by using BCREs with some brief examples. By using BCREs, we can define the behaviour of a concurrent system as an expression that represents a set of observable action sequences from concurrent state machines.

Chapter 3

Complete Axiom System

In this chapter we introduce the axiom system F_c that provides a systematic way to prove the language equivalence of BCREs. We also show that our axiom system F_c is complete, that is, a proof always is derivable in F_c if two BCREs have the same language. According to the completeness, we can check equivalences between any BCREs systematically. The main purpose of introducing the system F_c is that we use F_c as a logical base to ensure the correctness of our thread extraction method realized as an iterative equivalent transformation process of BCREs in Chapter 5.

3.1 Axiom system F_c

The axiom system F_c consists of 21 axioms and two inference rules. Each axiom denotes an atomic relationship between two expressions in terms of language equivalence. All axioms are formed in a schema $\alpha = \beta$, and $L(\alpha) = L(\beta)$ holds for every axioms. The table 3.1 is a list of all axioms in the system F_c. In this table $\alpha, \beta, \cdots$ are BCREs and $a, b, \cdots$ are symbols. The axiom A_1 to A_{11} are known as algebraic properties of regular expressions. They are originally from the axiom system F_1' proposed by A.Salomma in [6]. The inference rules of F_c are also the same as F_1':

- R1 (Substitution). Assume that $\alpha = \beta$ and $\gamma = \delta$. Then one may infer the equation $\gamma[\beta/\alpha] = \delta$ where $\gamma[\beta/\alpha]$ is the result of replacing an occurrence of α in γ by β.
- R2 (Solution of equations). Assume that β does not possess an *empty words property*(e.w.p.). Then one may infer the equation $\alpha = \beta^*\gamma$ from the equation $\alpha = \beta\alpha + \gamma$.

where it is stated that α possesses an e.w.p. if and only if $\epsilon \in L(\alpha)$. We simply write $\vdash \alpha = \beta$ if an equation $\alpha = \beta$ can be derivable in F_c.

$$
\begin{array}{lll}
A_1 & \alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma \\
A_2 & \alpha(\beta\gamma) = (\alpha\beta)\gamma \\
A_3 & \alpha + \beta = \beta + \alpha \\
A_4 & \alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma \\
A_5 & (\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma \\
A_6 & \alpha + \alpha = \alpha \\
A_7 & \alpha\bot^* = \alpha \\
A_8 & \alpha\bot = \bot \\
A_9 & \alpha + \bot = \alpha \\
A_{10} & \alpha^* = \bot^* + \alpha\alpha^* \\
A_{11} & \alpha^* = (\bot^* + \alpha)^* \\
C_1 & \alpha\,[\,s\,]\,\bot = \bot \\
C_2 & \alpha\,[\,s\,]\,\bot^* = \begin{cases} \alpha & \text{if } \sigma(\alpha) \cap s = \phi \\ \bot & \text{otherwise.} \end{cases} \\
C_3 & \alpha\,[\,s\,]\,\beta = \beta\,[\,s\,]\,\alpha \\
C_4 & x\alpha\,[\,s\,]\,y\beta = \begin{cases} x(\alpha\,[\,s \cup \{x\}\,]\,\beta) & \text{if } x = y \\ x(\alpha\,[\,s\,]\,y\beta) & \text{if } x \neq y,\ x \notin \sigma(\beta) \cup s,\ y \in \sigma(\alpha) \cup s \\ x(\alpha\,[\,s\,]\,y\beta) + y(x\alpha\,[\,s\,]\,\beta) & \text{if } x \neq y,\ x \notin \sigma(\beta) \cup s,\ y \notin \sigma(\alpha) \cup s \\ \bot & \text{if } x \neq y,\ x \in \sigma(\beta) \cup s,\ y \in \sigma(\alpha) \cup s \end{cases} \\
C_5 & (\alpha + \beta)\,[\,s\,]\,\gamma = \alpha\,[\,s \cup (\sigma(\beta) \cap \sigma(\gamma))\,]\,\gamma + \beta\,[\,s \cup (\sigma(\alpha) \cap \sigma(\gamma))\,]\,\gamma \\
C_6 & \alpha \parallel \bot = \bot \\
C_7 & \alpha \parallel \bot^* = \alpha \\
C_8 & \alpha \parallel \beta = \beta \parallel \alpha \\
C_9 & x\alpha \parallel y\beta = x(\alpha \parallel y\beta) + y(x\alpha \parallel \beta) \\
C_{10} & (\alpha + \beta) \parallel \gamma = \alpha \parallel \gamma + \beta \parallel \gamma
\end{array}
$$

Table 3.1: The axioms in F_c

3.2 Soundness

An equation $\alpha = \beta$ is said to be valid if and only if $L(\alpha) = L(\beta)$ holds. The axiom system F_c is said to be sound if and only if all derivable equations are valid.

THEOREM **3.1**

The axiom system F_c is sound.

PROOF

It is obvious that the axioms A_1 to A_{11} are valid and the rule R1 preserves validity. It is well known that if a regular expression β does not possess an e.w.p, then $\alpha = \beta\alpha + \gamma$ has only one solution $\alpha = \beta^*\gamma$ (cf. [7] or [8]). Therefore, $R2$ also preserves validity. As for C_1 to C_{10}, new in the system F_c, they are also valid. Thus the Theorem 3.1 follows. □

See the Appendix for detailed validity proof for C_1 to C_{10}.

3.3 Completeness

It is stated that F_c is complete if and only if $\vdash \alpha = \beta$ can be derivable for any α and β which satisfies $L(\alpha) = L(\beta)$. This section proves the completeness of F_c.

In the reminder of this section, a proof is described in three parts. First, in the Section 3.3.1, we set out some important definitions and lemmas that are referred from other parts. Then in the Section 3.3.2, we prove that all BCREs are *equationally characterized* as Theorem 3.2. Finally, in Section 3.3.3, we prove that the system F_c is complete if the BCREs are equationally characterized (Theorem 3.3).

3.3.1 Preliminary

The following lemma describes the basic properties of BCREs.

LEMMA **3.1**

Suppose that α, β, γ and δ are BCREs, then the following holds.

1. $\vdash \alpha = \alpha$
2. $\vdash \alpha^* = \beta^*$ *and* $\vdash \beta = \alpha$, *if* $\vdash \alpha = \beta$
3. $\vdash \alpha = \gamma$, *if* $\vdash \alpha = \beta$ *and* $\vdash \beta = \gamma$
4. $\vdash \alpha + \gamma = \beta + \delta$ *and* $\vdash \alpha\gamma = \beta\delta$, *if* $\vdash \alpha = \beta$ *and* $\vdash \gamma = \delta$

5. $\vdash \perp\alpha = \perp$

6. $\vdash \perp^*\alpha = \alpha$

7. $\vdash \gamma[\beta/\alpha] = \gamma$, *if* $\vdash \alpha = \beta$

It is easy to show that these are derivable in F_c by using the rules $R1$, $R2$ and the axioms A_1 to A_{11}. In the remainder of this section, we use the above equations and the substitution rule $R1$ without explicitly referring to them.

DEFINITION **3.1**

Let ξ *be a finite set of BCREs:* $\xi = \{\alpha_1, \alpha_2, \cdots, \alpha_r\}$ *for some natural number* r. $\chi(\xi)$ *is defined as follows.*

$$\chi(\xi) \equiv \left\{ \sum_{i=1}^{r} \omega_i \;\middle|\; \omega_1, ..., \omega_r \in \xi,\ \omega_i \neq \omega_j \text{ for all } i \neq j. \right\}$$

where $1 \leq r$,$1 \leq i \leq r$ *and* $1 \leq j \leq r$. *For example,*

$$\chi(\{\alpha, \beta, \gamma\}) = \{\alpha+\beta+\gamma, \alpha+\gamma+\beta, \beta+\alpha+\gamma, \beta+\gamma+\alpha, \gamma+\alpha+\beta, \gamma+\beta+\alpha\}$$

DEFINITION **3.2**

Let ξ *be a finite set of BCREs and* $\xi \neq \phi$, *then* $\psi(\xi)$ *is defined as the following.*

$$\psi(\xi) \equiv \left\{ \omega \;\middle|\; \omega \in \chi(\Omega),\ \Omega \in 2^{\xi} \right\}$$

LEMMA **3.2**

Assume that ξ *is a non-empty finite set of BCREs. Then* $\xi \subseteq \psi(\xi)$.

PROOF

Let x be a BCRE that belongs to ξ. Such a BCRE always exists because ξ is non-empty (by hypotheses). It follows that the set $\{x\}$ belongs to the power set of ξ (i.e., $\{x\} \in 2^{\xi}$), and $x \in \chi(\{x\})$. By the definition of ψ, it follows that $x \in \psi(\xi)$. Therefore, $\xi \subseteq \psi(\xi)$ thus completing the proof. □

LEMMA **3.3**

Suppose that $\xi = \{\alpha_1, \cdots, \alpha_r\}$ *and* $\omega \in \chi(\xi)$, *then,*

$$\vdash \omega = \sum_{j=1}^{r} \alpha_j$$

PROOF

According to the definition of χ, some $\omega_1, \cdots, \omega_r \in \xi$ exist and, $\omega \equiv \omega_1 + \cdots + \omega_r$ where $\omega_i \neq \omega_j$ for all $i \neq j$. Therefore, all $\omega_1, \cdots, \omega_r$ are r numbers of BCREs being different from each other. Since ξ has just r number of elements, $\alpha_1, \cdots, \alpha_r$ can be obtain by properly sorting $\omega_1, \cdots, \omega_r$. Hence, by A_3 and A_9,

$$\vdash \omega = \omega_1 + \cdots + \omega_r = \alpha_1 + \cdots + \alpha_r$$

□

LEMMA **3.4**

Let ξ_1 and ξ_2 be non-empty finite sets of BCREs and, assuming $\omega_1 \in \chi(\xi_1)$ and $\omega_2 \in \chi(\xi_2)$. Then, some ω_3 exists, $\omega_3 \in \chi(\xi_1 \cup \xi_2)$ and

$$\vdash \omega_1 + \omega_2 = \omega_3$$

holds.

PROOF

1. Suppose that $\xi_1 = \xi_2$, then, some r exists and $\xi_1 = \xi_2 = \{\alpha_1, \cdots, \alpha_r\}$. By lemma 3.3,

   $$\vdash \omega_1 = \omega_2 = \omega_3 = \sum_{j=1}^{r} \alpha_j$$

   Hence, $\vdash \omega_1 + \omega_2 = \omega_3$ holds obviously.

2. Suppose $\xi_1 \neq \xi_2$. Let us prove the lemma from the following three cases.

   (a) In the case $\xi_1 \subseteq \xi_2$, there are some m and n such that

   $$\xi_1 = \{\alpha_1, \cdots, \alpha_m\}, \quad \xi_2 = \{\alpha_1, \cdots, \alpha_m, \beta_1, \cdots, \beta_n\}$$

   holds. By lemma 3.3,

   $$\vdash \omega_1 = \sum_{j=1}^{m} \alpha_j, \quad \vdash \omega_2 = \omega_3 = \sum_{j=1}^{m} \alpha_j + \sum_{j=1}^{n} \beta_j,$$

   Therefore,

   $$\vdash \omega_1 + \omega_2 = \sum_{j=1}^{m} \alpha_j + \sum_{j=1}^{n} \beta_j = \omega_3$$

   (b) Suppose that $\xi_2 \subseteq \xi_1$. By using a similar approach to case (a), it is easy to prove that $\vdash \omega_1 + \omega_2 = \omega_3$

(c) In another case, let $\gamma_1, \cdots, \gamma_r \in \xi_1 \cap \xi_2$, then, some m and n exist and

$$\xi_1 = \{\alpha_1, \cdots, \alpha_m, \gamma_1, \cdots, \gamma_r\}, \quad \xi_2 = \{\beta_1, \cdots, \beta_m, \gamma_1, \cdots, \gamma_r\}$$

holds. Hence, by lemma 3.3,

$$\vdash \omega_1 = \sum_{j=1}^{m} \alpha_j + \sum_{j=1}^{r} \gamma_j, \quad \vdash \omega_2 = \sum_{j=1}^{n} \beta_j + \sum_{j=1}^{r} \gamma_j, \quad \vdash \omega_3 = \sum_{j=1}^{m} \alpha_j + \sum_{j=1}^{n} \beta_j + \sum_{j=1}^{r} \gamma_j,$$

Therefore $\vdash \omega_1 + \omega_2 = \omega_3$.

From 1 and 2, the lemma clearly holds.

□

LEMMA **3.5**

Let ξ be a finite set of BCREs and ω_1 and ω_2 be in $\psi(\xi)$. Then, there is some $\omega_3 \in \psi(\xi)$ and $\vdash \omega_1 + \omega_2 = \omega_3$ holds.

PROOF

By the definition of ψ, there are some $\Omega_1, \Omega_2 \in 2^\xi$ such that $\omega_1 \in \chi(\Omega_1)$ and $\omega_2 \in \chi(\Omega_2)$ holds. By the Lemma 3.4, some $\omega_3 \in \chi(\Omega_1 \cup \Omega_2)$ exists and $\vdash \omega_1 + \omega_2 = \omega_3$ holds. Be reminded that $\Omega_1 \cup \Omega_2 \in 2^\xi$ because both $\Omega_1, \Omega_2 \in 2^\xi$ by a property of power sets. According to the definition of ψ, $\chi(\Omega) \in \psi(\xi)$ holds for all $\Omega \in 2^\xi$. Therefore, $\omega_3 \in \psi(\xi)$. □

LEMMA **3.6**

Assuming that ξ is a finite set of BCREs on σ. Suppose that for all $\alpha \in \xi$ and $\beta \in \psi(\xi)$ some $x_1, x_2, \ldots, x_r \in \sigma$ exist such that

$$\vdash \alpha = \sum_{j=1}^{r} x_j \beta + \delta(\alpha)$$

holds. Then, for all $\alpha' \in \psi(\xi)$ some $\beta' \in \psi(\xi)$ exists and

$$\vdash \alpha' = \sum_{j=1}^{r} x_j \beta' + \delta(\alpha)$$

PROOF

Assume that for all $\alpha \in \xi$, some $\beta \in \psi(\xi)$ and $x_1, x_2, \ldots, x_r \in \sigma$ exist and

$$\vdash \alpha = \sum_{j=1}^{r} x_j \beta + \delta(\alpha)$$

holds, and assume that $\alpha' \in \psi(\xi)$. By the definition of ψ, there are some $\alpha_1, \ldots, \alpha_n \in \xi$ such that $\alpha' \equiv \sum_{i=1}^{n} \alpha_i$ holds. By the assumption, some β_i exists for all $\alpha_i (1 \leq i \leq n)$ that satisfies $\vdash \alpha_i = \sum_{j=1}^{r} x_j \beta_i + \delta(\alpha_i)$. Thus,

$$\vdash \alpha' = \sum_{i=1}^{n} \left(\sum_{j=1}^{r} x_j \beta_i + \delta(\alpha_i) \right) = \sum_{i=1}^{n} \sum_{j=1}^{r} x_j \beta_i + \sum_{i=1}^{n} \delta(\alpha_i)$$

Be reminded that δ is defined as $\perp$ or $\perp^*$, $\vdash \sum_{i=1}^{n} \delta(\alpha_i) = \perp \; or \perp^*$ holds. Hence, we can replace $\sum_{i=1}^{n} \delta(\alpha_i)$ with $\delta(\alpha')$. Therefore,

$$\vdash \alpha' = \sum_{i=1}^{n} \left(\sum_{j=1}^{r} x_j \beta_i \right) + \delta(\alpha') = \sum_{j=1}^{r} x_j \left(\sum_{i=1}^{n} \beta_i \right) + \delta(\alpha')$$

holds. Since $\beta_i \in \psi(\xi)$ holds for all i, by the Lemma 3.4, There is some $\beta' \in \psi(\xi)$ and $\vdash \sum_{i=1}^{n} \beta_i = \beta'$ holds. Thus,

$$\vdash \alpha' = \sum_{j=1}^{r} x_j \beta' + \delta(\alpha')$$

□

LEMMA **3.7**

Suppose that $\alpha_1, \cdots, \alpha_r$ and $\beta_1, \cdots, \beta_r$ are BCREs on σ, $s \in \subseteq$ and $x_1, \cdots, x_r \in \sigma$. Then, the following holds.

$$\vdash \sum_{j=1}^{r} x_j \alpha_j [\, s \,] \sum_{j=1}^{r} x_j \beta_j = \sum_{j=1}^{r} x_j \gamma_j$$

where $\gamma_j \in \psi(\xi_0)$ for $1 \leq j \leq r$ and, ξ_0 is a finite set defined as follows

$$\begin{aligned} \xi_0 &= \{\alpha_j [\, s \,] \beta_k \mid 1 \leq j \leq r, 1 \leq k \leq r, s \in 2^\sigma\} \\ &\cup \{\alpha_j [\, s \,] x_k \beta_k \mid 1 \leq j \leq r, 1 \leq k \leq r, s \in 2^\sigma\} \\ &\cup \{x_j \alpha_j [\, s \,] \beta_k \mid 1 \leq j \leq r, 1 \leq k \leq r, s \in 2^\sigma\} \\ &\cup \{\perp\} \end{aligned}$$

PROOF

By the axiom C_4 and C_5, some s_{jk} exists for all $j = 1, \cdots, r$ and $k = 1, \cdots, r$ such that

$$\vdash \sum_{j=1}^{r} x_j \alpha_j [\, s \,] \sum_{j=1}^{r} x_j \beta_j = \sum_{j=1}^{r} \sum_{k=1}^{r} (x_j \alpha_j [\, s_{jk} \,] x_k \beta_k)$$

By the axiom C_4, some s'_{jk} exists for all $j = 1, \cdots, r$ and $k = 1, \cdots, r$ such that

$$\vdash x_j\alpha_j\,[\,s_{jk}\,]\,x_k\beta_k = \begin{cases} x_j(\alpha_j\,[\,s'_{jk}\,]\,\beta_k) & or \\ x_j(\alpha_j\,[\,s'_{jk}\,]\,x_k\beta_k) & or \\ x_k(x_j\alpha_j\,[\,s'_{jk}\,]\,\beta_k) & or \\ x_j(\alpha_j\,[\,s'_{jk}\,]\,x_k\beta_k) + x_k(x_j\alpha_j\,[\,s'_{jk}\,]\,\beta_k) & or \\ \bot \end{cases}$$

By the axiom A_8 and A_9, any terms including $\bot$ can be added. Thus,

$$\vdash x_j\alpha_j\,[\,s_{jk}\,]\,x_k\beta_k = \begin{cases} x_j(\alpha_j\,[\,s'_{jk}\,]\,\beta_k) + x_k\bot & or \\ x_j(\alpha_j\,[\,s'_{jk}\,]\,x_k\beta_k) + x_k\bot & or \\ x_j\bot + x_k(x_j\alpha_j\,[\,s'_{jk}\,]\,\beta_k) & or \\ x_j(\alpha_j\,[\,s'_{jk}\,]\,x_k\beta_k) + x_k(x_j\alpha_j\,[\,s'_{jk}\,]\,\beta_k) & or \\ x_j\bot + x_k\bot \end{cases}$$

Therefore, some $\mu_{jk}, \nu_{jk} \in \xi_0$ exist for all $j = 1, \cdots, r$ and $k = 1, \cdots$ such that

$$\vdash x_j\alpha_j\,[\,s_{jk}\,]\,x_k\beta_k = x_j\mu_{jk} + x_k\nu_{jk}$$

holds. Hence,

$$\vdash \sum_{j=1}^{r}\sum_{k=1}^{r}(x_j\alpha_j\,[\,s\,]\,x_k\beta_k) = \sum_{j=1}^{r}\sum_{k=1}^{r}(x_j\mu_{jk} + x_k\nu_{jk}) = \sum_{j=1}^{r}x_j\sum_{k=1}^{r}(\mu_{jk} + \nu_{jk})$$

By the Lemma 3.2, $\mu_{jk}, \nu_{jk} \in \psi(\xi_0)$ holds since $\mu_{jk}, \nu_{jk} \in \xi_0$. Then, by the Lemma 3.4, $\gamma_{jk} \in \psi(\xi_0)$ exists for all $j = 1, \cdots, r$ and $k = 1, \cdots, r$ such that $\vdash \mu_{jk} + \nu_{jk} = \gamma_{jk}$. Hence,

$$\vdash \sum_{j=1}^{r}x_j\sum_{k=1}^{r}(\mu_{jk} + \nu_{jk}) = \sum_{j=1}^{r}x_j\sum_{k=1}^{r}\gamma_{jk}$$

therefore,

$$\vdash \sum_{j=1}^{r}x_j\alpha_j\,[\,s\,]\sum_{j=1}^{r}x_j\beta_j = \sum_{j=1}^{r}x_j\gamma_j$$

where $\gamma_j \equiv \sum_{k=1}^{r}\gamma_{jk}$. □

3.3.2 Equationally characterized

DEFINITION **3.3**

It is stated that the expression α is equationally characterised if there is a finite number of the symbol $x_1, \cdots, x_r$ and the expression $\alpha_1, \cdots, \alpha_n$ such that $\alpha \equiv \alpha_1$ and,

$$\vdash \alpha_i = \sum_{j=1}^{r} x_j \alpha_{ij} \ + \ \delta(\alpha_i) \tag{3.1}$$

where $\delta(\alpha_i) \equiv \bot$ or $\delta(\alpha_i) \equiv \bot^*$ for all i and, for each i and j there is some k $(1 \leq k \leq n)$ such that $\alpha_{ij} \equiv \alpha_i$.

THEOREM **3.2**

All BCREs are equationally characterized.

PROOF

We will prove this lemma by induction of the syntax of BCREs. The first step of the proof is for the base case. By A_6 to A_9,

$$\begin{aligned} \vdash \bot &= \sum_{j=1}^{r} x_j \bot + \bot \\ \vdash x_i &= x_1 \bot + \cdots + x_i \bot^* + \cdots + x_r + \bot \quad (1 \leq i \leq r) \\ \vdash \bot^* &= \sum_{j=1}^{r} x_j \bot + \bot^* \end{aligned}$$

Therefore, $\bot, \bot^*$ and x_i $(i = 1, \cdots, r)$ are equationally characterized.

The next step is the induction. Assume that α and β are BCREs and equationally characterized. That is, assume that $\alpha_1, ..., \alpha_m$ and $\beta_1, ..., \beta_n$ exist for some finite numbers m and n. $\alpha \equiv \alpha_1$ and $\beta \equiv \beta_1$. Then,

$$\vdash \alpha_u = \sum_{j=1}^{r} x_j \alpha_{uj} + \delta(\alpha_u) \tag{3.2}$$

$$\vdash \beta_v = \sum_{j=1}^{r} x_j \beta_{vj} + \delta(\beta_v) \tag{3.3}$$

holds for all $u = 1, \cdots, m$ and $v = 1, \cdots, n$ where $\alpha_{uj} \equiv \alpha_u$ and $\beta_{vj} \equiv \beta_v$ for all $j = 1, \cdots, r$.

Then, we prove that $\alpha + \beta$, $\alpha\beta$, α^*, $\alpha \,[\, s \,]\, \beta$ and $\alpha \| \beta$ are equationally characterized.

(1) PROOF FOR $\alpha + \beta$

Let ξ be a finite set of BCREs as follows.

$$\xi = \{\alpha_u + \beta_v \mid 1 \leq u \leq n, 1 \leq v \leq m\}$$

Assume that $\omega \in \xi$, then by (3.2) and (3.3), some u and v exist such that

$$\vdash \omega = \sum_{j=1}^{r} x_j(\alpha_{uj} + \beta_{vj}) + \delta(\alpha_u) + \delta(\beta_v)$$

Because we have

$$\vdash \bot + \bot = \bot, \quad \vdash \bot + \bot^* = \bot^* + \bot = \bot, \quad \vdash \bot^* + \bot^* = \bot^*$$

we obtain for all $\omega \in \xi$,

$$\vdash \omega = \sum_{j=1}^{r} x_j(\alpha_{uj} + \beta_{vj}) + \delta(\omega)$$

where $\delta(\omega) \equiv \bot$ or $\delta(\omega) \equiv \bot^*$ and all of the expressions $\alpha_{uj} + \beta_{vj}$ are in ξ. Since $\alpha + \beta \equiv \alpha_1 + \beta_1 \in \xi$, this implies that $\alpha + \beta$ is equationally characterized.

(2) PROOF FOR $\alpha\beta$

Let ξ be a set of regular expressions.

$$\xi = \{\alpha_u\beta + \beta_{v_1} + \cdots + \beta_{v_h} \mid 1 \leq u \leq n, 0 \leq h, 1 \leq v_1 < v_2 < \cdots < v_h \leq m\}$$

Assume that $\omega \in \xi$, that is, there is some u, h and $v_1, \cdots, v_h$ such that $\omega \equiv \alpha_u\beta + \sum_{i=1}^{h} \beta_{v_i}$. Since α_u and β_{v_i} are equationally characterized, by A_3 to A_6,

$$\begin{aligned} \vdash \omega &= \left(\sum_{j=1}^{r} x_j\alpha_{uj} + \delta(\alpha_u)\right)\beta + \sum_{i=1}^{h}\left(\sum_{j=1}^{r} x_j\beta_{v_ij} + \delta(\beta_{v_i})\right) \\ &= \sum_{j=1}^{r} x_j\left(\alpha_{uj}\beta + \sum_{i=1}^{h}\beta_{v_ij}\right) + \delta(\alpha_u)\beta + \sum_{i=1}^{h}\delta(\beta_{v_i}) \end{aligned} \tag{3.4}$$

where $\delta(\alpha_u) \equiv \bot$ or $\bot^*$, and $\delta(\beta_{v_i}) \equiv \bot$ or $\bot^*$ for all i. Suppose first that $\delta(\alpha_u) \equiv \bot$. Then $\delta(\alpha_u)\beta = \bot$ holds. Hence by (3.4) and A_9,

$$\vdash \omega = \sum_{j=1}^{r} x_j\left(\alpha_{uj}\beta + \sum_{i=1}^{h}\beta_{v_ij}\right) + \delta(\omega) \tag{3.5}$$

where $\delta(\omega) \equiv \bot$ or $\bot^*$. On the other hand, if $\delta(\alpha_u) \equiv \bot^*$, by (3.4),

$$= \sum_{j=1}^{r} x_j\left(\alpha_{uj}\beta + \sum_{i=1}^{h}\beta_{v_ij}\right) + \sum_{j=1}^{r} x_j\beta_{1j} + \delta(\beta_1) + \sum_{i=1}^{h}\delta(\beta_{v_i})$$

Therefore,

$$\vdash \omega = \sum_{j=1}^{r} x_j\left(\alpha_{uj}\beta + \beta_{1j} + \sum_{i=1}^{h}\beta_{v_ij}\right) + \delta(\omega) \tag{3.6}$$

where $\delta(\omega) \equiv \perp$ or $\perp^*$. (3.5) and (3.6) implies that there is some $\omega' \in \xi$ and $\vdash \omega = \sum_{j=1}^{r} x_j \omega' + \delta(\omega)$ holds for all $\omega \in \xi$. Since $\alpha\beta \equiv \alpha_1\beta \in \xi$, it is concluded that $\alpha\beta$ is equationally characterized. □

(3) PROOF FOR α^*

Since $\alpha \equiv \alpha_1$ is equationally characterized,

$$\vdash \alpha = \sum_{j=1}^{r} x_j \alpha_{1j} + \delta(\alpha)$$

Hence by A_8 or A_{11},

$$\vdash \alpha^* = \left(\sum_{j=1}^{r} x_j \alpha_{1j} \right)^*$$

Then by A_{10},

$$\vdash \alpha^* = \left(\sum_{j=1}^{r} x_j \alpha_{1j} \right) \left(\sum_{j=1}^{r} x_j \alpha_{1j} \right)^* + \perp^* = \sum_{j=1}^{r} x_j \alpha_{1j} \alpha^* + \perp^* \tag{3.7}$$

Let ξ be a set of regular expressions.

$$\xi = \{(\alpha_{u_1} + \cdots + \alpha_{u_h})\alpha^* \mid 1 \leq h, 1 \leq u_1 < u_2 < \cdots < u_h \leq n\}$$

Assume that $\omega \in \xi$. Then some h exist such that $\omega \equiv \left(\sum_{i=1}^{h} \alpha_{u_i} \right) \alpha^*$. Suppose that $\sum_{i=1}^{h} \alpha_{u_i}$ does not possess an e.w.p. Then,

$$\vdash \omega = \sum_{i=1}^{h} \left(\sum_{j=1}^{r} x_j \alpha_{u_i j} + \perp \right) \alpha^* = \sum_{j=1}^{r} x_j \left(\sum_{i=1}^{h} \alpha_{u_i j} \alpha^* \right) + \perp \tag{3.8}$$

If $\sum_{i=1}^{h} \alpha_{u_i}$ possesses an e.w.p., we obtain

$$\begin{aligned} \vdash \omega &= \sum_{j=1}^{r} x_j \sum_{i=1}^{h} \alpha_{u_i j} \alpha^* + \alpha^* \\ &= \sum_{j=1}^{r} x_j \sum_{i=1}^{h} \alpha_{u_i j} \alpha^* + \sum_{j=1}^{r} x_j \alpha_{1j} \alpha^* + \perp^* \\ &= \sum_{j=1}^{r} x_j \left(\alpha_{1j} + \sum_{i=1}^{h} \alpha_{u_i j} \right) \alpha^* + \perp^* \end{aligned} \tag{3.9}$$

By (3.7), (3.8) and (3.9), it follows that for all $\omega \in \xi \cup \{\alpha^*\}$, some $\omega' \in \xi \cup \{\alpha^*\}$ exists and

$$\omega = \sum_{j=1}^{r} x_j \omega' + \delta(\omega)$$

where $\delta(\omega) \equiv \perp$ or $\perp^*$. Hence it is concluded that α^* is equationally characterized. □

(4) PROOF FOR $\alpha\,[\,s\,]\,\beta$

$\xi_1, ..., \xi_5$ are finite sets of BCREs defined as follows.

$$\begin{aligned}
\xi_1 &\equiv \{\alpha_u\,[\,s\,]\,\beta_v \mid 1 \le u \le m, 1 \le v \le n \; and \; s \in 2^\sigma\} \\
\xi_2 &\equiv \{x_j\alpha_u\,[\,s\,]\,\beta_v \mid 1 \le u \le m, 1 \le v \le n, 1 \le j \le r \; and \; s \in 2^\sigma\} \\
\xi_3 &\equiv \{\alpha_u\,[\,s\,]\,x_j\beta_v \mid 1 \le u \le m, 1 \le v \le n, 1 \le j \le r \; and \; s \in 2^\sigma\} \\
\xi_4 &\equiv \{\alpha_u\,[\,s\,]\,\perp^* \mid 1 \le u \le m, s \in 2^\sigma\} \\
\xi_5 &\equiv \{\perp^*\,[\,s\,]\,\beta_v \mid 1 \le v \le n, s \in 2^\sigma\} \\
\xi_6 &\equiv \{\alpha_u \mid 1 \le u \le m\} \\
\xi_7 &\equiv \{\beta_v \mid 1 \le v \le n\}
\end{aligned}$$

where σ is a finite set of symbols.

First, we prove the following: if $\omega \in \xi_i$ $(1 \le i \le 5)$, then some $\gamma_1, \cdots, \gamma_r \in \psi(\xi_1 \cup \cdots \cup \xi_5)$ exist and

$$\vdash \omega = \sum_{j=1}^{r} x_j\gamma_j + \delta(\omega)$$

Assume that $\omega \in \xi_1$, by the definition of ξ_1, some u, v and s exists such that $\omega \equiv \alpha_u\,[\,s\,]\,\beta_v$ holds. Since α and β are equationally characterized, α_u and β_v are also equationally characterizedfor every u and v. Hence, the following holds.

$$\vdash \alpha_u\,[\,s\,]\,\beta_v = \left(\sum_{j=1}^{r} x_j\alpha_{uj} + \delta(\alpha_u)\right)\,[\,s\,]\,\left(\sum_{j=1}^{r} x_j\beta_{vj} + \delta(\beta_v)\right)$$

By the axioms C_4 and C_5, the following equation can be derived.

$$\vdash \alpha_u\,[\,s\,]\,\beta_v = \pi_1 + \pi_2 + \pi_3 + \pi_4$$

where

$$\begin{aligned}
\pi_1 &= \sum_{j=1}^{r} x_j\alpha_{uj}\,[\,s_1\,]\,\sum_{j=1}^{r} x_j\beta_{vj} \\
\pi_2 &= \sum_{j=1}^{r} x_j\alpha_{uj}\,[\,s_2\,]\,\delta(\beta_v) \\
\pi_3 &= \delta(\alpha_u)\,[\,s_3\,]\,\sum_{j=1}^{r} x_j\beta_{vj} \\
\pi_4 &= \delta(\alpha_u)\,[\,s_4\,]\,\delta(\beta_v)
\end{aligned}$$

and $s_1, \cdots, s_4 \in 2^\sigma$. Then, for π_1, by the Lemma 3.7, some $\gamma_{1j} \in \xi_0$ exists (Assuming that ξ_0 here is the same as the one in the Lemma 3.7) and it follows that

$$\vdash \pi_1 \equiv \sum_{j=1}^{r} x_j\alpha_{uj}\,[\,s_1\,]\,\sum_{j=1}^{r} x_j\beta_{vj} = \sum_{j=1}^{r} x_j\gamma_{1j} \tag{3.10}$$

According to the definition of $\xi_0, \cdots, \xi_3$, it clearly follows that $\xi_0 = \xi_1 \cup \xi_2 \cup \xi_3$. Hence, by the Lemma 3.2, $\gamma_{1j} \in \psi(\xi_1 \cup \xi_2 \cup \xi_3)$.

For π_2, by the axiom C_1 to C_3,

$$\vdash \pi_2 \equiv \sum_{j=1}^{r} x_j \alpha_{uj} [\, s_2 \,] \delta(\beta_v) = \begin{cases} \bot & if \quad \delta(\beta_v) \equiv \bot \\ \sum_{j=1}^{r} x_j \alpha'_{uj} & if \quad \delta(\beta_v) \equiv \bot^* \end{cases}$$

where $\alpha'_{uj} \equiv \bot$ if $x_j \in s_2$, otherwise $\alpha'_{uj} \equiv \alpha_{uj} [\, s \,] \bot^*$. By the axioms A_6 and A_8,

$$\vdash \bot = \sum_{j=1}^{r} \bot = \sum_{j=1}^{r} x_j \bot$$

Hence,

$$\vdash \pi_2 \quad = \quad \sum_{j=1}^{r} x_j \gamma_{2j} \tag{3.11}$$

where $\gamma_{2j} \in \xi_4 \cup \{\bot\}$ holds for all $j = 1, \cdots, r$. The following also holds in the similar way to (3.11):

$$\vdash \pi_3 = \sum_{j=1}^{r} x_j \gamma_{3j} \tag{3.12}$$

where for all $j = 1, \cdots, r$, $\gamma_{3j} \equiv \bot$ or $\gamma_{3j} \in \xi_5 \cup \{\bot\}$.

For π_4, by the axiom C_1-C_3,

$$\vdash \pi_4 = \delta(\alpha_u) [\, s_4 \,] \delta(\beta_v) = \begin{cases} \bot^* & if \qquad \delta(\alpha_u) \equiv \delta(\beta_v) \equiv \bot^* \\ \bot & otherwise \end{cases}$$

Hence, δ can be defined for all $u = 1, \cdots, r$, $v = 1, \cdots, r$ and s_4 as follows.

$$\vdash \pi_4 = \delta(\alpha_u [\, s_4 \,] \beta_v)$$

s_4 can be replaced with s since s_4 does not effect the definition of δ. Hence,

$$\vdash \pi_4 = \delta(\alpha_u [\, s \,] \beta_v) \tag{3.13}$$

By (3.10), (3.11), (3.12) and (3.13),

$$\begin{aligned} \vdash \alpha_u [\, s \,] \beta_v &= \pi_1 + \pi_2 + \pi_3 + \pi_4 \\ &= \sum_{j=1}^{r} x_j \gamma_{1j} + \sum_{j=1}^{r} x_j \gamma_{2j} + \sum_{j=1}^{r} x_j \gamma_{3j} + \delta(\alpha_u [\, s \,] \beta_v) \\ &= \sum_{j=1}^{r} x_j (\gamma_{1j} + \gamma_{2j} + \gamma_{3j}) + \delta(\alpha_u [\, s \,] \beta_v) \end{aligned}$$

where $\delta(\alpha_u\,[\,s\,]\,\beta_v)$ is $\perp^*$ if $\delta(\alpha_u) \equiv \delta(\beta_v) \equiv \perp^*$. Otherwise, $\perp$. Since $\gamma_{1j}, \gamma_{2j}, \gamma_{3j} \in \psi(\xi_1 \cup \cdots \cup \xi_5 \cup \{\perp\})$ and by the Lemma 3.5, There is some $\gamma_j \in \psi(\xi_1 \cup \ldots \cup \xi_5 \cup \{\perp\})$ such that

$$\vdash \gamma_{1j} + \gamma_{2j} + \gamma_{3j} = \gamma_j$$

holds for all $j = 1, \cdots, r$. Hence,

$$\vdash \omega = \sum_{j=1}^{r} x_j \gamma_j + \delta(\omega) \tag{3.14}$$

Assume that $\omega \in \xi_2$, that is, $\omega \equiv x_i \alpha_u\,[\,s\,]\,\beta_v \in \xi_2$. Since β_v is equationally characterized, it follows that

$$\vdash \omega = x_i \alpha_u\,[\,s\,]\,\left(\sum_{j=1}^{r} x_j \beta_{vj} + \delta(\beta_v)\right)$$

By the axiom C_4, there are some s_1 and s_2 such that

$$\vdash x_i \alpha_u\,[\,s\,]\,\left(\sum_{j=1}^{r} x_j \beta_{vj} + \delta(\beta_v)\right) = \pi_1 + \pi_2$$

where

$$\begin{aligned} \pi_1 &= x_i \alpha_u\,[\,s_1\,] \sum_{j=1}^{r} x_j \beta_{vj} \\ \pi_2 &= x_i \alpha_u\,[\,s_2\,]\,\delta(\beta_v) \end{aligned}$$

For π_1, suppose that $\alpha_{uj} \equiv \alpha_u$ for $j = i$ and $\alpha_{uj} \equiv \perp$ for $j \neq i$.Then,

$$\vdash \pi_1 \equiv x_i \alpha_u\,[\,s_1\,] \sum_{j=1}^{r} x_j \beta_{vj} = \sum_{j=1}^{r} x_j \alpha_{uj}\,[\,s_1\,] \sum_{j=1}^{r} x_j \beta_{vj}$$

By the Lemma 3.7, some $\gamma_{1j} \in \xi_0$ exists and

$$\vdash \sum_{j=1}^{r} x_j \alpha_{uj}\,[\,s_1\,] \sum_{j=1}^{r} x_j \beta_{vj} = \sum_{j=1}^{r} x_j \gamma_{1j} \tag{3.15}$$

holds. e reminded that $\xi_0 = \xi_1 \cup \xi_2 \cup \xi_3$, and by the Lemma 3.2,

$$\gamma_{1j} \in \psi(\xi_1 \cup \ldots \cup \xi_3)$$

holds.

For π_2, by using a similar process in the proof for $\omega \in \xi_1$, $\gamma_{2j} \in \psi(\xi_4) \cup \{\perp\}$ exists for all $j = 1, \cdots, r$ such that

$$\vdash \pi_2 = \sum_{j=1}^{r} x_j \gamma_{2j} \tag{3.16}$$

By (3.15) and (3.16),

$$\vdash \omega = \sum_{j=1}^{r} x_j \gamma_{1j} + \sum_{j=1}^{r} x_j \gamma_{2j} = \sum_{j=1}^{r} x_j (\gamma_{1j} + \gamma_{2j})$$

Since $\gamma_{1j} \in \psi(\xi_1 \cup \xi_2 \cup \xi_3)$ and $\gamma_{2j} \in \psi(\xi_4) \cup \{\bot\}$, it follows that $\gamma_{1j}, \gamma_{2j} \in \psi(\xi_1 \cup \cdots \cup \xi_4 \cup \{\bot\})$. By the Lemma 3.3, $\gamma_j \in \psi(\xi_1 \cup \cdots \cup \xi_4 \cup \{\bot\})$ exists for all $j = 1, \cdots, r$ such that $\vdash \gamma_{1j} + \gamma_{2j} = \gamma_j$ holds. Thus $\gamma_j \in \psi(\xi_1 \cup \cdots \cup \xi_4 \cup \{\bot\})$ exists such that

$$\vdash \omega = \sum_{j=1}^{r} x_j \gamma_j$$

for every $\omega \in \xi_2$. By the axiom A_9,

$$\vdash \omega = \sum_{j=1}^{r} x_j \gamma_j + \delta(\omega)$$

where $\delta(\omega) \equiv \bot$.

Assume that $\omega \in \xi_3$. Then, for all $\omega \in \xi_3$, some $\gamma_j \in \psi(\xi_2 \cup \xi_3 \cup \xi_5 \cup \{\bot\})$ exist and $\vdash \omega = \sum_{j=1}^{r} x_j \gamma_j + \delta(\omega)$ also holds where $\delta(\omega) \equiv \bot$. This can be proved following the same approach as the proof for $\omega \in \xi_2$.

Next, let us consider $\omega \in \xi_4$. Since α_u is equationally characterized, we have

$$\vdash \omega = \sum_{j=1}^{r} x_j \alpha_{uj} + \delta(\omega) \, [\, s \,] \, \bot^* = \left(\sum_{j=1}^{r} x_j \alpha_{uj} \, [\, s \,] \, \bot^* \right) + \delta(\omega) \, [\, s \,] \, \bot^*$$

According to C_1-C_3,

$$\begin{aligned} \vdash \sum_{j=1}^{r} x_j \alpha_{uj} \, [\, s \,] \, \bot^* &= \sum_{j=1}^{r} x_j \alpha'_{uj} \\ \vdash \delta(\omega) \, [\, s \,] \, \bot^* &= \bot^* \; or \; \bot \end{aligned}$$

where $\alpha'_{uj} \equiv \bot$ if $x_j \in s_2$, otherwise $\alpha'_{uj} \equiv \alpha_{uj} \, [\, s \,] \, \bot^*$. Hence, it follows that some $\gamma_j \in \xi_4 \cup \{\bot\}$ exists and $\omega = \sum_{j=1}^{r} x_j \gamma_j + \delta(\omega)$ holds. In a similar way it is easy to show that the same equation follows for $\omega \in \xi_5$ and $\gamma_j \in \xi_5$.

Finally, in the case of $\omega \in \xi_6, \xi_7$, since α_u and β_v are equationally characterized, it is clear that

$$\vdash \alpha_u = \sum_{j=1}^{r} x_j \alpha_{uj} + \delta(\alpha_u), \quad \vdash \beta_u = \sum_{j=1}^{r} x_j \beta_{uj} + \delta(\beta_u)$$

where $\alpha_{uj} \in \xi_6$ and $\beta_{uj} \in \xi_7$.

Thus, according to the results of the discussion above for $\omega \in \xi_1 \cdots \xi_7$, it is proved that $\gamma_j \in \psi(\xi_1 \cup \cdots \cup \xi_7 \cup \{\perp\})$ exists such that

$$\vdash \omega = \sum_{j=1}^{r} x_j \gamma_j + \delta(\omega)$$

for $\omega \in \xi_1 \cup \cdots \cup \xi_7 \cup \{\perp\}$. By the Lemma 3.6, this also follows for all $\omega \in \psi(\xi_1 \cup \cdots \cup \xi_7 \cup \{\perp\})$. Therefore, every $\omega \in \psi(\xi_1 \cup \cdots \cup \xi_7 \cup \{\perp\})$ can be characterized with finite number of equations. Thus, all $\omega \in \psi(\xi_1 \cup \cdots \cup \xi_7 \cup \{\perp\})$ are equationally characterized. By $\alpha\,[\,s\,]\,\beta \equiv \alpha_1\,[\,s\,]\,\beta_1 \in \xi_1$ and $\xi_1 \subseteq \psi(\xi_1 \cup \cdots \cup \xi_7 \cup \{\perp\})$, it follows that $\alpha\,[\,s\,]\,\beta \in \psi(\xi_1 \cup \cdots \cup \xi_7 \cup \{\perp\})$. Hence, $\alpha\,[\,s\,]\,\beta$ is equationally characterized.

(5) PROOF FOR $\alpha \| \beta$

Assume that α and β are equationally characterized. Then (3.2) and (3.3) hold and, $x_1, \cdots, x_r$ exist for some $\alpha_1, \cdots, \alpha_m$, $\beta_1, \cdots, \beta_n$ such that

$$\vdash \alpha \| \beta \;=\; \left(\sum_{j=1}^{r} x_j \alpha_{uj} + \delta(\alpha_u) \right) \| \left(\sum_{j=1}^{r} x_j \beta_{vj} + \delta(\beta_v) \right)$$

Given that

$$\begin{aligned}
\xi_1 &\equiv \{\alpha_u \| \beta_v | 1 \le u \le m, 1 \le v \le n\} \\
\xi_2 &\equiv \{x_j \alpha_u \| \beta_v | 1 \le u \le m, 1 \le v \le n, 1 \le j \le r\} \\
\xi_3 &\equiv \{\alpha_u \| x_j \beta_v | 1 \le u \le m, 1 \le v \le n, 1 \le j \le r\}
\end{aligned}$$

Then, in the same way as the proof for $\alpha\,[\,s\,]\,\beta$, the following can be derived for all $\omega \in \xi_1 \cup \xi_2 \cup \xi_3 \cup \{\perp\}$ that $\gamma_j \in \psi(\xi_1 \cup \xi_2 \cup \xi_3 \cup \{\perp\})$ exists such that $\vdash \omega = \sum_{j=1}^{r} x_j \gamma_j + \delta(\omega)$ holds. Therefore, every $\omega \in \psi(\xi_1 \cup \xi_2 \cup \xi_3 \cup \{\perp\})$ are equationally characterized. Since $\alpha \| \beta \equiv \alpha_1 \| \beta_1 \in \psi(\xi_1 \cup \cdots \cup \xi_3 \cup \{\perp\})$, it clearly follows that $\alpha \| \beta$ is equationally characterized.

According to the proofs (1) to (5), the induction is completed and it can be concluded that all BCREs are equationally characterized. □

3.3.3 Proof for Completeness

In the reminder of this section, we prove that $\vdash \alpha = \beta$ is always derivable if α and β are equationally characterized and $\alpha = \beta$ is valid. Since all BCREs are equationally characterized,

the same proof strategy in the completeness theorem for Regular Expression can also be used. The proof shown in this section is based on the proof in A. Salomaa's completeness theorem[6]. We start with the following two lemmas (Lemma 3.8,3.9). Then we prove the completeness theorem (Theorem 3.3).

LEMMA **3.8**

Let n be a natural number and assume that for all $i = 1, \cdots, n$

$$\vdash \alpha_i = \sum_{j=1}^{r} \gamma_{ij}\alpha_j + \gamma_i \tag{3.17}$$

$$\vdash \beta_i = \sum_{j=1}^{r} \gamma_{ij}\beta_j + \gamma_i \tag{3.18}$$

where none of r_{ij} possess an e.w.p. Then, $\vdash \alpha_i = \beta_i$ for all $i = 1, \cdots, n$.

PROOF

The proof is by induction on the number n. If $n = 1$, then (3.17)(3.18) has the form:

$$\vdash \alpha_1 = \gamma_{11}\alpha_1 + \gamma_1, \quad \vdash \beta_1 = \gamma_{11}\beta_1 + \gamma_1$$

Hence By $R2$,

$$\vdash \alpha_1 = (\gamma_{11})^*\gamma_1 = \beta_1$$

Assuming that $2 \leq n$ and that the lemma holds for the numbers $1, \cdots, n-1$. Given that (3.17)(3.18) holds for $i = n$, then it clearly follows that

$$\begin{aligned} \vdash \alpha_n &= \sum_{j=1}^{n-1} \gamma_{nj}\alpha_j + \gamma_{nn}\alpha_n + \gamma_n \\ \vdash \beta_n &= \sum_{j=1}^{n-1} \gamma_{nj}\beta_j + \gamma_{nn}\beta_n + \gamma_n \end{aligned}$$

By applying the rule $R2$,

$$\vdash \alpha_n = (\gamma_{nn})^* \left(\sum_{j=1}^{n-1} \gamma_{nj}\alpha_j + \gamma_n \right) \tag{3.19}$$

$$\vdash \beta_n = (\gamma_{nn})^* \left(\sum_{j=1}^{n-1} \gamma_{nj}\beta_j + \gamma_n \right) \tag{3.20}$$

According to (3.19)(3.20), α_n can be eliminated from (3.17) and, β_n so is from (3.18). Then, for all $i = 1, \cdots, n-1$

$$\begin{aligned} \vdash \alpha_i &= \sum_{j=1}^{n-1} \gamma_{ij}\alpha_j + \gamma_{in}(\gamma_{nn})^* \left(\sum_{j=1}^{n-1} \gamma_{nj}\alpha_j + \gamma_n \right) + \gamma_i \\ \vdash \beta_i &= \sum_{j=1}^{n-1} \gamma_{ij}\beta_j + \gamma_{in}(\gamma_{nn})^* \left(\sum_{j=1}^{n-1} \gamma_{nj}\beta_j + \gamma_n \right) + \gamma_i \end{aligned}$$

By applying the axioms A_3 to A_5, it holds for all $i = 1, \cdots, n-1$ that

$$\begin{aligned} \vdash \alpha_i &= \sum_{j=1}^{n-1} (\gamma_{ij} + \gamma_{in}(\gamma_{nn})^* \gamma_{nj})\alpha_j + \gamma_i' \\ \vdash \beta_i &= \sum_{j=1}^{n-1} (\gamma_{ij} + \gamma_{in}(\gamma_{nn})^* \gamma_{nj})\beta_j + \gamma_i' \end{aligned}$$

where $\gamma_i' \equiv \gamma_i + \gamma_{in}(\gamma_{nn})^* \gamma_n$. It is obvious that $\gamma_{ij} + \gamma_{in}(\gamma_{nn})^* \gamma_{nj}$ does not possess an e.w.p. By the induction hyp.,

$$\vdash \alpha_i = \beta_i$$

it follows for all $i = 1, \cdots, n-1$. Hence, by (3.19) and (3.20),

$$\vdash \alpha_n = \beta_n$$

This completes the induction and also the proof of the lemma. □

LEMMA **3.9**

Assuming that α and β are BCREs, $\alpha = \beta$ is valid and

$$\vdash \alpha = \sum_{j=1}^{r} x_j \alpha_j + \delta(\alpha), \tag{3.21}$$

$$\vdash \beta = \sum_{j=1}^{r} x_j \beta_j + \delta(\beta), \tag{3.22}$$

where $\delta(\alpha) \equiv \perp$ or $\delta(\alpha) \equiv \perp^*$ and, $\delta(\beta) \equiv \perp$ or $\delta(\beta) \equiv \perp^*$. Then $\delta(\alpha) \equiv \delta(\beta)$ and $\alpha_j = \beta_j$ for all $j = 1, \cdots r$.

PROOF

Assume that $\alpha = \beta$ is valid and, (3.21) and (3.22) hold. It is clear that $\sum_{j=1}^{r} x_j \alpha_j$ and $\sum_{j=1}^{r} x_j \beta_j$ do not possess an e.w.p. Hence,

1. If α and β possess an e.w.p, it follows that $\delta(\alpha) \equiv \delta(\beta) \equiv \perp^*$.
2. If α and β do not possess an e.w.p, it follows that $\delta(\alpha) \equiv \delta(\beta) \equiv \perp$.
3. If either α or β possesses an e.w.p, it contradicts that $\alpha = \beta$ is valid.

Therefore, $\delta(\alpha) \equiv \delta(\beta)$ always holds.

Now, assume that some j exists such that $\alpha_j \neq \beta_j$. Then, by the definition of L, the sets $L(x_j \alpha_j)$ for $j = 1, \cdots, r$ are disjoint. In other words, for all $j = 1, \cdots, r$ and $k = 1, \cdots, r$,

$L(x_j\alpha_j) \cap L(x_k\alpha_k) = \phi$ holds if $j \neq k$. The same holds with $L(x_j\beta_j)$. This implies the following:

$$\sum_{j=1}^{r} x_j\alpha_j \neq \sum_{j=1}^{r} x_j\beta_j$$

This contradicts that $\alpha = \beta$ is valid. Hence, $\alpha_j = \beta_j$ for all $j = 1, \cdots, r$. □

Now, we can show the completeness of the axiom system F_c. The following proof is similar to A.Salomaa's completeness proof [6].

THEOREM **3.3**

F_c is complete.

PROOF

Assume that α and β are BCREs and $\alpha = \beta$ is valid. By the theorem 3.2 α and β are equationally characterized. Hence, for some $\alpha_1, \cdots, \alpha_m$ and $\beta_1, \cdots, \beta_n$, (3.2) and (3.3) hold where $\alpha \equiv \alpha_1$ and $\beta \equiv \beta_1$. Then, by the lemma 3.9,

$$\begin{aligned} \vdash \alpha = \alpha_1 &= \sum_{j=1}^{r} x_j\alpha_j^1 + \delta(\alpha) \\ \vdash \beta = \beta_1 &= \sum_{j=1}^{r} x_j\beta_j^1 + \delta(\alpha) \end{aligned}$$

where for all α_j^1 and β_j^1, some k and l exist such that $\alpha_j^1 \equiv \alpha_k$ and $\beta_j^1 \equiv \beta_l$. By the lemma 3.9 again, for all $j = 1, \cdots, r$,

$$\begin{aligned} \vdash \alpha_j^1 &= \sum_{j=1}^{r} x_j\alpha_j^2 + \delta(\alpha_j^1), \\ \vdash \beta_j^1 &= \sum_{j=1}^{r} x_j\beta_j^2 + \delta(\alpha_j^1), \end{aligned}$$

where for all α_j^2 and β_j^2, some k and l exist such that $\alpha_j^2 \equiv \alpha_k$ and $\beta_j^2 \equiv \beta_l$. This procedure is carried on until no new pair of α_j^k and β_j^k appears. Thus, there is some $u \leq mn$ and α^i and β^i for $i = 1, \cdots, u$ can be defined as follows.

$$\begin{aligned} \vdash \alpha^i &= \sum_{j=1}^{r} x_j\alpha_j^i + \gamma_i \\ \vdash \beta^i &= \sum_{j=1}^{r} x_j\beta_j^i + \gamma_i \end{aligned}$$

where for all α_j^i and β_j^i, some $k \leq u$ and $l \leq u$ exist such that $\alpha_j^i \equiv \alpha^k$ and $\beta_j^i \equiv \beta^l$ holds. Since we have $\vdash \alpha\perp = \perp$, for all $i = 1, \cdots, u$,

$$\vdash \alpha^i = \sum_{j=1}^{u} \gamma_{ij}\alpha^j + \gamma_i,$$

$$\vdash \beta^i = \sum_{j=1}^{u} \gamma_{ij}\beta^j + \gamma_i,$$

where for each i and j, either $\gamma_{ij} \equiv \perp$ or $\gamma_{ij} \equiv x_{j_1} + \cdots + x_{j_v}$ for some $1 \leq v$ and $1 \leq j_1 < \cdots < j_v \leq r$. Thus, none of the expressions γ_{ij} possess an e.w.p. This implies, by the lemma 3.8,

$$\vdash \alpha^i = \beta^i \quad (i = 1, \cdots, u)$$

In particular, we have $\vdash \alpha = \beta$ for $i = 1$. Thus, the theorem 3.3 follows. □

The next theorem follows immediately from the completeness of F_c.

THEOREM **3.4**

If α is a BCRE, then there is at least one expression β that does not contain the $[\,s\,]$ and $\|$ operators, and $\vdash \alpha = \beta$ follows.

PROOF

It is obvious by the semantics of BCREs that their language class is equivalent to that of regular expressions. Therefore, for every BCRE α there is a regular expression β that does not contain $[\,s\,]$ and $\|$, and $L(\alpha) = L(\beta)$ holds. Then by the completeness of F_c, it is derivable that $\vdash \alpha = \beta$. □

3.4 Theorems

This section describes some important equivalence relationships of BCREs as theorems. We omit the proof here but detailed proofs for each of the theorems can be seen in the thesis appendix .

THEOREM **3.5 (Associative Law)**

Assume that $\sigma(\alpha) \cup \sigma(\beta) \subseteq \sigma(\alpha\,[\ ]\,\beta)$ and $\sigma(\beta) \cup \sigma(\gamma) \subseteq \sigma(\beta\,[\ ]\,\gamma)$ then

$$\alpha\,[\ ]\,(\beta\,[\ ]\,\gamma) = (\alpha\,[\ ]\,\beta)\,[\ ]\,\gamma$$

Note that associative law on the $[\ ]$ operator does not generally follow. For example, $\vdash (a\,[\ ]\,(b\,[\ ]\,(a+b)) = a\|b$ but $\vdash (a\,[\ ]\,b)\,[\ ]\,(a+b) = (a\|b)\,[\ ]\,(a+b) = \perp$

THEOREM **3.6 (Associative Law (2))**

Assume that $s \supseteq (\sigma(\alpha) \cap \sigma(\beta)) \cup (\sigma(\beta) \cap \sigma(\gamma)) \cup (\sigma(\gamma) \cap \sigma(\alpha))$ then

$$\alpha\,[\,s\,]\,(\beta\,[\,s\,]\,\gamma) = (\alpha\,[\,s\,]\,\beta)\,[\,s\,]\,\gamma$$

This theorem means that if a common set of communication symbols s is used over the entire system, then associative law is followed.

THEOREM **3.7 (Operator swapping)**

$$\vdash \alpha \,[\, s \,]\, \beta = \alpha \parallel \beta \; if \; \sigma(\alpha) \cap \sigma(\beta) \cup s = \phi$$

This theorem illustrates the condition where we can exchange the $[\, s \,]$ and $\parallel$ operators. We can do so even if there is no communication symbol between α and β.

THEOREM **3.8 (Extraction)**

$$\vdash (\alpha x \beta) \,[\, s \,]\, (\gamma y \delta) = (\alpha \,[\ ]\, \gamma)(x \beta \,[\, s \,]\, y \delta)$$

where $x, y \in (\sigma(x\beta) \cap \sigma(y\delta)) \cup s$ *and* $(\sigma(\alpha x \beta) \cup s) \cap \sigma(\gamma) = (\sigma(\gamma y \delta) \cup s) \cap \sigma(\alpha) = \phi$

This theorem means that if x and y are communication symbols and, α and γ do not possess communication symbols, then α and γ can be extracted as concurrent threads $(\alpha \,[\ ]\, \gamma)$. Note that $\vdash (\alpha \,[\ ]\, \gamma) = \alpha \| \gamma$ holds by the Theorem 3.7since α and γ has no communication symbols.

3.5 Summary

This chapter presented the axiom system F_c for the language equivalence of BCREs and proved that F_c is sound and complete. As a result of the soundness and completeness, with our system we can prove equivalence for any two expressions if the expressions have the same language; that is, they represent the same behaviour. In the last section, we proved some important theorems in respect to BCREs.

Chapter 4

Modelling Concurrent Systems

In this chapter, we formally define a model of concurrent system in two different ways. One is a concurrent object model that is defined as a set of concurrent objects. The other is a concurrent thread model that is constructed by a global state machine and sets of concurrent threads assigned to a state in the global state machine.

4.1 Concurrent object model

The concurrent object model is a behaviour model which represents composite behaviour of concurrent objects. Assume that there are n number of objects for some $n \geq 1$. Then, the concurrent object model for these objects is defined as follows.

$$O_1 [\, s_1 \,] (O_2 [\, s_2 \,] (\cdots (O_{n-1} [\, s_{n-1} \,] O_n) \cdots)) \tag{4.1}$$

where s_i is a set of communication symbols between O_i and the composition of $O_{i+1}, \cdots, O_n$. Note that if we omit s_i; that is, if $s_i = \phi$ is given, the language of the resulting concurrent object model is the same as that if we define

$$s_i = \sigma(O_i) \cap \sigma(O_{i+1} [\, s_{i+2} \,] (O_{i+2} [\, s_{i+3} \,] (\cdots (O_{n-1} [\, s_{n-1} \,] O_n) \cdots)))$$

Behaviour represented by the concurrent object model is sensitive of the order of object occurrence $O_1, \cdots, O_n$. For example, let us consider three objects; a, b and $a + b$. Then $\vdash a [\ ] (b [\ ] (a + b)) = a [\ ] b = a \| b$ while $\vdash (a + b) [\ ] (a [\ ] b) = (a + b) [\ ] (a \| b) = \bot$.

Now, we show an example model of an auto-locking door system. This system consists of three objects: Card-Reader, Door and Timer as depicted in Figure 4.1,4.2 and 4.3

This model can be formalised as follows.

$$(un.tb.(to + op.te.cl).lo)^* [\ ] (ca.un.lo)^* [\ ] (tb.tb^*.(to + te))^*$$

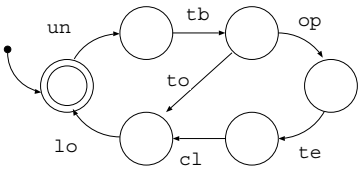


Figure 4.1: Door

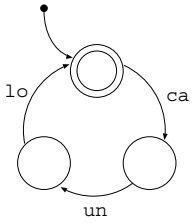


Figure 4.2: Card Reader

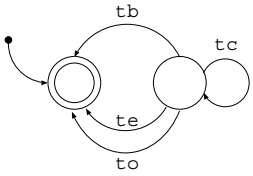


Figure 4.3: Timer

Note that some of the parenthesises for the binding priority of $[\ ]$ are omitted because $[\ ]$ operators in this context satisfy the associative law according to theorem 3.5.

4.2 Concurrent thread model

We begin this section with an intuitive explanation of a concurrent thread model. Then, we provide the definition of a concurrent thread model, which is a model of a concurrent system that focuses on a switching set of concurrent threads according to the transitions of a global state in a system. A concurrent thread model is constructed with a global state machine and sets of threads assigned to each state. Figure 4.4 depicts an example of a concurrent thread model. In this Figure, there is a global state machine with some threads assigned; one or more threads can be assigned to a state. For example, the 'state1' in Figure 4.4 contains two threads that are executed concurrently.

An expression representing a concurrent thread model in Figure 4.4 can be obtained as follows. First, we define a global state machine using BCREs. Although there is no transition labels in the global state machine in Figure 4.4, state identifiers can be used as transition labels and so it is easy to obtain an expression for the global state machine in Figure 4.4 as follows.

$$(state_0.(state_1 + state_2).state_3)^* \tag{4.2}$$

Then we define a state as a BCRE that represents threads that are allocated in each state. It is clearly obtained from Figure 4.4 that

$$\begin{aligned} state_0 &\equiv (a.b) \\ state_1 &\equiv ((c.d) \,\|\, (e.f + g)) \\ state_2 &\equiv (h.i)^* \\ state_3 &\equiv j \end{aligned} \tag{4.3}$$

By substituting (4.3) to (4.2), it is derived that

$$((a.b).((c.d \,\|\, (e.f + g)) + (h.i)^*).j)^*$$

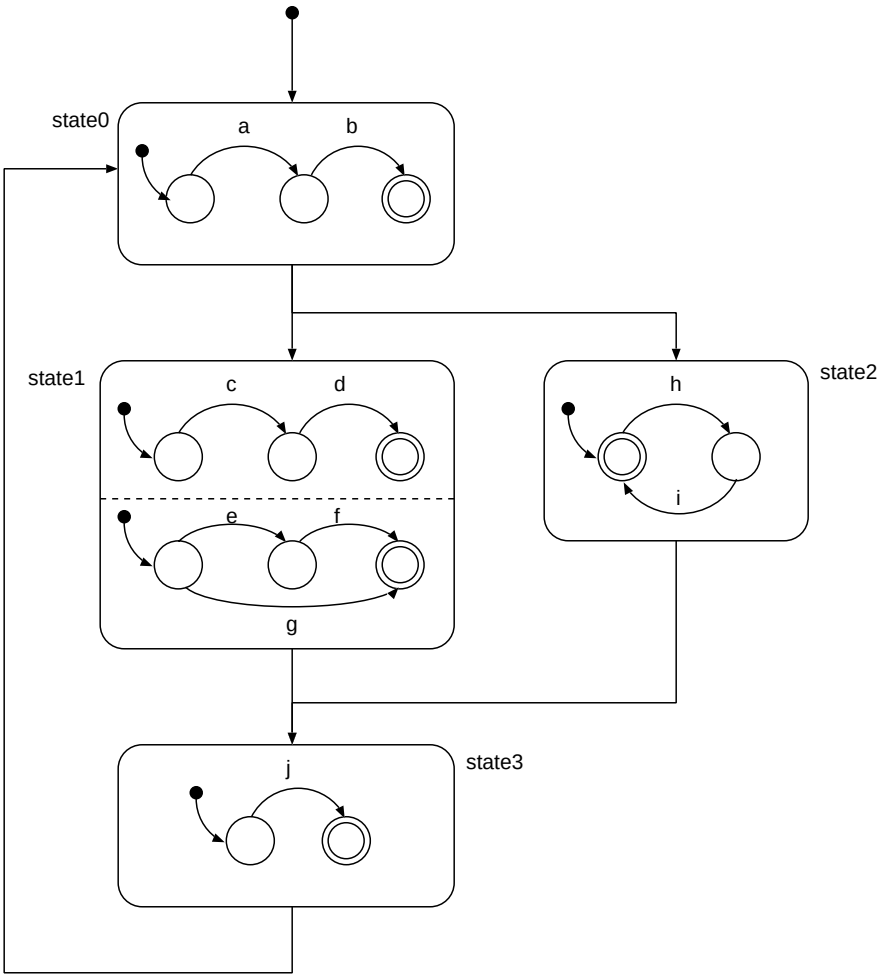


Figure 4.4: Concurrent Thread Model

This is an expression of the model in Figure 4.4.

Syntactically, the concurrent thread model is defined as an expression that has no $[\,s\,]$ operator. However, such a definition with respect only to syntax is over-simplified. We should focus on a concurrent thread model that correctly corresponds to a concurrent object model, and also on a semantic level. Some additions to the definition are needed to illustrate what the correspondence is. The following is a definition of a concurrent thread model that corresponds to a concurrent object model. Informally, a concurrent thread model is an expression that has

1. no $[\,s\,]$ operator
2. no interleaving expression
3. no over-concurrency

An interleaving expression is an expression that contains any interleaving of symbols from two or more different objects. For example, assume that there are two objects defined by the

expressions ab and cd respectively. Then the concurrent object model for these objects are defined as $ab[\ ]cd$. In this case, the corresponding concurrent thread model is exactly $ab||cd$; however there are some other equivalent expressions such as $a(b \parallel cd)+c(ab||d)$, and $ac(b \parallel d)+abcd+cdab+ca(b \parallel d)$. In these expressions, some of concurrent behaviour between ab and cd is serialised. Such sequences are called interleaving expressions. Be reminded that the purpose of focusing on the concurrent thread model is to clarify the essential concurrency in a system. If there is any interleaving expression, it means that some information of the concurrent behaviour is lost. On the other hand, the over-concurrent expression indicates internal concurrency of an object that is not allowed in concurrent object models. For example, there is an object defined with an expression $ab + ba$; by the semantics of BCREs, it is equivalent to $a||b$. However, we should not consider $a||b$ as a concurrent thread model for $ab+ba$ because there is no concurrency in the original $ab+ba$. A concurrent thread model should not contain any excess of concurrency that is not implied in the corresponding concurrent object model.

We now define the concurrent thread model in a formal way. We begin by clarifying which symbols belong to which objects. The following definition is a projection from a symbol to an object to which the symbol belongs.

DEFINITION **4.1**

Assume that there are n number of objects $O_1, \cdots, O_n$ and a concurrent object model defined as $O_1[\ s_1\ ](O_2[\ s_2\ ](\cdots(O_{n-1}[\ s_{n-1}\ ]O_n)\cdots))$. Then, for a single symbol x

$$\zeta(O, x) = \bigcup_{i=1}^{n} \{O_i | x \in \sigma(O_i)\}$$

and,

- $\zeta(O, \alpha^*) = \zeta(O, \alpha)$
- $\zeta(O, \alpha||\beta) = \zeta(O, \alpha + \beta) = \zeta(O, \alpha\beta) = \zeta(O, \alpha) \cup \zeta(O, \beta)$

For example, assume that $O_1 \equiv x+y$ and $O_2 \equiv yz$. Then $\zeta(O, x) = \{O_1\}, \zeta(O, y) = \{O_1, O_2\}$ and $\zeta(O, z) = \{O_2\}$.

Next, we formalise interleaving expressions as $I(O)$ that denotes a set of all possible sets of interleaving expressions of O. Note that by this definition of $I(O)$, an expression that possesses one or more interleaving expressions as its sub-expressions is also regarded as an interleaving expression.

DEFINITION **4.2 (Interleaving Expressions)**

Assume that O is a concurrent object model. Then a set of interleaving expressions $I(O)$ is defined as follows.

$$I(O) \equiv \{\alpha\beta \mid \zeta(O, \alpha) \cap \zeta(O, \beta) = \phi \text{ or } \alpha \in I(O) \text{ or } \beta \in I(O)\} \cup$$

$$\{\alpha^* \mid \alpha \in I(O)\} \cup$$
$$\{\alpha + \beta \mid \alpha \in I(O) \text{ or } \beta \in I(O)\} \cup$$
$$\{\alpha \,\|\, \beta \mid \alpha \in I(O) \text{ or } \beta \in I(O)\}$$

For example, if $O \equiv O_1 [\ ] O_2$, $\zeta(O, x) = \{O_1\}$ and $\zeta(O, y) = \{O_2\}$ then, xy and $x + y$ are interleaving expressions.

On the other hand, a set of all the possible over-concurrent expressions of O can be defined as follows. Notice that by the definition of $H(O)$, an expression which possesses one or more over-concurrent expressions is also an over-concurrent expression.

DEFINITION **4.3 (Over-concurrent Expressions)**
Suppose that O is a concurrent object model. Then a set of over-concurrent expressions $H(O)$ is defined as follows.

$$\begin{aligned} H(O) \equiv\ & \{\alpha \| \beta \mid \zeta(O, \alpha) \cap \zeta(O, \beta) \neq \phi\} \cup \\ & \{\alpha^* \mid \alpha \in H(O)\} \cup \\ & \{\alpha\beta \mid \alpha \in H(O) \text{ or } \beta \in H(O)\} \cup \\ & \{\alpha + \beta \mid \alpha \in H(O) \text{ or } \beta \in H(O)\} \end{aligned}$$

For example, $a \| b$ is in $H(O)$ for $O \equiv ab + ba$ because $\zeta(O, a) = \zeta(O, b) = \{O\}$

Now, we finally define the concurrent thread model.

DEFINITION **4.4 (Concurrent Thread Model)**
For any concurrent object model O, an expression $T(O)$ is a concurrent thread model of O if

1. *$T(O)$ has no $[\, s\,]$ operator.*
2. *$L(T(O)) = L(O)$*
3. *$T(O) \notin I(O) \cup H(O)$*

For example, assume that $O \equiv O_1 [\ ] O_2$, $O_1 \equiv axb$ and $O_2 \equiv ayb$. Then $a(x \| y)b$ and $a(xy + yx)b$ are equivalent to O. However, $a(xy + yx)b$ is not a concurrent thread model of O because $\zeta(O, x) = \{O_1\}$ and $\zeta(O, y) = \{O_2\}$ hence $xy + yx \in I(O)$. On the other hand, $a(x \| y)b$ is exactly a concurrent thread model of O.

4.3 Summary

In this Chapter we presented how to model concurrent object and concurrent thread models using BCREs. The semantics of these models are given as the language of BCREs. We also defined the correspondence between these models based on language equivalence. In the next Chapter we focus on the method of how to transform a concurrent object model into a concurrent thread model.

Chapter 5

Extracting Threads

Let O be an object model and let $T(O)$ be a thread model of O. If O and $T(O)$ represent the same system behaviour, $L(O) = L(T(O))$ holds. Here we are concerned with transforming an object model to a thread model, that is, to derive $T(O)$ from O in a systematic way. According to the completeness of our axiom system, $\vdash O = T(O)$ can always be proved. If $T(O)$ is known, a derivation path of $\vdash O = T(O)$ can always be obtained. However, our problem here is different. The problem is how to derive a unknown $T(O)$ from O. It is not a trivial problem, and is more complex than proving $\vdash O = T(O)$ with the axiom system. In this chapter, we propose a systematic way to solve the problem. First, we define the idea of thread-extractable form. Then we prove that any object model can be transformed to thread-extractable form. Finally, we present a way of transforming a thread model from a thread-extractable form.

5.1 Thread-extractable form

Consider an expression that begins with a sequence of non-communication symbols, succeeded by a communication symbol. Thread-extractable form is a set of such expressions defined as follows.

DEFINITION **5.1 (Thread-extractable form)**
Let $\{x_1, \cdots, x_r\}$ be a set of symbols. Then, it is stated that α is thread-extractable if following there are some $\alpha_1, \cdots, \alpha_n$ for $\{z_1, \cdots, z_{r'}\} \subseteq \{x_1, \cdots, x_r\}$. It is also stated that α_1 is a thread-extractable form of α.

$$\vdash \alpha_i \quad = \quad \sum_{j=1}^{h(\alpha_i)} \theta(\alpha_i, j) z(\alpha_i, j) \eta(\alpha_i, j) + \theta(\alpha_i, 0) \tag{5.1}$$

where $\vdash \alpha = \alpha_1$, and there are some $k (1 \leq k \leq n)$ for all i and j such that $\alpha_k \equiv \eta(\alpha_i, j)$ holds. $h(\alpha_i)$ is a natural number decided by α_i. $\theta(\alpha_i, j) \in \Theta$ and $z(\alpha_i, j) \in \{z_1, \cdots, z_{r'}\} \cup \perp$ for $j =$

$0, \cdots, h(\alpha_i)$ Θ *is a set of expressions, and it follows that* $\theta \in \Theta$ *implies* $\{z_1, \cdots, z_{r'}\} \cap \sigma(\theta) = \phi$ *for any expression* θ*. That is,* Θ *is a set of expressions that do not contain any symbol in* $\{z_1, \cdots, z_{r'}\}$*. Note that* $\perp$ *and* $\perp^*$ *are also in* Θ

Now, let us show that any expression is thread-extractable if the expression has no $[\, s \,]$ and $\|$ operators. In the remainder of this chapter, we use the axiom A_1 to A_{11} in proofs without it being explicitly referred to.

THEOREM **5.1**
All BCREs are thread-extractable.

PROOF

We will prove the theorem by induction on the syntax of BCREs. Let $\{x_1, \cdots, x_r\}$ be a set of symbols and assuming $\{z_1, \cdots, z_{r'}\}$ is an arbitrary subset of $\{x_1, \cdots, x_r\}$. We start with the base step.

1. Base step

It clearly follows that $\perp$, $\perp^*$ and any symbol $x_i \in \{x_1, \cdots, x_r\}$ are thread-extractable according to the following equations.

$$\begin{aligned}
\vdash \quad \perp &= \perp\perp\perp + \perp \\
\vdash \quad \perp^* &= \perp\perp\perp + \perp^* \\
\vdash \quad x_i &= \perp^* x_i \perp^* + \perp \quad if\ x_i \in \{z_1, \cdots, z'_r\} \\
\vdash \quad x_i &= \perp\perp\perp + x_i \quad if\ x_i \notin \{z_1, \cdots, z'_r\}
\end{aligned}$$

Next, suppose that α and β are thread-extractable then we will prove that $\alpha + \beta$, $\alpha\beta$ and α^* are thread-extractable.

2. For $\alpha + \beta$

By induction hypothesis, α is thread-extractable; therefore (5.1) holds. There are some $\beta_1, \cdots, \beta_m$, $\vdash \beta = \beta_1$ and

$$\vdash \beta_i = \sum_{j=1}^{h(\beta_i)} \theta(\beta_i, j) z(\beta_i, j) \eta(\beta_i, j) + \theta(\beta_i, 0) \tag{5.2}$$

holds for all $i = 1, \cdots, m$ where $\theta(\beta, j) \in \Theta$ and $\eta(\beta, j)$ is among $\beta_i (1 \leq i \leq m)$. By (5.1) and (5.2), it follows that

$$\begin{aligned} \vdash \alpha + \beta &= \sum_{j=1}^{h(\alpha_1)} \theta(\alpha_1, j) z(\alpha_1, j) \eta(\alpha_1, j) + \sum_{j=1}^{h(\beta_1)} \theta(\beta_1, j) z(\beta_1, j) \eta(\beta_1, j) + \theta(\alpha_1, 0) + \theta(\beta_1, 0) \\ &= \sum_{j=1}^{h(\alpha_1+\beta_1)} t(\alpha_1 + \beta_1, j) + \theta(\alpha_1 + \beta_1, 0) \end{aligned} \tag{5.3}$$

where $h(\alpha_1 + \beta_1) = h(\alpha_1) + h(\beta_1)$, $\theta(\alpha_1 + \beta_1, 0) \equiv \theta(\alpha_1, 0) + \theta(\beta_1, 0)$, and t is defined as follows.

$$\begin{aligned} t(\alpha_1 + \beta_1, i) &\equiv \theta(\alpha_1, i) z(\alpha_1, i) \eta(\alpha_1, i) \quad for \quad 1 \leq i \leq h(\alpha_1) \\ t(\alpha_1 + \beta_1, j + h(\alpha_1)) &\equiv \theta(\beta_1, j) z(\beta_1, j) \eta(\beta_1, j) \quad for \quad 1 < j \leq h(\beta_1) \end{aligned}$$

The structure of the form (5.3) is similar to the definition of the thread-extractable form such as (5.1). Hence, it is concluded that $\alpha + \beta$ is thread-extractable.

3. For $\alpha\beta$

By induction hypothesis, α is thread-extractable. Thus by (5.1), it follows for all α_i $(1 \leq \imath \leq n)$ that

$$\begin{aligned} \vdash \alpha_i\beta &= \sum_{j=1}^{h(\alpha_i)} \theta(\alpha_i, j) z(\alpha_i, j) \eta(\alpha_i, j) \beta + \theta(\alpha_i, 0)\beta \\ &= \sum_{j=1}^{h(\alpha_i)} \theta(\alpha_i, j) z(\alpha_i, j) \eta(\alpha_i, j) \beta + \sum_{j=1}^{h(\beta_1)} \theta(\alpha_i, 0)\theta(\beta_1, j) z(\beta_1, j) \eta(\beta_1, j) + \theta(\alpha_i, 0)\theta(\beta_1, 0) \end{aligned}$$

Since $\eta(\alpha_i, j)$ is among $\alpha_1, \cdots, \alpha_n$, $\eta(\alpha_i, j)\beta$ is also among $\alpha_1\beta, \cdots, \alpha_n\beta$. Therefore, the structure of the above expression is similar to (5.1). Hence, $\alpha_i\beta$ is thread-extractable. Considering that $\vdash \alpha\beta = \alpha_1\beta$, it is concluded that $\alpha\beta$ is thread-extractable.

4. For α^*

By induction hypothesis, α is thread-extractable; therefore, there are some $\alpha_1, ...\alpha_n$ such that $\vdash \alpha = \alpha_1$ and (5.1) holds. Hence,

$$\begin{aligned} \vdash \quad & \alpha^* = \perp^* + \alpha\alpha^* \\ = \quad & \perp^* + \sum_{j=1}^{h(\alpha_1)} \theta(\alpha_1, j) z(\alpha_1, j) \eta(\alpha_1, j) \alpha^* + \theta(\alpha_1, 0)\alpha^* \end{aligned} \tag{5.4}$$

It also follows for $1 \leq i \leq n$ that

$$\vdash \alpha_i\alpha^* = \sum_{j=1}^{h(\alpha_i)} \theta(\alpha_i, j) z(\alpha_i, j) \eta(\alpha_i, j) \alpha^* + \theta(\alpha_i, 0)\alpha^* \tag{5.5}$$

By (5.4) and the rule $R2$, it is proved that

$$\vdash \alpha^* = \sum_{j=1}^{h(\alpha_1)} (\theta'(\alpha_1,0))^* \theta(\alpha_1,j) z(\alpha_1,j) \eta(\alpha_1,j) \alpha^* + (\theta'(\alpha_1,0))^* \tag{5.6}$$

where $\theta'(\alpha,0) \equiv \theta(\alpha,0)$ if $\epsilon \notin L(\theta'(\alpha,0))$. Otherwise $\theta'(\alpha,0)$ is an expression such that $\vdash \theta(\alpha,0) = \theta'(\alpha,0) + \perp^*$ and $\epsilon \notin L(\theta'(\alpha,0))$ holds. By substituting (5.6) to (5.5),

$$\begin{aligned} \vdash \alpha_i \alpha^* &= \sum_{j=1}^{h(\alpha_i)} \theta(\alpha_i,j) z(\alpha_i,j) \eta(\alpha_i,j) \alpha^* \\ &+ \sum_{j=1}^{h(\alpha_1)} \theta(\alpha_i,0)(\theta'(\alpha_1,0))^* \theta(\alpha_1,j) z(\alpha_1,j) \eta(\alpha_1,j) \alpha^* \\ &+ \theta(\alpha_i,0)(\theta'(\alpha_1,0))^* \end{aligned}$$

Since $\eta(\alpha_i,j)$ is among $\alpha_1, \cdots, \alpha_n$, $\eta(\alpha_i,j)\alpha^*$ is also among $\alpha_1\alpha^*, \cdots, \alpha_n\alpha^*$. Therefore, the structure of the expression above is similar to (5.1). Thus, it is clear that $\alpha_i\alpha^*$ is thread-extractable. According to this result, it is also proved that the structure of (5.6) is likewise similar to (5.1). Hence, α^* is thread-extractable.

From what has been discussed above, it is concluded that all expressions without $[\, s \,]$ and $\|$ operators are thread-extractable. By this result, the rest of the proof is mostly clear. By the semantics of BCREs, there are some equivalent expressions without $[\, s \,]$ and $\|$ operators for any BCRE. By the completeness of our axiom system, we can always derive such an expression. It is already proved that all expressions without $[\, s \,]$ and $\|$ operators are thread-extractable. Hence any expression with $[\, s \,]$ and $\|$ operators are also thread-extractable. □

5.2 Transformation method

In this section, we present a systematic way of how to transform a concurrent object model into a concurrent thread model. This method is defined as a procedure with three steps. First, a concurrent object model is transformed into a thread-extractable form without interleaving and over-concurrent expression. Next, the thread-extractable form is transformed into a BCRE-labeled automata. Finally, a BCRE-labeled automata is transformed into a concurrent thread model. We describe the details of these steps in the following sections.

5.2.1 Object model to thread-extractable form

We begin with a method for transforming an object model that consists of two objects into a thread-extractable form. It has already been shown that any concurrent object model $\alpha \,[\, s \,]\, \beta$ is

a thread-extractable form. However, such a thread-extractable form may have some interleaving or over-concurrent expressions which should not appear in a concurrent thread model. In this section, we will show a way of transformation from an object model into a thread-extractable form without interleaving or over concurrent expressions.

LEMMA **5.1**

Let O be an arbitrary concurrent object model. Assume that there are some thread-extractable forms for each α and β such that $\alpha, \beta \notin I(O) \cup H(O)$ holds. Then there is a thread-extractable form for $\alpha\,[\,s\,]\,\beta$ which is not in $I(O) \cup H(O)$.

PROOF

By the semantics of BCREs, it clearly follows that $\vdash \alpha\,[\,s\,]\,\beta = \alpha\,[\,(\sigma(\alpha) \cap \sigma(\beta)) \cup s\,]\,\beta$. Thus in the remainder of this proof, we assume that $\sigma(\alpha) \cap \sigma(\beta) \subseteq s$ instead of using an arbitrary s.

Assume that there is a thread-extractable form for each α and β such that $\alpha, \beta \notin I(O) \cup H(O)$ holds. Therefore (5.1) and (5.2) holds for $\{z_1, \cdots, z_r\} = s$. Thus, it follows for all $1 \leq u \leq n$ and $1 \leq v \leq m$ that

$$\vdash \alpha_u\,[\,s\,]\,\beta_v = \left(\sum_{j=1}^{h(\alpha_u)} T(\alpha_u, j) + \theta(\alpha_u, 0)\right)\,[\,s\,]\,\left(\sum_{j=1}^{h(\beta_v)} T(\beta_v, j) + \theta(\beta_v, 0)\right)$$

where $T(\alpha_u, j) = \theta(\alpha_u, j)z(\alpha_u, j)\eta(\alpha_u, j)$ and $T(\beta_v, j) = \theta(\beta_v, j)z(\beta_v, j)\eta(\beta_v, j)$. Then by the axioms C_3 and C_5,

$$\sum_{j=1}^{h(\alpha_u)}\sum_{j=1}^{h(\beta_v)} (T(\alpha_u, j)\,[\,s\,]\,T(\beta_v, j)) + \sum_{j=1}^{h(\alpha_u)} (T(\alpha_u, j)\,[\,s\,]\,\theta(\beta_v, 0)) + \sum_{j=1}^{h(\beta_v)} (\theta(\alpha_u, 0)\,[\,s\,]\,T(\beta_v, j)) + (\theta(\alpha_u, 0)\,[\,s\,]\,\theta(\beta_v, 0))$$

By the theorem 3.7, 3.8 and the axiom C_4,

$$\sum_{i=1}^{h(\alpha_u)}\sum_{j=1}^{h(\beta_v)} (\theta(\alpha_u, i)||\theta(\beta_v, j))z(i, j)(\eta(\alpha_u, i)\,[\,s\,]\,\eta(\beta_v, j)) + (\theta(\alpha_u, 0)||\theta(\beta_v, 0)) \tag{5.7}$$

where $z(i, j) = z(\alpha_u, i)$ if $z(\alpha_u, i) = z(\beta_v, j)$, otherwise $z(i, j) = \bot$. Since α_u and β_v are thread-extractable, $\eta(\alpha_u, i)\,[\,s\,]\,\eta(\beta_v, j)$ are also among $\alpha_u\,[\,s\,]\,\beta_v$ $(1 \leq u \leq n, 1 \leq v \leq m)$. By $\vdash \alpha\bot = \bot\alpha = \bot$ and $\vdash \alpha + \bot = \bot + \alpha = \alpha$ (cf.[6]), all $z(i, j) = \bot$ can be eliminated from (5.7). Then the structure of (5.7) is similar to (5.1). Hence all $\alpha_u\,[\,s\,]\,\beta_v$ are thread-extractable. By the assumption, and the definition of I and H, $\theta(\alpha_u, i)$, $\theta(\beta_v, j)$ and $\eta(\alpha_u, i)\,[\,s\,]\,\eta(\beta_v, j)$ are not in $I(O) \cup H(O)$. Therefore, it is obvious that (5.7) is not in $I(O) \cup H(O)$. Hence, $\alpha_1\,[\,s\,]\,\beta_1$

is a thread-extractable form for $\alpha\,[\,s\,]\,\beta$ and not in $I(O) \cup H(O)$. Therefore, the lemma follows. □

The Lemma 5.1 ensures that thread-extractable form for $\alpha\,[\,s\,]\,\beta$ as $\alpha_1\,[\,s\,]\,\beta_1$ exists and that it is not in $I(O) \cup H(O)$; namely it is neither an interleaving nor an over-concurrent expression.

5.2.2 Transforming function

According to the result of Theorem 5.1 and Lemma 5.1, we can define a systematic transformation function X_z, which is a morphism from an expression into a closed set of equations as in the schema $\alpha = \beta$ that represents equations in a thread-extractable form.

Let α and β be expressions that do not contain $[\,s\,]$ and $\|$ operators. z is a set of symbols. Let $h(\alpha)$ be a natural number,and $\theta(\alpha, j)$ be an expression. $z(\alpha, j)$ is a member of z and $\eta(\alpha, j)$ is an expression. They are obtained from a thread-extractable form of α. In other words, it follows that

$$\alpha = \sum_{j=1}^{h(\alpha)} \theta(\alpha, j) z(\alpha, j) \eta(\alpha, j) + \theta(\alpha, 0) \in X_z(\alpha)$$

Thus for all $1 \leq j \leq h(\alpha)$, $\sigma(\theta(\alpha, j)) \cap z = \phi$ and $\sigma(z(\alpha, j)) \subseteq z$ holds. Now, together with the above assumptions, X_z can be recursively defined as a minimal set as follows.

1. $X_z(\perp) = \{\perp = \perp\perp\perp + \perp\}$
2. $X_z(\perp^*) = \{\perp^* = \perp\perp\perp + \perp^*\} \cup X_z(\perp)$
3. $X_z(x) = \{x = \perp\perp\perp + x\} \cup X_z(\perp)$ if $x \notin z$.
4. $X_z(x) = \{x = \perp^* x \perp^* + \perp\} \cup X_z(\perp^*)$ if $x \in z$.
5. $X_z(\alpha + \beta) = \{\ \alpha + \beta = \rho(\alpha\beta)\ \} \cup \bigcup_{j=1}^{h(\alpha)} X_z(\eta(\alpha, j)) \cup \bigcup_{j=1}^{h(\beta)} X_z(\eta(\beta, j))$
6. $X_z(\alpha\beta) = \{\ \alpha\beta = \rho(\alpha\beta)\ \} \cup \bigcup_{j=1}^{h(\alpha)} X_z(\eta(\alpha, j)\beta) \cup \bigcup_{j=1}^{h(\beta)} X_z(\eta(\beta, j))$
   if there is no γ such that $\beta \equiv \gamma^*$ holds.
7. $X_z(\alpha^*) = \{\ \alpha^* = \rho(\alpha^*)\ \} \cup \bigcup_{j=1}^{h(\alpha)} X_z(\eta(\alpha, j)\alpha^*, z)$
8. $X_z(\beta\alpha^*) = \{\ \beta\alpha^* = \rho(\beta\alpha^*)\ \} \cup \bigcup_{j=1}^{h(\alpha)} X_z(\eta(\alpha, j)\alpha^*) \cup \bigcup_{j=1}^{h(\beta)} X_z(\eta(\beta, j)\alpha^*)$

9. $X_z(\alpha\,[\,s\,]\,\beta) = \{\ \alpha\,[\,s\,]\,\beta = \rho(\alpha\,[\,s\,]\,\beta)\ \} \cup \bigcup_{i=1}^{h(\alpha)} \bigcup_{j=1}^{h(\beta)} X_z(\eta(\alpha,i)\,[\,s\,]\,\eta(\beta,j))$

where ρ is defined as follows.

$$
\begin{aligned}
\rho(\alpha+\beta) &\equiv \sum_{j=1}^{h(\alpha)} \theta(\alpha,j)z(\alpha,j)\eta(\alpha,j) + \sum_{j=1}^{h(\beta)} \theta(\beta,j)z(\beta,j)\eta(\beta,j) + \theta(\alpha,0) + \theta(\beta,0) \\
\rho(\alpha\beta) &\equiv \sum_{j=1}^{h(\alpha)} \theta(\alpha,j)z(\alpha,j)\eta(\alpha,j)\beta + \sum_{j=1}^{h(\beta)} \theta(\alpha,0)\theta(\beta,j)z(\beta,j)\eta(\beta,j) + \theta(\alpha,0)\theta(\beta,0) \\
\rho(\alpha^*) &\equiv \sum_{j=1}^{h(\alpha)} (\theta'(\alpha,0))^*\theta(\alpha,j)z(\alpha,j)\eta(\alpha,j)\alpha^* + (\theta'(\alpha,0))^* \\
\rho(\beta\alpha^*) &\equiv \sum_{j=1}^{h(\beta)} \theta(\beta,j)z(\beta,j)\eta(\beta,j)\alpha^* + \sum_{j=1}^{h(\alpha)} \theta(\beta,0)(\theta'(\alpha,0))^*\theta(\alpha,j)z(\alpha,j)\eta(\alpha,j)\alpha^* + \\
&\quad \theta(\beta,0)(\theta'(\alpha,0))^* \\
\rho(\alpha\,[\,s\,]\,\beta) &\equiv \sum_{i=1}^{h(\alpha)} \sum_{j=1}^{h(\beta)} (\theta(\alpha,i)||\theta(\beta,j))Z(i,j)(\eta(\alpha,i)\,[\,s\,]\,\eta(\beta,j)) + (\theta(\alpha,0)||\theta(\beta,0))
\end{aligned}
$$

where $\theta'(\alpha,0) \equiv \theta(\alpha,0)$ if $\epsilon \notin L(\theta'(\alpha,0))$. Otherwise $\theta'(\alpha,0)$ is an expression such that $\vdash \theta(\alpha,0) = \theta'(\alpha,0) + \perp^*$ and $\epsilon \notin L(\theta'(\alpha,0))$ holds. $Z(i,j) \equiv z(\alpha,i)$ if $z(\alpha,i) = z(\beta,j)$, otherwise $Z(i,j) = \perp$.

Note that the sub z of X_z represents a set of communication symbols. If the concurrent object model $\alpha\,[\,s\,]\,\beta$ is given, the thread-extractable form for $\alpha\,[\,s\,]\,\beta$ can be calculated $X_z(\alpha\,[\,s\,]\,\beta)$ where $z = (\sigma(\alpha) \cap \sigma(\beta)) \cup s$.

5.2.3 Thread-extractable form to automata

In this section, we show how to transform a thread-extractable form to an automata representation called a BCRE-labelled automata. Assume that a thread-extractable form consists of a set of equations $Q = \{\alpha_1 = \beta_1, \cdots, \alpha_n = \beta_n\}$ such that

$$\beta_i \equiv \sum_{j=1}^{h(\alpha_i)} \theta(\alpha_i,j)z(\alpha_i,j)\eta(\alpha_i,j) + \theta(\alpha_i,0)$$

Then, a BCRE-labelled automata of Q: $G(Q) =< N, B, T, S, F >$ is defined as follows.

1. $N = \{\alpha_1, \cdots, \alpha_n, \zeta\}$
2. $B = \{\theta(\alpha_i,j) \mid 1 \le i \le n, 0 \le j \le h(\alpha_i)\}$
3. $T = \{< \alpha_i, \theta(\alpha_i,j)z(\alpha_i,j), \eta(\alpha_i,j) > \mid 1 \le i \le n, 1 \le j \le h(\alpha_i)\} \cup$
   $\{< \alpha_i, \theta(\alpha_i,0), \zeta >\}$

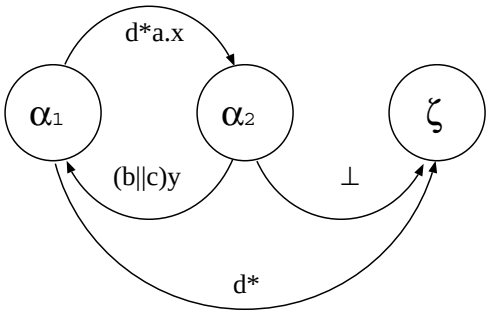


Figure 5.1: BCRE-labelled automata

4. $S = \alpha_1$

5. $F = \{\zeta\}$

where N represents a set of states, B is a set of labels, S is the initial state and F is a set of terminal states. T is a set of transitions. Each transition is defined as $< t_s, l, t_e >$: t_s is a state from which the transition leaves, t_e is a state to which the transition heads, and l is a transition label. The following is an example of a thread-extractable form of $(ax(b \parallel c)y + d)^*$ where x and y are communication symbols.

$$\begin{aligned} Q = \quad \{ \quad & \alpha_1 = d^* a x \alpha_2 + d^*, \\ & \alpha_2 = (b \parallel c) y \alpha_1 + \perp \} \end{aligned}$$

where $\alpha_1 \equiv (ax(b \parallel c)y + d)^*$ and $\alpha_2 \equiv (b \parallel c)y(ax(b \parallel c)y + d)^*$. From this thread-extractable form by the procedure 5.2.3, the automata for $G(Q)$ can be constructed as follows, and is depicted in Figure 5.2.3.

$$\begin{aligned} N &= \{\alpha_1, \alpha_2\, \zeta\} \\ T &= \{< \alpha_1, d^* a x, \alpha_2 >, < \alpha_1, d^*, \zeta >, < \alpha_2, (b \parallel c)y, \alpha_1 >, < \alpha_2, \perp, \zeta >\} \\ S &= \alpha_1 \\ F &= \{\zeta\} \end{aligned}$$

5.2.4 Automata to Thread model

Let us begin with transforming a BCRE-labelled automata to a BCRE. We can directly apply a transformation method from an automata to a regular expression. Since this transformation is a problem well-known and solved in the automata domain, we are not concerned here with details

of this transformation. For example, one traditional solution can be seen in [9]. A corresponding BCRE of BCRE-labelled automata is not uniquely determined. One or more expressions may be obtained from one BCRE-labelled automata.

For example, $(d^*ax(b\|c)y)^* + d* + d^*ax\perp$ is one such expression obtained from an automata in Figure 5.2.3. The meaning of the resulting BCRE from the BCRE-labelled automata is clearly equivalent to the thread-extractable form that the BCRE-labelled automata is based on. Hence $\vdash (ax(b \,\|\, c)y + d)^* = (d^*ax(b\|c)y)^* + d* + d^*ax\perp$ holds.

5.2.5 Transformation Procedure

Assume that $O \equiv \alpha\,[\,s\,]\,\beta$ is a concurrent object model. Then, a concurrent thread model $T(O)$ can be obtained by the following procedure.

PROCEDURE **5.1**

1. *Calculate $X_z(O)$ according to Section 5.2.2 where $z = (\sigma(\alpha) \cap \sigma(\beta)) \cup s$.*
2. *Calculate $G(X_z(O))$ according to Section 5.2.3.*
3. *Transform $G(X_z(O))$ to $T(O)$ according to Section 5.2.4*

The following theorem guarantees that such a $T(O)$ is exactly a concurrent thread model of O.

THEOREM **5.2**

Let $T(O)$ be obtained from a concurrent object model O according to procedure 5.1. Then, it follows that

1. *$T(O)$ has no $[\,s\,]$ operator.*
2. *$L(T(O)) = L(O)$ holds.*
3. *$T(O) \notin I(O) \cup H(O)$.*

PROOF

First, since there is no $[\,s\,]$ operator in each of the objects, it follows that the labelled set of $G(X_z(O))$ does not contain $[\,s\,]$. Therefore $T(O)$ does not have $[\,s\,]$, either. It is fairly obvious that the language of O is preserved through the transformation procedure. Hence $\vdash T(O) = O$. Finally, it is clear from the definition of X_z that $\beta \notin I(O) \cup H(O)$ for any $\alpha = \beta \in X_z(O)$; thus, it can be proved that the results of the procedure in Section 5.2.3 and 5.2.4 do not contain any interleaving or over-concurrent expressions. Hence $T(O) \notin I(O) \cup H(O)$. □

5.2.6 Transformation for more than three objects

It is possible to adopt our transformation method to a model of more than three objects in the following way. Assume that $(\alpha\,[\,s_1\,]\,\beta)\,[\,s_2\,]\,\gamma$ is a concurrent object model. First, obtaining a thread-extractable form δ for $\alpha\,[\,s_1\,]\,\beta$. Then, calculating a thread-extractable form for $\delta\,[\,s_2\,]\,\gamma$. In a similar way, we can obtain a thread-extractable form for any number of concurrent objects. However, if there are three or more objects, there are cases where a thread-extractable form can not be obtained. For example,

$$(\alpha x\beta\,[\ ]\,\gamma y\delta)\,[\ ]\,xy$$

where x and y are communication symbols. By transforming $\alpha x\beta\,[\ ]\,\gamma y\delta$ into a thread model,

$$(\alpha x\beta \| \gamma y\delta)\,[\ ]\,xy$$

Our method can not transform this expression any more. To handle such an expression, a way is needed for transforming any expression that possesses $\|$ operators; this is part of our future work.

5.2.7 Terminating Property

Our transformation procedure is always terminated. This is proved easily by the following. First, since a thread-extractable form consists of a finite number of $\alpha_1, \cdots, \alpha_n$, a thread-extractable form can be obtained by finite steps. Second, since a thread-extractable form consists of a finite number of expressions, then, BCRE-labelled automata can be obtained in finite steps. Hence, it is obvious that our transformation always terminates in a finite number of steps.

5.2.8 Complexity

This section looks at the complexity of our transformation method by discussing the number of states and transitions of BCRE-labelled automata generated by our transformation method.

Assume that α and β be expressions which have no $[\,s\,]$ and $\|$ operators. Then we consider a BCRE-labelled automata $G(X_z(\alpha\,[\,s\,]\,\beta))$ where $z = (\sigma(\alpha) \cap \sigma(\beta)) \cup s)$. There are states less than $|\alpha|_e|\beta|_e + 1$ and transitions less than $|\alpha|_e|\beta|_e(|\alpha|_t|\beta|_t + 1)$. $|\alpha|_e$ is the number of equations in $X_z(\alpha)$ and $|\alpha|_t$ is defined as follows

$$\begin{aligned} |x|_t &= 1 \\ |\alpha + \beta|_t &= |\alpha\beta|_t = |\alpha|_t + |\beta|_t \\ |\alpha^*|_t &= 2|\alpha|_t \end{aligned}$$

where x is a symbol, α and β are expressions. Intuitively, $|\alpha|_t$ is similar to the length of α except that $|\alpha^*|_t$ is considered to be twice its length. If there is an n depth of nested closure operators, $|\alpha|_t$ is relative to 2^n times its length in the worst case. For example, $|((\alpha^*)^*)^*|_t = 2^3|\alpha|_t$.

Now, we focus on some details. Suppose that $|\alpha|_e$ and $|\beta|_e$ are a number of equations in $X_z(\alpha)$ and $X_z(\beta)$ respectively. According to (5.3) in Lemma 5.1, the transformation process ends when it completes exploring all possible $\eta(\alpha, i)\,[\,s\,]\,\eta(\beta, j)$ that are among $\alpha_u\,[\,s\,]\,\beta_v$ ($1 \leq u \leq |\alpha|_e$, $1 \leq v \leq |\beta|_e$). That is, $X_z(\alpha\,[\,s\,]\,\beta)$ is a subset of the following set of equations:

$$X_z(\alpha\,[\,s\,]\,\beta) \subseteq \{\ \alpha_u\,[\,s\,]\,\beta_v = \rho(\alpha_u\,[\,s\,]\,\beta_v) \mid \alpha_u \in \acute{X}_z(\alpha), \beta_v \in \acute{X}_z(\beta)\ \}$$

where $\acute{X}_z$ is a set of L.H.S. of a member of X_z such that

$$\acute{X}_z(\alpha) = \{\alpha \mid \exists\beta.\alpha = \beta \in X_z(\alpha)\}$$

See Lemma C.4 in the Appendix for a proof of this property. Since the number of equations in a thread-extractable form follows the number of $\alpha_u\,[\,s\,]\,\beta_v$ being explored, that is clearly less than $|\alpha|_e|\beta|_e$. According to the definition of G, the number of states in $G(X_z(\alpha\,[\,s\,]\,\beta))$ is the same as the number of equations in $X_z(\alpha\,[\,s\,]\,\beta)$. Hence, it is concluded that the BCRE-labelled automata also possess less than $|\alpha|_e|\beta|_e + 1$ number of states. Note that $+1$ is for the terminal state that does not appear among $(\alpha_u\,[\,s\,]\,\beta_v)$.

On the other hand, we discuss the number of transitions in $G(X_z(\alpha\,[\,s\,]\,\beta))$. First, lets consider Figure 5.2 that depicts an equation in $X_z(\alpha\,[\,s\,]\,\beta)$. From Figure 5.2, it is clear that the number of transitions from an each state $\alpha_u\,[\,s\,]\,\beta_u$ equals $h(\alpha_u)h(\beta_v) + 1$. Therefore, the number of transitions in $X_z(\alpha\,[\,s\,]\,\beta)$ is less than the summation of $h(\alpha_u)h(\beta_v) + 1$ for all u and v, that is

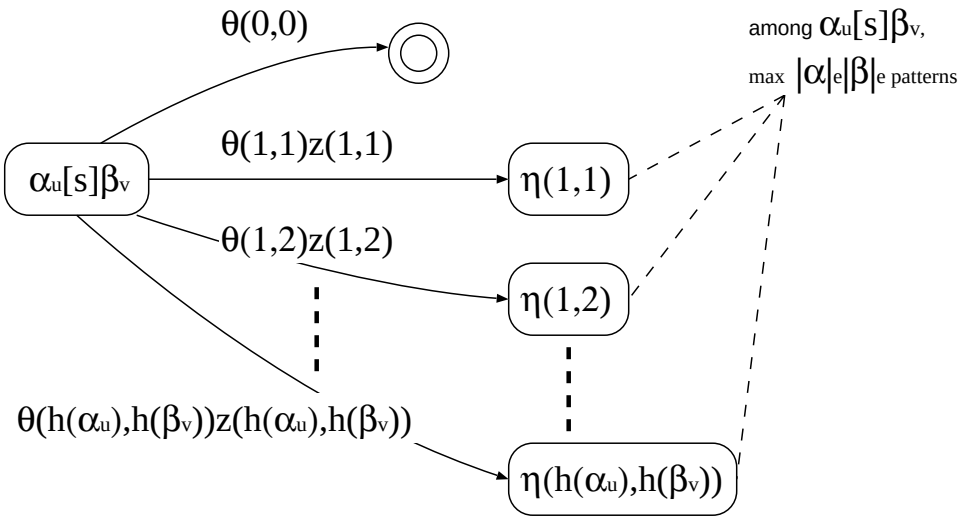
$$\sum_{u=1}^{|\alpha|_e}\sum_{v=1}^{|\beta|_e} h(\alpha_u)h(\beta_v) + 1$$

Now, we consider the upper-bounds of $h(\alpha_u)$ and $h(\beta_v)$. According to the definition of X_z, it is easy to prove the following inequations.

$$\begin{aligned} h(x) &= 1 \\ h(\alpha + \beta) &\leq h(\alpha) + h(\beta) \\ h(\alpha\beta) &\leq h(\alpha) + h(\beta) \\ h(\alpha^*) &\leq h(\alpha) \end{aligned}$$

where x is a symbol, $\perp$ or $\perp^*$. α and β are expressions. Hence, it is easy to prove that $h(\alpha) \leq |\alpha|_l$ for any α. $|\alpha|_l$ is the length of α defined as follows.

$$|x|_l = 1$$



where $\theta(i,j) = \theta(\alpha_{u,}i) \| \theta(\beta_{v,}j)$ and $\eta(i,j) = \eta(\alpha_{u,}i)[s]\eta(\beta_{v,}j)$

Figure 5.2: thread-extractable form as a BCRE-labelled automata

$$
\begin{aligned}
|\alpha + \beta|_l &= |\alpha\beta|_l = |\alpha|_l + |\beta|_l \\
|\alpha^*|_l &= |\alpha|_l
\end{aligned}
$$

The following lemma is useful for analyzing the upper-bound of the length of the expressions.

LEMMA **5.2**

Let γ and δ be expressions without $[\, s\,]$ and $\|$ operators then, for any z,

$$|\delta|_l \leq |\gamma|_t \textit{ if } \delta \in \acute{X}_z(\gamma) \textit{ follows.}$$

See the Appendix for a proof of this lemma.

By Lemma 5.2, it is clear that $h(\alpha_u)h(\beta_v) \leq |\alpha|_l|\beta|_l \leq |\alpha|_t|\beta|_t$. Therefore, it can be proved that

$$\sum_{u=1}^{|\alpha|_e} \sum_{v=1}^{|\beta|_e} h(\alpha_u)h(\beta_v) + 1 \leq |\alpha|_e|\beta|_e(|\alpha|_t|\beta|_t + 1)$$

Hence, it is concluded that the number of transitions in $G(X_z(\alpha\,[\, s\,]\,\beta))$ is less than $|\alpha|_e|\beta|_e(|\alpha|_t|\beta|_t + 1)$.

5.2.9 Summary

This Chapter presented a method for transformation from a concurrent object model to a concurrent thread model. The method is valid and has a terminating property. The function X_z in Section 5.2.2 represents the systematic transforming function, which is useful as a logical

basis for implementation of automatic transformation engines. In the last section, we discussed the cost of our transformation method with respect to a number of states and transitions in a resulting BCRE-labelled automata of transformation. In the worst case, our method generates automata that includes $|\alpha|_e|\beta|_e + 1$ numbers of states for transforming $\alpha\,[\,s\,]\,\beta$ where $|\alpha|_e$ is the number of equations in a thread-extractable form of α.

Chapter 6

Examples

In this chapter we demonstrate our approach with three examples.

1. A media player
2. An automatic locking door
3. A PCM device driver

These three examples have different characteristics. The first, Media player, is an example of the equivalent transformation of BCREs by the axiom system F_c. In the second, the Automatic locking door example, the transformation is performed by the method shown in Chapter 5. The third example, the PCM device driver, demonstrates that it is possible to derive real implementation from our concurrent thread model. We show a heuristic mapping from the concurrent thread model to C implementation. The detailed transformation way is omitted in this example.

6.1 Media player

This example begins by obtaining a concurrent object model from scenarios, then deriving a concurrent thread model.

6.1.1 Modelling objects

Let us begin with the scenarios that represent requirements of a simple video and audio player; they are smaller than the behaviour of real software, but adequate for showing the essence of our transformation.

```
Scenario 1. Playing a video file with audio data.
 The following actions should be performed when the media player
 receives a play request for a file that contains both video and audio data.
 A) The Main Thread creates instances of Audio and Video Threads.
 B) Video Thread and Audio Thread play video and audio synchronously.
 C) The Main Thread deletes the instance of the Audio and Video Threads.

Scenario 2. Playing a video file without audio data.
The following actions should be performed when a video file without audio
 data is given to the media player.
 A) The Main Thread creates instances of Audio and Video Threads.
 B) The Audio Thread confirms that no audio data exists in the target file.
    Then it should send NO_AUDIO events to The Main Thread and the Video Thread.
 C) The Main Thread deletes the instance of the Video Thread. The Video Thread
    plays the video data. The Main Thread should also play dummy sound data.
 D) The Main Thread deletes the instance of the Video Thread when
    the Video Thread finishes playing.

Scenario 3. Playing an audio file without video data.
  The following actions should be performed when an audio file without video
  data is given to the media player.
  A) The Main Thread creates instances of Audio and Video Threads.
  B) The Video Thread confirms that no video data exists and it should send
     NO_VIDEO events to the Main and Audio Threads.
  C) The Main Thread deletes the instance of the Video Thread. The Audio Thread
    plays the audio data. The Main Thread should also play dummy video data.
  D) The Main Thread deletes the instance of the Audio Thread when
    it finishes playing.
```

Table 6.1: Example scenarios

Figure 6.1 shows the sequence diagrams obtained from the scenarios in Figure 6.1. There are 3 objects in the scenarios; Main Thread, Video Thread and Audio Thread. In this figure, $Main$, $Video$, $Audio$ are the representations for each of the objects, respectively.

Since any communication between objects is synchronised, The directions of the communications are ignored when we turn these diagrams into BCREs. However, in Figure 6.1, arrows are used to help understanding.

In our approach, all communications between objects are defined by event symbols, such as CREATE_AUDIO, SYNC_PLAY in Figure 6.1. The meaning of events is shown in Table 6.2.

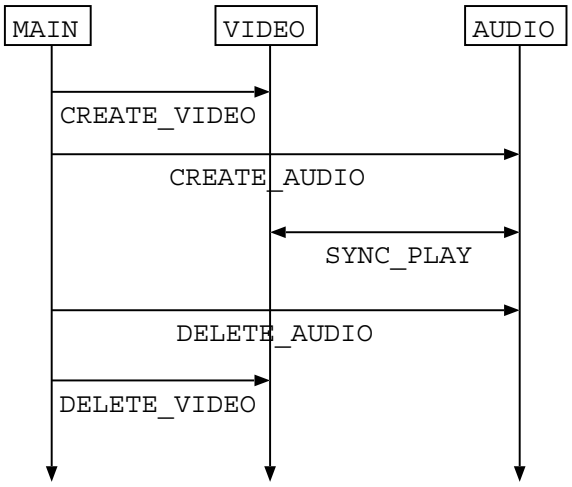
We can systematically transform the sequence diagrams in Figure 6.1 into state machines. First, we extract the behaviour of an object as a BCRE from each sequence diagram. The extraction procedure is as follows.

1. Collect all events directed from/to the object in the sequence diagram.

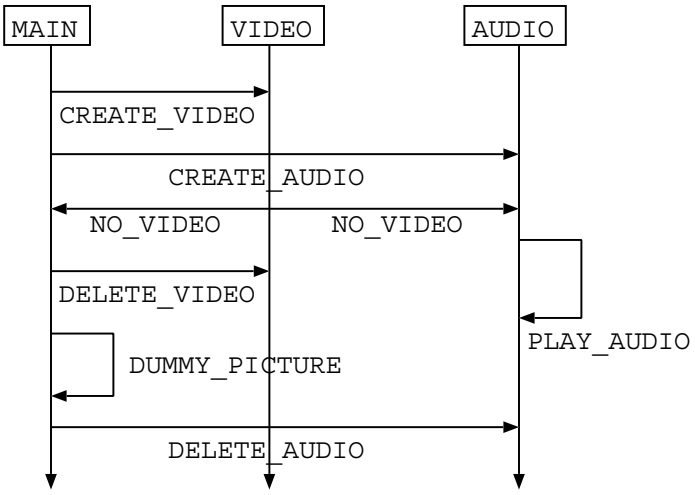
2. Join the events with '.' operators in the order of event occurrence in the sequence diagram.

For example, the following expressions are obtained as a set of partial behaviours of object MAIN.

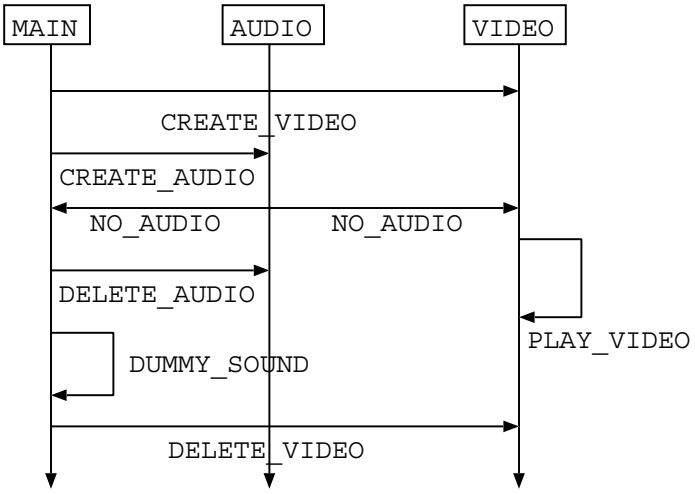
- $C_V.C_A.D_A.D_V$ (obtained from the scenario 1)



The sequence diagram for Scenario 1.



The sequence diagram for Scenario 2.



The sequence diagram for Scenario 3.

Figure 6.1: Sequence diagrams

```
CREATE_VIDEO  Create Video Thread.
CREATE_AUDIO  Create Audio Thread.
DELETE_VIDEO  Delete Video Thread.
DELETE_AUDIO  Delete Audio Thread.
SYNC_PLAY     Play video and audio data with synchronize.
PLAY_AUDIO    Play audio data.
PLAY_VIDEO    Play video data.
NO_AUDIO      An event which represents no audio data exists
NO_VIDEO      An event which represents no video data exists
DUMMY_VIDEO   Play dummy video data.
DUMMY_SOUND   Play dummy sound data.
```

Table 6.2: Event Descriptions

- $C_V.C_A.N_V.D_V.D_P.D_A$ (from the scenario 2)
- $C_V.C_A.N_A.D_A.D_S.D_V$ (from the scenario 3)

where each symbol; C_V, C_A, D_A,$\cdots$ denotes abbreviations of CREATE_VIDEO, CREATE_AUDIO, DELETE_AUDIO,$\cdots$, respectively.

Then, we define the behaviour of the object MAIN as a summation of these sequences. The state machine for MAIN can be generated by connecting all the expressions with the $+$ operator, that is,

$$C_V.C_A.D_A.D_V + C_V.C_A.N_V.D_V.D_P.D_A + C_V.C_A.N_A.D_A.D_S.D_V$$

The state machine represented by this expression has non-deterministic transitions with respect to $C_V.C_A$. To remove such transitions, we use the distributive property of the operation $+$ of a regular expression such as the axioms A_4 and A_5. Then, the above expression can be turned into the following expression by applying the axiom.

$$C_V.C_A.(D_A.D_V + N_V.D_V.D_P.D_A + N_A.D_A.D_S.D_V)$$

The state machines for VIDEO and AUDIO can be obtained in a similar way. Table 6.3 shows the expressions for all objects and Figure 6.2 shows the corresponding state machines.

6.1.2 Transformation

Returning to the state machines in Figure 2.1. The concurrent object model for our Media Player is defined as

$$MAIN\,[\,]\,(VIDEO\,[\,]\,AUDIO)$$

$$
\begin{aligned}
MAIN &\equiv C_V.C_A.(D_A.D_V + N_V.D_V.D_P.D_A + N_A.D_A.D_S.D_V) \\
VIDEO &\equiv C_V.(S_P + N_V + N_A.P_V).D_V \\
AUDIO &\equiv C_A.(S_P + N_A + N_V.P_A).D_A
\end{aligned}
$$

Table 6.3: Expressions for Each Objects

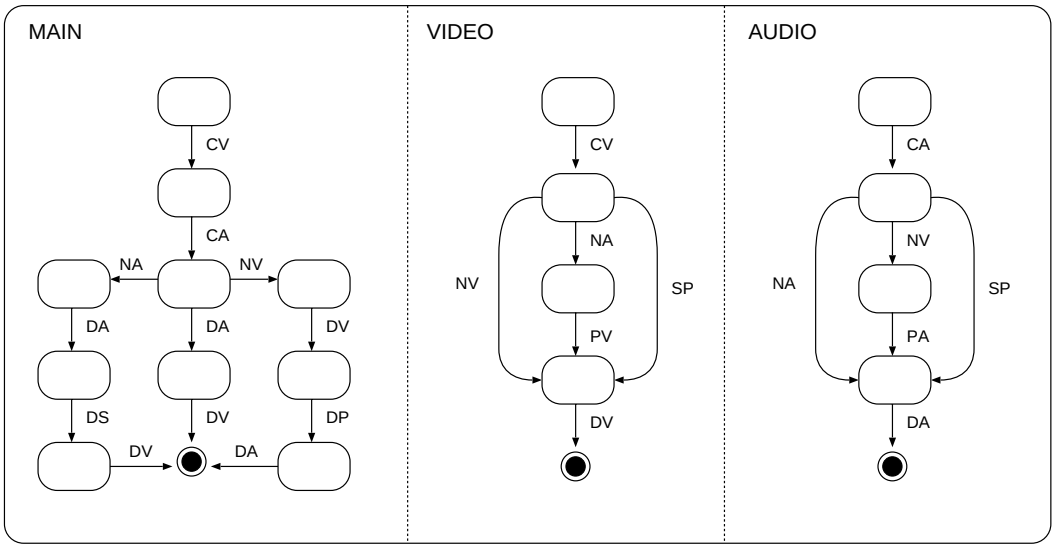


Figure 6.2: Objects

where $MAIN$, $VIDEO$ and $AUDIO$ are defined as in Table 6.3. Note that the order of the composite pairing does not effect the result because Theorem 3.5 holds in this case; that is,

$$MAIN\,[\ ]\,(VIDEO\,[\ ]\,AUDIO) = (MAIN\,[\ ]\,VIDEO)\,[\ ]\,AUDIO$$

holds.

Now, we begin by first transforming $(VIDEO\,[\ ]\,AUDIO)$.

$$\vdash VIDEO\,[\ ]\,AUDIO = C_V.(S_P + N_V + N_A.P_V).D_V\,[\ ]\,C_A.(S_P + N_A + N_V.P_A).D_A$$

By Theorem 3.8

$$= (C_V \,\|\, C_A).((S_P + N_V + N_A.P_V).D_V\,[\ ]\,(S_P + N_A + N_V.P_A).D_A)$$

By Axiom A_5,

$$= (C_V \,\|\, C_A).(S_P.D_V + (N_V + N_A.P_V).D_V\,[\ ]\,(S_P + N_A + N_V.P_A).D_A)$$

By Axiom C_5,

$$
\begin{aligned}
= \quad & (C_V \,\|\, C_A).((S_P.D_V\,[\,N_A, N_V\,]\,(S_P + N_A + N_V.P_A).D_A) + \\
& ((N_V + N_A.P_V).D_V\,[\,S_P\,]\,(S_P + N_A + N_V.P_A).D_A))
\end{aligned}
$$

By Axiom C_5,

$$= (C_V \,\|\, C_A).((S_P.D_V\,[\,N_A, N_V\,]\,S_P.D_A) + (S_P.D_V\,[\,S_P, N_A, N_V\,]\,N_A.D_A) + (S_P.D_V\,[\,S_P, N_A, N_V\,]\,N_V.P_A.D_A)) + ((N_V + N_A.P_V).D_V\,[\,S_P\,]\,(S_P + N_A + N_V.P_A).D_A))$$

By Axiom C_4,

$$= (C_V \,\|\, C_A).((S_P.(D_V\,[\,S_P, N_A, N_V\,]\,D_A) + \perp + \perp) + ((N_V + N_A.P_V).D_V\,[\,S_P\,]\,(S_P + N_A + N_V.P_A).D_A))$$

By theorem 3.8,

$$= (C_V \,\|\, C_A).(S_P.(D_V \,\|\, D_A) + ((N_V + N_A.P_V).D_V\,[\,S_P\,]\,(S_P + N_A + N_V.P_A).D_A))$$

By the Axiom A_5,

$$= (C_V \,\|\, C_A).(S_P.(D_V \,\|\, D_A) + (N_V.D_V + N_A.P_V.D_V\,[\,S_P\,]\,S_P.D_A + N_A.D_A + N_V.P_A.D_A))$$

By Axiom C_5,

$$= (C_V \,\|\, C_A).(S_P.(D_V \,\|\, D_A) + ((N_V.D_V\,[\,S_P, N_A\,]\,S_P.D_A + N_A.D_A + N_V.P_A.D_A)) + ((N_A.P_V.D_V\,[\,S_P, N_V\,]\,S_P.D_A + N_A.D_A + N_V.P_A.D_A)))$$

By Axiom C_3, C_4 and C_5,

$$= (C_V \,\|\, C_A).(S_P.(D_V \,\|\, D_A) + (N_V.D_V\,[\,S_P, N_A\,]\,N_V.P_A.D_A) + (N_A.P_V.D_V\,[\,S_P, N_V\,]\,N_A.D_A))$$

By Axiom C_4,

$$= (C_V \,\|\, C_A).(S_P.(D_V \,\|\, D_A) + (N_V.(D_V\,[\,S_P, N_A, N_V\,]\,P_A.D_A)) + (N_A.(P_V.D_V\,[\,S_P, N_A, N_V\,]\,D_A))$$

By Axiom C_4 and the Theorem 3.8,

$$= (C_V \,\|\, C_A).(S_P.(D_V \,\|\, D_A) + (N_V.(D_V \,\|\, P_A.D_A)) + (N_A.(P_V.D_V \,\|\, D_A))$$

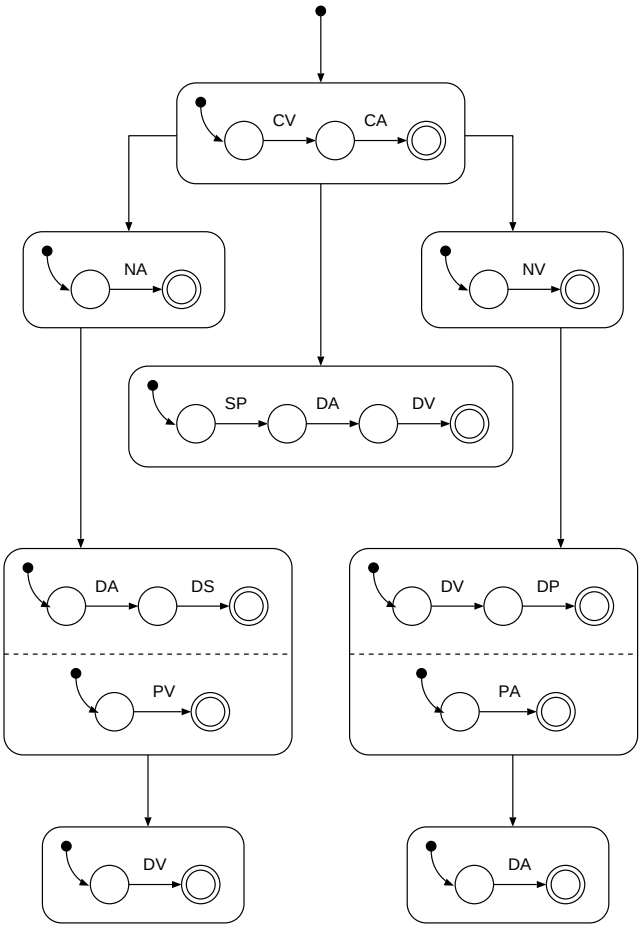


Figure 6.3: Concurrent Thread Model of Media Player

Then, we can also compose $MAIN$ and $(VIDEO[\ ]AUDIO)$ in a similar way. The following expression can be derived from $MAIN[\ ](VIDEO[\ ]AUDIO)$.

$$C_V.C_A.(S_P.D_A.D_V + N_V.(D_V.D_P \parallel P_A).D_A + N_A.(D_A.D_S \parallel P_V).D_V)$$

Since there is no interleaving expression or over-concurrent expression, this expression represents a concurrent thread model of $MAIN[\ ](VIDEO[\ ]AUDIO)$. Figure 6.1.2 depicts the model represented by this expression.

6.2 Automatic locking door system

In this section, we show an example of our transformation method that was described in a previous chapter.

6.2.1 Modelling objects

The target model is an automatic locking door system, which consists of three objects: a Key-card Reader(O_1), a Door(O_2) and a Timer(O_3) The behaviour of each object is depicted as in Figure 6.4. Table 6.5 represents the behaviour of each object defined by BCREs and 6.4 has brief descriptions of the symbols.

This door system behaves as follows. The door is open after the key-card object is inserted to the Key-card Reader object. Then, the door is locked once the door is open and closed. If a timeout event comes from the Timer object before the door is open, then the door is automatically locked.

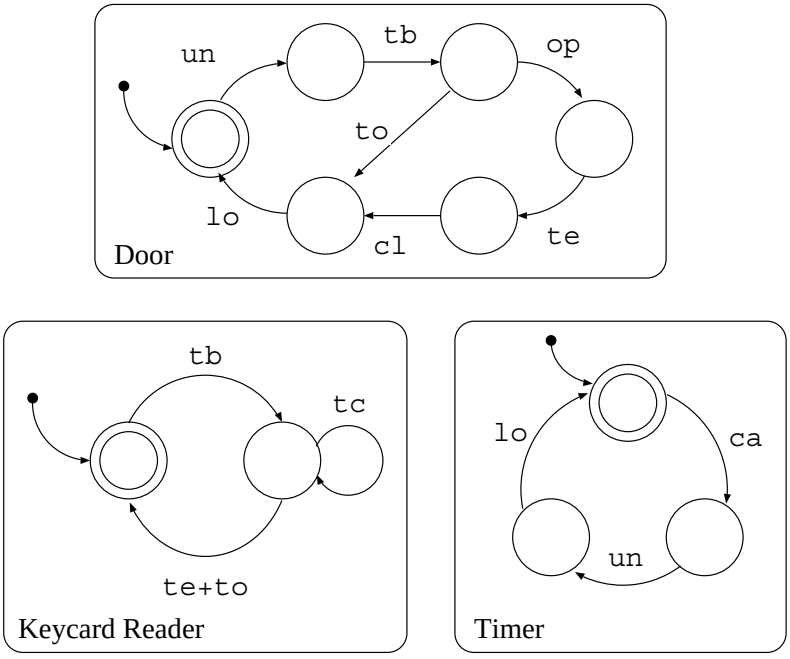


Figure 6.4: Concurrent objects in the automatic locking door system

6.2.2 Transformation

Let us begin by transforming $O_1 [\ ] O_2$. Note that in the remainder of this section, $\vdash \bot + \alpha = \alpha + \bot = \alpha$ is used without being referred to to eliminate $\bot$ in the expressions.

In a thread-extractable form on a set of communication symbols $s \equiv \{un, lo\}$ that are between O_1 and O_2 is derived as follows. By Theorem 5.1, $\vdash O_1 = \bot^*.un.O_1'O_1 + \bot^*$ where $\vdash O_1' = tb.(to + op.te.cl).lo$. Since it clearly follows that $\vdash O_1'O_1 = (tb.(to + op.te.cl)).lo.O_1$, a thread-extractable form of O_1 is derived as

$$O_{11} \equiv \bot^* un.O_{12} + \bot^*$$

ca	Key-card is inserted
un	Unlock the door
lo	Lock the door
tb	Start Timer
to	Timeout
te	Stop Timer
tc	Timer Clock
op	Open the door
cl	Close the door

Table 6.4: Symbol Descriptions

$$\begin{aligned} O_1 &\equiv (un.tb.(to + op.te.cl).lo)^* \\ O_2 &\equiv (ca.un.lo)^* \\ O_3 &\equiv (tb.tc^*.(to + te))^* \end{aligned}$$

Table 6.5: Object definitions in BCREs

$$O_{12} \equiv tb.(to + op.te.cl).lo.O_{11}$$

where $\vdash O_1 = O_{11}$. In a similar way,a thread-extractable form between O_2 and O_3 it can be found that

$$\begin{aligned} O_{21} &\equiv ca.un.O_{22} + \perp^* \\ O_{22} &\equiv \perp^*.lo.O_{21} \end{aligned}$$

where $\vdash O_2 = O_{21}$. Next, we calculate a thread-extractable form of $O_1 [\ ] O_2$. Assume that $s \equiv \{un, lo\}$, then it follows that

$$\begin{aligned} \vdash O_1 [\ ] O_2 &= O_{11} [\, s \,] O_{21} \\ &= (\perp^*.un.O_{12} [\, s \,] ca.un.O_{22}) \\ &\quad +(\perp^* [\, s \,] ca.un.O_{22}) \\ &\quad +(\perp^*.un.O_{12} [\, s \,] \perp^*) \\ &\quad +(\perp^* [\, s \,] \perp^*) \end{aligned}$$

$$= \quad ca.un.(O_{12} \,[\, s \,]\, O_{22}) + \bot^*$$

In a similar way, it is derived that

$$\vdash O_{12} \,[\, s \,]\, O_{22} = tb.(to + op.tf.cl).lo.(O_{11} \,[\, s \,]\, O_{21})$$

It is obvious that $O_{11} \,[\, s \,]\, O_{21}$ and $O_{12} \,[\, s \,]\, O_{22}$ are thread-extractable forms of $O_1 \,[\ ]\, O_2$.

Now, we derive a thread-extractable form of $O_1 \,[\ ]\, O_2$ and O_3 from a set of communication symbols: $s' \equiv \{tb, to, te\}$ By Lemma 5.1, a thread-extractable form of $O_1 \,[\ ]\, O_2$ is easy to obtain as P_1 as follows.

$$\begin{aligned} P_1 &\equiv ca.un.tb.P_2 + \bot^* \\ P_2 &\equiv \bot^*.to.P_3 + op.te.P_4 \\ P_3 &\equiv lo.ca.un.tb.P_2 + lo \\ P_4 &\equiv cl.lo.ca.un.tb.P_2 + cl.lo \end{aligned}$$

Similarly, a thread-extractable form of O_3 can be derived as O_{31}:

$$\begin{aligned} O_{31} &\equiv \bot^*.tb.O_{32} + \bot^*.tb.O_{33} + \bot^* \\ O_{32} &\equiv tc^*.to.O_{31} \\ O_{33} &\equiv tc^*.te.O_{31} \end{aligned}$$

Then according to Lemma 5.1, a thread-extractable form of $(O_1 \,[\ ]\, O_2) \,[\ ]\, O_3 = P_1 \,[\, s' \,]\, O_{31}$ can be derived as follows.

$$\begin{aligned} \vdash \quad & P_1 \,[\, s' \,]\, O_{31} = ca.un.tb.(P_2 \,[\, s' \,]\, O_{32}) \\ & + ca.un.tb.(P_2 \,[\, s \,]\, O_{33}) + \bot^* \\ \vdash \quad & P_2 \,[\, s' \,]\, O_{32} = tc^*.to.(P_3 \,[\, s' \,]\, O_{31}) \\ \vdash \quad & P_2 \,[\, s' \,]\, O_{33} = (op \| tc^*).te.(P_4 \,[\, s' \,]\, O_{31}) \\ \vdash \quad & P_3 \,[\, s' \,]\, O_{31} = lo.ca.un.tb.(P_2 \,[\, s' \,]\, O_{32}) \\ & + lo.ca.un.tb.(P_2 \,[\, s' \,]\, O_{33}) + lo \\ \vdash \quad & P_4 \,[\, s \,]\, O_{31} = cl.lo.ca.un.tb.(P_2 \,[\, s' \,]\, O_{32}) \\ & + cl.lo.ca.un.tb.(P_2 \,[\, s' \,]\, O_{33}) + cl.lo \end{aligned}$$

By applying Procedure 5.1, the state machine as in Figure 6.5 can then be obtained. The language of this state machine is

$$\begin{aligned} & ca.un.tb.(tc^*.to.lo + (tc^* \| op).te.cl.lo + \\ & (tc^*.to.lo.ca.un.tb + (tc^* \| op).te.cl.lo.ca.un.tb)^*) \end{aligned}$$

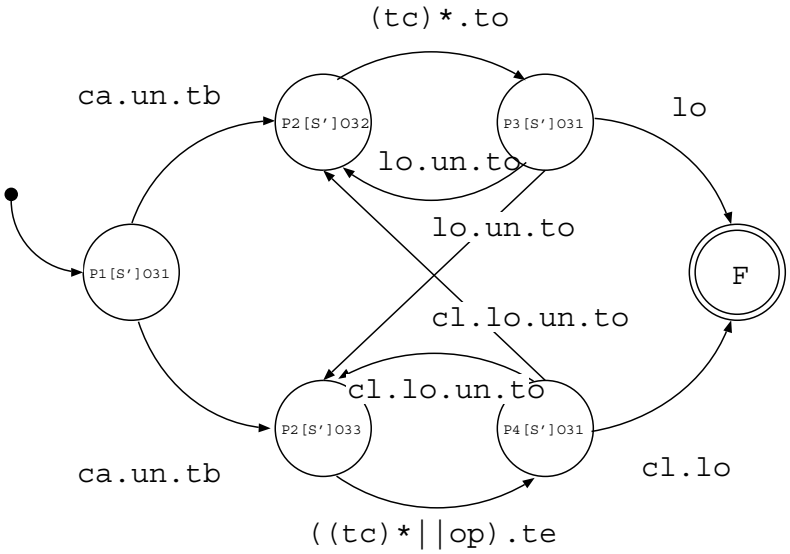


Figure 6.5: Thread-extractable form of $(O_1 [\ ] O_2) [\ ] O_3$

; this is a concurrent thread model. By applying our axiom system for compaction of the state machine, it is then represented by

$$(ca.un.tb.(tc^*.to + (tc^*||op).te.cl).lo)^*$$

Figure 6.6 depicts a concurrent thread model corresponding to this expression.

6.3 PCM device driver

In this section we show a PCM device driver development to ascertain how our approach works in the development of a real application. We also set out how to map a concurrent thread model into real source code in C. The mapping process is heuristic; however, it illustrates that an actual implementation based on a concurrent thread model is a reality.

The target driver is a synthesiser of a PCM data stream. It synthesises some PCM data on the fly so that some PCM channels can be played through just a single digital to analog converter called a DAC or D/A converter. The PCM channel is an abstraction of a PCM data stream in which the volume and frequency can be controlled.

6.3.1 Environment

It is assumed that the target system for this driver has the following hardware.

- A single CPU

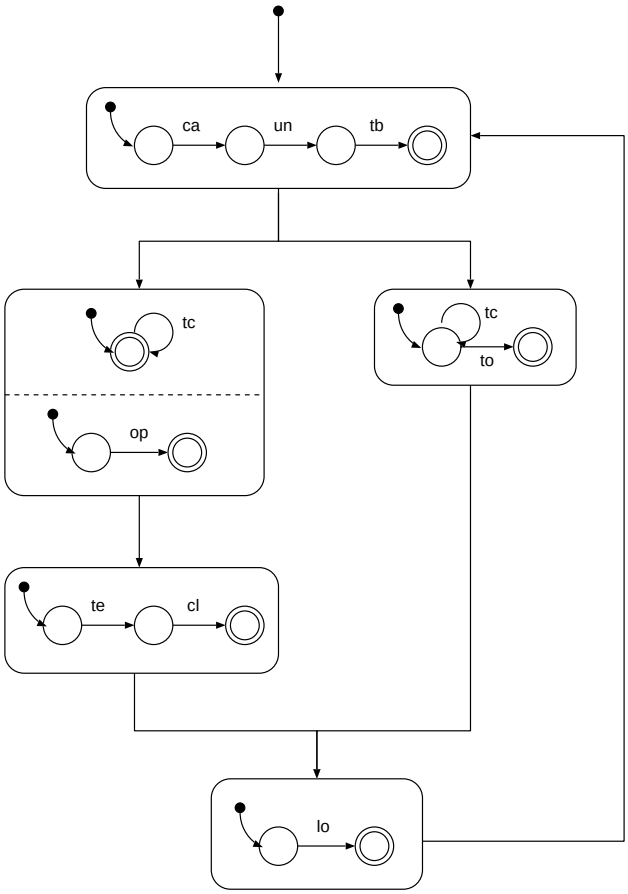


Figure 6.6: Concurrent Thread Model for $(O_1 [\ ] O_2) [\ ] O_3$

- A hardware clock oscillator

- A digital to analog converter (DAC)

In addition, it is assumed that an operating system that has at least the following functions is running on the system.

- I/O function
  This is used to write values from the driver to the DAC.

- Interrupt handler
  This function is used to notify an event from the hardware clock to software.

- Semaphore
  This is used to implement the mutual exclusion of threads in the driver software.

We implement this driver in C language.

6.3.2 Specification

The requirement specification for the PCM driver software is as follows.

- It can play and stop up to 3 PCM channels concurrently.
- The frequency can be changed for each channel.
- The volume can be changed for each channel.
- Both frequency and volume can be changed on the fly.

The application interface (API) of the driver is shown in Table 6.6

Entry	Function	Arguments
$PLAY$	Start playing	Number of channel, The top address of PCM data, The end address of PCM data
$STOP$	Stop playing	The number of a channel
$FREQ$	Change frequency	The number of a channel, Frequency
VOL	Change volume	The number of a channel, Volume

Table 6.6: Application Interface for the PCM driver.

6.3.3 Analysis

In the analysis phase, we define the classes and behaviours of objects in the system. Then we define the behaviour of objects with a concurrent object expression. We start by defining the classes.

Class definition

First, we analysed the system and defined some classes as follows.

These classes cover the following part of the system.

- API is a class for API entry of the driver. Each method in API directly corresponds to a driver entry.

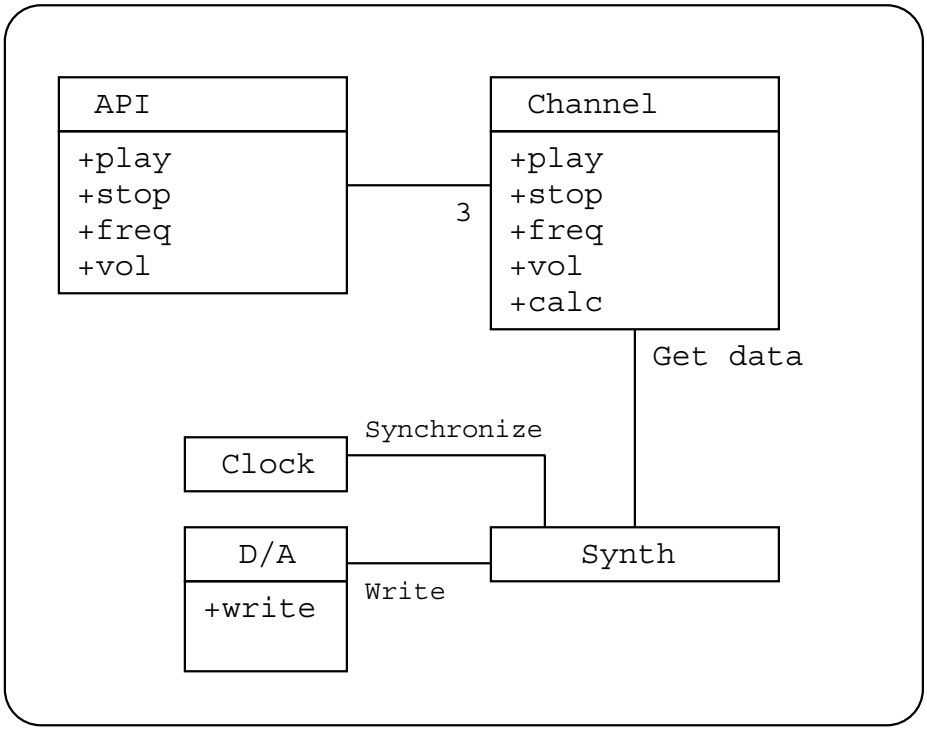


Figure 6.7: Class Diagram

- Channel is a class for each channel. Methods play, stop, freq and vol mean start playing, stop playing, change frequency and change volume for the corresponding channel. calc is a method for calculating the output value of the channel. Once calc is called, it generates the output for 1 clock time.
- D/A is an abstraction of Digital to analog converter. 'write' is a method for writing a value to the D/A converter. The real output of D/A follows the value.
- Synth is a synthesizer for 3 channels and Clock is a class for a hardware clock. Synth is synchronised with clock signals. Each time it synchronises, it synthesises an output value from the output values of all channels. Then, it writes the synthesised value to the D/A converter.
- Clock directly corresponds to the hardware clock.

Behaviour definition

The behaviour of each objects is defined as in Table 6.7. API, SYN, CLK, DAC in the Table 6.7 are instances of class API, Synth, Clock and D/A. The driver plays 3 PCM channels concurrently, so three instances of class Channel are required. Object CH_0,CH_1 and CH_2, respectively, represent these instances.

Each action represents the following functions.

- $play, stop, freq.vol$ in API correspond to API entries.
- p_i, s_i, f_i, v_i corresponds to $play, stop, freq, vol$, respectively, methods of CH_i.
- $calc$ is an action that means communication between CH_0,CH_1,CH_2. In this action, objects CH_0,CH_1,CH_2 calculate their output values using these 3 values, SYN prepares its output value for D/A.
- clk corresponds to the event from the clock.
- $write$ corresponds to the write method of class D/A.

Object name	Behaviour definition in BCREs
API	$(play.(p_0 + p_1 + p_2) + stop.(s_0 + s_1 + s_2) + freq.(f_0 + f_1 + f_2) + vol.(v_0 + v_1 + v_2))^*$
CH_0	$(p_0 + s_0 + f_0 + v_0 + calc)^*$
CH_1	$(p_1 + s_1 + f_1 + v_1 + calc)^*$
CH_2	$(p_2 + s_2 + f_2 + v_2 + calc)^*$
SYN	$(clk.calc.write)^*$
CLK	clk^*
DAC	$write^*$

Table 6.7: Behaviour of objects

The expression in Table 6.7 represents the following behaviour of the object.

- Object API invokes an appropriate channel after one of the API entries is called. For example, one of play methods of the channel objects (p_0, p_1 or p_2) is called after an API entry PLAY($play$) is called.
- Objects CH_0, CH_1, CH_2 repeatedly start playing, stop playing, change frequency, change volume or calculate their output value.
- SYN synchronises with a clock event (clk) from the hardware clock, then calculates the output value ($calc$) and sends it to the D/A converter using the $write$ method.

- CLK repeatedly generates a clock event.
- DAC repeatedly accepts the $write$ action from SYN.

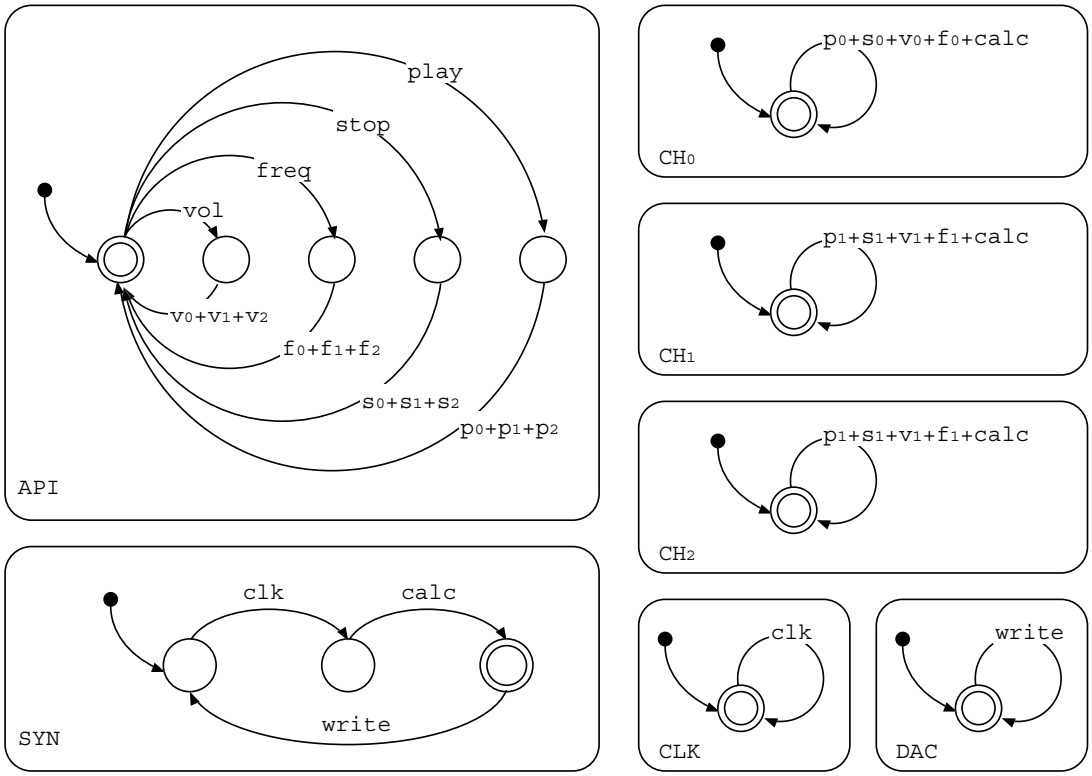


Figure 6.8: Concurrent object model for PCM device driver

Analysis Model

The design model for the PCM device driver is defined as the following concurrent object expression.

$$API[\ ]CH_0[\ ]CH_1[\ ]CH_2[\ ]SYN[\ ]CLK[\ ]DAC$$

where we use the name of an object as an abbreviation of an actual expression. For example, $CLK[\ ]DAC$ simply means $clock^*[\ ]write^*$.

6.3.4 Design and implementation

The first thing to do after the analysis phase is to derive a concurrent thread model from the concurrent object model of our PCM device driver. We omit details of the transformation here.

A concurrent thread model for a PCM device driver can be obtained as follows.

$$(play.(p_0+p_1+p_2)+stop.(s_0+s_1+s_2)+freq.(f_0+f_1+f_2)+vol.(v_0+v_1+v_2))^* \,||\, (clk.calc.write)^*$$

Figure 6.3.4 depicts this concurrent thread model.

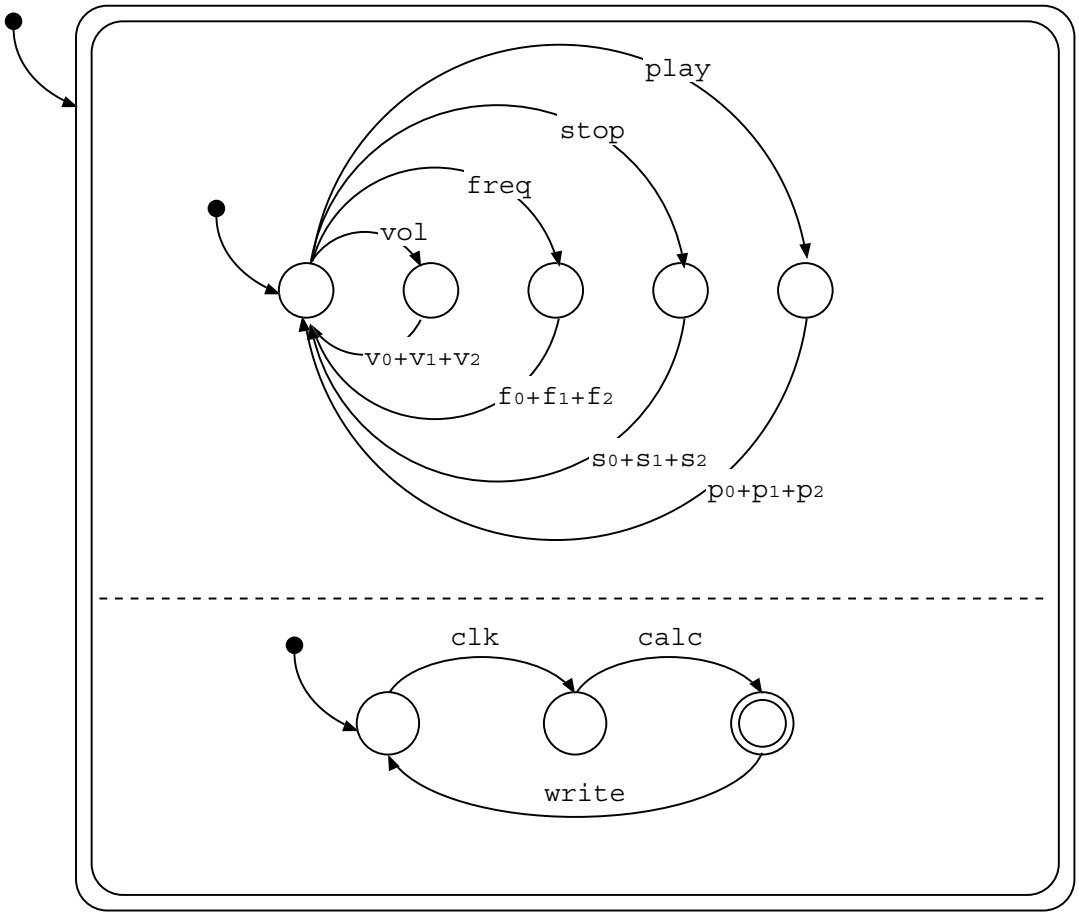


Figure 6.9: Concurrent thread model for PCM device driver

Next, we implement our PCM device driver based on this concurrent thread model. Before we implement C code, we divide the concurrent thread model into some fragments that can be directly mapped into C source code.

Extracting μ-threads

μ-threads is a fragment of a concurrent thread model that can be directly mapped into functions in C. In this development, μ-thread is roughly defined as follows.

μ-threads

- μ-thread is a pair of an action initiated by an external event and an succeeding action sequence that does not contain any external events.

- μ-threads are executed exclusively if they belong to the same thread.

where an external event is an action that indicates an incoming event from outside of the software.

We extracted μ-threads from the concurrent thread model as 6.8. The method of how to extract μ-threads is heuristic, as follows. First, we obtain μ-threads from

$$(play.(p_0 + p_1 + p_2) + stop.(s_0 + s_1 + s_2) + freq.(f_0 + f_1 + f_2) + vol.(v_0 + v_1 + v_2))^*$$

In this thread $play, stop, freq, vol$ are the external events because these actions clearly correspond to the API interface of the PCM device driver. All occur when one of the API entries is called from a client of the driver software. According to the definition of μ-threads, we can extract 4 μ-threads $PLAY, STOP, FREQ, VOL$ as shown in Table 6.8.

Name	Expression
$PLAY$	$play.(p_0 + p_1 + p_2)$
$STOP$	$stop.(s_0 + s_1 + s_2)$
$FREQ$	$freq.(f_0 + f_1 + f_2)$
VOL	$vol.(v_0 + v_1 + v_2)$
CLK	$clk.calc.write$

Table 6.8: μ-threads

Then, on the other hand, for thread $(clk.calc.write)^*$, the action clk is regarded as the outside event. Since clk corresponds to an event from the hardware clock, it comes from outside the driver. We can extract only one sub-thread CLK as shown in Table 6.8

Now, we can directly implement 5 μ-threads as in the Table 6.8 as C functions. See the Appendix to view the actual implementation in C code. The names of functions in the code are the same as the names of threads in Table 6.8.

Interface for external event

Since μ-thread is always invoked by an external event, there must be an interface that receives the external event on the μ-thread.

Since μ-threads $PLAY, STOP, FREQ$ and VOL correspond to API entries, function entries themselves are regarded as the interface for external events. Then the arguments of these

functions are implemented according to the API specification. For example, API VOL has two arguments. One is the number of channels and the other is the amount of volume. The definition of function VOL is as follows.

```
void VOL(int ch, int value);
```

μ-thread CLK is synchronised with hardware clock signals from the object CLK. We suppose that the hardware clock interrupts the operating system and the function corresponding to μ-thread CLK is called by the operating system. The address of CLK is registered as an interrupt handler for the hardware clock event when the driver is initialised. (Note that this initialisation code is omitted in the sample source code.)

Implement the body of μ-threads

The inside of the functions should be implemented considering the meaning of the μ-threads. Each function is filled with proper code that realizes the behaviour of μ-threads.

For example, VOL is a thread that behaves as $vol.(v_0 + v_1 + v_2)$. We implement the body of the function as follows.

```
void VOL(int ch, int value)
{
  volume[ch] = value;
}
```

The variable 'volume' is an array that stores the volume of 3 channels. This code represents the behaviour of $(v_0 + v_1 + v_2)$ but the code may seem to be different from the behaviour of $(v_0 + v_1 + v_2)$. `volume[ch] = value` is regarded as an abbreviation of the following code.

```
switch(ch){
case 0:
  volume[0] = value;
  break;
case 1:
  volume[1] = value;
  break;
case 2:
  volume[2] = value;
```

```
  break;
default:
  /* ERROR */
  break;
}
```

This code intuitively corresponds to $(v_0 + v_1 + v_2)$.

Implement the mutual exclusion

According to the concurrent thread model, threads $PLAY, STOP, FREQ$ and VOL should not be executed concurrently. However, we implement these 4 ω threads as functions so they are capable to be executed concurrently if the external events come again before the functions finish their process. To prevent such a situation, a binary semaphore can be used to make certain μ-threads are executed exclusively.

Some code for handling semaphores is added to the functiosn $PLAY, STOP, FREQ$ and VOL as follows. `ENTER(sem)` is inserted at the beginning of the function and `LEAVE(sem)` is also inserted at the end of function. `sem` is a binary semaphore. `ENTER` is a function that turns the semaphore up and the caller thread of this function enters the critical section. `LEAVE` is also a function turns the semaphore down and the caller thread leaves the critical section. Only one thread can enter the critical section; so a thread is blocked if the semaphore is up when it enters. The thread waits until the semaphore is down. For instance, function VOL is implemented as follows.

```
void VOL(int ch, int value){
  ENTER(sem);
  volume[ch] = value;
  LEAVE(sem);
}
```

Similarly, ENTER and LEAVE are added to $PLAY, STOP$ and $FREQ$ functions. Finally, $PLAY, STOP, FREQ$ and VOL are always exclusively executed. Then, the implementation is finalised and the driver correctly follows the behaviour represented by the concurrent thread model.

Chapter 7

Related Works

This section compares our research with works from two points of view. First, as an object-oriented design method, we compare our approach with two real-time embedded software development methods: OCTOPUS method[4] and the SES (synchronized execution sequences) approach[3], which use thread-based models as their software design model. Second, as a logical basis for concurrent systems, we compare our BCREs with CREs[2] and several process theories[14, 15, 17, 16]

7.1 OCTOPUS Method

The OCTOPUS method[4] is a development method for real-time software. Similar to our transformation method, it proposes a way to extract concurrent components from an object-oriented software analysis model. However, there are many differences between the two approaches . An overview of OCTOPUS's procedure of extracting concurrent components follows.

1. It specifies all possible communication between objects and makes a communication graph of the objects.

2. It specifies as synchronous or asynchronous all communications in the graph.

3. It divides all objects into object groups, in which each group can involve objects related with synchronous communications with each other, but not asynchronous ones.

Then, the groups obtained by the above procedure are mapped into the task, thread or processes provided by operating systems, and then they are concurrently executed in the real software. Note that the groups deduced by this procedure are not deterministic. Further, a high level of skill is required of the developer for deciding the object groups.

7.2 SES approach

The SES approach is a software design method based on the SES model for real-time embedded software. A SES model consists of a global automata and some SESs. The SES is a sequence of internal actions of the system. The global automata controls the timing of the execution of SESs mapped to each of the states of the automata. As well, the SES model satisfies the following properties

- No SESs have a blocking operation.
- Any state transition in the global automata must be initiated when all executing SESs are terminated.

This architecture of the SES model is almost similar to our concurrent thread model; the only difference is that any state transition in the SES model is performed by deterministic events, while there is no explicit transition events in our concurrent thread model.

7.3 Concurrent Regular Expressions

As we discussed in Chapter 2, our BCREs are based on Concurrent Regular Expressions (CREs). The difference between BCREs and CREs can be summarized as below.

- BCREs do not have *alpha-closure* and *renaming* operators.
- The semantics of the synchronous composition operator is different.

CREs were proposed as a description language of the behaviour of Petri nets[12]. Since an infinite number of concurrent sequences are observable from Petri nets, there is an operator called alpha-closure that can handle infinite concurrency. As well, CREs have a operator that renames any symbols in CREs. In the definition of BCREs, these two operators are omitted for the following reasons. For the interleaving closure operator, we assumed that a real system can be modelled with a finite number of objects or threads. Therefore we decided to eliminate the alpha-closure from our BCREs. For the renaming operator, this can be used for hiding symbols and observing partial behaviour of a system. We do not take this operator into BCREs in order to simplify our transformation method. Although renaming is not supported, our concurrent object and thread models can be described as set out in Chapter 4. However, we are undecided whether the renaming operation is actually necessary for modelling concurrent object or thread models. To clarify this problem, we will need more application examples of BCREs through

real system developments. We will add this operator to BCREs if it should become apparent that renaming is a useful or essential idea.

As for the difference of semantics, the following is the definition of synchronous composition in CREs.

$$L(\alpha\,[\ ]\,\beta) = \{\omega \mid \omega/\sigma(\alpha) \in \alpha, \omega/\sigma(\beta) \in \beta\}$$

where $\sigma(\alpha)$ denotes a set of symbols in α. ω/σ denotes the restriction of the sequence ω to the symbol of σ. (Note that more precisely, $\omega/ \in (\sigma(\alpha) \cup \sigma(\beta))^*$ is required as a condition.) On the other hand, the synchronous composition in BCREs is defined as follows.

$$L(\alpha\,[\,s\,]\,\beta) = \{\omega \mid \omega/\sigma(\alpha) \cup s \in \alpha, \omega/\sigma(\beta) \cup s \in \beta\}$$

The difference between these two definitions is that $[\ ]$ is not distributive. Namely, it does not follow that

$$\vdash (\alpha + \beta)\,[\ ]\,\gamma = (\alpha\,[\ ]\,\gamma) + (\beta\,[\ ]\,\gamma)$$

whereas the $[\,s\,]$ operator is distributive, that is,

$$(\alpha + \beta)\,[\,s\,]\,\gamma \quad = \quad \alpha\,[\,s \cup (\sigma(\beta) \cap \sigma(\gamma))\,]\,\gamma + \beta\,[\,s \cup (\sigma(\alpha) \cap \sigma(\gamma))\,]\,\gamma$$

follows as seen as Axiom C_5. Distributiveness is very important for complete axiomatisation of BCREs. Hence, we proposed a different definition for synchronous composition. We understand that complete axiom systems for CREs have never been studied.

7.4 Process Theory

A number of theories for concurrent systems have been proposed, for example, CSP[14], CCS[15], π-calculus[17] and ACP[16]. They are called *process calculi*. Strictly speaking an ACP is called process algebra since its semantics are based on an algebraic method by a set of axioms. In this thesis, we simply call these approaches *process theory*.

Process theory handles several kinds of equivalence in concurrent systems. Equivalence of BCREs is similar to the language equivalence of automata. In process theory, such equivalence is called *trace equivalence*. It seems possible to define our model and transformation method based on process theory. However, there is no process theory that is sufficient to treat our model and transformation method. They are suitable to denote our concurrent object model but not for our concurrent thread model due to some limitations in their syntax. Moreover, even if we choose the process theory, they could not be used as an immediate solution of our transformation

problem. No systematic method, similar to our model transformation method in Chapter 5, has been discussed in process theory. Thus, in this paper we have solved the problem with BCREs. In the remainder of this section, we explain in more detail why BCREs are preferred as our formalism instead of process theory. The reasons why we chose BCREs follow.

1. BCREs consist of simple syntax and semantics based on trace equivalence.

2. The syntax of BCREs is appropriate for our concurrent thread model.

3. BCREs explicitly discriminate concurrent behaviour with communication from that without communication utilizing the operators $\|$ and $[\,s\,]$.

In respect of the first reason, we defined consistency between concurrent objects and concurrent threads based on trace equivalence, which is similar to language equivalence. Internal states in a system are ignored under this equivalence; thus, in our approach, we do not need to explicitly model states of a system. However, in process theory, a system is generally modelled by a set of processes where each process explicitly represents a state in a system. Such state-aware modelling is redundant in our approach.

Readers may think it is possible to define consistency between concurrent objects and threads based on an another style of equivalence, for example, the strong or weak bisimulality of process theory. Such could be the case, however, in here we have adopted trace equivalence as the first step to challenge the model transformation problem, since trace equivalence is basic and the most simple equivalence of behaviour, and thus satisfactory for tackling the problem.

In respect of the second reason, we defined a concurrent thread model to obtain information of how many, when, and what kind of threads are executed concurrently. Let us consider a BCRE $((x+y)z\|pq*)r$. In accord with this expression, it is easily understood that the following actions are executed concurrently.

1. x or y is executed, then z is executed.

2. Iteration of q occurs after p.

Then, finally r is executed. In contrast, it is difficult to model the system with explicit denotation of such information using the general syntax of process calculus. We see that the following processes define behaviour similar to $((x+y)z\|pq*)r$ in the manner of CCS or π-calculus.

$$\begin{aligned} P_0 &= P_1|P_2 \\ P_1 &= x.P_3 + y.P_3 \\ P_2 &= p.P_4 + p.P_5 \end{aligned}$$

$$
\begin{aligned}
P_3 &= z.P_5 \\
P_4 &= q.P_4 + p.P_5 \\
P_5 &= r.\delta
\end{aligned}
$$

where δ denotes a terminating process. $P_1|P_2$ represents concurrent process with P_1 and P_2. It is hard to obtain information like (a) and (b) from this process definition. According to the syntax of CCS and π-calculus, any successive process or action is not allowed for concurrent processes. $P_1|P_2$ should appear at the end of right-hand side in a definition of process. As for CSP, whose syntax is little different from CCS in several areas, there is also no way to denote successive actions after a concurrent process definition. Therefore, neither CCS, π-calculus nor CSP can denote a explicit process or action that is executed just after concurrent processes, like $(x\|y)z$ in BCREs. Furthermore, loop or iteration of actions are denoted as recursive processes; thus, it is difficult to understand which set of processes is considered as a set of closed processes t constituting the loop or iteration. On the other hand, BCREs have the closure ($*$) operator for loops of actions so that loop behaviour can also be denoted as simply as a sequence. Compared with the other process theories, ACP is based on a reasonably different syntax closer to BCREs. It allows a notation like $(x\|y)z$ similar to BCREs but it does not have a closure operation. (Note that there is a system BPA* [18] that is a subset of ACP with an extending closure operation. A complete axiom system for strong equivalence of BPA* has been presented[11]. However, BPA* has no algebraic operator for concurrency and communication.)

In respect of the third reason; it is very important for our approach to distinguish concurrency between objects from between threads. In BCREs, these two types of concurrency can be handled separately by $[\,s\,]$ and $\|$, respectively. Give the grace of such separation, our transformation method has been precisely defined as a procedure that eliminates all $[\,s\,]$ operators and replaces these with equivalent expressions defined with the $\|$ operator. By contrast, CCS and π-calculus use the operator $\|$ for concurrent processes without regard to communication. As for ACP, there are two different operations called 'free merge' and 'merge with communication'; The latter is for concurrent processes with communication and the former is for the other processes. However, the operation 'merge with communication' is defined as an extension of 'free merge' so two different operators cannot be used simultaneously. CSP has the operator $\|$ for concurrency with communication, and the operator $|||$ for concurrency without communication.

Chapter 8

Conclusion and Future Work

In this thesis, we investigated how to obtain information about concurrency: 'How many and what kind of thread are executed concurrently' from an object-oriented behavioural model. We presented the two orthogonal models for object-oriented and concurrent sequence-based aspects; the concurrent object model and the concurrent thread model, respectively. Then we proposed a systematic method for transformation from a concurrent object model to a concurrent thread model. Since the concurrent thread model explicitly represents information of concurrency, by using our transformation method it is possible to systematically obtain information of concurrency from object-oriented models.

In Chapter 2, we presented basic concurrent regular Expressions (BCREs); which are an extension of two operators to the regular expressions for handling communication and concurrency. In Chapter 3, we described an axiom system F_c for BCREs. We also proved that the axiom system was sound and complete. Language equivalence of BCREs can be proved by this system. In Chapter 4, we formalised the concurrent object and concurrent thread models using BCREs. In Chapter 5, we proposed a method of how to transform a concurrent object model to a concurrent thread model. This method is sound; that is, any behaviour represented in a model is preserved after transformation. Moreover, we proved that our transformation method can handle any concurrent object model that consists of two objects. An issue with this method is that some models that contain more than three objects cannot be transformed by our method. For completeness of transformation, we hope to extend our method to more than three object models. In Chapter 6, we demonstrated our transformation of three example models: a media player, an auto-locking door and a PCM device driver system. All models in these examples were successfully transformed . Especially in the PCM device driver system example, we showed a heuristic mapping from the concurrent thread model to C source code. This result showed that it is possible to implement real software that is based on our concurrent

thread model.

Future work will include completing a model transformation method for three or more objects. We plan to implement a system for automatic transformation because it is hard to perform the transformation manually.

We also plan to introduce specific real-time information into our concurrent thread model so that we can discuss the satisfiability problems of real-time constraints.

Appendix A

Proof for Soundness of F_C

This appendix provides the proof for soundness of F_c. It is enough for proving soundness to show all the axioms in F_c are valid. However, here we only prove that the axioms C_4 and C_5 are vaild. For the other axioms, it is easy to prove straightforward from the semantics of BCREs, therefore we omit the detailed proofs for them. In this appendix, We begin with some lemmas that are useful in the succeeding proofs. Then, we provide the proofs for C_4 and C_5.

A.1 Preliminary

LEMMA **A.1**

- $\sigma(\alpha\beta) = \sigma(\alpha) \cup \sigma(\beta)$ *if* $L(\alpha) \neq \phi$ *and* $L(\beta) \neq \phi$
- $\sigma(\alpha + \beta) = \sigma(\alpha) \cup \sigma(\beta)$

$\sigma(\alpha)$ means a set of symbols which occur in $L(\alpha)$. It is almost clear from the definition of σ and L. We omit a detailed proof. In the remainder of this section, we will use this lemma without explicit reference to.

LEMMA **A.2**

Assume that $\omega/\sigma(\alpha) \cup s \cup t \in L(\alpha)$ *where* s *and* t *are sets of symbols,* α *is a BCRE and* ω *is a sequence. Then* $t \cap \sigma(\omega) = \phi$ *implies* $\omega/\sigma(\alpha) \cup s \cup t = \omega/\sigma(\alpha) \cup s$.

PROOF

$t \cap \sigma(\omega) = \phi$ means that ω has no symbol among t. Hence, t will not effect the result of the restriction of ω. Then the lemma clearly holds. □

LEMMA **A.3**

$\omega/\{x\} \cup \sigma(\alpha) \cup s \in L(\alpha)$ implies $x \notin \sigma(\omega)$ or $x \in \sigma(\alpha)$. where x is a symbol, s is a set of symbol, α is a BCRE and ω is a sequence.

PROOF

Assume that $\omega/\{x\} \cup \sigma(\alpha) \cup s \in L(\alpha)$. If $x \in \sigma(\omega/\{x\} \cup \sigma(\alpha) \cup s)$ then $\sigma(\omega/\{x\} \cup \sigma(\alpha) \cup s) \subseteq \sigma(\alpha)$ clearly holds. Hence, $x \in \sigma(\alpha)$. On the other hand, if $x \notin \sigma(\omega/\{x\} \cup \sigma(\alpha) \cup s)$, then it is obvious that $x \notin \sigma(\omega)$.Therefore, Lemma A.3 holds. □

LEMMA **A.4**

$L(x(\alpha\,[\,s\,]\,y\beta)) \subseteq L(x\alpha\,[\,s\,]\,y\beta)$ where $x \neq y$ and $x \notin \sigma(\beta) \cup s$.

PROOF

Assume that $x \neq y$, $x \notin \sigma(\beta) \cup s$ and $\omega \in L(x(\alpha\,[\,s\,]\,y\beta))$. Then, according to the definition of L, some ω' exists such that $\omega = x\omega'$ such that

$$\omega' \in (\sigma(\alpha) \cup \sigma(y\beta))^*, \tag{A.1}$$

$$\omega'/\sigma(\alpha) \cup s \in L(\alpha), \tag{A.2}$$

$$\omega'/\sigma(y\beta) \cup s \in L(y\beta) \tag{A.3}$$

It clearly follows by (A.1) that

$$x\omega' \in (\sigma(x\alpha) \cup \sigma(y\beta))^* \tag{A.4}$$

Assume that $x \in \sigma(\omega')$. Then By (A.1), it clearly holds that $x \in \sigma(\alpha) \cup \sigma(\beta)$. By the assumption, $x \notin \sigma(\beta)$ holds. Thus $x \in \sigma(\alpha)$ and this implies $\omega'/\{x\} \cup \sigma(\alpha) \cup s \equiv \omega'/\sigma(\alpha) \cup s$. In opposite, assume that $x \notin \sigma(\omega')$, it is clear that $\omega'/\{x\} \cup \sigma(\alpha) \cup s \equiv \omega'/\sigma(\alpha) \cup s$. Accordingly, by (A.2), $\omega'/\{x\} \cup \sigma(\alpha) \cup s \in L(\alpha)$ follows. Therefore,

$$x\omega'/\sigma(x\alpha) \cup s \in L(x\alpha) \tag{A.5}$$

By the assumption, it is obvious that $x \notin \sigma(y\beta) \cup s$. Therefore, $x/\sigma(y\beta) \cup s$ is a empty word. Accordingly, by (A.3),

$$x\omega'/\sigma(y\beta) \cup s \in L(y\beta) \tag{A.6}$$

By (A.4), (A.5) and (A.6), $x\omega' \in L(x\alpha\,[\,s\,]\,y\beta)$. This implies

$$L(x(\alpha\,[\,s\,]\,y\beta)) \subseteq L(x\alpha\,[\,s\,]\,y\beta) \tag{A.7}$$

□

A.2 Axiom C_4

We prove that the axiom C_4 is vaild through the following four cases.

$$
\begin{array}{lllll}
C_{41} & x\alpha\,[\,s\,]\,x\beta & = & x(\alpha\,[\,s \cup \{x\}\,]\,\beta) & \\
C_{42} & x\alpha\,[\,s\,]\,y\beta & = & x(\alpha\,[\,s\,]\,y\beta) & \text{if } x \neq y,\ x \notin \sigma(\beta) \cup s,\ y \in \sigma(\alpha) \cup s \\
C_{43} & x\alpha\,[\,s\,]\,y\beta & = & x(\alpha\,[\,s\,]\,y\beta) + y(x\alpha\,[\,s\,]\,\beta) & \text{if } x \neq y,\ x \notin \sigma(\beta) \cup s,\ y \notin \sigma(\alpha) \cup s \\
C_{44} & x\alpha\,[\,s\,]\,y\beta & = & \bot & \text{if } x \neq y,\ x \in \sigma(\beta) \cup s,\ y \in \sigma(\alpha) \cup s
\end{array}
$$

Proof for C_{41}

Let ω be in $L(x\alpha\,[\,s\,]\,x\beta)$. Then by the definition of L, it follows that

$$\omega \in (\sigma(x\alpha) \cup \sigma(x\beta))^*, \tag{A.8}$$

$$\omega/\sigma(x\alpha) \cup s \in L(x\alpha), \tag{A.9}$$

$$\omega/\sigma(x\beta) \cup s \in L(x\beta) \tag{A.10}$$

By (A.9) and (A.10), it is obvious that the first symbol of ω is x. Hence, $\omega = x\omega'$ for some ω'. Since $x/\sigma(x\alpha) \cup s \equiv x$ so $x\omega'/\sigma(x\alpha) \cup s \equiv x(\omega'/\sigma(\alpha) \cup s) \in L(x\alpha)$. Therefore, $\omega'/\sigma(x\alpha) \cup s \in L(\alpha)$. In the same way, it follows that $\omega'/\sigma(x\beta) \cup s \in L(\beta)$. Thus

$$\omega'/\sigma(\alpha) \cup \{x\} \cup s \in L(\alpha) \tag{A.11}$$

$$\omega'/\sigma(\beta) \cup \{x\} \cup s \in L(\beta) \tag{A.12}$$

By (A.8), $x\omega' \in (\{x\} \cup \sigma(\alpha) \cup \sigma(\beta))^*$. Hence, ω' also in $(\{x\} \cup \sigma(\alpha) \cup \sigma(\beta))^*$. Assume that $x \in \sigma(\alpha) \cup \sigma(\beta)$ then $\omega' \in (\sigma(\alpha) \cup \sigma(\beta))^*$ clearly follows. On the other hand, in the case where $x \notin \sigma(\alpha) \cup \sigma(\beta)$, assume that $x \in \sigma(\omega')$. Then, it contradict with (A.11) and (A.12). Therefore, $x \notin \sigma(\omega')$. This implies,

$$\omega' \in (\sigma(\alpha) \cup \sigma(\beta))^* \tag{A.13}$$

By (A.11), (A.12), (A.13), according to the definition of L,

$$\omega' \in L(\alpha\,[\,\{x\} \cup s\,]\,\beta)$$

Thus $\omega \equiv x\omega' \in L(x(\alpha\,[\,\{x\} \cup s\,]\,\beta)$ follows. Therefore, it follows that

$$L(x\alpha\,[\,s\,]\,x\beta) \subseteq L(x(\alpha\,[\,\{x\} \cup s\,]\,\beta)) \tag{A.14}$$

Next, let ω be in $L(x(\alpha\,[\,s \cup \{x\}\,]\,\beta))$. Then $\omega \equiv x\omega'$ for some ω' and

$$\omega' \in (\sigma(\alpha) \cup \sigma(\beta))^* \tag{A.15}$$

$$\omega'/\sigma(\alpha) \cup s \cup \{x\} \in L(\alpha) \tag{A.16}$$

$$\omega'/\sigma(\beta) \cup s \cup \{x\} \in L(\beta) \tag{A.17}$$

By (A.15),

$$x\omega' \in (\{x\} \cup \sigma(\alpha) \cup \sigma(\beta))^* \tag{A.18}$$

By (A.16) and the definition of L, $x(\omega'/\sigma(\alpha) \cup s \cup \{x\}) \in L(x\alpha)$. Therefore, since $x/\sigma(\alpha) \cup s \cup \{x\} \equiv x$, it follows that

$$x\omega'/\sigma(\alpha) \cup s \cup \{x\} \in L(x\alpha) \tag{A.19}$$

In the same way,

$$x\omega'/\sigma(\beta) \cup s \cup \{x\} \in L(x\beta) \tag{A.20}$$

By (A.18)(A.19) and (A.20), and the definition of L, $x\omega' \in L(x\alpha\,[\,s\,]\,x\beta)$ follows. Therefore,

$$L(x(\alpha\,[\,s \cup \{x\}\,]\,\beta) \subseteq L(x\alpha\,[\,s\,]\,x\beta) \tag{A.21}$$

By (A.14) and (A.21),

$$x(\alpha\,[\,s \cup \{x\}\,]\,\beta) = x\alpha\,[\,s\,]\,x\beta$$

that is, C_{41} is valid. □

Proof for C_{42}

Assume that $x \neq y$, $x \notin \sigma(\beta) \cup s$ and $y \in \sigma(\alpha) \cup s$. Let ω be in $L(x\alpha\,[\,s\,]\,y\beta)$, that is,

$$\omega \in (\sigma(x\alpha) \cup \sigma(y\beta))^*, \tag{A.22}$$

$$\omega/\sigma(x\alpha) \cup s \in L(x\alpha), \tag{A.23}$$

$$\omega/\sigma(y\beta) \cup s \in L(y\beta) \tag{A.24}$$

Suppose that the first symbol of ω is y. Then ,by the assumption $y \in \sigma(\alpha) \cup s$, the first symbol of $\omega/\sigma(\alpha) \cup s$ is y. It contradict with $A.23$. Hence, the first symbol of ω is not y. If the first symbol of ω is not x, it also contradict with $A.23$. This implies that some ω' exist such that $\omega \equiv x\omega'$ holds.

From (A.23), $x\omega'/\{x\}\cup\sigma(\alpha)\cup s \in L(x\alpha)$ holds. According to the definition of L, $\omega'/\{x\}\cup\sigma(\alpha)\cup s \in L(\alpha)$. By (A.23) and Lemma A.3, $x \notin \sigma(\omega')$ or $x \in \sigma(\alpha)$. In the both cases, it clearly follows from (A.23) that

$$\omega'/\sigma(\alpha)\cup s \in L(\alpha) \tag{A.25}$$

Since $x \neq y$ and $x \notin \sigma(\beta)\cup s$ by the assumption, $x/\sigma(y\beta)\cup s$ is an empty word. Hence, $x\omega'/\sigma(y\beta)\cup s \equiv (x/\sigma(y\beta)\cup s)(\omega'/\sigma(y\beta)\cup s) \equiv \omega'/\sigma(y\beta)\cup s$. Then by (A.24)

$$\omega'/\sigma(y\beta)\cup s \in L(y\beta) \tag{A.26}$$

Be reminded again that $x \notin \sigma(\omega')$ or $x \in \sigma(\alpha)$ holds. Then it is clear from (A.22) that

$$\omega' \in (\sigma(\alpha)\cup\sigma(y\beta))^* \tag{A.27}$$

By (A.25), (A.26), (A.27) and the definition of L, $\omega' \in L(\alpha\,[\,s\,]\,y\beta)$ holds. Therefore, $x\omega' \in L(x(\alpha\,[\,s\,]\,y\beta))$. Hence, $L(x\alpha\,[\,s\,]\,y\beta) \subseteq L(x(\alpha\,[\,s\,]\,y\beta))$. On the other hand, by the assumption and Lemma A.4, $L(x(\alpha\,[\,s\,]\,y\beta)) \subseteq L(x\alpha\,[\,s\,]\,y\beta)$ clearly holds. Consequently, $L(x(\alpha\,[\,s\,]\,y\beta)) = L(x\alpha\,[\,s\,]\,y\beta)$ follows. □

Proof for C_{43}

Assume that $x \neq y$, $x \notin \sigma(\beta)\cup s$, $y \notin \sigma(\alpha)\cup s$ and ω be in $L(x\alpha\,[\,s\,]\,y\beta)$. Then, suppose that x is the first symbol of ω. In the same way of the proof for C_{42}, it follows that

$$\omega \in L(x(\alpha\,[\,s\,]\,y\beta))$$

On the other hand, assuming that y is the first symbol of ω, then in the similar way to the proof for C_{42}, it is easy to see:

$$\omega \in L(y(x\alpha\,[\,s\,]\,\beta))$$

Thus $\omega \in L(x(\alpha\,[\,s\,]\,y\beta)) \cup L(y(x\alpha\,[\,s\,]\,\beta))$ follows. According to the definition of L,

$$L(x(\alpha\,[\,s\,]\,y\beta)) \cup L(y(x\alpha\,[\,s\,]\,\beta)) = L(x(\alpha\,[\,s\,]\,y\beta) + y(x\alpha\,[\,s\,]\,\beta))$$

Therefore,

$$L(x\alpha\,[\,s\,]\,y\beta)) \subseteq L(x(\alpha\,[\,s\,]\,y\beta) + y(x\alpha\,[\,s\,]\,\beta)) \tag{A.28}$$

Now, assume that $x \neq y$, $x \notin \sigma(\beta) \cup s$ and $y \notin \sigma(\alpha) \cup s$ again. According to the definition of L,

$$L(x(\alpha\,[\,s\,]\,y\beta) + y(x\alpha\,[\,s\,]\,\beta)) = L(x(\alpha\,[\,s\,]\,y\beta)) \cup L(y(x\alpha\,[\,s\,]\,\beta))$$

Then, by Lemma A.4,

$$L(x(\alpha\,[\,s\,]\,y\beta)) \subseteq L(x\alpha\,[\,s\,]\,y\beta)$$

By C_3 and Lemma A.4 again,

$$L(y(x\alpha\,[\,s\,]\,\beta)) \subseteq L(x\alpha\,[\,s\,]\,y\beta)$$

Hence,

$$L(x(\alpha\,[\,s\,]\,y\beta) + y(x\alpha\,[\,s\,]\,\beta)) \subseteq L(x\alpha\,[\,s\,]\,y\beta) \tag{A.29}$$

By (A.28) and (A.29),

$$L(x(\alpha\,[\,s\,]\,y\beta) + y(x\alpha\,[\,s\,]\,\beta)) = L(x\alpha\,[\,s\,]\,y\beta)$$

□

Proof for C_{44}

It is almost obvious from the definition of L that $L(x\alpha\,[\,s\,]\,y\beta) = \phi$. Therefore the axiom is valid.

A.3 Axiom C_5

This section proves the validity of the axiom C_5. Assume that $\omega \in L((\alpha + \beta)\,[\,s\,]\,\gamma)$. Then by the definition of L,

$$\omega \in (\sigma(\alpha) \cup \sigma(\beta) \cup \sigma(\gamma))^* \tag{A.30}$$

$$\omega/\sigma(\alpha) \cup \sigma(\beta) \cup s \in L(\alpha + \beta) \tag{A.31}$$

$$\omega/\sigma(\gamma) \cup s \in L(\gamma) \tag{A.32}$$

By (A.31), $\omega/\sigma(\alpha) \cup \sigma(\beta) \cup s \in L(\alpha)$ or $L(\beta)$. We consider by first for the case where $\omega/\sigma(\alpha) \cup \sigma(\beta) \cup s \in L(\alpha)$. Since $\sigma(\alpha) \cup \sigma(\beta) = \sigma(\alpha) \cup (\sigma(\beta) \cap \overline{\sigma(\alpha)})$, we have $\omega/\sigma(\alpha) \cup (\sigma(\beta) \cap \overline{\sigma(\alpha)}) \cup s \in L(\alpha)$. Then by Lemma $A.3$ for all $x \in \sigma(\beta) \cap \overline{\sigma(\alpha)}$, $x \notin \sigma(\omega)$ or $x \in \sigma(\alpha)$. By the assumption it is obvious that $x \notin \sigma(\alpha)$ because $x \in \sigma(\beta) \cap \overline{\sigma(\alpha)}$. Thus, only just $x \notin \sigma(\omega)$

holds. This implies that $\sigma(\omega) \cap (\sigma(\beta) \cap \overline{\sigma(\alpha)}) = \phi$, therefore $\sigma(\omega) - (\sigma(\beta) \cap \overline{\sigma(\alpha)}) = \sigma(\omega)$ holds. Since $\sigma(\omega) = \sigma(\alpha) \cup \sigma(\beta) \cup \sigma(\gamma)$ by (A.30), thus $\sigma(\omega) = \sigma(\omega) - (\sigma(\beta) \cap \overline{\sigma(\alpha)}) = \sigma(\alpha) \cap \sigma(\gamma)$ follows. Therefore,

$$\omega \in (\sigma(\alpha) \cap \sigma(\gamma))^* \tag{A.33}$$

Now, considering the following equation:

$$\begin{aligned} \omega/\sigma(\alpha) \cup \sigma(\beta) \cup s &= \omega/\sigma(\alpha) \cup ((\sigma(\beta) \cap \sigma(\gamma)) \cup (\sigma(\beta) \cap \overline{\sigma(\gamma)})) \cup s \\ &= \omega/\sigma(\alpha) \cup ((\sigma(\beta) \cap \sigma(\gamma)) \cup (\sigma(\beta) \cap \overline{\sigma(\gamma)}) \cap \overline{\sigma(\alpha)}) \cup s \\ &= \omega/\sigma(\alpha) \cup (\sigma(\beta) \cap \sigma(\gamma) \cap \overline{\sigma(\alpha)}) \cup (\sigma(\beta) \cap \overline{\sigma(\gamma)} \cap \overline{\sigma(\alpha)}) \cup s \end{aligned}$$

Remind that $\sigma(\omega) \cap (\sigma(\beta) \cap \overline{\sigma(\alpha)}) = \phi$, it is clear that $\sigma(\omega) \cap (\sigma(\beta) \cap \overline{\sigma(\gamma)} \cap \overline{\sigma(\alpha)})$ is also ϕ. Therefore by Lemma A.2

$$\begin{aligned} & \omega/\sigma(\alpha) \cup (\sigma(\beta) \cap \sigma(\gamma) \cap \overline{\sigma(\alpha)}) \cup (\sigma(\beta) \cap \overline{\sigma(\gamma)} \cap \overline{\sigma(\alpha)}) \cup s \\ = \ & \omega/\sigma(\alpha) \cup (\sigma(\beta) \cap \sigma(\gamma) \cap \overline{\sigma(\alpha)}) \cup s \\ = \ & \omega/\sigma(\alpha) \cup (\sigma(\beta) \cap \sigma(\gamma)) \cup s \end{aligned}$$

Hence, $\omega/\sigma(\alpha) \cup \sigma(\beta) \cup s = \omega/\sigma(\alpha) \cup (\sigma(\beta) \cap \sigma(\gamma)) \cup s$ follows. This and by (A.31) we have

$$\omega/\sigma(\alpha) \cup (\sigma(\beta) \cap \sigma(\gamma)) \cup s \in L(\alpha) \tag{A.34}$$

Since $\sigma(\gamma) \cup (t \cap \sigma(\gamma)) = \sigma(\gamma)$, $\omega/\sigma(\gamma) \cup s = \omega/\sigma(\gamma) \cup (t \cap \sigma(\gamma)) \cup s$ for any set of symbols t. Hence by (A.32),

$$\omega/\sigma(\gamma) \cup (\sigma(\beta) \cap \sigma(\gamma)) \cup s \in L(\gamma) \tag{A.35}$$

By (A.33), (A.34), (A.35) and the definition of L, $\omega \in L(\alpha\,[\,s \cup (\sigma(\beta) \cap \sigma(\gamma))\,]\,\gamma)$ holds.

On the other hand, for the case where $\omega/\sigma(\alpha) \cup \sigma(\beta) \cup s \in L(\beta)$, then in the similar way to the proof above, it is easy to see that $\omega \in L(\beta\,[\,s \cup (\sigma(\alpha) \cap \sigma(\gamma))\,]\,\gamma)$. Thus $\omega \in L((\alpha\,[\,s \cup (\sigma(\beta) \cap \sigma(\gamma))\,]\,\gamma)) \cup L((\beta\,[\,s \cup (\sigma(\alpha) \cap \sigma(\gamma))\,]\,\gamma))$. Hence,

$$L((\alpha + \beta)\,[\,s\,]\,\gamma) \subseteq L((\alpha\,[\,s \cup (\sigma(\beta) \cap \sigma(\gamma))\,]\,\gamma)) \cup L((\beta\,[\,s \cup (\sigma(\alpha) \cap \sigma(\gamma))\,]\,\gamma)) \tag{A.36}$$

Next we look at the case where $\omega \in L(\alpha\,[\,s \cup (\sigma(\beta) \cap \sigma(\gamma))\,]\,\gamma)$. By this assumption, we have

$$\omega \in (\sigma(\alpha) \cup \sigma(\gamma))^* \tag{A.37}$$

$$\omega/\sigma(\alpha) \cup (\sigma(\beta) \cap \sigma(\gamma)) \cup s \in L(\alpha) \tag{A.38}$$

$$\omega/\sigma(\gamma) \cup (\sigma(\alpha) \cap \sigma(\gamma)) \cup s \in L(\gamma) \tag{A.39}$$

From (A.37), we can obtain

$$\omega \in (\sigma(\alpha) \cup \sigma(\beta) \cup \sigma(\gamma))^* \tag{A.40}$$

By (A.37), $\sigma(\omega) \subseteq \sigma(\alpha) \cup \sigma(\gamma)$ follows. Hence, $\sigma(\omega) \cap (\sigma(\beta) \cap \overline{\sigma(\alpha)} \cap \overline{\sigma(\gamma)}) = \phi$ clearly holds. Therefore, by Lemma A.2

$$\begin{aligned}
\omega/\sigma(\alpha) \cup (\sigma(\beta) \cap \sigma(\gamma)) \cup s &= \omega/\sigma(\alpha) \cup (\sigma(\beta) \cap \overline{\sigma(\alpha)} \cap \sigma(\gamma)) \cup s \\
&= \omega/\sigma(\alpha) \cup (\sigma(\beta) \cap \overline{\sigma(\alpha)} \cap \sigma(\gamma)) \cup (\sigma(\beta) \cap \overline{\sigma(\alpha)} \cap \overline{\sigma(\gamma)}) \cup s \\
&= \omega/\sigma(\alpha) \cup (\sigma(\beta) \cap \overline{\sigma(\alpha)}) \cup s \\
&= \omega/\sigma(\alpha) \cup \sigma(\beta) \cup s
\end{aligned}$$

Thus by (A.39), $\omega/\sigma(\alpha) \cup \sigma(\beta) \cup s \in L(\alpha)$. Since $L(\alpha) \subseteq L(\alpha + \beta)$, it follows that

$$\omega/\sigma(\alpha) \cup \sigma(\beta) \cup s \in L(\alpha + \beta) \tag{A.41}$$

Since $\sigma(\gamma) \cup (\sigma(\alpha) \cap \sigma(\gamma)) = \sigma(\gamma)$, then by (A.39),

$$\omega/\sigma(\gamma) \cup s \in L(\gamma) \tag{A.42}$$

Therefore by (A.40),(A.41) and (A.42),

$$\omega \in L((\alpha + \beta) [\, s \,] \gamma)$$

In the similar way, the above equation also follows where $\omega \in L(\beta [\, s \cup (\sigma(\alpha) \cap \sigma(\gamma)) \,] \gamma)$. Therefore,

$$L(\alpha [\, s \cup (\sigma(\beta) \cap \sigma(\gamma)) \,] \gamma) \cup L(\beta [\, s \cup (\sigma(\alpha) \cap \sigma(\gamma)) \,] \gamma) \subseteq L((\alpha + \beta) [\, s \,] \gamma) \tag{A.43}$$

Then by (A.36) and (A.43),

$$L((\alpha [\, s \cup (\sigma(\beta) \cap \sigma(\gamma)) \,] \gamma)) \cup L((\beta [\, s \cup (\sigma(\alpha) \cap \sigma(\gamma)) \,] \gamma)) = L((\alpha + \beta) [\, s \,] \gamma)$$

From the definition of L, this clearly implies

$$L((\alpha [\, s \cup (\sigma(\beta) \cap \sigma(\gamma)) \,] \gamma) + (\beta [\, s \cup (\sigma(\alpha) \cap \sigma(\gamma)) \,] \gamma)) = L((\alpha + \beta) [\, s \,] \gamma)$$

□

Appendix B

Proofs for Section 3.4

This appendix provides the proofs for theorems in Chapter 3 and the lemmas which required for proving those theorems. Since our axiom system F_c is complete, it is possible to prove these theorems utilizing the derivation rules of F_c. However, in this section, we prove the theorems directly based on the semantics of BCREs in order to make our proofs precise.

B.1 Preliminary

Preceding to the theorems, we prove some lemmas which are required to prove the theorem in the next section. The following Lemma B.1 represents the basic properties of restriction. Note that in the reminder of this appendix we use these properties without being explicit referred to.

LEMMA **B.1**
Let s, t and u be a set of symbols, α be an BCRE and ω be a sequence.

1. $\omega/s/t = \omega/s \cap t$
2. $\omega/s/t = \omega/t/s$
3. $\omega/s/t = \omega/t$ *if* $s \supseteq t$
4. $\omega/s \cup u = \omega/t \cup u$ *if* $\omega/s = \omega/t$
5. $\omega/\sigma(\alpha) = \omega$ *if* $\omega \in L(\alpha)$
6. $\omega \in s^*$ *if* $\omega = \omega/s$

It is easy to prove these properties. We omit the proof here. □

The next Lemma B.2 and B.3 also represent simple properties between restriction and the language of BCREs.

LEMMA **B.2**

1. $\omega \cdot x/s \in L(\alpha x)$ *implies* $\omega/s \in L(\alpha)$.

2. $\omega/\sigma(\alpha) \cup s \in L(\alpha)$ *implies* $\omega/\sigma(\alpha) \in L(\alpha)$.

PROOF

It is obvious from the definition of L and restriction. □

.

LEMMA **B.3**

$$\omega/\sigma(\alpha) \cup \sigma(\gamma) = \omega/\sigma(\alpha)\ if\ \omega \in (\sigma(\alpha) \cup \sigma(\beta))^*\ and\ \sigma(\gamma) \cap \sigma(\beta) = \phi$$

PROOF

Assuming that $\omega \in (\sigma(\alpha) \cup \sigma(\beta))^*$ and $\sigma(\gamma) \cap \sigma(\beta) = \phi$. Then assume that $\omega/\sigma(\alpha) \cup \sigma(\gamma) \neq \omega/\sigma(\alpha)$. By this assumption, there is some x in ω such that $x \notin \sigma(\alpha)$ and $x \in \sigma(\gamma)$. By the assumption $\omega \in (\sigma(\alpha) \cup \sigma(\beta))^*$, $x \in (\sigma(\alpha) \cup \sigma(\beta))$. Hence by $x \notin \sigma(\alpha)$, it follows that $x \in \sigma(\beta)$. Therefore $x \in \sigma(\beta) \cap \sigma(\gamma)$. However, $\sigma(\gamma) \cap \sigma(\beta) = \phi$ clearly holds from the assumption. and clearly contradicts $x \in \sigma(\beta) \cap \sigma(\gamma)$. Hence it is concluded as proof by contradiction that $\omega/\sigma(\alpha) \cup \sigma(\gamma) = \omega/\sigma(\alpha)$. □

LEMMA **B.4**

Assume that $\omega_1 \cdot \omega_2 \in L(\alpha x \beta)$, $x \notin \sigma(\alpha)$, ω_1 *has only one* x *and the* x *is at the end of* ω_1. *Then* $\omega_1 \in L(\alpha x)$.

PROOF

Suppose that

$$\omega_1 \cdot \omega_2 = a_0 \cdots a_n \cdot x \cdot b_0 \cdots b_m$$

where $a_0 \cdots a_n \in L(A)$ and $b_0 \cdots b_m \in L(B)$. Then let us consider the following three cases.

1. if $\omega_1 = a_0 \cdots a_n \cdot x$ then $\omega \in L(\alpha x)$.

2. if $\omega_1 = a_0 \cdots a_k$ $(0 \leq k \leq n)$ then $a_k = x$. It contradicts $x \notin \sigma(A)$.

3. if $\omega_1 = a_0 \cdots a_n \cdot x \cdot b_0 \cdots b_k$ $(0 \leq k \leq m)$ then $b_k = x$. It contradicts the assumption where ω_1 has a single x.

Therefore, it is concluded that $\omega_1 = a_0 \cdots a_n \cdot x \in L(\alpha x)$ and the lemma holds. □

LEMMA **B.5**

Let α, β and γ be BCREs. Then $\omega \in (\sigma(\alpha) \cup \sigma(\gamma))^*$ if $\omega \in (\sigma(\alpha) \cup \sigma(\beta))^*$, $\omega/\sigma(\beta) \in L(\gamma)$ and $\sigma(\gamma) \subseteq \sigma(\beta)$.

PROOF

Assume that $\omega \in (\sigma(\alpha) \cup \sigma(\beta))^*$, $\omega/\sigma(\beta) \in L(\gamma)$ and $\sigma(\gamma) \subseteq \sigma(\beta)$. Since $\omega/\sigma(\beta) \in L(\gamma)$, $\omega/\sigma(\beta) = \omega/\sigma(\beta)/\sigma(\gamma)$ holds. Then by $\sigma(\gamma) \subseteq \sigma(\beta)$, $\omega/\sigma(\beta) = \omega/\sigma(\gamma)$ follows. Hence, ω does not have a symbol in $\overline{\sigma(\gamma)} \cap \sigma(\beta)$. Since $\omega \in (\sigma(\alpha) \cup \sigma(\beta))^*$, it follow that $\omega \in ((\sigma(\alpha) \cup \sigma(\beta)) \cap \overline{(\overline{\sigma(\gamma)} \cap \sigma(\beta))})^* = ((\sigma(\alpha) \cap \overline{\sigma(\beta)}) \cup \sigma(\gamma))^*$. It is clear that $((\sigma(\alpha) \cap \overline{\sigma(\beta)}) \cup \sigma(\gamma))^* \subseteq (\sigma(\alpha) \cup \sigma(\gamma))^*$. Thus $\omega \in (\sigma(\alpha) \cup \sigma(\gamma))^*$ and the lemma holds. □

B.2 Theorem 3.5 (Associative Law (1))

Assume that $\sigma(\alpha) \cup \sigma(\beta) \subseteq \sigma(\alpha\,[\ ]\,\beta)$ and $\sigma(\beta) \cup \sigma(\gamma) \subseteq \sigma(\beta\,[\ ]\,\gamma)$ then

$$\alpha\,[\ ]\,(\beta\,[\ ]\,\gamma) = (\alpha\,[\ ]\,\beta)\,[\ ]\,\gamma$$

Proof

Assume that $\sigma(\beta) \cup \sigma(\gamma) \subseteq \sigma(\beta\,[\ ]\,\gamma)$, then $\sigma(\beta\,[\ ]\,\gamma) = \sigma(\beta) \cup \sigma(\gamma)$ holds since it is clear that $\sigma(\beta) \cup \sigma(\gamma) \supseteq \sigma(\beta\,[\ ]\,\gamma)$. From this and the semantics of BCREs, it follows that

$$\begin{aligned} L(\alpha\,[\ ]\,(\beta\,[\ ]\,\gamma)) = \{\omega \mid\ & \omega \in (\sigma(\alpha) \cup \sigma(\beta) \cup \sigma(\gamma))^*, \\ & \omega/\sigma(\alpha) \in L(\alpha), \\ & \omega/\sigma(\beta) \cup \sigma(\gamma) \in L(\beta\,[\ ]\,\gamma)\} \end{aligned}$$

By the semantics again,

$$\begin{aligned} L(\beta\,[\ ]\,\gamma) = \{\omega \mid\ & \omega \in (\sigma(\beta) \cup \sigma(\gamma))^*, \\ & \omega/\sigma(\beta) \in L(\beta), \\ & \omega/\sigma(\gamma) \in L(\gamma)\} \end{aligned}$$

Thus it follows that

$$L(\alpha\,[\ ]\,(\beta\,[\ ]\,\gamma)) = \{\omega \mid\ \omega \in (\sigma(\alpha) \cup \sigma(\beta) \cup \sigma(\gamma))^*,$$

$$
\begin{aligned}
&\omega/\sigma(\alpha) \in L(\alpha),\\
&\omega/\sigma(\beta) \cup \sigma(\gamma) \in (\sigma(\beta) \cup \sigma(\gamma))^*\\
&\omega/\sigma(\beta) \cup \sigma(\gamma)/\sigma(\beta) \in L(\beta),\\
&\omega/\sigma(\beta) \cup \sigma(\gamma)/\sigma(\gamma) \in L(\gamma)\}
\end{aligned}
$$

It is clear that the condition $\omega/\sigma(\beta) \cup \sigma(\gamma) \in (\sigma(\beta) \cup \sigma(\gamma))^*$ at the third line in the above equation holds for every ω. Therefore it can be eliminated from the above conditions. Moreover, it follows for all α and β that $\omega/\sigma(\alpha) \cup \sigma(\beta)/\sigma(\alpha) = \omega/\sigma(\alpha)$. Hence,

$$
\begin{aligned}
L(\alpha\,[\ ]\,(\beta\,[\ ]\,\gamma)) = \{\omega \mid\ & \omega \in \sigma(\alpha) \cup \sigma(\beta) \cup \sigma(\gamma),\\
& \omega/\sigma(\alpha) \in L(\alpha),\\
& \omega/\sigma(\beta) \in L(\beta),\\
& \omega/\sigma(\gamma) \in L(\gamma)\}
\end{aligned}
\tag{B.1}
$$

In the similar way, it is also proved that $L((\alpha\,[\ ]\,\beta)\,[\ ]\,\gamma)$ equals (B.1) by assuming $L(\alpha) \cup L(\beta) \subseteq L(\alpha\,[\ ]\,\beta)$. Thus the theorem follows. □

B.3 Theorem 3.6 (Associative Law (2))

Assume that $(\sigma(\alpha) \cap \sigma(\beta)) \cup (\sigma(\beta) \cap \sigma(\gamma)) \cup (\sigma(\gamma) \cap \sigma(\alpha)) \subseteq s$ then,

$$\alpha\,[\,s\,]\,(\beta\,[\,s\,]\,\gamma) = (\alpha\,[\,s\,]\,\beta)\,[\,s\,]\,\gamma$$

Proof

Let us prove that $L(\alpha\,[\,s\,]\,(\beta\,[\,s\,]\,\gamma)) = P$ where

$$(\sigma(\alpha) \cap \sigma(\beta)) \cup (\sigma(\beta) \cap \sigma(\gamma)) \cup (\sigma(\gamma) \cap \sigma(\alpha)) \subseteq s$$

and

$$
\begin{aligned}
P = \{\ \omega \quad \mid\quad & \omega \in (\sigma(\alpha) \cup \sigma(\beta) \cup \sigma(\gamma))^*, && \text{(B.2)}\\
& \omega/\sigma(\alpha) \cup s \in L(\alpha), && \text{(B.3)}\\
& \omega/\sigma(\beta) \cup s \in L(\beta), && \text{(B.4)}\\
& \omega/\sigma(\gamma) \cup s \in L(\gamma)\ \ \} && \text{(B.5)}
\end{aligned}
$$

Assume that $(\sigma(\alpha)\cap\sigma(\beta))\cup(\sigma(\beta)\cap\sigma(\gamma))\cup(\sigma(\gamma)\cap\sigma(\alpha)) \subseteq s$ and $\omega \in L(\alpha\,[\,s\,]\,(\beta\,[\,s\,]\,\gamma))$. Then, from the definition of L, it is easy to show that

$$
\begin{aligned}
L(\alpha\,[\,s\,]\,(\beta\,[\,s\,]\,\gamma)) = \{\ \omega \quad \mid\ & \omega \in (\sigma(\alpha)\cup\sigma(\beta\,[\,s\,]\,\gamma))^*, && \text{(B.6)}\\
& \omega/\sigma(\alpha)\cup s \in L(\alpha), && \text{(B.7)}\\
& \omega/\sigma(\beta\,[\,s\,]\,\gamma)\cup s \in (\sigma(\beta)\cup\sigma(\gamma))^*,\\
& \omega/\sigma(\beta\,[\,s\,]\,\gamma)\cup s/\sigma(\beta)\cup s \in L(\beta), && \text{(B.8)}\\
& \omega/\sigma(\beta\,[\,s\,]\,\gamma)\cup s/\sigma(\gamma)\cup s \in L(\gamma)\quad \}
\end{aligned}
$$

Now, we consider the following two cases:

1. Assume that $\sigma(\beta\,[\,s\,]\,\gamma) \supseteq \sigma(\beta)$. It is obvious that $\sigma(\beta\,[\,s\,]\,\gamma)\cup s \supseteq \sigma(\beta)\cup s$. Then it follows that $\omega/\sigma(\beta\,[\,s\,]\,\gamma)\cup s/\sigma(\beta)\cup s = \omega/\sigma(\beta)\cup s$. Therefore by (B.8),

   $$\omega/\sigma(\beta)\cup s \in L(\beta)$$

2. Assume that $\sigma(\beta\,[\,s\,]\,\gamma) \subseteq \sigma(\beta)$.

   Since $\omega \in (\sigma(\alpha)\cup\sigma(\beta\,[\,s\,]\,\gamma))^*$, it follows that $\omega = \omega/\sigma(\alpha)\cup\sigma(\beta\,[\,s\,]\,\gamma)$ Therefore,

   $$
   \begin{aligned}
   & \omega/\sigma(\beta\,[\,s\,]\,\gamma)\cup s\\
   = \ & \omega/\sigma(\alpha)\cup\sigma(\beta\,[\,s\,]\,\gamma)/\sigma(\beta\,[\,s\,]\,\gamma)\cup s\\
   = \ & \omega/(\sigma(\alpha)\cup\sigma(\beta\,[\,s\,]\,\gamma))\cap(\sigma(\beta\,[\,s\,]\,\gamma)\cup s)\\
   = \ & \omega/\sigma(\beta\,[\,s\,]\,\gamma)\cup(\sigma(\alpha)\cap s)\\
   = \ & \omega/\sigma(\beta\,[\,s\,]\,\gamma)\cup(\sigma(\alpha)\cap((\sigma(\alpha)\cap\sigma(\beta))\cup(\sigma(\beta)\cap\sigma(\gamma))\cup(\sigma(\gamma)\cap\sigma(\alpha))\cup s'))\\
   = \ & \omega/\sigma(\beta\,[\,s\,]\,\gamma)\cup(\sigma(\alpha)\cap\sigma(\beta))\cup(\sigma(\alpha)\cap\sigma(\gamma))\cup(\sigma(\alpha)\cap s') && \text{(B.9)}
   \end{aligned}
   $$

   where s' is a set of symbols such that

   $$s = (\sigma(\alpha)\cap\sigma(\beta))\cup(\sigma(\beta)\cap\sigma(\gamma))\cup(\sigma(\gamma)\cap\sigma(\alpha))\cup s')$$

   holds. In the similar way, it also follows that

   $$
   \begin{aligned}
   & \omega/\sigma(\beta)\cup s\\
   = \ & \omega/\sigma(\alpha)\cup\sigma(\beta\,[\,s\,]\,\gamma)/\sigma(\beta)\cup s\\
   = \ & \omega/(\sigma(\alpha)\cap\sigma(\beta))\cup(\sigma(\beta\,[\,s\,]\,\gamma)\cap\sigma(\beta))\cup(\sigma(\alpha)\cup\sigma(\beta\,[\,s\,]\,\gamma))\cap s)
   \end{aligned}
   $$

   Since $\sigma(\beta\,[\,s\,]\,\gamma) \subseteq \sigma(\beta)$, $\sigma(\beta\,[\,s\,]\,\gamma)\cap\sigma(\beta) = \sigma(\beta\,[\,s\,]\,\gamma)$ and according to the assumption of s,

   $$(\sigma(\alpha)\cup\sigma(\beta\,[\,s\,]\,\gamma))\cap s$$

$$
\begin{aligned}
&= (\sigma(\alpha) \cup \sigma(\beta\,[\,s\,]\,\gamma)) \cap ((\sigma(\alpha) \cap \sigma(\beta)) \cup (\sigma(\beta) \cap \sigma(\gamma)) \cup (\sigma(\gamma) \cap \sigma(\alpha)) \cup s') \\
&= (\sigma(\alpha) \cap \sigma(\alpha) \cap \sigma(\beta)) \cup (\sigma(\alpha) \cap \sigma(\beta) \cap \sigma(\gamma)) \cup \\
&\quad (\sigma(\alpha) \cap \sigma(\gamma) \cap \sigma(\alpha)) \cup (\sigma(\alpha) \cap s') \cup \\
&\quad (\sigma(\beta\,[\,s\,]\,\gamma) \cap \sigma(\alpha) \cap \sigma(\beta)) \cup (\sigma(\beta\,[\,s\,]\,\gamma) \cap \sigma(\beta) \cap \sigma(\gamma)) \cup \\
&\quad (\sigma(\beta\,[\,s\,]\,\gamma) \cap \sigma(\gamma) \cap \sigma(\alpha)) \cup (\sigma(\beta\,[\,s\,]\,\gamma) \cap s') \\
&= (\sigma(\alpha) \cap \sigma(\beta)) \cup (\sigma(\alpha) \cap \sigma(\beta) \cap \sigma(\gamma)) \cup \\
&\quad (\sigma(\alpha) \cap \sigma(\gamma)) \cup (\sigma(\alpha) \cap s') \cup \\
&\quad (\sigma(\beta\,[\,s\,]\,\gamma) \cap \sigma(\alpha)) \cup (\sigma(\beta\,[\,s\,]\,\gamma) \cap \sigma(\gamma)) \cup \\
&\quad (\sigma(\beta\,[\,s\,]\,\gamma) \cap \sigma(\gamma) \cap \sigma(\alpha)) \cup (\sigma(\beta\,[\,s\,]\,\gamma) \cap s') \\
&= (\sigma(\alpha) \cap \sigma(\beta)) \cup (\sigma(\alpha) \cap \sigma(\gamma)) \cup (\sigma(\alpha) \cap s') \cup \\
&\quad (\sigma(\beta\,[\,s\,]\,\gamma) \cap \sigma(\gamma)) \cup (\sigma(\beta\,[\,s\,]\,\gamma) \cap s')
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&\omega/(\sigma(\alpha) \cap \sigma(\beta)) \cup s \\
&= \omega/(\sigma(\alpha) \cap \sigma(\beta)) \cup \sigma(\beta\,[\,s\,]\,\gamma) \cup ((\sigma(\alpha) \cup \sigma(\beta\,[\,s\,]\,\gamma)) \cap s) \\
&= \omega/\sigma(\beta\,[\,s\,]\,\gamma) \cup (\sigma(\alpha) \cap \sigma(\beta)) \cup (\sigma(\alpha) \cap \sigma(\gamma)) \cup (\sigma(\alpha) \cap s')
\end{aligned}
$$

By (B.9),

$$\omega/\sigma(\beta) \cup s = \omega/\sigma(\beta\,[\,s\,]\,\gamma) \cup s$$

Hence by (B.8), $\omega/\sigma(\beta) \cup s \in L(\beta)$.

Accordingly, it follows from the cases 1 and 2 that

$$\omega/\sigma(\beta) \cup s \in L(\beta) \tag{B.10}$$

In the similar way, it can be proved that

$$\omega/\sigma(\gamma) \cup s \in L(\gamma) \tag{B.11}$$

By (B.6), it clearly follows that

$$\omega \in (\sigma(\alpha) \cup \sigma(\beta) \cup \sigma(\gamma))^* \tag{B.12}$$

Then by (B.7), (B.10), (B.11) and (B.12), $\omega \in P$ holds. Therefore it is concluded that

$$L(\alpha\,[\,s\,]\,(\beta\,[\,s\,]\,\gamma)) \subseteq P \tag{B.13}$$

Now, on the other hand, we start with assuming $\omega \in P$, namely, (B.2) to (B.5) follow. Then since $\sigma(\beta) \cup \sigma(\gamma) \supseteq \sigma(\beta)$, it follows that $\omega/\sigma(\beta) \cup s = \omega/\sigma(\beta) \cup \sigma(\gamma) \cup s/\sigma(\beta) \cup s$. Hence by (B.4),

$$\omega/\sigma(\beta) \cup \sigma(\gamma) \cup s/\sigma(\beta) \cup s \in L(\beta) \tag{B.14}$$

In the similar way, it follows from (B.5) that

$$\omega/\sigma(\beta) \cup \sigma(\gamma) \cup s/\sigma(\gamma) \cup s \in L(\gamma) \tag{B.15}$$

By (B.4) and (B.5), it can be proved that

$$\omega/\sigma(\beta) \cup \sigma(\gamma) \cup s \in (\sigma(\beta) \cup \sigma(\gamma))^* \tag{B.16}$$

Then by (B.14),(B.15), (B.16) and the definition of L, we can obtain that

$$\omega/\sigma(\beta) \cup \sigma(\gamma) \cup s \in L(\beta\,[\,s\,]\,\gamma) \tag{B.17}$$

By (B.16), it is obvious that $\omega/\sigma(\beta) \cup \sigma(\gamma) \cup s = \omega/\sigma(\beta) \cup \sigma(\gamma)$. Therefore, By (B.17),

$$\omega/\sigma(\beta) \cup \sigma(\gamma) \in L(\beta\,[\,s\,]\,\gamma) \tag{B.18}$$

By (B.2),(B.18) and Lemma B.5

$$\omega \in (\sigma(\alpha) \cup \sigma(\beta\,[\,s\,]\,\gamma))^* \tag{B.19}$$

By (B.17) again, $\omega/\sigma(\beta) \cup \sigma(\gamma) \cup s = \omega/\sigma(\beta) \cup \sigma(\gamma) \cup s/\sigma(\beta\,[\,s\,]\,\gamma) = \omega/\sigma(\beta\,[\,s\,]\,\gamma)$ holds. Thus,

$$\omega/\sigma(\beta\,[\,s\,]\,\gamma) \in L(\beta\,[\,s\,]\,\gamma) \tag{B.20}$$

Assume that $s \not\subseteq \sigma(\beta\,[\,s\,]\,\gamma)$. Then by (B.17) and $\sigma(\beta\,[\,s\,]\,\gamma) \subseteq \sigma(\beta) \cup \sigma(\gamma)$, ω clearly has no symbol in $s \cap \overline{\sigma(\beta\,[\,s\,]\,\gamma)}$. Hence, $\omega/\sigma(\beta\,[\,s\,]\,\gamma) = \omega/\sigma(\beta\,[\,s\,]\,\gamma) \cup s$ holds. On the other hand, if we suppose that $s \subseteq \sigma(\beta\,[\,s\,]\,\gamma)$, it is obvious that $\omega/\sigma(\beta\,[\,s\,]\,\gamma) = \omega/\sigma(\beta\,[\,s\,]\,\gamma) \cup s$. Then by (B.20),

$$\omega/\sigma(\beta\,[\,s\,]\,\gamma) \cup s \in L(\beta\,[\,s\,]\,\gamma)$$

Accordingly, by the definition of L,

$$\omega/\sigma(\beta\,[\,s\,]\,\gamma) \cup s \in (\sigma(\beta) \cup \sigma(\gamma))^* \tag{B.21}$$

$$\omega/\sigma(\beta\,[\,s\,]\,\gamma) \cup s/\sigma(\beta) \cup s \in L(\beta) \tag{B.22}$$

$$\omega/\sigma(\beta\,[\,s\,]\,\gamma) \cup s/\sigma(\gamma) \cup s \in L(\gamma) \tag{B.23}$$

By (B.3), (B.19), (B.21), (B.22), (B.23) and the definition of L, it holds that $\omega \in L(\alpha\,[\,s\,]\,(\beta\,[\,s\,]\,\gamma))$. Hence,

$$P \subseteq L(\alpha\,[\,s\,]\,(\beta\,[\,s\,]\,\gamma)) \tag{B.24}$$

By (B.11) and (B.24), it is proved that $P = L(\alpha\,[\,s\,]\,(\beta\,[\,s\,]\,\gamma))$.

In the similar way, it also can be proved that $P = L((\alpha\,[\,s\,]\,\beta)\,[\,s\,]\,\gamma))$. We omit a detailed proof since it is just a symmetry of the proof for $P = L(\alpha\,[\,s\,]\,(\beta\,[\,s\,]\,\gamma))$. It is thus concluded that $L((\alpha\,[\,s\,]\,\beta)\,[\,s\,]\,\gamma)) = L(\alpha\,[\,s\,]\,(\beta\,[\,s\,]\,\gamma))$ and the theorem holds. □

B.4 Theorem 3.7

$$\vdash \alpha\,[\,s\,]\,\beta = \alpha \parallel \beta \text{ if } (\sigma(\alpha) \cap \sigma(\beta)) \cup s = \phi$$

Proof

Assuming that $(\sigma(\alpha) \cap \sigma(\beta)) \cup s = \phi$, then in such case the axiom C_1 to C_5 are regarded as

$$\begin{array}{lll}
C_1 & \alpha\,[\,s\,]\,\perp & = \perp \\
C_2 & \alpha\,[\,s\,]\,\perp^* & = \alpha \\
C_3 & \alpha\,[\,s\,]\,\beta & = \beta\,[\,s\,]\,\alpha \\
C_4 & x\alpha\,[\,s\,]\,y\beta & = x(\alpha\,[\,s\,]\,y\beta) + y(x\alpha\,[\,s\,]\,\beta) \\
C_5 & (\alpha+\beta)\,[\,s\,]\,\gamma & = \alpha\,[\,s\,]\,\gamma + \beta\,[\,s\,]\,\gamma
\end{array}$$

Then by comparing this to the axiom C_6 to C_{10},

$$\begin{array}{lll}
C_6 & \alpha\|\perp & = \perp \\
C_7 & \alpha\|\perp^* & = \alpha \\
C_8 & \alpha\|\beta & = \beta\|\alpha \\
C_9 & x\alpha\|y\beta & = x(\alpha\|y\beta) + y(x\alpha\|\beta) \\
C_{10} & (\alpha+\beta)\|\gamma & = \alpha\|\gamma + \beta\|\gamma
\end{array}$$

it is obvious that they can simulates each other by replacing $\|$ with $[\,s\,]$ or vice versa. Since the F_c is complete, we can say that all possible equivalent relationship on $[\,s\,]$ and $\|$ operation

can be represented only with these axioms. This implies that we can swap $\alpha \,[\, s \,]\, \beta$ to $\alpha \,\|\, \beta$ or vice versa in any expressions if $\sigma(\alpha) \cap \sigma(\beta) \cap s = \phi$. Hence the lemma follows. □

B.5 Theorem 3.8 (Extraction)

$$\vdash (\alpha x \beta) \,[\, s \,]\, (\gamma y \delta) = (\alpha \,[\ ]\, \gamma)(x\beta \,[\, s \,]\, y\delta)$$

where $x, y \in (\sigma(x\beta) \cap \sigma(y\delta)) \cup s$ and $(\sigma(\alpha x\beta) \cup s) \cap \sigma(\gamma) = (\sigma(\gamma y\delta) \cup s) \cap \sigma(\alpha) = \phi$

This theorem means that if x and y are communication symbols and, α and γ does not possess communication symbols, then α and γ can be extracted as concurrent threads, namely $(\alpha \,[\ ]\, \gamma)$.

Proof

Assume that $x, y \in (\sigma(x\beta) \cap \sigma(y\delta)) \cup s$ and $(\sigma(\alpha x\beta) \cup s) \cap \sigma(\gamma) = (\sigma(\gamma y\delta) \cup s) \cap \sigma(\alpha) = \phi$. Then let us consider the following two cases.

1. Assume that $\omega \in L((\alpha \,[\ ]\, \gamma)(x\beta \,[\, s \,]\, y\delta))$. Then it is clear that

$$\omega \in (\sigma(\alpha x\beta) \cup \sigma(\gamma y\delta))^* \tag{B.25}$$

   According to the definition of L, there are some ω_1 and ω_2 such that $\omega_1 \in L(\alpha \,[\ ]\, \gamma))$, $\omega_2 \in L(x\beta \,[\, s \,]\, y\delta)$ and $\omega \equiv \omega_1 \cdot \omega_2$ hold. By the definition of L, it follows from $\omega_1 \in L(\alpha \,[\ ]\, \gamma)$ that $\omega_1/\sigma(\alpha) \in L(\alpha)$. By the assumption, $(\sigma(\alpha x\beta) \cup s) \cap \sigma(\gamma) = \phi$. Thus by Lemma B.3, $\omega_1/\sigma(\alpha x\beta) \cup s = \omega_1/\sigma(\alpha)$. Hence,

$$\omega_1/\sigma(\alpha x\beta) \cup s \in L(\alpha) \tag{B.26}$$

   In the similar way, it is proved that

$$\omega_1/\sigma(\gamma y\delta) \cup s \in L(\gamma) \tag{B.27}$$

   By the definition of L and $\omega_2 \in L(x\beta \,[\, s \,]\, y\delta)$, $\omega_2 \in (\sigma(x\beta) \cup \sigma(x\delta))^*$ and $\omega_2/\sigma(x\beta) \cup s \in L(x\beta)$ follows. Hence by Lemma B.3, $\omega_2/\sigma(\alpha) \cup \sigma(x\beta) \cup s = \omega_2/\sigma(x\beta) \cup s$ holds. Therefore,

$$\omega_2/\sigma(\alpha x\beta) \cup s \in L(x\beta) \tag{B.28}$$

holds. In the similar way, it also follows that

$$\omega_2/\sigma(\gamma y\delta) \cup s \in L(y\delta) \tag{B.29}$$

By (B.26),(B.27),(B.28) and (B.29),

$$\omega/\sigma(\alpha x\beta) \cup s = (\omega_1/\sigma(\alpha x\beta) \cup s) \cdot (\omega_2/\sigma(\alpha x\beta) \cup s) \in L(\alpha x\beta) \tag{B.30}$$

$$\omega/\sigma(\gamma x\delta) \cup s = (\omega_1/\sigma(\gamma x\delta) \cup s) \cdot (\omega_2/\sigma(\gamma x\delta) \cup s) \in L(\gamma x\delta) \tag{B.31}$$

By (B.25),(B.30) and (B.31), $\omega \in L(\alpha x\beta\,[\,s\,]\,\gamma y\delta)$. Hence it is concluded that

$$L((\alpha\,[\ ]\,\gamma)(x\beta\,[\,s\,]\,y\delta)) \subseteq L(\alpha x\beta\,[\,s\,]\,\gamma y\delta) \tag{B.32}$$

2. Assume that $\omega \in L(\alpha x\beta\,[\,s\,]\,\gamma y\delta)$. If $x \neq y$ is assumed it is obvious from the definition of L,

$$L(\alpha x\beta\,[\,s\,]\,\gamma y\delta) = L((\alpha\,[\ ]\,\gamma)(x\beta\,[\,s\,]\,y\delta)) = \phi$$

Hence in the remainder of this proof we only focus on the case where $x = y$ holds. Then the definition of L, it is obvious that

$$\omega \in (\sigma(\alpha x\beta) \cup \sigma(\gamma x\delta))^* \tag{B.33}$$

$$\omega/\sigma(\alpha x\beta) \cup s \in L(\alpha x\beta) \tag{B.34}$$

$$\omega/\sigma(\gamma x\delta) \cup s \in L(\gamma x\delta) \tag{B.35}$$

By (B.34) and the definition of L, there are some ω_1 and ω_2 such that

$$\omega = \omega_1 \cdot \omega_2 \tag{B.36}$$

$$\omega_1/\sigma(\alpha x\beta) \cup s \in L(\alpha x) \tag{B.37}$$

$$\omega_2/\sigma(\alpha x\beta) \cup s \in L(\beta) \tag{B.38}$$

$$Last(\omega_1) = x \tag{B.39}$$

holds where $Last(\omega)$ means the last symbol of the sequence ω. Then by (B.35) and (B.36),

$$\omega_1 \cdot \omega_2/\sigma(\gamma x\delta) \cup s \in L(\gamma x\delta) \tag{B.40}$$

By the assumption $(\sigma(\alpha x\beta) \cup s) \cap \gamma = \phi$, it follows that $x \notin L(\gamma)$. Hence by Lemma B.4, (B.39) and (B.40),

$$\omega_1/\sigma(\gamma x\delta) \in L(\gamma x) \tag{B.41}$$

Accordingly, there is some ω_1' such that $\omega_1 = \omega_1' \cdot x$ and $\omega_1' \cdot x/\sigma(\gamma x\delta) \in L(\gamma x)$. Thus by Lemma B.2, it is easy to prove that $\omega_1'/\sigma(\gamma x\delta) \in L(\gamma)$. Hence, by Lemma B.2,

$$\omega_1'/\sigma(\gamma) \in L(\gamma) \tag{B.42}$$

In the similar way, it follows that $\omega_1'/\sigma(\alpha x\beta) \in L(\alpha)$ since $\omega_1' \cdot x/\sigma(\alpha x\beta) \in L(\alpha x)$ by (B.37). Thus by Lemma B.2,

$$\omega_1'/\sigma(\alpha) \in L(\alpha) \tag{B.43}$$

By (B.42) and (B.43)

$$\omega_1 \in L(\alpha\,[\ ]\,\gamma) \tag{B.44}$$

Now, it is clear that $\omega = \omega_1' \cdot x \cdot \omega_2'$ and there is no x in ω_1. Then by (B.33), (B.34) and (B.35), it can be proved that

$$\begin{aligned} x \cdot \omega_2' &\in (\sigma(x\beta) \cup \sigma(x\delta))^* \\ x \cdot \omega_2'/\sigma(x\beta) \cup s &\in L(x\beta) \\ x \cdot \omega_2'/\sigma(x\delta) \cup s &\in L(x\delta) \end{aligned}$$

Therefore, by the definition of L,

$$x \cdot \omega_2' \in L(x\beta\,[\,s\,]\,x\delta) \tag{B.45}$$

Since $\omega = \omega_1' \cdot x \cdot \omega_2$, By (B.44), (B.45) and the assumption of $x = y$,

$$\omega \in L((\alpha\,[\ ]\,\gamma)(x\beta\,[\,s\,]\,y\delta))$$

Hence,

$$L(\alpha x\beta\,[\,s\,]\,\gamma y\delta) \subseteq L((\alpha\,[\ ]\,\gamma)(x\beta\,[\,s\,]\,y\delta)) \tag{B.46}$$

By (B.32) and (B.46),

$$L(\alpha x\beta\,[\,s\,]\,\gamma y\delta) \subseteq L((\alpha\,[\ ]\,\gamma)(x\beta\,[\,s\,]\,y\delta))$$

Hence, it is concluded that

$$L(\alpha x\beta\,[\,s\,]\,\gamma y\delta) = L((\alpha\,[\ ]\,\gamma)(x\beta\,[\,s\,]\,y\delta))$$

and the theorem holds. □

Appendix C

Proof for Lemma 5.2

C.1 Preliminary

In this section, we assume that z is a set of symbols and $h(\alpha)$ and $\eta(\alpha, j)$ is from R.H.S. of the member of $X_z(\alpha)$ such that

$$\alpha = \sum_{j=1}^{h(\alpha)} \theta(\alpha, j) z(\alpha, j) \eta(\alpha, j) + \theta(\alpha, 0) \in X_z(\alpha)$$

We also use a set $\acute{X}_z$ to obtain L.H.S. of a member of X_z.

$$\acute{X}_z(\alpha) = \{\alpha \mid \exists \beta . \alpha = \beta \in X_z(\alpha)\}$$

LEMMA **C.1**
$\eta(\alpha, j) \in \acute{X}_z(\alpha)$ *for* $1 \leq j \leq h(\alpha)$.

PROOF

It is obvious from the definition of X_z that $X_z(\eta(\alpha, j)) \subseteq X_z(\alpha)$ for all j. Thus $X_z(\eta(\alpha, j)) \subseteq X_z(\alpha)$. Then it is also clear by the definition of X_z that $\alpha \in \acute{X}_z(\alpha)$. Therefore, $\eta(\alpha, j) \in \acute{X}_z(\eta(\alpha, j)) \subseteq \acute{X}_z(\alpha)$. □

LEMMA **C.2**

$$X_z(\alpha\beta) \subseteq \{\alpha'\beta = \rho(\alpha'\beta) \mid \alpha' \in \acute{X}_z(\alpha)\ \} \cup \bigcup_{j=1}^{h(\beta)} X_z(\eta(\beta, j))$$

where $\rho(\alpha'\beta)$ *is defined in the definition of* X_z.

PROOF

Let $\{\alpha_1, \cdots, \alpha_n\} = \acute{X}_z(\alpha)$ for some n and $\alpha_1 \equiv \alpha$. According to the definition of X_z, for all $1 \leq u \leq n$,

$$X_z(\alpha_u\beta) = \{\alpha_u\beta = \rho(\alpha_u\beta)\} \cup \bigcup_{j=1}^{h(\alpha_u)} X_z(\eta(\alpha_u, j)\beta) \cup \bigcup_{j=1}^{h(\beta)} X_z(\eta(\beta, j))$$

Since $\eta(\alpha_u, j) \in \{\alpha_1, \cdots \alpha_n\} = \acute{X}_z(\alpha)$ by Lemma C.1, we can obtain the following equations.

$$X_z(\alpha_u\beta) = \{\alpha_u\beta = \rho(\alpha_u\beta)\} \cup \bigcup_{j=1}^{h(\alpha_u)} X_z(\alpha_{uj}\beta) \cup \bigcup_{j=1}^{h(\beta)} X_z(\eta(\beta, j))$$

where $\alpha_{uj} \in \{\alpha_1, \cdots \alpha_n\}$. According to the definition, $X_z(\alpha_u)$ is a minimal set. Thus it is obvious that

$$X_z(\alpha_u\beta) \subseteq \{\alpha_j\beta = \rho(\alpha_j\beta) \mid 1 \leq j \leq n\} \cup \bigcup_{j=1}^{h(\beta)} X_z(\eta(\beta, j))$$

and the lemma holds. □

LEMMA **C.3**

$$X_z(\beta\alpha^*) \subseteq \{\ \beta'\alpha^* = \rho(\beta'\alpha^*) \mid \beta' \in \acute{X}_z(\alpha) \cup \acute{X}_z(\beta)\ \}$$

where $\rho(\beta'\alpha^*)$ is according to the definition of X_z.

PROOF

Suppose that $\acute{X}_z(\alpha) = \{\alpha_1, \cdots, \alpha_n\}$ and $\acute{X}_z(\beta) = \{\beta_1, \cdots, \beta_m\}$ where $\alpha \equiv \alpha_1$ and $\beta \equiv \beta_1$. Then by the definition of X_z, it follows for all $1 \leq u \leq n$ and $1 \leq v \leq m$ that

$$\begin{aligned} X_z(\alpha_u\alpha^*) &= \{\alpha_u\alpha^* = \rho(\alpha_u\alpha)\} \cup \bigcup_{j=1}^{h(\alpha)} X_z(\eta(\alpha, j)\alpha^*) \cup \bigcup_{j=1}^{h(\alpha_u)} X_z(\eta(\alpha_u, j)\alpha^*) \\ X_z(\beta_v\alpha^*) &= \{\beta_v\alpha^* = \rho(\beta_v\alpha)\} \cup \bigcup_{j=1}^{h(\alpha)} X_z(\eta(\alpha, j)\alpha^*) \cup \bigcup_{j=1}^{h(\beta_v)} X_z(\eta(\beta_v, j)\alpha^*) \end{aligned}$$

Since $\eta(\alpha_u, j) \in \{\alpha_1, \cdots, \alpha_n\}$ and $\eta(\beta_v, j) \in \{\beta_1, \cdots, \beta_n\}$ by Lemma C.1,

$$\begin{aligned} X_z(\alpha_u\alpha^*) &= \{\alpha_u\alpha^* = \rho(\alpha_u\alpha)\} \cup \bigcup_{j=1}^{h(\alpha_1)} X_z(\alpha_{1j}\alpha^*) \cup \bigcup_{j=1}^{h(\alpha_u)} X_z(\alpha_{uj}\alpha^*) \\ X_z(\beta_v\alpha^*) &= \{\beta_v\alpha^* = \rho(\beta_v\alpha)\} \cup \bigcup_{j=1}^{h(\alpha_1)} X_z(\alpha_{1j}\alpha^*) \cup \bigcup_{j=1}^{h(\beta_v)} X_z(\beta_{vj}\alpha^*) \end{aligned}$$

where α_{uj} is among $\{\alpha_1, \cdots, \alpha_n\}$ and β_{vj} is among $\{\beta_1, \cdots, \beta_n\}$. By the definition of X_z, X_z is a minimal set, therefore it follows that

$$\begin{aligned} X_z(\beta_v\alpha^*) \quad \subseteq \quad & \{\alpha_j\alpha^* = \rho(\alpha_j\alpha^*) \mid 1 \leq j \leq n\} \cup \\ & \{\beta_j\alpha^* = \rho(\beta_j\alpha^*) \mid 1 \leq j \leq m\} \end{aligned}$$

for all $1 \leq v \leq m$. Remind that $X_z(\beta\alpha^*) = X_z(\beta_1\alpha^*)$. Thus the lemma follows. □

LEMMA **C.4**

$$X_z(\alpha\,[\,s\,]\,\beta) \subseteq \{\ \alpha'\,[\,s\,]\,\beta' = \rho(\alpha'\,[\,s\,]\,\beta') \mid \alpha' \in \acute{X}_z(\alpha), \beta' \in \acute{X}_z(\beta)\ \}$$

PROOF

Suppose that $\acute{X}_z(\alpha) = \{\alpha_1, \cdots, \alpha_n\}$ and $\acute{X}_z(\beta) = \{\beta_1, \cdots, \beta_m\}$ where $\alpha \equiv \alpha_1$ and $\beta \equiv \beta_1$. Then by the definition of X_z, it follows for all $1 \leq u \leq n$ and $1 \leq v \leq m$ that

$$X_z(\alpha_u\,[\,s\,]\,\beta_v) = \{\alpha_u\,[\,s\,]\,\beta_v = \rho(\alpha_u\,[\,s\,]\,\beta_v)\ \} \cup \bigcup_{i=1}^{h(\alpha_u)} \bigcup_{j=1}^{h(\beta_v)} (\eta(\alpha_u, j))\,[\,s\,]\,\eta(\beta_v, j))$$

Since $\eta(\alpha_u, j) \in \{\alpha_1, \cdots, \alpha_n\}$ and $\eta(\beta_v, j) \in \{\beta_1, \cdots, \beta_n\}$ by Lemma C.1,

$$X_z(\alpha_u\,[\,s\,]\,\beta_v) = \{\alpha_u\,[\,s\,]\,\beta_v = \rho(\alpha_u\,[\,s\,]\,\beta_v)\ \} \cup \bigcup_{i=1}^{h(\alpha_u)} \bigcup_{j=1}^{h(\beta_v)} (\alpha_{uj})\,[\,s\,]\,\beta_{vj})$$

where α_{uj} is among $\{\alpha_1, \cdots, \alpha_n\}$ and β_{vj} is among $\{\beta_1, \cdots, \beta_n\}$. By the definition of X_z, X_z is a minimal set, therefore it follows that

$$X_z(\alpha_u\,[\,s\,]\,\beta_v) \subseteq \{\alpha_j\,[\,s\,]\,\beta_k = \rho(\alpha_j\,[\,s\,]\,\beta_k) \mid 1 \leq j \leq n, 1 \leq k \leq m\}$$

□

C.2 Lemma 5.2

Let γ and δ are expressions without $[\,s\,]$ and $||$ operators then, for any z,

$$|\delta|_l \leq |\gamma|_t \text{ if } \delta \in \acute{X}_z(\gamma) \text{ follows.}$$

PROOF

In this proof, we assume $h(\alpha)$, $\theta(\alpha, j)$ and $\eta(\alpha, j)$ are expressions such that

$$\alpha = \sum_{j=1}^{h(\alpha)} \theta(\alpha, j) z(\alpha, j) \eta(\alpha, j) + \theta(\alpha, 0) \in X_z(\alpha)$$

Then we prove this lemma by induction on the structure of BCREs. As the base step, it is obvious by the definition of X_z that the lemma holds for $\gamma \equiv \bot$, $\gamma \equiv \bot^*$ and $\gamma \equiv x$ (x is a symbol). Next, as induction steps, assume that the lemma holds for $\gamma \equiv \alpha$ and $\gamma \equiv \beta$ and then let us consider the following cases.

1. Suppose that $\delta \in \acute{X}_z(\alpha + \beta)$. Then, from the definition of X_z, it follows that $\delta \equiv \alpha + \beta$, $\delta \in \acute{X}_z(\eta(\alpha, j))$ or $\delta \in \acute{X}_z(\eta(\beta, k))$. It is clear that $|\alpha + \beta|_l \leq |\alpha + \beta|_t$. By induction hypothesis and $X_z(\eta(\alpha, j)) \subseteq X_z(\alpha)$ for all $1 \leq j \leq h(\alpha)$, $|\delta|_l \leq |\alpha|_t$ if $\delta \in \acute{X}_z(\eta(\alpha, j))$. In the similar way, it can be proved that $|\delta|_l \leq |\beta|_t$ if $\delta \in \acute{X}_z(\eta(\beta, j))$. Thus, it is concluded that $\delta \in \acute{X}_z(\alpha + \beta)$ implies that $|\delta|_l \leq |\alpha + \beta|_t$ and the lemma holds for $\gamma \equiv \alpha + \beta$.

2. Assume that $\delta \in \acute{X}_z(\alpha\beta)$. Then by Lemma C.2, $\delta \equiv \alpha'\beta$ or $\delta \in \acute{X}_z(\eta(\beta, j))$ where $\alpha' \in \acute{X}_z(\alpha)$ and $1 \leq j \leq h(\beta)$. By induction hypothesis, $|\alpha'|_l \leq |\alpha|_t$ clearly holds. Hence, if $\delta \equiv \alpha'\beta$ then, $|\delta|_l = |\alpha'\beta|_l = |\alpha'|_l + |\beta|_l \leq |\alpha|_t + |\beta|_l \leq |\alpha|_t + |\beta|_t = |\alpha\beta|_t$ follows (Note that it immediately follows by the definition that $|\alpha|_l \leq |\alpha|_t$ for any α). On the other hand, if $\delta \in \acute{X}_z(\eta(\beta, j))$ then, it is clear that $\delta \in \acute{X}_z(\beta)$. Thus by induction hypothesis, $|\delta|_l \leq |\beta|_t \leq |\alpha\beta|_t$ holds. Therefore it is conluded that $|\delta|_l \leq |\alpha\beta|_t$ in the all cases.

3. Suppose that $\delta \in \acute{X}_z(\beta\alpha^*)$. Then by Lemma C.3, $\delta \equiv \beta'\alpha^*$ where $\beta' \in \acute{X}_z(\alpha) \cup \acute{X}_z(\beta)$ If $\beta' \in \acute{X}_z(\alpha)$ then by induction hypothesis, $|\beta'|_l \leq |\alpha|_t$. Therefore $|\delta|_l = |\beta'\alpha^*| = |\beta'|_l + |\alpha|_l \leq |\alpha_l| + |\alpha|_l \leq |\alpha^*|_t$.

4. Suppose that $\delta \in \acute{X}_z(\alpha^*)$. Then by the definition of X_z, $\delta \equiv \alpha^*$ or $\delta \in \acute{X}_z(\eta((,\alpha), j))$ $(1 \leq j \leq h(\alpha))$. If $\delta \equiv \alpha^*$, it is obvious that $|\delta|_l \leq |\alpha^*|_t$. On the other hand, if $\delta \in \acute{X}_z(\eta((,\alpha), j)\alpha^*)$, we can prove that $|\delta|_l \leq |\alpha^*|_t$ in the similar way to 3.

By 1 to 4, the lemma clearly holds. □

Appendix D

A PCM device driver implementation

```
/* Clock Frequency (Hz) */
#define INPUT_CLOCK 50000
#define CH_MAX 3
static int f[CH_MAX], v[CH_MAX];
static short *start_adr[CH_MAX];
static short *current_adr[CH_MAX];
static short *end_adr[CH_MAX];
static double counter[CH_MAX];
static double counter_step[CH_MAX];
static int playflag[CH_MAX];
static Semaphore sem=0;
/* play.(p_0+p_1+p_2) */
void PLAY(int ch, short *start, short *end){
  ENTER(sem);
  playflag[ch] = 1;
  start_adr[ch] = start;
  end_adr[ch] = end;
  LEAVE(sem);
}
/* stop.(s_0+s_1+s_2) */
void STOP(int ch){
  ENTER(sem);
  playflag[ch] = 0;
  LEAVE(sem);
```

```
}
/* freq.(f_0+f_1+f_2) */
void FREQ(int ch, int value){
  ENTER(sem);
  f[ch] = value;
  counter[ch] = 0.0;
  counter_step[ch] = f[ch]/INPUT_CLOCK;
  LEAVE(sem);
}
/* vol.(v_0+v_1+v_2) */
void VOL(int ch, int value){
  ENTER(sem);
  v[ch] = value;
  LEAVE(sem);
}
/* clock.calc.write */
void CLK(){
  short mix = 0;
  int i;
  for(i=0;i<3;i++) {
    if(playflag[i]) {
      /* calcurate the address counter */
      counter[i] += counter_step[i];
      if(counter[i]>=1.0) {
        counter[i]-=1.0;
        current_adr[i]++;
      }
      /* check the end of data */
      if(current_adr[i]==end_adr[i])
        playflag[i] = 0;
      mix += (*(current_adr[i]) * v[i]) >> 4;
    }
  }
  /* I/O Access */
  DAC_WRITE(mix);
```

```
}
```

Bibliography

[1] J.Rumbaugh, M.Blaha, W.Premerlani, F.Eddy, W.Lorenson. *Object-Oriented Modeling and Design*, Prentice Hall, 1991.

[2] Vijay K. Garg, M.T. Ragunath. *Concurrent regular expressions and their relationship to Petri nets*, Theoretical Computer Science 96, pp.285-304, 1992.

[3] Toshiaki Aoki and Takuya Katayama. *SES Model for Object-Oriented Time Critical System Development*, Proceedings of the IEEE International Conference on Artificial Intelligence and Computational Intelligence for Decision, Control, and Automation in Engineering and Industrial Applications ACIDCA'2000, pp.19-24, 2000.

[4] Maher Awad, Juha Kuusela and Jurgen Ziegler. *Object-Oriented Technology for Real-Time Systems*, Prentice Hall, 1996.

[5] Bran Selic, Garth Gullekson and Paul T.Ward, *Real-Time Object-Oriented Modeling*, John Wiley and Sons, 1994.

[6] Arto Salomaa, *Two Complete Axiom Systems for the Algebra of Regular Events*, Journal of the Association for Computing Machinery, Vol.13, No. 1, pp158-169, 1966.

[7] Arto Salomaa, *Axiom systems for regular expressions of tinite automata.* Ann. Univ. Turku., Ser. A I 75 (1964)

[8] Arden, D. N. *Delayed logic and finite state machines*. In Theory of Computing Machine Design, pp. 1-35. U. of Michigan Press, Ann Arblr. 1960.

[9] John E. Hopcroft and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing, 1979

[10] Bruce P. Douglass, *Real-Time UML: Developing Efficient Objects for Embedded Systems*, Addison-Wesley, 1999.

[11] W. J. Fokkink and H. Zantema, *Basic process algebra with iteration: completeness of its equational axioms*, The Computer Journal, 37(4), pp 259-267, 1994.

[12] W. Reisig, *Petri Nets, An Introduction*, lecture notes in Computer Science, SpringerVerlag, 1985.

[13] J. Rumbaugh, I. Jacobson and G. Booch, *The Unified Modeling Language Reference Manual*, Addison Wesley, 1999.

[14] C.A.R. Hoare. *Communicating Sequential Process*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1985.

[15] Robin Milner. *Communication and Concurrency*, Prentice Hall, 1989.

[16] J.C.M. Baeten. *Applications of Process Algebra*, Cambridge Tracts in Theoretical Computer Science 17, Cambridge University Press, 1990.

[17] Robin Milner. Communicating and mobile systems: the π-calculus, Cambridge University Press, 1999.

[18] J.A. Bergstra, I. Bethke, and A. Ponse. *Process algebra with iteration and nesting*. The Computer Journal, volume 37, issue 4, 1994.

[19] J. Magee and J. Kramer. *Concurrency: State Models and Java Programs* , Wiley, 1999.

Publications

[1] 岡崎光隆, 片山卓也: 並行オブジェクトシステム動作解析のための実行スレッド自動抽出, 日本ソフトウェア科学会 第一回ディペンダブルソフトウェアワークショップ dsw2004, pp.85-94, 2004.

[2] 岡崎光隆, 青木利晃, 片山卓也, 並行オブジェクトモデルから並行スレッドモデルへの変換法, 情報処理学会 ソフトウェア工学研究会 研究報告 2003-SE-140, 2003.

[3] M. Okazaki, T. Aoki and T. Katayama: Formalizing sequence diagram and state machines using Concurrent Regular Expression, Proc. of 2nd International Workshop on Scenarios and State Machines: Models, Algorithms, and Tools SCESM'03, pp 74-79, 2003.

[4] M. Okazaki, T. Aoki, and T. Katayama, Extracting threads from concurrent objects for the design of embedded systems, Proceedings of Asia-Pacific Software Engineering Conference APSEC 2002, pp.107-116, 2002.

[5] 岡崎光隆, 青木利晃, 片山卓也: 並行動作するオブジェクトからの処理列の抽出法, 日本ソフトウェア科学会 FOSE2001, pp.147-150, 2001.