Towards the Theoretical Foundation of Choreography*

Qiu Zongyan, Zhao Xiangpeng, Cai Chao, and Yang Hongli
LMAM & Department of Informatics, School of Math., Peking University, Beijing, CHINA
zyqiu@pku.edu.cn, {zxp,caic,yhl}@math.pku.edu.cn

ABSTRACT

With the growth of interest on the web services, people pay increasingly attention to the choreography, that is, to describe collaborations of participants in accomplishing a common business goal from a global viewpoint. In this paper, based on a simple choreography language and a role-oriented process language, we study some fundamental issues related to choreography, especially those related to implementation, including semantics, projection and natural projection, dominant role in choices and iterations, etc. We propose the concept of *dominant role* and some novel languages structures related to it. The study reveals some clues about the language, the semantics, the specification and the implementation of choreography.

Categories and Subject Descriptors

H.3.5 [**Info. Sys.**]: Misc.—*Web-based services*; D.2.1 [**Soft.**]: SE—*Specifications*; D.3.1 [**Prog. Language**]: Formal Definitions and Theory—*Semantics*

General Terms

Design, Languages, Theory, Verification

Keywords

Choreography, Semantics, Implementation, Projection, Dominant Role, Dominated Choice, Dominated Loop

1. INTRODUCTION

Web services promise the interoperability of various applications running on heterogeneous platforms over the Internet, and are gaining more and more attention. Web service composition refers to the process of combining web services to provide value-added services, which has received much interest in supporting enterprise application integration.

The description of the single services locally from cooperation of other services is called an *orchestration*. The *de facto* standard for orchestration is BPEL [2].

On another level, we need to specify protocols among different services to achieve a business goal. WS-CDL [9, 8] is a W3C candidate recommendation designed for describing the collaborative observable behavior of multiple services that interact with each other, where the behaviors are performed by the participants, and the specification written in WS-CDL, called a *choreography*, provides a global view.

Extensive writing exists on the specification of parallel and distributive systems, for example CSP [7] and CCS [12]. However, almost all of them support intrinsically the specification from the local viewpoint. Even for the communication, the specification is still local, as it expresses when a process sends or receives a message from a specific channel. Furthermore, there is not a special concept of the *participants* in the computation.

With the blooming of web technology, real computations are increasingly established as a kind of processes that various computing facilities take part in, which are independent entities and might reside in any place throughout the world. For accomplishing the goal of the computation, they should not only have "correct" functionalities, but also interact with each other correctly. With the interaction becoming more complex, the problems related to specify the interaction of the participants become harder and harder. Moreover, it is even harder to verify the interaction locally. This is the motivation behind the design of WS-CDL.

However, as quoted from the specification, WS-CDL is "not an execution business process description language or an implementation language". A choreography designates a business task performed by multiple roles, but does not give an implementation composed of a set of participants. The specification of the individual participants is at the level of BPEL-like languages. For the successful development of the web-based applications, it is urgent and important to understand the approaches for connecting the global view of the choreography with the local view of each participant.

The work presented here is motivated by this requirement. It is an effort to make a clear view of choreography at an abstract level for a better understanding of what is required by the choreography to each of the roles, and further the choreography itself and related issues involved.

To explore the essence of choreography, we define a small language *Chor* as a model of simplified WS-CDL, and a simple process language for describing roles from a local viewpoint, both with formal syntax and semantics. With these models, we discuss the concept of *projection*, which maps a choreography to a set of role processes, and define the *implementation* of choreography based on projections. We propose a *natural projection* and two structural conditions related to it. Then we discuss another projection that

*Supported by NNSF of China (No. 60573081).

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

remedies the relative ordering problem where a choreography violates the sequential condition.

For the correct implementation of the choice structures in a choreography, we find that the *dominant role* of a choice is a very important concept. To ensure the roles processes take consistently the same branch in their independent version of a choice (in the choreography), when the choice is projected, there must be only one role – the dominant one – to make the real choice, and the other roles simply follow the dominator's decision. Otherwise, the consistency cannot be guaranteed.

With this recognition, we suggest to introduce *dominant role* as an explicit concept on the choreography level. We extend *Chor* and the role language by some new structures, including *dominated choice*, and *dominated loop*, with formal syntax and semantics of these structures and relatives. Furthermore, we define a projection that maps, as well as other structures, the dominated choice and dominated loop in *Chor* to the corresponding choice and loop structures for the dominant and dominated roles in the role language. As shown in the paper, the choreography designer can select the dominant role for each choice or loop as they want. The projection can generate all the necessary structures and synchronizing actions, and produce a correct implementation, i.e., the parallel composition of the resulted roles will correctly show the behavior described by the choreography.

The basic part of *Chor* and the process language are defined in Section 2 and Section 3. Section 4 is devoted to projection and implementation, where we provide a detailed discussion about natural projection and structural conditions related to this projection. With the concept of dominant role, we extend our framework in Section 5, and deal with the implementation of choices and loops. A number of related general issues are discussed in Section 6. Finally we list some related work and give a conclusion.

2. THE LANGUAGE *Chor*

Finite roles take part in a choreography C, where each role is associated with a number of basic local activities:

$$\mathcal{R}_C = \{R^1, \ldots R^n\}$$
$$locals(R^i) = \{\mathsf{a}^i_1, \ldots, \mathsf{a}^i_{n_i}\}$$

We use meta-variable a^i to denote an arbitrary activity of R^i, and use $a, a_1, \ldots$ for activities of any role.

The communication from role R^i to R^j takes the form of $c^{[i,j]}$, where c is a channel name. We use $\mathsf{c}, \mathsf{c1}, \ldots$ to represent concrete communication names in examples. We do not care about the messages transferred in communications, nor the variables receiving the messages.

The syntax of the choreography language *Chor* is given in **Fig.1** (upper part). A basic activity can be a local activity of a role R^i, a communication, or skip which does nothing. The composition structures considered here are the sequential composition, choice, and parallel composition. A choreography (a "program") is simply an activity A.

We call R^i the *performer* of local activity a^i, and call R^i and R^j the performers of $c^{[i,j]}$. We use $locals(C)$ ($locals(R^i)$) to denote the set of all local activities in C (of R^i), use $comms(C)$ ($comms(R^i)$) for the set of all communication activities in C (performed by R^i), use $acts(C)$ ($acts(R^i)$) for the set of all activities appearing in C (performed by R^i), and use $\alpha, \beta, \ldots$, possibly with subscript, to denote an arbitrary (local or communication) activity.

Figure 1: Syntax & Semantics of *Chor*

A	$::= BA$	(basic activities)
	$\mid A; A$	(sequential composition)
	$\mid A \sqcap A$	(choice)
	$\mid A \parallel A$	(parallel composition)
BA	$::= \mathsf{skip}$	(no action)
	$\mid a^i$	(activity in role R^i)
	$\mid c^{[i,j]}$	(communication)

Basic:	$[\![\mathsf{skip}]\!] \mathrel{\widehat{=}} \{\langle\rangle\}$ $\quad [\![a]\!] \mathrel{\widehat{=}} \{\langle a\rangle\}$ $[\![c^{[i,j]}]\!] \mathrel{\widehat{=}} \{\langle c^{[i,j]}\rangle\}$
Sequential:	$[\![A_1; A_2]\!] \mathrel{\widehat{=}} [\![A_1]\!] \frown [\![A_2]\!]$
Choice:	$[\![A_1 \sqcap A_2]\!] \mathrel{\widehat{=}} [\![A_1]\!] \cup [\![A_2]\!]$
Parallel:	$[\![A_1 \parallel A_2]\!] \mathrel{\widehat{=}} interlv([\![A_1]\!], [\![A_2]\!])$

2.1 A Trace Semantics of *Chor*

We consider the meaning of a choreography as the set of all possible traces of its execution. A trace is a sequence of activities of the form $\langle \alpha_1, \alpha_2, \ldots, \alpha_n \rangle$, where n is the length, and $\langle\rangle$ is the empty trace with length 0. We will use $t, t_1, \ldots$ to denote traces, and use $T, T_1, \ldots$ to denote trace sets. We use operator $\frown$ for trace concatenation, and lift it to trace sets on either or both sides:

$$t \frown T \mathrel{\widehat{=}} \{t \frown t' \mid t' \in T\}$$
$$T \frown t \mathrel{\widehat{=}} \{t' \frown t \mid t' \in T\}$$
$$T_1 \frown T_2 \mathrel{\widehat{=}} \{t_1 \frown t_2 \mid t_1 \in T_1, t_2 \in T_2\}$$

We need a function *interlv* to interleave activities of two traces and give the corresponding trace set. The definition is routine and is omitted here. We lift *interlv* to trace sets:

$$interlv(T_1, T_2) \mathrel{\widehat{=}} \{t \mid \exists t_1 \in T_1, t_2 \in T_2 \bullet \text{ such that } t \in interlv(t_1, t_2)\}$$

We also use filter operator $\downharpoonright$ on traces and trace sets. The trace(s) $t \downharpoonright S$ (or $T \downharpoonright S$) retains only the elements of S in t (or T), and keeps their orders unchanged.

Definition 1. (Trace Semantics of Choreography). The semantics of choreography C is the trace set $[\![C]\!]$ defined by rules in **Fig.1** (lower part). Here we adopt the interleaving semantics for parallel composition. □

We assume that sequential composition operator ";" has higher priority, while "$\sqcap$" and "$\parallel$" have the same priority.

2.2 Laws and Examples

Many laws hold for choreographies in *Chor*. We list some of them in **Fig.2**, where A, A_1, A_2, A_3 are arbitrary activities, and $=$ means semantical equivalence. The proofs are straightforward, and are omitted here. Many other laws can be found, which are out of the main focus of this paper.

In company with other laws, the **Unit** and **Idempotent** laws can be used to simplify a choreography by removing all unnecessary skip and replacing $A \sqcap A$ with A, while keeping the semantics unaltered. Since the choice $\sqcap$ can always be moved outward by the **Distribution laws** of $\sqcap$, any choreography can be transformed into the form $A'_1 \sqcap \ldots \sqcap A'_m$, where no choice appears in each of $A'_1, \ldots, A'_m$. Using the laws in the other direction, we can transform a choreography to a form where each $\sqcap$ is restricted to a minimal scope. Thus we have following definition.

Figure 2: Laws in *Chor*

Association and Symmetry:	
$(A_1; A_2); A_3 = A_1; (A_2; A_3)$	(; assoc.)
$(A_1 \sqcap A_2) \sqcap A_3 = A_1 \sqcap (A_2 \sqcap A_3)$	($\sqcap$ assoc.)
$(A_1 \parallel A_2) \parallel A_3 = A_1 \parallel (A_2 \parallel A_3)$	($\parallel$ assoc.)
$A_1 \sqcap A_2 = A_2 \sqcap A_1$	($\sqcap$ sym.)
$A_1 \parallel A_2 = A_2 \parallel A_1$	($\parallel$ sym.)
Unit and Idempotent:	
$\mathsf{skip}; A = A; \mathsf{skip} = A$	(; unit)
$\mathsf{skip} \parallel A = A$	($\parallel$ unit)
$A \sqcap A = A$	($\sqcap$ idem.)
Choice Distribution:	
$(A_1 \sqcap A_2); A = (A_1; A) \sqcap (A_2; A)$	(;-$\sqcap$ distr.1)
$A; (A_1 \sqcap A_2) = (A; A_1) \sqcap (A; A_2)$	(;-$\sqcap$ distr.2)
$(A_1 \sqcap A_2) \parallel A = (A_1 \parallel A) \sqcap (A_2 \parallel A)$	($\parallel$-$\sqcap$ distr.)

Figure 3: Syntax & Semantics of Role Language

$P ::= BP$ (basics)	$BP ::= \mathsf{skip}$ (no action)
$\mid P; P$ (sequential)	$\mid a$ (local)
$\mid P \sqcap P$ (choice)	$\mid c!$ (send)
$\mid P \parallel P$ (parallel)	$\mid c?$ (receive)

Skip: $\mathsf{skip} \xrightarrow{\langle\rangle} \epsilon$ **Local:** $a \xrightarrow{\langle a \rangle} \epsilon$

Sequential: $\dfrac{P_1 \xrightarrow{\sigma} P_1'}{P_1; P_2 \xrightarrow{\sigma} P_1'; P_2}$ $\epsilon; P_2 \xrightarrow{\langle\rangle} P_2$

Choice: $P_1 \sqcap P_2 \xrightarrow{\langle\rangle} P_1$ $P_1 \sqcap P_2 \xrightarrow{\langle\rangle} P_2$

Parallel: $\epsilon \parallel \epsilon \xrightarrow{\langle\rangle} \epsilon$

$$\frac{P_1 \xrightarrow{\sigma} P_1'}{P_1 \parallel P_2 \xrightarrow{\sigma} P_1' \parallel P_2} \qquad \frac{c? \in fst(P_1) \quad c! \in fst(P_2)}{P_1 \parallel P_2 \xrightarrow{\langle c \rangle} P_1/c? \parallel P_2/c!}$$

$$\frac{P_2 \xrightarrow{\sigma} P_2'}{P_1 \parallel P_2 \xrightarrow{\sigma} P_1 \parallel P_2'} \qquad \frac{c! \in fst(P_1) \quad c? \in fst(P_2)}{P_1 \parallel P_2 \xrightarrow{\langle c \rangle} P_1/c! \parallel P_2/c?}$$

Definition 2. (Choice Normal Form). A choreography of the form $A_1' \sqcap \ldots \sqcap A_m'$, with no choice in $A_1', \ldots, A_m'$, is called in the *Distributed Choice Normal Form*. A choreography in which the scope of every $\sqcap$ can not be limited further is called in the *Minimal Choice Normal Form*. □

Obviously, any choreography has an equivalent *Distributed Choice Normal Form*, and a *Minimal Choice Normal Form*.

One interesting fact is, although a choreography may have parallel structures, no deadlock can happen in its execution.

THEOREM 1 (DEADLOCK-FREENESS). *Suppose that every basic activity in a choreography will terminate, then the choreography will always terminate.*

PROOF. All traces in the trace set of a choreography can be constructed using rules in **Fig.1** unconditionally. □

Now we present some examples.

EXAMPLE 1. *Here is a simple choreography in Chor:*

$$C_1 = (\mathsf{a}_1^1 \parallel \mathsf{a}_1^2); \mathsf{c1}^{[1,2]}; \mathsf{a}_2^2; \mathsf{c2}^{[2,1]}$$

The trace set of C_1 is as follows:

$$[\![C_1]\!] = \{\langle \mathsf{a}_1^1, \mathsf{a}_1^2, \mathsf{c1}^{[1,2]}, \mathsf{a}_2^2, \mathsf{c2}^{[1,2]} \rangle, \langle \mathsf{a}_1^2, \mathsf{a}_1^1, \mathsf{c1}^{[1,2]}, \mathsf{a}_2^2, \mathsf{c2}^{[2,1]} \rangle\}$$

Here is another very simple choreography:

$$C_2 = (\mathsf{a}_1^1 \parallel \mathsf{a}_1^2); \mathsf{a}_2^1 \qquad [\![C_2]\!] = \{\langle \mathsf{a}_1^1, \mathsf{a}_1^2, \mathsf{a}_2^1 \rangle, \langle \mathsf{a}_1^2, \mathsf{a}_1^1, \mathsf{a}_2^1 \rangle\}$$

We will meet this example again in **Sec.4.1** *etc., because it shows some interesting features of choreographies.* □

3. A LANGUAGE FOR ROLES

A choreography describes the interaction among roles from a global view. It is intended to be implemented by the coordination of a set of independent processes. In order to study the relationship between the globally described choreography and the coordinative activities of each role, we define a simple language for the roles here.

The role language is given in **Fig.3**. The only difference from *Chor* is that it takes a local view on communications, where sending actions and receiving actions represent roles' local view of interactions. A sending action and a receiving action engage in a handshake when they have the same channel name and the two roles involved are ready to perform them. We can define the sets $locals(P)$, $comms(P)$ and $acts(P)$ too, as in **Sec.2**.

We define the semantics of processes as trace sets too. The semantic rules are of the form $P \xrightarrow{\sigma} P'$, meaning that process P transforms to P' after executing activities in σ, which is a sequence of activities (i.e. a trace). The rules are listed in **Fig.3** (lower part), where ϵ denotes the empty process with no command. The two rules on the lower right are for *synchronization*. Function *fst* (for *first*) is defined as:

$$fst(\epsilon) = fst(\mathsf{skip}) = fst(P_1 \sqcap P_2) \mathrel{\widehat{=}} \emptyset \qquad fst(\alpha) \mathrel{\widehat{=}} \{\alpha\}$$
$$fst(P_1; P_2) \mathrel{\widehat{=}} fst(P_1) \qquad fst(P_1 \parallel P_2) \mathrel{\widehat{=}} fst(P_1) \cup fst(P_2)$$

We define $fst(P_1 \sqcap P_2)$ as $\emptyset$ to force choices taking a branch in the first. Moreover, P/α is the process resulted from P after executing α, as defined below:

$$\mathsf{skip}/\alpha \mathrel{\widehat{=}} \bot \qquad \alpha/\alpha' \mathrel{\widehat{=}} \begin{cases} \epsilon & \text{when } \alpha = \alpha' \\ \bot & \text{when } \alpha \neq \alpha' \end{cases}$$
$$(P_1; P_2)/\alpha \mathrel{\widehat{=}} P_1/\alpha; P_2 \qquad (P_1 \sqcap P_2)/\alpha \mathrel{\widehat{=}} \bot$$
$$(P_1 \parallel P_2)/\alpha \mathrel{\widehat{=}} \begin{cases} P_1/\alpha \parallel P_2 & \text{when } \alpha \in fst(P_1) \\ P_1 \parallel P_2/\alpha & \text{when } \alpha \in fst(P_2) \\ \bot & \text{otherwise} \end{cases}$$

where $\bot$ denotes undefined, and only the defined cases are used here. In the parallel case, when both two conditions hold in the same time, we take them as two separate rules.

If no rule is applicable to a non-empty process, we say it is deadlocked. A rule is introduced to represent this situation with a special symbol $\boxtimes$:

$$\frac{P \neq \epsilon \text{ and } P \text{ is deadlocked}}{P \xrightarrow{\langle \boxtimes \rangle} \epsilon}$$

To define the trace set of processes, we define:

$$\frac{P \xrightarrow{\sigma} P'}{P \overset{\sigma}{\Longrightarrow} P'} \qquad \frac{P \xrightarrow{\sigma} P' \quad P' \overset{\sigma'}{\Longrightarrow} P''}{P \overset{\sigma \frown \sigma'}{\Longrightarrow} P''}$$

Now, we can have the definition:

Definition 3. (Traces of Process) If $P \overset{\sigma}{\Longrightarrow} \epsilon$, then σ is

Figure 4: The Natural Projection

$$
\begin{array}{lll}
nproj(\mathsf{skip}, k) & \widehat{=}\ \mathsf{skip} & \\
nproj(a^i, k) & \widehat{=}\ a^i & \text{if } k = i \\
nproj(a^i, k) & \widehat{=}\ \mathsf{skip} & \text{if } k \neq i \\
nproj(c^{[i,j]}, k) & \widehat{=}\ c^{[i,j]}! & \text{if } k = i \\
nproj(c^{[i,j]}, k) & \widehat{=}\ c^{[i,j]}? & \text{if } k = j \\
nproj(c^{[i,j]}, k) & \widehat{=}\ \mathsf{skip} & \text{if } k \neq i \wedge k \neq j \\
nproj(A_1;A_2, k) & \widehat{=}\ nproj(A_1, k); nproj(A_2, k) & \\
nproj(A_1 \sqcap A_2, k) & \widehat{=}\ nproj(A_1, k) \sqcap nproj(A_2, k) & \\
nproj(A_1 \parallel A_2, k) & \widehat{=}\ nproj(A_1, k) \parallel nproj(A_2, k) &
\end{array}
$$

called a trace of P. The semantics of process P is the trace set $[\![P]\!] \widehat{=} \{\sigma | P \stackrel{\sigma}{\Longrightarrow} \epsilon\}$. □

It is easy to see that all laws in **Fig.2** hold still for processes. Besides, although we adopt the same notation $[\![\cdot]\!]$ for two different languages, it will not cause confusion.

4. PROJECTION AND IMPLEMENTATION

A choreography is a global description of a collaborative task performed by multiple partners. We think that each role in a choreography is a concrete entities taking part in the task and should be implemented by a distinguishable, independent process. A reasonable definition of implementation is based on *projection* and *local conformance* (see **Sec.6**).

A *projection* is a procedure which takes a choreography C with n roles and delivers n processes in the role language. The combination of these role mimic the behavior of C. In the simplest case, for projection *proj*, we want to have[1]:

$$[\![proj(C,1) \parallel \ldots \parallel proj(C,n)]\!] = [\![C]\!] \quad (1)$$

In a sense, processes $proj(C,1), \ldots, proj(C,n)$ make up an implementation of C. We will use $proj(C)$ as a shorthand for $proj(C,1) \parallel \ldots \parallel proj(C,n)$.

Please note that, if the execution of the left hand side of (1) runs into deadlock, some of its traces will end with $\boxtimes$. Since no trace in $[\![C]\!]$ has $\boxtimes$ as a part, the equation will never hold. Conversely, if we have (1), the role processes will never deadlock (THEOREM 1).

No standard projection is defined in WS-CDL. We study a simple one, and then consider the problems recognized.

4.1 Natural Projection

The *natural projection* (*nproj*) defined in **Fig.4** is a simple partition following the structure of choreographies, where $[i,j]$ is taken as a part of the channel name. The projection may leave some skip and $P \sqcap P$ in role processes. A procedure can be introduced to simplify the results. We will omit this detail, and present the simplified results directly.

EXAMPLE 2. *Let us apply nproj to C_1 of* EXAM.1*:*

$$nproj(C_1,1) = \mathsf{a}_1^1; \mathsf{c1}^{[1,2]}!; \mathsf{c2}^{[2,1]}?$$
$$nproj(C_1,2) = \mathsf{a}_1^2; \mathsf{c1}^{[1,2]}?; \mathsf{a}_2^2; \mathsf{c2}^{[2,1]}!$$

It is easy to see that $[\![nproj(C_1)]\!] = [\![C_1]\!]$. However, this is not always the case. Consider C_2 in EXAM.1*:*

$$nproj(C_2,1) = \mathsf{a}_1^1; \mathsf{a}_2^1 \qquad nproj(C_2,2) = \mathsf{a}_1^2$$
$$[\![nproj(C_2)]\!] = \{\langle \mathsf{a}_1^2, \mathsf{a}_1^1, \mathsf{a}_2^1\rangle, \langle \mathsf{a}_1^1, \mathsf{a}_1^2, \mathsf{a}_2^1\rangle, \langle \mathsf{a}_1^1, \mathsf{a}_2^1, \mathsf{a}_1^2\rangle\}$$

[1]We might use other similar equations, e.g. (4) in **Sec.4.3.**

Obviously, this is not the same as $[\![C_2]\!]$ (see EXAM.1). □

Sometimes Equation (1) may fail with *nproj*. In these cases, we can blame the fault to the choreography under consideration, or to the projection used (*nproj* here), or to the semantics we choose, or even to the languages. We will investigate the problem further on some of these issues.

4.2 Restricted Natural Choreography

The *natural projection* seems intuitive. Thus, from one point of view, we can take *nproj* as a criterion to distinguish the "good" choreographies from the "bad" ones, and say that C_1 in EXAM.2 is good, while C_2 is bad.

Definition 4. (Restricted Natural Choreography). Suppose C is a choreography with n roles. We call C a *restricted natural choreography* (*RN* choreography for short), if

$$[\![nproj(C)]\!] = [\![C]\!] \quad (2)$$

Definition 4 defines a restricted level of "well-formedness". If a choreography is RN, we can efficiently partition it into a set of processes whose parallel composition shows the expected behavior. It is natural to ask a question: what structures make an RN choreography? We will propose some sufficient conditions here.

4.2.1 Sequential Composition

The problem related to sequential composition is to keep the relative order among the processes produced by the projection. We have seen a counterexample in EXAM.2, where the processes can not keep the relative order of their activities, and an extra trace presents.

To describe a condition for sequential compositions, we need to define two activity sets: $lead(A)$ includes all activities that can be executed first in the execution of A, and $end(A)$ includes activities that can be the last activity of A. These two sets can be defined recursively. The following condition guarantees that a sequential composition never breaks Equation (2):

Condition 1. (Sequential Composition). A sequential composition $A_1; A_2$ is restricted natural (RN), if it satisfies:

$$\forall \alpha_1 \in end(A_1), \alpha_2 \in lead(A_2) \bullet \quad \alpha_1 \text{ and } \alpha_2 \text{ have a common performer.} \quad (3)$$

The concept *performer* is defined in ***Sec.2***. □

Here are some choreographies where all sequential compositions presented are RN:

$$\mathsf{a}_1^1; \mathsf{a}_2^1 \qquad (\mathsf{a}_1^1 \parallel \mathsf{a}_1^2); \mathsf{c}^{[1,2]} \qquad (\mathsf{c1}^{[1,3]} \parallel \mathsf{a}_1^2; \mathsf{c2}^{[2,4]}); \mathsf{c3}^{[1,2]}; \mathsf{a}_1^1$$

The last two example show that, when one side of ";" is a parallel or a choice, we need to consider each of the branches separately. We will discuss the choices further in next subsection.

EXAMPLE 3. *A simplest counterexample is $C_3 = \mathsf{a}_1^1; \mathsf{a}_1^2$. After projection we get*

$$nproj(C_3,1) = \mathsf{a}_1^1 \qquad nproj(C_3,2) = \mathsf{a}_1^2$$
$$[\![nproj(C_3)]\!] = \{\langle \mathsf{a}_1^1, \mathsf{a}_1^2\rangle, \langle \mathsf{a}_1^2, \mathsf{a}_1^1\rangle\}$$

The relative order of a_1^1 and a_1^2 is not kept. We can add a communication to separate the two activities, and obtain an RN choreography $\mathsf{a}_1^1; c^{[1,2]}; \mathsf{a}_1^2$. Also, C_2 in EXAM.2 *can be modified into an RN one similarly. This remedy procedure will be discussed further in* **Sec. 4.3**. □

EXAMPLE 4. *Condition 1 has a sad consequence: we have no way to enforce local activities of more than two roles to terminate at the same time, when they run in different parallel branches. Consider* $C_4 = (\mathsf{a}_1^1 \parallel \mathsf{a}_1^2 \parallel \mathsf{a}_1^3); \mathsf{a}_2^1$. *No matter what and how many communications introduced, we can not make it an RN choreography. Here is a try:*

$$C_4' = (\mathsf{a}_1^1 \parallel \mathsf{a}_1^2 \parallel \mathsf{a}_1^3); \mathsf{c1}^{[1,2]}; \mathsf{c2}^{[1,3]}; \mathsf{a}_2^1$$
$$[\![nproj(C_4')]\!] = [\![C_4']\!] \cup \{\langle \mathsf{a}_1^1, \mathsf{a}_1^2, \mathsf{c1}^{[1,2]}, \mathsf{a}_1^3, \mathsf{c2}^{[1,3]}, \mathsf{a}_2^1\rangle, \langle \mathsf{a}_1^2, \mathsf{a}_1^1, \mathsf{c1}^{[1,2]}, \mathsf{a}_1^3, \mathsf{c2}^{[1,3]}, \mathsf{a}_2^1\rangle\}$$

Other arrangements will have similar results. □

Some researchers proposed to add the multipartite communication activities to solve this problem, e.g. [3].

4.2.2 Choice

The *nproj* maps a choice in a choreography to a choice in each role process. Within the parallel composition, these processes run independently, thus are not guaranteed to take consistently the same branch in the run of their own version of the choice. Inconsistent choices can result in extra traces, or even deadlock. The simplest example showing the problem is $\mathsf{a}_1^1 \sqcap \mathsf{a}_1^2$, which has trace set $\{\langle \mathsf{a}_1^1\rangle, \langle \mathsf{a}_1^2\rangle\}$. After projection and parallel composition, we get $(\mathsf{a}_1^1 \sqcap \mathsf{skip}) \parallel (\mathsf{a}_1^2 \sqcap \mathsf{skip})$ with trace set $\{\langle\rangle, \langle \mathsf{a}_1^1\rangle, \langle \mathsf{a}_1^2\rangle, \langle \mathsf{a}_1^1, \mathsf{a}_1^2\rangle, \langle \mathsf{a}_1^2, \mathsf{a}_1^1\rangle\}$.

For a choice to bring no trouble, one possibility is that at most one role process really makes a choice in its version of the choice; for each of the other processes, all branches in its version of the choice are the same, thus it makes no real choice here. We have the following definition:

Condition 2. (Choice). For a choice, if we can determine a special role, called *dominant role* hereafter, while for each of the other roles involved, all branches of those roles' version of this choice are the same and thus can be merged together, then this choice is an RN choice. □

Now we give some examples to illustrate this condition.

EXAMPLE 5. *Choreography* C_5 *includes an RN choice with communication activities:*

$$C_5 = \mathsf{a}_1^1; (\mathsf{a}_2^1; \mathsf{c}^{[1,2]} \sqcap \mathsf{a}_3^1; \mathsf{c}^{[1,2]}); \mathsf{a}_1^2$$
$$nproj(C_5, 1) = \mathsf{a}_1^1; (\mathsf{a}_2^1 \sqcap \mathsf{a}_3^1); \mathsf{c}^{[1,2]}!$$
$$nproj(C_5, 2) = \mathsf{c}^{[1,2]}?; \mathsf{a}_1^2$$

R^1 *is the dominant role here, and the choice in* R^2 *disappears. It is easy to see that* $[\![nproj(C_5)]\!] = [\![C_5]\!]$.

Here is a negative example:

$$C_6 = \mathsf{a}_1^1; (\mathsf{a}_2^1; \mathsf{c1}^{[1,2]} \sqcap \mathsf{a}_3^1; \mathsf{c2}^{[1,3]}); \mathsf{a}_1^2$$
$$nproj(C_6, 1) = \mathsf{a}_1^1; (\mathsf{a}_2^1; \mathsf{c1}^{[1,2]}! \sqcap \mathsf{a}_3^1; \mathsf{c2}^{[1,3]}!)$$
$$nproj(C_6, 2) = (\mathsf{c1}^{[1,2]}? \sqcap \mathsf{skip}); \mathsf{a}_1^2$$
$$nproj(C_6, 3) = \mathsf{c2}^{[1,3]}? \sqcap \mathsf{skip}$$

Because none of the choices in the three role processes can shrink away, we cannot have $[\![nproj(C_6)]\!] = [\![C_6]\!]$. *Furthermore, the parallel composition may run into deadlock.* □

4.2.3 Structural Theory of RN Choreography

For parallel composition, we find no extra conditions. The interaction between parallel and sequential composition has been studied. Now we consider the interaction between parallel composition and choice. EXAM.5 shows some cases for this interaction. Here are some more examples:

EXAMPLE 6. *Consider:*

$$C_7 = (\mathsf{a}_1^1; \mathsf{c1}^{[1,2]} \sqcap \mathsf{a}_2^1; \mathsf{c1}^{[1,2]}) \parallel \mathsf{c2}^{[2,1]}; \mathsf{a}_1^2$$
$$nproj(C_7, 1) = (\mathsf{a}_1^1; \mathsf{c1}^{[1,2]}! \sqcap \mathsf{a}_2^1; \mathsf{c1}^{[1,2]}!) \parallel \mathsf{c2}^{[2,1]}?$$
$$nproj(C_7, 2) = \mathsf{c1}^{[1,2]}? \parallel \mathsf{c2}^{[2,1]}!; \mathsf{a}_1^2$$

The dominant role of the choice is R^1. *No matter which branch* R^1 *chooses, it can always synchronize with* R^2. *It is not hard to see that* $[\![nproj(C_7)]\!] = [\![C_7]\!]$.

Now consider a similar example involving parallel composition of two choices with different dominant roles:

$$C_8 = (\mathsf{a}_1^1; \mathsf{c1}^{[1,2]} \sqcap \mathsf{a}_2^1; \mathsf{c1}^{[1,2]}) \parallel (\mathsf{c2}^{[2,1]}; \mathsf{a}_1^2 \sqcap \mathsf{a}_2^2; \mathsf{c2}^{[2,1]})$$
$$nproj(C_8, 1) = (\mathsf{a}_1^1; \mathsf{c1}^{[1,2]}! \sqcap \mathsf{a}_2^1; \mathsf{c1}^{[1,2]}!) \parallel \mathsf{c2}^{[2,1]}?$$
$$nproj(C_8, 2) = \mathsf{c1}^{[1,2]}? \parallel (\mathsf{c2}^{[2,1]}!; \mathsf{a}_1^2 \sqcap \mathsf{a}_2^2; \mathsf{c2}^{[2,1]}!)$$

Dominant roles of the two choices are R^1 *and* R^2, *respectively. No matter how* R^1 *and* R^2 *choose, the synchronization will be fine, and* $[\![nproj(C_8)]\!] = [\![C_8]\!]$. □

Ideally, we hope to find a set of structural conditions, that are both sufficient and necessary to distinguish RN choreographies from the others. However, we can only prove that the conditions given above are sufficient.

THEOREM 2 (RESTRICTED NATURAL CHOREOGRAPHY). *Suppose choreography* C *is in its* Minimal Choice Normal Form (*Definition 2*). *If each of the sequential compositions in* C *satisfies* Condition 1 *and each of the choices in* C *satisfies* Condition 2, *then* C *is restricted natural.*

PROOF. The proof is in our report [13]. □

A counterexample showing the conditions are not necessary is $\mathsf{a}_1^1; \mathsf{a}_1^2 \sqcap \mathsf{a}_1^2; \mathsf{a}_1^1$. It is RN but violates *Condition 1.*

4.3 Other Projections

As pointed in **Sec.4.1**, we can also blame *nproj* for the failure of Equation (1) in general. Taking another projection may make an equation similar to (1) hold for more choreographies. In previous work [13], we proposed a projection to remedy the ordering of activities among different roles. Given a choreography C with n roles, which might violate *Condition 1* (but not *Condition 2*), the projection inserts first some communications (in either direction) into C in the suitable positions to enforce the relative orderings between roles, to produce a choreography C', such that

$$[\![nproj(C')]\!] \downarrow acts(C) = [\![C]\!] \tag{4}$$

Here $\downarrow$ is the filter operation described in **Sec.2**. Note that (4) is true for some non-RN choreographies. Therefore we have an extended class of "good" choreographies.

The problem related to *Condition 2* involves the consistent choice in different roles. To solve this problem, we need to extend *Chor*. We explore the problem related to choices and iterations in the next section.

5. AN EXTENDED FRAMEWORK

Practically, choreographies with choices violating *Condition 2* are common. It is not reasonable to classify them as non-implementable. As an example, we consider now a real case, where a customer asks two sellers for the price and other relevant information of some product. After receiving two replies, the customer makes a choice based on some criterion, and informs the sellers. Depending on the customer's

choice, the sellers will either continue to trade or simply go to rest. If we made this scenario as a choreography, there must be a choice in which all the three participants need to make their choices in a consistent way. Thus, a choice structure violating *Condition 2* is inevitable here.

In the study, we find that *dominant role* is a very important concept. It is the key of *Condition 2*, where we try to determine if there is only one role who makes a real choice. Now we consider to introduce it *explicitly* into our languages. In fact, when a designer writes a choice involving multiple roles in a choreography, she/he has a clear idea in mind that one of the roles is special who makes the real choice, while the others should follow this role's decision. Reasonably, in the scenario above, the customer is the dominator, and the sellers are dominated by the customer's choice.

We will explore this idea in this section, and introduce a dominated choice structure into *Chor* as a substitution of the ordinary choice. We need also the corresponding concept to express the communication of the dominance from one role to other roles in the role language, and thus introduce a guarded choice structure with communications as guards for this. We define a new projection, an extension of *nproj*, to produce correct role processes. After the study of choices, we turn to loops. Based on the concept of *dominant role*, we introduce another new structure called *dominated loop*. We will extend the role language with some iteration structures, and define the projection rules.

From the view at the end of **Sec.4.1**, this part of the work is an effort related to the language and projection, towards correctness of Equation (1) in general. In the forthcoming discussion, we will not consider problems related to sequential composition, are they are already discussed before. The new projection proposed can deal with all the extensions. With it, all these structures are "implementable" in a reasonable sense.

5.1 Dominated Choice

Now we consider the language extension to *Chor* for the reasonable implementation of choice structures.

5.1.1 Language Extensions

The first modification to *Chor* is a new choice structure introduced to replace the ordinary one, with the form:

$$A ::= \ldots \mid A \overset{i}{\sqcap} A \mid \ldots$$

$A \overset{i}{\sqcap} A$ represents a choice with R^i as its dominant role. The semantics of this choice is the same as $A \sqcap A$ from the global view, i.e., the choreographic view. On the other hand, i plays a critical role in the projection discussed below. Here is the semantic rule for it:

$$[\![A \overset{i}{\sqcap} A]\!] \mathrel{\widehat{=}} [\![A_1]\!] \cup [\![A_2]\!]$$

All the laws held for the ordinary choice are true when we replace it with the dominated choice. An important and interesting property is:

$$[\![A \overset{i}{\sqcap} A]\!] = [\![A \overset{j}{\sqcap} A]\!]$$

even if $i \neq j$. It is not a surprise, because in choreographies, the roles always make choice consistently. The designation of i has no effect on semantics of this level, but only on the implementation. We can consider the designation as a directive, similar to the *pragma* in some programming languages, which gives the compiler clues for compiling while having no effect on the semantics.

Because $(A_1 \overset{i}{\sqcap} A_2) \overset{i}{\sqcap} A_3 = A_1 \overset{i}{\sqcap} (A_2 \overset{i}{\sqcap} A_3)$, we can introduce multi-branch choice structures as a simple extension.

Now we use the extended language to express the example discussed in the beginning of this section, a scenario where a customer wants to buy some product from two sellers.

EXAMPLE 7. *Suppose the customer is role R^1, and the two sellers are R^2 and R^3 respectively. Here is a choreography that describes the protocol of their interactions:*

$$C_9 = (\mathsf{c1}^{[1,2]}; \mathsf{a}_1^2; \mathsf{c2}^{[2,1]} \parallel \mathsf{c3}^{[1,3]}; \mathsf{a}_1^3; \mathsf{c4}^{[3,1]}); \mathsf{a}_1^1; (\mathsf{c5}^{[1,2]}; \mathsf{a}_2^2; \mathsf{c6}^{[2,1]} \overset{1}{\sqcap} \mathsf{c7}^{[1,3]}; \mathsf{a}_2^3; \mathsf{c8}^{[3,1]})$$

Initially, the customer informs sellers what is wanted. The sellers do some local work (a_1^2 and a_1^3 respectively), and then send responses to the customer ($\mathsf{c2}^{[2,1]}$ and $\mathsf{c4}^{[3,1]}$). After some local work a_1^1, the customer chooses a seller, that is described by a dominated choice. In each branch of the choice, the customer continues the trade with one seller.

Now we calculate the trace sets. To make things clear, we use T_0 to represent the trace set produced by the parallel structure. We have

$$[\![C_9]\!] = T_0 \frown \langle a_1^1 \rangle \frown \{\langle \mathsf{c5}^{[1,2]}; \mathsf{a}_2^2; \mathsf{c6}^{[2,1]} \rangle, \langle \mathsf{c7}^{[1,3]}; \mathsf{a}_2^3; \mathsf{c8}^{[3,1]} \rangle\}$$

This example will be used throughout this section. □

The role language has a real extension — the guarded choice:

$$P ::= \ldots \mid c_1?{\rightarrow}P_1 \,[\!]\, c_2?{\rightarrow}P_2$$

As to the semantics, we need only to extend the definition of *fst* and P/α, but not the semantic rules, because the guarded choice can be reduced only with other parallel branches. Here are the definitions:

$$fst(c_1?{\rightarrow}P_1 \,[\!]\, c_2?{\rightarrow}P_2) \mathrel{\widehat{=}} \{c_1?, c_2?\}$$

$$(c_1?{\rightarrow}P_1 \,[\!]\, c_2?{\rightarrow}P_2)/\alpha \mathrel{\widehat{=}} \begin{cases} P_1 & \text{when } \alpha = c_1? \\ P_2 & \text{when } \alpha = c_2? \\ \bot & \text{otherwise} \end{cases}$$

We could take a general form $c_1 \rightarrow P_1 \,[\!]\, c_2 \rightarrow P_2$, where both c_1 and c_2 can be either send or receive. However, for the problems considered here, current definition is enough. On the other hand, the multi-branch guarded choice $c_1?{\rightarrow} P_1 \,[\!]\, \ldots \,[\!]\, c_m?{\rightarrow}P_m$ is a necessity. With some extension on *fst* and P/α, it is easy to include this extended structure into our language. We omit the detailed definition here.

Now we present some laws relevant to guarded choices, where $=$ represents semantical equivalence, as in **Fig.2**:

$$c_1?{\rightarrow}P_1 \,[\!]\, c_2?{\rightarrow}P_2 = c_2?{\rightarrow}P_2 \,[\!]\, c_1?{\rightarrow}P_1$$
$$c_1?{\rightarrow}P_1; Q \,[\!]\, c_2?{\rightarrow}P_2; Q = (c_1?{\rightarrow}P_1 \,[\!]\, c_2?{\rightarrow}P_2); Q$$
$$c?{\rightarrow}P_1 \,[\!]\, c?{\rightarrow}P_2 = c?; (P_1 \sqcap P_2)$$

We assume that $[\!]$ has the same priority as $\sqcap$, thus has less grouping power than the sequential composition. We will still use $[\![\cdot]\!]$ for the semantics of both *Chor* and the role language. It causes no confusion.

5.1.2 Projection

Now we consider a projection *dproj* for the extended *Chor*, which will take care of the dominant marks on choices.

Suppose $A_1 \overset{i}{\sqcap} A_2$ is a structure in a choreography with n roles. For each number $j \neq i$, we should introduce two fresh channels, namely c'_j and c''_j. The projection of $A_1 \overset{i}{\sqcap} A_2$ on role R^j $(j \neq i)$ takes the form of a guarded choice:

$$dproj(A_1 \overset{i}{\sqcap} A_2, j) \mathrel{\widehat{=}} c'_j?{\rightarrow}dproj(A_1, j) \,[\!]\, c''_j?{\rightarrow}dproj(A_2, j)$$

For role R^i, the projection produces an ordinary choice:

$$dproj(A_1 \overset{i}{\sqcap} A_2, i) \mathrel{\widehat{=}} \gamma_1; dproj(A_1, i) \sqcap \gamma_2; dproj(A_2, i)$$

where

$$\gamma_1 = \|_{j \in i..n \wedge j \neq i}\ c'_j! \qquad \gamma_2 = \|_{j \in i..n \wedge j \neq i}\ c''_j!$$

As a result, when the execution of the roles arrive at their versions of the choice structure, role R^i makes the real choice, and notifies all the other roles which branch it selects. Thus, those roles will take the same branch in their versions of the choice consistently.

These projection rules involve consistent selection of the new channel names, thus can not be defined separately. However, the rules above are accurate enough, and can be implemented straightforwardly. The rules of *dproj* for the basic activities and other structures take the same form as what for *nproj* (**Fig.4**). We do not rewrite them again.

EXAMPLE 8. *Now consider the customer-seller choreography in* EXAM.7. *Applying dproj for each role, we have:*

$$\begin{aligned}
dproj(C_9, 2) &= \mathsf{c1}^{[1,2]}?; a_1^2; \mathsf{c2}^{[2,1]}!; \\
&\quad (\mathsf{c9}^{[1,2]}?{\rightarrow}\mathsf{c5}^{[1,2]}?; a_2^2; \mathsf{c6}^{[2,1]}! \,[\!]\, \mathsf{c10}^{[1,2]}?{\rightarrow}\mathsf{skip}) \\
dproj(C_9, 3) &= \mathsf{c3}^{[1,3]}?; a_1^3; \mathsf{c4}^{[3,1]}!; \\
&\quad (\mathsf{c11}^{[1,3]}?{\rightarrow}\mathsf{skip} \,[\!]\, \mathsf{c12}^{[1,3]}?{\rightarrow}\mathsf{c7}^{[1,3]}?; a_2^3; \mathsf{c8}^{[3,1]}!) \\
dproj(C_9, 1) &= \mathsf{c1}^{[1,2]}?; a_1^2; \mathsf{c2}^{[1,2]}?; a_1^2; \\
&\quad (\ \ (\mathsf{c9}^{[1,2]}! \parallel \mathsf{c11}^{[1,3]}!); \mathsf{c5}^{[1,2]}!; \mathsf{c6}^{[1,2]}? \\
&\quad \ \sqcap (\mathsf{c10}^{[1,2]}! \parallel \mathsf{c12}^{[1,3]}!); \mathsf{c7}^{[1,3]}!; \mathsf{c8}^{[1,2]}?)
\end{aligned}$$

We list the process for R^1 as the last one only for making the presentation clearer. When R^1 takes the first branch in the choice, R^2 and R^3 will receive a notification from channels $\mathsf{c9}^{[1,2]}$ and $\mathsf{c11}^{[1,3]}$ respectively. Then R^3 will end immediately, while R^2 going on its work further. If R^1 takes the other branch, things will be reversed.

Now we consider an interesting modification, to see what happens if we make R^2 the dominator of the choice. We call this modification C'_9, and have the following results:

$$\begin{aligned}
dproj(C'_9, 1) &= \mathsf{c1}^{[1,2]}?; a_1^2; \mathsf{c2}^{[1,2]}?; a_1^2; \\
&\quad (\ \mathsf{c9}^{[2,1]}?{\rightarrow}\mathsf{c5}^{[1,2]}!; \mathsf{c6}^{[1,2]}? \\
&\quad \ [\!]\, \mathsf{c10}^{[2,1]}?{\rightarrow}\mathsf{c7}^{[1,3]}!; \mathsf{c8}^{[1,2]}?) \\
dproj(C'_9, 3) &= \mathsf{c3}^{[1,3]}?; a_1^3; \mathsf{c4}^{[3,1]}!; \\
&\quad (\ \mathsf{c11}^{[2,3]}?{\rightarrow}\mathsf{skip} \\
&\quad \ [\!]\, \mathsf{c12}^{[2,3]}?{\rightarrow}\mathsf{c7}^{[1,3]}?; a_2^3; \mathsf{c8}^{[3,1]}!) \\
dproj(C'_9, 2) &= \mathsf{c1}^{[1,2]}?; a_1^2; \mathsf{c2}^{[2,1]}!; \\
&\quad (\ \ (\mathsf{c9}^{[2,1]}! \parallel \mathsf{c11}^{[2,3]}!); \mathsf{c5}^{[1,2]}?; a_2^2; \mathsf{c6}^{[2,1]}! \\
&\quad \ \sqcap (\mathsf{c10}^{[2,1]}! \parallel \mathsf{c12}^{[2,3]}!){\rightarrow}\mathsf{skip})
\end{aligned}$$

Although this implementation may be thought unfair because R^2 might tend to make the choice self-beneficially, the implementation works perfectly.

Please notice that both $dproj(C_9)$ and $dproj(C'_9)$ have the intended behavior. In any run, the role processes will perform a course of activities described in the choreography, with some auxiliary synchronization. Furthermore, they will never run into deadlock. □

5.1.3 Retrospect

We list some interesting issues related to dominated choices here:

- The communications inserted by *dproj* are inevitable for obtaining a consistent implementation. This kind of synchronization is rather tedious if written by hand. However, they can be generated automatically as shown. This benefits choreography designers.
- The communications inserted by *dproj* serve only for synchronization. They are not substantial in the sense they do not imply any real work. After filtering them out with $\downarrow acts(C)$, we obtain the original trace set. Thus, what we achieve is an equation similar to (4).
- From the $[\![A \overset{i}{\sqcap} A]\!] = [\![A \overset{j}{\sqcap} A]\!]$ and an equation similar to (4), we know that with any set of the designations of dominant roles for each choice in the choreography, the implementation obtained will be equivalent on an abstract level, (i.e. the choreography level). In this case, choreography designers can choose any role as the dominator for a choice, thus they can choose the most reasonable one. The projection will take care of all details of the implementation. A formal proof of such a theorem is an aspect of our ongoing work.
- The projection rules given are for the most general cases. We have many opportunities to reduce the unnecessary communications for a more efficient implementation. This can be seen as optimization, and is an interesting future work.

Thinking about the problem related to the choice structure further, we can find that the dominated choice $A \overset{i}{\sqcap} A$ is not a necessity theoretically. In fact, we can make an arbitrary choreography C "implementable" by randomly selecting one role as the dominator for each choice in C, and use the technique presented above to produce the role processes. The result will work fine and is deadlock-free.

However, it makes a real sense for the choreography designer to determine the dominant roles, because the random selection might produce a result that is unacceptable practically. As shown in the customer-seller example, taking one seller as the dominant role of the choice, we can also get a consistent implementation in the role process level. But this designation is unacceptable practically, because the dominant seller will make the choice in favor of her/himself generally, resulting in an unfair implementation.

Anyway, *dominant role* is the most important concept in the implementation, no matter whether explicitly specified or randomly selected.

5.2 Dominated Loop

One important weakness of the framework developed yet is that the choreographies can only have behavior with finite traces, because there is no iterative or recursive structure in the languages. Now we extend *Chor* further to deal with

this problem. With the help of the concept *dominant role*, we have a clear approach already.

5.2.1 Language Extensions

People usually use $*P$ to denote an abstract loop, which causes its body P to execute zero of more times. Theoretically, $*P$ is in fact a choice with infinite branches

$$*P = \mathsf{skip} \sqcap P \sqcap (P;P) \sqcap (P;P;P) \sqcap \ldots$$

Thus we can apply the same technique for choices to introduce loop structures into our languages. Because all roles should go consistently on either iterating further or exiting, we need a dominant role to make the decision and notify the others, to enforce them going always the same way.

We introduce a new structure *dominated loop* into *Chor*, with the syntax:

$$A ::= \ldots \mid \overset{i}{*}\, A$$

It means that activity A will be executed zero or more times, and R^i is the dominant role of this loop who decides if the next iteration is necessary.

To define the semantics of $\overset{i}{*}\, A$, we introduce the star operator $\langle\cdot\rangle^*$ on traces. For any trace t, we have:

$$t^0 \mathrel{\widehat{=}} \langle\rangle \qquad t^{n+1} \mathrel{\widehat{=}} t^n \frown t \qquad t^* \mathrel{\widehat{=}} \bigcup_{i\in\mathbb{N}} \{t^i\}$$

where $\mathbb{N}$ is the set of natural numbers. We lift the star operator to trace sets:

$$T^* \mathrel{\widehat{=}} \bigcup_{t\in T} t^*$$

Now we can define the semantics of a loop:

$$[\![\overset{i}{*}\, A]\!] \mathrel{\widehat{=}} [\![A]\!]^*$$

We assume that $*$ has the highest priority. Thus, it applies only to the minimal structure after it. Here are some laws for the loop structure:

$$\begin{aligned}
\overset{i}{*}\, A &= \mathsf{skip} \sqcap A; \overset{i}{*}\, A \\
\overset{i}{*}\, (A_1; A_2) &= \mathsf{skip} \sqcap A_1; \overset{i}{*}\, (A_2; A_1); A_2 \\
\mathsf{skip} \sqcap \overset{i}{*}\, A &= \overset{i}{*}\, A \\
A \sqcap \overset{i}{*}\, A &= \overset{i}{*}\, A \\
\overset{i}{*}\, A &= \overset{j}{*}\, A
\end{aligned}$$

The last law holds even if $i \neq j$. Thus, the designation of the dominant role is also a directive with no effect on the (globally-viewed) semantics of the choreography.

We need to introduce two new structures into the role language, to express the behavior of the dominant and dominated roles. The dominant role iterates on its own, while the dominated role should continue or stop its iteration according to the notification from the dominator. We call the structure for the dominated roles as *dominated loop* in the role language, which is a kind of guarded command:

$$P ::= \ldots \mid *P \mid c_1? \,[\!]\!*\, c_2? \!\rightarrow P$$

The process $*P$ repeats the behavior of P by zero or more times. For the *dominated loop* $c_1? \,[\!]\!*\, c_2? \!\rightarrow P$, if a handshake happens on c_1, the loop terminates immediately; otherwise, if a handshake happens on c_2, then the loop waits for the next handshake after the execution of P. The loop can be seen as a kind of guarded loop proposed by Edsgar W. Dijsktra, with a breaking mechanism. Because it is not symmetric, we adopt a non-symmetric symbol $[\!]\!*$.

Here are the semantical rules for the simple loop:

$$*P \xrightarrow{\langle\rangle} \epsilon \qquad * P \xrightarrow{\langle\rangle} P; *P$$

For the dominated loop, we need not to define new rules, but only to extend the definition of *fst* and P/α, as what we did for the guarded choice. Here are the relevant definitions:

$$\begin{aligned}
\mathit{fst}(*P) &\mathrel{\widehat{=}} \emptyset \\
\mathit{fst}(c_1? \,[\!]\!*\, c_2? \!\rightarrow P) &\mathrel{\widehat{=}} \{c_1?, c_2?\} \\
(c_1? \,[\!]\!*\, c_2? \!\rightarrow P)/\alpha &\mathrel{\widehat{=}} \begin{cases} \epsilon & \text{when } \alpha = c_1? \\ P & \text{when } \alpha = c_2? \\ \bot & \text{otherwise} \end{cases}
\end{aligned}$$

We assume that $*$ has the highest priority too, and $[\!]\!*$ has the same priority as $\sqcap$ and $\|$, thus has less grouping power than the sequential composition.

5.2.2 Projection

Now we consider the part of *dproj* related to dominated loops. Suppose C is a choreography with n roles including $\overset{i}{*}\, A$ as a part. When this dominated loop is projected into the role language, there will be an ordinary loop in the process corresponding to R^i, and a dominated loop in each of the rest role processes.

For each role R^j where $j \neq i$, we need to introduce two fresh channels c'_j and c''_j, with the projection rule:

$$\mathit{dproj}(\overset{i}{*}\, A, j) \mathrel{\widehat{=}} c''_j? \,[\!]\!*\, c'_j? \!\rightarrow \mathit{dproj}(A, j)$$

For role R^i, we have:

$$\mathit{dproj}(\overset{i}{*}\, A, i) \mathrel{\widehat{=}} *(\gamma_1; \mathit{dproj}(A, i)); \gamma_2$$

where we have also

$$\gamma_1 = \|_{j\in i..n \wedge j\neq i}\, c'_j! \qquad \gamma_2 = \|_{j\in i..n \wedge j\neq i}\, c''_j!$$

Let us see an example:

EXAMPLE 9. *Now consider an extension of the scenario described in* EXAM.7. *Suppose in this time, the customer wants to buy a number of products from the two sellers:*

$$C_{10} = \overset{1}{*}\, C_9$$

There is a simple dominated loop, where each separate trade is described by choreography C_9. For implementation, we project C_{10} to the three roles:

$$\begin{aligned}
\mathit{dproj}(C_{10}, 2) &= \mathsf{c13}? \,[\!]\!*\, \mathsf{c14}? \!\rightarrow \mathit{dproj}(C_9, 2) \\
\mathit{dproj}(C_{10}, 3) &= \mathsf{c15}? \,[\!]\!*\, \mathsf{c16}? \!\rightarrow \mathit{dproj}(C_9, 3) \\
\mathit{dproj}(C_{10}, 1) &= *((\mathsf{c14}! \parallel \mathsf{c16}!); \mathit{dproj}(C_9, 1)); (\mathsf{c13}! \parallel \mathsf{c15}!)
\end{aligned}$$

The results of projecting C_9 have been given in EXAM.8, *and are omitted here. It is easy to verify that the parallel composition of these three processes shows the intended behavior.*

We can also designate a seller, e.g., R^2, as the dominant role of the loop, and also as the dominator of the choice embedded in the loop:

$$C'_{10} = \overset{2}{*}\, C'_9$$

In this case, we have

$$\begin{aligned}
\mathit{dproj}(C'_{10}, 1) &= \mathsf{c13}? \,[\!]\!*\, \mathsf{c14}? \!\rightarrow \mathit{dproj}(C'_9, 1) \\
\mathit{dproj}(C'_{10}, 3) &= \mathsf{c15}? \,[\!]\!*\, \mathsf{c16}? \!\rightarrow \mathit{dproj}(C'_9, 3) \\
\mathit{dproj}(C'_{10}, 2) &= *((\mathsf{c14}! \parallel \mathsf{c16}!); \mathit{dproj}(C_9, 2)); (\mathsf{c13}! \parallel \mathsf{c15}!)
\end{aligned}$$

Now seller R^2 can sell as many products as she/he wants. Although hard to accept, the implementation works fine. □

All the discussions in **Sec.5.1.3** are applicable to the loops. Especially, we can also use ordinary loop structures and let the projection to choose randomly a dominator for each loop in the choreography, and make a viable implementation with the technique shown above. However, from the practical viewpoint, taking the concept of dominant role explicitly makes a real sense.

6. REMARKS

In this paper we present some preliminary results of an effort towards the theoretical foundation of the choreography. Many related issues are not touched or only partially developed. We discuss some of them in this section.

Implementation. WS-CDL document [9] defines a choreography as a multi-participant contract from a global perspective. It does not define clearly the "correct" implementation of a choreography, but proposes only the concept *conformance*. It says "each participant can then use the global definition to build and test solutions that conform to it. The global specification is in turn realized by combination of the local systems, on the basis of appropriate infrastructure support". This leaves the definition of implementation open, and creates confusion.

Some researchers take a *global view* where a choreography is thought just a (business) process, and an implementation is any of the processes which shows the behavior required by the choreography. For example, N. Busi *et al.* [4] gave a definition following this vein. On the other hand, we take the *role-based view* or *local view*, and consider that a choreography describes a (business) process implemented by a set of roles. An implementation is a set of processes where each one implements one of the roles in a clear way, and the combination of them has the required behavior.

The fundamental dissimilitude of these two viewpoints is whether the roles and activities described in the choreography must have their reincarnations in any admitted implementation. The local view says that these details are really important, but the global view says it is not the case.

We think that the definition of implementation based on the *global view* is too loose. The extreme case is an implementation with only one process, which executes all activities of all roles in the choreography, and makes all the communications as local assignments. The approach proposed here takes the way of projection and local conformance, where each role is a real process in any valid implementation which runs independently in parallel with other processes corresponding to other roles. Thus, the projections is very important in the implementation, and should be studied further. We also defined and studied the concept of *local conformance* in previous work [13].

Semantics. The semantics used for *Chor* and generally, for choreographies, is also a topic to study. We take here all the relative order between activities described in choreography significant, even if they appear in different (and independent) roles, thus distinguish choreographies such as

$$a^1; a^2 \qquad a^2; a^1 \qquad a^1 \parallel a^2$$

Is the distinction important in the choreography world? All related work we read regards the relative speed of roles, but without a discussion about its rationality. However, from our view of point, this is questionable.

Although the semantics used here is useful in exploring properties and problems of choreographies, it may not be the best choice, because the intention here is to describe collaborations between independent web services. Neglecting the independency of roles, we may obtain some results which are not important practically.

Furthermore, to ensure the relative order, we need to insert many synchronizing actions (**Sec.4.3**) which may be "unnecessary", and produce a slow implementation. If we do no care about the relative speed of different roles, we can get rid of *Condition 1*, and may obtain more effective implementations. Now we are looking for other reasonable semantics that can reflect the situation better. We hope that with the semantics, the correctness of our implementation approach can be proved formally.

In fact, here we have a dilemma: if we allow roles to run freely except observing the explicit synchronization written in the choreography, the three choreographies listed above will be semantically indistinguishable. This might bring confusions to the choreography designers too.

Language. As shown in this paper, we are still in the early stage of defining a language for choreography. We propose some novel structures for the choreography language and the process language in this paper, based mainly on the implementation view point. The most important concept is *dominant role*. From our view, this concept must have its position in any language designed for choreographies.

There are some other clues about the design of the languages. In our previous paper [13], we gave a simple example which shows one weak point of the common structural flow structures for the description of choreography. Another example is the controversial issues about the control structure *workunit* of WS-CDL [3]. Aalst *et al.* listed some challenges including defining a "real" choreography language in their paper [16]. It seems that lot work should be done before a widely accepted result in this field.

7. CONCLUSION

The rapid-developing web technology is creating an environment where increasingly computation tasks are carried out by multiple processes (services) residing over the Internet. Due to the nature of web services, to guarantee the correct interaction of independent, communicating services become even more critical [14]. This is the motivation under the development of WS-CDL [9] for specifying choreography, i.e., the global observation of business protocol among participants (roles). Because of these situations, a deep and thorough understanding of the essence of choreography and relevant problems is urgent and important.

N. Busi *et al.* formalized choreography and orchestration using process algebra, with a global view on conformance [4] (see **Sec.6**). J. Mendling and M. Hafner proposed a simple projection from WS-CDL to BPEL [11] without a discussion about its correctness. Another interesting reference is the Pi4SOA project [1]. In our previous work, we defined formal models for choreography, and proposed and studied the natural projection and Equation (1), in the context of verification [17] and implementation [13].

As the work related to ours the most, Carbone, *et al.* [5] studied a two-level paradigm for the description of communication behaviors, on the global message flows and endpoint behavior levels. Three principles for well-structured global description and a theory for projection were developed, with a condition and a framework for relating the global description of communication-centric software to its local correspondents. Fu *et al.* [6] proposed a framework for modeling global behaviors of electronic services composed by autonomous peers. The global protocol is specified with Büchi automata. A trace semantics recording the sending messages is given, while projection of protocol, projection of trace, and the semantics of composed peers are defined. The Sequencing Constraints language of the SSDL protocol framework [15] pays attentions on the description of protocols for multiparty collaboration using message-oriented programming abstractions. Li and He [10] discussed also the relationship between choreography and orchestration. Compared with our work, none of these work introduced the concept of dominant role explicitly into the languages, nor dominated choice and dominated loop structures.

In this paper, many basic concepts related to choreography are studied. A small language *Chor* for choreography and a simple process language for the roles from local viewpoints are defined, both with formal syntax and semantics. Based on them, we discuss the concept of projections, which map a given choreography into a set of role processes. A special mapping named *natural projection* is discussed in detail. With this projection, we define a level of well-formedness, and propose two structural conditions. We discuss also a projection which can remedy the sequential order problem of choreographies.

The most important contribution of this paper is recognition of the concept *dominant role* which is critical in the implementation of any choice or iteration structures in choreographies. We suggest to take *dominant role* as a language-level concept, and attach it to each choice and loop structures. With this concept, choreography designers can express their desire clearer. And we can have also a clearer model for the implementation of choreographies.

Based on the concept *dominant role*, we extend our languages with some novel language structures related to dominant roles, including dominated choice and dominated loop structures on both choreography and role process levels. We propose a projection, which is an extension of the natural projection with rules about dominated choices and dominated loops. There are different rules for the dominant role and dominated roles. As a result, the parallel composition of the generated role processes will be an implementation of the choreography. We show that the selection of the dominant role for each choice or loop has no effect on the choreography-level semantics, but only on the implementation. Thus, the choreography designer can make their selection as desired, and the projection will always generate a correct implementation, where all necessary actions to synchronize the processes are generated automatically.

We discuss some additional problems related to choreography in the paper, including the local viewpoint about the implementation vs. the global one, the rationality of the semantics, etc. A number of examples are presented in some sections to help the readers get better understanding of the properties of choreographies, and of the interesting phenomena appeared in this field.

8. ACKNOWLEDGMENTS

The authors would like to thank Shengchao Qin and Xueshan Feng for their helpful comments.

9. REFERENCES

[1] Pi4SOA. `http://www.pi4soa.org/`.

[2] T. Andrews, F. Curbera, H. Dholakia, *et al.* Business Process Execution Language for Web Services, version 1.1, 2003.5. `http://ifr.sap.com/bpel4ws/`.

[3] A. Barros, M. Dumas, and P. Oaks. A Critical Overview of the Web Services Choreography Description Language. `www.bptrends.com`, 2005.

[4] N. Busi, R. Gorrieri, C. Guidi, *et al.* Choreography and orchestration conformance for system design. In *Coordination 2006, LNCS 4038*. Springer, 2006.

[5] M. Carbone, K. Honda, and N. Yoshida. Structured global programming for communication behaviour. `http://www.pi4tech.com/xwiki/bin/view/research/papers`, 2006.

[6] X. Fu, T. Bultan, and J. Su. Conversation protocols: A formalism for specification and verification of reactive electronic services. In *Proceedings of CIAA 2003, LNCS 2759*, pages 188–200. Springer, 2003.

[7] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[8] N. Kavantzas. Aggregating web services: Choreography and WS-CDL. Technical report, Oracle Corporation, 2004.

[9] N. Kavantzas, D. Burdett, G. Ritzinger, *et al.* Web Services Choreography Description Language, version 1.0, 2005. `http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/`.

[10] J. Li and J. He. Conformance validation between choreography and orchestration. Software Eng. Inst., East China Normal University, manuscript, 2006.

[11] J. Mendling and M. Hafner. From inter-organizational workflows to process execution: Generating BPEL from WS-CDL. In *OTM 2005, LNCS 3762*. Springer, 2005.

[12] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[13] Z. Qiu, C. Cai, X. Zhao, and H. Yang. Exploring into the essence of choreography. Preprint 2006-63 of Institute of Mathematics, Peking University, `http://www.math.pku.edu.cn:8000/en/preindex.php`.

[14] G. Salaun, L. Bordeaux, and M. Schaerf. Describing and reasoning on web services using process algebra. In *2nd Inter. Conf. on Web Services*. IEEE, 2004.

[15] J. W. Simon Woodman, Savas Parastatidis. Sequencing Constraints SSDL Protocol Framework. `http://ssdl.org`.

[16] W. van der Aalst, M. Dumas, A. ter Hofstede, *et al.* Life after BPEL? In *WS-FM 2005, LNCS 3670*. Springer, 2005.

[17] X. Zhao, H. Yang, and Z. Qiu. Towards the formal model and verification of web services choreography description language. In *WS-FM 2006, LNCS 4184*. Springer, 2006.