Chapter 1

INTRODUCTION

1.1. Informal Definition of The Subsystem Construction Problem

In the design of distributed systems, such as communication protocols, protocol converters and controllers for discrete event systems, we often need to solve the following problem: given a requirement specification $S_c$ and a subsystem $M_0$, it is necessary to generate a missing subsystem $C$ such that the two subsystems, $M_0$ and $C$, constitute the system $M$ and satisfy the overall abstract specification $S_c$, as shown in Fig.1.1.

![Fig.1.1 The subsystem construction problem.]

The requirement specification $S_c$ is also called the *service specification* of the system, which specifies what services the system $M$ should provide to the system environment. The lines with arrows in Fig.1.1 denote the interaction between $M_0$ and $C$, and the interaction between the system and its environment.
The nature of the subsystem construction depends upon the service Sc to be provided, the characteristics of the subsystem M0 and how M0 and C interact. For example, is the interaction between M0 and C synchronized or through message passing? Which events in M0 can be observed and controlled by C (i.e., can be disabled when desired)?

The attempt to solve the subsystem construction problem raises several questions: What constitutes a correct solution C, i.e., what does the term "satisfy" means? Under what conditions is such a solution possible? And how to construct such a subsystem C? To answer these questions rigorously, we turn to formal methods, in which concurrent systems such as the subsystems M0 and C, and the service specification Sc are modeled by mathematical objects (e.g., labelled transition systems in this thesis). Experiences have shown that such methods are the best way to deal with the complexities of reasoning about the behaviour of parallel and distributed systems.

There are several notions used to define what "M satisfies Sc" means, for instance the testing preorder [Clea93] [Brin87]1, which will be defined in Chapter 2. In addition, the following requirements should also be considered

1) If the service specification Sc contains real-time requirements, the constructed system should also satisfy these timing requirements. This is important for real-time system design.

2) The algorithm should be general, i.e., it should be able to solve a large class of problems, instead of a few special ones. The algorithm should have low computational complexity, and be suitable for implementation.

1.2. The Motivation of the Project

Our work on the subsystem construction problem is motivated by the possible applications that require a solution to this problem. Below we will briefly introduce several important

---

1 In [Brin87] the testing preorder is called reduction relation. We call it testing preorder in this thesis because its relation with testing equivalence [Clea93] [Clev92].
applications including network protocols, protocol conversion, and discrete event system controllers.

1.2.1. Network Protocol Design

A protocol is a set of rules governing the interactions between communicating entities, which is often described at two different levels of abstraction: the service specification and the protocol specification, respectively. A service specification of a protocol describes the behavior of the protocol from the user's viewpoint based on a set of service primitives through which users can access the provided services. A service primitive is considered as an elementary interaction between a service user and the service provider. A service specification defines the constraints on service primitives by specifying the valid sequences of the service primitives. A protocol specification describes the behavior of each entity within the protocol in response to the requests from its users, messages from the other entities via the channels, and internally initiated actions (time-outs for example). More detailed explanation about these concepts can be found in [Boch90b] [Boch90c].

Recently, the service concept has been widely used in protocol synthesis [Boch86] [Prob91] [Ferh89] [Kant96] [Khou95] [Khou94b]. The protocol design problem shown in Fig.1.2 consists of finding the protocol entities D1 and D2 from the given service specification Sc and the channel specification Ch, such that the interaction between D1 and D2 via the channel provides the services specified by Sc.

If one of the protocol entities, for instance D1, is given, then we can construct the other protocol entity D2 by solving the subsystem construction problem. Therefore, the protocol design problem becomes a problem of communicating subsystem construction. Examples of this kind of applications can be found in [Merl83] and [Parr89].

![Fig.1.2 A protocol model.](image-url)
One difficulty with the application is how to obtain one of the protocol entities. This is often considered as another problem, and we will not discuss it in this thesis.

1.2.2. Protocol Conversion

One of the difficulties that arise in interconnecting heterogeneous networks is the problem of protocol mismatches [Gree86] - incompatible protocols are used in the heterogeneous networks, which makes it impossible to communicate with each other directly. To overcome the protocol mismatch problem, a protocol converter can be used as an intermediary between the incompatible protocols. The protocols and the converter together form an internetworking system.

Consider two given protocols D and F (Fig.1.3). Suppose the two protocols provide similar services, but differ in certain details, and we want D1 ( F1) to interoperate with F2 (D2) via a protocol converter C. Note that the two entities of a protocol are represented by different symbols, since they may be different in function. The converter C has to perform message mapping for the messages received from one protocol and send them to the other protocol such that the same semantic is kept; further, instead of mapping a single message one by one, it is often necessary to consider a sequence of messages in one protocol and map them into a sequence of messages in the other protocol in such a manner that the interconnected systems provide the required services specified by S\(_c\) as shown in Fig.1.4. The service specification S\(_c\) defines the minimal properties required by the users (higher layers) of the interworking protocols, called a *global service specification*. Note that we consider only D1 interworking with F2 here. D2 interworking with F1 can be constructed similarly.

Since the components D1, Chd, Chf, and F2 are given and can be composed into a single subsystem M0 containing these components, it is therefore clear that the protocol conversion problem is a subsystem construction problem, as shown in Fig.1.4(b). We may see that this is a special case of subsystem construction since the converter C does not directly interact with the service user. In Chapter 3, we will give a general survey of existing methods for protocol conversion.

In certain cases, two protocols may not be able to provide the service specified by a given specification S\(_c\), for any converter C; this is the so-called *hard mismatch* problem defined in [Gree86]. In this case, service enhancement is needed in order to achieve the goal of interworking since the two protocols are unconvertible.
1.2.3. Discrete Event System Controllers [Rama89]

A discrete event system is a dynamic system that evolves, in accordance with abrupt occurrences of physical events, at possibly unknown irregular time intervals. For example, an event may correspond to the arrival or departure of a customer in a queue, the completion of a task or the failure of a machine in a manufacturing system, transmission of a packet in a computer network, or the occurrence of a disturbance or a changed set point in
Fig. 1.1 can be considered as the interaction between a discrete system M0 and its controller C, as shown in Fig.1.5. The control consists of disabling or enabling controllable events through switching a sequence of elements in \( \gamma \) (the control input) on and off, in response to the observed sequence \( \beta \) previously generated by M0. Such a controller is often called a supervisor. The controller design problem is to design a controller that selects control inputs in such a way that the given discrete event system behaves according to a constraint \( S_c \). Roughly, the constraint \( S_c \) can be viewed as requiring that certain undesirable sequences of events, which may happen according to the specification of M0, are not permitted to occur, while at the same time, certain other desirable sequences of M0 are permitted. C and M0 are assumed to run in parallel and interact in the following fashion. When M0 is at state q and C is at state p, an event e can occur only if e is possible in both C and M0 at the point (q, p). This implies that those transitions of M0 disabled by C do not appear in the transition structure of C; while those transitions enabled by C and which are possible in M0 do appear in the transition structure of C. The sequences of events generated while M0 is under the control of C is called the closed-loop behavior of the system.

From the above discussion and Fig.1.5, we see that the control problem for discrete event systems can be considered as a special case of the problem of subsystem construction.

### 1.3. Current Practice and Problems about Subsystem Construction

A number of formal methods have been developed to solve the subsystem construction problem. Merlin and Bochmann first proposed a solution [Merl83] under trace equivalence for the problem shown in Fig.1.6, where the subsystem C interact with the system.
environment. The algorithm proposed in [Calv89] solves a special case of the problem described in Fig.1.6, where the subsystem C does not interact with the system environment. Since a safety property and a progress property (these properties will be explained after Definition 2.17) are used for the correctness, the constructed system has no deadlock. Recently, a method was proposed to use the bisimulation relation for comparing the constructed system with the given service specification [Parr89] [Qin91]. There are two common limitations of these methods:

1) A common feature of these methods is that the algorithms are developed under the assumption that the events that synchronize the execution between M0 and C are all controllable by the subsystem C. Therefore, it is difficult to apply the methods to certain parallel and distributed systems. For example, in the design of controllers for discrete event systems, one usually assumes that an observable event is not necessarily controllable.

2) The service specification must be deterministic for the methods proposed in [Parr89], [Qin91] and [Calv89]. This may limit the application of the methods. The concept of a deterministic finite state model will be defined in Chapter 2.

3) The service specification Sc, subsystems M0 and C are specified by a model without timing constraints. Two problems may be raised by the lack of timing constraints. One is the performance problem: without a timed model, we can not quantitatively analyze the system behaviour and its performance. The other is the correctness problem: In certain systems, the correctness may depend on the timing of certain events.

Concerning the methods dedicated to controller design, Ramadge and Wonham [Rama89] have surveyed a number of papers in the literature. To the author’s best knowledge, all the
existing methods in the field of controller design only use trace equivalence (or trace inclusion, to be defined in Chapter 2) to compare a given constraint with the behaviour of the controlled system. The reason is that existing methods for controller design do not consider the possible interaction between M0 and other systems. When the interaction between M0 and its environment should be taken into account, one often has to deal with nondeterministic systems (the protocol conversion problem can be considered as a special case of such a controller design [Calv89]). In this case, trace equivalence is not strong enough to avoid deadlock.

1.4. Objectives

Observing all the problems described above, new methods are required to solve the subsystem construction problem. The objective of our work is to develop a new design method for the subsystem construction problem, in order to overcome the limitations of the existing methods mentioned above. In particular, our work is aimed at:

1) Generalizing the communicating subsystem construction methods described in [Merl83], [Parr89], [Calv89] and [Qin91], by considering the case that not all observable events are necessarily controllable by C.

2) Developing an algorithm for subsystem construction that is able to deal with nondeterministic service specification (i.e., the design constraint).

3) Extending the algorithm to a timed model.

4) Showing applications of the algorithm in protocol engineering and controller design for discrete event systems.

1.5. Contributions

The original contributions of this thesis can be summarized in the following three aspects:

- A method for the diagnosis of testing preorder. Compared with existing methods, our method saves both time and space significantly. This method can be applied not only to subsystem construction, but also to other software design problem using testing preorder. This result is a by-product of our work on subsystem
construction.

- An algorithm for communicating subsystem construction. This algorithm
  1) generalizes the existing subsystem construction methods such that an observable event is not necessarily controllable (objective 1);
  2) can be applied to the nondeterministic service specification (objective 2); the importance of dealing with nondeterministic service specifications will be discussed in Chapter 6 by an example;
  3) generalizes the existing controller design methods by using the testing preorder for comparing the constructed system and its service specification;
  4) has been extended to a dense time model for timed subsystem construction (objective 3).

- A protocol conversion method for constructing optimized protocol converters by using our subsystem construction algorithm. The method has the following advantages in comparison with related works [Calv89] (objective 4):
  1) An optimized protocol converter is generated.
  2) The algorithm can be applied to nondeterministic service specification.
  3) The computation may be less complex.

Most of the content in this thesis has been published in [Tao95], [Tao95a], [Tao95b], and [Tao95c].

1.6. Outline of the Thesis

This thesis is organized as follows. In Chapter 2, mathematical models, the finite labelled transition system [Nico87], the finite state models with dense time [Alur92][Khou95] [Khou94b], and related notions are defined.
In Chapter 3, we will give a survey of related work: related methods for general subsystem construction, protocol conversions, and controller design for discrete event systems.

In Chapter 4, an algorithm for diagnosis of testing preorder is presented. This algorithm will be used in the following chapters.

In Chapter 5, two important concepts, controllability and observability, are introduced in the context of labelled transition systems. These concepts are directly adapted from the existing theory of controller design for discrete event systems where they are defined in the context of automata. A general algorithm is presented for solving the subsystem construction problem.

In Chapter 6, the algorithm developed in Chapter 5 is used for protocol conversion. Since the algorithm developed in Chapter 5 is so general that the constructed system may not be optimized, the optimization problem for protocol conversion is also considered.

In Chapter 7, we extend the algorithm presented in Chapter 5 by using a dense time finite state model and timed design constraints.

Finally in Chapter 8, we give our conclusions and discuss future work.