A formal method for synthesizing optimized protocol converters and its application to mobile data networks *

Zhongping Tao [a,**], Gregor v. Bochmann [b] and Rachida Dssouli [b]

[a] *Nortel Technology, P.O. Box 3511, Station C, MS 160, Ottawa, ON, Canada K1Y 4H7*
[b] *Université de Montréal, Département d'Informatique et de Recherche Operationnelle, C.P. 6128, Succ. A, Montréal, PQ, Canada H3C 3J7*

As mobile information networks are expanding rapidly, we expect to integrate voice, paging, electronic mail and other wireless information services. Interworking units that perform protocol conversion at the boundaries of different networks will play an important role. In this paper, we propose an efficient algorithm for constructing optimized protocol converters to achieve interoperability between heterogeneous data networks. This algorithm first derives constraints from two given protocols, and apply the constraints to channel specifications, thus removing message sequences that do not contribute to system progress. Then, an optimized converter is generated from a given service specification, the two protocol specifications and the modified channel specifications. A reduction relation is used to compare the service specification and the constructed internetworking system in order to deal with the problem of nondeterministic services. Compared with related works, our method has two advantages: (1) it generates an optimized converter; (2) it can be applied to the case that the service specification is nondeterministic. The application of the method to mobile networks is given by an example.

1. Introduction

With the expanding of mobile information networks, new protocols are proposed to address the requirement of mobile environment. One of the difficulties that arise in interconnecting different mobile and fixed data networks is the problem of protocol mismatch [8] – incompatible protocols are used in heterogeneous networks. For example, if a TCP connection is setup where one endpoint is mobile, the efficiency of the TCP connection is very poor because of the higher rate of lost status and data messages caused by the fading environment over a wireless link [5]. To solve this problem, a modified version of the transport layer protocol should be implemented in mobile host and protocol conversion is necessary at a base station.

The problem of protocol conversion can be explained informally as follows. Consider two different protocols $A = (A_1, A_2)$ and $B = (B_1, B_2)$ (figure 1). Suppose the two protocols provide similar services, but differ in certain details, and we want A_1 to communicate with B_2 through a protocol converter C. The converter C receives messages from one protocol, interprets them, and delivers appropriate messages to the other protocol in a well-defined order such that the semantics of the messages does not change. The protocols and the converter together form an internetworking system and provide the required services specified by Sc as shown in figure 2(a). An interconnection between A_2 and B_1 can be defined similarly.

Protocol conversion is a complex problem since multiple protocols are considered. It is difficult to design a correct protocol converter by informal, heuristic methods. A formal approach is a reasonable choice in this area, which may minimize design errors and simplify design procedures. Several formal methods have been proposed for protocol conversion in the last several years, which can also be used to address the protocol mismatch problem in mobile networks. These methods can roughly be classified into the following two classes: the bottom-up method and the top-down method.

A bottom-up method begins with analyzing heuristically the low level functions of the protocols in order to find out a design constraint, for example, a message mapping relation between protocols. The constraint is used to construct a

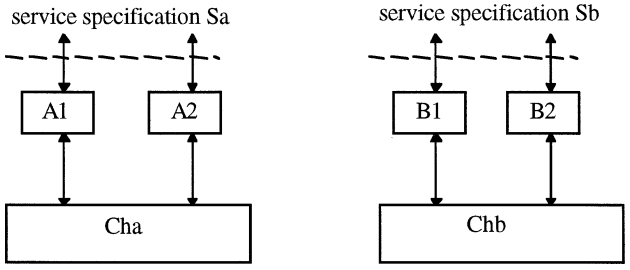


Figure 1. Two protocols A and B.

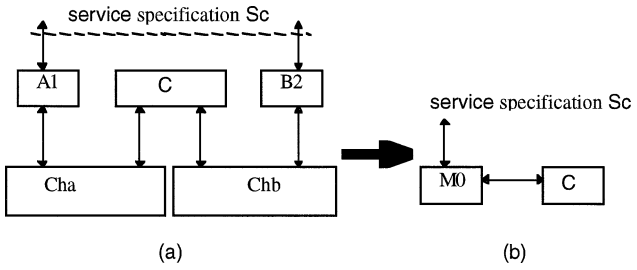


Figure 2.

* This work is supported by the IDACOM-NSERC-CWARC Industrial Research Chair on Communication Protocols. Previous version of this paper appeared in IC3N '95, Las Vegas, USA, 1995.

** This work was completed while the author was in Université de Montréal.

converter from the Cartesian cross product of A_2 and B_1 [16,18–20,22]. The common limitations of the bottom-up method are:

(1) A message mapping relation is required which can only be obtained heuristically.

(2) It is difficult to validate the correctness of a given message mapping relation.

(3) No service requirement of the interworking system is explicitly used, therefore, the generated converter needs to be verified against a given service specification.

A top-down method explicitly uses a service specification of the interworking system as the semantic constraint. The main methods are outlined below. In [3], the concept of a service adapter is proposed for the concatenation of communication services provided by two different protocols. A service adapter receives a service primitive from the service interface of one protocol, interprets it and sends it to the service interface of the other protocol. The automatic construction of a protocol converter from two given protocol specifications and a service adapter is described in [2]. In [17], a two-stage approach is developed to derive protocol converter. In the first stage, a service adapter from the service specifications of the two protocols is constructed by using the method proposed in [14]. In the second phase, a protocol converter is constructed by directly composing the service adapter and the underlying protocol specifications. Okumura discussed under what conditions the constructed system will inherit the properties from the given protocols. It is possible that the converter constructed in this way may contain states and transitions that are never executed. An efficient algorithm is presented in [9] to remove these states and transitions. The basic idea is to remove from the underlying protocol entities composed with the service adapter those service primitives (and related states) that are unmatched with the service adapter, and those that can be reached only from the unmatched service primitives; then the algorithm constructs the strongly connected components starting with the initial state, and discards the rest of the machine. The disadvantage of the method discussed above is that there may not exist a service adapter for two given protocols even if a protocol converter does exist. Therefore, the application of the method is limited.

Calvert and Lam proposed a top-down method [6,11], which uses a safety property and a progress property to guarantee the correctness of a converter. The algorithm is divided into two phases. In the first phase, a set of states and transitions is constructed inductively by searching the giving protocol entities and the service specification under the safety constraint. The result is a specification with the maximum trace set satisfying the safety property. In the second phase, the states and transitions in the specification that violate the progress property are iteratively removed. If the final specification is not empty after the algorithm terminates, then the converter is obtained. The advantage of this method is that it does not have the limitation of the methods proposed in [9,17], i.e., the algorithm will generate a protocol converter if it exists. However, there are also some limitations:

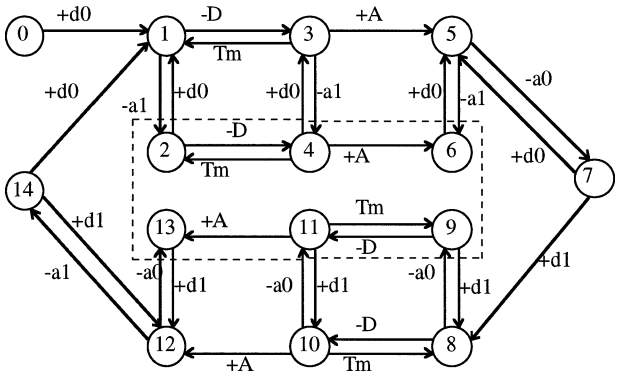


Figure 3. An unoptimized converter.

(1) The converter may contain superfluous states and transitions that do not contribute to the system progress, and may be harmful for system performance. An example given in [6,11] is shown in figure 3, where the states in the dotted box are superfluous. The converter can send back an unnecessary acknowledgment to a protocol even after receiving a data message correctly. If these states and transitions are not removed, the system performance will certainly be worse than optimal. Calvert suggested that these states and transitions should be better removed by hand. However, it is not clear how to do so in general.

(2) The service specification Sc must be deterministic.

In summary, the protocol conversion methods reported in the literature are far from satisfactory due to the limitations discussed above. Nevertheless, we favor top-down method because of the following reasons:

(1) The service specification is used explicitly; it is not necessary to validate the system against its service specification after the converter is generated.

(2) It does not require the establishment of a message mapping relation between protocols in advance, the most difficult part of the bottom-up method.

In this paper, we propose an approach to overcome some of the limitations of the top-down method. We observed that the channel specifications may contain message sequences not contributing to system progress. The top-down method proposed in [6,11] does not remove these unnecessary behaviours when a converter is derived from the channel specifications and protocol specifications. Our approach first derives constraints from the given protocols and applies them to the channel specifications, thus removing the message sequences of the channel specifications that do not contribute to system progress. Then an optimized converter is generated from a given service specification, the two protocol specifications and the modified channel specifications. Since the unnecessary states and transitions

are removed from the first step, this approach may reduce computation. We use the reduction relation [4,12] to compare the constructed system and the service specification. This allows the treatment of nondeterministic service specifications. Compared with related works, our method has two advantages:

(1) It generates an optimized converter.

(2) It can be applied to nondeterministic service specifications.

The paper is organized as follows. Section 2 will introduce preliminary definitions and formalize the protocol conversion problem. In section 3, some theoretical aspects of our approach will be presented. In section 4, an algorithm for protocol conversion will be developed. An example is given to show the application of the algorithm to mobile network.

2. Definition of the problem

2.1. General definitions

A Finite Labeled Transition System (FLTS) is chosen in this paper to model communication protocols, services and channel specifications.

Definition 1 (FLTS) [7]. A nondeterministic FLTS M is a four-tuple $M = (Q, \Sigma, \delta, q_0)$, where:

- Q is a finite set of states.
- Σ is a set of observable events.
- δ is a transition relation, $\delta : Q \times (\Sigma \cup \{\tau\}) \to 2^Q$.
- q_0 is the initial state.

Intuitively, Q represents the set of possible states of system M; Σ contains the events of which computations may consist, and δ defines how the state of an FLTS changes according to its current state and its interaction with the environment through the execution of events. For states $p, p' \in Q$, and event $e \in \Sigma$, a transition $p - e \to p' \in \delta$ means that event e is enabled when the current state is p. If e is also enabled in the environment, then it may be executed, causing the state instantaneously to become p'. An *internal event*, represented by τ, is distinct from all events in Σ. Any transition $p - \tau \to p' \in \delta$ is called an *internal transition*. Internal transitions occur without the participation (through interaction) of the environment. The presence of internal events introduces nondeterministic behaviour, which means that the state after a given sequence of internal events may not be uniquely determined. Nondeterminism may also be introduced by observable events: for $p - e \to p' \in \delta$ and $p - e \to q' \in \delta$, if $q' \neq p'$, then after executing event e at state p, the next system state can not be determined for the executed event e. Each event is considered atomic, that is, no other event can overlap with an atomic event during the time interval from the initiation to the termination of the event. Some useful notations are introduced below. Note that $\Sigma_o \subseteq \Sigma$ is the set of observable events, which interact with a given system environment modeled by another FLTS.

Σ_o	A set of observable events for a given system environment.
$q - e \to$	$\exists q'$, such that $q - e \to q'$.
$q - e \not\to$	There is not a state q' such that $q - e \to q'$.
$q - \tau^k \to q'$	State q' can be reached from state q by executing a sequence of k internal events.
$q - \sigma \to q_n$	State q_n can be reached from state q by executing a sequence of events $\sigma = e_1 e_2 \dots e_n$ ($e_i \in \Sigma \cup \{\tau\}$).
$q - \sigma \to$	There exist a state q_n such that $q - \sigma \to q_n$.
$E(M)$	The set of all execution sequences of M, e.g., $E(M) = \{\sigma \mid q_0 - \sigma \to\}$.
$q = \Rightarrow_{\Sigma_o} q'$	There is an execution sequence $\sigma \in (\Sigma - \Sigma_o \cup \{\tau\})^*$ such that $q - \sigma \to q'$, where $\Sigma_o \subseteq \Sigma$.
$q = e \Rightarrow_{\Sigma_o} q'$	There are two execution sequences $\sigma_1, \sigma_2 \in (\Sigma - \Sigma_o \cup \{\tau\})^*$ and two states q_1 and q_2 such that $q - \sigma_1 \to q_1$, $q_1 - e \to q_2$ and $q_2 - \sigma_2 \to q'$, in short $q_1 - \sigma_1 e \sigma_2 \to q'$.
$q = t \Rightarrow_{\Sigma_o} q'$	For a sequence of events $t = e_1 \dots e_n$, where $e_i \in \Sigma_o$, $\exists \sigma_0, \dots, \sigma_n \in (\Sigma - \Sigma_o \cup \{\tau\})^*$ such that $q - \sigma_0 e_1 \sigma_1 e_2 \dots e_n \sigma_n \to q'$. t is called a trace.
$q = t \not\Rightarrow_{\Sigma_o}$	There is not a state q' such that $q = t \Rightarrow_{\Sigma_o} q'$.
$q = t \Rightarrow_{\Sigma_o}$	$\exists q'$, such that $q = t \Rightarrow_{\Sigma_o} q'$.
$\mathrm{Tr}_{\Sigma_o}(M)$	The set of traces of an FLTS M, that is, $\mathrm{Tr}_{\Sigma_o}(M) = \{t \mid q_0 = t \Rightarrow_{\Sigma_o}\}$ for $\Sigma_o \subseteq \Sigma$.
$p(S)$	For a given set S, $p(S)$ denotes a power set of S, i.e., set of subsets of S.
q after t	q is a state after trace t, i.e., $q_0 = t \Rightarrow_{\Sigma_o} q$.
$\mathrm{Ref}_{\Sigma_o}(M, q)$	The refusal set of an FLTS M at state q for Σ_o, i.e., $\mathrm{Ref}_{\Sigma_o}(M, q) = \{e \mid q = e \not\Rightarrow_{\Sigma_o}$ and $e \in \Sigma_o\}$.

For two interacting FLTSes, we say that the events that are executed jointly are *directly coupled events*. The interaction between FLTSes can be specified by assigning the same name to events that are directly coupled. To avoid confusion, we assume that all uncoupled events in the interacting FLTSes have a unique name. It is reasonable to make this assumption since we can rename uncoupled events to make them unique among the set of events executed by each FLTS. Directly coupled events *may* be invisible to the environment of the system, hence we should model them by internal events for convenience. For system analysis, we define the *composition* of two FLTSes into a single FLTS as follows.

Definition 2 (Coupled product). A *coupled product* $M_1 \,||\, M_2$ of two FLTSes $M_1 = (Q_1, \Sigma_1, \delta_1, p_0)$ and $M_2 = (Q_2,$

$\Sigma_2, \delta_2, q_0)$ is an FLTS $M = (Q, \Sigma, \delta p, (p_0, q_0))$ such that:

- Q is a subset of $Q_1 \times Q_2$; each element is of the form (p, q), where $p \in Q_1$, $q \in Q_2$;
- $\Sigma = (\Sigma_1 \cup \Sigma_2) - (\Sigma_1 \cap \Sigma_2)$;
- $(p_0, q_0) \in Q$ is the initial state;
- δp is the transition relation defined on Q such that for $p, p' \in Q_1$, $q, q' \in Q_2$:
  (1) $(p, q) - e_1 \rightarrow (p', q)$ if $p - e_1 \rightarrow p'$ and $e_1 \in (\Sigma_1 - \Sigma_2) \cup \{\tau\}$;
  (2) $(p, q) - e_1 \rightarrow (p, q')$ if $q - e_2 \rightarrow q'$ and $e_2 \in (\Sigma_2 - \Sigma_1) \cup \{\tau\}$;
  (3) $(p, q) - \tau \rightarrow (p', q')$ if $p - e \rightarrow p'$, $q - e \rightarrow q'$ and $e \in \Sigma_1 \cap \Sigma_2$;
  (4) for other cases, no transition is defined.

In some cases, we need to compose two FLTSes that do not directly interact with each other. This can be done by computing the so-called *Cartesian cross product*, written $M_1 \times M_2$, which is identical to the coupled product $M_1 || M_2$ in the case that $\Sigma_1 \cap \Sigma_2 = \varnothing$.

Definition 3 (# product). A *# product* $M_1 \# M_2$ of two FLTSes $M_1 = (Q_1, \Sigma_1, \delta_1, p_0)$ and $M_2 = (Q_2, \Sigma_2, \delta_2, q_0)$ is an FLTS $M = (Q, \Sigma, \delta p, (p_0, q_0))$ where:

- Q is a subset of $Q_1 \times Q_2$;
- $\Sigma = \Sigma_1 \cup \Sigma_2$ is the set of events;
- (p_0, q_0) is the initial state;
- δp is the transition relation defined on Q such that for $p, p' \in Q_1$ and $q, q' \in Q_2$:
  (1) $(p, q) - e_1 \rightarrow (p', q)$ if $p - e_1 \rightarrow p'$ and $e_1 \in (\Sigma_1 - \Sigma_2) \cup \{\tau\}$;
  (2) $(p, q) - e_2 \rightarrow (p, q')$ if $q - e_2 \rightarrow q'$ and $e_2 \in (\Sigma_2 - \Sigma_1) \cup \{\tau\}$;
  (3) $(p, q) - e \rightarrow (p', q')$ if $p - e \rightarrow p'$ and $q - e \rightarrow q'$ with $e \in \Sigma_1 \cap \Sigma_2$;
  (4) for other cases, no transition is defined.

Compared with the coupled product, the only difference is that each directly coupled event $e \in \Sigma_1 \cap \Sigma_2$ in # product is still observable.

Definition 4 (Strong bisimulation relation) [15]. A binary relation ξ on states is a strong bisimulation if for each $\langle p, q \rangle \in \xi$ and each event $e \in \Sigma \cup \{\tau\}$, the following two conditions are true:

(1) whenever $p - e \rightarrow p'$ then there is a state q' such that $q - e \rightarrow q'$ and $(p', q') \in \xi$;

(2) whenever $q - e \rightarrow q'$ then there is a state p' such that $p - e \rightarrow p'$ and $(p', q') \in \xi$.

We write $p \cong q$ if $(p, q) \in \xi$.

Intuitively, a bisimulation can be thought of as a matching between states that have the property that if two states are matched then each execution sequence starting from one state must be matched by the execution sequence starting from the other state.

Definition 5 (Strong bisimulation equivalence of two FLTSes) [15]. Given two FLTSes $M_1 = (Q_1, \Sigma_1, \delta_1, p_0)$ and $M_2 = (Q_2, \Sigma_2, \delta_2, q_0)$, M_1 and M_2 are strong bisimulation equivalent if there is a strong bisimulation relation ξ which contains $\langle p_0, q_0 \rangle$, written $M_1 \cong M_2$.

Definition 6 (Reduction relation). Given two FLTSes $M_1 = (Q_1, \Sigma_1, \delta_1, p_0)$ and $M_2 = (Q_2, \Sigma_2, \delta_2, q_0)$, M_1 and M_2 satisfy reduction relation for $\Sigma_o \subseteq \Sigma_1 \cup \Sigma_2$, written $M_1 \angle_{\Sigma_o} M_2$, if the following conditions are satisfied:

(1) $\mathrm{Tr}_{\Sigma_o}(M_1) \subseteq \mathrm{Tr}_{\Sigma_o}(M_2)$.

(2) For any $t \subseteq \mathrm{Tr}_{\Sigma_o}(M_1) \cap \mathrm{Tr}_{\Sigma_o}(M_2)$ and any q after t in M_1, there is a state p after t in M_2 such that $\mathrm{Ref}_{\Sigma_o}(M_1, q) \subseteq \mathrm{Ref}_{\Sigma_o}(M_2, p)$.

This definition is similar to the reduction relation defined in [4,12]. The difference is that we define the traces and rejected events by using Σ_o, instead of Σ_1 and Σ_2. This definition can be explained by two concepts: the first is the *safety property* – the set of all traces of M_1 is limited to the set of traces of M_2; the second is the *progress property*: placed in any environment whose interface with M_1 or M_2 is defined by Σ_o, an event after a trace that is rejected by M_1 must also be rejected by M_2. The progress property implies that M_1 can not deadlock when M_2 can not deadlock. $M_1 \angle_{\Sigma_o} M_2$ iff M_1 and M_2 satisfy the safety property and the progress property.

Definition 7 (Submachine). An FLTS $M' = (Q', \Sigma', \delta', q_0')$ is a submachine of another FLTS $M = (Q, \Sigma, \delta, q_0)$ if (a) $Q' \subseteq Q$, (b) $\Sigma' \subseteq \Sigma$, (c) $\delta' \subseteq \delta$, and (d) $q_0' = q_0$.

2.2. Formal definition of protocol conversion

The protocol conversion problem informally explained in the Introduction can be formally defined by the following expression:

$$A_1 \,||\, \mathrm{Cha} \,||\, C \,||\, \mathrm{Chb} \,||\, B_2 \angle_{\Sigma_S} \mathrm{Sc}, \tag{1}$$

where Cha and Chb denote the channels between the protocol entities as shown in figure 1, and Σ_S is the set of observable events of Sc. Since A_1, Cha, Chb and B_2 are given, this expression can be represented by $M_0 || C \angle_{\Sigma_S} \mathrm{Sc}$, where $M_0 = (A_1 || \mathrm{Cha}) \times (\mathrm{Chb} || B_2)$ as shown in figure 2(b).

In addition, the converter should satisfy the following requirements:

(1) The converter C should have no unnecessary transitions and states.

(2) The converter C should work in such a way that A_1 communicates with C as if A_1 communicates with A_2, and B_2 communicates with C as if B_2 communicates with B_1. Therefore, the following condition should be satisfied at the service interface:

$$\text{Sc} \angle_{\Sigma_{a_1}} A_1 \,||\, \text{Cha} \,||\, A_2 \quad \text{and} \quad \text{Sc} \angle_{\Sigma_{b_2}} B_1 \,||\, \text{Chb} \,||\, B_2. \quad (2)$$

Definition 8 (Maximum solution). Given $M_0 = (Q, \Sigma_0, \delta, p_0)$ and $\text{Sc} = (Q_S, \Sigma_S, \delta_S, q_0)$, a converter C is a maximum solution if for any other converter C' satisfying $M_0 || C' \angle_{\Sigma_S}$ Sc, we have $\text{Tr}_{\Sigma_o}(C') \subseteq \text{Tr}_{\Sigma_o}(C)$, where $\Sigma_o = \Sigma_0 - \Sigma_S$ is the set of events of C and C'.

A maximum solution is not exactly what we want, since it may contain superfluous states and transitions. Protocol conversion often deals with well-designed protocols that have already been used in existing networks. It is reasonable to assume that these protocols have no superfluous states and transitions, when considered alone.

To construct an optimized deterministic protocol converter, our basic argument is that if the converter C does not do more than A_2 and B_1 can do, i.e., $\text{Tr}_{\Sigma_{a_2}}(C) \subseteq \text{Tr}_{\Sigma_o}(A_2)$ and $\text{Tr}_{\Sigma_{b_1}}(C) \subseteq \text{Tr}_{\Sigma_o}(B_1)$, where Σ_{a_2} is the set of events of A_2 and Σ_{b_1} is the set of events of B_1, then C will have no transitions that do not contribute to system progress. According to the discussion above, we formalize the concept of optimized converters as follows.

Definition 9 (Optimized converter). A deterministic converter C is optimized if C has maximum sequences (in the sense of a maximum solution defined above) under the following condition: $\text{Tr}_{\Sigma_{a_2}}(C) \subseteq \text{Tr}_{\Sigma_o}(A_2)$ and $\text{Tr}_{\Sigma_{b_1}}(C) \subseteq \text{Tr}_{\Sigma_o}(B_1)$.

3. Foundations of our method

3.1. Refusal graph

To deal with nondeterministic service specifications, we will introduce the concept of *refusal graph*, shortly Rgraph. The following definition is used when we define an Rgraph for a given FLTS.

Definition 10 (After set). Given an FLTS $M = (Q, \Sigma, \delta, q_0)$ and $\Sigma_o \subseteq \Sigma$, we define the *after set* of a state $p \in Q$ as $A_{\Sigma_o}(p) = \{p' \mid p \Longrightarrow_{\Sigma_o} p'\}$.

The *After* set $A_{\Sigma_o}(p)$ intuitively describes all the reachable states from a state p by executing zero, one or more events $e \in \Sigma - \Sigma_o \cup \{\tau\}$.

Definition 11 (Refusal graph). An Rgraph is a 5-tuple $G_{\Sigma_o} = (S, \Sigma_o, \delta, R, s_0)$, where:

- S is a finite set of states.
- Σ_o is a set of events.
- $\delta : S \times \Sigma_o \to S$ is a transition relation.
- $R : S \to p(p(\Sigma_o))$ is a mapping from a state $s \in S$ to a set of subsets of Σ_o.
- $s_0 \in S$ is the initial state.

This definition is similar to the definition of an FLTS. However, there are two differences: first, from the definition of the transition relation $\delta : S \times \Sigma_o \to S$, an Rgraph is deterministic; second, there is a set of subsets of Σ_o associated with each state s in S, written $R(s)$.

Definition 12 (Correspondence between an FLTS and an Rgraph). Given an FLTS $M = (Q, \Sigma, \delta, q_0)$ and $\Sigma_o \subseteq \Sigma$, we say that $G_{\Sigma_o}(M) = (S, \Sigma_o, \delta', R, s_0)$ is the corresponding Rgraph of M iff:

(1) $S = \{s_i \mid s_i = \{q \in Q \mid q_0 = t \Rightarrow_{\Sigma_o} q\}, t \in \text{Tr}_{\Sigma_o}(M)\}$.

(2) δ' is the transition relation: $\forall s_i$, $s_j \in S$ and $\forall e \in \Sigma_o$, we have $s_j - e \to s_i$ iff $s_i = \bigcup_{p' \in \Psi} A_{\Sigma_o}(p')$, where

$$\Psi = \{p' \mid \exists p \in s_j \text{ and } p - e \to p'\}.$$

(3) $\forall s_i \in S$, $R(s_i) = \{\text{Ref}_{\Sigma_o}(M, p) \mid p \in s_i\}$.

(4) $s_0 = A_{\Sigma_o}(q_0)$.

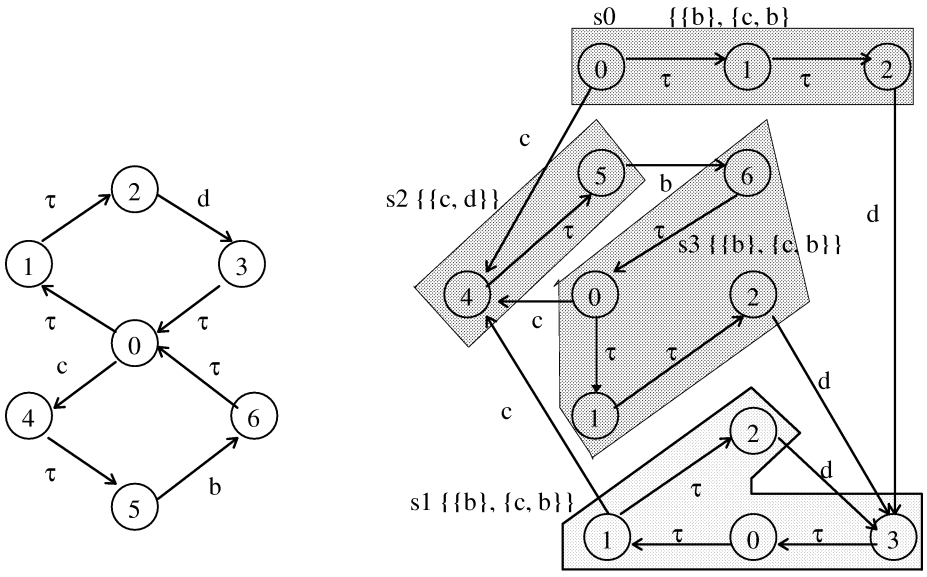
In this definition, a set of states in M is considered as one state in $G_{\Sigma_o}(M)$. This is similar to the method given in [13] for transforming a nondeterministic finite state machine to a trace equivalent deterministic finite state machine, except that the refusal set is ignored in [13]. By ignoring the set of refusal sets for each state in $G_{\Sigma_o}(M)$ we get a deterministic FLTS, denoted as $P_{\Sigma_o}(M)$.

Example 1. For the given FLTS M specified by figure 4(a), where $\Sigma_o = \{c, d, b\}$, the obtained refusal graph is shown in figure 4(b). In figure 4(b), we have the shadowed boxes: $s_0 = A_{\Sigma_o}(0)$, $s_1 = A_{\Sigma_o}(3)$, $s_2 = A_{\Sigma_o}(4)$, $s_3 = A_{\Sigma_o}(6)$. The refusal sets are shown beside each state of the Rgraph.

3.2. Existence of protocol converter

Our goal is to construct a deterministic converter C such that $M_0 \,||\, C \angle_{\Sigma_o}$ Sc. This can be implemented in two steps: first, we compute $M_0 \,\#\, G_{\Sigma_S}(\text{Sc})$ to obtain all execution sequences that satisfy the safety property because of $\text{Tr}_{\Sigma_S}(M_0 \# G_{\Sigma_S}(\text{Sc})) \subseteq \text{Tr}_{\Sigma_S}(\text{Sc})$. Second, since some behaviours in $M_0 \# G_{\Sigma_S}(\text{Sc})$ may not satisfy the progress property, we need to prevent them from happening for avoiding deadlocks. To do so, we first define the concept of a machine with grouped states.

Definition 13 (Machine with grouped states (MGS)). Given an FLTS $M = (Q, \Sigma, \delta, q_0)$, we define a corresponding machine with grouped states, written $\text{MGS}_{\Sigma_o}(M) = (\mathbf{S}, \Sigma, \delta', s_{0_{q_0}})$, where:



(a) (b)

Figure 4. (a) An FLTS M. (b) The Rgraph of M.

(1) $\mathbf{S} = \bigcup \mathbf{s}_i$, where $\mathbf{s}_i \cap \mathbf{s}_j = \varnothing$ for $i \neq j$; each $\mathbf{s}_i \in \mathbf{S}$ represents a group of states $\mathbf{s}_i = \{s_{i_q} \mid q \in s_i\}$, where s_i is defined in the corresponding Rgraph $G_{\Sigma_o}(M)$.

(2) For any $\mathbf{s}_i, \mathbf{s}_j \in \mathbf{S}$, and any $s_{i_q} \in \mathbf{s}_i$, $s_{j_p} \in \mathbf{s}_j$, we have $s_{i_q} - e \rightarrow s_{j_p}$ in $\text{MGS}_{\Sigma_o}(M)$ iff $q - e \rightarrow q$ in M for $e \in \Sigma \cup \{\tau\}$.

$\text{MGS}_{\Sigma_o}(M)$ is different from $G_{\Sigma_o}(M)$ in four aspects: first, the state space is different; second, the set of events is Σ, instead of Σ_o; third, the transition relation in $\text{MGS}_{\Sigma_o}(M)$ is defined by using the states in $\mathbf{S}$; fourth, there are no refusal set explicitly associated with the states in $\text{MGS}_{\Sigma_o}(M)$. An important relationship between $\text{MGS}_{\Sigma_o}(M)$ and M is stated by the following lemma.

Lemma 1. For any FLTS $M = (Q, \Sigma, \delta, p)$ and any $\Sigma_o \subseteq \Sigma$, $\text{MGS}_{\Sigma_o}(M) \cong M$.

To implement the two-step construction, we face a fundamental problem: what behaviour in M_0 can be executed through the interaction with a deterministic converter C? To answer this question, we have the following important result.

Lemma 2. Given $M_0 = (Q_0, \Sigma_0, \delta_0, p_0)$, $\text{Sc} = (Q_S, \Sigma_S, \delta_S, q_{0_S})$ and $\Sigma_o = \Sigma_0 - \Sigma_S$, let H be a submachine of $\text{MGS}_{\Sigma_o}(M_0 \,\#\, G_{\Sigma_S}(\text{Sc}))$, $H \cong M_0 \,\#\, P_{\Sigma_o}(H)$ iff H satisfies the following two conditions:

(1) For each state $s_{i_{(p,s)}} \in H$, where p is a state of M_0 and s is a state of $G_{\Sigma_S}(\text{Sc})$, and $\forall e \in \Sigma_S \cup \{\tau\}$, $p - e \rightarrow p'$ in M_0 implies there must be a state $s_{i_{(p',s')}}$ in H such that $s_{i_{(p,s)}} - e \rightarrow s_{i_{(p',s')}}$.

(2) For each $s_{i_{(p,s)}}$ and state p in M_0 such that $(p - e \rightarrow) \wedge (s_{i_{(p,s)}} - e \nrightarrow)$ then for any state $s_{i_{(p',s')}}$, we have $s_{i_{(p',s')}} - e \nrightarrow$.

The first condition implies that if an event in $\Sigma_S \cup \{\tau\}$ occurs following an execution sequence in $E(H)$, then the extended execution sequence must remains in $E(H)$, provided that the extended execution sequence is in $E(M_0)$. The second condition means that for a given $s_{i_{(p,s)}}$, if an event $e \in \Sigma_o$ is enabled in M_0 at state p, but is disabled at state $s_{i_{(p,s)}}$ in H, then all states in $\mathbf{s}_i$ should disable event e. The result of this lemma shows that the interaction between M_0 and $P_{\Sigma_o}(H)$ behaves exactly like what H does if H satisfies the two conditions. We call this property of H *well-behaved* with respect to M_0, Σ_o and Sc.

Theorem 1. Given $M_0 = (Q_0, \Sigma_0, \delta_0, p_0)$, $\text{Sc} = (Q_S, \Sigma_S, \delta_S, q_{0_S})$ and $\Sigma_o = \Sigma_0 - \Sigma_S$, we have:

(1) If $C = (Q_c, \Sigma_o, \delta_c, q_{0_c})$ is a deterministic solution such that $M_0 \,||\, C \angle_{\Sigma_S} \text{Sc}$ then there exists a submachine H of $\text{MGS}_{\Sigma_o}(M_0 \# G_{\Sigma_S}(\text{Sc}))$ having the well-behaved property with respect to M_0 and Sc and satisfying $H \angle_{\Sigma_S} \text{Sc}$.

(2) If H is a submachine of $\text{MGS}_{\Sigma_o}(M_0 \,\#\, G_{\Sigma_S}(\text{Sc}))$ having the well-behaved property with respect to M_0, Σ_o and Sc and satisfying $H \angle_{\Sigma_S} \text{Sc}$ then $P_{\Sigma_o}(H)$ is a solution such that $M_0 \,\#\, P_{\Sigma_o}(H) \angle_{\Sigma_S} \text{Sc}$.

Theorem 1 implies that we can construct a protocol converter by finding a submachine H of $\text{MGS}_{\Sigma_o}(M_0 \# G_{\Sigma_S}(\text{Sc}))$, which has the well-behaved property with respect to M_0, Σ_o and Sc and satisfies $H \angle_{\Sigma_S} \text{Sc}$.

3.3. Optimization

As we have discussed in section 2.2, the maximum solution satisfying the condition in theorem 1 is not exactly what we want, since it may contain unnecessary states and transitions. To solve this problem, we have the following two basic observations:

(1) Some superfluous transitions and states in the protocol converter are due to the property of the channel's behaviour: the channel is able to transmit any messages (including the unnecessary ones). So the unnecessary message sequences are also included in M_0. However, the existing algorithms have not used any effective measure to remove them [3,11,14].

(2) Some superfluous transitions and states may be due to superfluous transitions and states in the given protocols. We will not deal with the problems due to unoptimized protocols. Therefore, we make the assumption that there are no superfluous transitions and states in the given protocols.

Based on the observations above, the following theorem describes the basic idea of how to generate an *optimized* protocol converter.

Theorem 2. Given two protocols $A = (A_1, A_2)$ and $B = (B_1, B_2)$, a global service specification Sc, and channel specifications Cha and Chb, let $\text{Cha}' = \text{Cha} \,\#\, P_{\Sigma_{a_2}-\Sigma_S}(A_2)$, $\text{Chb}' = \text{Chb} \,\#\, P_{\Sigma_{b_1}-\Sigma_S}(B_1)$, $M = (A_1 \,||\, \text{Cha}) \times (\text{Chb} \,||\, B_2)$ and $M_0 = (A_1 \,||\, \text{Cha}') \times (\text{Chb}' \,||\, B_2)$, if there exists a deterministic converter C' such that $M \,||\, C' \angle_{\Sigma_S}$ Sc, then:

(1) There is a maximum solution C such that $M_0 || C \angle_{\Sigma_S}$ Sc.

(2) C is an optimized converter for the given protocols A, B and service specification Sc.

This theorem implies that an optimized converter can be obtained by first obtaining constraints from the given protocol entities, and removing those message sequences by using the constraints from the given channel specifications that may result in superfluous states and transitions in the converter, then the modified channel specifications are used for constructing an optimized converter.

4. Algorithm for protocol conversion

4.1. The algorithm

Based on theorems 1 and 2, it is easy to develop an algorithm for protocol conversion. The following algorithm is divided into five steps. In step 1, the constraints for optimization are derived from the given protocol entities according to theorem 2, and are applied to the channel specifications. M_0 is obtained by composing the protocol specifications and the modified channel specifications. In the second step, the execution sequences in M_0 that violate the safety property are removed by computing $M_0' = M_0 \,\#\, G_{\Sigma_S}(\text{Sc})$. The states and transitions that do not satisfy the first condition of lemma 2 are deleted by marking them Bad States (BD). In the third step, we construct a submachine H' of $\text{MGS}_{\Sigma_o}(M_0')$ such that the states and transitions that violate the second condition of lemma 2 are removed. In step 4, we construct a submachine H of H' such that the states and transitions that do not satisfy theorem 1 are marked out, and the converter is obtained in step 5 by computing $C = P_{\Sigma_o}(H)$.

Algorithm Conversion

/* Input: the protocols $A = (A_1, A_2)$ and $B = (B_1, B_2)$, the channel Cha of protocol A, the channel Chb of protocol B, service specification $\text{Sc} = (Q_S, \Sigma_S, \delta_S, q_{0_S})$, and $\Sigma_o = \Sigma_0 - \Sigma_S$.
/* Output: an optimized protocol converter C.
Begin

Step 1:

(1) Derive the constraints: $A_2' = P_{\Sigma_{a_2}-\Sigma_S}(A_2)$ and $B_1' = P_{\Sigma_{b_1}-\Sigma_S}(B_1)$.

(2) Imposing the constraints to the channel specifications: $\text{Cha}' = \text{Cha} \,\#\, A_2'$ and $\text{Chb}' = \text{Chb} \,\#\, B_1'$.

(3) Construct $M_0 = (A_1 \,||\, \text{Cha}') \times (\text{Chb}' \,||\, B_2)$.

Step 2:

Compute $M_0' = M_0 \,\#\, G_{\Sigma_S}(\text{Sc})$ and mark any state (p, s) of M_0' BS ("Bad State") if there is a state p' in M_0 such that $p - e \to p'$ for $e \in \Sigma_S \cup \{\tau\}$, but there is not a state s' in $G_{\Sigma_S}(\text{Sc})$ such that $s - e \to s'$. The result is denoted as $M_0' = (Q_p, \Sigma_p, \delta_p, (p_0, s_0))$, where s_0 is the initial state of $G_{\Sigma_S}(\text{Sc})$ and p_0 is the initial state of M_0.

Step 3:

Create $\mathbf{s}_0 = \{s_{0_{(p,s_k')}} \mid (p, s_k') \in A_{\Sigma_o}((p_0, s_0'))\}$ and mark it TP ("To be Processed").
For $\forall e \in \Sigma_S \cup \{\tau\}$, $\forall s_{0_{(p,s_k')}} \in \mathbf{s}_0$ and $\forall s_{0_{(p',s_m')}} \in \mathbf{s}_0$, create a transition labelled e from $s_{0_{(p,s_k')}}$ to $s_{0_{(p',s_m')}}$ whenever $(p, s_k') - e \to (p', s_m')$ exists in M_0'.
Do the following while there is an $\mathbf{s}_i$ marked TP:

(1) If there is a state $s_{i_{(p,s_k')}} \in \mathbf{s}_i$ marked BS then mark $\mathbf{s}_i$ BS; otherwise for $\forall e \in \Sigma_o$ do the following:

(a) Compute

$$\mathbf{s}_i(e) = \bigcup_{s_{i_{(p,s_k')}} \in \mathbf{s}_i} \{A_{\Sigma_o}((p', s_m')) \mid (p, s_k') - e \to (p', s_m') \in \delta_p\}.$$

(b) If there is a state $(p'', s_h') \in \mathbf{s}_i(e)$ marked BS then remove all transitions labelled e from any state $s_{i_{(p,s_k')}} \in \mathbf{s}_i$; otherwise do the following:

(i) if $\mathbf{s}_i(e)$ is not empty and there is no previously created $\mathbf{s}_j$ containing exactly all the state pairs in $\mathbf{s}_i(e)$, do the following:

- Create such an $\mathbf{s}_j$ containing all the state pairs in $\mathbf{s}_i(e)$, i.e., $\mathbf{s}_j = \{s_{j_{(p,s'_k)}} \mid (p, s'_k) \in \mathbf{s}_i(e)\}$.
- For $\forall e' \in \Sigma_S \cup \{\tau\}$, $\forall s_{j_{(p,s'_k)}} \in \mathbf{s}_j$ and $\forall s_{j_{(p',s'_m)}} \in \mathbf{s}_j$, create a transition labelled e' from $s_{j_{(p,s'_k)}}$ to $s_{j_{(p',s'_m)}}$ whenever $(p, s'_k) - e \rightarrow (p', s'_m)$ exists in M'_0.
- Mark $\mathbf{s}_j$ TP;

(ii) For $\forall s_{i_{(p,s'_k)}} \in \mathbf{s}_i$ and $\forall s_{j_{(p',s'_m)}} \in \mathbf{s}_j$, create a transition labelled e from $s_{i_{(p,s'_k)}}$ to $s_{j_{(p',s'_m)}}$ whenever $(p, s'_k) - e \rightarrow (p', s'_m)$ exists in M'_0.

(2) Change the mark of $\mathbf{s}_i$ from TP to PD ("ProcesseD").

Step 4:

Repeat

(a) For each $\mathbf{s}_j$ marked BS in H' and any state $s_{j_{(p',s'_m)}} \in \mathbf{s}_j$, if there is an $\mathbf{s}_i$ in H' and any state $s_{i_{(p,s'_k)}} \in \mathbf{s}_i$ such that $s_{i_{(p,s'_k)}} - e \rightarrow s_{j_{(p',s'_m)}}$ then remove all transitions labelled e from any state $s_{i_{(p,s'_k)}} \in \mathbf{s}_i$.

(b) For each $\mathbf{s}_i$ and each state $s_{i_{(p,s'_k)}} \in \mathbf{s}_i$, if there is no Rf $\in R(s'_k)$ such that $\text{Ref}_{\Sigma_S}(H', s_{i_{(p,s'_k)}}) \subseteq$ Rf then mark $\mathbf{s}_i$ BS.

Until no more $\mathbf{s}_i$ has been marked BS in the last Repeat.

Step 5:

If $\mathbf{s}_0$ is marked BS then report "no solution", otherwise compute $C = P_{\Sigma_o}(H)$. (The states marked BS do not belong to H.)

End

Since M_0 and Sc are assumed to be finite, this algorithm will eventually terminate. It is obvious that the computational complexity of step 3 is exponential in the worst case. However, according to our experience, the number of states of $\text{MGS}_{\Sigma_o}(M'_0)$ is of the same order as M'_0 for many applications. Step 4 of the algorithm can be implemented more efficiently by recursively checking only the states in which at least one transition is removed by the most recent manipulations of the algorithm.

Theorem 3. If there exists a deterministic converter C' such that $A_1 \,||\, \text{Cha} \,||\, C' \,||\, \text{Chb} \,||\, B_2 \angle_{\Sigma_S}$ Sc, then algorithm conversion will generate an optimized protocol converter C such that $A_1 \,||\, \text{Cha} \,||\, C \,||\, \text{Chb} \,||\, B_2 \angle_{\Sigma_S}$ Sc.

This theorem shows that our algorithm will always generate a protocol converter if it exists. Hence, our method has not the limitation of the methods using a service adapter [3,14,17]. In the following section, we will give an example to show the application of our method to mobile networks. To simplify the presentation, we use a deterministic service specification Sc. An example of using a nondeterministic service specification can be found in [21].

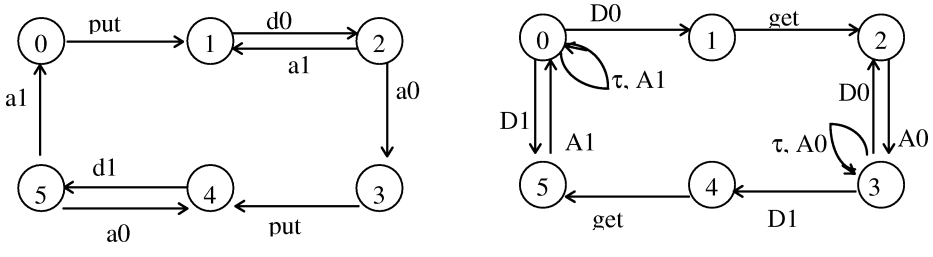
4.2. An example

In this section, we apply our algorithm to protocol conversion for mobile data networks. The configuration with a wireless channel between a base station and a mobile terminal is assumed asymmetric. The asymmetry is due to the fact that the mobile terminal has limited resources and smaller processing capability than the base station. In order to accommodate this asymmetry, it is propose in [1] that we should put as much intelligence as possible in terms of processing in the base stations and make the mobile terminals relatively simple:

(1) Since timers consume a lot of processing resources, they should always be implemented at the base station regardless of whether it is transmitting or receiving.

(2) The intelligence in terms of processing status messages and making decisions is implemented in base stations. Thus, a time-out in base station may trigger sending status messages.

In this example, the protocol $P = (P_1, P_2)$ used by the mobile terminal and the base station is shown in figure 5. The "put" and "get" events constitute the interface with the user. The events labelled with τ are internal events that model *time-out* or message *loss*. Other events are coupled with the channels. The protocol attaches a one bit sequence number to each message transmitted. Sending data messages are denoted as d_i (where $i = 1, 2$), and receiving data messages are denoted as D_i. For each received



(a) The sender P1 (b) The receiver P2

Figure 5. The P protocol.

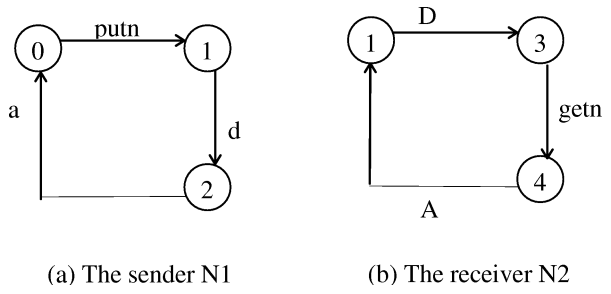


Figure 6. The N protocol.

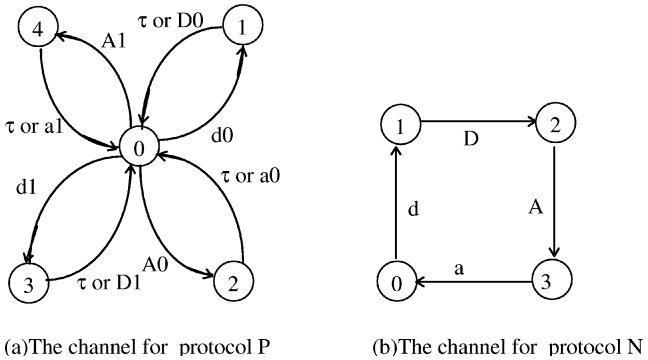


Figure 7. The channel specifications.

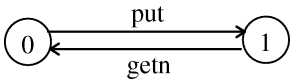


Figure 8. The service specification Sc.

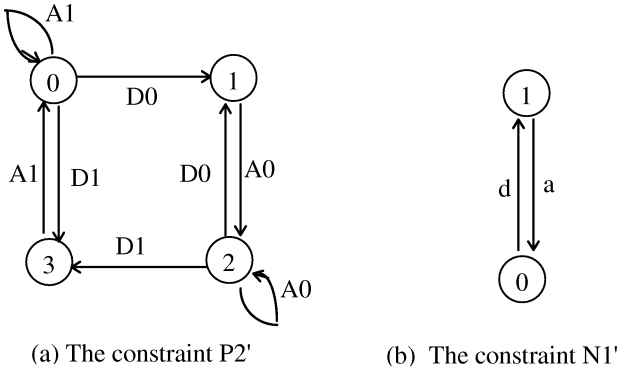


Figure 9. The generated constraints in the first step.

data message D_i, the receiver returns an acknowledge message A_i. If the sender received an acknowledgment a_i whose sequence number does not match the sequence number of the last-sent data message, the message will be ignored. In order to simplify our example, a non-sequence protocol $N = (N_1, N_2)$ with reliable channel is used between the base station and a fixed terminal, as shown in figure 6. The "putn" and "getn" events constitute the interface with the user. Sending data messages are denoted as d, and receiving data messages are denoted as D. For each received data message D, the receiver returns an acknowledge message A. Both protocols guarantee that a message will be delivered exactly once. The two channel specifications are depicted in figure 7. The desired service specification is shown in figure 8. Figure 9 shows the constraints, P_2' and N_1', obtained from the two protocol entities P_2 and N_1, respectively. The modified channel specifications, $\text{Chp}' = \text{Chp} \,\#\, P_2'$ and $\text{Chn}' = \text{Chn} \,\#\, N_1'$, are given in figure 10. Figure 11 is the refusal graph for the service specification. Since M_0 and $\text{MGS}_{\Sigma_o}(M_0 \,\#\, G_{\Sigma_S}(\text{Sc}))$ are too big to be presented due to limitation of space, we do not show them in this paper. Figure 12(a) is the optimized converter C constructed by our algorithm. For comparison, figure 12(b) shows an unoptimized protocol converter obtained by the algorithm proposed in [6,11], in which six states and twenty four transitions are superfluous.

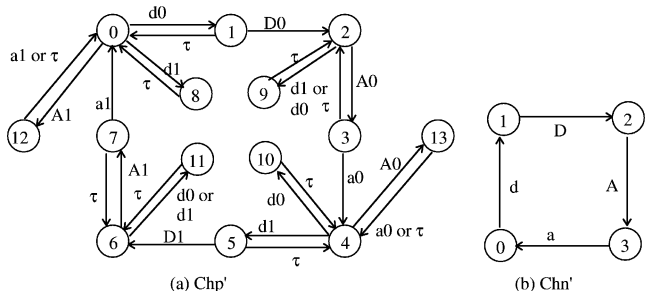


Figure 10. The modified channel specifications Chp′ and Chn′.

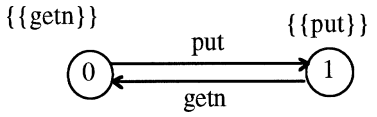


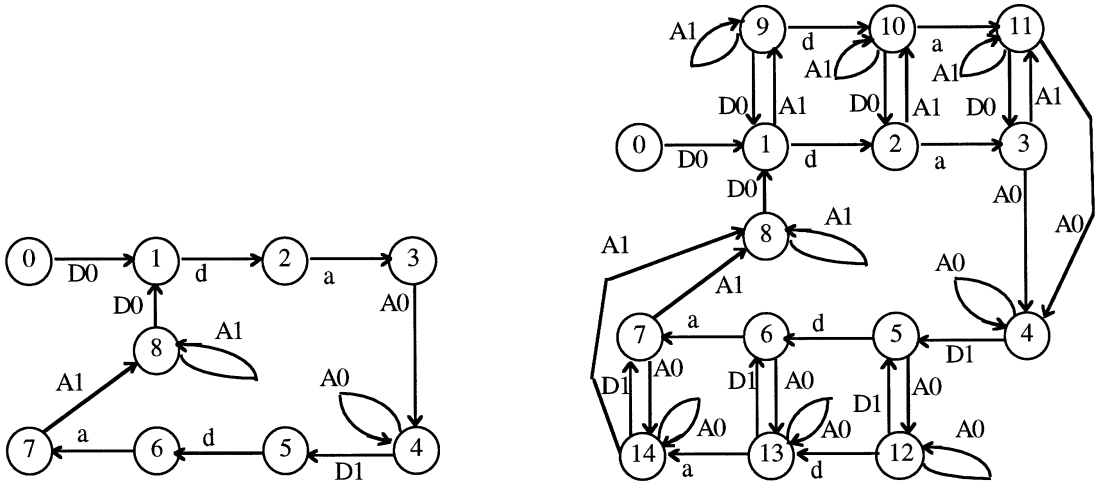
Figure 11. The specification of $G_{\Sigma_S}(\text{Sc})$.

5. Conclusions

In this paper we have defined the concept of an optimized protocol converter, and proposed a top-down algorithm to construct optimized converters from a given service specification and two protocol specifications. The basic idea is to derive constraints from the protocol specifications and impose the constraints on the channel specifications. The refusal graph and a reduction relation are used to deal with the problem of nondeterministic services. Compared with related works, our method has the following advantages: (1) it generates an optimized converter; (2) The service specification may be nondeterministic.

Appendix

Proof of lemma 1. Construct a relation $\zeta = \{\langle p, s_{i_p}\rangle \mid p$ is a state of M, s_{i_p} is a state of $\text{MGS}_{\Sigma_o}(M)\}$, then for any $\langle p, s_{i_p}\rangle \in \zeta$, $p - e \rightarrow p'$ in M implies $s_{i_p} - e \rightarrow s_{j_{p'}}$ in $\text{MGS}_{\Sigma_o}(M)$; and $s_{i_p} - e \rightarrow s_{j_{p'}}$ implies $p - e \rightarrow p'$ according to definition 13. Hence $\langle p', s_{j_{p'}}\rangle \in \zeta$. Therefore, ζ is a strong bisimulation relation and $M \cong \text{MGS}_{\Sigma_o}(M)$. □



(a) The optimized converter (b) An unoptimized converter

Figure 12.

Proof of lemma 2.

($\Rightarrow$) Assume that $H \cong M_0 \,\#\, P_{\Sigma_o}(H)$, then there is a strong bisimulation relation ζ over (States of H) $\times$ (States of $M_0 \,\#\, P_{\Sigma_o}(H)$) such that $\langle s_{0_{(p_0,s'_0)}}, (p_0, s_0)\rangle \in \zeta$ according to definition 5. It is easy to prove that $\langle s_{i_{(q,s'_k)}}, (p, s_j)\rangle \in \zeta$ if $i = j$ and $p = q$. If there is a state $s_{i_{(p,s'_k)}}$ in H, where p is a state of M_0 and s'_k is a state of P_{Σ_S}(Sc), and an event $e \in \Sigma_S$, such that $p - e \rightarrow p'$ in M_0 but $s_{i_{(p,s'_k)}} - e \nrightarrow$ in H (note that this is impossible for $e \in \Sigma_o \cup \{\tau\}$), then at state (p, s_i) of $M_0 \,\#\, P_{\Sigma_o}(H)$, we have $(p, s_i) - e \rightarrow$ according to the # product, where s_i is a state of $P_{\Sigma_o}(H)$. Therefore, $\langle s_{i_{(p,s'_k)}}, (p, s_i)\rangle \notin \zeta$. This is a contradiction with $H \cong M_0 \,\#\, P_{\Sigma_o}(H)$. Hence, the first condition is true.

If $(p - e \rightarrow) \wedge (s_{i_{(p,s'_k)}} - e \nrightarrow)$ and there is a state $s_{i_{(p',s'_m)}}$ in H such that $s_{i_{(p',s'_m)}} - e \rightarrow$, then $(p, s_i) - e \rightarrow$ must be true in $M_0 \,\#\, P_{\Sigma_o}(H)$ according to the # product. Therefore, $\langle s_{i_{(p,s'_k)}}, (p, s_i)\rangle \notin \zeta$. This is also a contradiction with $H \cong M_0 \,\#\, P_{\Sigma_o}(H)$. Hence, the second condition is true.

($\Leftarrow$) Assuming H is well-behaved. We can construct a relation ζ over (States of H) $\times$ (States of $M_0 \,\#\, P_{\Sigma_o}(H)$) such that $\langle s_{i_{(q,s'_k)}}, (p, s_j)\rangle \in \zeta$ iff $i = j$ and $q = p$. Obviously $\langle s_{0_{(p_0,s'_0)}}, (p_0, s_0)\rangle \in \zeta$, where s'_0 and s_0 are the initial states of P_{Σ_S}(Sc) and $P_{\Sigma_o}(H)$, respectively. For any $\langle s_{i_{(p,s'_k)}}, (p, s_i)\rangle \in \zeta$, under the assumption that H is well-behaved we have the following result: there is a state (p', s_n) such that $(p, s_i) - e \rightarrow (p', s_n)$ in $M_0 \,\#\, P_{\Sigma_o}(H)$ iff there is a state $s_{n_{(p',s'_m)}}$ such that $s_{i_{(p,s'_k)}} - e \rightarrow s_{n_{(p',s'_m)}}$ in H according to the # product. So, $\langle s_{n_{(p',s'_m)}}, (p', s_n)\rangle \in \zeta$. Therefore, we have $H \cong M_0 \,\#\, P_{\Sigma_o}(H)$. □

Proof of theorem 1. (1) If there is a deterministic solution C such that $M_0 \,||\, C \angle_{\Sigma_S}$ Sc, let $H' = M_0 \,\#\, C$, then we have $H' \angle_{\Sigma_S}$ Sc, and H' is well-behaved from lemma 2. Let $H'' = \text{MGS}_{\Sigma_o}(H')$ and $H''' = \text{MGS}_{\Sigma_o}(M_0 \,\#\, P_{\Sigma_S}(\text{Sc}))$. We construct a submachine H from H''' such that $\text{Tr}_{\Sigma_o}(H) = \text{Tr}_{\Sigma_o}(H'')$ (this can always be satisfied since $M_0 \,||\, C \angle_{\Sigma_S}$ Sc), and we construct a strong bisimulation relation ζ over (States of H'') $\times$ (States of H) such that $\langle s_{i_{(p,c)}}, s'_{k_{(q,s''_n)}}\rangle \in \zeta$ iff $p = q$ and there is a trace $t \in \text{Tr}_{\Sigma_o}(H) \cap \text{Tr}_{\Sigma_o}(H'')$ such that $s_0 - t \rightarrow s_i$ in $P_{\Sigma_o}(H'')$ and $s'_0 - t \rightarrow s'_k$ in $P_{\Sigma_o}(H)$. For any $\langle s_{i_{(p,c)}}, s'_{k_{(p,s''_n)}}\rangle \in \zeta$ and $s_{i_{(p,c)}} - e \rightarrow s_{j_{(p',c')}}$, we have $s'_{k_{(p,s''_n)}} - e \rightarrow s'_{m_{(p',s''_h)}}$ according to $H' \angle_{\Sigma_S}$ Sc and the # product. If $e \notin \Sigma_o$, then $i = j$ and $k = m$, hence, $\langle s_{j_{(p',c')}}, s'_{m_{(p',s''_h)}}\rangle \in \zeta$. If $e \in \Sigma_o$, then there must be a trace $t \in \text{Tr}_{\Sigma_o}(H) \cap \text{Tr}_{\Sigma_o}(H'')$ such that $s_0 - t \rightarrow s_i$ in $P_{\Sigma_o}(H'')$ and $s'_0 - t \rightarrow s'_k$ in $P_{\Sigma_o}(H)$ according to the # product, hence, $\langle s_{j_{(p',c')}}, s'_{m_{(p',s''_h)}}\rangle \in \zeta$. Therefore, ζ is a strong bisimulation relation and we have $H \cong H''$. Because of lemma 1 we have $H' \cong H''$ since $H'' = \text{MGS}_{\Sigma_o}(H')$. Therefore, $H \cong H'$. From $H' \angle_{\Sigma_S}$ Sc and $H \cong H'$, we have $H \angle_{\Sigma_S}$ Sc. H is well-behaved because H' is well-behaved.

(2) If such an H exists, according to lemma 2, $M_0 \,\#\, P_{\Sigma_o}(H) \cong H$. Hence, $M_0 \,||\, P_{\Sigma_o}(H) \cong H|_{\Sigma_o}$ where $H|_{\Sigma_o}$ denotes the FLTS derived from H by replacing every event $e \in \Sigma_o$ with an internal event τ. From the condition $H \angle_{\Sigma_S}$ Sc, we have $M_0 \,||\, P_{\Sigma_o}(H) \angle_{\Sigma_S}$ Sc. □

Proof of theorem 2. Assuming there is a converter C' such that $M \,||\, C' \angle_{\Sigma_S}$ Sc, then there is a submachine H of $\text{MGS}_{\Sigma_o}(M \# G_{\Sigma_S}(\text{Sc}))$ satisfying the conditions of theorem 1, where $\Sigma_o = \Sigma_0 - \Sigma_S$. We can construct an FLTS H' from $P_{\Sigma_{a_2} - \Sigma_S}(A_2) \,\#\, H \,\#\, P_{\Sigma_{b_1} - \Sigma_S}(B_1)$ such that $H' \angle_{\Sigma_S}$ Sc (note the conditions Sc $\angle_{\Sigma_{a_1} - \Sigma_o} A_1 \,||\, \text{Cha} \,||\, A_2$ and Sc $\angle_{\Sigma_{b_2} - \Sigma_o} B_1 \,||\, \text{Chb} \,||\, B_2$ in section 2.2). Since $E(H)$ is well-behaved with respect to $E(M_0)$ and Σ_o, from the construction of H' and the proof of theorem 1, $E(H')$ is well-behaved with respect to $E(M_0)$ and Σ_o as well. Therefore, $P_{\Sigma_o}(H')$ satisfies $M_0 \,||\, P_{\Sigma_o}(H') \angle_{\Sigma_S}$ Sc. This implies that there is a solution C satisfying $(A_1 \,||\, \text{Cha}) \,||\, P_{\Sigma_{a_2} - \Sigma_S}(A_2) \,\#\, C' \,\#\, P_{\Sigma_{b_1} - \Sigma_S}(B_1) \,||\, (\text{Chb} \,||\, B_2) \angle_{\Sigma_S}$ Sc. Hence, C is a solution satisfying $(A_1 \,||\, \text{Cha}') \,||\, C \,||\, (\text{Chb}' \,||\, B_2) \angle_{\Sigma_S}$ Sc.

From the definition of M_0, we have $\mathrm{Tr}_{\Sigma_{a_2}}(C) \subseteq \mathrm{Tr}_{\Sigma_o}(A_2)$ and $\mathrm{Tr}_{\Sigma_{b_1}}(C) \subseteq \mathrm{Tr}_{\Sigma_o}(B_1)$. Hence, C is an optimized converter satisfying $M_0 \,||\, C \angle_{\Sigma_S}$ Sc. □

Proof of theorem 3. The algorithm will generate a solution when $\mathbf{s}_0$ is not marked BS. Hence, we need to show that if there is a solution C', then $\mathbf{s}_0$ will not be marked BS.

(1) Let $M_0'' = \mathrm{MGS}_{\Sigma_o}(M_0)$ and $H' = M_0'' \,\#\, C'$. Since C' is a solution, H' is not empty, and $\mathbf{s}_0$ is included in H'. According to lemma 2, H' satisfies all conditions of theorem 1. Clearly, step 2, step 3 and step 4(a) will not remove any $\mathbf{s}_i$ and transitions of H' since H' is well-behaved.

(2) Because $H' \angle_{\Sigma_S}$ Sc, all of the transitions contained in H' will not be removed by step 4(b). Hence, if there is a deterministic solution C' satisfying $M_0 || C' \angle_{\Sigma_S}$ Sc, then the algorithm will generate a solution C satisfying $M_0 \,||\, C \angle_{\Sigma_S}$ Sc. From step 1 and theorem 2, C is optimized. □

References

[1] E. Ayanoglu et al., AIRMAIL: A link-layer protocol for wireless networks, Wireless Networks 1 (1995) 47–60.

[2] G. v. Bochmann, Deriving protocol converters for communication gateways, IEEE Transactions on Communications 38(9) (September 1990).

[3] G. v. Bochmann et al., Design principles for communication gateways, IEEE Journal on Selected Areas in Communications 8(1) (January 1990).

[4] E. Brinksma, G. Scollo and C. Steenbergen, LOTOS specification, their implementations, and their tests, in: *Proceedings of IFIP Workshop PSTV* (1987).

[5] K. Brown and S. Singh, Network, architecture and communication protocols for mobile computing (1996).

[6] K.L. Calvert and S.S. Lam, Deriving a protocol converter: A top-down method, in: *Proceedings of ACM SIGCOMM '89*.

[7] R. De Nicola, Extensional equivalences for transition systems, Acta Informatica 24 (1987).

[8] P.E. Green Jr, Protocol conversion, IEEE Transactions on Communications 34(3) (March 1986).

[9] D.M. Kristol et al., Efficient gateway synthesis from formal specifications, in: *Proceedings of ACM SIGCOMM '91*. See also IEEE/ACM Transactions on Networking 1(2) (April 1993).

[10] S.S. Lam, Protocol conversion, IEEE Transactions on Software Engineering 14 (March 1988).

[11] S.S. Lam and K.L. Calvert, Formal methods for protocol conversion, IEEE Journal on Selected Areas in Communications 8(1) (January 1990).

[12] G. Leduc, A framework based on implementation relations for implementing LOTOS specifications, Computer Networks and ISDN Systems 25 (1992).

[13] H.R. Lewis, *Elements of the Theory of Computation* (Prentice-Hall, Englewood Cliffs, NJ, 1981) pp. 59–62.

[14] P. Merlin and G. v. Bochmann, On the construction of submodule specifications and communication protocols, ACM TOPLAS 5(1) (1983).

[15] R. Milner, *Communication and Concurrency* (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[16] K. Okumura, A formal protocol conversion method, in: *Proceedings of ACM SIGCOMM '86*.

[17] K. Okumura, Generation of proper adapters and converters from a formal service specification, in: *Proceedings of IEEE INFOCOM '90*.

[18] M. Rajagopal et al., Synthesizing a protocol converter from executable protocol traces, IEEE Transactions on Computers 40(4) (April 1991).

[19] J.C. Shu and M.T. Liu, A synchronization model for protocol conversion, in: *Proc. IEEE INFOCOMM '89*, Ottawa, Canada (1989).

[20] J.C. Shu and M.T. Liu, An approach to indirect protocol conversion, Computer Networks and ISDN Systems 21 (1991).

[21] Z.P. Tao, A formal method for the design of real-time communicating subsystems and controllers, Ph.D. Thesis, Université de Montréal.

[22] Y.W. Yao, W.S. Chen and M.T. Liu, A modular approach to constructing protocol converters, in: *Proc. IEEE INFOCOM '90*, San Francisco, CA (1990).

Zhongping Tao received B.S. and M.S. degrees in telecommunications from HuaZhong University of Science and Technology, China, and a Ph.D. in computer science from the University of Montreal. He is currently employed by the Wireless Networks Division of Nortel Technology, Ottawa. His research interests include network architecture and internetworking, signaling systems, and protocols for mobile networks and broadband networks.

Gregor v. Bochmann is a professor at the University of Montreal since 1972 and holds the Hewlett-Packard-NSERC-CITI chair of industrial research on communication protocols. He is also one of the scientific directors of the Centre de Recherche Informatique de Montreal (CRIM). He has worked in the areas of programming languages, compiler design, communication protocols, and software engineering and has published many papers and some books in these areas. He has also been actively involved in the standardization of formal description techniques for OSI communication protocols and services. From 1977 to 1978 he was a visiting professor at the Ecole Polytechnique Federale, Lausanne, Switzerland. From 1979 to 1980 he was a visiting professor in the Computer Systems Laboratory, Stanford University, California. From 1986 to 1987 he was a visiting researcher at Siemens, Munich. His present work is aimed at methodologies for the design, implementation and testing of communication protocols and distributed systems. Ongoing projects include applications to high-speed protocols, distributed systems management and quality of service negotiation for distributed multimedia applications. He is a Fellow of the ACM.

Rachida Dssouli. Photograph and biography not available at time of publication.