# A Top-Down Method for Synthesizing Optimized Protocol Converters*

**Z.P.Tao, G.v. Bochmann and R. Dssouli**

Université de Montréal, Département d'Informatique et de
Recherche Operationnelle, C.P. 6128, Succ. "A"
Montréal, P.Q., Canada H3C-3J7
Email: {tao, bochmann, dssouli}@iro.umontreal.ca

***Abstract:*** In this paper, we propose a top-down method for constructing optimized protocol converters to achieve interoperability between heterogeneous computer networks. This method first generates a converter from a given service specification of the internetworking system and two protocol specifications, based on two important concepts: controllability and observability. The reduction relation [1] is used to compare the desired service specification and the internetworking system. Then an algorithm is developed for optimizing the converter. Compared with related works reported in [2], our method has two advantages: (1) It generates an optimized converter; and (2) the service specification for internetworking systems may be nondeterministic.

## 1. Introduction

One of the difficulties that arise in interconnecting heterogeneous networks is the problem of protocol mismatches [3] - 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; it receives messages from a protocol, interprets them, and delivers appropriate messages to the other protocols in a well-defined order. The protocols and the converter together form an internetworking system. We will define the protocol conversion problem formally in the next section.

A number of formal methods have been proposed for protocol conversion in the last several years. These methods can roughly be classified into the following two classes:

**1) Bottom-up methods,** which begin with analyzing heuristically the low level functions of the protocols to be converted in order to find PDU level constraints (for example message mapping relations), and these constraints are used to construct a converter from given protocol entities[8, 6, 12, 15, 13, 14]. A converter is correct if it satisfies the constraints, and has no deadlock nor livelock [12]. The common limitations of the bottom-up methods are: (1) a message mapping set is required which can only be obtained heuristically, and it is difficult to validate its correctness; (2) no service requirement of the interworking system is explicitly used, therefore the generated converter needs to be verified against the service specification after it is constructed.

**2) Top-down methods,** which use a service specification of interworking systems as the semantic constraint. The main methods are outlined below.

In [9], a two-stage approach was developed to derive protocol converters. In the first stage, a service adapter from the service specifications of the two protocols is constructed. In the second phase, a PDU level protocol converter can be constructed by directly composing the service adapter and the underlying protocol specifications. Since the trace equivalence relation is used in the first stage, the obtained internetworking system may lead to a deadlock state. Okumura discussed under what conditions the constructed internetworking system will inherit the properties from the original protocols.

It is possible that the converter constructed in this way may contain some states and transitions that are never executed. An efficient algorithm was presented in [4] to remove the superfluous 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 by the service adapter, and those that can be reached *only* from the unmatched service primitives; then the algorithm computes and retains the strongly connected component starting with the initial state, and discards the rest of the machine.

The disadvantage of the methods discussed above is that there may not exists a service adapter for two given protocols even if a PDU level converter does exist, therefore the application of the methods is limited.

Calvert [2] proposed a top-down method by using a service specification for the internetworking system. A safety property and a progress property are used to guarantee the correctness of the 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, channel specifications, 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.

A disadvantage of this approach is that the converter may contain superfluous states and transitions that may be harmful for the system performance. Another limitation is that the service specification of the internetworking system 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 the top-down method because of the following reasons:

1) The service specification is used explicitly; it is not necessary to validate the internetworking system against its service specification after the converter is generated.
2) It avoids using the message mapping set that is difficult to obtain and validate.

In this paper we propose an approach to overcome some of the limitations of the existing top-down methods. This approach first generates a converter from a given service specification of the internetworking system and two protocol specifications, based on two important concepts: controllability and observability. The reduction relation [1] is used to compare the desired service specification and the internetworking system. Then an algorithm is developed for optimizing the converter.

0-7803-2492-7/95 $4.00 © 1995 IEEE

Compared with related works reported in [2], our method has two advantages: (1) It generates an optimized converter; and (2) the service specification for the internetworking system may be nondeterministic.

This paper is organized as follows. Section 2 will give preliminary definitions and formalize the protocol conversion problem. In Section 3, firstly two important concepts, controllability and observability, are defined. Then, the theoretical basis of our approach will be given by using these concepts. In section 4, an algorithm for protocol conversion will be developed based on the idea presented in Section 3, and the correctness of the algorithm will be proved. Section 5 will propose an algorithm to optimize the converter generated by the algorithm proposed in Section 4, and an example shows the application of this algorithm.

## 2. Definition of the Problem

### 2.1. General Definitions

We will use the model of Finite Labelled Transition Systems (FLTS) [7]. An FLTS is defined as follows.

**Definition 1** (FLTS) [7]: A non deterministic FLTS M is a four-tuple $M = (Q, \Sigma, \delta, q0)$, where

- Q is a finite set of states.
- $\Sigma$ is a set of observable events.
- $\delta$ is a transition function, $\delta: (\Sigma \cup \{\tau\}) \times Q \to 2^Q$ with $\tau$ denoting an internal event, which defines a set $S \subseteq 2^Q$ of next states when an event $e \in \Sigma \cup \{\tau\}$ occurs in the current state $q \in Q$. When the FLTS is in state q, we say that the transition to q', written $q - e \to q'$ or $q' \in \delta(q, e)$, is enabled, where $\forall q, q' \in Q$, $e \in \Sigma \cup \{\tau\}$. The transition $q - e \to q'$ is said an incoming transition of q' and an outgoing transition of q.
- q0 is the initial state.

When the transition relation is defined by $\delta: \Sigma \times Q \to Q$, then such an FLTS is said *deterministic*.

The following notations are used in the rest of this paper.

| Notation | Meaning |
|---|---|
| $q - e \to$ | $\exists q'$, such that $q - e \to q'$. |
| $q - e \not\to$ | $\neg(q - e \to)$, i.e., there is no such a state q' such that $q - e \to q'$. |
| $q - \tau^k \to q'$ | An FLTS may engage in the sequence of k internal events, and after doing so, enters state q'. |
| $q - \sigma \to q_n$ | Assuming $\sigma = e_1 e_2 ... e_n$ ($e_i \in \Sigma \cup \{\tau\}$), then there exist states $q0, q_1, ..., q_n$ such that $q0 - e_1 \to q_1$, $q_1 - e_2 \to q_2$, ..., $q_{n-1} - e_n \to q_n$. $\sigma$ is called an execution sequence. |
| $q - \sigma \to$ | There exist states qn such that $q - \sigma \to q_n$ |
| E(M) | The set of all execution sequences of M, e.g., $E(M) = \{\sigma \mid q0 - \sigma \to\}$ |
| $q = e \Rightarrow q'$ | $\exists k0, k1 \in N$, such that $q - \tau^{k0} e \tau^{k1} \to q'$. |
| $q = e \Rightarrow$ | $\exists q'$, such that $q = e \Rightarrow q'$. |
| $q = e \not\Rightarrow$ | $\neg(q = e \Rightarrow)$, i.e., there is no such a state q' such that $q = e \Rightarrow q'$. |
| $q = t \Rightarrow q'$ | For $t = e1 ... en$ where e1, ..., en $\in \Sigma$, $\exists k0, ..., kn \in N$ such that $q - \tau^{k0} e1 \tau^{k1} e2 ... en \tau^{k0} \to q'$. t is called a trace. |
| $q = t \Rightarrow$ | $\exists q'$, such that $q = t \Rightarrow q'$. |
| $M\|_{\Sigma'}$ | Replace all the event $e \in \Sigma'$ by $\tau$ in M. |
| Tr(M) | Is the trace set of an FLTS M, i.e., $Tr(M) = \{t \mid q0 = t \Rightarrow\}$ |
| q *after* t | A state q in an FLTS M satisfying $q0 = t \Rightarrow q$ for a trace t and initial state q0. |
| Ref(M, t, q) | Is the refusal set of an FLTS M at state q after trace t, i.e., $Ref(M, t, q) = \{e \mid q = e \not\Rightarrow \text{ and } e \in \Sigma\}$. |

The behaviour of an FLTS is characterized by a set $\Sigma$ of its events and the order in which they are executed. The set $\Sigma$ contains all the externally visible events executed by the FLTS. Each event is considered atomic, i.e., no other event can overlap with an atomic event during the time interval from the initiation to the termination of the event. A system often contains two or more subsystems modelled by FLTSs. When two or more FLTSs are executed in parallel, they interact by requiring that an event in one FLTS be executed at the same time with another event in at least one other FLTS. If two or more FLTSs interact in this way, we say that the events required to be executed jointly are *directly coupled* , and together form an atomic event (or an interaction). To avoid confusion, we assume that all uncoupled events in the interacting FLTSs have a unique name. The interaction between the FLTSs can be specified by assigning the same name to events that are directly coupled. 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 FLTSs.

Directly coupled events may not be visible by the environment of the system, hence we may model them by internal events for convenience. In this way we define a *coupled product* of two FLTSs as follows.

**Definition 2** (*coupled product* ): A *coupled product* M1 || M2 of two FLTSs $M1 = (Q1, \Sigma1, \delta1, p0)$ and $M2 = (Q2, \Sigma2, \delta2, q0)$ is an FLTS $M = (Q, \Sigma, \delta p, (p0, q0))$ such that:

- Q is a subset of $Q1 \times Q2$; each element is of the form (p, q), where $p \in Q1, q \in Q2$;
- $\Sigma = (\Sigma1 \cup \Sigma2) - (\Sigma1 \cap \Sigma2)$;
- (p0, q0) is the initial state;
- $\delta p$ is the transition function defined on Q such that for $p, p' \in Q1, q, q' \in Q2$:
  1) $(p, q) - e_1 \to (p', q)$ if $p - e_1 \to p'$ and $e_1 \in (\Sigma1 - \Sigma2) \cup \{\tau\}$;
  2) $(p, q) - e_2 \to (p, q')$ if $q - e_2 \to q'$ and $e_2 \in (\Sigma2 - \Sigma1) \cup \{\tau\}$;
  3) $(p, q) - \tau \to (p', q')$ if $p - \mu \to p'$, $q - \mu \to q'$ and $\mu \in \Sigma1 \cap \Sigma2$
  4) for other cases, no transition is defined.

In some cases, we need to compose two FLTSs that do not directly interact with each other (they have no common events). This can be done by computing the so-called *Cartesian cross product* , written M1 × M2, which is identical to the coupled product M1 || M2 in the case that $\Sigma1 \cap \Sigma2 = \phi$.

**Definition 3** (*# product* ): A *# product* M1 # M2 of two FLTSs $M1 = (Q1, \Sigma1, \delta1, p0)$ and $M2 = (Q2, \Sigma2, \delta2, q0)$ is an FLTS $M = (Q, \Sigma, \delta p, (p0, q0))$ where:

- Q is a subset of $Q1 \times Q2$;
- $\Sigma = \Sigma1 \cup \Sigma2$ is the set of events;
- (p0, q0) is the initial state;

- $\delta p$ is the transition relation defined on Q such that for p, p' ∈ Q1 and q, q' ∈ Q2:
  1) $(p, q) - e_1 \rightarrow (p', q)$ if $p - e_1 \rightarrow p'$ and $e_1 \in \Sigma 1 \cup \{\tau\} - \Sigma 2$;
  2) $(p, q) - e_2 \rightarrow (p, q')$ if $q - e_2 \rightarrow q'$ and $e_2 \in \Sigma 2 \cup \{\tau\} - \Sigma 1$;
  3) $(p, q) - \mu \rightarrow (p', q')$ if $p - \mu \rightarrow p'$ and $q - \mu \rightarrow q'$ with $\mu \in \Sigma 1 \cap \Sigma 2$.
  4) for other cases, no transition is defined.

Compared with the coupled product, the only difference is that the directly coupled events in the # product are not hidden (as internal events).

**Definition 4** *(* product)*: A ** product* M1 * M2 of two FLTSs M1 = (Q1, Σ1, δ1, p0) and M2 = (Q2, Σ2, δ2, q0) is an FLTS transformed from M1#M2 by recursively removing any state (p, q) satisfying the following condition: there is no $e \in \Sigma \cup \{\tau\}$ such that $(p, q) - e \rightarrow$ and there is an event $e \in \Sigma \cup \{\tau\}$ such that $p - e \rightarrow$ or $q - e \rightarrow$.

**Definition 5** *(trace equivalent)*[7]: Two FLTSs M1 and M2 are trace equivalent, denoted M1 ~ M2, if Tr(M1) = Tr(M2).

**Definition 6***(reduction) [1]:* Given two FLTSs M1 = (Q1, Σ1, δ1, p0) and M2 = (Q2, Σ2, δ2, q0), M1 is a reduction of M2, written M1 ∠ M2, if the following conditions are satisfied:
  1) Trace (M1) ⊆ Trace(M2).
  2) For any t ⊆ Tr(M1) ∩ Tr(M2) and any q after t in M1, there is a state p after t in M2 such that Ref(M1, t, q) ⊆ Ref(M2, t, p).

**Definition 7** *(testing equivalence) [1]:* Given two FLTSs M1 and M2, M1 is testing equivalent to M2, written M1 ≈ M2, if M1 ∠ M2 and M2 ∠ M1.

**Definition 8** *(strong bisimulation relation)[11]:* A binary relation ξ on states is a strong bisimulation if for each $(p, q) \in \xi$ and each event $e \in \Sigma \cup \{\tau\}$, the following two conditions hold:
  1) Whenever $p - e \rightarrow p'$ then for some q', $q - e \rightarrow q'$ and $(p', q') \in \xi$;
  2) Whenever $q - e \rightarrow q'$ then for some p', $p - e \rightarrow p'$ and $(p', q') \in \xi$;

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

**Definition 9** *(strong bisimulation relation of two FLTS)[11]:* Given two FLTSs M1 = (Q1, Σ1, δ1, p0) and M2 = (Q2, Σ2, δ2, q0), M1 and M2 are strong bisimulation equivalent if there is a strong bisimulation relation ξ which contains the pair (p0, q0) of their initial states, written M1 ≅ M2.

**Definition 10** *(Submachine)*: An FLTS M' = (Q', Σ', δ', q0') is a submachine of another FLTS M = (Q, Σ, δ, q0) if (a) q0' = q0; (b) Q' ⊆ Q; (c) Σ' ⊆ Σ; and (d) δ' ⊆ δ.

### 2.2. Formal Definition of Protocol Conversion

The protocol conversion problem can be stated as follows. We consider two different protocols A and B (see Fig. 1). Suppose the two protocols provide similar services, but differ in certain details, and we want A1 ( B1) to interoperate with B2 (A2) via a protocol converter C, such that the internetworking system provides the required services specified by Sc as shown in Fig.2(a). This can be formally written as:

A1||Cha||C || Chb||B2 ∠ Sc

where Cha and Chb denote the channels between the protocol entities. Since A1, Cha, Chb, and B2 are given, this expression can be represented by:

M0||C ∠ Sc

where M0 = (A1||Cha)×(Chb||B2) as shown in Fig.2(b).

In addition, the following requirements are also important:

- *Conversion transparency:* The converter C should work in such a way that A1 communicates with B2 as if A1 communicates with A2, and B2 communicates with A1 as if B2 communicates with B1. *This implies that C should be some form of combination of A2 and B1.* Formally, this can equivalently be represented as a requirement on the service specification Sc:

  $Sc|_{\Sigma b2} \approx Sa|_{\Sigma a1}$ and

  $Sc|_{\Sigma a1} \approx Sb|_{\Sigma b2}$

  where Sa and Sb are the service specifications of protocol A and B, respectively.
- The constructed converter should have no superfluous transitions and states that do not contribute to the progress of the system.

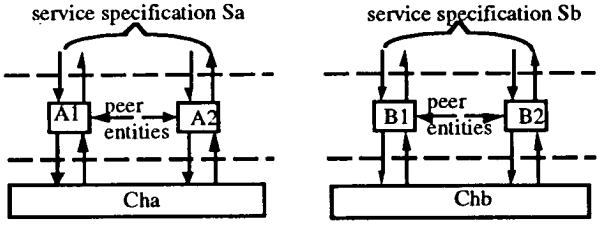



Fig.1 Two protocols A and B

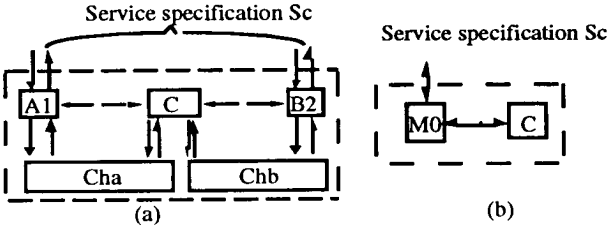



Fig.2 Problem statement of protocol conversion

## 3. Controllability and Observability

In this section, we first introduce two important concepts: the controllability and observability, which are similar to the concepts defined in [17], but have not exactly the same meaning

When an FLTS M interacts with another FLTS M', some o the events in M may not participate in the interaction (i.e., those events which do not need to execute jointly with the events in M'). Consequently, the event set Σ of M can be partitioned into two disjoint sets, e.g., $\Sigma = \Sigma o \cup \Sigma u$, where Σo denotes the set o events able to directly interact with M', and Σu denotes the se of events unable to directly interact with M'. We define a projection such that P: $\Sigma \rightarrow (\Sigma o \cup \{\varepsilon\})$ by:

$$P(\mu) = \begin{cases} \mu, \text{ if } \mu \in \Sigma o \\ \varepsilon, \text{ if } \mu \notin \Sigma o \end{cases}$$

For an internal event τ we define $P(\tau) = \varepsilon$, where ε denotes a null event. This function can be extended to an executio sequence by defining $P(\varepsilon) = \varepsilon$, $\varepsilon^n = \varepsilon$ and

$P(\sigma e) = P(\sigma)P(e)$ for $\sigma \in (\Sigma \cup \{\tau\})^*$ and $e \in \Sigma \cup \{\tau\}$.

where $\sigma e$ denotes the concatenation of $\sigma$ and e. Under the projection P, an execution sequence $\sigma$ of M is observed as the behaviour $P(\sigma) \subseteq \Sigma^*$ by M'.

**Definition 11** (Controllability): Given $M = (Q, \Sigma, \delta, q0)$ with $\Sigma = \Sigma o \cup \Sigma u$, and $K \subseteq E(M)$, K is said to be *controllable* if $(K (\Sigma u \cup \{\tau\})) \cap E(M) \subseteq K$, where $K (\Sigma u \cup \{\tau\})$ denotes a concatenation of an execution sequence in K with an event in $\Sigma u \cup \{\tau\}$.

This definition means that if an event in $\Sigma u \cup \{\tau\}$ occurs following a sample path in K, then the extended sample path must remains in K, provided that the extended path is in E(M).

**Definition 12** (Observability): Given $M = (Q, \Sigma, \delta, q0)$ with $\Sigma = \Sigma o \cup \Sigma u$, and $K \subseteq E(M)$, K is *observable* iff for any two execution sequences $\sigma$ and $\sigma'$ in K satisfying $P(\sigma) = P(\sigma')$, there does not exist an event $e \in \Sigma \cup \tau$ such that either $\sigma e \in K$ and $\sigma' e \in E(M) - K$ or $\sigma e \in E(M) - K$ and $\sigma' e \in K$.

This definition implies the condition of observability: if $P(\sigma) = P(\sigma')$ then all the one step continuations of $\sigma$ and $\sigma'$ that remain in E(M) yield the same result with respect to the membership of K. The projection P retains sufficient information to decide whether or not, after the occurrence of an event, the resultant execution sequence is in K.

**Definition 13** (*After set*): For $M = (Q, \Sigma, \delta, q0)$ with $\Sigma = \Sigma u \cup \Sigma o$, we define the relation $p\ R\ p'$ which is satisfied if $\exists e \in \Sigma u \cup \{\tau\}$ and $p, p' \in Q$ such that $p - e \rightarrow p'$. We define the *after set* of state p as $A(p) = \{p' \mid p\ R^*\ p'\} \cup \{p\}$, where $R^*$ represents the reflective and transitive closure of relation R.

The *After* set A(p) intuitively describes all the reachable states from p by executing zero, one or more events $e \in \Sigma u \cup \{\tau\}$. In order to deal with the observability problem in our algorithms, we define the following concept.

**Definition 14** (PB-machine): Given an FLTS $M = (Q, \Sigma, \delta, q0)$ and a projection P, an FTLS $M' = (Q', \Sigma', \delta', q0')$, with $Q \cap Q' = \varnothing$, is called a PB-machine of M, written B(M), if it satisfies the following two conditions:

1) The state space S of B(M) can be divided into disjoint subsets X0, X1, X2, ..., Xn (e.g., $Q' = X1 \cup X2 \cup ... \cup Xn$, there is not a state p such that $(p \in Xi) \wedge (p \in Xj) \wedge (i \neq j)$) satisfying:
   X0 = A(q0); for any Xi and Xj, $\forall e \in \Sigma o$, if $\exists q' \in Xi$ and $\exists q \in Xj$ such that $q - e \rightarrow q'$ then $Xi = \cup_{p \in Xj} \{A((p')) \mid p - e \rightarrow p'\}$, where $1 \leq i \leq n$, $0 \leq j \leq n$.
   We call Xi a *State-set.*
2) M and B(M) satisfy $M \cong B(M)$.

From this definition, a PB-machine has two useful properties:

1) For any Xi, if there is a state $p \in Xi$ and an execution sequence $\sigma$ such that $p0 - \sigma \rightarrow p$, then for any state $q \in Xi$ there must have an execution sequence $\sigma'$ such that $p0 - \sigma' \rightarrow q$ and $P(\sigma') = P(\sigma)$. This property is useful for checking the observability since the observability is defined based on $P(\sigma') = P(\sigma)$.
2) If $\{\sigma 1\}$ is uncontrollable or unobservable and there is a state $q \in Xi$ and an execution sequence $\sigma 1'$ such that $p0 - \sigma 1' \rightarrow q$ and $\sigma 1 = \sigma 1' \sigma 1''$, where $\sigma 1' \sigma 1''$ is a concatenation of $\sigma 1'$ with $\sigma 1''$, then for any execution sequence $\sigma 2$, if there is an execution sequence $\sigma 2'$ such that $p0 - \sigma 2' \rightarrow q'$, $q' \in Xi$ and $\sigma 2 = \sigma 2' \sigma 2''$, then $\{\sigma 2, ...\}$ must be unobservable. This property results in an efficient algorithm in this paper that guarantees finding a solution if it exists.

**Definition 15** (*P-machine*): A *P-machine* of a given FLTS $M = (Q, \Sigma, \delta, q0)$ with $P(\Sigma) = \Sigma o$ is defined by $P_{\Sigma o}(M) = (Q', \Sigma o, \delta', q0')$ based on B(M) where

- Q' is a set of states $\{q0, q1, ..., qn\}$;
- $\Sigma o = P(\Sigma)$ is a set of events;
- $\delta'$ is a transition relation defined on $\{q0, q1, ..., qn\}$ by $qj - \mu \rightarrow qi$ for $\mu \in \Sigma o$ if there is a state q in Xi and a state p in Xj such that $p - \mu \rightarrow q$, where Xi is defined in B(M), $0 \leq i \leq n$ and $0 \leq j \leq n$;
- q0 is the initial state.

In this definition, Xi ($0 \leq i \leq n$) is considered as one state qi. In the next section, we will use $X^{-1}(q_i)$ to denote the set of states Xi in M corresponding to the state $q_i$ in $P_{\Sigma o}(M)$. This definition is useful since our goal is to find a deterministic solution for a given (nondeterministic) FLTS M0 and a requirement specification Sc.

**Lemma 1:** Given two FLTSs $M = (Q, \Sigma, \delta, p0)$ and $M' = (Q', \Sigma', \delta', p0')$ with $\Sigma' \subseteq \Sigma$, $E(M') \subseteq E(M)$, and a projection $P(\Sigma) = \Sigma o$. $M' \cong M \# P_{\Sigma o}(M')$ iff E(M') is observable and controllable.

This lemma implies that only the behavior E(M') of E(M) is allowed to happen if M interacts with $P_{\Sigma o}(M')$ under the condition that E(M') is controllable and observable. The execution sequences in E(M) - E(M') are forbidden. Due to the complexity of the proof, we will not prove it here because of the limitation of space.

**Theorem 1:** Given $M0 = (Q0, \Sigma 0, \delta 0, q0)$, $Sc = (Qs, \Sigma s, \delta s, q0s)$ and a projection $P(\Sigma 0) = \Sigma 0 - \Sigma s = \Sigma o$. There is a deterministic protocol converter $C = (Qc, \Sigma o, \delta c, q0c)$ such that $M0 \| C \angle Sc$ iff there exists an FLTS H with $E(H) \subseteq E(M0)$ such that $H|_{\Sigma O} \angle Sc$ and E(H) is observable and controllable.

Proof: If such a H exists, according to lemma 1, $M0 * P_{\Sigma o}(H) \cong H$, therefore $M0 \| P_{\Sigma o}(H) \cong H|_{\Sigma O}$. From the condition $H|_{\Sigma O} \angle Sc$, we have $M0 \| C \angle Sc$, where $C = P_{\Sigma o}(H)$. On the other hand, if there is a solution C such that $M0 \| C \angle Sc$, let $H = M0 \# C$, we have $H|_{\Sigma O} \angle Sc$. It is easy to check that $C = P_{\Sigma o}(H)$, therefore $H = M0 \# P_{\Sigma o}(H)$. From lemma 1, E(H) is observable and controllable.

From theorem 1 and the construction of its proof, we have the following basic idea for constructing an algorithm for protocol conversion: in order to construct a protocol converter we can first find an FLTS $M0' = M0 * P_{\Sigma s}(Sc)$, and then find an FLTS H from M0' such that $H|_{\Sigma O} \angle Sc$ and E(H) is observable and controllable. Finally we can obtain a converter by computing $C = P_{\Sigma o}(H)$.

## 4. The basic algorithm

For convenience, given $M0 = (Q0, \Sigma 0, \delta 0, q0)$ and $Sc = (Qs, \Sigma s, \delta s, q0s)$, we define $\Sigma u = \Sigma 0 \cap \Sigma s$. The following algorithm is constructed according to theorem 1. It is divided into four steps explained briefly as follows:

*In the first step,* M0' is obtained by computing $M0' = M0 \# P_{\Sigma s}(Sc)$. A state of M0' can be represented by (p, q), where p is a state of M0 and q is a state of $P_{\Sigma s}(Sc)$. For any state (p, q) of M0' and $\mu \in \Sigma u$, if $p - \mu \rightarrow p'$ in M0, but not $q - \mu \rightarrow q'$ in $P_{\Sigma s}(Sc)$, then at state p in M0 there is an execution sequence $\sigma'$

satisfying $p0 - \sigma' \rightarrow p$ such that $\sigma'\mu \in E(M0)$, but $\sigma'\mu \notin E(M0')$. So (p, q) should be forbidden for controllability by marking it Bad State (denoted as BS in the algorithm). The merging of the states that are strong bisimulation equivalent is for simplifying the computation in later steps.

*In the second step,* all execution sequences of M0' that violate controllability and observability are removed. First M0' is transformed iteratively into an FLTS F = B(M0') starting from X0 = A((p0, q0)). During the process of the transformation, any Xi marked BS is not necessarily be expanded. Second, any execution sequences that do not satisfy the conditions of controllability and observability are removed from F to construct a submachine F' of F. In any constructed State-Set Xi, if a state (p, q) ∈ Xi is marked BS, then this Xi should be marked BS (for controllability and observability). For two State-Set Xi and Xj, if Xj is marked BS, there is a state (p, q) ∈ Xi, a state (p', q') ∈ Xj, and there is an event $\mu \in \Sigma o$ such that $(p, q)-\mu\rightarrow(p', q')$, then the algorithm forbids event μ at Xi (for observability).

Step 3 has two tasks. The first task is to retain controllability and observability of F' after some states and transitions, which violate reduction relation, are removed, similar to the method used in step 2. The second task is to remove the states and transitions from F' that violate the reduction relation $F'|_{\Sigma o} \angle Sc$. This can be done directly by the definition of the reduction relation: from the definition of the # product, for any state (p, q) in $F'|_{\Sigma o}$, there exists a trace t such that $p0 = t \Rightarrow p$ and for any state $qq \in X^{-1}(q)$, $q0 = t \Rightarrow qq$. In order to construct an FLTS H from F' such that $H|_{\Sigma o} \angle Sc$, for every (p, q) in F' we need to check if there exists a state $qq \in X^{-1}(q)$ such that $Ref(F'|_{\Sigma o}, t, (p, q)) \subseteq Ref(Sc, t, qq)$ for a trace t of Sc and $F'|_{\Sigma o}$. If such a qq does not exist, then the reduction relation is not satisfied at state (p, q), and (p, q) should be marked BS.

In step 4, a solution C = (Qc, Σo, δc, q0c) is constructed from H by computing $C = P_{\Sigma o}(H)$.

**Algorithm-GI**

/* Input: $M0 = (Q0, \Sigma 0, \delta 0, p0)$, $\Sigma o = P(\Sigma 0) = \Sigma 0 - \Sigma s$;
/* Sc = (Qs, Σs, δs, q0s).
/* Output: the solution C for M0||X ∠ Sc.

**Begin**

**Step 1** *Generate M0' from M0 and Sc:*

1) Compute $M0' = M0\#P_{\Sigma s}(Sc)$, and mark any state (p, q) of M0' BS if there is a state p' in M0 such that $p - \mu \rightarrow$ p' for $\mu \in \Sigma u$, but there is not a state q' in $P_{\Sigma s}(Sc)$ such that $q - \mu \rightarrow q'$.
2) Merging any states in M0' that are strong bisimulation equivalent; the result is denoted as $M0' = (Qp, \Sigma p, \delta p, q0_p)$;.

**Step 2** *Generate an FLTS F' from M0':*

Create $X0 = A((p0, q0_p))$ and mark it TP; /*TP = To be Processed */
Create a transition labelled $e \in \Sigma u \cup \{\tau\}$ from (p, q) ∈ X0 to (p', q') ∈ X0 whenever $(p, q) - e \rightarrow (p', q')$ exists.
Do the following while there is an Xi marked TP:

1) If there is a state (p, q) ∈ Xi marked BS then mark Xi BS; otherwise for every $e \in \Sigma o$ do the following:
   a) Compute
   $$Xi(e) = \bigcup_{(p,q)\in Xi} \{A((p', q'))|(p, q)-e\rightarrow(p', q')\};$$
   b) If there is a state (p, q) ∈ Xi(e) marked BS then forbid any transition labelled e at Xi; otherwise do the following:
      i) if Xi(e) is not empty and there are no previously created Xj containing exactly all the state pairs in Xi(e)), do the following:
         - Create such an Xj containing all the state pairs in Xi(e).
         - Create a transition labelled $ei \in \Sigma u \cup \{\tau\}$ from (p, q) ∈ Xi to (p', q') ∈ Xi whenever $(p, q) - ei \rightarrow (p', q')$ exists.
         - Mark Xj TP;
      ii) Create a transition labelled e from (p, q) ∈ Xi to (p', q') ∈ Xj whenever $(p, q) - e \rightarrow (p', q')$ exists.
2) Mark Xi PD. /* PD = ProcesseD */

**Step 3** *Generate an FLTS H from F':*

Repeat
   a) For each Xj marked BS in F' and a state (p', q') ∈ Xj, if there is an Xi and a state (p, q) ∈ Xi such that $(p, q) - e \rightarrow (p', q')$ then forbid any transition labelled e at Xi. /*for observability*/
   b) For each Xi and each state (p, q) ∈ Xi, if there is no state $qq \in X^{-1}(q)$ such that $Ref(F'|_{\Sigma o}, t, (p, q)) \subseteq Ref(Sc, t, qq)$ then mark Xi BS.

Until no more Xi marked BS.

**Step 4** *Generate the converter C from H:*

If X0 is marked BS then report "no solution C", otherwise compute $C = P_{\Sigma o}(H)$.

**End**

Since M0 and Sc are assumed to be finite, the algorithm will eventually terminate. It is obvious that the computational complexity of step 2 is exponential in the worst case. However, according to our experience the number of states of B(M0') is in the same order as M0' for many applications.

The algorithm can be implemented more efficiently in step 2 and 3 by recursively checking only the states in which at least one transition is removed by the most recent manipulations of the algorithm. However, for the sake of presentation here, we do not optimize it. Now we will prove the correctness of the algorithm.

**Theorem 2:** If there is a deterministic solution C' for M0||X ∠ Sc, then Algorithm-GI will generate a converter C such that M0||C ∠ Sc.

Proof 1) Let $M0'' = B(M0')$, $P(\Sigma 0) = \Sigma o = \Sigma 0 - \Sigma s$ and $H' = M0''\#C'$. Since C' is a solution, H' is not empty, and X0 is included in H'. According to Theorem 1, $H'|_{\Sigma o} \angle Sc$ and E(H') is observable and controllable. Clearly, step 2 and step 3(a) will not remove any State-Set and transition of H' since E(H') is observable and controllable.

2) Because $H'|_{\Sigma o} \angle Sc$, all of the State-Sets and transitions contained in H' will not be removed by step 3(b).

Since a State-Set can only be removed by step 2 and step 3, X0 will not be removed by Algorithm-GI from (1) and (2). Hence, if there is a deterministic solution C' for M0||X ∠ Sc, then Algorithm-GI is

able to generate a converter C. From theorem 1, we have M0||C ∠ Sc.

**Definition 16** *(maximum solution)*: A converter C is a maximum solution if for any other converter C' satisfying M0||C' ∠ Sc, we have Tr(C' ) ⊆ Tr(C).

**Lemma 2:** The converter C constructed by algorithm-GI is a maximum solution.

Proof: For any other solution C', let H' = M0*C', then any state and transition of H' will not be removed by Algorithm-GI from the proof of theorem 2. Therefore Tr(H') ⊆ Tr(H) is true. This implies Tr(C') ⊆ Tr(C).

**Example 1**: Fig.3 (a) is the service specification Sc, and Fig.3(b) is the specification M0. Fig.4(a) is the H specification obtained by step 3. Fig.4(b) is the converter obtained from H (the states that are strong bisimulation are merged), where Σo = {b, c, d, e}. Obviously, M0||C ∠ Sc holds. The state labelled "0" is the initial state in all the figures.

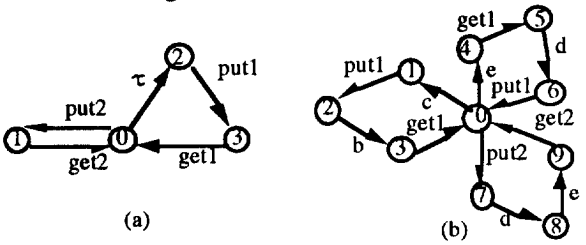



Fig.3 (a) the specification Sc, (b) the specification M0.

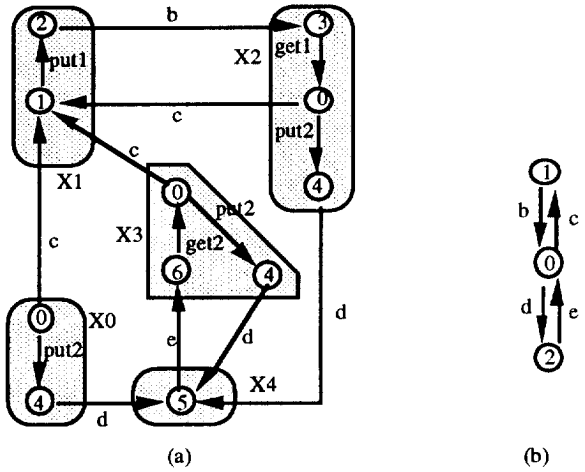



Fig.4 the M0' and the constructed C.

## 5. The algorithm for optimization

A disadvantage of algorithm-GI and the related work in [2] is that the solution may contain superfluous states and transitions that may be harmful for the system performance. An example converter given in [2] is shown in Fig.5. The states and transitions inside the dotted box are superfluous. The converter can send back a unnecessary acknowledgment to a protocol even after receiving the data message correctly. If these states and transitions are not removed, the system performance will certainly be worse than optimal, although the converter is correct functionally.

For this problem, we have the following two basic observations:

1) The 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). From the construction of M0, the unnecessary message sequences are also included in M0. However, the existing algorithms have no measure to remove them, when C is generated from M0.
2) Normally there is no superfluous states and transitions in the given protocol A and B; however, this information is not used by existing algorithms.

Based on the observation (2), we define an optimized converter as follows.

**Definition 17** *(optimized converter)*: A converter C is optimized with respect to A2 and B1 if C is a *maximum solution* under the following condition: Tr(C|$\Sigma b1$) ⊆ Tr(A2|$\Sigma a'$) and Tr(C|$\Sigma a2$) ⊆ Tr(B1|$\Sigma b'$), where $\Sigma a' = \Sigma a2 - \Sigma c$, $\Sigma b' = \Sigma b1 - \Sigma c$, $\Sigma a2$ is the set of events of A2 and $\Sigma b1$ is the set of events of B1.

Intuitively, the condition Tr(C|$\Sigma b1$) ⊆ Tr(A2|$\Sigma a'$) and Tr(C|$\Sigma a2$) ⊆ Tr(B1|$\Sigma b'$) implies that C should not do more than $P_{\Sigma o}(A2 \times B1)$ can do - if A2 and B1 have no superfluous transitions, then C has not.

The algorithm provided below is used to optimize a converter generated by the algorithm proposed in the last section (or by the method reported in [2]), based on definition 17. It works as follows. The first step is to construct the specification M0 from the protocol specifications and the channel specifications. The second step is to obtain a maximum solution C by solving the equation M0 || X ∠ Sc using Algorithm-GI or the algorithm presented in [2]. The third step is to obtain the constraints A2' and B1' from the given protocols A and B. The final step is to impose the constraints A2' and B1' to C by computing A2'*C*B1'. In this way we obtained the optimized converter $C_{opt}$.

**Algorithm-OI:**

/*Input: Protocol A = (A1, A2), B = (B1, B2), the Channels Cha
/*and Chb, the global service specification Sc
/*Output: an optimized converter Copt.

**Begin**

**Step 1**: Construct an FLTS M0 = (A1||Cha)×(Chb||B2).
**Step 2**: Construct an FLTS C such that M0 || C ∠ Sc by using Algorithm-GI or the algorithm proposed in [2].
**Step 3**: Obtain the constraints: $A2' = P_{\Sigma a2 - \Sigma s}(A2)$ and $B1' = P_{\Sigma b1 - \Sigma s}(B1)$.
**Step 4**: Compute $C_{opt}$ = A1'*C*B2'.

**End**

**Theorem 3:** The converter $C_{opt}$ generated by Algorithm-OI is optimized such that A1||Cha||$C_{opt}$||Chb||B2 ∠ Sc.

We do not prove this theorem here because of the limitation of space. The application of our method to a simple example used in [2] is given below. Fig.5 is the converter generated by the algorithm proposed in [2] (see Fig.27 in [2]). Fig.6 and Fig.7 are the two protocols. Fig.8 gives the specification of the constraints generated by step 3 of Algorithm-OI. Fig.9 is the result generated by step 4 of Algorithm-OI; it is easy to check that this is an optimized converter.

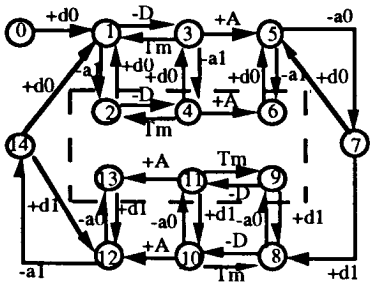
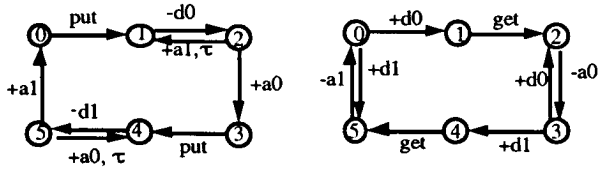


Fig.5 A converter with useless states and transitions

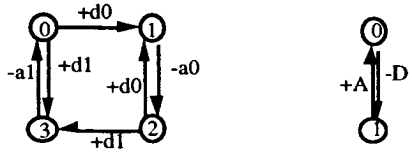


(a) The Sender (A0) (b) The Receiver (A1)

Fig. 6 The AB protocol specification

(a) The Sender (N0) (b) The Receiver (N1)

Fig.7 The NS protocol



(a) A1', (b) N0'

Fig.8 The generated constraints in the first step of Algorithm-OI.

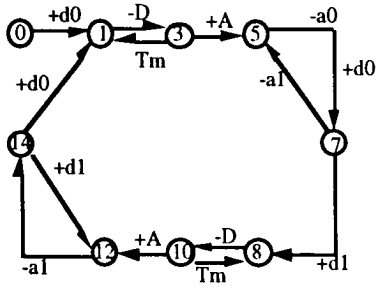


Fig.9 The optimized converter (the result of step 4 of Algorithm-OI).

## 6. Conclusions

In this paper, we have proposed a top-down method for synthesizing optimized protocol converters. Compared with the bottom-up methods, our method has two advantages: 1) Since the service specification of the internetworking system is used explicitly, the constructed system satisfies the design requirements specified by Sc; it is not necessary to validate the internetworking system against the service specification Sc after the converter is generated. 2) The algorithm does not need a message mapping set in advance, which can only be obtained heuristically and is difficult to validate for complex protocols. For the top-down methods reported in [9] and [4], a service adapter may not exist. Hence the application of the methods is limited. Compared with the method developed in [2], our method has two advantages: 1) the service specification Sc of the internetworking system may be nondeterministic; 2) the constructed converter is optimized. In addition, the complexity of Algorithm-OI is comparable with that of Algorithm-GI(or the algorithm proposed in [2]), since the computation of A2', B1', and A2'*C*B1' is less complex in comparison with the computation of C from M0 and Sc.

### References


[1] E. Brinksma, "A Theory for the Derivation of Tests", in Proceedings of IFIP Workshop PSTV, 1988.

[2] K. L. Calvert, et al, "Deriving a Protocol Converter: A Top-Down Method", Proceedings of ACM SIGCOMM '89.

[3] P. E. Green, JR, "Protocol Conversion", IEEE Transaction on Communications, Vol. 34, No. 3, March 1986.

[4] D. M. Kristol, et al, "Efficient Gateway Synthesis from Formal Specifications", Proceedings of ACM SIGCOMM'91. Also see "A Polynomial Algorithm for gateway Generation from Formal Specification", IEEE/ACM Trans. on Networking, Vol.1, No.2, April, 1993.

[5] S. S. Lam, et al, "Protocol Verification via Projections", IEEE Trans. Software Eng., Vol. 10, July 1984.

[6] S.S. Lam, "Protocol Conversion", IEEE Trans. Software Eng., Vol. 14, Mar.1988.

[7] Rocco De Nicola, "Extensional Equivalencies for Transition Systems", Acta Informatica 24, 1987.

[8] K. Okumura, "A Formal Protocol Conversion Method", Proceedings of ACM SIGCOMM '86.

[9] K. Okumura, "Generation of Proper Adapters and Converters from a Formal Service Specification", Proceedings of IEEE INFOCOM '90.

[10] Parrow, J. "Submodule Construction as Equation Solving in CCS", Theoretical Computer Science, 68, 1989.

[11] Robin Milner, *Communication and Concurrency,* Prentice Hall, 1989.

[12] M. Rajagopal, et al, "Synthesizing a Protocol Converter from Executable Protocol Traces", IEEE Transactions on Computers, Vol. 40, No. 4, April 1991.

[13] J.C. Shu, and M.T. Liu, "A Synchronization Model for Protocol Conversion", In Proc. IEEE INFOCOMM '89, Ottawa, Canada, 1989.

[14] J.C. Shu, and M.T. Liu, "An Approach to Indirect Protocol Conversion", Computer Networks and ISDN Systems 21, 1991.

[15] 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.

[16] Harry R. Lewis, *Elements of the Theory of Computation,* Printice-Hall, 1981, pp59-pp62.

[17] P.J.G. Ramadge and W.M. Wonham "The Control of Discrete Event Systems" Proceedings of the IEEE, Vol.77, No.1, Jan. 1989.



* This work is supported by the IDACOM-NSERC-CWARC Industrial Research Chair on Communication Protocols.