An Incremental Approach to Reachability Analysis of Distributed Programs with Synchronous Communication

K. C. Tai

Center for Communications and Signal Processing
Department of Computer Science
North Carolina State University

TR-92/20
November 1992
An Incremental Approach to Reachability Analysis of
Distributed Programs with Synchronous Communication

November 1, 1992

K. C. Tai
Department of Computer Science
Box 8206
North Carolina State University
Raleigh, NC 27695–8206, USA

Tel: (919) 515–7146
E-mail: kct@csc.ncsu.edu
Abstract

Each process in a distributed program with synchronous communication can be modeled as a communicating finite state machine (CFSM). For a set M of CFSMs, the reachability analysis of M is to derive a CFSM describing the external behavior of M and verify safety properties such as freedom from deadlocks or livelocks. The conventional approach to reachability analysis of M is to compose all CFSMs in M at the same time and derive all reachable states of M. This approach is impractical for analyzing large distributed programs.

In this paper we apply the theory of CCS (calculus of communicating systems) to define an incremental strategy for reachability analysis. A set of CFSMs is organized into a hierarchy. We present an algorithm that, for a given hierarchy of a set M of CFSMs, incrementally composes and reduces subsets of CFSMs in M and finally produces a minimum CFSM describing the external behavior of M. We also show that this incremental reachability analysis guarantees the detection of deadlocks and may also detect livelocks.

Our incremental strategy for reachability analysis is practical for analyzing large distributed programs and can be easily incorporated into an incremental development of distributed programs. Furthermore, if some components of a distributed program are modified or replaced, the effort for re-analysis can be significantly reduced by applying this incremental approach.

Keywords: distributed programs, reachability analysis, incremental analysis, deadlock detection
1. Introduction

In this paper we consider distributed programs using synchronous message-passing constructs, including blocking send/receive, remote procedure calls, and rendezvous [And91]. The reachability analysis of a distributed program P is to identify the external behavior of P and verify safety properties, such as freedom from deadlocks or livelocks, without having P's specification. The external behavior of P can also be used to determine the satisfaction of P's specifications [CES86].

Each process in a distributed program can be modeled as a communicating finite state machine (CFSM). For a set M of CFSMs, the conventional approach to reachability analysis, referred to as the all-at-once approach, is to compose all CFSMs in M at the same time and construct the reachability graph (RG) of M, which contains all reachable states of M. The number of states in the RG of M may be as high as the product of the numbers of states of individual CFSMs in M. This state explosion problem makes the all-at-once approach impractical for analyzing large distributed programs.

Let \( M = \{M_1, M_2, ..., M_n\} \), \( n>0 \), be a set of CFSMs, where \( M_i \), \( 0<i\leq n \), denotes a CFSM. The CFSMs in M can be organized into a hierarchy, which defines a hierarchical structure of M1, M2, ..., and Mn. For example, ((M1,M3), M4, (M2,M5)) is a hierarchy of M1 through M5. If M is derived from a distributed program, the structure of this program provides information for defining a hierarchy of M. Also, the interaction among CFSMs in M can be used to define a hierarchy of M.
In this paper we present an algorithm that, for a given hierarchy of a set \( M \) of CFSMs, incrementally composes and reduces subsets of CFSMs in \( M \) and finally produces a minimum CFSM describing the external behavior of \( M \). We also show that this incremental reachability analysis guarantees the detection of deadlocks and may also detect livelocks. The organization of this paper is as follows. Section 2 gives basic definitions. Section 3 presents our incremental approach to reachability analysis. Section 4 shows how to detect deadlock and livelock during incremental reachability analysis. Section 5 provides empirical results. Section 6 concludes this paper with a comparison of different strategies for incremental reachability analysis.

2. Preliminaries

A send (receive) command references a communication channel for sending (receiving) messages. There are three basic types of channel naming:

- **direct-naming**: the destination (source) of a send (receive) command is a process. Thus a pair of sender and receiver defines a channel.

- **port-naming**: the destination (source) of a send (receive) operation is a port (or entry), which has a unique receiver. Each port defines a channel.

- **mailbox-naming**: the destination (source) of a send (receive) operation is a mailbox, which may have any process as a sender or receiver. Each mailbox defines a channel.

A remote procedure call or a rendezvous can be expressed in terms of blocking send and receive as follows:

- A call to procedure or port \( T \) of process \( P \) can be represented as
  
  ```plaintext
  send ... to P.T;  
  receive ... from P;  
  ```
A procedure $T$ (for remote procedure calls) or an accept for port $T$ (for rendezvous) can be represented as

\[
\text{receive} \ldots \text{from } T \text{ with } Q; \quad (Q \text{ is a variable with a process name as its value.})
\]

\[
\ldots \text{(details of the acceptance operation)}
\]

\[
\text{send} \ldots \text{to } Q;
\]

The above receive command is of type \textit{port-naming with receive-any}, which is the same as port-naming except that the receive command has an additional variable, which is assigned to be the name of the sender when the command is executed. The above send command is of type \textit{direct-naming send with dynamic binding}, which has a process variable for its destination.

For the sake of simplicity, we consider only blocking send and receive commands in the remainder of this paper.

Let (channel name, message) be denoted as an \textit{event}. For an event $u$, a \textit{send operation} for $u$ is denoted as "$u"$, a \textit{receive operation} for $u"+u"$, and a \textit{synchronization operation} for $u"*u"$.

A CF$SM$ is a 5-tuple $(\Sigma, V, \sigma, s, t)$, where $\Sigma$ consists of send, receive, and synchronization operations, $V$ is a finite set of states, $\sigma$ is a nondeterministic state transition function and maps a state in $(V-t)$ and a element in $\Sigma$ into a subset of $V$, $s$ is the initial state, and $t$ is a set of final states, each indicating a termination of the CF$SM$. In this paper, a CF$SM$ $(\Sigma, V, \sigma, s, t)$ is represented as a directed graph $(V, E)$, where $E$ is the set of transitions, each labeled by a send, receive, or synchronization operation. A transition labeled by a send (receive, synchronization) operation is referred to as a send (receive, synchronization) transition. A transition of a state refers to a transition leaving this state.
Fig. 1 shows three CFSMs using direct-naming only. A transition labeled by "-(i,j,m)" indicates a send operation with Mi as the sender, Mj the receiver, and m the message. Similarly, a transition labeled by "+(i,j,m)" indicates a receive operation with Mi as the sender, Mj the receiver, and m the message. Also, a transition labeled by "*(i,j,m)" indicates a synchronization between Mi and Mj with delivery of message m from Mi to Mj.

Let M be a set of CFSMs. A channel in M is referred to as an internal channel in M if it involves CFSMs in M, but not CFSMs outside M. Let L a set of channels. RG(M,L) denotes the reachability graph of M with each channel in L that involves some CFSMs in M as an internal channel. (L may contain channels not involving any CFSM in M.) More precisely, RG(M,L) is the minimum CFSM that defines the set of sequences of (1) synchronization operations among CFSMs in M and (2) send and receive operations of CFSMs in M that involve channels not in L. Thus, RG(M,\{\}) is the minimum CFSM that defines the set of sequences of send, receive, and synchronization operations of CFSMs in M. RG(M,L) can be viewed as RG(M,\{\}) modified by deleting send and receiver transitions involving channels in L and then reducing it to a minimum CFSM. Let RG(M) denote the reachability graph of M with each channel in M as an internal channel. Thus, every transition in RG(M) is a synchronization transition.

If M uses direct-naming only, a channel in M involves two specific CFSMs and thus the set of internal channels of M can be determined directly from M. As an example, for two CFSMs M1 and M2 communicating with each other by using direct-naming, \{(1,2),(2,1)\} is the set of internal channels. Fig. 2 shows RG((M1,M2),\{(1,2),(2,1)\}), where M1 and M2 are defined in Fig. 1. If M uses port- or mailbox-naming, a port or mailbox may involve CFSMs outside M.
and thus the set of internal channels of M can be determined from M only if the set of channels communicating with CFSMs outside M is specified.

Let \( \text{EXT}_\text{RG}(M,L) \) be the minimum CFSM that is observational equivalent to \( \text{RG}(M,L) \). Since the definition of observational equivalence between two CFSMs is very complicated [Mil89], it is not given here. An alternative definition is that \( \text{EXT}_\text{RG}(M,L) \) is \( \text{RG}(M,L) \) modified by deleting synchronization transitions and then reducing it to a minimum, observational equivalent CFSM. Informally, \( \text{EXT}_\text{RG}(M,L) \) is \( \text{RG}(M,L) \) modified by deleting synchronization transitions and then reducing it to a minimum CFSM. Thus, \( \text{EXT}_\text{RG}(M,L) \) is the minimum CFSM that has the same external behavior as \( \text{RG}(M,L) \). If \( \text{RG}(M,L) \) contains only synchronization transitions, then \( \text{RG}(M,L) \) has empty external behavior and \( \text{EXT}_\text{RG}(M,L) \) contains only one state and no transitions. Since \( \text{RG}(M) \) contains only synchronization transitions, \( \text{EXT}_\text{RG}(M) \) contains only one state and no transitions. In this paper, "\( \equiv \)" denotes observational equivalence between two CFSMs.

A hierarchy of CFSMs denotes a set of CFSMs with a hierarchical structure. Formally, a hierarchy \( H \) of CFSMs \( M_1, M_2, \ldots \) and \( M_n, n \geq 1 \), is denoted as

\[
(H_1, H_2, \ldots, H_m)
\]

where \( m \geq 0 \), each \( H_j, 0 \leq j \leq m \), is either a hierarchy or \( M_i \) for some \( 0 \leq i \leq n \), and each \( M_i, 0 \leq i \leq n \), occurs only once in \( H \). Each \( H_j, 0 \leq j \leq m \), is said to be a component of \( H \). \( H \) is said to be sub-hierarchy of itself. For each \( H_j, 0 \leq j \leq m \), that is a hierarchy, (1) \( H_j \) is said to be a sub-hierarchy of \( H \), (2) a sub-hierarchy of \( H_j \), if it exists, is also said to be sub-hierarchy of \( H \), and (3) \( H \) is said to be the parent-hierarchy of \( H_j \). For examples, \( ((M_1,M_2),M_3,M_4) \), \( (M_1,(M_2,M_4),M_3) \), \( (((M_1,M_3),M_4),M_2) \), and \( ((M_1,M_4),(M_2,M_3)) \) are possible hierarchies of \( M_1 \).
through M4. (((M1,M3),M4),M2) has (M1,M3), ((M1,M3),M4), and (((M1,M3),M4),M2) as its sub-hierarchies.

The level of a hierarchy $H$ of CFSMs with respect to itself is one. For each sub-hierarchy of $H$, its level with respect to $H$ is one plus the level of its parent-hierarchy with respect to $H$. Thus, the levels of (M1,M3) and ((M1,M3),M4) with respect to (((M1,M3),M4),M2) are 3 and 2, respectively. The depth of a hierarchy $H$ of CFSMs, or $D(H)$, is defined as the maximum level of sub-hierarchies of $H$. A top-down traversal of a hierarchy $H$ of CFSMs is to visit $H$ first, then the level 2 hierarchies of $H$, and so on. A bottom-up traversal of a hierarchy $H$ is to visit the level $D(H)$ sub-hierarchies of $H$, then the level $(D(H)-1)$ sub-hierarchies of $H$, and so on.

3. An Incremental Approach to Reachability Analysis

For a set $M$ of CFSMs with a given set of internal channels, a CFSM describing the external behavior of $M$ can be obtained by applying the all-at-once approach to construct $RG(M,INT_C(M))$ and then reduce it to a minimum CFSM. This approach is very time- and space-consuming. An alternative is to define a hierarchy of $M$ and incrementally compose and reduce the CFSMs in $M$ according to this hierarchy. In this section, we present an algorithm for this incremental approach.

We first illustrate this incremental approach by considering the hierarchy ((M1,M2),M3), where M1, M2, and M3 are the CFSMs shown in Fig. 1. Since M1, M2 and M3 use direct-naming, the determination of internal channels for incremental analysis is easy. We first compose M1 and M2 with (1,2) and (2,1) as internal channels. The resulting CFSM is $RG(((M1,M2),{(1,2),(2,1)}),\ldots$
which is shown in Fig. 2. Then we derive the minimum CFSM that has the same external behavior as \( RG((M1,M2),\{(1,2),(2,1)\}) \). This minimum CFSM, shown in Fig. 3, is \( EXT\_RG((M1,M2),\{(1,2),(2,1)\}) \). The next step is to compose the CFSM in Fig. 3 and \( M3 \) with \( \{(1,3),(2,3),(3,1),(3,2)\} \) as the set of internal channels. The resulting CFSM of this composition is given in Fig. 4. Since the CFSM in Fig. 4 has synchronization transitions only, its minimum, observational equivalent CFSM is \( () \).

**Algorithm INC_RA for Incremental Reachability Analysis**

Let \( H \) be a hierarchy of a set \( M \) of CFSMs \( M1, M2, ..., \) and \( Mn, n>1 \). Assume that if \( M \) uses port- or mailbox-naming, then \( INT\_C(M) \), the set of internal channels of \( M \), is given. For the sake of simplicity, in step (1) it is assumed that \( M \) uses either direct-naming only or port/mailbox-naming only. If \( M \) uses both direct-naming and port/mailbox-naming, then step (1) of this algorithm needs to be modified slightly.

(1) For each sub-hierarchy \( B \) of \( H \), let \( INT\_C(B) \) be defined as \{ channels involving two or more components of \( B \), but not any CFSM outside \( B \) \}, which is constructed as follows:

Case (a): \( M \) uses direct-naming only.

(a.1) For each \( Mi, ,i<=n \), let \( USE\_C(Mi) = \{ (sender,receiver) : \text{the sender or receiver is in } Mi \} \). Let \( INT\_C(M) \) be the empty set.

(a.2) Perform a bottom-up traversal of \( H \) to visit sub-hierarchies of \( H \). For a sub-hierarchy \( B \), let

\[
INT\_C(B) = \{ (sender,receiver) : \text{the sender and receiver are in two different USE\_C sets of components of } B \}
\]

and
\[ \text{INT}\_C(M) = \text{INT}\_C(M) \cup \text{INT}\_C(B). \]

Case (b): M uses port- or mailbox-naming only.

(b.1) Perform a bottom-up traversal of \( H \) to visit sub-hierarchies of \( H \). For each sub-hierarchy \( B \) that is not \( H \), let \( \text{USE}\_C(B) = \{ \text{ports or mailboxes used in } B \} \)

(b.2) Let \( \text{INT}\_C(H) = \text{INT}\_C(M) \cap \{ \text{ports or mailboxes in at least two } \text{USE}\_C \text{ sets of components of } H \} \) and \( \text{LEFT}\_C(H) = \text{INT}\_C(M) - \text{INT}\_C(H) \).

(b.3) Perform a top-down traversal of \( H \) to visit sub-hierarchies of \( H \) that are not \( H \). For each \( B \) of these sub-hierarchies, let \( B^* \) denote the parent-hierarchy of \( B \). Define \( \text{INT}\_C(B) = \text{LEFT}\_C(B^*) \cap \{ \text{ports or mailboxes in at least two } \text{USE}\_C \text{ sets of components of } B \} \) and \( \text{LEFT}\_C(B) = \text{LEFT}\_C(B^*) - \text{INT}\_C(B) \).

(2) For each CFSM \( M_i, i > 0, \) in \( M \), let \( \text{INT}\_C(M_i) \) be the empty set and let \( \text{EXT}\_RG(M_i,\text{INT}\_C(M_i)) \) be \( M_i \).

(3) Perform a bottom-up traversal of \( H \) to visit sub-hierarchies of \( H \). For a sub-hierarchy \( B \), where \( B = (B_1, B_2, \ldots, B_m), m > 1, \)

let \( \text{RG}(B,\text{INT}\_C(B)) = \text{RG}( ( \text{EXT}\_RG(B_1,\text{INT}\_C(B_1)), \text{EXT}\_RG(B_2,\text{INT}\_C(B_2)), \ldots, \text{EXT}\_RG(B_m,\text{INT}\_C(B_m)) ), \text{INT}\_C(B) ) \).

Let \( \text{EXT}\_RG(B,\text{INT}\_C(B)) \) be the minimum CFSM that is observational equivalent to \( \text{RG}(B,\text{INT}\_C(B)) \).

When algorithm \text{INC}\_RG terminates, \( \text{RG}(H,\text{INT}\_C(H)) \) and \( \text{EXT}\_RG(H,\text{INT}\_C(H)) \) have been constructed. Since \( \text{INT}\_C(M) \) denotes the set of internal channels for all of CFSMs in \( H \), \( \text{RG}(H,\text{INT}\_C(H)) \) and \( \text{EXT}\_RG(H,\text{INT}\_C(H)) \) are also referred to as \( \text{IRG}(H,\text{INT}\_C(M)) \) and
EXT_IRG(H,INT_C(M)), respectively, where "IRG" stands for "incremental reachability graph". Below we show that EXT_IRG(H,INT_C(M)) is the minimum CFSM describing the external behavior of M. Thus, algorithm INC_RG is more efficient than the all-at-once approach for identifying the external behavior of a set of CFSMs. The following lemma is derived from the restriction laws of CCS (on page 80 of [Mil89]) and is needed for the proof of theorem 2.

**Lemma 1.** Let M and M" be sets of CFSMs and L a set of channels.

(a) \( RG(M,L) = RG(M,(L U L")) \), where L" is a set of channels not involving any CFSMs in M.

(b) \( RG((RG(M,L),RG(M"),L)),L*) = RG((M,M"),(L U L*)) \), where L and L* are disjoint and no channel in L involves CFSMs in both M and M".

**Theorem 2.** Let H be a hierarchy of a set M of CFSMs M1, M2,..., and Mn, n>1. Assume that if M uses port- or mailbox naming, then INT_C(M), the set of internal channels of M, is given. Assume that algorithm INC_RG has been applied to H.

1. IRG(H,INT_C(M)) is the minimum CFSM that defines the set of sequences of (1) synchronization operations involving components of H and (2) send and receive operations of CFSMs in M that involve channels not in INT_C(M).

2. \( IRG(H,INT_C(M)) = RG(M,INT_C(M)) \) and \( EXT_IRG(H,INC_(M)) = EXT_RG(M,INT_C(M)) \).

Proof. The proof for the theorem is by induction on the depth of a hierarchy. For a hierarchy with depth being 1, the theorem is obviously true. Assume that the theorem is true for hierarchies with depth being n, n>=1. Let H be a hierarchy of M with depth being n+1, and H=(H1,H2,...,Hm), m>1. Let Ni, 0<i<=m, be the set of CFSMs included in Hi and INT_C(Ni) be the set of internal channels in Ni according to algorithm INC_RA. Then (a) M is the union
of N1, N2, ..., and Nm, and (b) INT_C(N1), INT_C(N2), ..., INT_C(Nm), and INT_C(H) are mutually exclusive and the union of them is INT_C(M). (1) is true because INT_C(H) is disjoint from any of INT_C(N1), INT_C(N2), ..., and INT_C(Nm). Note that if an element was also an element of INT_C(Mi) for some i, 0<i<=m, then some synchronization operations involving components of H could be missing in IRG(H,INT_C(M)). The proof for (2) is as follows. By definition, IRG(H,INT_C(M)) = RG(H,INT_C(H)). According to step (3) of algorithm INC_RG,

IRG(H,INT_C(M)) = RG( ( EXT_RG(H1,INT_C(H1)), EXT_RG(H2,INT_C(H2)), ..., 

EXT_RG(Hm,INT_C(Hm) ),INT_C(H)).

For 1<=i<=m, EXT_RG(Hi,INT_C(Hi)) = EXT_IRG(Hi,INT_C(Ni)) == IRG(Hi,INT_C(Ni)) == RG(Ni,INT_C(Ni)). Thus,

IRG(H,INT_C(M)) == RG( ( RG(N1,INT_C(N1)), RG(N2,INT_C(N2)), ..., 

RG(Nm,INT_C(Nm) ),INT_C(H)).

Let INT_C" be the union of INT_C(N1), INT_C(N2), ..., and INT_C(Nm). Following lemma 1(a), we can replace each INT_C(Ni), 0<i<=m, in the right-hand side of the above equation with INT_C". Hence,

IRG(H,INT_C(M)) == RG((RG(N1,INT_C"),RG(N2,INT_C"),...,RG(Nm,INT_C")),INT_C(H)).

Note that INT_C" and INT_C(H) are disjoint and no channel in INT_C" involves CFSMs in two or more of N1, N2, ..., and Nm. Following lemma 1(b), we have

IRG(H,INT_C(M)) == RG((N1,N2,...,Nm),INT_C(M)) = RG(M,INT_C(M)). Since IRG(H,INT_C(M)) and RG(M,INC_(M)) are observational equivalent, their minimum, observational equivalent CFSMs are equal and therefore EXT_IRG(H,INT_C(M)) = EXT_RG(H,INC_(M)).

Q.E.D.
4. Deadlock and Livelock Detection in Incremental Reachability Analysis

In this section, we show that incremental reachability analysis using algorithm INC_RG guarantees the detection of deadlocks and may also detect livelocks. Let M be a set of CFSMs and L a set of channels. A state in RG(M,L) is called a deadlock state if (1) S has no successor state, (2) S is not a final state (i.e., at least one CFSM in M is not in a final state when M is in state S), and (3) at least one path from the initial state of RG(M,L) to S contains only synchronization transitions. Condition (3) is required because if every path from the initial state to S contains some send or receive transitions involving channels not in L, then whether S will be entered depends upon the interaction between M and other CFSMs. A deadlock state of M is said to involve a CFSM in M if this CFSM is not in a final state when M is in this deadlock state. Since RG(M) contains synchronization transitions only, a deadlock state in RG(M) is a state that has no transitions and is not a final state. Assume that in Fig. 1, the label associated with the transition from state S33 to state S34 is changed from "+(2,3,e)" to "+(2,3,g)". Then the state (S14,S23,S33) in Fig. 4 becomes a deadlock state.

Since observational equivalence is used in algorithm INC_RG and in the proof for theorem 2, one important question is whether two observational equivalent CFSMs have the same result on deadlock. Consider RG(M,L) and EXT_RG(M,L). If RG(M,L) contains some send or receive transitions involving channels not in L, then RG(M,L) contains a deadlock state if and only if EXT_RG(M,L) contains a deadlock state. (The deadlock state in EXT_RG(M,L) is entered from the initial state through the special symbol "e", denoting an internal operation.) If RG(M,L) has only synchronization transitions, then RG(M,L) has empty external behavior and EXT_RG(M,L) contains only one state and no transitions. The only state in EXT_RG(M,L) satisfies the definition
of a deadlock state, although RG(M,L) does not necessarily contain a deadlock state. Thus, two
observational equivalent CFSMs have the same result on deadlock only if they have non-empty
external behavior. From theorem 2 and the above discussion, we have the following theorem.

Theorem 3. Let H be a hierarchy of a set M of CFSMs M1, M2,..., and Mn, n>1. Assume that
if M uses port- or mailbox naming, then INT_C(M), the set of internal channels of M, is given.
(a) IRG(H,INT_C(M)) contains a deadlock state if and only if RG(M,INT_C(M)) contains a
deadlock state.
(b) If EXT_IRG(M,INT_C(M)) has non-empty external behavior, EXT_IRG(H,INT_C(M))
contains a deadlock state if and only if RG(M,INT_C(M)) contains a deadlock state.

Let M be a set of CFSMs M1, M2, ..., and Mn, n>1, and L a set of channels. A state in RG(M,L)
is said to be a livelock state if (1) S has at least one successor state, (2) at least one path from
the initial state of RG(M,L) to S contains only synchronization transitions, and (3) there exists
at least one CFSM Mi, 0<i<=n, such that (a) Mi has not entered its final state and (b) each of
S and its reachable states has no transitions involving Mi (i.e., Mi cannot make any progress after
entering state S). A livelock state S of M is said to involve a CFSM if each of S and its
reachable states has no transitions involving this CFSM. Following the definitions of deadlock
and livelock states, we have the following theorem.

Theorem 4. Let H be a hierarchy of a set M of CFSMs M1, M2,..., and Mn, n>1. Assume that
if M uses port- or mailbox naming, then INT_C(M), the set of internal channels of M, is given.
Let B be a sub-hierarchy of H that is not H and let M" be the set of states in B. If
EXT_IRG(B,INT_C(M")) has non-empty external behavior and contains a deadlock state, then
RG(M, INT_C(M)) contains a livelock or deadlock state involving all CFSMs in M.

Consider the deadlock example used earlier. If the modified CFSM in Fig. 4 is combined with other CFSMs, the resulting CFSM contains a livelock or deadlock state involving M1, M2, and M3 and this state is entered from the initial state via the sequence of transitions labeled "*(2,3,b)", "*(1,3,c)", and "*(1,3,f)".

5. Empirical Studies of Incremental Reachability Analysis and Deadlock Detection

We have compared algorithm INC_RG and the all-at-once approach by applying them to the following six concurrent programs written in CCS:

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABP</td>
<td>Alternating Bit Protocol (two versions)</td>
</tr>
<tr>
<td>CSMA</td>
<td>Carrier Sense/Multiple Access</td>
</tr>
<tr>
<td>GS</td>
<td>Gas Station Problem (with one operator, one pump, and three customers) [Hel85]</td>
</tr>
<tr>
<td>DME</td>
<td>Dekker's Mutual Exclusion Algorithm</td>
</tr>
<tr>
<td>PME</td>
<td>Peterson's Mutual Exclusion Algorithm</td>
</tr>
</tbody>
</table>

For each of these CCS programs, say M, we performed the following tasks:

(a) defined a hierarchy, let INT_C(M) = \{direct-naming channels involving only CFSMs in M and ports or mailboxes used in M\}, and applied algorithm INC_RG.

(a.1) determined the numbers of states in the resulting EXT_IRG, which is the same as that of EXT_RG. This number is referred to as #EXT.
(a.2) determined the maximum of total number of states at any time during the execution of algorithm INC_RG. This number is referred to as \#IRG_max.

Since the value of \#IRG_max depends upon the hierarchy chosen, we considered several possible hierarchies and selected the one with the smallest value of \#IRG_max.

(b) let \(\text{INT}_C(M) = \{\text{direct-naming channels involving only CFSMs in } M \text{ and ports or mailboxes used in } M\}\), and applied the all-at-once approach to determine the number of states in \(\text{RG}(M, \text{INT}_C(M))\). This number is referred to as \#RG_closed.

(c) let \(\text{INT}_C(M) = \{\}\), and applied the all-at-once approach to determine the number of states in \(\text{RG}(M, \{\})\). This number is referred to as \#RG_open.

These CCS program were analyzed by using Concurrency Workbench [CPS91] to determine the values of \#EXT, \#IRG_max, \#RG_closed, and \#RG_open, as well as the existence of deadlocks. The results of this analysis are shown in the following table. The "n/r" entry in the table indicates "no result" because the analysis for the entry did not terminate after two days of execution.

<table>
<thead>
<tr>
<th>Program</th>
<th>#EXT</th>
<th>#IRG_max</th>
<th>#RG_closed</th>
<th>#RG_open</th>
<th>Deadlock?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABP-I</td>
<td>9</td>
<td>40</td>
<td>41</td>
<td>180</td>
<td>Yes</td>
</tr>
<tr>
<td>ABP-II</td>
<td>2</td>
<td>12</td>
<td>12</td>
<td>180</td>
<td>No</td>
</tr>
<tr>
<td>CSMA</td>
<td>12</td>
<td>31</td>
<td>36</td>
<td>560</td>
<td>No</td>
</tr>
<tr>
<td>GS</td>
<td>1</td>
<td>14</td>
<td>14</td>
<td>n/r</td>
<td>Yes</td>
</tr>
<tr>
<td>DME</td>
<td>2</td>
<td>114</td>
<td>126</td>
<td>800</td>
<td>No</td>
</tr>
<tr>
<td>PME</td>
<td>2</td>
<td>31</td>
<td>31</td>
<td>288</td>
<td>No</td>
</tr>
</tbody>
</table>

For ABP-II, GS, and PME, the hierarchy chosen for incremental analysis is exactly the all-at-once approach and thus the value of \#IRG_max is the same as that of \#RG_closed. The reason is that during an intermediate stage of incremental analysis, the number of states may be larger than the value of \#RG_closed. This indicates that the selection of a hierarchy for a set of CFSMs is critical.
The above six programs are too small to show a significant difference between the value of #IRG_max and that of #RG_closed. We are currently using larger programs for empirical studies and studying how to define a hierarchy for a set of CFSMs in order to produce a "minimum" or "near minimum" values of #IRG_max.

Note that if a CCS program uses direct- or port-naming, the value of #RG_open in the above table is larger than necessary. The reason is that each action name in a CCS program is viewed as a mailbox and the Concurrency Workbench does not attempt to determine whether each action name in a CCS program can be viewed as a direct-naming channel or a port. Consider the set M of CFSMs M1 and M2 using direct-naming. RG(M, {}) contains send and receive transitions involving channels (1,2) and (2,1). Such transitions (and possibly some states) should be deleted since they are redundant. After the deletion of redundant transitions and states in RG(M, {}), the value of #RG_open is reduced to that of #RG_closed. Similarly, consider a set M" of CFSMs using port-naming with port R associated with a CFSM in M". RG(M", {}) contains send transitions involving port R and such transitions (and possibly some states) are redundant.

6. Conclusion

In this paper we have described, for a given hierarchy of CFSMs, an incremental approach to reachability analysis. We have also shown that this incremental reachability analysis guarantees the detection of deadlocks and may also detect livelocks. This incremental approach is practical for analyzing large distributed programs. During an incremental development of a large distributed program, incremental reachability analysis can be incorporated into the development process.
Assume that a module M of a distributed program P is modified or replaced. If module M has the same EXT_IRG as its original version, then there is no need to re-analyze P for verification of properties such as freedom from deadlock or livelock. If module M has a different EXT_IRG, then portions of P that are affected by module M can be incrementally re-analyzed in bottom-to-top order. The re-analysis of P stops at a sub-hierarchy of P if the EXT_IRG of this sub-hierarchy is not changed due to the changes in module M. Thus, incremental analysis can significantly reduce the effort for re-analysis of a large distributed program due to correction or enhancement of this program.

Incremental reachability analysis of a set of CFSMs with synchronous communication has been studied by some researchers. In [LSU89] algorithms for incremental composition and reduction of CFSMs with direct-naming were given. At each composition step, a pair of CFSMs were combined. No quantitative way of ranking CFSMs for incremental composition was used. The paper presented three heuristic rules for reducing a CFSM to a smaller, observational equivalent one. However, the three rules do not necessarily produce a minimum, observational equivalent CFSM. The paper claimed that incremental analysis of the Q.931 protocol produced less than 1,000 states, while the all-at-once approach produced over 60,000 states. It is not clear, however, that the reachability graph produced by the all-at-once approach contains redundant transitions and states as mentioned in section 5.

In [SKB90] an algorithm for incremental composition of CFSMs with direct-naming was given. Each transition of a CFSM may be associated with one input (receive command) and one or more outputs (send commands). At each composition step, a pair of CFSMs were combined into one by (1) matching pairs of transitions in these two CFSMs such that an output of one transition is the
input of the other transition, and (2) keeping transitions that neither take input from nor produce output to the other CFSM. There was no discussion of whether the resulting CFSM is observational equivalent to the original set of CFSMs.

In [YeY91] a prototype tool for incremental analysis of programs written in an Ada-like design language called PAL was described. For a PAL program, the tool transforms it into a set of process graphs, one for each task, and performs composition and simplification of these process graphs. The tool can also determine the equivalence of process graphs. The axioms of composition, simplification, and equivalence used in this tool are based on ACP [BeK84], which is a different theory of process algebra from CCS and CSP [Hoa85].

Our paper is different from the above mentioned papers in several aspects. First, we define the CFSM model to include message passing constructs using direct-naming, port-naming and mailbox-naming. Second, we show how to perform incremental analysis of a given hierarchy of CFSMs with a given set of internal channels. At each step of this incremental analysis, a minimum CFSM describing the external behavior of CFSMs analyzed so far is produced. Third, we apply the theory of CCS to provide a formal proof that our incremental analysis guarantees the detection of deadlocks. Fourth, we discuss the relationship between deadlocks and livelocks in incremental analysis.
Acknowledgment

I would like to express my gratitude to Pramod Koppol for carrying out the empirical studies reported in section 5 and for his helpful comments on this paper. Also, I would like to thank Dr. Rance Cleaveland for helpful discussion on CCS and Concurrency Workbench and for providing several CCS programs used in our empirical studies. The CCS program for the gas station problem was provided by Ufuk Celikkian.

References


Figure 1: CFSMs M1, M2 and M3

Figure 2: RG((M1,M2),{(1,2),(2,1)})
Figure 3: $\text{EXT\_RG}((M_1, M_2), \{(1,2), (2,1)\})$

Figure 4: $\text{RG}((\text{EXT\_RG}((M_1, M_2), \{(1,2), (2,1)\}), M_3), \{(1,3), (2,3), (3,1), (3,2)\})$