Formal Analysis of the Alternating Bit Protocol Using Stuttering Refinement

Shiva Nejati

Department of Computer Science, University of Toronto,
Toronto, ON M5S 3G4, Canada.
Email: `shiva@cs.toronto.edu`

1 Description

The alternating bit protocol is a technique that can be used to implement a reliable communication over faulty channels. Typically, in communication protocols, the problem is that a transmission channel may lose, duplicate or corrupt a message. Therefore, the sender cannot assume that the message has been successfully received by the receiver.

In the alternating bit protocol, the communication system consists of two lossy, unbounded and FIFO channels: one transfers messages from the *sender* to the *receiver*, i.e. `s2r` channel, and the other transfers acknowledgments from the *receiver* to the *sender*, i.e. `r2s` channel. In order for the receiving end points to distinguish between copies of the same message and copies of different message, messages and acknowledgments are sent tagged with the bits 0 and 1 alternately. During the communication, there may be an arbitrary number of copies of a message (or an acknowledgment) on channels `s2r` and `r2s`. However, it has been proved that there are at most two distinct messages (or acknowledgments) on the channels. Therefore, binary flags are sufficient for distinguishing between messages and the redundant copies. The alternating bit protocol and its components are shown in Figure 1.

Figure 2 shows a formal description of the alternating bit protocol similar to that given in [Man01]. The *sender* communicates with the channel via a register `smsg` and a flag `svalid`. The *sender* assigns a new message to `smsg` provided it is invalid, i.e. `svalid` is false. Then, it sends the content of `smsg` tagged with `sflag` across `s2r` until it receives an acknowledgment for that message on `r2s`. The *receiver* works in a dual manner. It sends acknowledgments for a received message across `r2s` until it receives a new message. Notice that the *receiver* sends the flag of a received message as the acknowledgment for that message. The *receiver* also has a flag `rvalid` that determines if the *receiver* can read the new message.

2 Generating Abstraction

In this section, we generate an abstract model of the alternating bit protocol based on stuttering abstraction definition. In order to highlight the differences between stuttering abstraction and conventional 3-valued abstraction, we make some changes in the formal description of the protocol. As mentioned earlier, on channels `s2r` and `r2s` there are at most two distinct messages. Therefore, either a channel consists of copies

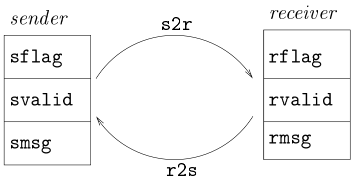

Fig. 1. Alternating bit protocol

```
(1)  ¬svalid              → smsg := m; svalid := true;
(2)  svalid               → s2r := s2r : ⟨smsg, sflag⟩;
(3)  s2r ≠ ϵ ∧ ¬rvalid    →
(4)                         if flag(head(s2r)) = rflag then
(5)                             s2r := tail(s2r);
                            else
(6)                             s2r := tail(s2r); rvalid = true;
(7)                             rflag := flag(head(s2r));
(8)                             rmsg := info(head(s2r));
(9)  r2s := r2s : ⟨rflag⟩;
(10) r2s ≠ ϵ              →
(11)                        if head(r2s) = sflag then
(12)                            r2s := tail(r2s); svalid := false;
(13)                            sflage := ¬sflag;
                            else
(14)                            r2s := tail(r2s);
(15) rvalid := false;
```

Fig. 2. The formal description of alternating bit protocol

(a) s2r: 0...00$\overset{i}{\overset{\downarrow}{1}}$...11 s2rcntr = i

(b) s2r: 0.........00

Fig. 3. Channel s2r:(a) s2r consists of copies of the old messages 0 and copies of the new message 1; (b) s2r only consists of copies of the message 0;

of one message (Figure 3(b)) or it is composed of two distinct sublists, each sublist consists of copies of one message (Figure 3(a)). In either cases, there is at most one new message for the receiver on the channels. Using this fact, we present a slightly modified description of the protocol (Figure 4). We use two integer variables s2rcntr and r2scntr to point to the position of the new message on the channels. Since there is at most one new message on the channels, one integer variable suffices to keep track of the location of the new message. When the *sender* assigns a new message to smsg, the variable s2rcntr is set to one (Figure 4 line (1)). Whenever a copy of smsg is sent across s2r, we increase s2rcntr by one. Therefore, s2rcntr points to the location of the new message on channel s2r (Figure 3(a)). Clearly, the *receiver* can deliver the new message, when all the redundant copies have been removed from s2r, or in other words, when the length of s2r is equal to s2rcntr. Therefore, the condition $\mathtt{flag}(\mathtt{head}(\mathtt{s2r})) \neq \mathtt{rflag}$ holds if and only if the length of s2r is equal to s2rcntr, i.e. $\mathtt{len}(\mathtt{s2r}) = \mathtt{s2rcntr}$ (Figure 4 line (6)). Similarly, we use variable r2scntr to distinguish between the new acknowledgment and redundant copies of the old acknowledgment.

```
(1)  ¬svalid                 → smsg := m; svalid := true; s2rcntr := 1;
(2)  svalid                  → s2r := s2r : ⟨smsg, s2rcntr⟩; s2rcntr := s2rcntr + 1;
(3)  s2r ≠ ε ∧ ¬rvalid       →
(4)                             if flag(head(s2r)) = rflag then
(5)                                s2r := tail(s2r);
(6)                             else %len(s2r) = s2rcntr
(7)                                s2r := tail(s2r); rvalid = true;
(8)                                rmsg := info(head(s2r));
(9)                                r2scntr := 1;
(10) r2s := r2s : ⟨r2scntr⟩;
(11) r2scntr := r2scntr + 1;
(12) r2s ≠ ε                 →
(13)                            if head(r2s) = sflag then
(14)                             %len(r2s) = r2scntr
(15)                               r2s := tail(r2s); svalid := false;
                                else
(16)                               r2s := tail(r2s);
(17) rvalid := false;
```

Fig. 4. The modified description of the alternating bit protocol

To generate an abstract model, we choose the following abstraction predicates:

$$\begin{aligned}
\varphi_1 &= \texttt{svalid} \\
\varphi_2 &= \texttt{rvalid} \\
\varphi_3 &= (\texttt{len}(\texttt{s2r}) \geq \texttt{s2rcntr}) \\
\varphi_4 &= (\texttt{len}(\texttt{r2s}) \geq \texttt{r2scntr}) \\
\varphi_5 &= (\texttt{rmsg} = \texttt{smsg})
\end{aligned}$$

We consider abstract boolean variables b_1, b_2, b_3, b_4, and b_5 as the boolean variables corresponding to the abstraction predicates, i.e. each boolean variable b_i represents all the concrete states satisfying predicate φ_i.

Since we are interested in 3-valued abstractions, we assume that in every abstract state, each variable b_i can be true, false, or maybe. We follow the Cartesian abstraction technique [GHJ01] to construct the abstract state graph. Each abstract state a_i can be represented as a conjunction of abstract variables $b_1 \dots b_5$, i.e. monomials. Notice that monomials do not necessarily include all the variables $b_1 \dots b_5$. The value of those variables that are not present in a monomial can be interpreted as maybe. For example, in the abstract state corresponding to the monomial $b_1 \wedge \neg b_3$, b_1 is true, b_3 is false, and b_2, b_4, and b_5 are maybe. In general, in our abstract model, each abstract state a_i can be represented as follows:

$$a_i = \bigwedge_{1 \leq i \leq 4} (b_i = l_i)$$

where $l_i \in \{\top, \bot, M\}$.

Since for each abstract variable there are three possible values, the abstract domain has potentially 3^5 states. Notice that we do not need to consider all 3^5 states because most of them are not reachable. For example, a state corresponding to $\neg b_1 \wedge b_2$ is not reachable because it is not consistent with any of the guards of the rules shown in Figure 4, i.e. in Figure 4, there is no command with a guard $\neg\texttt{svalid} \wedge \texttt{rvalid}$. Therefore, state $\neg b_1 \wedge b_2$ does not have any outgoing transitions. Given the fact that we do not have any deadlock state in this protocol, we can conclude that $\neg b_1 \wedge b_2$ is not reachable.

In order to systematically explore the abstract state graph, we have to construct the abstract transition relation. We use the following theorem to compute the abstract transitions.

Theorem 1. *Let ψ_1 and ψ_2 be boolean combinations of predicates $\varphi_1 \dots \varphi_n$, and let*

$$\texttt{pre}(\psi) = \{s \mid \forall s' \cdot R(s, s') \Rightarrow s' \models \psi\}$$

*There is an over-approximation transition between ψ_1 and ψ_2 iff the following formula is **satisfiable**:*

$$\psi_1 \wedge \neg\texttt{pre}(\neg\psi_2)$$

*and there is an under-approximation transition between ψ_1 and ψ_2 iff for some $k \in \mathbb{N}$ the following formula becomes **unsatisfiable**:*

$$\psi_1 \wedge \textstyle\bigwedge_{0 \leq i < k} \Phi_i(\psi_1, \psi_2)\ (F1)$$

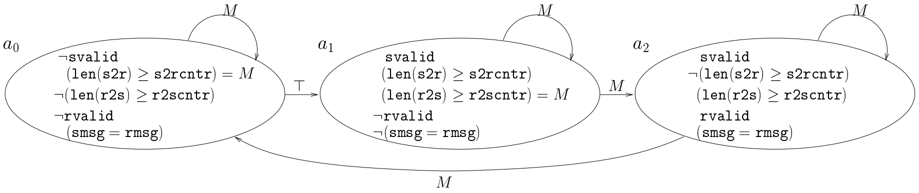

Fig. 5. 3-valued abstraction

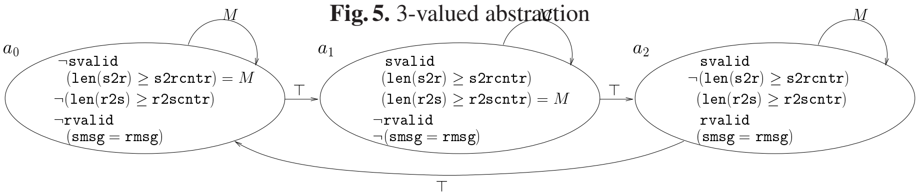

Fig. 6. Stuttering abstraction

where

$$\begin{aligned} \Phi_0(\psi_1, \psi_2) &= \mathtt{pre}(\neg\psi_2) \\ \Phi_i(\psi_1, \psi_2) &= \mathtt{pre}(\neg\psi_1 \vee \Phi_{i-1}(\psi_1, \psi_2))\ \textit{for}\ 1 \leq i < k \end{aligned}$$

The transitions of the abstract model in Figure 5 are generated based on Theorem 1, when in the formula (F1) we set $k = 1$. However, a more conclusive abstraction can be generated if in (F1) we set k to a greater value or even if we do not put any bound on k. Notice that if we do not bound the variable k then (F1) cannot be automatically decided by a theorem prover. However, in general, it is not hard to manually prove that the formula (F1) is unsatisfiable for some unbouded k. For example, in our example, the unsatisfiability of (F1) for an unbounded k follows from the fact that the channels are finite.

Figure 6 shows an abstract model that is computed based on Theorem 1 when k is unbounded. As shown in the figures, the model in Figure 6 is more refined than the model in Figure 5. Here, we explain why the transitions (a_1, a_2) and (a_2, a_0) are M in the abstract model in Figure 5 whereas they are $\top$ in the abstract model in Figure 6.

The predicate $\psi_1 = \mathtt{svalid} \wedge \neg\mathtt{rvalid} \wedge \mathtt{len(s2r)} \geq \mathtt{s2rcntr} \wedge \neg(\mathtt{rmsg} = \mathtt{smsg})$ represents the concrete states corresponding to a_1, and $\psi_2 = \mathtt{svalid} \wedge (\mathtt{len(s2r)} < \mathtt{s2rcntr}) \wedge \mathtt{rvalid} \wedge (\mathtt{rmsg} = \mathtt{smsg})$ represents the concrete states corresponding to a_2. According to Theorem 1, there is a $\top$ transition from a_1 to a_2 iff for some k, we have:

$$\forall s \cdot s \models \psi_1 \Rightarrow s \not\models \Phi_0(\psi_1, \psi_2) \vee ... \vee s \not\models \Phi_{k-1}(\psi_1, \psi_2)\ (F2)$$

First, we compute $\neg\Phi_i(\psi_1, \psi_2)$ for $0 \leq i < k$:

- $i = 0$:

$$\begin{aligned} &\neg\Phi_0(\psi_1, \psi_2) \\ &= \neg\mathtt{pre}(\neg\psi_2) \\ &= \neg\mathtt{pre}(\neg(\mathtt{svalid} \wedge \mathtt{len(s2r)} < \mathtt{s2rcntr} \wedge \mathtt{rvalid} \wedge \mathtt{rmsg} = \mathtt{smsg})) \\ &= \mathtt{svalid} \wedge \neg\mathtt{rvalid} \wedge \mathtt{len(s2r)} = \mathtt{s2rcntr} \wedge \mathtt{len(s2r)} > 0 \wedge \neg(\mathtt{rmsg} = \mathtt{smsg}) \end{aligned}$$

Notice that a state s satisfies $\neg\mathtt{pre}(\neg\psi)$ if and only if there exists some s' such that $R(s, s')$ and $s' \models \psi$.

– $i = 1$:

$$
\begin{aligned}
&\neg\Phi_1(\psi_1, \psi_2) \\
= &\neg\mathtt{pre}(\neg\psi_1 \vee \mathtt{pre}(\neg\psi_2)) \\
= &\neg\mathtt{pre}(\neg(\psi_1 \wedge \neg\mathtt{pre}(\neg\psi_2))) \\
= &\neg\mathtt{pre}(\neg(\mathtt{svalid} \wedge \neg\mathtt{rvalid} \wedge \mathtt{len(s2r)} = \mathtt{s2rcntr} \wedge \mathtt{len(s2r)} > 0) \wedge \neg(\mathtt{rmsg} = \mathtt{smsg})) \\
= &\mathtt{svalid} \wedge \neg\mathtt{rvalid} \wedge \mathtt{len(s2r)} = \mathtt{s2rcntr} + 1 \wedge \mathtt{len(s2r)} > 1 \wedge \neg(\mathtt{rmsg} = \mathtt{smsg})
\end{aligned}
$$

– ...
– In general:

$$
\begin{aligned}
&\neg\Phi_i(\psi_1, \psi_2) \\
= &\neg\mathtt{pre}(\neg\psi_1 \vee \Phi_1(\psi_1, \psi_2)) \\
= &\neg\mathtt{pre}(\neg(\psi_1 \wedge \neg\Phi_1(\psi_1, \psi_2))) \\
= &\neg\mathtt{pre}(\neg(\mathtt{svalid} \wedge \neg\mathtt{rvalid} \wedge \mathtt{len(s2r)} = \mathtt{s2rcntr} + (\mathtt{i}-1) \wedge \mathtt{len(s2r)} > (\mathtt{i}-1)) \wedge \neg(\mathtt{rmsg} = \mathtt{smsg})) \\
= &\mathtt{svalid} \wedge \neg\mathtt{rvalid} \wedge \mathtt{len(s2r)} = \mathtt{s2rcntr} + \mathtt{i} \wedge \mathtt{len(s2r)} > \mathtt{i} \wedge \neg(\mathtt{rmsg} = \mathtt{smsg})
\end{aligned}
$$

Thus, the formula (F2) can be simplified to:

$$
\begin{aligned}
\exists k \cdot \forall s \cdot\ s \models \mathtt{svalid} \wedge \neg\mathtt{rvalid} \wedge\ & \mathtt{len(s2r)} \geq \mathtt{s2rcntr} \wedge \neg(\mathtt{rmsg} = \mathtt{smsg}) \\
\Rightarrow \quad s \models \mathtt{svalid} \wedge \neg\mathtt{rvalid} \wedge \big(\ & \mathtt{len(s2r)} = \mathtt{s2rcntr} \wedge \mathtt{len(s2r)} > 0 \quad \bigvee \\
& \mathtt{len(s2r)} = \mathtt{s2rcntr} + 1 \wedge \mathtt{len(s2r)} > 1 \quad \bigvee \\
& .... \quad \bigvee \\
& \mathtt{len(s2r)} = \mathtt{s2rcntr} + (\mathtt{k}-1) \wedge \mathtt{len(s2r)} > (\mathtt{k}-1) \\
& \big) \wedge \neg(\mathtt{rmsg} = \mathtt{smsg})
\end{aligned}
$$

Since we always have $\mathtt{s2rcntr} > 0$,

$$
\begin{aligned}
\exists k \cdot \forall s \cdot\ s \models \mathtt{svalid} \wedge \neg\mathtt{rvalid} \wedge \mathtt{s2rcntr} > 0 \wedge\ & \mathtt{len(s2r)} \geq \mathtt{s2rcntr} \wedge \neg(\mathtt{rmsg} = \mathtt{smsg}) \\
\Rightarrow \quad s \models \mathtt{svalid} \wedge \neg\mathtt{rvalid} \wedge \mathtt{s2rcntr} > 0 \wedge \big(\ & \mathtt{len(s2r)} = \mathtt{s2rcntr} \quad \bigvee \\
& \mathtt{len(s2r)} = \mathtt{s2rcntr} + 1 \quad \bigvee \\
& .... \quad \bigvee \\
& \mathtt{len(s2r)} = \mathtt{s2rcntr} + (k-1) \\
& \big) \wedge \neg(\mathtt{rmsg} = \mathtt{smsg})
\end{aligned}
$$

Therefore, there is a $\top$ transition between a_1 and a_2 if and only if:

$$
\begin{aligned}
\exists k \in \mathbb{N} \cdot \forall s \cdot\ s \models\ & \mathtt{len(s2r)} \geq \mathtt{s2rcntr} \\
\Rightarrow \quad s \models\ & \big(\mathtt{len(s2r)} = \mathtt{s2rcntr} \quad \bigvee \\
& \mathtt{len(s2r)} = \mathtt{s2rcntr} + 1 \quad \bigvee \\
& .... \quad \bigvee \\
& \mathtt{len(s2r)} = \mathtt{s2rcntr} + (k-1) \\
& \big)
\end{aligned}
$$

Since the length of the channel `s2r` is finite, the above formula is a tautology. That is the formula $(F1)$ in Theorem 1 is unsatisfiable. Therefore, there is $\top$ transition from a_1 to a_2. We can prove in a similar manner that the transition between a_2 and a_1 is a $\top$ transition. This time we use the fact that the length of `r2s` is finite.

3 Model Checking

- $AG(\mathtt{svalid} \Rightarrow A[\mathtt{svalid}\, W\ (\mathtt{smsg} = \mathtt{rmsg})])$ is true in both models.
- $AG(\mathtt{svalid} \Rightarrow E[\mathtt{svalid}\, U\ (\mathtt{smsg} = \mathtt{rmsg})])$ is true in Figure 6.
- $AG(\mathtt{rvalid} \wedge (\mathtt{smsg} = \mathtt{rmsg}) \Rightarrow A[\mathtt{rvalid} \wedge (\mathtt{smsg} = \mathtt{rmsg})\, W\ (\mathtt{smsg} = \mathtt{rmsg}) \wedge \neg\mathtt{rvalid}])$ is true in both models.
- $AG(\mathtt{rvalid} \wedge (\mathtt{smsg} = \mathtt{rmsg}) \Rightarrow E[\mathtt{rvalid} \wedge (\mathtt{smsg} = \mathtt{rmsg})\, U\ (\mathtt{smsg} = \mathtt{rmsg}) \wedge \neg\mathtt{rvalid}])$ is true in Figure 6.

4 Discussion

- Most of the previous approahces to verifying ABP prtocol simplifies the problem, (e.g. by bounding the length of the channels, etc). Here, we didn't impose any restrictions on the description of the protocol.
- Our solution is less complicated than that given in [Man01]. Our solution involves less manual efforts, and some parts of it can be done automatically. Furthurmore, in contrast to [Man01], the abstract model is constructed automatically.
- The atomic propositions of the abstract model can be 3-valued.
- In general, the reachability between two states can be automatically computed by model checking techniques and fixpoint computations. However, model checking is not applicable to unbounded systems. In this case study, we used deductive methods to compute reachablity between the states of an unbounded system.

References

[GHJ01] P. Godefroid, M. Huth, and R. Jagadeesan. "Abstraction-based Model Checking using Modal Transition Systems". In K.G. Larsen and M. Nielsen, editors, *Proceedings of 12th International Conference on Concurrency Theory (CONCUR'01)*, volume 2154 of *LNCS*, pages 426–440, Aalborg, Denmark, 2001. Springer.

[Man01] P. Manolios. *"Mechanical Verification of Reactive Systems"*. PhD thesis, University of Texas, Austin, 2001.