

Modeling Concurrent and Probabilistic Systems

Lecture 11: Extensions of the Alternating Bit Protocol

Joost-Pieter Katoen Thomas Noll

Software Modeling and Verification Group

RWTH Aachen University

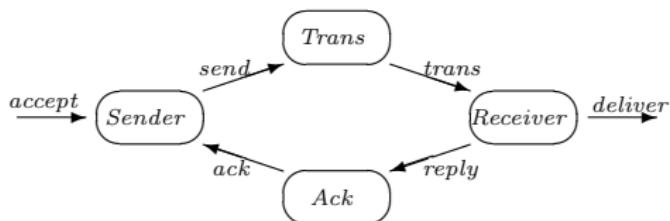
noll@cs.rwth-aachen.de

<http://www-i2.informatik.rwth-aachen.de/i2/mcps07/>

Winter Semester 2007/08

- 1 Repetition: The Alternating Bit Protocol
- 2 Handling Duplication of Messages
- 3 Concluding Remarks
- 4 Modeling Mobile Concurrent Systems

Repetition: The Alternating Bit Protocol



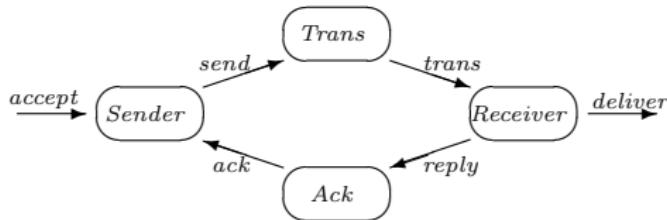
The **overall system** is given by

$$ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) = \text{new } L (\text{Sender} \parallel \text{Trans} \parallel \text{Ack} \parallel \text{Receiver})$$

where

$$L := \{ \text{send}_{db}, \text{trans}_{db}, \text{reply}_b, \text{ack}_b \mid db \in F \} \cup \{ \text{trans}_\perp, \text{ack}_\perp \}$$

Repetition: Implementation of Sender



Sender accepts $d \in D$ via $accept_d$ and repeatedly sends frames of the form $d0$ over *Trans* until it receives the acknowledgment 0 over *Ack*. For the next data item, control bit 1 is used and so on (\Rightarrow “Alternating Bit Protocol”).

Formally, for $b \in \{0, 1\}$ and $d \in D$:

$$\begin{aligned} Sender &= Sender_0 \\ Sender_b &= \sum_{d \in D} accept_d \cdot Send_{db} \\ Send_{db} &= \overline{send_{db}} \cdot Wait_{db} \\ Wait_{db} &= \underbrace{ack_b \cdot Sender_{1-b}}_{\text{successful}} + \underbrace{ack_{1-b} \cdot Send_{db}}_{\text{error}} + ack_{\perp} \cdot Send_{db} \end{aligned}$$

Repetition: Implementation of Receiver

Receiver gets frames of the form db or \perp . In the first case, if b has the expected value, d is forwarded via $deliver_d$, and b is returned via Ack . Otherwise the transmission is re-initiated by returning the “wrong” control bit $1 - b$ to *Sender*.

Formally, for $b \in \{0, 1\}$ and $d \in D$:

$$\begin{aligned} Receiver &= Receiver_0 \\ Receiver_b &= \sum_{d \in D} trans_{db}.Reply_{db} \\ &+ \sum_{d \in D} trans_{d(1-b)}. \overline{reply_{1-b}}. Receiver_b \\ &+ trans_{\perp}. \overline{reply_{1-b}}. Receiver_b \\ Reply_{db} &= \overline{deliver_d}. \overline{reply_b}. Receiver_{1-b} \end{aligned}$$

Repetition: Correctness of ABP I

Theorem

$$ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \simeq \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$$

Remark: because of internal τ -steps in ABP , $ABP \sim \text{Buffer}$ cannot hold.

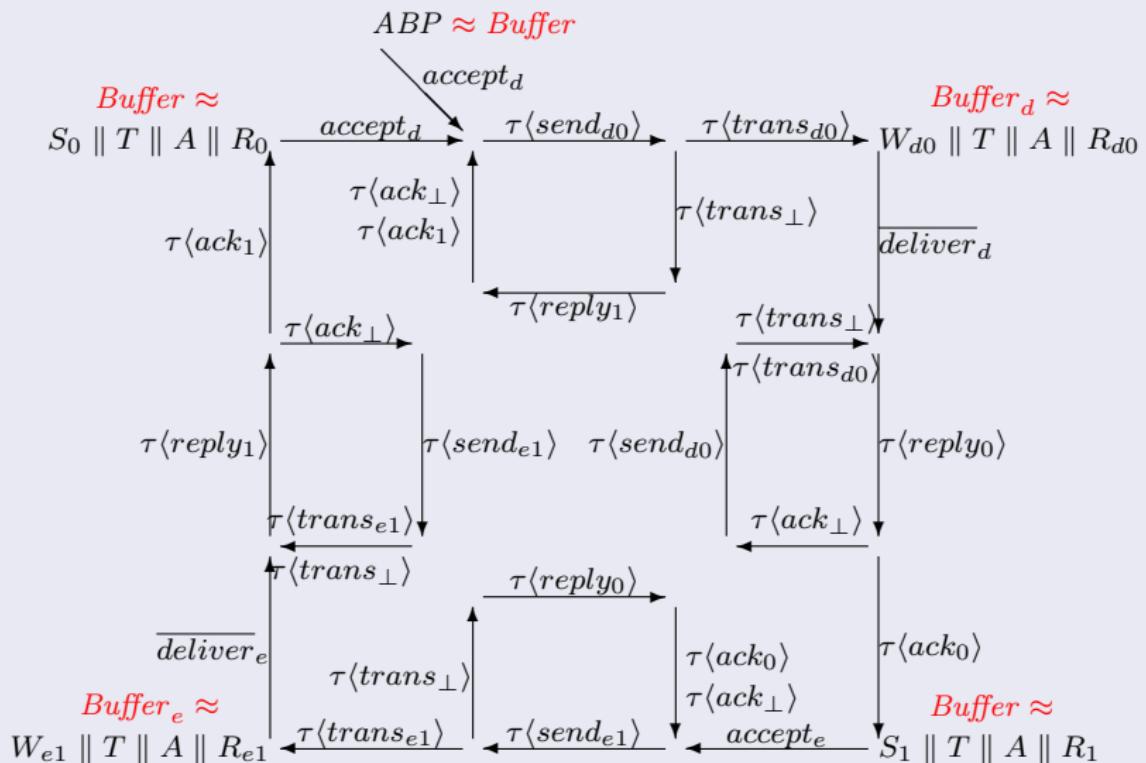
Proof.

- ① Construct transition system of $ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$
(next slide; $S = \text{Sender}$, $W = \text{Wait}$, $T = \text{Trans}$, $A = \text{Ack}$,
 $R = \text{Receiver/Reply}$, $d, e \in D$; without restrictions)
- ② Show that $ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \approx \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$
- ③ $ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \not\sim \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$
 $\implies ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \simeq \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$



Repetition: Correctness of ABP II

Proof (continued).



Repetition: Duplication of Messages

Duplication of messages can be modelled as follows:

$$\begin{aligned} Trans &= \sum_{f \in F} send_f. \underbrace{(\overline{trans}_f. Trans +}_{\text{successful}} \\ &\quad \underbrace{\overline{trans}_\perp. Trans +}_{\text{error}} \\ &\quad \underbrace{\overline{trans}_f. \overline{trans}_f. Trans)}_{\text{duplication}} \end{aligned}$$

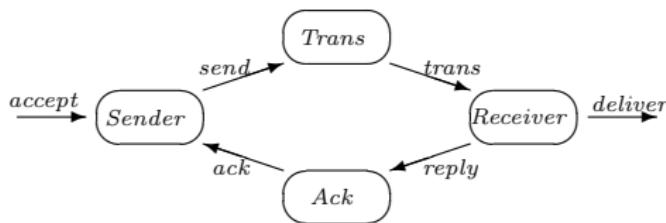
$$Ack = \sum_{b \in \{0,1\}} reply_b. \underbrace{(\overline{ack}_b. Ack + \overline{ack}_\perp. Ack)}_{\text{successful}} + \underbrace{\overline{ack}_b. \overline{ack}_b. Ack)}_{\text{error/duplication}}$$

Observation: with the original definition of *Sender* and *Receiver*, **deadlocks** are possible

- 1 Repetition: The Alternating Bit Protocol
- 2 Handling Duplication of Messages
- 3 Concluding Remarks
- 4 Modeling Mobile Concurrent Systems

- **Idea:** allow *Sender* and *Receiver* to transmit \perp frames:
 - *Receiver* $\xrightarrow{\text{reply}} \perp$: message not received
 - *Sender* $\xrightarrow{\text{send}} \perp$: acknowledgment not received
- Allows to distinguish corrupted and duplicated frames

Modified Implementation of Sender



For $b \in \{0, 1\}$ and $d \in D$:

$$\begin{aligned} \text{Sender} &= \text{Sender}_0 \\ \text{Sender}_b &= \sum_{d \in D} \text{accept}_d \cdot \text{Send}_{db} \\ \text{Send}_{db} &= \overline{\text{send}_{db}} \cdot \text{Wait}_{db} \\ \text{Wait}_{db} &= \underbrace{\text{ack}_b \cdot \text{Sender}_{1-b}}_{\text{successful}} + \underbrace{\text{ack}_{\perp} \cdot \text{Send}_{db}}_{\text{error, restart}} + \underbrace{\text{ack}_{1-b} \cdot \text{Wait}_{db}}_{\text{duplication, ignore}} \end{aligned}$$

Modified Implementation of Receiver

For $b \in \{0, 1\}$ and $d \in D$:

$$\begin{aligned} Receiver &= Receiver_0 \\ Receiver_b &= \sum_{d \in D} trans_{db}.Reply_{db} \\ &+ trans_{\perp}.\overline{reply_{\perp}}.Receiver_b \\ &+ \sum_{d \in D} trans_{d(1-b)}.Receiver_b \\ Reply_{db} &= \overline{deliver_d}.\overline{reply_b}.Receiver_{1-b} \end{aligned}$$

The Overall System

$$ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \\ = \text{new } L(\text{Sender} \parallel \text{Trans} \parallel \text{Ack} \parallel \text{Receiver})$$

$$\text{Sender} = \text{Sender}_0$$

$$\text{Sender}_b = \sum_{d \in D} \text{accept}_d \cdot \text{Send}_{db}$$

$$\text{Send}_{db} = \text{send}_{db} \cdot \text{Wait}_{db}$$

$$\text{Wait}_{db} = \text{ack}_b \cdot \text{Sender}_{1-b} + \text{ack}_\perp \cdot \text{Send}_{db} + \text{ack}_{1-b} \cdot \text{Wait}_{db}$$

$$\text{Receiver} = \text{Receiver}_0$$

$$\text{Receiver}_b = \sum_{d \in D} \text{trans}_{db} \cdot \text{Reply}_{db} \\ + \text{trans}_\perp \cdot \text{reply}_\perp \cdot \text{Receiver}_b \\ + \sum_{d \in D} \text{trans}_{d(1-b)} \cdot \text{Receiver}_b$$

$$\text{Reply}_{db} = \text{deliver}_d \cdot \text{reply}_b \cdot \text{Receiver}_{1-b}$$

$$\text{Trans} = \sum_{f \in F} \text{send}_f \cdot (\overline{\text{trans}_f} \cdot \text{Trans} + \overline{\text{trans}_\perp} \cdot \text{Trans} + \\ \overline{\text{trans}_f} \cdot \overline{\text{trans}_f} \cdot \text{Trans})$$

$$\text{Ack} = \sum_{b \in \{0,1\}} \text{reply}_b \cdot (\overline{\text{ack}_b} \cdot \text{Ack} + \overline{\text{ack}_\perp} \cdot \text{Ack} + \overline{\text{ack}_b} \cdot \overline{\text{ack}_b} \cdot \text{Ack})$$

where $L := \{\text{send}_{db}, \text{trans}_{db}, \text{reply}_b, \text{ack}_b \mid db \in F\}$
 $\cup \{\text{send}_\perp, \text{trans}_\perp, \text{reply}_\perp, \text{ack}_\perp\}$

Again:

Theorem 11.1

$$ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \simeq \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$$

Proof.

on the board

$(S = \text{Sender}/\text{Send}, W = \text{Wait}, T = \text{Trans}, A = \text{Ack},$
 $R = \text{Receiver}/\text{Reply}, d, e \in D; \text{ without restrictions})$

□

- 1 Repetition: The Alternating Bit Protocol
- 2 Handling Duplication of Messages
- 3 Concluding Remarks
- 4 Modeling Mobile Concurrent Systems

- Handling **loss of messages**: by introducing **timeouts** (see 7th exercise sheet)
- **Validity of correctness proof** (τ -cycles in *ABP*, but not in *Buffer*)?

Simplest case:

$$A(a) = \tau.A + a.\text{nil} \quad \simeq \quad B(a) = \tau.a.\text{nil}$$

Even more: every LTS containing τ -cycles is observationally congruent to one without τ -cycles

- There are notions of equivalence which distinguish **divergent** (τ -cycles) and **convergent** (no τ -cycles) processes
- **But:**
 - they are more complicated than standard bisimulation
 - (weak) bisimulation allows the proportion between the speeds of processes to vary unboundedly – why not infinite?
 - if convergence is essential, it can be assured separately

- 1 Repetition: The Alternating Bit Protocol
- 2 Handling Duplication of Messages
- 3 Concluding Remarks
- 4 Modeling Mobile Concurrent Systems

Observation: CCS imposes a **static communication structure**: if $P, Q \in Prc$ want to communicate, then both must syntactically refer to the same action name

- ⇒ every potential communication partner known beforehand,
no dynamic passing of communication links
- ⇒ **no mobility**

Goal: develop calculus in the spirit of CCS which supports mobility

- ⇒ **π -calculus**

Example 11.2 (Dynamic access to resources)

- Server S controls access to printer P
- Client C wishes to use P
- In **CCS**: P and C must share some action name a
 $\Rightarrow C$ could access P without being granted it by S
- In **π -calculus** :
 - initially only S has access to P (using link a)
 - using another link b , C can request access to P
- Formally:

$$\underbrace{\bar{b}\langle a \rangle . S'}_S \parallel \underbrace{b(c) . \bar{c}\langle d \rangle . C'}_C \parallel \underbrace{a(e) . P'}_P$$
$$\xrightarrow{\tau} S' \parallel \bar{a}\langle d \rangle . C' \parallel a(e) . P'$$
$$\xrightarrow{\tau} S' \parallel C' \parallel P' [e \mapsto d]$$

- a : link to P
- b : link between S and C
- c : “placeholder” for a
- d : data to be printed
- e : “placeholder” for d

Example 11.2 (Dynamic access to resources; continued)

- Different rôles of action name a :
 - in interaction between S and C :
object transferred from S to C
 - in interaction between C and P :
name of **communication link**
- Intuitively, names represent **access rights**:
 - a : for P
 - b : for S
 - d : for data to be printed
- If a is only way to access P
 $\implies P$ “**moves**” from S to C