Analyzing Paxos with Fault-Tolerant Multiparty Session Types

Bachelor thesis by Nicolas Daniel Torres
Date of submission: September 7, 2021

1. Review: Prof. Dr. Kirstin Peters
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department
Institut
Arbeitsgruppe

Contents

1 Introduction

In distributed computing, it is often necessary for coordinating processes to reach consensus, i.e., agree on the value of some data that are needed during computation. These processes agree on the same values to ensure correct computation, which necessitates a correct consensus algorithm. Thus, proving the correctness of consensus algorithms is important.

Due to the presence of faulty processes consensus algorithms are designed to be fault-tolerant. To achieve fault-tolerance these algorithms must satisfy the following properties: termination, integrity, and agreement[1].

Proving these properties can be complicated. Dynamic analysis tools lead to big state-spaces so static analysis is preferable. Multiparty Session Types are particularly interesting since session typing can ensure absence of communication errors and deadlocks, and protocol conformance[3]. However, to properly model unreliable communication between processes a fault-tolerant extension to Multiparty Session Types is necessary.

Peters, Nestmann, and Wagner developed such an extension.

2 Model and Analysis

2.1 Technical Preliminaries

First, we define the sorts, some additional notation, and use them to define the global type. Afterwards we define some sets and functions to create the processes.

2.1.1 Sorts

The sorts utilize type variables, which represent mathematical variables that range over types.

$\text{Maybe } \ a = \text{Just } \ a \mid \text{Nothing}$

$\text{Value} = \text{Set of values.}$

$\text{Promise } \ a = \text{Promise } \ (\text{Maybe } \ (\text{Proposal } \ a)) \ \mid \ \text{Nack } \ \mathbb{N}$

$\text{Proposal } \ a = \text{Proposal } \ \mathbb{N} \ a$

2.1.2 Notation

TODO im Grunde einfach von Peters et al klauen.

2.1.3 Global Type

Since each proposer has its own session the global type can be defined for one proposer. A quorum of acceptors A_Q is assumed.

The last phase of Paxos contains no inter-process communication, so it is not modeled in the global type.

$$
\begin{aligned}
G_{p,A_Q} = (\mu X) \bigodot_{a \in A_Q} p \to_u a : l1a \langle \mathbb{N} \rangle . \bigodot_{a \in A_Q} a \to_u p : l1b \langle \text{Promise Value} \rangle . \\
p \to_w A_Q : Accept. \left(\bigodot_{a \in A_Q} p \to_u a : l2a \langle \text{Proposal Value} \rangle \right) .end \\
\oplus Restart.X \\
\oplus Abort.end
\end{aligned}
$$

We can distinguish the individual steps of the Paxos algorithm by the labels $l1a$, $l1b$, and $l2a$.

In the first two steps the proposer sends its proposal number to each acceptor in A_Q and then listens for their responses. In step 2a the proposer decides whether to send an accept message or restart the algorithm. This decision is broadcast to all acceptors in A_Q.

2.1.4 Functions and Sets

$\text{Bool} = \{true, false\}$

$\text{prNumber} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ returns a proposal number when given two natural numbers.

$\text{promValue} : [\text{Promise } a] \to a$ if none of the promises in the given list contain a value a new value is returned. A promise contains a value if it is of the form $\text{Promise } (\text{Just } v)$. v is the value.

$\text{anyNack} : [\text{Promise } a] \to \text{Bool}$
$\text{anyNack}\,([]) = false$
$\text{anyNack}\,((\text{Nack } \_ : \_)) = true$
$\text{anyNack}\,((\_ : xs)) = \text{anyNack}\,(xs)$

$\text{promCount} : [\text{Promise } a] \to \mathbb{N}$
$\text{promCount}\,([]) = 0$
$\text{promCount}\,((\text{Promise } \_ : xs)) = 1 + \text{promCount}\,(xs)$
$\text{promCount}\,((\_ : xs)) = \text{promCount}\,(xs)$

$\mathrm{gt} : a \to \mathrm{Maybe}\ \ a \to \mathrm{Bool}$
$\mathrm{gt}\,(\_, \mathrm{Nothing}) = \mathit{true}$
$\mathrm{gt}\,(a, \mathrm{Just}\ \ b) = a > b$

$\mathrm{ge} : a \to \mathrm{Maybe}\ \ a \to \mathrm{Bool}$
$\mathrm{ge}\,(\_, \mathrm{Nothing}) = \mathit{true}$
$\mathrm{ge}\,(a, \mathrm{Just}\ \ b) = a \geq b$

$\mathrm{nFromPr} : \mathrm{Proposal}\ \ a \to \mathbb{N}$
$\mathrm{nFromPr}\,(\mathrm{Proposal}\ \ n\ \_) = n$

$\mathrm{genA_Q}\,(i, ac, pc)$ returns a set A_Q with $A_Q \subseteq A = \{pc+1, \ldots, pc+ac\}$ and $|A_Q| > \frac{|A|}{2}$.

2.1.5 Processes

System Initialization

$$\begin{aligned}
&\mathrm{Sys}\left(pc, ac, \overrightarrow{V}\right) = \overline{a}\,[1]\,(t)\,.\,\mathrm{P^p_{init}}\,(1, \mathrm{genA_Q}\,(1, ac, pc)\,, 1, [])\\
&\mid a\,[1]\,(t)\,.\Pi_{2\leq i\leq pc}\ \ \mathrm{P^p_{init}}\,(i, \mathrm{genA_Q}\,(i, pc, ac)\,, i, [])\\
&\mid \Pi_{pc<j\leq pc+ac}\ \ \mathrm{P^a_{init}}\left(j, pc, \overrightarrow{V}_{j,n}, \overrightarrow{V}_{j,pr}\right)
\end{aligned}$$

$$\mathrm{P^p_{init}}\left(i, A_Q, n, \overrightarrow{V}\right) = \overline{b_i}\left[a_1, \ldots, a_{|A_Q|}\right](s)\,.\,\mathrm{P^p}$$

$$\mathrm{P^a_{init}}\,(j, pc, n, pr) = \Pi_{1\leq i\leq pc}\ b_i\,[j]\,(s)\,.\,\mathrm{P^a}$$

Proposer

$$\begin{aligned}
&\mathrm{P^p} = (\mu X)\left(\bigodot\nolimits_{j\in A_Q}\ \ s\,[i,j]!_u l1a\,\langle \mathrm{prNumber}\,(n,i)\rangle\right).\\
&\quad\left(\bigodot\nolimits_{j\in A_Q}\ \ s\,[j,i]?_u l1b\,\langle\bot\rangle\,(v_j)\right).\\
&\quad \text{if}\ \ \mathrm{anyNack}\left(\overrightarrow{V}\right)\ \ \text{or}\ \ \mathrm{promCount}\left(\overrightarrow{V}\right) < \left\lceil\frac{|A_Q|}{2}\right\rceil\\
&\qquad \text{then}\ \ s\,[i, A_Q]!_w \mathit{restart}.\,\mathrm{update}\,(n, n+1)\,.X\\
&\qquad \text{else}\\
&\qquad\ \ s\,[i, A_Q]!_w \mathit{accept}.\bigodot\nolimits_{j\in A_Q}\ \ s\,[i,j]!_u l2a\left\langle \mathrm{Proposal}\ \ \mathrm{prNumber}\,(n,i)\ \ \mathrm{promValue}\left(\overrightarrow{V}\right)\right\rangle.\\
&\qquad\ \ \mathit{end}
\end{aligned}$$

Acceptor

```
$\mathrm{P^a} = (\mu X)\, s\,[i,j]?_u l1a\,\langle\bot\rangle\,(n')\,.$
  if $n' =\bot$
    then $\mathrm{P^a_{cont}}$
    else
      if $\mathrm{gt}\,(n',n)$
      then $\mathrm{update}\,(n,n')\,.s\,[j,i]!_u l1b\,\langle\mathrm{Promise}\ pr\rangle\,.\,\mathrm{P^a_{cont}}$
      else $s\,[j,i]!_u l1b\,\langle\mathrm{Nack}\ n\rangle\,.\,\mathrm{P^a_{cont}}$

$\mathrm{P^a_{cont}} = s\,[i,j]?_w Accept.s\,[i,j]?_u l2a\,\langle\bot\rangle\,(pr')\,.$
    if $pr' =\bot$
      then $X$
      else
        if $\mathrm{ge}\,(\mathrm{nFromPr}\,(pr')\,,n)$
          then $\mathrm{update}\,(pr,pr')\,.\,\mathrm{update}\,(n,\mathrm{nFromPr}\,(pr'))\,.X$
          else $X$
  $\oplus\, Restart.X$
  $\oplus\, Abort.end$
```

3 Evaluation

RESULTS

4 Discussion

DISCUSSION