An Extension of PlusCal for Modeling Distributed Algorithms

Heba Alkayed, Horatiu Cirstea, Stephan Merz

University of Lorraine, CNRS, Inria, Nancy, France

September 28, 2020

Introduction

Formal Specification Languages

- Algorithms modeled using TLA$^+$ can be formally verified using the TLA$^+$ Toolbox
- PlusCal algorithms have a more familiar syntax and can be translated to TLA$^+$

Distributed PlusCal Algorithms

Motivation

An extension of PlusCal with a syntax that offers constructs for modeling distributed algorithms naturally

Features

- Introduces
  - Sub-processes
  - Communication channels
- Can be translated into a TLA+ specification

Motivating example

Lamport's Mutex Algorithm

- An algorithm for Mutual Exclusion in Distributed Systems
- Critical section requests are ordered based on logical clocks
- Processes exchange 3 types of messages
  - Request
  - Acknowledge
  - Release
- Processes asynchronously receive messages from each other

Algorithm in PlusCal: main process

```
\* Variables must be declared globally to be used by the
    inter-playing processes representing this algorithm
variables network, clock ...

(**--algorithm LamportMutex {

process (proc \in Proc) {
 ncs: while (TRUE) {
      \* non-critical section
 try: \* multicast a message requesting access to cs
 enter: \* wait for acknowledgements
 cs: \* critical section
 exit: \* multicast the release message
 } \* end while
} \* end process
```

Process executing the main algorithm

Algorithm in PlusCal: helper process

```
process (comm \in Comm) {

 rcv: while (TRUE) {
       with (prc = node(self), ...) {
        \* handle request, ack and release messages
       }
     } \* end while
} \* end process
```

Process handling messages

Algorithm in PlusCal: helper process

```
Proc == 1 ..  N
Comm == N+1..N+N
node(c) == c - N
```

```
process (comm \in Comm) {
 rcv: while (TRUE) {
       with (prc = node(self), ...) {
         \* handle request, ack and release messages
       }
     } \* end while
} \* end process
**)
```

Lamport Mutex in Distributed PlusCal

```
fifos network[Proc, Proc];
process(p \in Proc)
     variables ..
{
     ncs: while (TRUE) {\*non-critical section
     ...
     exit: \* multicast the
           \* release message
    } \* end while
}  {
    rcv: while (TRUE) {\* receive msg from channel
         \* handle request, ack and release messages
         ...
    } \* end while
} \* end message handling thread
**)
```

sub-process executing the main algorithm

message handling sub-process

Modeling channels

Declaration (in PlusCal)

```
network=[p,q \in Proc |-> ⟨⟩]
```

Modeling channels

Declaration (in PlusCal)

```
network=[p,q \in Proc |-> ⟨⟩]
```

Declaration (in Distributed PlusCal)

```
fifos network[Proc, Proc];
```

Modeling channels

Declaration (in PlusCal)

```
network=[p,q \in Proc |-> ⟨⟩]
```

Declaration (in Distributed PlusCal)

```
fifos network[Proc, Proc];
```

Operation (in PlusCal)

```
macro mcast(p, m) {
 network := [s,d \in Proc |->
 IF s = p /\ d # p
 THEN Append(network[s,d], m)
 ELSE network[s,d]]
}
mcast(self, Request(clock));
```

Modeling channels

Declaration (in PlusCal)

```
network=[p,q \in Proc |-> ⟨⟩]
```

Declaration (in Distributed PlusCal)

```
fifos network[Proc, Proc];
```

Operation (in PlusCal)

```
macro mcast(p, m) {
 network := [s,d \in Proc |->
 IF s = p /\ d # p
 THEN Append(network[s,d], m)
 ELSE network[s,d]]
}
mcast(self, Request(clock));
```

Operation (in Distributed PlusCal)

```
  \* 1st argument: channel name
  \* 2nd argument specifies
     recipients and message

multicast(network,
          [self, p \in Proc |->
           Request(clock)]);
```

Distributed PlusCal

General Structure of an algorithm

```
(* --algorithm <algorithm name>
(* Declaration section *)
variables <variable declarations>
channels <channel declarations>
fifos <fifo declarations>
(* ... *)
(* Processes section *)
process (<name> [= | \in] <Expr>))
  variables <variable declarations>
  <subprocesses>
*)
```

Operations on channels

- Supported operators
  - `send(ch, el)`
  - `receive(ch, var)`
  - `broadcast(ch, [x` $\in$ `S` $\mapsto$ `e(x)]`
  - `multicast(ch, [x` $\in$ `S` $\mapsto$ `e(x)]`
  - `clear(ch)`

Translation of Unordered Channels

$$\textit{channel}\ \langle \textit{id} \rangle [\langle \textit{Expr}_1 \rangle, \ldots, \langle \textit{Expr}_N \rangle];$$

- Translation based on TLA$^+$ sets
  - `send(chan[e], msg)` ≜

    ```
    chan’ = [chan EXCEPT ![e] = chan[e] \cup {msg}]
    ```
  - `receive(chan[e], var)` ≜

    ```
    \E temp \in chan[e]:
      /\ var’ = temp
      /\ chan’ = [chan EXCEPT ![e] = chan[e] \ {temp}]
    ```

Translation of FIFO Channels

$$\textit{fifo}\ \langle \mathit{id} \rangle [\langle \mathit{Expr}_1 \rangle, \ldots, \langle \mathit{Expr}_N \rangle];$$

- Translation based on TLA$^+$ sequences
  - `send(chan[e], msg)` $\triangleq$

    ```
    chan' = [chan EXCEPT ![e] = Append(@, msg)]
    ```

  - `receive(chan[e], var)` $\triangleq$

    ```
    /\ Len(chan[e]) > 0
    /\ var' = Head(chan[e])
    /\ chan' = [chan EXCEPT ![e]= Tail(@)]
    ```

Program counter

- The variable *pc* is indexed by processes and sub-processes

```
pc = [self \in ProcSet|->
            CASE self \in $P_1$ -> << $lbl_{11}$, $lbl_{12}$, ...>>
              [] self \in ...]
```

where the lbl_{ij} are the entry labels of the subprocesses of the process type P_i.

Translation to TLA$^+$

```
exit: clock := clock + 1;
      multicast(network, [self, p \in Proc \ {self} |->
                                    Release(clock)]);
```

```
exit(self) ==
    /\ pc[self][1] = "exit"
    /\ clock' = [clock EXCEPT ![self] = clock[self] + 1]
    /\ network' = [<<slf, p>> \in DOMAIN network |->
        IF
            slf = self /\ p \in Proc \ { self }
        THEN
            Append(network[slf, p], Release(clock'[self]))
        ELSE
            network[slf, p]]

    /\ pc' = [pc EXCEPT ![self][1] = "ncs"]
    /\ UNCHANGED << req, ack, sndr, msg >>
```

Multicast
Translation

Contributions and future work

Contributions

- An extension of PlusCal offering the possibility to define
  - Sub-Processes
  - Communication Channels
- A backward compatible translator to TLA$^+$
  `https://github.com/hebaalkayed/DistributedPlusCal`

Future Work

- Introduce more types of communication channels
- Consider defining channel operations in a TLA$^+$ module