An Extension of PlusCal for Modeling Distributed Algorithms

Heba Alkayed, Horatiu Cirstea, Stephan Merz

University of Lorraine, CNRS, Inria, Nancy, France

September 12, 2020

Introduction

Formal methods for distributed algorithms

- 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

- An extension of PlusCal offering constructs for modeling distributed algorithms
- Can be translated into a TLA+ specification
- Introduces
  - Sub-processes
  - Communication channels

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>
*)
```

Communication Channels

- Classified by the way they handle the addition and removal of messages
  - Unordered channels
  - FIFO 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)`

Unordered Channels

- **channel** or **channels**, as shown below.

$$\textbf{channel}\ \langle id \rangle [\langle Expr_1 \rangle, \ldots, \langle Expr_N \rangle];$$

- based on TLA$^+$ sets
- Operator translation to TLA$^+$
  - `send(ch, el)` $\triangleq$ `chan' = chan \cup {el}`
  - `receive(chan, var)` $\triangleq$

```
\E e \in chan:
    /\ var' = e
    /\ chan' =  chan \ {e}
```

FIFO Channels

- **fifo** or **fifos**, as shown below.

$$\textbf{fifo}\ \langle id \rangle [\langle Expr_1 \rangle, \ldots, \langle Expr_N \rangle];$$

- based on TLA$^+$ sequences
- Operator translation to TLA$^+$
  - `send(ch, el)` $\triangleq$ `chan' = Append(chan, el)`
  - `receive(chan, var)` $\triangleq$

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

Sub-Processes

- Enables the process to execute multiple tasks in parallel.
- Each sub-process consists of labeled statements.
- The special variable pc was modified to have the following definition

$$pc = [self \in ProcSet \mapsto [self \in IdSet \mapsto \langle "lbl", \ldots \rangle]]$$

where `ProcSet` is a set that contains all the process identifiers, and the labels that appear in the sequence are the entry point actions for each sub-process.

Distributed PlusCal Example

```
 process(n \in Nodes)
     variables clock = 0, req = [n \in Nodes |-> 0],
               ack = {}, sndr, msg;
   { \* thread executing the main algorithm
ncs: while (TRUE) {
       skip;  \* non-critical section
try:   clock := clock + 1; req[self] := clock; ack :=
    {self};
       multicast(network, [self, nd \in Nodes |->
    Request(clock)]);
enter: await (ack = Nodes /\ \A n \in Nodes \ {self} :
    beats(self, n));
cs:    skip;  \* critical section
exit:  clock := clock + 1;
       multicast(network, [self, n \in Nodes \ {self} |->
                           Release(clock)]);
     } \* end while
  }
```

Translation to TLA+

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

```
exit(self) ==
    /\ pc[self] [1] = "exit"
    /\ clock' = [clock EXCEPT ![self] = clock[self] + 1]
    /\ network' = [<<slf, n>> \in DOMAIN network |->
         IF
             slf = self /\ n \in Nodes \ { self }
         THEN
             Append(network[slf, n], Release(clock'[self]))
         ELSE
             network[slf, n]]
    /\ pc' = [pc EXCEPT ![self][1] = "ncs"]
    /\ UNCHANGED << req, ack, sndr, msg >>
```

Contributions and future work

Contributions

- An extension of PlusCal called Distributed PlusCal
- Distributed PlusCal offers constructs that are designed for modeling distributed algorithms
- A backward compatible translator that translates from Distributed PlusCal and PlusCal to TLA+

Future Work

In the future we aim to introduce more types of communication channels and channel operators.