FORMAL METHODS IN NETWORKING

COMPUTER SCIENCE 598D, SPRING 2010

PRINCETON UNIVERSITY

LIGHTWEIGHT MODELING

IN PROMELA/SPIN AND ALLOY

Pamela Zave

AT&T Laboratories—Research

Florham Park, New Jersey, USA

LIGHTWEIGHT MODELING

DEFINITION

- constructing a very abstract model of the core concepts of a system
- using a "push-button" analysis tool to explore its properties

"analysis" is more
general than "verification"

WHY IS IT "LIGHTWEIGHT"?

- because the model is very abstract in comparison to a real implementation, and focuses only on core concepts, it is small and can be constructed quickly
- because the analysis tool is "push-button", it yields results with little effort

in contrast,
theorem proving is not "push-button"

WHAT IS ITS VALUE?

- it is a design tool that reveals conceptual errors early

decades of research on
software engineering proves
that the cost of fixing a bug
rises exponentially with the
delay in its discovery

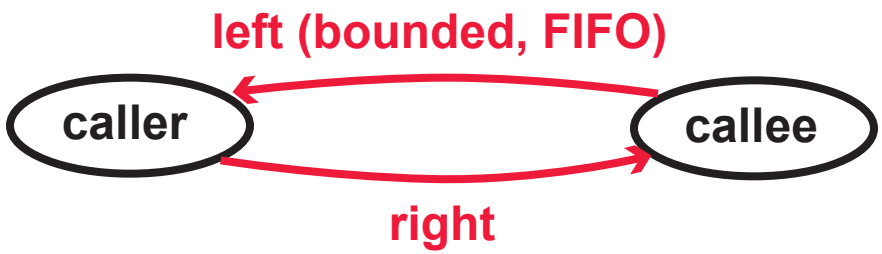
- it is a documentation tool that provides complete, consistent, and unambiguous information to implementors and users
- it is easy (at least to get started) and fun!

"If you like surprises, you will
love lightweight modeling."
ÑPamela Zave

Read introduction to Software Abstractions for Daniel Jackson's view .

```
mtype = { invite, accept, reject }
chan left = [3] of {mtype};
chan right = [3] of {mtype};
proctype caller (chan in, out) {
            out!invite;
inviting:   do
            :: in?accept; goto confirmed
            :: in?reject; goto end
            od;
confirmed:  do
            :: in?invite; out!accept
            :: out!invite; in?accept
            od;
end:        skip
}
proctype callee (chan in, out) {
            in?invite;
invited:    do
            :: out!accept; goto confirmed
            :: out!reject; goto end
            od;
confirmed:  do
            :: in?invite; out!accept
            :: out!invite; in?accept
            od;
end:        skip
}
init { atomic { run caller(left,right);
                run callee(right,left)
    }       }
```

SIP VERSION 1



do statement executes zero or more guarded commands

a guarded command can be executed only if its guard is true/executable

chan?mtype reads a message of type *mtype* from *chan*

executable iff. *chan* is not empty and its first message is of type *mtype*

chan!mtype writes a message of type *mtype* to *chan*

executable iff. *chan* is not full and holds messages of type *mtype*

nondeterminism models:

user choice

distributed concurrency

SIP VERSION 2

FIXES DEADLOCK DISCOVERED IN VERSION 1

```
proctype caller (chan in, out) {
                out!invite;
inviting:   do
                :: in?accept; goto confirmed
                :: in?reject; goto end
                od;
confirmed:  do
                :: in?invite; out!accept
                :: out!invite; goto reInviting
                od;
reInviting:  do
                :: in?accept; goto confirmed
                :: in?race; goto confirmed
                :: in?invite; out!race
                od;
end:        skip
}
```

```
proctype callee (chan in, out) {
                in?invite;
invited:    do
                :: out!accept; goto confirmed
                :: out!reject; goto end
                od;
confirmed:  do
                :: in?invite; out!accept
                :: out!invite; goto reInviting
                od;
reInviting:  do
                :: in?accept; goto confirmed
                :: in?race; goto confirmed
                :: in?invite; out!race
                od;
end:        skip
}
```