

COMP424/524-06A Topics in Software Engineering

Part I – CTL Model Checking
12. Fairness Assumptions

Robi Malik

THE UNIVERSITY OF WAIKATO DEPARTMENT OF COMPUTER SCIENCE
TARI ROROHIKO

Problems when Proving Liveness

"If the mouse never gets to move,
it will never reach room 3."

We cannot prove:

$AG AF \text{ mouse} = 3$

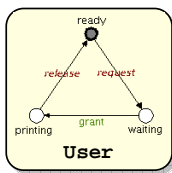
To prove liveness properties in the system,
we must make **fairness assumptions**.

We can prove:

$AG A(G F \text{ mouse_moves} \Rightarrow F \text{ mouse} = 3)$

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 4

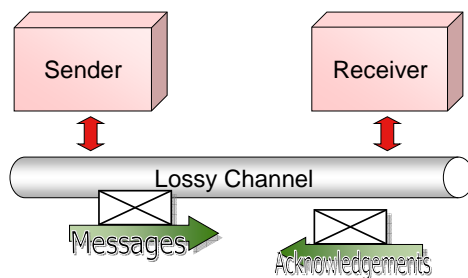
Implicit Assumptions of Liveness



- Every state will be entered eventually.
- The user will request to print infinitely often.
- Does this correctly represent the modelling assumptions?

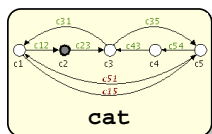
© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 2

Communication Protocols

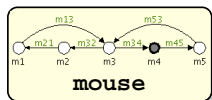


© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 5

Liveness and Composition

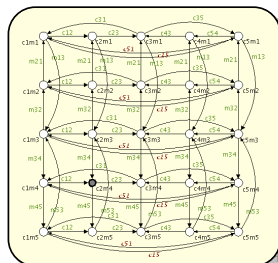


cat



mouse

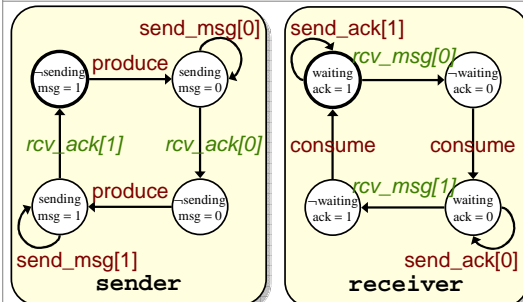
$AG AF m3$ ✓



$AG AF m3$ ✗

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 3

The Alternating Bit Protocol



© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 6

Sender and Receiver in SMV

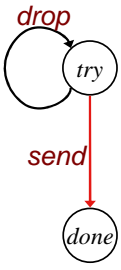
```

MODULE sender(ack)
VAR
  produce: boolean;
  sending: boolean;
  msg: boolean;
ASSIGN
  produce := case
    !sending: {0, 1};
    1: 0;
  esac;
  init(sending) := 0;
  next(sending) := case
    !sending: produce;
    msg=ack: 0;
  esac;
  init(msg) := 1;
  next(msg) := case
    produce: !msg;
    1: msg;
  esac;
MODULE receiver(msg)
VAR
  consume: boolean;
  waiting: boolean;
  ack: boolean;
ASSIGN
  consume := case
    !waiting: {0, 1};
    1: 0;
  esac;
  init(waiting) := 1;
  next(waiting) := case
    !waiting: consume;
    ack != msg: 0;
  esac;
  init(ack) := 1;
  next(ack) := case
    waiting: msg;
    1: ack;
  esac;

```

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 7

Justice



Definition:

A transition is called **just** if it cannot happen that the transition is continuously enabled without being taken eventually.

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 10

A Property ...

“Every message sent by the sender eventually arrives at the receiver.”

AG (sender.msg=0 \Rightarrow **AF** msg.output=0)
AG (sender.msg=1 \Rightarrow **AF** msg.output=1)

Cannot be verified:
 Channels may lose messages indefinitely ...

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 8

Justice in NuSMV

JUSTICE
 !loss

SPEC
 AG (sender.msg=0 \rightarrow AF msg.output=0)

LTLSPEC
 G F !loss \rightarrow G (sender.msg=0 \rightarrow F msg.output=0)

Consider only paths where \neg loss holds infinitely often.

Equivalent PLTL formula

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 11

Assuming Fairness in the Model

```

MODULE channel(input)
VAR
  output: boolean;
  loss: boolean;
ASSIGN
  init(output) := 1;
  next(output) := case
    loss: output;
    1: input;
  esac;
JUSTICE
  !loss

```

Fairness assumption:

The channel does not lose all messages.

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 9

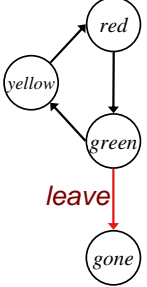
Another Property ...

“Every message produced by the sender is eventually consumed by the receiver.”

AG (sender.produce \Rightarrow **AF** receiver.consume)

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 12

Compassion



Definition:

A transition is called **compassionate** if it cannot happen that the transition is enabled infinitely often without being taken eventually.

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 13

Compassion in NuSMV

```

COMPASSION
(car=waiting & lights=green,
 car=crossing)
LTLSPEC
G F car=gone
LTLSPEC
((G F (car=waiting & lights=green)
 -> G F car=crossing)
 -> G F car = gone))
  
```

Equivalent version without compassion

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 16

Another Example ...

```

MODULE Car(lights)
VAR
car: {coming, crossing, waiting, gone};
ASSIGN
init(car) := gone;
next(car) := case
  car=gone: {gone, coming};
  car=coming & lights=red: waiting;
  car=coming & lights=yellow: {waiting, crossing};
  car=coming & lights=green: crossing;
  car=waiting & lights=red: waiting;
  car=waiting & lights=yellow: waiting;
  car=waiting & lights=green: {waiting, crossing};
  car=crossing: gone;
esac;
SPEC
AG AF car=gone
  
```

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 14

Reading

Bérard et. al.:

Chapter 7 – Safety Properties
 Chapter 8 – Liveness Properties
 Chapter 10 – Fairness Properties

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 17

Compassion in NuSMV

```

COMPASSION
( $\phi$ ,  $\psi$ )
  
```

Consider only paths that satisfy the following condition.

“If ϕ is true infinitely often, then ψ also must be true infinitely often.”

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 15

Last Words

Do not blindly trust a model checker that prints true.

- Check properties that you expect to fail.
- Check the preconditions of implications.
- Check your assumptions.

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-12 Slide 18