

# Algorithms for Model Checking

## 2IW50

Lecture 7b: Example for Data Abstraction:  
Bounded Retransmission Protocol

Jaco van de Pol

`vdpol@cwi.nl`

`http://www.cwi.nl/~vdpol/amc.html`

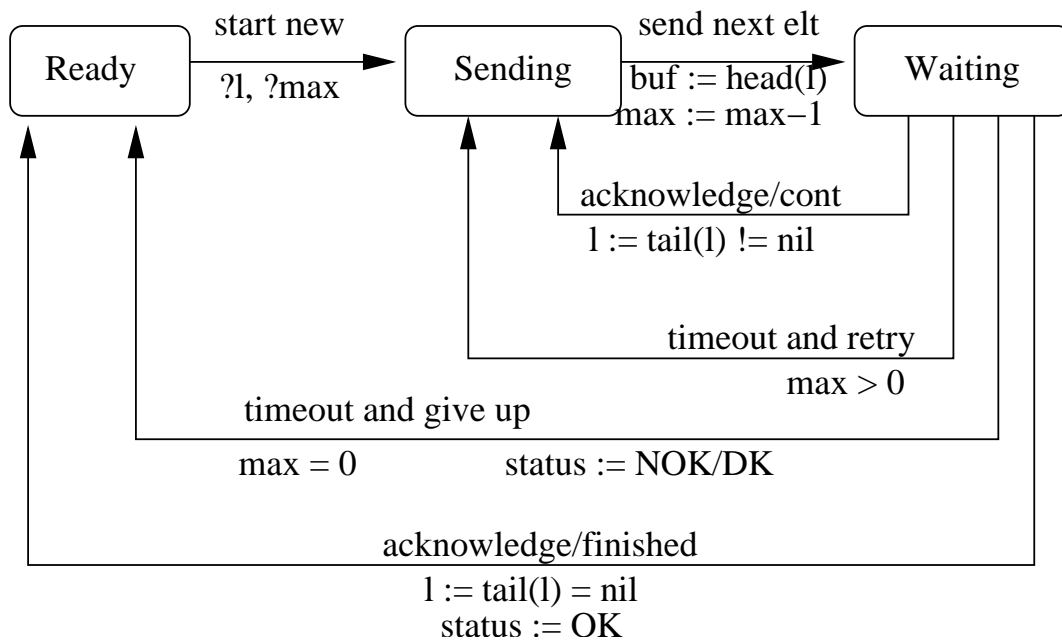
Technische Universiteit Eindhoven  
Centrum voor Wiskunde en Informatica

# Bounded Retransmission: Intro

We model the sender of the Bounded Retransmission Protocol.

- It sends elements from a list  $l$  one by one over an unreliable channel.
- On time-out it will resend elements, at most  $max$  times.
- The status indicates if the list arrived correctly.

Informal sketch of behaviour:



## States:

We have the following variables ( $v_i$ ) with their domain ( $D_i$ ).

(in applications, variables have different domains)

- `list: List[Data]`
- `max: Nat`
- `state: {ready, sending, waiting}`
- `status: {OK, NOK, DK}`
- `buf: Data`

Example states (if `Data = Nat`) are:

- `(list = [3,4], max = 5, state = sending, status = OK, buf = 3)`
- `(list = [4], max = 0, state = waiting, status = NOK, buf = 4)`

## Bounded Retransmission: Specification

- Initial states:  $\mathcal{S}_0 := ( \text{state} = \text{ready} )$
- Possible transitions:

**Start\_new\_transmission =**

`state = ready /\ state' = sending`

**Send\_next\_element =**

`state = sending /\ state' = waiting  
/\ Max = Max' + 1 /\ buf' = head(list)  
/\ list' = list`

**Get\_acknowledgement =**

`state = waiting /\ list' = tail(list) /\  
( (tail' = [] /\ state' = ready  
    /\ status' = OK )  
\/ (tail' != [] /\ state' = sending  
    /\ max' = max)  
)`

## Bounded Retransmission: Specification (2)

- Some more transitions:

**Timeout\_and\_retry =**

```
state = waiting /\ max > 0
  /\ state' = sending /\
  /\ list = list' /\ max = max'
```

**Timeout\_and\_give\_up =**

```
state = waiting /\ max = 0 /\
state' = ready /\ status' =
  (if list = [] then DK else NOK)
```

- Finally, we specify the full transition relation:

**$\mathcal{R} :=$**

```
  Start_new_transmission
  \/ Send_next_element
  \/ Get_acknowledgement
  \/ Timeout_and_retry
  \/ Timeout_and_give_up
```

## BRP: data abstraction

- Actually, the Kripke Structure is infinite.
- To study the control-aspects of the system by model checking, we decide to abstract from the data and the counter. (we need a finite abstraction).
- Abstract domains:
  - $A_{List} := \{empty, non\_empty\}$
  - $A_{Nat} := \{\cdot\}$  (effectively removed)
  - $A_{Data} := \{\cdot\}$
  - $A_{State} := State$  (unchanged)
  - $A_{Status} := Status$
- mapping:
  - $h(n : Nat) = h(d : Data) = \cdot$
  - $h([]) = empty, h([x|l]) = non\_empty$
  - $h(s : State) = s, h(s : Status) = s.$

## Abstract states/labels:

- Examples of abstract labels  $\widehat{AP}$ :  
 $\widehat{list} = empty, \widehat{list} = non\_empty, \widehat{max} = \cdot,$   
 $\widehat{state} = waiting, \text{etc.}$
- Examples of labels in  $L'$ :
- $L'((list = [3,4], max = 5,$   
 $state = sending, status = OK,$   
 $buf = 3)) =$   
 $(\widehat{list} = non\_empty, \widehat{max} = \cdot,$   
 $\widehat{state} = sending, \widehat{status} = OK,$   
 $\widehat{buf} = \cdot)$
- So we can still express properties like:

$$\mathbf{AG} (\widehat{status} = OK \rightarrow \widehat{list} = empty)$$

# Bounded Retransmission: Abstract Specification

(I drop  $\hat{\cdot}$  on variables)

Note that the abstract specification is more non-deterministic:

- Initial states:  $\mathcal{S}_0 := ( \text{state} = \text{ready} )$
- Possible transitions:

**Start\_new\_transmission =**

```
state = ready /\ state' = sending
```

**Send\_next\_element =**

```
state = sending /\ state' = waiting  
\ list' = list
```

**Get\_acknowledgement =**

```
state = waiting /\  
( (tail' = empty /\ state' = ready  
  /\ status' = OK )  
\ (tail' != non_empty  
  /\ state' = sending)  
)
```

## Bounded Retransmission: Abstract Specification (2)

- Some more transitions:

**Timeout\_and\_retry =**

```
state = waiting
/\ state' = sending
/\ list = list'
```

**Timeout\_and\_give\_up =**

```
state = waiting /\
state' = ready /\ status' =
(if list = empty then DK else NOK)
```

- Finally, we specify the full transition relation:

**$\mathcal{R} :=$**

```
Start_new_transmission
\// Send_next_element
\// Get_acknowledgement
\// Timeout_and_retry
\// Timeout_and_give_up
```