Modeling systems and specifying temporal properties
a gentle introduction

Metodi Formali per il Software
Laurea Magistrale in Informatica, AA 2013-2014
Prof.ssa Anna Labella

Esercitatore: Federico Mari
mari@di.uniroma1.it

Model Checking Laboratory Group
http://mclab.di.uniroma1.it/

Dipartimento di Informatica
Sapienza Università di Roma

7 aprile 2014
Agenda

- Introduction on modelling and specification
- Mutual exclusion
- The ferryman
- The alternating bit protocol
Introduction
Mutual exclusion

The ferryman
The alternating bit protocol
Modelling systems

Modelling Problem

Input

- requirements of real system in natural language

Output

- computational model of a system (as a finite set of automata interacting between them)

Usual problems when modelling

- Identify all single processes in the input real system
- Formalise dynamics of a single process
- Formalise communication among processes
- Choose the right level of abstraction
Introduction

Specification of properties

Specification Problem

Input

▶ computational model of a system
▶ properties to be verified on the given system

Output

▶ formal properties in temporal logic (LTL, CTL, CTL*, ...) modelling input properties

Usual problems in specification

▶ Link properties to elements of the input computational model
▶ Understand if a property is **safety** (it must hold for all states of the system), **liveness** (depending on evolution of time, e.g. *eventually in the future*), **fairness** (guaranteeing that bad behaviour does not repeat forever), ...
▶ Verification or planning?

F. Mari (mari.di.uniroma1.it)  MF.1: Modeling systems and specifying temporal properties  7 aprile 2014
Modelling and Specification

Description of real system

Modelling

Formal model with some behaviours of real system

Description of properties over real system

Specification

$\text{AG } \varphi$

$\text{AG } (\varphi \rightarrow \text{AF } \psi)$

Temporal logic properties specifying input properties on abstract model A
Introduction

Mutual exclusion

The ferryman

The alternating bit protocol
Shared resources
Mutual exclusion

Description of real system

- $N > 1$ processes
- 1 shared resource
- Each process wants to access the shared resource: critical session
- Each critical session must be as small as possible
- Only one process can be in its critical session at a time

Problem

Find a protocol for determining which process is allowed to enter its critical section at which time
Mutual exclusion

Description of properties

Safety
Only one process is in its critical section at any time

Liveness
Whenever any process requests to enter its critical section, it will eventually be permitted to do so

Non-blocking
A process can always request to enter its critical section

No strict sequencing
Processes need not enter their critical section in strict sequence
Modelling of real system
A first attempt

- 2 processes
- Each process has three states:
  - non-critical state \((n)\)
  - trying to enter its critical state \((t)\)
  - critical state \((c)\)
- Each process undergoes transitions in the cycle \(n \rightarrow t \rightarrow c \rightarrow n \rightarrow \ldots\)
Modelling of real system
A first attempt

Mutual exclusion

F. Mari (mari.di.uniroma1.it)  MF.1: Modeling systems and specifying temporal properties  7 aprile 2014  10 / 40
Mutual exclusion

Specification of properties

Safety

Only one process is in its critical section at any time

\[
G \neg (c_1 \land c_2)
\]
Mutual exclusion

Specification of properties

Safety
Only one process is in its critical section at any time

Specification (LTL)
\[ G \neg (c_1 \land c_2) \]

Note
Specification is True
Specification of properties

Liveness
Whenever any process requests to enter its critical section, it will eventually be permitted to do so.
Mutual exclusion

Specification of properties

Liveness
Whenever any process requests to enter its critical section, it will eventually be permitted to do so

Specification (LTL)
$G \left( t_1 \land F \ c_1 \right)$

Note
Specification is False
Mutual exclusion

Specification of properties

Non-blocking
A process can always request to enter its critical section
⇒ for each process $i$ and for each state $n_i$; there is a successor $t_i$

Specification
This property cannot be expressed in LTL
Specification of properties

No strict sequencing
Processes need not enter their critical section in strict sequence
⇒ There is a path with two distinct states satisfying $c_1$ such that no state in between them has that property
⇒ We cannot say in LTL “there exists a path”

But we can complement the property and check for complementation

No strict sequencing negation
All paths having a $c_1$ period which ends cannot have a further $c_1$ state until a $c_2$ state occurs
Specification of properties

No strict sequencing (negation)
All paths having a $c_1$ period which ends cannot have a further $c_1$ state until a $c_2$ state occurs
Mutual exclusion

Specification of properties

No strict sequencing (negation)
All paths having a $c_1$ period which ends cannot have a further $c_1$ state until a $c_2$ state occurs

Specification (LTL)

$$G \left( c_1 \rightarrow c_1 W (\neg c_1 \land \neg c_1 W c_2) \right)$$

Note

Specification (negation) is False

$\Rightarrow$ No strict sequencing is True
Model refinement

- Verification of liveness property failed on the first modelling attempt
- ⇒ The model needs to be refined

MODEL REFINEMENT THROUGH PROPERTY VERIFICATION

Refinement of mutual exclusion first attempt model

Split $s_3$ into two states, distinguishing which process asked for its critical section first and giving precedence to it (exercise)
Model refinement through property verification
Introduction
Mutual exclusion

The ferryman
The alternating bit protocol
The ferryman
Description of real system

- Actors: ferryman, and the goods (goat, cabbage, and wolf)
- Actors must cross a river
- The ferryman can carry one good at a time on his boat
- Unsafe situations:
  - the goat and the cabbage cannot stay without ferryman on the same river bank
  - the wolf and the goat cannot stay without ferryman on the same river bank

Problem
Can the ferryman transport all the goods to the other side, without any conflicts occurring?
Description of real system

- Actors: ferryman, and the goods (goat, cabbage, and wolf)
- Actors must cross a river
- The ferryman can carry one good at a time on his boat
- Unsafe situations:
  - the goat and the cabbage cannot stay without ferryman on the same river bank
  - the wolf and the goat cannot stay without ferryman on the same river bank

Problem

Can the ferryman transport all the goods to the other side, without any conflicts occurring?

⇒ This is a **planning problem**

Plan

A **plan** is a sequence of (state, action) pairs such that from any **initial state** the system is driven to the **goal state**
Description of properties

- Is there a path from the initial state such that it has a state along it at which all the goods are on the other side, and during the transitions to that state the goods are never left in an unsafe, conflicting situation?
Modelling of real system

- Each actor (agent) is modelled with an automaton with actions on edges
- Agents
  - \( f \): ferryman
  - \( g \): goat
  - \( c \): cabbage
  - \( w \): wolf
- Agents can be in the initial bank (value 0) or in the destination bank (value 1)
- We introduce variable \( \text{carry} \) taking a value indicating whether the goat, cabbage, wolf or nothing is carried by the ferryman
Modelling of real system

The ferryman \( f \) (blue edges are without any action)
Modelling of real system

Variable carry (blue edges are without any action)
Modelling of real system

The goat g (blue edges are without any action)

\[ f = 1 \land \text{carry}' = "G" \land f' = 0 \]

\[ f = 0 \land \text{carry}' = "G" \land f' = 1 \]

Next state variables

Next state variables are primed: carry', f', ...

They represent the value of corresponding variable at next time step

Non primed variables represent the value at this time step
Modelling of real system

The cabbage c (blue edges are without any action)

\[ f = 1 \land \text{carry}' = "C" \land f' = 0 \]

\[ f = 0 \land \text{carry}' = "C" \land f' = 1 \]
Modelling of real system

The wolf $w$ (blue edges are without any action)

\[ f = 1 \land carry' = "W" \land f' = 0 \]

\[ f = 0 \land carry' = "W" \land f' = 1 \]
Modelling of real system

Automata composition

Variable f

\[
\begin{align*}
\text{start} & \rightarrow 0 & 1 \\
\end{align*}
\]

Variable carry

\[
\begin{align*}
\text{start} & \rightarrow 0 & G & W & C \\
\end{align*}
\]

Variable g

\[
\begin{align*}
f & = 1 \land \text{carry}' = "G" \land f' = 0 \\
\text{start} & \rightarrow 0 & 1 \\
\end{align*}
\]

Variable c

\[
\begin{align*}
f & = 1 \land \text{carry}' = "C" \land f' = 0 \\
\text{start} & \rightarrow 0 & 1 \\
\end{align*}
\]

Variable w

\[
\begin{align*}
f & = 1 \land \text{carry}' = "W" \land f' = 0 \\
\text{start} & \rightarrow 0 & 1 \\
\end{align*}
\]
Specification of properties

Strategy to avoid unsafe situation
If the goat and the cabbage, or the wolf and the goat, are on the same river bank then the goat is with the ferryman

Planning problem goal
The goat, the cabbage, and the wolf are on the other river bank (value 1)
Specification of properties

Strategy to avoid unsafe situation
If the goat and the cabbage, or the wolf and the goat, are on the same river bank then the goat is with the ferryman

Planning problem goal
The goat, the cabbage, and the wolf are on the other river bank (value 1)

Specification (LTL)
\(((g = c \lor w = g) \rightarrow g = f) \U (f \land g \land c \land w)\)
The ferryman

Specification of properties

Strategy to avoid unsafe situation
If the goat and the cabbage, or the wolf and the goat, are on the same river bank then the goat is with the ferryman

Planning problem goal
The goat, the cabbage, and the wolf are on the other river bank (value 1)

Specification (LTL)
\[ ((g = c \lor w = g) \rightarrow g = f) \ U \ (f \land g \land c \land w) \]

Specification for planning problem (LTL)
\[ \neg (((g = c \lor w = g) \rightarrow g = f) \ U \ (f \land g \land c \land w)) \]
\[ \Rightarrow \text{Finding a counterexample yields to a plan} \]
The ferryman

Plan

Step 1

Variable f

start → 0 → 1

Variable carry

start → 0 → G → W → C → 1

Variable g

f = 1 ∧ carry' = “G” ∧ f’ = 0

start → 0 → 1

Variable c

f = 1 ∧ carry' = “C” ∧ f’ = 0

start → 0 → 1

Variable w

f = 1 ∧ carry' = “W” ∧ f’ = 0

start → 0 → 1

F. Mari (mari.di.uniroma1.it) MF.1: Modeling systems and specifying temporal properties 7 aprile 2014 29 / 40
Plan

Step 2

Variable f

Variable carry

Variable g

Variable c

Variable w
Plan

Step 3

Variable f

Variable carry

Variable g

Variable c

Variable w

\[ f = 1 \land \text{carry}' = "G" \land f' = 0 \]

\[ f = 0 \land \text{carry}' = "G" \land f' = 1 \]

\[ f = 1 \land \text{carry}' = "C" \land f' = 0 \]

\[ f = 0 \land \text{carry}' = "C" \land f' = 1 \]

\[ f = 1 \land \text{carry}' = "W" \land f' = 0 \]

\[ f = 0 \land \text{carry}' = "W" \land f' = 1 \]
The ferryman

Plan

Step 4

Variable f

start → 0 → 1

Variable carry

start → 0 → G → W → C

Variable g

\[ f = 1 \land \text{carry}' = "G" \land f' = 0 \]

start → 0 → 1

Variable c

\[ f = 1 \land \text{carry}' = "C" \land f' = 0 \]

start → 0 → 1

Variable w

\[ f = 1 \land \text{carry}' = "W" \land f' = 0 \]

start → 0 → 1
Plan
Step 5

Variable f

\[ f = g \]

Variable carry

\[ f = g \]

Variable c

\[ f = c \]

Variable w

\[ f = w \]
The ferryman

Plan

Step 6

Variable f

Variable carry

Variable g

Variable c

Variable w
Plan

Step 7

Variable f

Variable carry

Variable g

\[ f = 1 \land \text{carry}' = "G" \land f' = 0 \]

\[ f = 0 \land \text{carry}' = "G" \land f' = 1 \]

Variable c

\[ f = 1 \land \text{carry}' = "C" \land f' = 0 \]

\[ f = 0 \land \text{carry}' = "C" \land f' = 1 \]

Variable w

\[ f = 1 \land \text{carry}' = "W" \land f' = 0 \]

\[ f = 0 \land \text{carry}' = "W" \land f' = 1 \]
Plan

Step 8

Variable f

start → 0 → 1

f = 0 \land \text{carry}' = "G" \land f' = 0
f = 1 \land \text{carry}' = "G" \land f' = 1

Variable carry

f = g

Variable c

f = 0 \land \text{carry}' = "C" \land f' = 1

Variable w

f = 0 \land \text{carry}' = "W" \land f' = 1

f = 1 \land \text{carry}' = "W" \land f' = 0
Introduction
Mutual exclusion

The ferryman
The alternating bit protocol
The alternating bit protocol: motivations

Alice  <->  Insecure channel  <->  Bob

I love you

So do I

WOW!!

What?

Yeah!!

Charlie
Description of real system

- Alice and Bob want to communicate
- On a lossy line, i.e. a line which may lose or duplicate messages
- The line does not lose infinitely many messages (fairness)
- The line does not corrupt messages
- Protocol uses acknowledgements
The alternating bit protocol

Description of real system

- Agents
  - Sender
  - Receiver
  - Message channel
  - Ack channel
- Figure shows two instances of Alternating Bit Protocol (ABP)
  1. Sender: Alice, Receiver: Bob
  2. Sender: Bob, Receiver: Alice
- Sender sends again the same message until he receives the corresponding ack
- Receiver sends again the same ack until he receives a message with expected bit
The alternating bit protocol

Description of properties

Safety

If the message bit 1 has been sent and the correct acknowledgement has been returned, then a 1 was indeed received by the receiver (the same holds for 0)

Liveness

Messages get through eventually

⇒ For any state there is inevitably a future state in which the current message has got through

Similarly, acknowledgements get through eventually
Modelling of real system

The sender

Input variable: ack boolean

Sending status (st)

- The sender sends the bit in variable message1
- The ABP control bit is in variable message2
- Fairness: The sender must be run infinitely often
Modelling of real system

The receiver

Input variables message1 (actual message), message2 (ABP control bit) boolean

Variable ack

Receiving status (st)
The alternating bit protocol

Modelling of real system
The acknowledgement channel (1 bit)

Input variable input boolean

Variable output

▶ Fairness: The one bit channel must be run infinitely often
▶ Fairness: Variable forget cannot be always 1, for both input 1 and 0
Modelling of real system
The message channel (2 bits)

Input variables input1, input2 boolean

Variables outputI, I in \{1, 2\}

Variable forget

- **Fairness:** The two bit channel must be run infinitely often
- **Fairness:** Variable forget cannot be always 1, for all inputs (4 cases)
Modelling of real system

The whole protocol

Note
Variable forget models insecure channel (non-deterministic)
Specification of properties

Safety

If the message bit 1 has been sent and the correct acknowledgement has been returned, then a 1 was indeed received by the receiver (the same holds for 0)

<table>
<thead>
<tr>
<th>Specifying (LTL)</th>
</tr>
</thead>
</table>
| G(Sender.
| st = sent ∧ |
| Sender.
| message1 = 1 |
| →(MsgCh.
| output1 = 1) |

Note

Safety specification is True

F. Mari (mari.di.uniroma1.it)  MF.1: Modeling systems and specifying temporal properties  7 aprile 2014  39 / 40
Specification of properties

Safety

If the message bit 1 has been sent and the correct acknowledgement has been returned, then a 1 was indeed received by the receiver (the same holds for 0)

Specification (LTL)

$$G (Sender.\text{st} = \text{sent} \land Sender.\text{message1} = 1 \rightarrow \text{MsgCh.output1} = 1)$$

Note

Safety specification is True
Specification of properties

Liveness

*Messages* get through eventually

⇒ For any state there is inevitably a future state in which the current message has got through

Similarly, *acknowledgements* get through eventually
The alternating bit protocol

Specification of properties

Liveness

Messages get through eventually
⇒ For any state there is inevitably a future state in which the current message has got through
Similarly, acknowledgements get through eventually

Specification (LTL)

\[ \text{G} \ F \ (\text{Sender.} \ st = \text{sent}) \]
\[ \text{G} \ F \ (\text{Receiver.} \ st = \text{received}) \]

Note

Liveness specification is True
ARE THERE ANY QUESTIONS?

DO YOU EVER FEEL ALONE WHEN YOU'RE WITH PEOPLE?

I TRY TO.