Unit-3: Linear-time properties

B. Srivathsan

Chennai Mathematical Institute

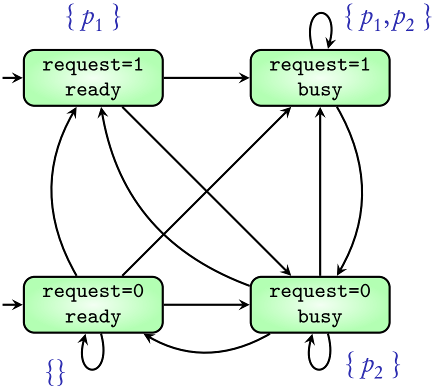
NPTEL-course

July - November 2015

Module 3:
Invariants

Atomic propositions AP $= \{ p_1, p_2 \}$

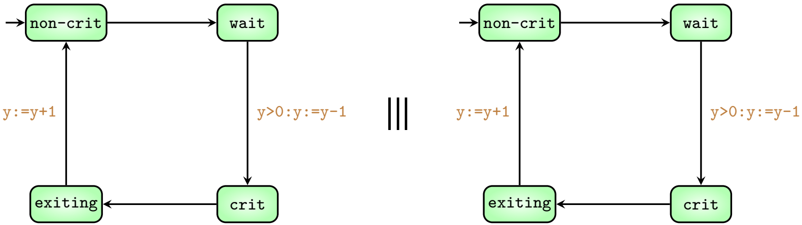
p_1: `request=1` $\quad$ p_2: `status=busy`



Atomic propositions $\mathrm{AP} = \{ p_1, p_2, p_3, p_4 \}$

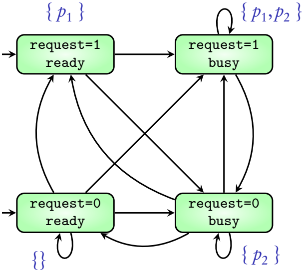
p_1: `pr1.location=crit` p_2: `pr1.location=wait`

p_3: `pr2.location=crit` p_4: `pr2.location=wait`



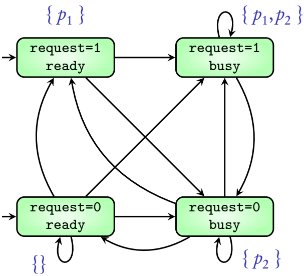
Atomic propositions $\mathrm{AP} = \{ p_1, p_2 \}$

p_1: `request=1` p_2: `status=busy`



Atomic propositions AP $= \{ p_1, p_2 \}$

p_1: `request=1` p_2: `status=busy`



AP-INF = set of **infinite words** over *PowerSet*(**AP**)

Property 1: p_1 is always true

$\{ A_0 A_1 A_2 \cdots \in$ AP-INF $\mid$ **each** A_i contains $p_1 \}$

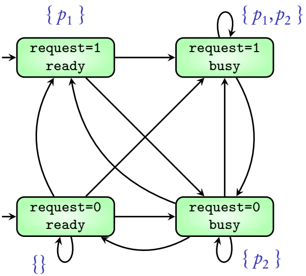
$\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\ldots$

$\{p_1\}\{p_1,p_2\}\{p_1\}\{p_1,p_2\}\{p_1\}\{p_1,p_2\}\ldots$

$\vdots$

Atomic propositions AP $= \{ p_1, p_2 \}$

p_1: `request=1` p_2: `status=busy`



AP-INF = set of **infinite words** over *PowerSet*(**AP**)

Property 1: p_1 is always true

$\{ A_0 A_1 A_2 \cdots \in$ AP-INF $\mid$ **each** A_i contains $p_1 \}$

$\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\ldots$

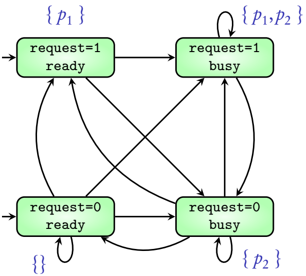
$\{p_1\}\{p_1,p_2\}\{p_1\}\{p_1,p_2\}\{p_1\}\{p_1,p_2\}\ldots$

$\vdots$

Property 1 is written as $\mathbf{G}\ p_1$

Atomic propositions AP = { p_1, p_2 }

p_1: `request=1` p_2: `status=busy`



AP-INF = set of **infinite words** over *PowerSet*(**AP**)

Property 1: p_1 is always true

{ $A_0 A_1 A_2 \cdots \in$ AP-INF | **each** A_i contains p_1 }

$\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\ldots$

$\{p_1\}\{p_1,p_2\}\{p_1\}\{p_1,p_2\}\{p_1\}\{p_1,p_2\}\ldots$

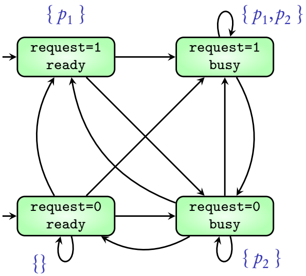
$\vdots$

Property 1 is written as **G** p_1

Above TS **does not satisfy** G p_1

Atomic propositions AP = $\{ p_1, p_2 \}$

p_1: `request=1` p_2: `status=busy`



AP-INF = set of **infinite words** over *PowerSet*(**AP**)

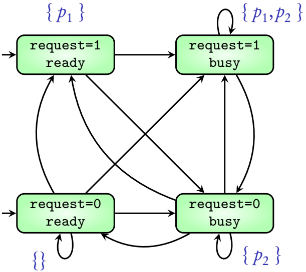
Property 2: $p_1 \wedge \neg p_2$ is always true

$\{ A_0 A_1 A_2 \cdots \in$ AP-INF $\mid$ **each** A_i satisfies $p_1 \wedge \neg p_2 \}$

$\{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \ldots$

Atomic propositions AP = $\{ p_1, p_2 \}$

p_1: `request=1` p_2: `status=busy`



AP-INF = set of **infinite words** over *PowerSet*(**AP**)

Property 2: $p_1 \wedge \neg p_2$ is always true

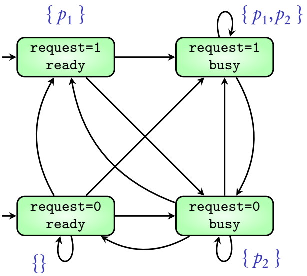
$\{ A_0 A_1 A_2 \cdots \in$ AP-INF | **each** A_i satisfies $p_1 \wedge \neg p_2 \}$

$\{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \ldots$

Property 2 is written as $\mathbf{G}\ p_1 \wedge \neg p_2$

Atomic propositions AP $= \{ p_1, p_2 \}$

p_1: `request=1` p_2: `status=busy`



AP-INF = set of **infinite words** over *PowerSet*(**AP**)

Property 2: $p_1 \wedge \neg p_2$ is always true

$\{ A_0 A_1 A_2 \cdots \in$ AP-INF $\mid$ **each** A_i satisfies $p_1 \wedge \neg p_2 \}$

$\{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \dots$

Property 2 is written as $\mathbf{G}\ p_1 \wedge \neg p_2$

Above TS **does not satisfy** $\mathrm{G}\ p_1 \wedge \neg p_2$

Invariants

AP-INF = set of **infinite words** over *PowerSet*(**AP**)

Property: ϕ is **always true**

(where ϕ is a boolean expression over AP)

$\{ A_0 A_1 A_2 \cdots \in$ AP-INF | **each** A_i satisfies ϕ $\}$

Invariants

AP-INF $=$ set of **infinite words** over *PowerSet*(**AP**)

Property: ϕ is **always true**
(where ϕ is a boolean expression over AP)

$\{ A_0 A_1 A_2 \cdots \in$ AP-INF $\mid$ **each** A_i satisfies ϕ $\}$

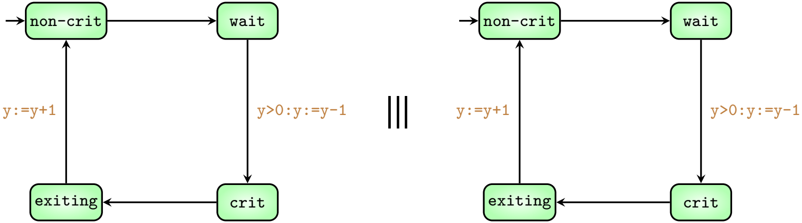
A property of the above form is called **invariant** property

It is written as G ϕ

Atomic propositions AP = $\{ p_1, p_2, p_3, p_4 \}$

p_1: `pr1.location=crit` p_2: `pr1.location=wait`

p_3: `pr2.location=crit` p_4: `pr2.location=wait`



Above TS satisfies invariant property $\mathbf{G}\ \neg (p_1 \wedge p_3)$

Algorithm

Input: A TS and property $\mathrm{G}\ \phi$

Output: Does TS satisfy invariant $\mathrm{G}\ \phi$?

Algorithm

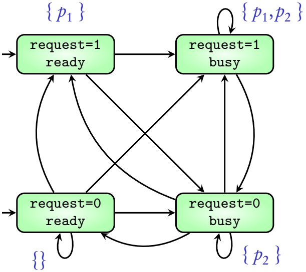
Input: A TS and property G ϕ

Output: Does TS satisfy invariant G ϕ?

A TS satisfies an invariant ϕ

if and only if

every **reachable state** of the TS satisfies ϕ



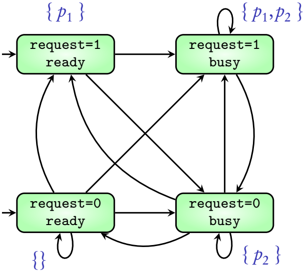
Property to check: $\mathbf{G}\, p_1$

set R, **stack** U, **bool** b
for all initial states s
  if $s \notin R$ **then**
    visit(s)
  endif
return b

procedure visit (states s)
  $push(s, U)$; $R := R \cup \{ s \}$
  while (U is not empty)
    $s' := top(U)$
    if $Post(s') \subseteq R$ **then**
      $pop(U)$
      $b := b \wedge (s' \models \phi)$
    else
      let $s'' \in Post(s') \setminus R$
      $push(s'', U)$
      $R := R \cup \{s''\}$
    endif
  endwhile

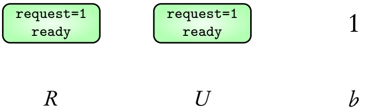
1

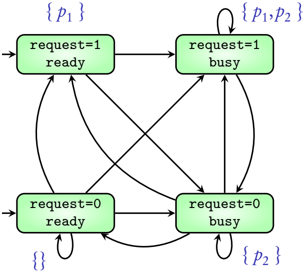
R U b



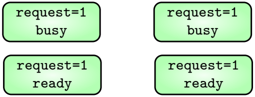
Property to check: $\mathbf{G}\, p_1$

```
set R, stack U, bool b
for all initial states s
        if s ∉ R then
            visit(s)
        endif
return b
procedure visit (states s)
        push(s, U);    R := R ∪ { s }
        while (U is not empty)
            s' := top(U)
            if Post(s') ⊆ R then
                pop(U)
                b := b ∧ (s' ⊨ ϕ)
            else
                let s'' ∈ Post(s') \ R
                push(s'', U)
                R := R ∪ {s''}
            endif
        endwhile
```





Property to check: G p_1

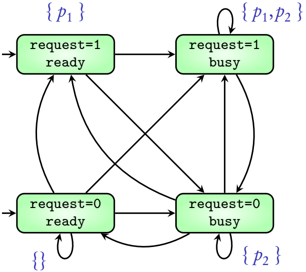


1

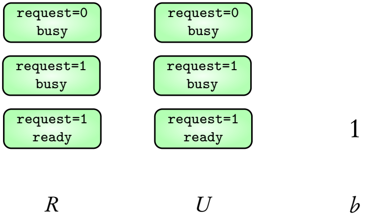
R U b

set R, **stack** U, **bool** b
for all initial states s
  if $s \notin R$ **then**
    visit(s)
  endif
return b

procedure visit (states s)
  $push(s, U)$; $R := R \cup \{ s \}$
  while (U is not empty)
    $s' := top(U)$
    if $Post(s') \subseteq R$ **then**
      $pop(U)$
      $b := b \wedge (s' \models \phi)$
    else
      let $s'' \in Post(s') \setminus R$
      $push(s'', U)$
      $R := R \cup \{s''\}$
    endif
  endwhile

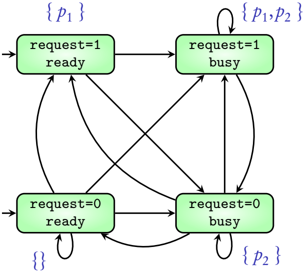


Property to check: $\mathbf{G}\ p_1$

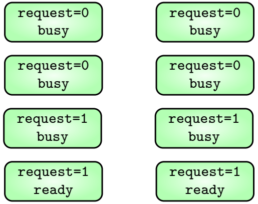


set R, **stack** U, **bool** b
for all initial states s
  if $s \notin R$ **then**
    visit(s)
  endif
return b

procedure visit (states s)
  $push(s, U)$; $R := R \cup \{ s \}$
  while (U is not empty)
    $s' := top(U)$
    if $Post(s') \subseteq R$ **then**
      $pop(U)$
      $b := b \wedge (s' \models \phi)$
    else
      let $s'' \in Post(s') \setminus R$
      $push(s'', U)$
      $R := R \cup \{s''\}$
    endif
  endwhile



Property to check: $\mathbf{G}\, p_1$

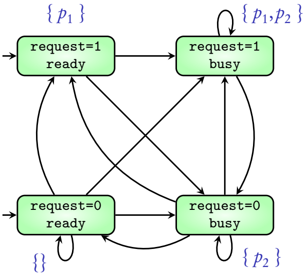


1

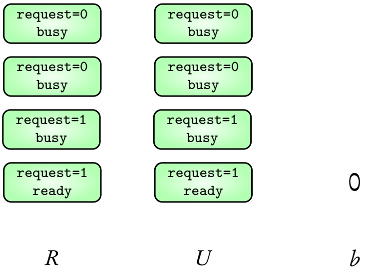
R U b

set R, **stack** U, **bool** b
for all initial states s
  if $s \notin R$ **then**
    visit(s)
  endif
return b

procedure visit (states s)
  $push(s, U)$; $R := R \cup \{ s \}$
  while (U is not empty)
    $s' := top(U)$
    if $Post(s') \subseteq R$ **then**
      $pop(U)$
      $b := b \wedge (s' \models \phi)$
    else
      let $s'' \in Post(s') \setminus R$
      $push(s'', U)$
      $R := R \cup \{s''\}$
    endif
  endwhile



Property to check: G p_1



set R, **stack** U, **bool** b

for all initial states s

  if $s \notin R$ **then**

    visit(s)

  endif

return b

procedure visit (states s)

  $push(s, U)$; $R := R \cup \{ s \}$

  while (U is not empty)

    $s' := top(U)$

    if $Post(s') \subseteq R$ **then**

      $pop(U)$

      $b := b \wedge (s' \models \phi)$

    else

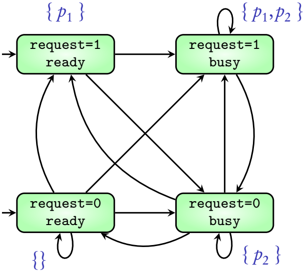
      let $s'' \in Post(s') \setminus R$

      $push(s'', U)$

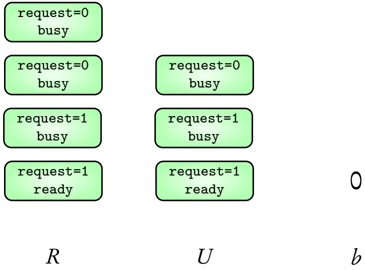
      $R := R \cup \{s''\}$

    endif

  endwhile

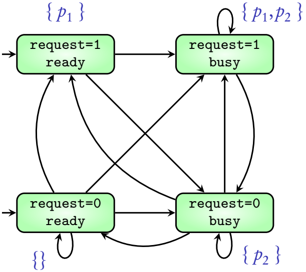


Property to check: $\mathbf{G}\ p_1$

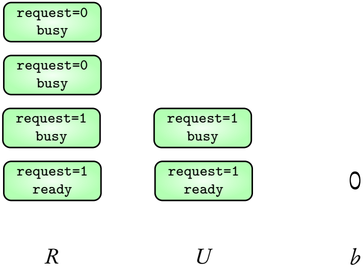



```
set R, stack U, bool b
for all initial states s
        if s ∉ R then
            visit(s)
        endif
return b

procedure visit (states s)
        push(s,U);    R := R ∪ { s }
        while (U is not empty)
            s′ := top(U)
            if Post(s′) ⊆ R then
                pop(U)
                b := b ∧ (s′ ⊨ ϕ)
            else
                let s″ ∈ Post(s′) \ R
                push(s″,U)
                R := R ∪ {s″}
            endif
        endwhile
```

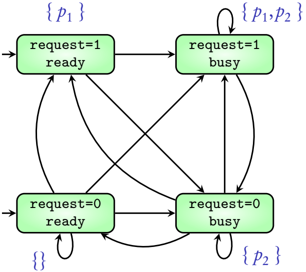


Property to check: G p_1

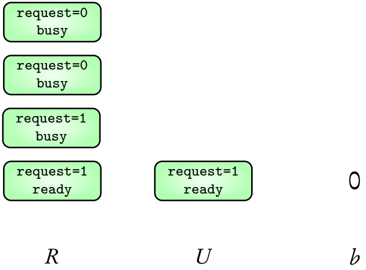


set R, **stack** U, **bool** b
for all initial states s
  if $s \notin R$ **then**
    visit(s)
  endif
return b

procedure visit (states s)
  $push(s, U)$; $R := R \cup \{ s \}$
  while (U is not empty)
    $s' := top(U)$
    if $Post(s') \subseteq R$ **then**
      $pop(U)$
      $b := b \wedge (s' \models \phi)$
    else
      let $s'' \in Post(s') \setminus R$
      $push(s'', U)$
      $R := R \cup \{s''\}$
    endif
  endwhile



Property to check: G p_1



set R, **stack** U, **bool** b

for all initial states s

  if $s \notin R$ **then**

    visit(s)

  endif

return b

procedure visit (states s)

  $push(s, U)$; $R := R \cup \{ s \}$

  while (U is not empty)

    $s' := top(U)$

    if $Post(s') \subseteq R$ **then**

      $pop(U)$

      $b := b \wedge (s' \models \phi)$

    else

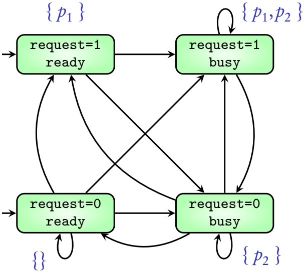
      let $s'' \in Post(s') \setminus R$

      $push(s'', U)$

      $R := R \cup \{s''\}$

    endif

  endwhile



Property to check: G p_1

request=0 busy

request=0 busy

request=1 busy

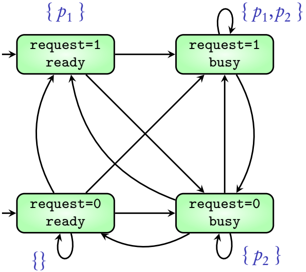
request=1 ready

0

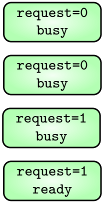
R U b

set R, **stack** U, **bool** b
for all initial states s
  if $s \notin R$ **then**
    visit(s)
  endif
return b

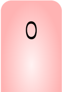
procedure visit (states s)
  $push(s, U)$; $R := R \cup \{ s \}$
  while (U is not empty)
    $s' := top(U)$
    if $Post(s') \subseteq R$ **then**
      $pop(U)$
      $b := b \wedge (s' \models \phi)$
    else
      let $s'' \in Post(s') \setminus R$
      $push(s'', U)$
      $R := R \cup \{s''\}$
    endif
  endwhile



Property to check: G p_1



Property **not satisfied**



R U b

```
set R, stack U, bool b
for all initial states s
        if s ∉ R then
            visit(s)
        endif
return b

procedure visit (states s)
        push(s, U);    R := R ∪ { s }
        while (U is not empty)
            s' := top(U)
            if Post(s') ⊆ R then
                pop(U)
                b := b ∧ (s' ⊨ ϕ)
            else
                let s'' ∈ Post(s') \ R
                push(s'', U)
                R := R ∪ {s''}
            endif
        endwhile
```

Invariants

G ϕ

Algorithm to check invariants