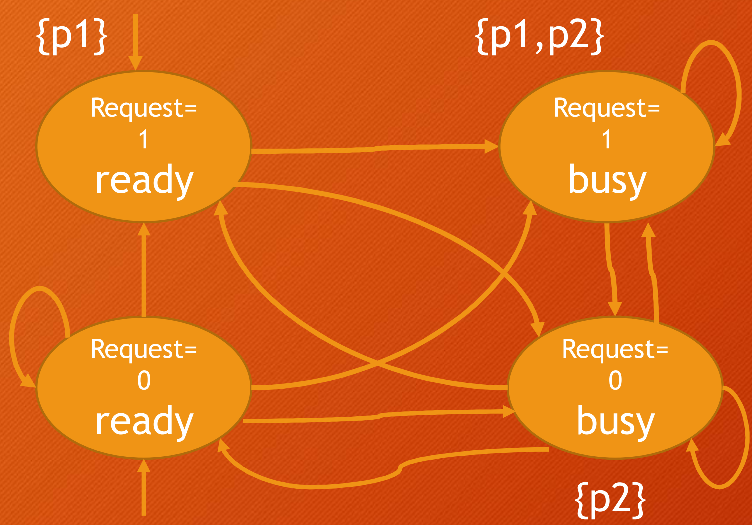
SE2003
Formal Methods in Software Engineering

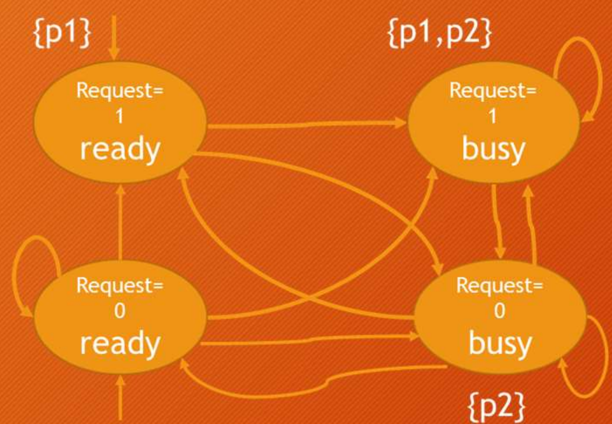
Spring-2024




```
MODULE main
VAR
    request : boolean;
    status : {ready, busy};
ASSIGN
    init(status)  := ready;
    next(status) := case
        request = TRUE : busy;
        TRUE : {ready, busy};
    esac;
```

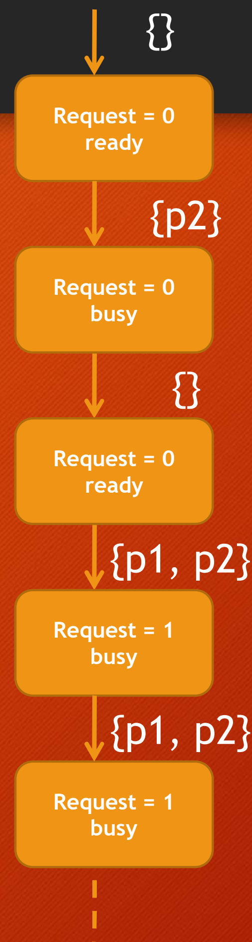
Atomic propositions

p1: (request = 1)   p2: (status = busy)

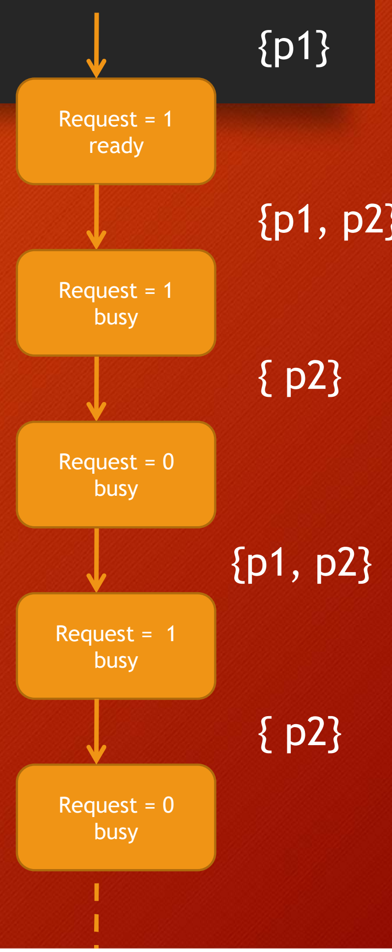


Atomic propositions
p1: (request = 1) p2: (status = busy)

Executions Trace



Executions Trace

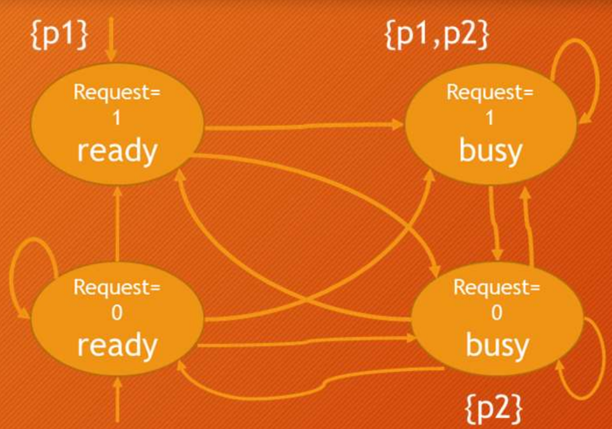


AP = {p1, p2, …, pk}

Power Set(AP) = {{}, {p1},…,{pk}
{p1,p2}, {p1,p3},…,{p1,pk}
…
{p1, p2,…,pk}}

Trace (Execution) is an infinite word over PowerSet(AP)

Traces(TS) is the {Trace(σ) | σ is an execution of the TS}



Atomic propositions
p1: (request = 1) p2: (status = busy)

Traces
{}{}{}{}{}....
{}{p1}{p2}{p2}...
{p1}{p1,p2}{p2}{p1,p2}...
{} {p1,p2} {p1,p2} {p1,p2}...
.
.
.

Traces of a TS describes its behavior with respect to the atomic propositions

Property of a system?

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

Property 1: p1 is always true

$\{A_0A_1A_2\ldots \in$ AP-INF | each A_i contains p1$\}$
{p1}{p1}{p1}…
{p1}{p1,p2}{p1}{p1,p2}…

.
.
.

Property 2: p1 is true at least one and p2 is always true
$\{A_0A_1A_2\ldots \in$ AP-INF | exists A_i containing p1 and every A_i contains p2$\}$
{p2}{p1p2}{p2}{p2}{p2}{p2}{p1,p2}{p2}…
{p1,p2}{p2}{p2}…

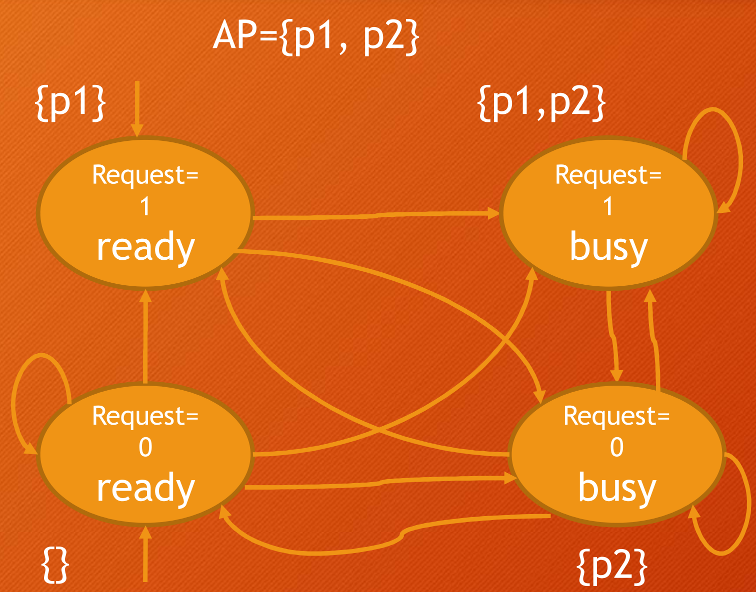
.
.
.

Property of a system?

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

A property over AP is a subset of AP-INF

When does a system satisfies a property?



Property
G p1 ?

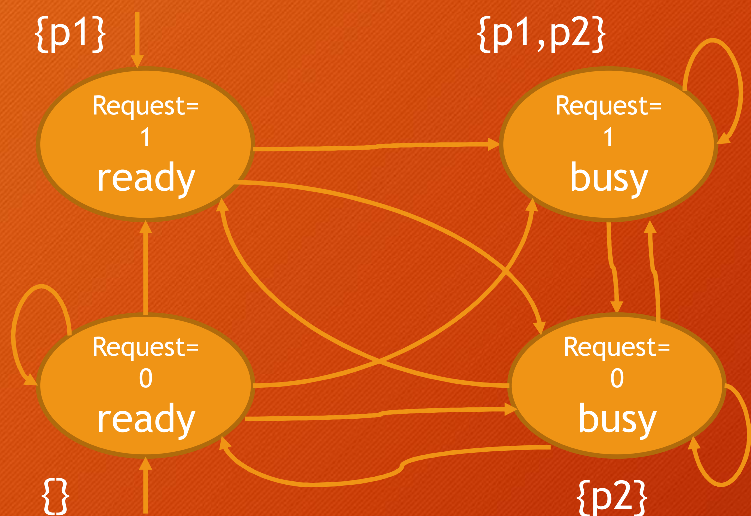
Transition system TS satisfies a property p if
Traces(TS) ⊆ p

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

It is a set of words also called linear time property

Invariants

AP={p1, p2}
p1: Request = 1 p2: status = busy



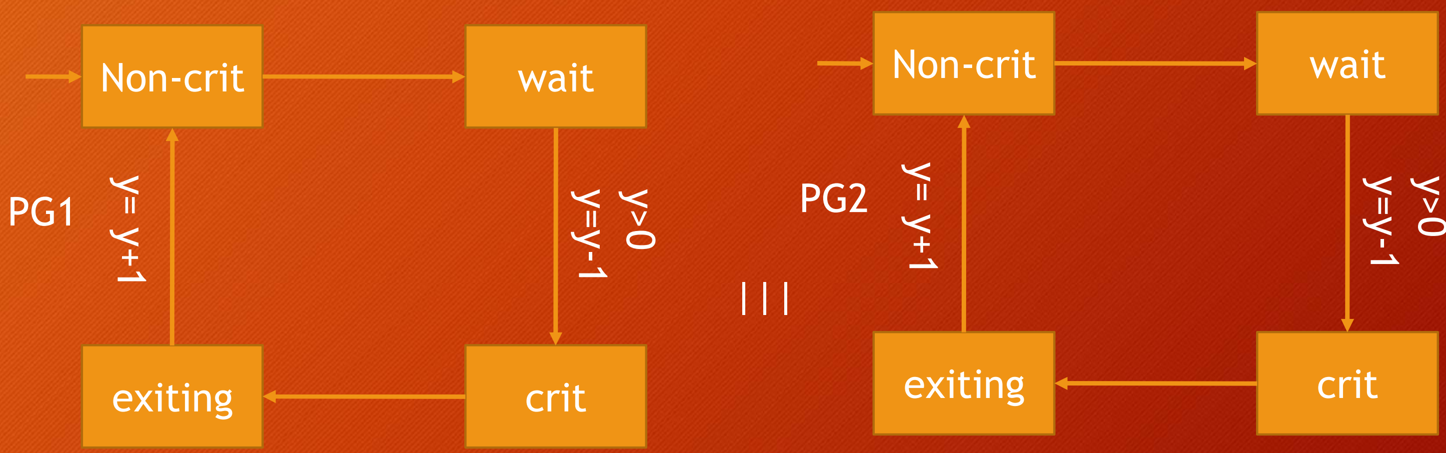
AP = {p1, p2, p3, p4}

p1: PG1.location = crit

p2: PG1.location = wait

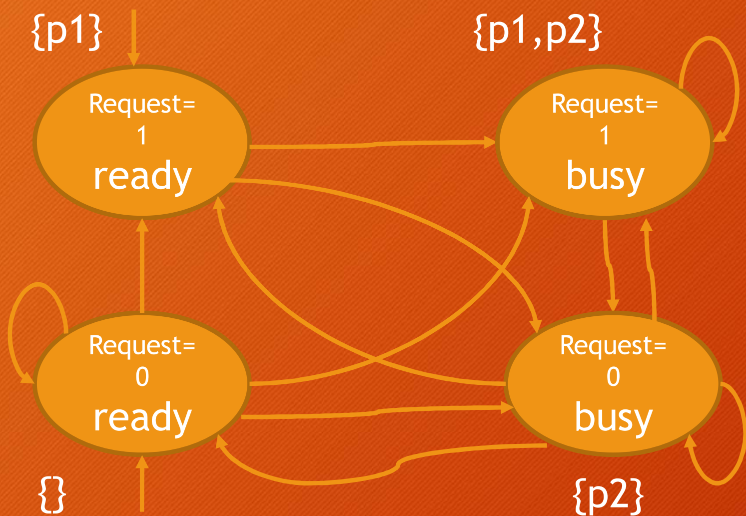
p3: PG2.location = crit

p4: PG2.location = wait



Invariants

AP={p1, p2}
p1: Request = 1 p2: status -busy



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

Property 1: p1 is always true

$\{A_0A_1A_2\ldots \in$ AP-INF | each A_i contains p1$\}$
{p1}{p1}{p1}...
{p1}{p1,p2}{p1}{p1,p2}...
.
.
.

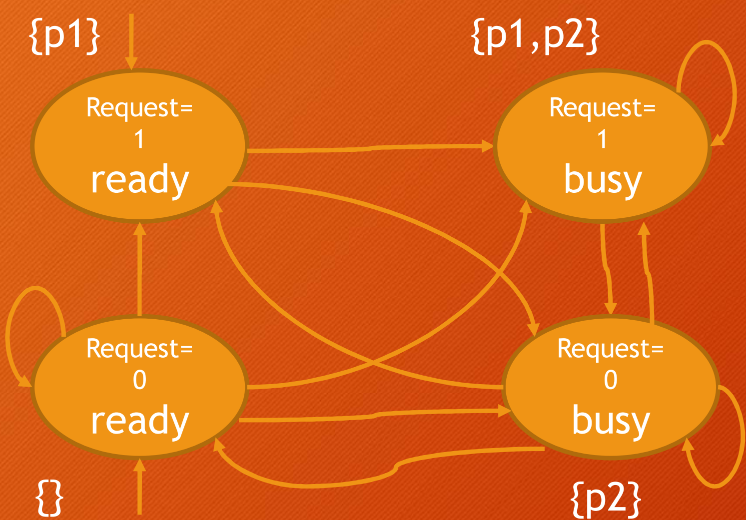
Property p1 is written as G p1

TS doe not satisfy G p1

Invariants

AP={p1, p2}

p1: Request = 1 p2: status -busy



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

Property 1: p1 ^ not p2 is always true

{$A_0A_1A_2$… ϵ AP-INF | each A_i contains p1 ^ not p2}

{p1}{p1}{p1}…

Property p1 is written as G p1 ^ ! p2

TS doe not satisfy G p1 ^ ! p2

Invariant

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

Property 1: φ is always true
where φ is a Boolean expression over AP

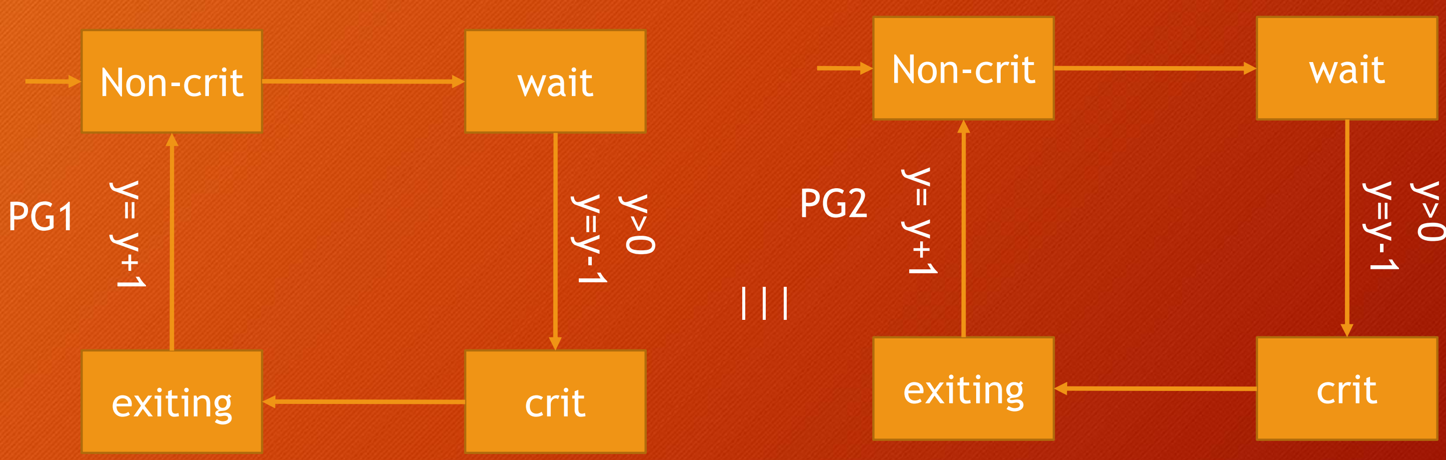
$\{A_0 A_1 A_2 \ldots \in \text{AP-INF} \mid \text{each } A_i \text{ satisfies } \varphi\}$

A property of the above form is called invariant property
it is written G φ

AP = {p1, p2, p3, p4}

p1: PG1.location = crit   p2: PG1.location = wait

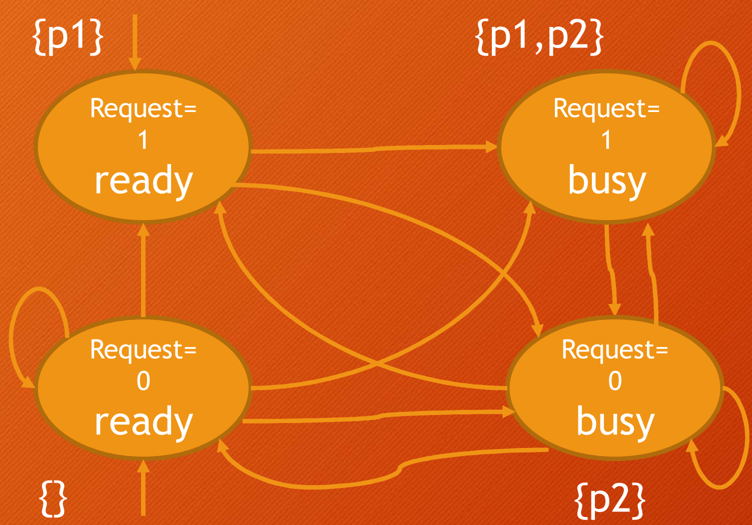
p3: PG2.location = crit   p4: PG2.location = wait



G ! (p1 ^ p3) ???

Safety properties

AP={p1, p2}
p1: Request = 1 p2: status = busy



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

Property if p1 is true then next step is p2 is true
$\{A_0A_1A_2\ldots \in$ AP-INF | IF A_i contains p1
then A i+1 contains p2}

{p1}{p2}{p1}{p1,p2}{p2}{{p1}{p1,p2}…
{p2}{p2}{p2}…
{}{}{}{}….
.
.
.

Property is written as G(p1 → X p2) ???

G (p1→XXp2)

Always: if p1 is true then in the next-to-next step p2 is true

F(p1 ^ X ! p1)

Somewhere : p1 is true and in the next step it becomes false

G(Xp2 → p1)

Always: if p2 is true then in the previous step p1 is true