Course 02158

Invariants

Hans Henrik Løvengreen

DTU Compute

Invariants

- Let I be a *state predicate* over the global state of a concurrent program P
- I is an *invariant* of P if it holds at any time in any execution
- Safety properties can be expressed as invariants (+ *history variables*)

Syntax

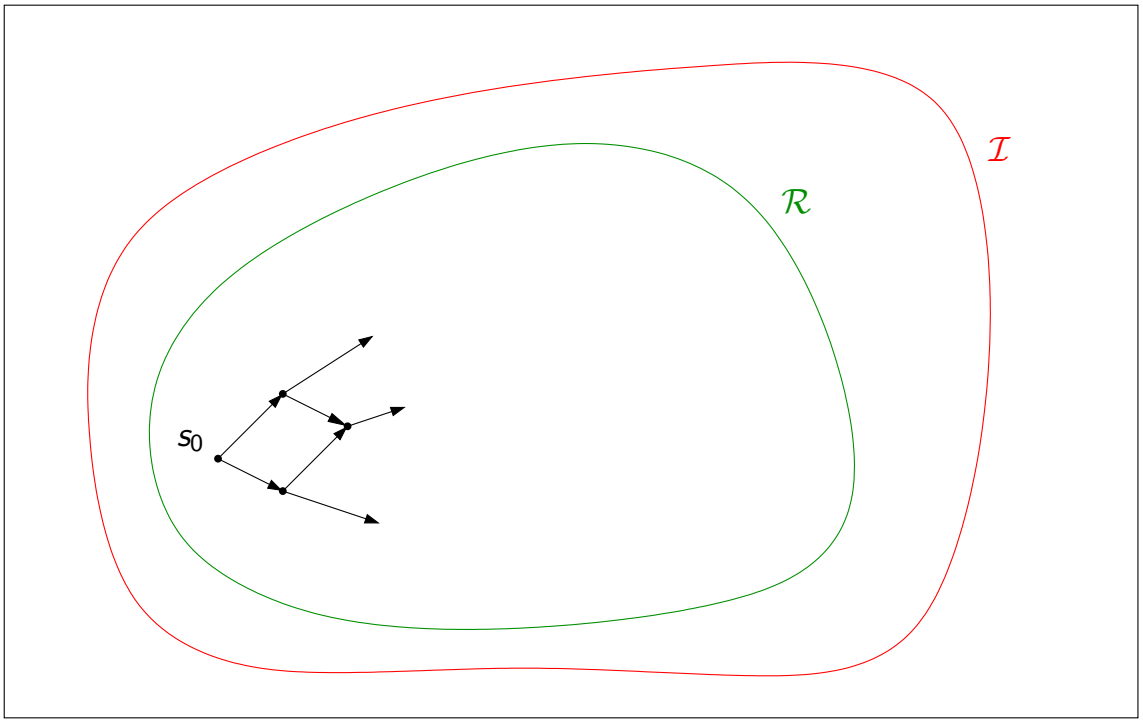
- May refer to the value of *global variables*, e.g. $x = 5$ or $\neg done$.
- May refer to the *control state* of each process, e.g. $\pi_i = l$.
- Abbreviations (for action a in process P_i)

$$\begin{aligned} at\ l &\triangleq \pi_i = l \\ at\ a &\triangleq \pi_i = preloc(a) \end{aligned}$$

- For a statement $S : a_1; a_2; \ldots; a_n$:

$$\begin{aligned} in\ S &\triangleq at\ a_1 \vee at\ a_2 \vee \ldots \vee a_n \\ at\ S &\triangleq at\ a_1 \\ after\ S &\triangleq after\ a_n \end{aligned}$$

Invariance Notion



Inductive Invariance Technique

- Let there be given a concurrent program with atomic actions
- A state predicate I is said to be *inductive* if
  - I holds for the initial state.
  - Any atomic action a of the program *preserves* I, i.e. for any state s for which I is satisfied, it is either the case that:
    a) a cannot be executed in s, or
    b) the execution of a in state s results in a state s' that again satisfies I.
- If I is inductive, it is an invariant of the program.
- To show a), *known invariants* and may be used.
- If I fails to be inductive, it may be *strenghtened*.

Critical Region with Shared Variables

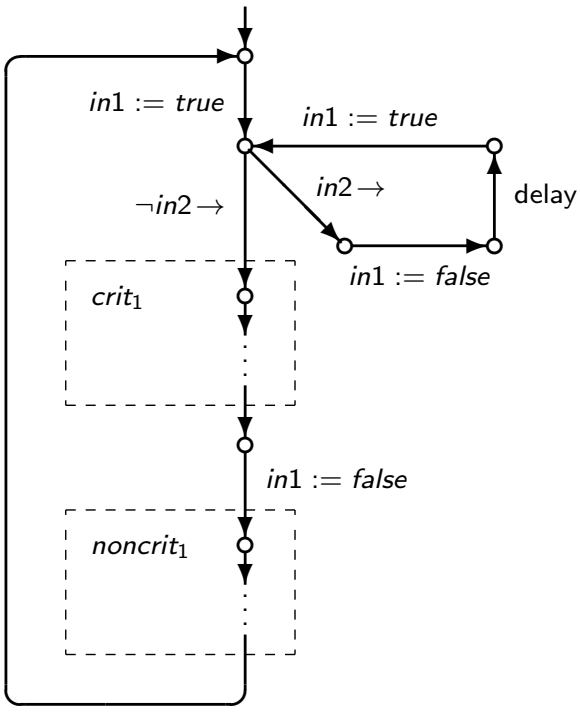
Fine-grained attempt III (Back-off)

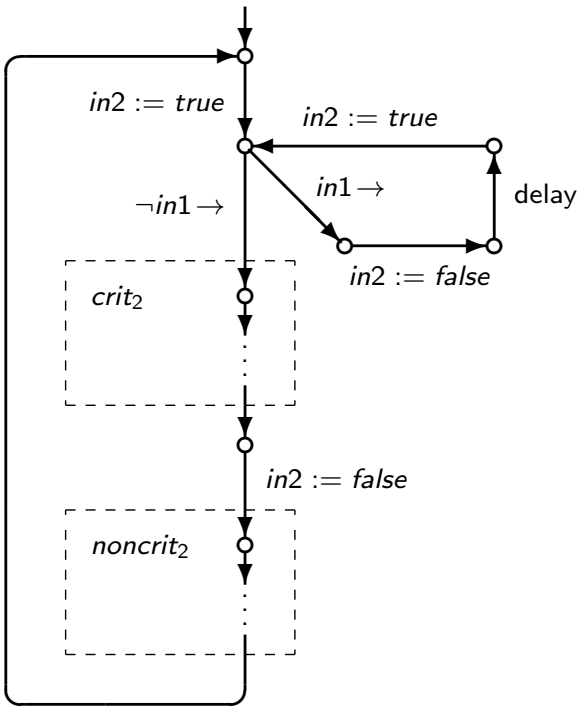
- **var** $in1, in2 : bool := false$;

process P_1
loop
$in1 := true$;
while $in2$ **do** { $in1 := false$;
delay;
$in1 := true$}
critical section$_1$;
$in1 := false$;
noncritical section$_1$
end loop

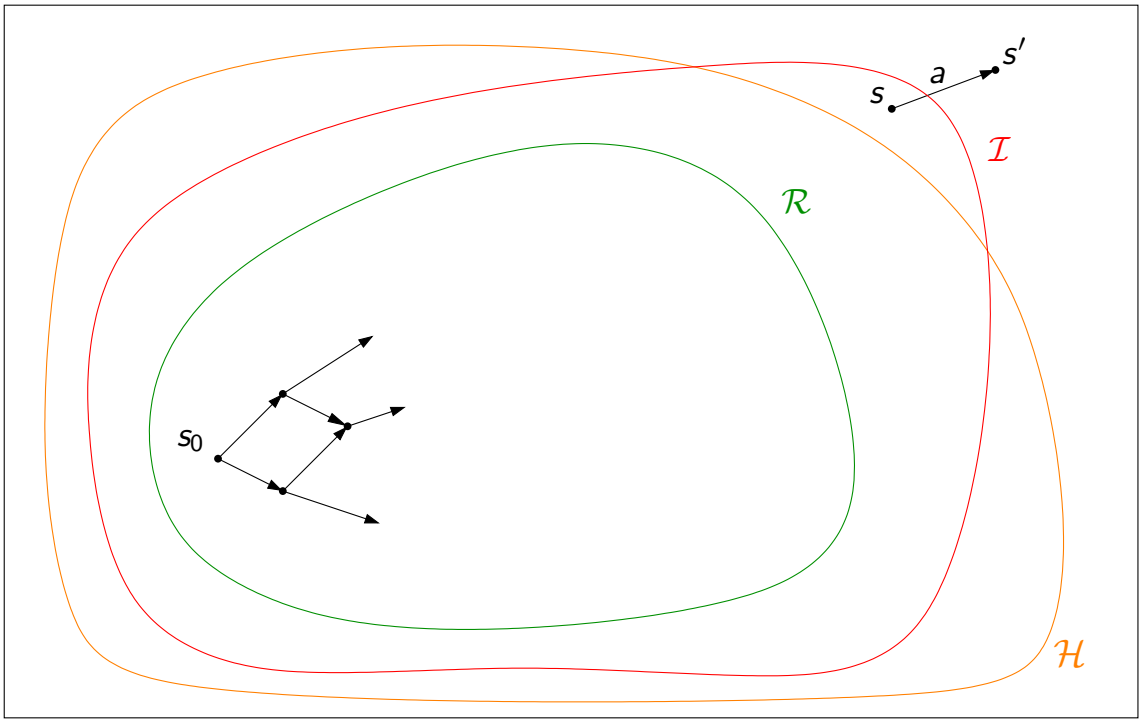
process P_2
loop
$in2 := true$;
while $in1$ **do** { $in2 := false$;
delay;
$in2 := true$}
critical section$_2$;
$in2 := false$;
noncritical section$_2$
end loop

Example: Back-off attempt





Invariance Using Auxiliary Invariant



Backoff attempt — Invariants

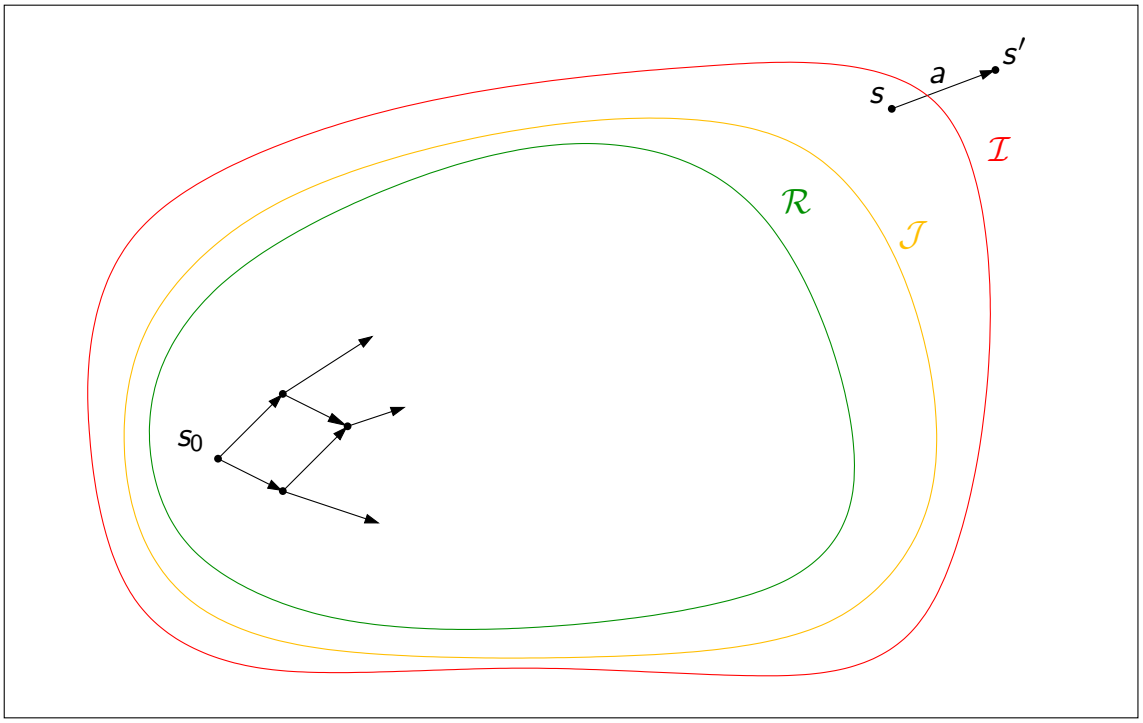
- Mutual Exlusion

$$I \triangleq \neg(in\ crit_1 \wedge in\ crit_2)$$

- Local invariants

$$I_1 \triangleq in\ crit_1 \Rightarrow in_1$$
$$I_2 \triangleq in\ crit_2 \Rightarrow in_2$$

Invariance Using Strengthened Invariant



Example: Peterson's algorithm

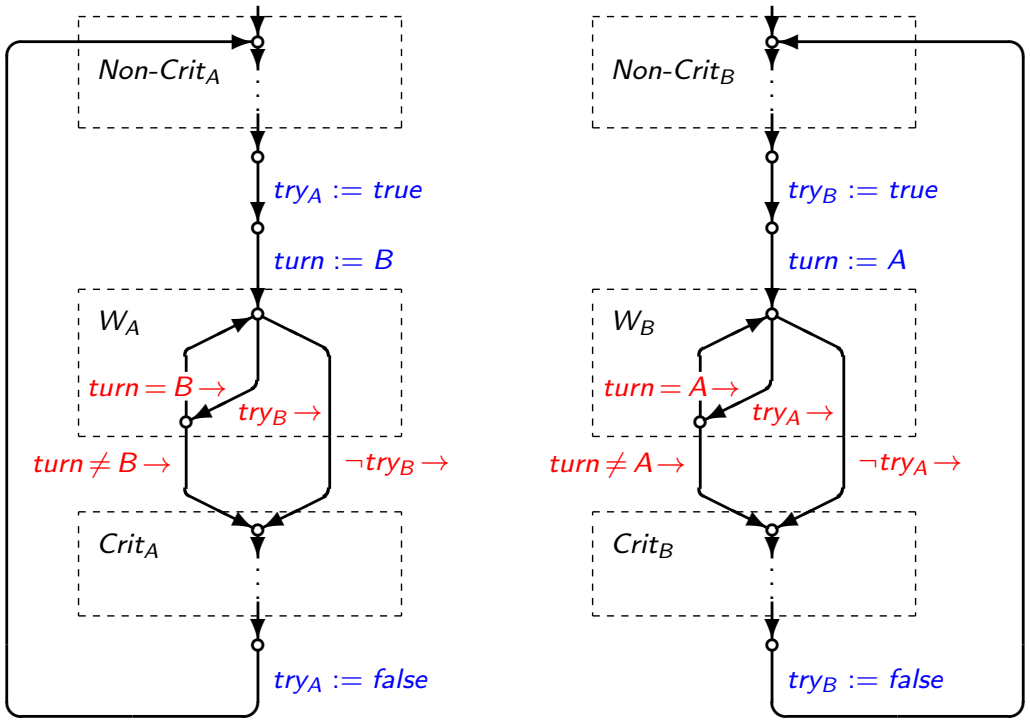
- **var** try_A, try_B : *boolean*;
  $turn$: (A, B);

  $try_A := false$; $try_B := false$; $turn := A$;

process P_A
  repeat
    $Non\text{-}Crit_A$;
    $try_A := true$;
    $turn := B$;
W_A: **while** $try_B \wedge turn = B$ **do** ;
    $Crit_A$;
    $try_A := false$
  forever

process P_B
  repeat
    $Non\text{-}Crit_B$;
    $try_B := true$;
    $turn := A$;
W_B: **while** $try_A \wedge turn = A$ **do** ;
    $Crit_B$;
    $try_B := false$
  forever

Example: Peterson's algorithm



Peterson's algorithm – Invariants

- Mutual Exlusion

$$I \triangleq \neg(in\ Crit_A \wedge in\ Crit_B)$$

- Local invariants

$$I_1 \triangleq in\ W_A..Crit_A \Rightarrow try_A$$
$$I_2 \triangleq in\ W_B..Crit_B \Rightarrow try_B$$

- Value invariants

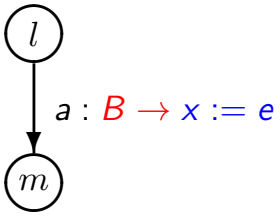
$$I_3 \triangleq turn = A \vee turn = B$$

- Auxiliary invariants

$$I_a \triangleq in\ W_A \wedge in\ Crit_B \Rightarrow turn = B$$
$$I_b \triangleq in\ W_B \wedge in\ Crit_A \Rightarrow turn = A$$

Pre/post Notation

- Let a be a given action:



- Let $(\ldots)'$ refer to the state *after* execution of a
- Action a *preserves* predicate I iff

$$I \wedge B \wedge at\ l \wedge \text{"effect of } a\text{"} \Rightarrow I'$$

where the effect is formally given by

$$\begin{array}{ll} x' = e \wedge at\ m', & \\ y' = y & \text{for all variables } y \text{ different from } x, \\ q' = q & \text{for all control predicates } q \text{ not involving } l \text{ or } m. \end{array}$$

Pre/post Notation — Example

Peterson's algorithm

- Prove: $I_a \triangleq in\ W_A \wedge in\ Crit_B \Rightarrow turn = B$ is an invariant
- Inititially: $at\ Non\text{-}Crit_A \Rightarrow \neg in\ W_A \Rightarrow I_a$

Process	Action	Proof	Argument
P_A	$turn := B$	${I_a}'$	$turn' = B$
P_B	$turn := A$	$\neg in\ {Crit_B}'$ ${I_a}'$	$at\ {W_B}'$
	$\neg try_A \to$	$\neg in\ W_A$ $\neg in\ {W_A}'$ ${I_a}'$	Cond., I_1 $in\ {W_A}' = in\ W_A$
	$turn \neq A \to$	$turn = B$ $turn' = B$ ${I_a}'$	Cond., I_3 $turn' = turn$