Course 02158

Temporal Logic

Hans Henrik Løvengreen

DTU Compute

Temporal Logic

- A *logic* that talks about temporal relations between situations

Formulas

- A temporal formula talks about a sequence of states

$$\sigma = s_0, s_1, s_2, \ldots$$

- The formula is relative to a given position i: *"now"*
- A *state predicate* p talks about "now"
- Temporal operators:

$\Box P$	P holds *always* (from now)
$\Diamond P$	P holds *eventually* (from now)

- Combinations:

$\Box\Diamond P$	P holds *over and over again*
$\Diamond\Box P$	P holds *eventually forever*

Temporal Logic Semantics

- Holds at specific point i of $\sigma = s_0, s_1, s_2, \ldots$.

$$
\begin{array}{lcl}
(\sigma, i) \models p & \triangleq & [\![p]\!]s_i \\
(\sigma, i) \models P \wedge Q & \triangleq & (\sigma, i) \models P \;\text{ and }\; (\sigma, i) \models Q \\
(\sigma, i) \models P \vee Q & \triangleq & (\sigma, i) \models P \;\text{ or }\; (\sigma, i) \models Q \\
(\sigma, i) \models \neg P & \triangleq & (\sigma, i) \not\models P \\
(\sigma, i) \models \Box P & \triangleq & \forall j \geq i : (\sigma, j) \models P \\
(\sigma, i) \models \Diamond P & \triangleq & \exists j \geq i : (\sigma, j) \models P \\
(\sigma, i) \models \forall x : P & \triangleq & (\sigma, i) \models P[c/x] \;\text{ for all constants }\; c \\
(\sigma, i) \models \exists x : P & \triangleq & (\sigma, i) \models P[c/x] \;\text{ for some constant }\; c
\end{array}
$$

- Holds for execution: $\sigma \models P \triangleq (\sigma, 0) \models P$
- Holds for program: $\mathbf{Prog} \models P \triangleq \forall \sigma \in \mathit{Exec}(\mathbf{Prog}) : \sigma \models P$
- Holds: $\models P \triangleq \forall \sigma : \sigma \models P$

Some Temporal Laws

- $\neg\Box P \Leftrightarrow \Diamond\neg P$
  $\Box P \Rightarrow P$
  $P \Rightarrow \Diamond P$

  $\Box(P \Rightarrow Q) \Rightarrow (\Box P \Rightarrow \Box Q)$
  $\Box P \wedge \Diamond Q \Rightarrow \Diamond(P \wedge Q)$
- $\Box(P \wedge Q) \Leftrightarrow (\Box P \wedge \Box Q)$
  $\Diamond(P \vee Q) \Leftrightarrow (\Diamond P \vee \Diamond Q)$

  $\Diamond\Box(P \wedge Q) \Leftrightarrow (\Diamond\Box P \wedge \Diamond\Box Q)$
  $\Box\Diamond(P \vee Q) \Leftrightarrow (\Box\Diamond P \vee \Box\Diamond Q)$
- $\Box\Box P \Leftrightarrow \Box P$
  $\Diamond\Diamond P \Leftrightarrow \Diamond P$

  $\Box\Diamond\Box P \Leftrightarrow \Diamond\Box P$
  $\Diamond\Box\Diamond P \Leftrightarrow \Box\Diamond P$
- $\Box(P \vee Q) \Rightarrow (\Box P \vee \Diamond Q)$

Dancing Puzzle

I'm never gonna not dance again

$$\Box\neg \quad \Diamond \quad \neg \quad \Diamond Dance$$

$$\Updownarrow$$

$$\Box\Box\neg\neg\Diamond Dance$$

$$\Updownarrow$$

$$\Box\Box\Diamond Dance$$

$$\Updownarrow$$

$$\Box\Diamond Dance$$

The Leads-to Operator

Definition

- $P \leadsto Q \;\triangleq\; \Box(P \Rightarrow \Diamond Q)$ "P always leads to Q"

Some rules

- $\Box(P \Rightarrow Q) \Rightarrow (P \leadsto Q)$
- $(P \leadsto Q) \wedge (Q \leadsto R) \Rightarrow (P \leadsto R)$
- $(P \leadsto R) \wedge (Q \leadsto R) \Rightarrow ((P \vee Q) \leadsto R)$
- $(P \leadsto Q) \vee (P \leadsto R) \Rightarrow (P \leadsto (Q \vee R))$

Liveness of Critical Regions

- **Obligingness:**

$$\mathit{in\ entry}_i \wedge (\forall j \neq i : \Box \mathit{in\ noncrit}_j) \leadsto \mathit{at\ crit}_i$$

- **Resolution:**

$$(\exists i : \mathit{in\ entry}_i) \leadsto (\exists j : \mathit{at\ crit}_j)$$

- **Fairness:**

$$\mathit{in\ entry}_i \leadsto \mathit{at\ crit}_i$$

Fair Scheduling

- A *schedule* determines the choices among enabled actions
- Given $a : \langle\, B \rightarrow S \,\rangle$ in some process P_i

Weak Fairness (WF)

- $\Box(\mathit{at}\ a \wedge B) \leadsto \mathit{after}\ a$
- Also known as *fair process exectution*
- Every process must be visited infinitely often
- Usually implemented by round-robin scheduling

Strong Fairness (SF)

- $\Box\Diamond(\mathit{at}\ a \wedge B) \leadsto \mathit{after}\ a$
- Can be implemented using WF + queues
- Hard to implement in general

Fairness: Example

- **var** *go* : *bool* := *true*;
  $\quad x : integer := 0;$

process P_1
  while *go* **do**
    $a\colon\ \langle\, x := x + 1 \,\rangle$

process P_2
  $b\colon\ \langle\,$ **await** x **mod** $3 = 0;\ \ go := false\,\rangle$

$$\begin{tikzcd}
0 \arrow[r, "a"] \arrow[dr, "b"'] & 1 \arrow[r, "a"] & 2 \arrow[r, "a"] & 3 \arrow[r, "a"] \arrow[dr, "b"'] & 4 \arrow[r, "a"] & 5 \arrow[r, "a"] & 6 \arrow[r, "a"] \arrow[dr, "b"'] & 7 \arrow[r, "a"] & 8 \arrow[r, "a"] & 9 \arrow[r, "a"] \arrow[dr, "b"'] & 10 \\
 & {} & & & {} & & & {} & & & {}
\end{tikzcd}$$

- Weak fairness does **not** guarantee termination.
- Strong fairness guarantees termination.

Liveness Properties of Program Constructs

Basic progress examples

- $l\colon \langle\, x := e \,\rangle$ — $at\ l \leadsto after\ l$
- $l\colon$ **if** B **then** S_1 **else** S_2 — $at\ l \leadsto at\ S_1 \vee at\ S_2$

Derived rules

- For any S

$$in\ S \Rightarrow (\Box in\ S \vee \Diamond after\ S)$$

- For $l\colon (S_1; S_2)$

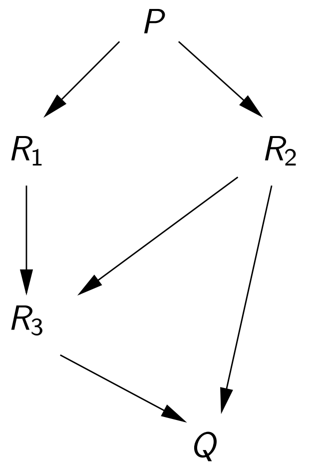
$$\frac{at\ S_1 \leadsto after\ S_1 \qquad at\ S_2 \leadsto after\ S_2}{at\ l \leadsto after\ l}$$

- For $w\colon$ **while** B **do** S

$$\frac{in\ S \leadsto after\ S}{in\ w \wedge \Box\neg B \leadsto after\ w}$$

Proof Lattices

-

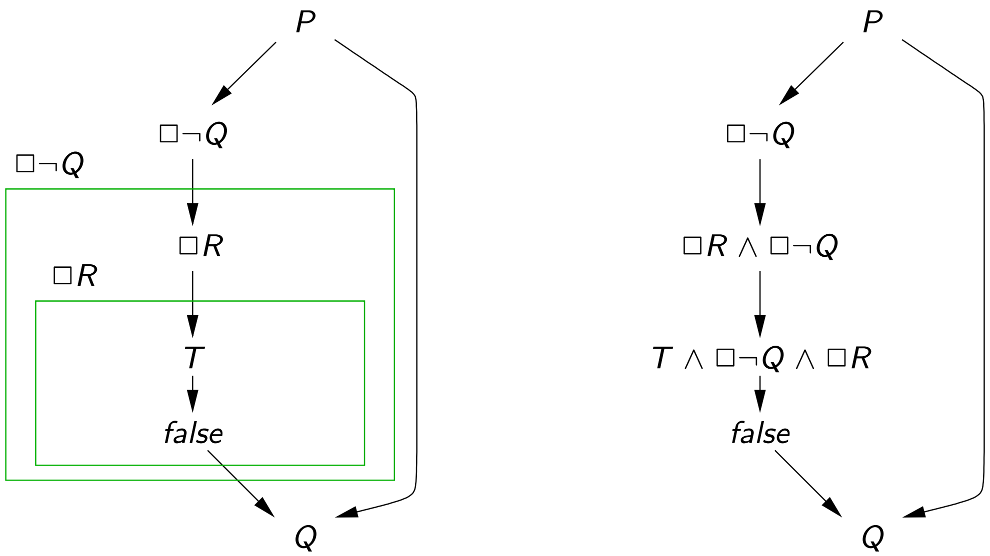


represents the formulas

$$\left\{ \begin{array}{l} P \leadsto R_1 \lor R_2 \\ R_1 \leadsto R_3 \\ R_2 \leadsto Q \lor R_3 \\ R_3 \leadsto Q \end{array} \right.$$

Proof Lattices — frames

-



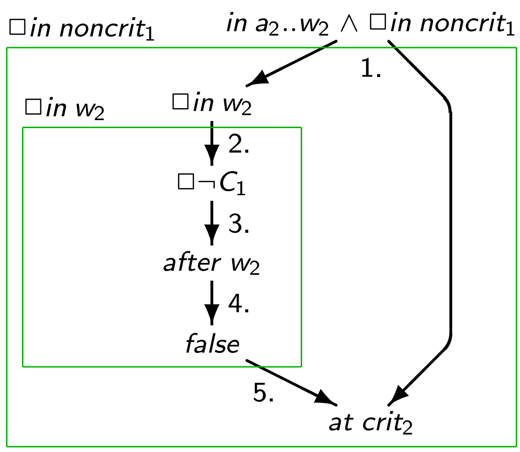
Example: Asymmetric Back-off

- **var** $C_1, C_2 : bool := false$;

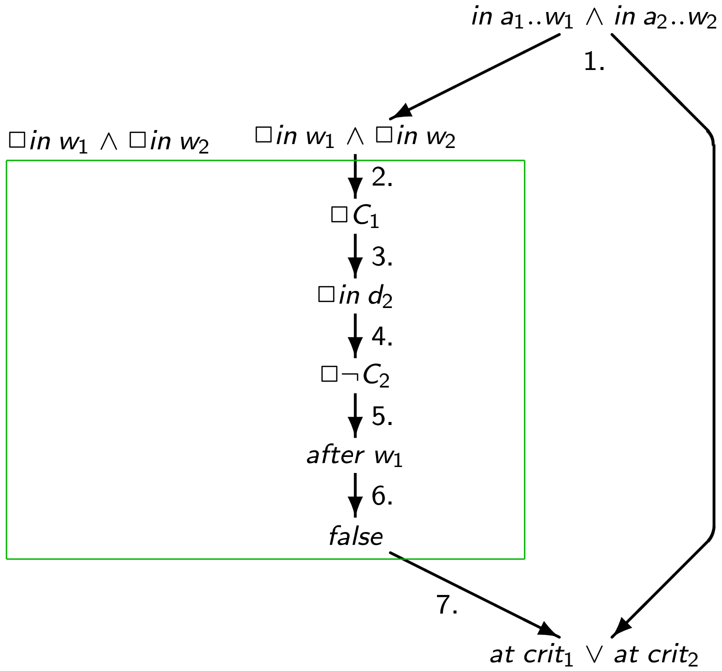
process P_1
  repeat
    $noncrit_1$;
a_1: $C_1 := true$;
w_1:   **while** C_2 **do skip**;
    $crit_1$;
x_1: $C_1 := false$
  forever

process P_2
  repeat
    $noncrit_2$;
a_2: $C_2 := true$;
w_2:   **while** C_1 **do**
      { b_2: $C_2 := false$;
        d_2: **while** C_1 **do skip**;
        e_2: $C_2 := true$ };
    $crit_2$;
x_2: $C_2 := false$
  forever

Example — Obligingness for P_2



Example — Resolution



Example — Fairness for P_1