

# Part 3: Safety and liveness

## Safety vs. liveness

Safety: something “bad” will never happen

Liveness: something “good” will happen  
(but we don’t know when)

## Safety vs. liveness for sequential programs

Safety: the program will never produce a wrong result (“partial correctness”)

Liveness: the program will produce a result (“termination”)

## Safety vs. liveness for sequential programs

Safety: the program will never produce a wrong result (“partial correctness”)

Liveness: the program will produce a result (“termination”)

## Safety vs. liveness for state-transition graphs


**Safety:** those properties whose violation  
always has a finite witness

(“if something bad happens on an infinite run, then it happens already on some finite prefix”)

**Liveness:** those properties whose violation  
never has a finite witness

(“no matter what happens along a finite run, something good could still happen later”)

This is much  
easier.



Safety: the properties that can be  
checked on finite executions

Liveness: the properties that cannot be  
checked on finite executions

(they need to be checked on  
infinite executions)

## Example: Mutual exclusion

It cannot happen that both processes are in their critical sections simultaneously.

## Example: Mutual exclusion

It cannot happen that both processes are in their critical sections simultaneously.

Safety



## Example: Bounded overtaking

Whenever process P1 wants to enter the critical section, then process P2 gets to enter at most once before process P1 gets to enter.

## Example: Bounded overtaking

Whenever process P1 wants to enter the critical section, then process P2 gets to enter at most once before process P1 gets to enter.

Safety

## Example: Starvation freedom

Whenever process P1 wants to enter the critical section, provided process P2 never stays in the critical section forever, P1 gets to enter eventually.

## Example: Starvation freedom

Whenever process P1 wants to enter the critical section, provided process P2 never stays in the critical section forever, P1 gets to enter eventually.

Liveness

## Example: Starvation freedom

Whenever process P1 wants to enter the critical section, **provided process P2 never stays in the critical section forever**, P1 gets to enter eventually.

**Liveness**

## LTL (Linear Temporal Logic)

- safety & liveness

- linear time

[Pnueli 1977; Lichtenstein & Pnueli  
1982]

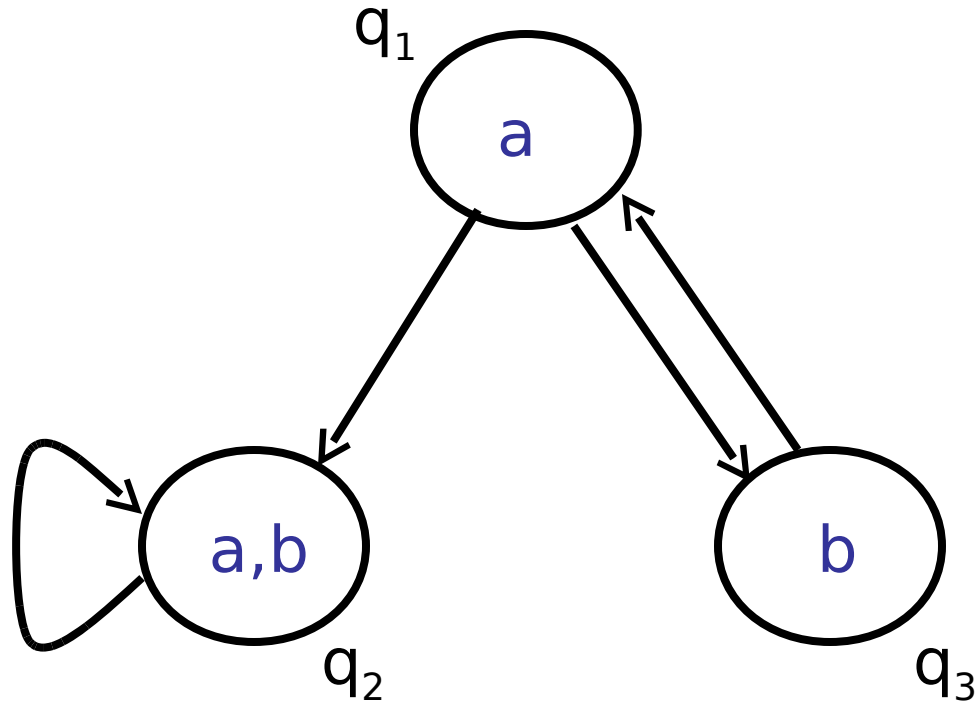
# LTL Syntax

$\varphi ::= a \mid \varphi \wedge \varphi \mid \neg \varphi \mid O \varphi \mid \varphi U \varphi$

## LTL Model

infinite trace  $t = t_0 t_1 t_2 \dots$   
(sequence of  
observations)





Run:  $q_1 \rightarrow q_3 \rightarrow q_1 \rightarrow q_3 \rightarrow q_1 \rightarrow q_2 \rightarrow q_2 \rightarrow$

Trace:  $a \rightarrow b \rightarrow a \rightarrow b \rightarrow a \rightarrow a,b \rightarrow a,b \rightarrow$

Language of **deadlock-free** state-transition graph  $K$  at state  $q$  :

$L(K,q)$  = set of infinite traces of  $K$  starting at  $q$

$(K,q) \models^\forall \varphi$       iff      for all  $t \in L(K,q)$ ,  $t \models$   
 $\varphi$

$(K,q) \models^\exists \varphi$       iff      exists  $t \in L(K,q)$ ,  $t \models$   
 $\varphi$

## LTL Semantics

$t \models a$  iff  $a \in t_0$

$t \models \varphi \wedge \psi$  iff  $t \models \varphi$  and  $t \models \psi$

$t \models \neg\varphi$  iff not  $t \models \varphi$

$t \models \bigcirc \varphi$  iff  $t_1 t_2 \dots \models \varphi$

$t \models \varphi \cup \psi$  iff exists  $n \geq 0$  s.t.  
1. for all  $0 \leq i < n$ ,  $t_i t_{i+1} \dots \models \varphi$   
2.  $t_n t_{n+1} \dots \models \psi$

$(K, q) \models^\forall \varphi$  iff  $\neg (K, q) \models^\exists \neg\varphi$

## Defined modalities

$\bigcirc$

$X$  next

$U$

$U$  until

$\Diamond \varphi = \text{true } U \varphi$

$F$  eventually

$\Box \varphi = \neg \Diamond \neg \varphi$

$G$  always

$\varphi W \psi = (\varphi U \psi) \vee \Box \varphi$   
until)

$W$  waiting-for (weak-  
until)

# Important properties

Invariance

$\Box a$

safety

$\Box \neg (pc1=in \wedge pc2=in)$

Sequencing  
safety

$a W b W c W d$

$\Box (pc1=req \Rightarrow$

$(pc2 \neq in) W (pc2=in) W (pc2 \neq in) W$

$(pc1=in))$

Response

$\Box (a \Rightarrow \Box b)$

liveness

$\Box (pc1=req \Rightarrow \Box (pc1=in))$

## Composed modalities

$\Box\Box a$

infinitely often  $a$

$\Box\Box a$

almost always  $a$

## Example: Starvation freedom

Whenever process P1 wants to enter the critical section, **provided process P2 never stays in the critical section forever**, P1 gets to enter eventually.

$$\begin{aligned} & \Box \Box (pc2=in \Rightarrow O (pc2=out)) \Rightarrow \\ & \Box (pc1=req \Rightarrow \Box (pc1=in)) \end{aligned}$$

## State-transition graph

$Q$	set of states	$\{q_1, q_2, q_3\}$
$A$	set of atomic observations	$\{a, b\}$
$\rightarrow \subseteq Q \times Q$	transition relation	$q_1 \rightarrow q_2$
$[ ]: Q \rightarrow 2^A$	observation function	$[q_1] = \{a\}$



$$(K, q) \models^{\forall} \phi$$

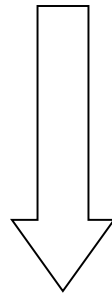


Tableau construction  
(Vardi-Wolper)

$(K', q', BA)$  where  $BA \subseteq K'$

Is there an infinite path starting from  $q'$   
that hits  $BA$  infinitely often?

Is there a path from  $q'$  to  $p \in BA$  such that  $p$  is a  
member of a strongly connected component of  $K'$ ?

```
dfs(s) {  
    add s to dfsTable  
    for each successor t of s  
        if ( $t \notin$  dfsTable) then dfs(t)  
    if ( $s \in BA$ ) then { seed := s; ndfs(s) }  
}
```

```
ndfs(s) {  
    add s to ndfsTable  
    for each successor t of s  
        if ( $t \notin$  ndfsTable) then ndfs(t)  
        else if ( $t = \text{seed}$ ) then report error  
}
```