

# Tecniche di Specifica e di Verifica

Linear Time Temporal Logic

# Temporal Logics: The context

- *Kripke Structures* model systems.
- *Temporal logics* model the dynamic behavioral properties of systems.
  - **Linear Time**
  - Branching Time
- *Model checking* can be used to determine if a system has the desired behavioral property.

# Properties of computations: local properties

Refer to **immediate successors** or **predecessors** of the current state.

*Examples:*

**Some/every immediate successor state satisfies the property  $\phi$ :**

- The system may enable the process  $i$  at the next state.
- If the light was red at the previous state and is orange now, it must turn green at the next state.

**Some/every immediate predecessor satisfies the property  $\phi$  (usually expressed as conditionals):**

- If the process  $i$  is currently enabled, the scheduler must have disabled the process  $j$  at the previous state.
- If train is entering the tunnel now, the semaphore must have been switched red on the other end at the previous moment.

***Local properties*** can be iterated a fixed number of times, but ***not indefinitely***.

# Universal properties of computations: invariance, safety

*Invariance properties* are properties that **must always hold** along the computation, while *safety properties* describe events that **must never happen** along the computation.

## Invariance:

- The greatest common divisor of **X** and **Y** remains the same throughout the execution.

## Safety:

- No *deadlock* will ever occur.
- At least one process will be enabled at any moment of time.
- Not more than one process will ever be in its *critical section* (e.g., not more than one train will ever be in the tunnel) at the same time.
- A resource will never be available to two or more processes simultaneously.

Also, **partial correctness** properties:

- If a *pre-condition* **P** holds at all initial states, then a *post-condition* **Q** will/must hold at all accepting (terminating) states.

# Existential properties of computations: eventualities, liveness

*Eventuality, liveness properties*: those that **will (must) happen sometime** during the computation.

*Examples*:

- The execution of the program will terminate.
- If the train has entered the tunnel, it will eventually leave it.
- Once a printing job is activated, eventually it will be completed.
- If a message is sent, eventually it will be delivered.

Also, **total correctness** properties:

- If a *pre-condition* **P** holds at the initial state, then the computation will reach an accepting (terminating) state, where the *post-condition* **Q** will hold.

# Properties of computations: fairness, precedence

*Fairness properties*: All processes will be treated “*fairly*” by the operating system (the scheduler, etc.)

*Examples*:

- *Weak fairness*: Every continuous request is eventually granted.
- *Strong fairness*: If a request is repeated infinitely often then it is eventually granted.
- *Impartiality*: Every process is scheduled infinitely often.
- *Precedence*: The event  $\alpha$  will occur before the event  $\beta$ , which may or may not occur at all.
  - If the train has entered the tunnel, it will eventually leave it (before any other train has entered it).

# Reachability properties in transition systems

- All important properties of computations can be expressed in terms of *reachability* or *non-reachability* of states with specific atomic properties.
- For instance, *eventuality* is just *reachability* of a “**good state**”, while *safety* is *non-reachability* of “**bad states**”, *fairness* corresponds to *repeated reachability*, etc.
- More generally, we may be interested in *reachability* of a state or a set of states along some or all paths starting from a given state (or, set of states); this is called **forward reachability**.
- Likewise we may be interested in the states from which a state (or a set of states) is reachable; this is called **backward reachability**.

# Linear time temporal logics.

- *LTL (Linear Time Temporal Logic)*
  - **Syntax**
  - **Semantics**
  - The Model Checking Problem.
  - Its solution.



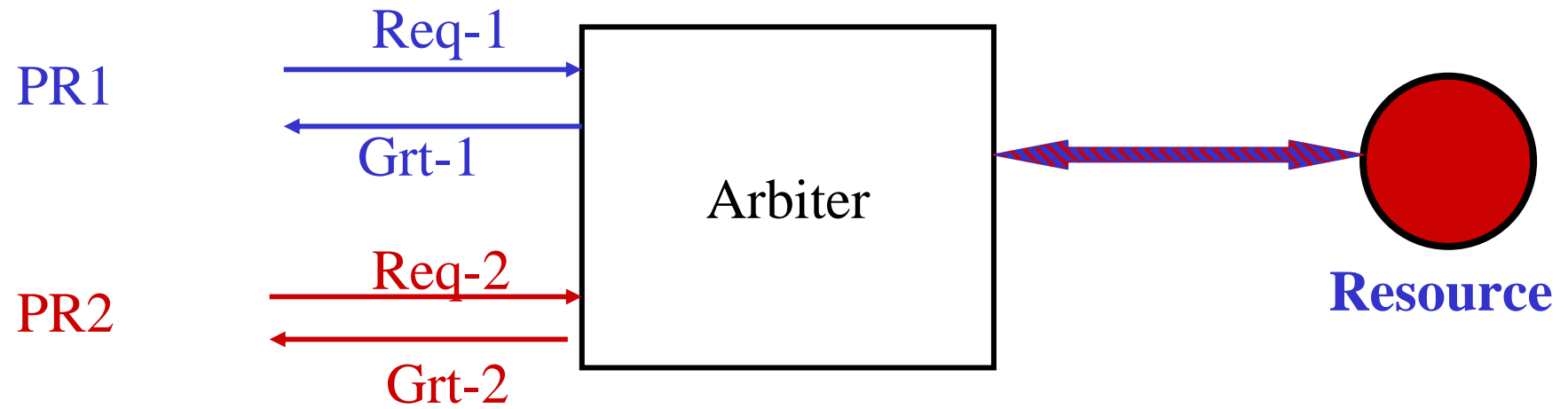
# The Application

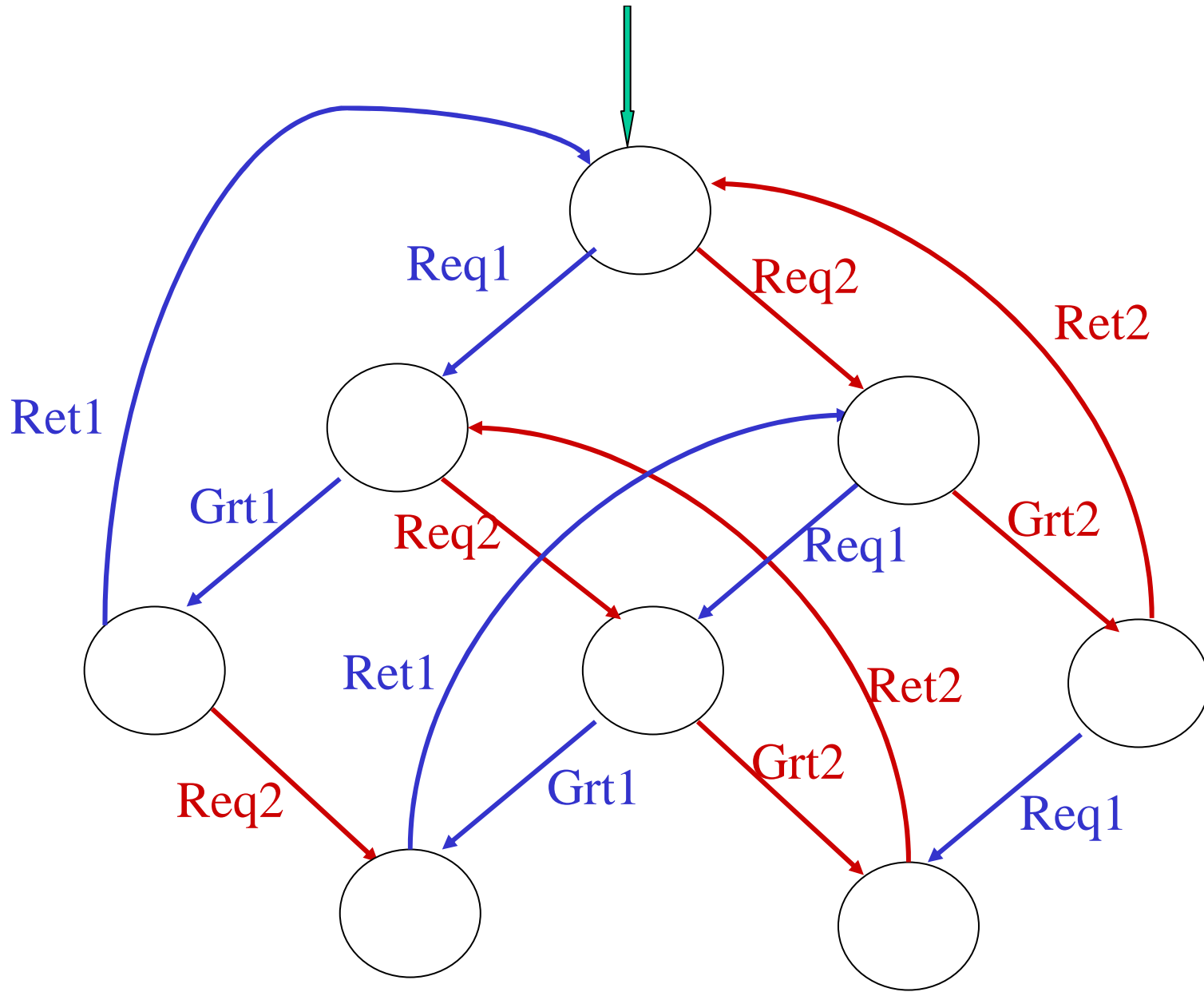
- Model a system to be verified as a Kripke structure:
  - Transition System  $\mathbf{TS} = (\mathbf{S}, \mathbf{S}_0, \mathbf{R})$
  - $\mathbf{AP}$  = A finite set of atomic propositions.
    - Basic assertions about the system
  - $\mathbf{L} : \mathbf{S} \rightarrow 2^{\mathbf{AP}}$  = The set of subsets of  $\mathbf{AP}$ .
  - $\mathbf{p} \in \mathbf{L}(\mathbf{s})$  ----  $\mathbf{p}$  is true at  $\mathbf{s}$ .
  - $\mathbf{p} \notin \mathbf{L}(\mathbf{s})$  ----  $\mathbf{p}$  is not true at  $\mathbf{s}$ .
- $\mathbf{K} = (\mathbf{S}, \mathbf{S}_0, \mathbf{R}, \mathbf{AP}, \mathbf{L})$  ---- Kripke Structure

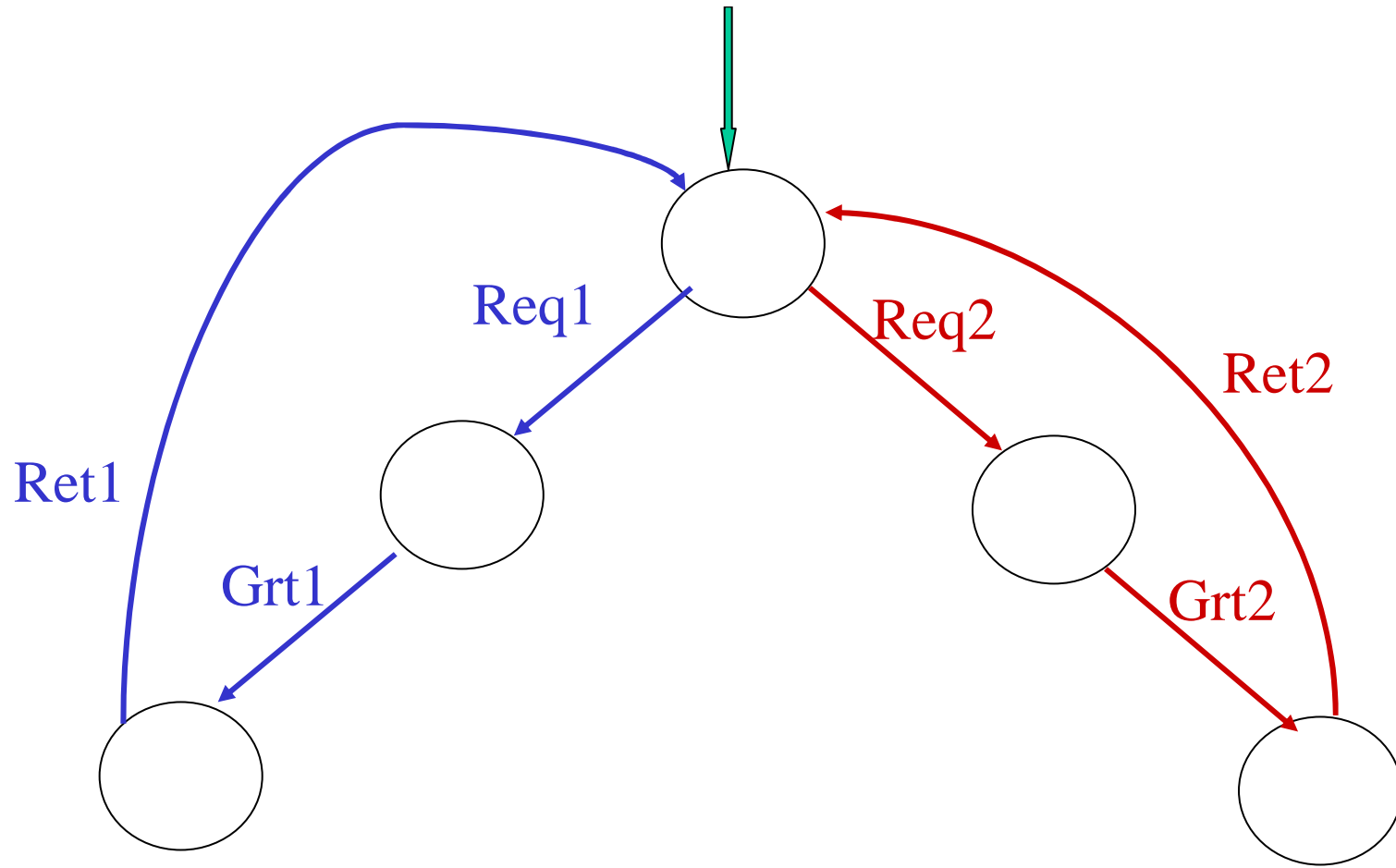
# The Application

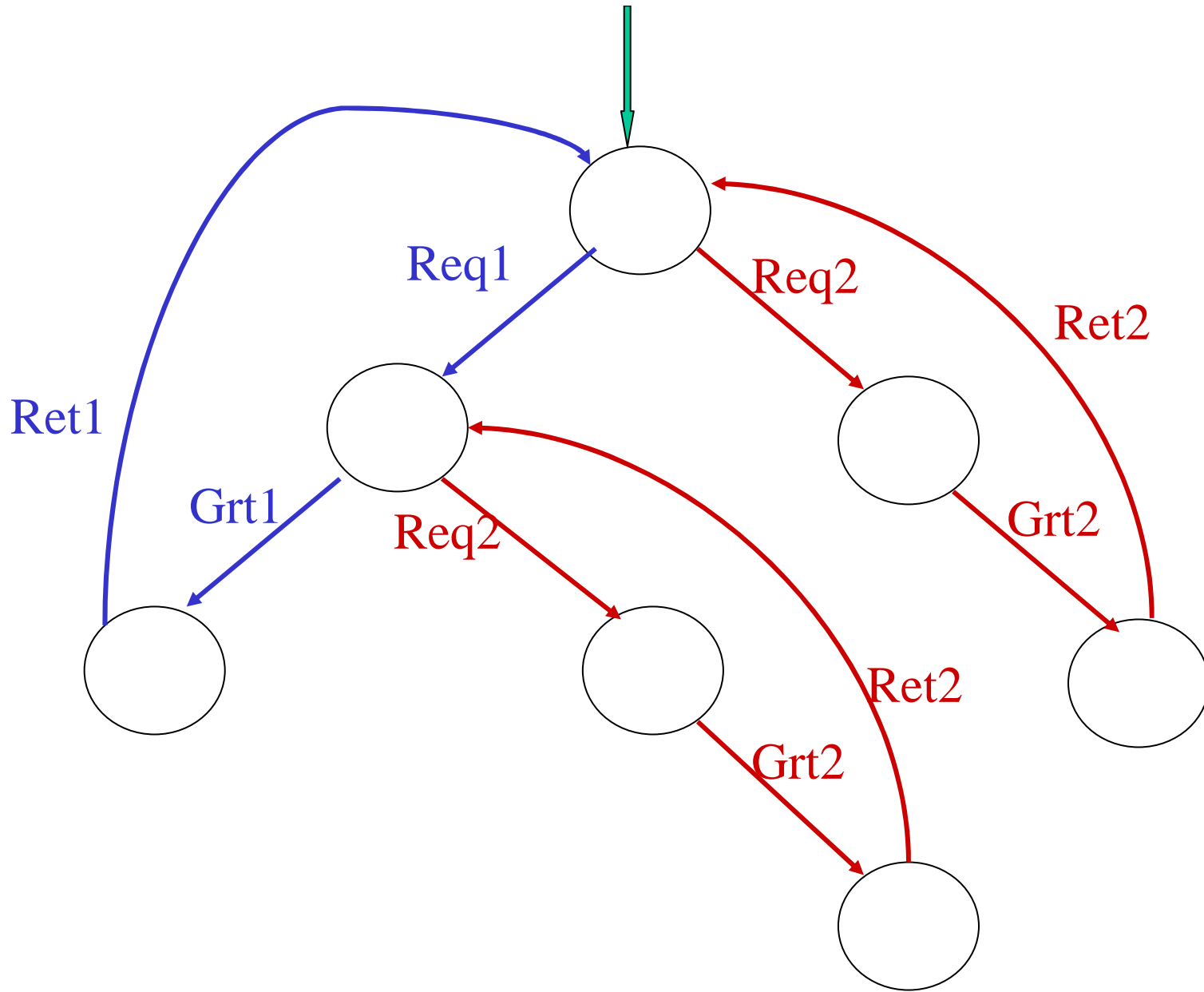
- *The computations* of the Kripke structure  $\mathbf{K}$  will be the *models* for **LTL** formulas.
- *The property* to be verified is captured as an LTL formula  $\varphi$ .
- The modeled system  $\mathbf{K}$  has the property  $\varphi$  *iff every computation of  $\mathbf{K}$  is a model of  $\varphi$ .*
- We need to verify (*model check*) whether:
  - $\mathbf{K} \models \varphi$

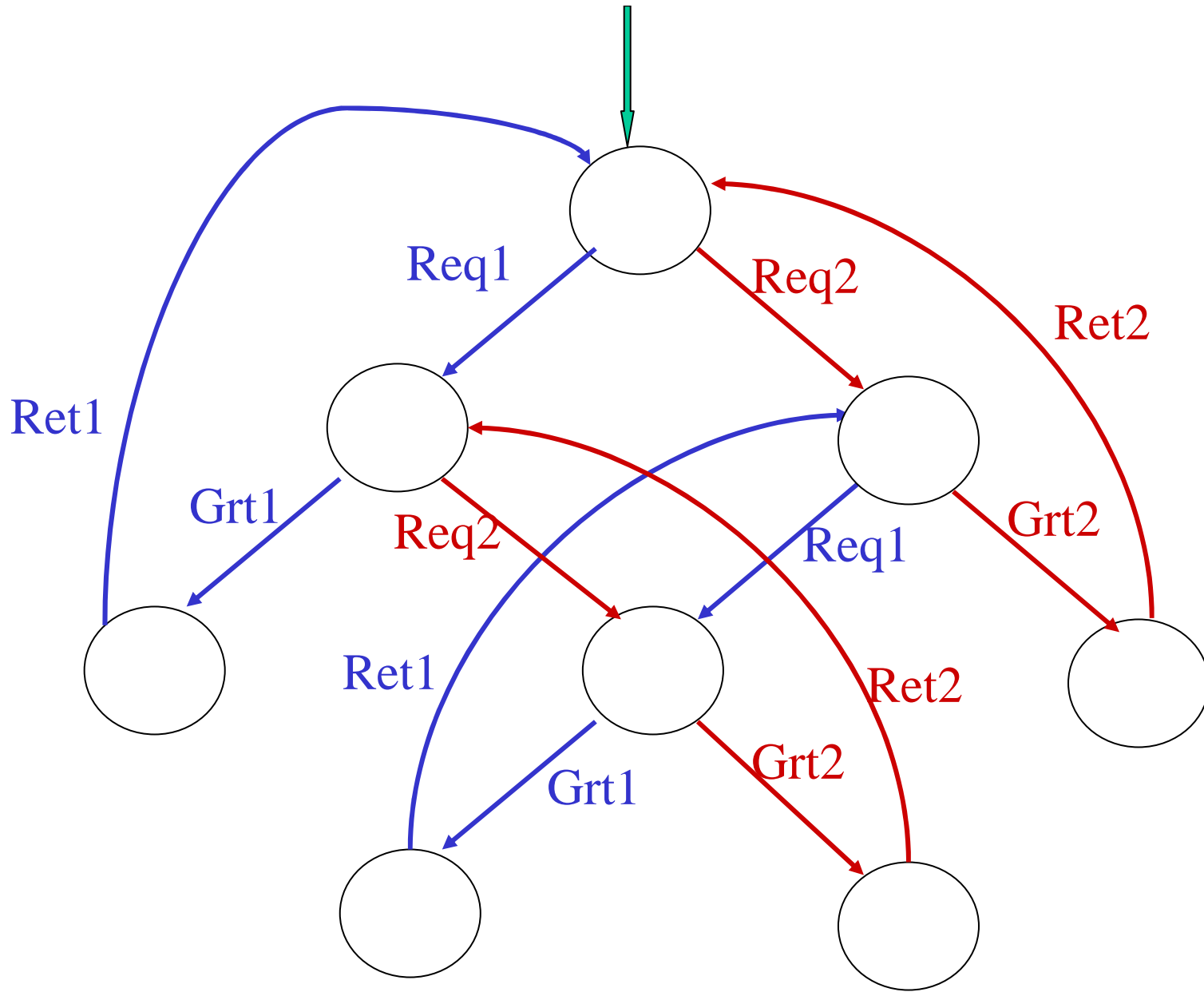
# An Example

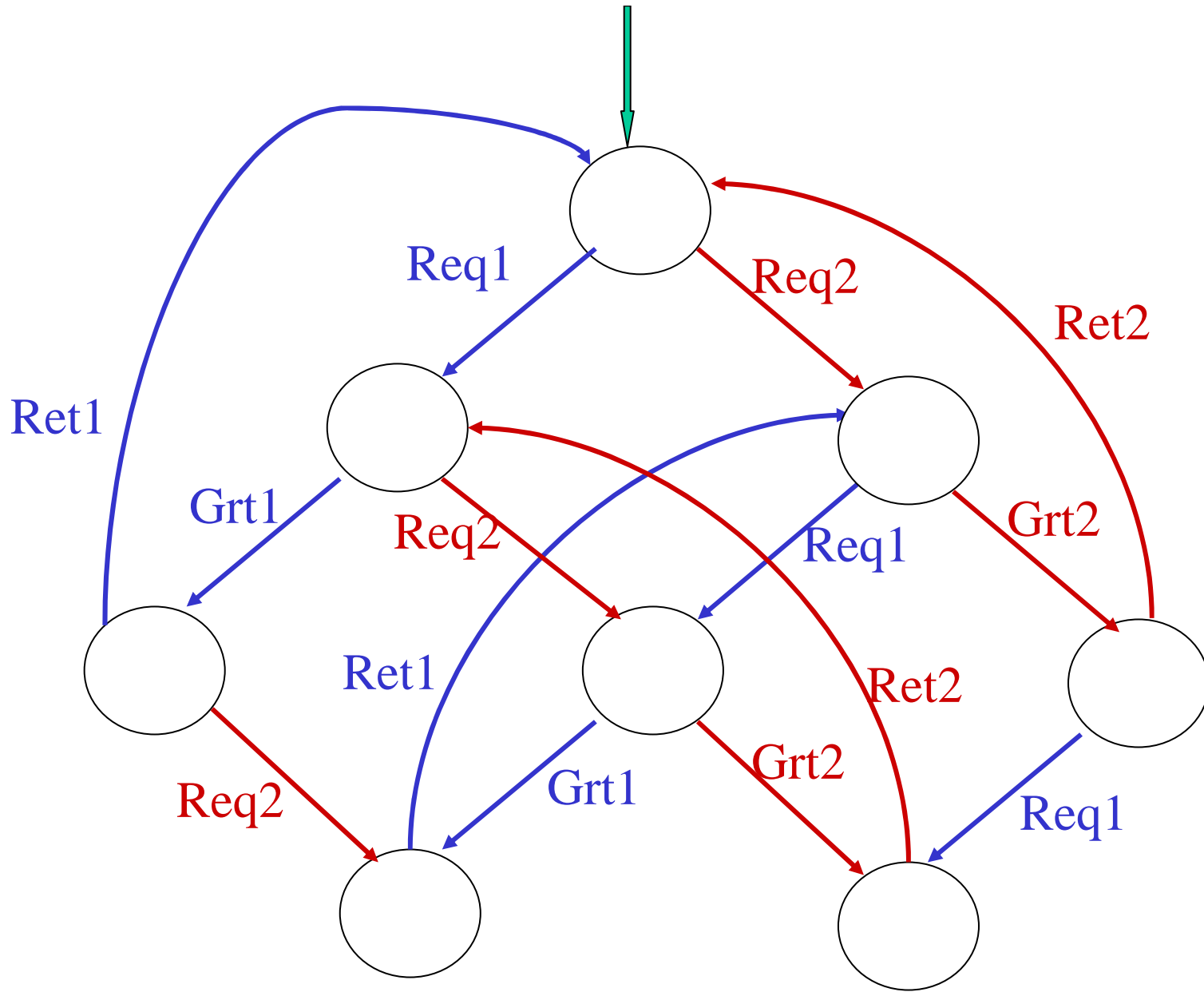






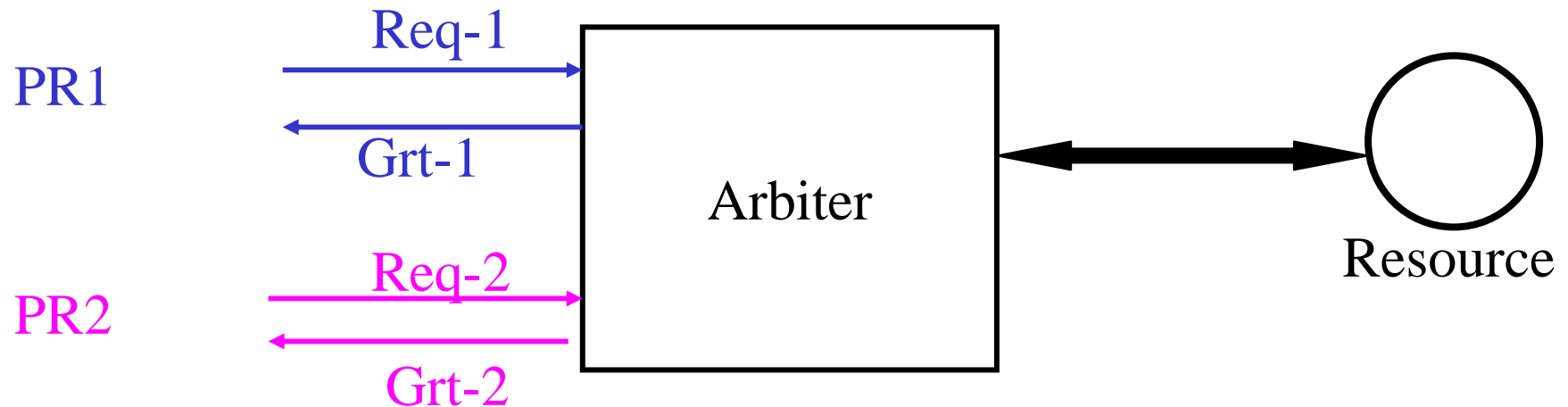








# A set of Atomic Propositions



R1 – Process 1 is *idle*

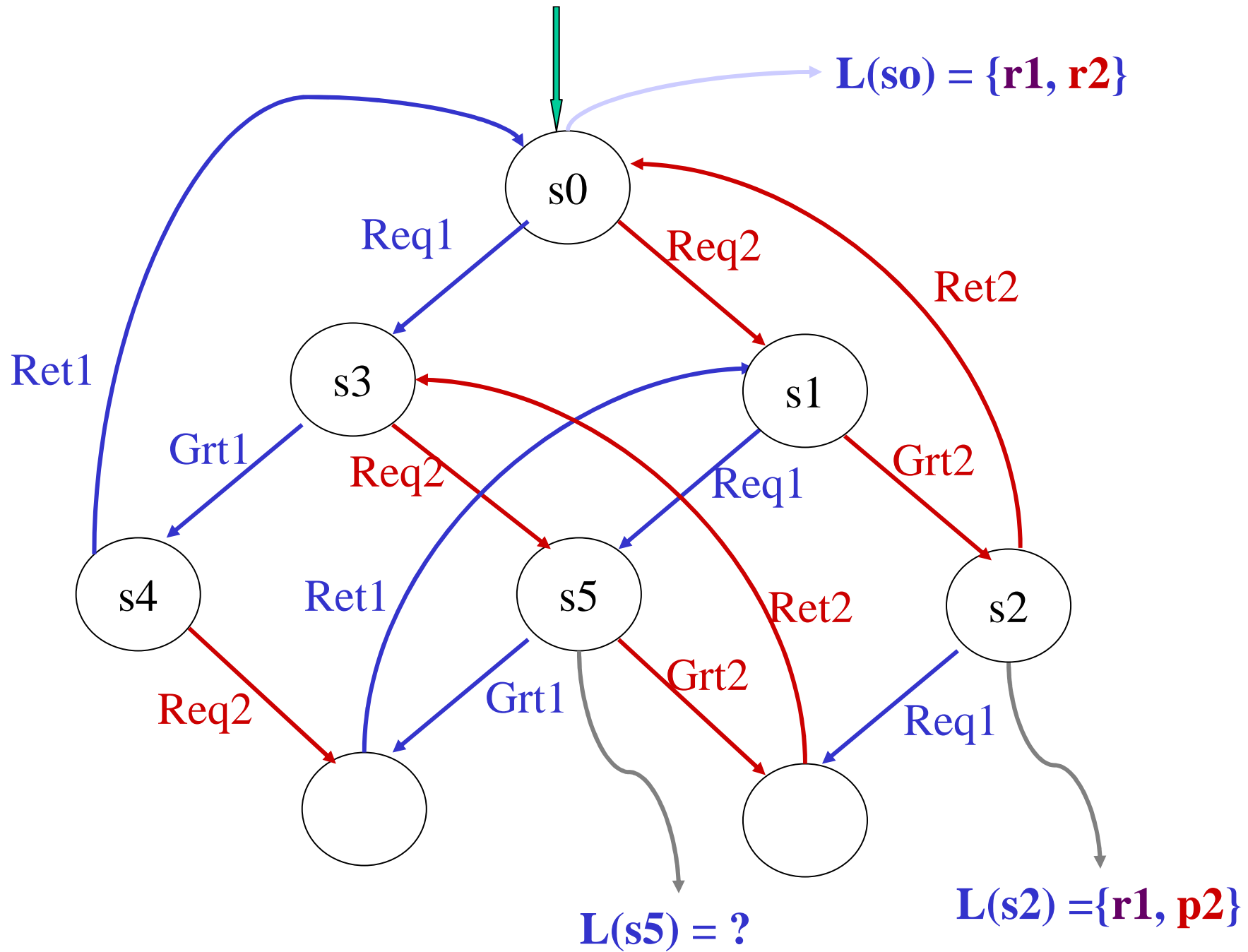
W1 – Process 1 is *waiting*

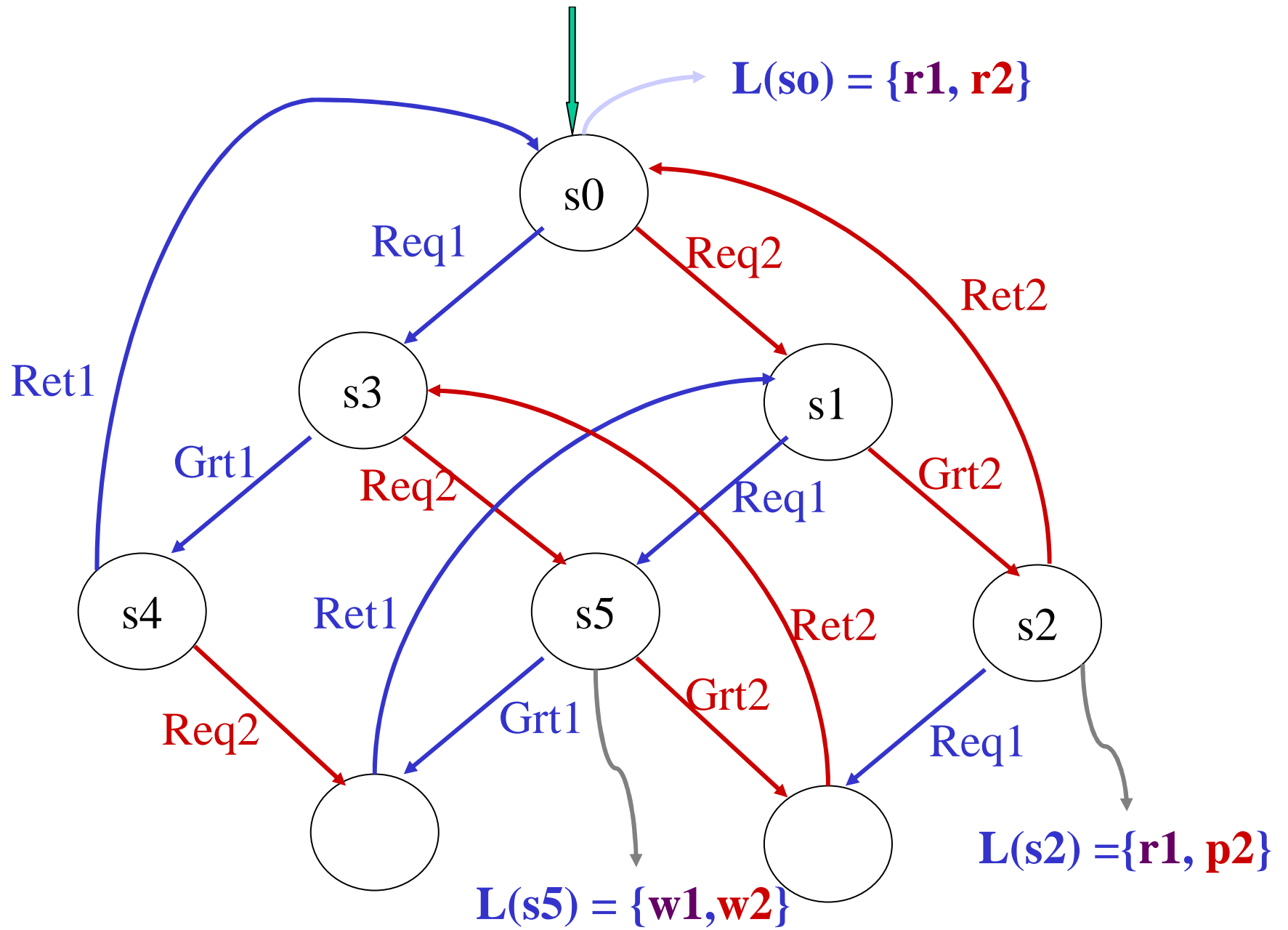
P1 – Process 1 is *using* the resource.

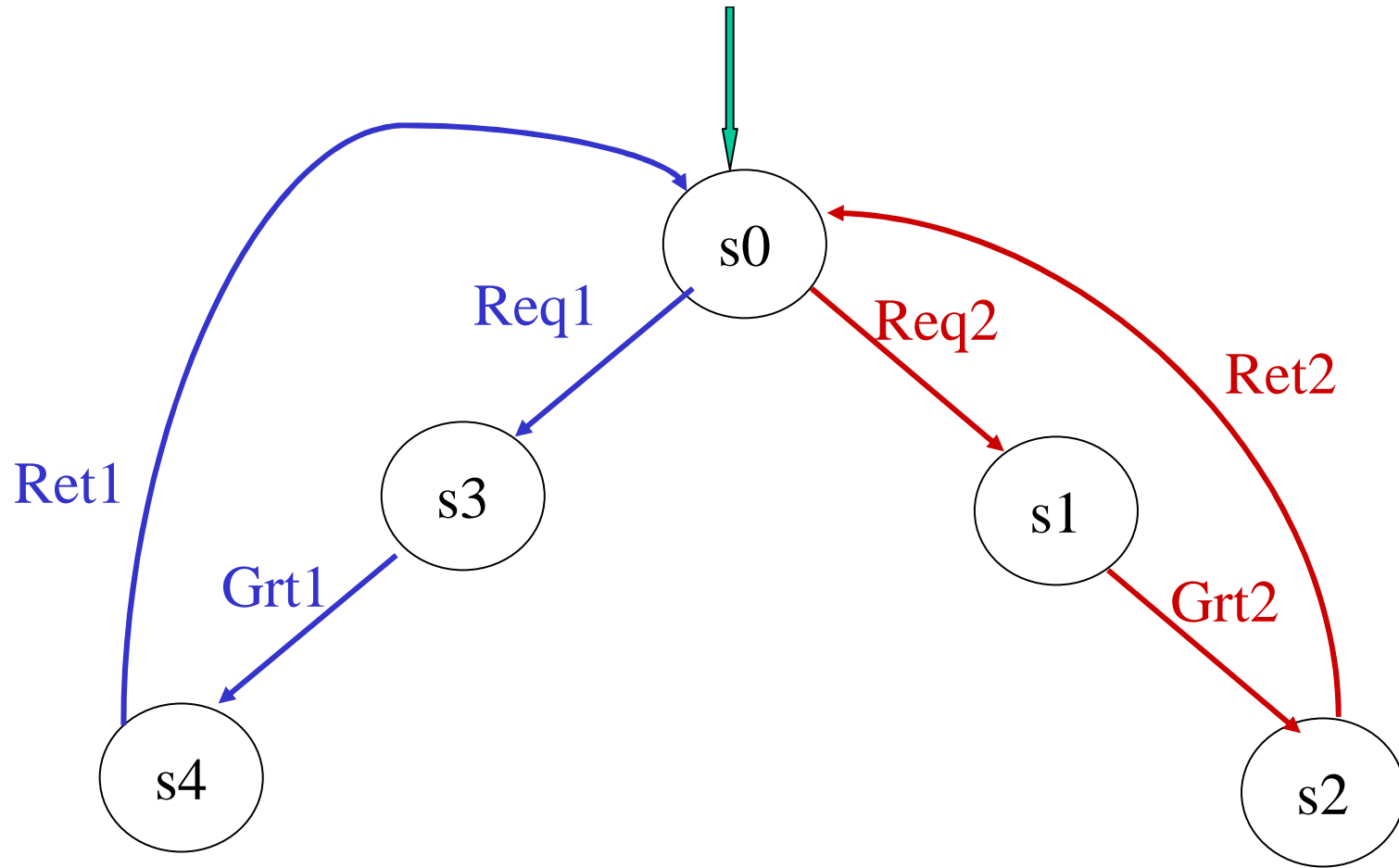
AP = { R1, W1, P1, R2, W2, P2 }

# The context

- Model a system to be verified as a Kripke structure:
  - Transition system  $\mathbf{TS} = (\mathbf{S}, \mathbf{S}_0, \mathbf{R})$
  - $\mathbf{AP}$  = A finite set of atomic propositions.
    - Basic assertions about the system
  - $\mathbf{L} : \mathbf{S} \longrightarrow 2^{\mathbf{AP}} =$  The set of subsets of  $\mathbf{AP}$ .
  - $\mathbf{p} \in \mathbf{L}(\mathbf{s})$  ----  $\mathbf{p}$  is true at  $\mathbf{s}$
  - $\mathbf{p} \notin \mathbf{L}(\mathbf{s})$  ----  $\mathbf{p}$  is not true at  $\mathbf{s}$ .
- $\mathbf{K} = (\mathbf{S}, \mathbf{S}_0, \mathbf{R}, \mathbf{AP}, \mathbf{L})$  ---- Kripke structure







s0      s3      s4      s0      s1      s2      s0      s3 ...

{r1, r2} {w1, r2} {u1, r2} {r1, r2} {r1, w2} {r1, p2} {r1, r2} {w1, r2}...

# Assertions about a computation

s0            s3            s4            s0            s1            s2            s0            s3 ...

{r1, r2} {w1, r2} {p1, r2} {r1, r2} {r1, w2} {r1, p2} {r1, r2} {w1, r2}..

- If at some stage Process 1 is **waiting** then at some **later** stage it is **printing** (i.e. using the resource).
- **At no stage** are both processes using the resource.
- If a process is waiting then it does so **until** it starts to use the resource.
- **There is a stage** at which both processes are waiting.

# The Application

- $\mathbf{K} = (\mathbf{S}, \mathbf{S}_0, \mathbf{R}, \mathbf{AP}, \mathbf{L})$
- Every computation (sequence of states) can be viewed as a sequence of subsets of **AP**.
- $s_0 \ s_1 \ s_2 \ \dots \ \text{----} \ \mathbf{L}(s_0) \ \mathbf{L}(s_1) \ \mathbf{L}(s_2) \ \dots$
- These **AP-computations** will be the models for the formulas of LTL.

- **Verification :**
  - Every **AP-computation** of **K** is a model of  $\varphi$

# Linear Time Temporal Logic (LTL)

- Syntax :
  - $AP = \{p_0, p_1, \dots, p_n\}$ , a finite set of Atomic Propositions.
- Formulas :
  - Every  $p_i$  in  $AP$  is a *LTL formula*.
  - If  $\varphi$  is a formula then  $\neg \varphi$  is a *LTL formula*.
  - If  $\varphi_1$  and  $\varphi_2$  are formulas then  $(\varphi_1 \vee \varphi_2)$  is a *LTL formula*.
  - If  $\varphi$  is a formula then  $X \varphi$ ,  $F \varphi$  and  $G \varphi$  are *LTL formulae* (**N**ext, **E**ventually, **A**lways).
  - If  $\varphi_1$  and  $\varphi_2$  are formulas then  $(\varphi_1 U \varphi_2)$  is a *LTL formula* (**U**ntil).

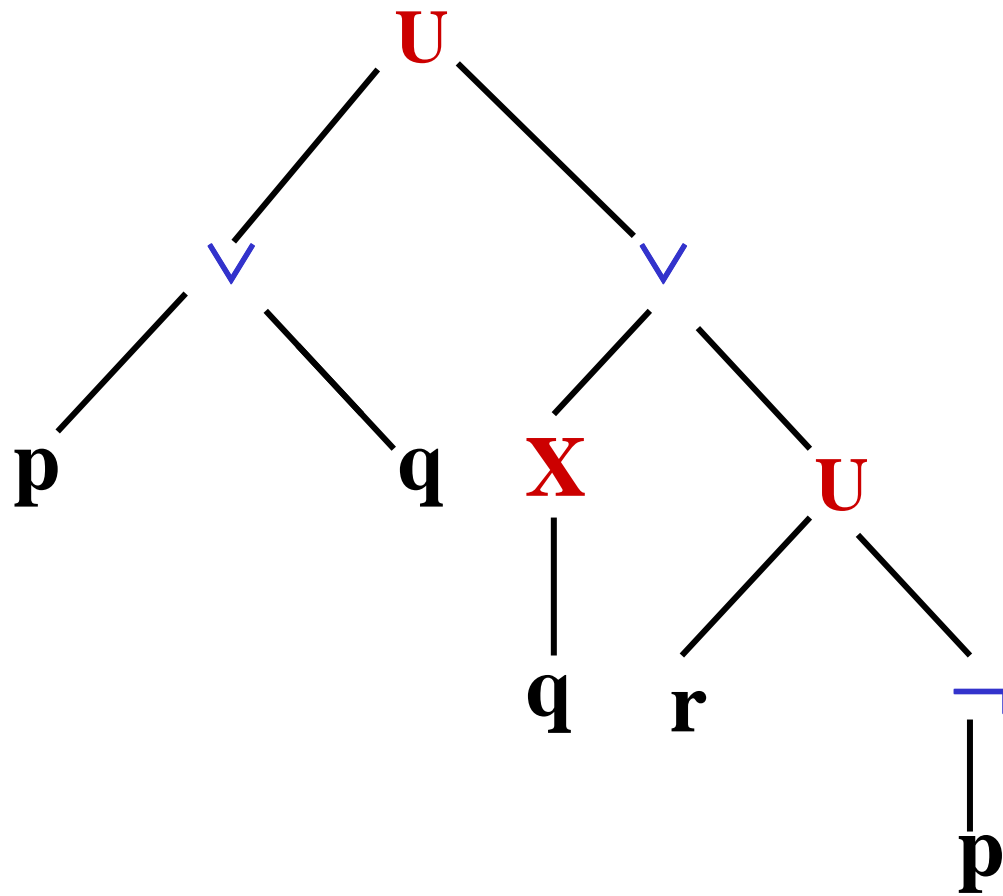


# Formulas

**LTL ::=  $p$  |  $\neg \varphi$  |  $\varphi_1 \vee \varphi_2$  |  $X \varphi$  |  $F \varphi$  |  $G \varphi$  |  $\varphi_1 U \varphi_2$**

- $p$  ;  $p \vee q$  ;  $(\neg p \vee q) \vee \neg(r \vee q)$
- $X q$  ;  $X(p \vee q)$  ;  $X((\neg p \vee q) \vee X\neg(r \vee q))$
- $(p \vee q) U (X r \vee (\neg q U (X \neg p)))$

$(p \vee q) \cup (\mathbf{X}q \vee (r \cup \neg p))$



# Semantics

- $\mathbf{AP}$  = A finite set of atomic propositions.
- $\Sigma = 2^{\mathbf{AP}}$  = The set of subsets of  $\mathbf{AP}$
- $\mathbf{AP} = \{ p, q, r \}$
- $\Sigma = \{ \emptyset, \{p\}, \{q\}, \{r\}, \{p,q\}, \{p,r\}, \{q,r\}, \{p,q,r\} \}$
- $\Sigma^\omega$  = The set of *infinite sequences* over  $\Sigma$ .

# Semantics

- $\mathbf{AP} = \{p, q, r\}$     $\Sigma = 2^{\mathbf{AP}}$
- $\Sigma = \{\emptyset, \{p\}, \{q\}, \dots, \{p, q, r\}\}$

$\sigma$ :      $\{p, r\}$     $\{q\}$     $\emptyset$     $\{p, q, r\}$     $\{r\}$  ...  
          |          |          |          |          |  
*path*:     0    $\longrightarrow$    1    $\longrightarrow$    2    $\longrightarrow$    3    $\longrightarrow$    4   ...

- At stage **0** of  $\sigma$ , **p** and **r** are true but not **q**;  
  at stage **2** of  $\sigma$  no member of **AP** is true....

# Semantics

- $\Sigma^\omega$  = The set of infinite sequences over  $\Sigma$ .
- $\sigma \in \Sigma^\omega$  --- A model
- $\sigma(i)$  -----  $i$ -th position of  $\sigma$
- $\{p\}$   $\{q,r\}$   $\emptyset$   $\{r, q\}$   $\{p, q, r\}$ .....
- $0$   $1$   $2$   $3$   $4$  .....
- $\sigma(0) = \{p\}$   $\sigma(2) = \emptyset$   $\sigma(3) = ?$

# Semantics

- $\mathbf{AP} \quad \Sigma = 2^{\mathbf{AP}}$
- $\Sigma^{\omega}$  = The set of infinite sequences over  $\Sigma$ .
- $\sigma \in \Sigma^{\omega}$  --- A model
- $\sigma(\mathbf{i})$  ----  $\mathbf{i}$ -th position of  $\sigma$
- $\varphi$ , a formula.

- $\sigma(\mathbf{i}) \models \varphi$ 
  - $\sigma(\mathbf{i})$  *satisfies*  $\varphi$
  - $\varphi$  is true in the  $\mathbf{i}$ -th position of  $\sigma$

# Semantics

**LTL** ::=  $p$  |  $\neg \varphi$  |  $\varphi_1 \vee \varphi_2$  |  $X \varphi$  |  $F \varphi$  |  $G \varphi$  |  $\varphi_1 U \varphi_2$

•  $\sigma = \Gamma_0 \Gamma_1 \Gamma_2 \dots \Gamma_i \Gamma_{i+1} \dots$

• Each  $\Gamma_j$  is a subset of **AP**.

•  $\sigma(i) \models p$  *iff*  $p \in \Gamma_i$

# Semantics

**LTL** ::=  $p$  |  $\neg \varphi$  |  $\varphi_1 \vee \varphi_2$  |  $X \varphi$  |  $F \varphi$  |  $G \varphi$  |  $\varphi_1 U \varphi_2$

- $AP = \{p, q, r\}$

- $\sigma = \{p, q\} \quad \{r\} \quad \emptyset \quad \{q, r\} \quad \{p, q, r\} \dots$

0

1

2

3

4

- $\sigma(0)$  *satisfies*  $q$

- $\sigma(1)$  *satisfies*  $r$

- $\sigma(2)$  does *not* *satisfy*  $q$  !



# Semantics

**LTL** ::=  $p$  |  $\neg \varphi$  |  $\varphi_1 \vee \varphi_2$  | **X**  $\varphi$  | **F**  $\varphi$  | **G**  $\varphi$  |  $\varphi_1$  **U**  $\varphi_2$

$\sigma = \Gamma_0 \Gamma_1 \Gamma_2 \dots \Gamma_i \Gamma_{i+1} \dots$

Each  $\Gamma_j$  is a subset of **AP**.

- $\sigma(i) \models \neg \varphi$  *iff*  $\sigma(i) \not\models \varphi$

# Semantics

**LTL** ::=  $p$  |  $\neg \varphi$  |  $\varphi_1 \vee \varphi_2$  | **X**  $\varphi$  | **F**  $\varphi$  | **G**  $\varphi$  |  $\varphi_1$  **U**  $\varphi_2$

$\sigma = \Gamma_0 \Gamma_1 \Gamma_2 \dots \Gamma_i \Gamma_{i+1} \dots$

Each  $\Gamma_j$  is a subset of **AP**.

- $\sigma(i) \models \varphi_1 \vee \varphi_2$  *iff*  $\sigma(i) \models \varphi_1$  **OR**  
 $\sigma(i) \models \varphi_2$

# Semantics

**LTL** ::=  $p$  |  $\neg \varphi$  |  $\varphi_1 \vee \varphi_2$  |  $X \varphi$  |  $F \varphi$  |  $G \varphi$  |  $\varphi_1 U \varphi_2$

**AP** = {**p**, **q**, **r**}

$\sigma = \{p, q\} \{r\} \emptyset \{q, r\} \{p, q, r\} \dots$

0 1 2 3 4

- $\sigma(0)$  satisfies  $\neg r$  ;  $\sigma(0)$  does *not* satisfy **r**
- $\sigma(1)$  satisfies **p**  $\vee$  **r** ;  $\sigma(1)$  satisfies **r**
- $\sigma(2)$  satisfies  $\neg(p \vee r)$  ?

# Semantics

$LTL ::= p \mid \neg \varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{X} \varphi \mid \mathbf{F} \varphi \mid \mathbf{G} \varphi \mid \varphi_1 \mathbf{U} \varphi_2$

$AP = \{p, q, r\}$

$\sigma = \{p, q\} \{r\} \emptyset \{q, r\} \{p, q, r\} \dots$

0      1      2      3      4

- $\sigma(2)$  satisfies  $\neg(p \vee r)$  ? **Yes!**
- $\sigma(2)$  does **not satisfy**  $p \vee r$



# Semantics

**LTL** ::=  $p$  |  $\neg \varphi$  |  $\varphi_1 \vee \varphi_2$  | **X**  $\varphi$  | **F**  $\varphi$  | **G**  $\varphi$  |  $\varphi_1$  **U**  $\varphi_2$

**AP** = {**p**, **q**, **r**}

$\sigma = \{p,q\}$     $\{r\}$     $\emptyset$     $\{q,r\}$     $\{p,q,r\}$  .....  
          0       1       2       3       4

- $\sigma(2)$  *satisfies* **X r** ;  $\sigma(3)$  *satisfies* **r**
- $\sigma(0)$  *satisfies* **X(p  $\vee$  r)** ;  $\sigma(1)$  *satisfies* **r**
- $\sigma(1)$  does *not satisfy* **X(p  $\vee$  r)**
  - $\sigma(2)$  does *not satisfy* **p  $\vee$  r**

# Semantics

$LTL ::= p \mid \neg \varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{X} \varphi \mid \mathbf{F} \varphi \mid \mathbf{G} \varphi \mid \varphi_1 \mathbf{U} \varphi_2$

$AP = \{p, q, r\}$

$\sigma = \{p, q\} \quad \{r\} \quad \emptyset \quad \{q, r\} \quad \{p, q, r\} \dots$   
                  0          1          2          3          4

- $\sigma(1)$  satisfies  $\mathbf{X}(\mathbf{X}\neg p)$  *iff*
  - $\sigma(2)$  satisfies  $\mathbf{X}\neg p$  *iff*
  - $\sigma(3)$  satisfies  $\neg p$  *iff*
  - $\sigma(3)$  does *not* satisfy  $p$





# Semantics

$LTL ::= p \mid \neg \varphi \mid \varphi_1 \vee \varphi_2 \mid X \varphi \mid \mathbf{F} \varphi \mid \mathbf{G} \varphi \mid \varphi_1 \mathbf{U} \varphi_2$

$AP = \{p, q, r\}$

$\sigma = \{p, q\} \quad \{r\} \quad \emptyset \quad \{q, r\} \quad \{p, q, r\} \dots$   
          0      1      2      3          4

- $\sigma(0)$  satisfies  $\mathbf{F}(X p)$  this is true since  
–  $\sigma(3)$  satisfies  $X p$  *iff*  
–  $\sigma(4)$  satisfies  $p$  is true since



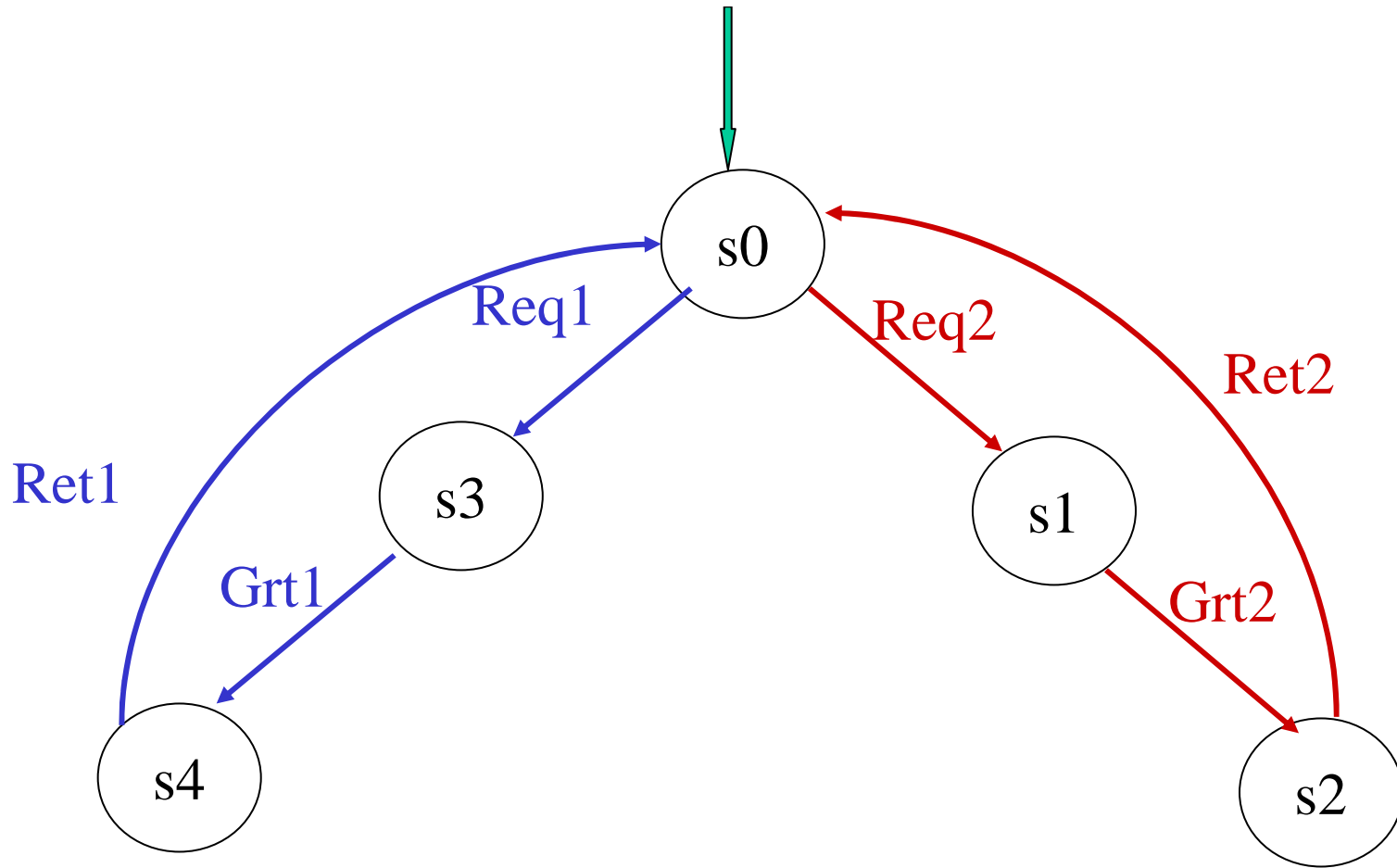


# Semantics

**LTL** ::=  $p$  |  $\neg \varphi$  |  $\varphi_1 \vee \varphi_2$  |  $X \varphi$  |  $F \varphi$  |  $G \varphi$  |  $\varphi_1 U \varphi_2$

- $k$  could be arbitrarily greater than  $i$ .
- $k = i$  is allowed and there is **no**  $i \leq j < k$

- $\sigma(i) \models \varphi_1 U \varphi_2$  *iff* there exists  $k \geq i$  s.t.
  - $\sigma(k) \models \varphi_2$
  - $\sigma(j) \models \varphi_1$  for every  $i \leq j < k$



s0      s3      s4      s0      s1      s2      s0      s3 ...

{r1,r2} {w1,r2} {p1,r2} {r1,r2} {r1,w2} {r1,p2} {r1,r2} {w1,r2}...

# An Example

$AP = \{r1, w1, p1, r2, w2, p2\}$

$\{r1, r2\}$   $\{w1, r2\}$   $\{p1, r2\}$   $\{r1, r2\}$   $\{r1, w2\}$   $\{r1, p2\}$   $\{r1, r2\}$   $\{w1, r2\}$  ...  
0            1            2            3            4            5            6            7

- $\sigma(1)$  satisfies  $(r2 \cup w2)$  ;
  - $\sigma(4)$  satisfies  $w2$  and
  - $\sigma(1), \sigma(2), \sigma(3)$  satisfy  $r2$ .

# An Example

AP = {r1, w1, p1, r2, w2, p2}

{r1,r2} {w1,r2} {p1,r2} {r1,r2} {r1,w2} {r1,p2} {r1,r2} {w1,r2}...

0

1

2

3

4

5

6

7

- $\sigma(1)$  does *not satisfy* (r2 U p2) ;
  - $\sigma(5)$  *satisfies* p2 and
  - $\sigma(1), \sigma(2), \sigma(3)$  *satisfy* r2.
  - but  $\sigma(4)$  does *not satisfy* r2 !

# An Example

$$AP = \{r1, w1, p1, r2, w2, p2\}$$

{r1,r2} {w1,r2} {p1,r2} {r1,r2} {r1,w2} {r1,p2} {r1,r2} {w1,r2}...

0 1 2 3 4 5 6 7

- $\sigma(1)$  does satisfy  $((r2 \vee w2) \cup p2)$  ;
  - $\sigma(5)$  satisfies  $p2$  and
  - $\sigma(1), \sigma(2), \sigma(3)$  satisfy  $r2$ , hence also  $(r2 \vee w2)$ .
  - $\sigma(4)$  satisfies  $w2$ , hence also  $(r2 \vee w2)$  !



# Models

- **AP**  $\Sigma^{\text{AP}} = 2$
- $\Sigma^\omega$  = The set of infinite sequences over  $\Sigma$ .
- $\sigma \in \Sigma^\omega$
- $\varphi$  an **LTL** formula.

- A path  $\sigma$  is a *model* of  $\varphi$  ( $\sigma \models \varphi$ ) *iff*  
–  $\sigma(0) \models \varphi$

# Validity in LTL

- **AP**       $\Sigma^{\text{AP}} = 2$
- $\Sigma^\omega$  = The set of infinite sequences over  $\Sigma$ .
- $\sigma \in \Sigma^\omega$
- $\varphi$  an **LTL** formula.

- $\varphi$  is *LTL-valid* ( $\models \varphi$ ) *iff for every*  $\sigma \in \Sigma^\omega$   
–  $\sigma \models \varphi$

# Basic LTL Language

We will use the *reduced LTL language*

$LTL ::= p \mid \neg \varphi \mid \varphi_1 \vee \varphi_2 \mid X \varphi \mid \varphi_1 U \varphi_2$

- $\varphi_1 \wedge \varphi_2 \dashv\vdash \neg(\neg\varphi_1 \vee \neg\varphi_2)$  (and)
- $\varphi_1 \supset \varphi_2 \dashv\vdash \neg\varphi_1 \vee \varphi_2$  (implies)
- $\varphi_1 \equiv \varphi_2 \dashv\vdash (\varphi_1 \supset \varphi_2) \wedge (\varphi_2 \supset \varphi_1)$  (iff)
- $AP = \{p_1, p_2, \dots, p_n\}$
- $\top \dashv\vdash p_1 \vee \neg p_1$  (true)

- **Fact** : In every model  $\sigma$ , at every  $i$ ,  
–  $\sigma(i) \models \top$

# Derived Operators

- **LTL** ::=  $p$  |  $\neg \varphi$  |  $\varphi_1 \vee \varphi_2$  |  $X \varphi$  |  $\varphi_1 U \varphi_2$
- **F** $\varphi \equiv (\top U \varphi)$  (future ; diamond:  $\diamond$ )

- **We gave the following semantics :**
  - $\sigma(i) \models F\varphi$  *iff* **there exists  $k \geq i$**  such that  $\sigma(k) \models \varphi$ .

# Derived Operators

- We gave the following semantics :
  - $\sigma(i) \models F\varphi$  *iff* there exists  $k \geq i$  such that  $\sigma(k) \models \varphi$ .

*Proof of*  $F\varphi \equiv (\top U \varphi)$

$\sigma(i) \models (\top U \varphi)$  *iff*

$\exists j \geq i, \sigma(j) \models \varphi$  and  $\forall i \leq k < j, \sigma(k) \models \top$  *iff*

$\exists j \geq i, \sigma(j) \models \varphi$  *iff*

$\sigma(i) \models F\varphi$

# Derived Operators

- $LTL ::= p \mid \neg \varphi \mid \varphi_1 \vee \varphi_2 \mid X \varphi \mid \varphi_1 U \varphi_2$
- $F\varphi \equiv (\top U \varphi)$
- $G\varphi \equiv \neg F\neg \varphi$  (invariant; box:  $\square$ )

- **We gave the following semantics :**
  - $\sigma(i) \models G \varphi$  *iff* for every  $k \geq i$  ,  
 $\sigma(k) \models \varphi$  .

# Derived Operators

- **LTL** ::=  $p$  |  $\neg \varphi$  |  $\varphi_1 \vee \varphi_2$  |  $X \varphi$  |  $\varphi_1 U \varphi_2$
- $(\psi \mathbf{R} \varphi) \equiv \neg (\neg \psi U \neg \varphi)$  (Releases)
- $\mathbf{G} \varphi \equiv (\perp \mathbf{R} \varphi)$

–  $\sigma(i) \models (\psi \mathbf{R} \varphi)$  *iff*

for each  $k \geq i$  (*if* for each  $i \leq j < k$   
 $\sigma(j) \not\models \psi$  then  $\sigma(k) \models \varphi$ )

# Derived Operators

- **LTL** ::=  $p$  |  $\neg \varphi$  |  $\varphi_1 \vee \varphi_2$  | **X**  $\varphi$  |  $\varphi_1$  **U**  $\varphi_2$
- $(\psi$  **R**  $\varphi) \equiv \neg (\neg\psi$  **U**  $\neg\varphi)$  (Releases)
- **G**  $\varphi \equiv (\perp$  **R**  $\varphi)$

–  $\sigma(i) \models (\psi$  **R**  $\varphi)$  *iff*

for each  $k \geq i$  (for some  $i \leq j < k$

$\sigma(j) \models \psi$  or  $\sigma(k) \models \varphi$ )



# Derived Operators

- $\text{LTL} ::= p \mid \neg \varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{X} \varphi \mid \varphi_1 \mathbf{U} \varphi_2$
- $(\psi \mathbf{W} \varphi)$  (Unless)

Give the semantics according to the following intuition:

- $(\psi \mathbf{W} \varphi)$ : if  $\psi$  must be true unless  $\varphi$  occurs (notice that  $\varphi$  may never occur).

Show that:  $(\psi \mathbf{W} \varphi) \equiv \mathbf{G} \psi \vee (\psi \mathbf{U} \varphi)$

Show that:  $\neg(\psi \mathbf{U} \varphi) \equiv ((\neg \varphi \wedge \psi) \mathbf{W} (\neg \psi \wedge \neg \varphi))$

# Derived Operators

- **LTL** ::=  $p$  |  $\neg \varphi$  |  $\varphi_1 \vee \varphi_2$  | **X**  $\varphi$  |  $\varphi_1$  **U**  $\varphi_2$
- **( $\psi$  **B**  $\varphi$ )**      (**Before**)

Give the semantics according to the following intuition:

- **( $\psi$  **B**  $\varphi$ )**: if  $\varphi$  ever occurs, then  $\psi$  must occur before  $\varphi$ .

Show that: **( $\psi$  **B**  $\varphi$ )**  $\equiv \neg ( \neg \psi$  **U**  $\varphi )$

# Some important validities

- ♦  $(\psi \mathbf{U} \varphi) \equiv \varphi \vee (\psi \wedge \mathbf{X} (\psi \mathbf{U} \varphi))$
- ♦  $(\psi \mathbf{R} \varphi) \equiv \varphi \wedge (\psi \vee \mathbf{X} (\psi \mathbf{R} \varphi)) \equiv$   
 $\equiv (\varphi \wedge \psi) \vee (\varphi \wedge \mathbf{X} (\psi \mathbf{R} \varphi))$
- ♦  $\mathbf{F}\varphi \equiv \varphi \vee \mathbf{X} \mathbf{F}\varphi$
- ♦  $\mathbf{G}\varphi \equiv \varphi \wedge \mathbf{X} \mathbf{G}\varphi$

# LTL: Some examples

- **Safety:** “it *never happens* that both A and B are printing at the same time”

$$\mathbf{G}(\neg (P_A \wedge P_B))$$

- **Liveness:** “*whenever* A is waiting, it will *eventually* print in the future”

$$\mathbf{G}(W_A \supset \mathbf{F} P_A)$$

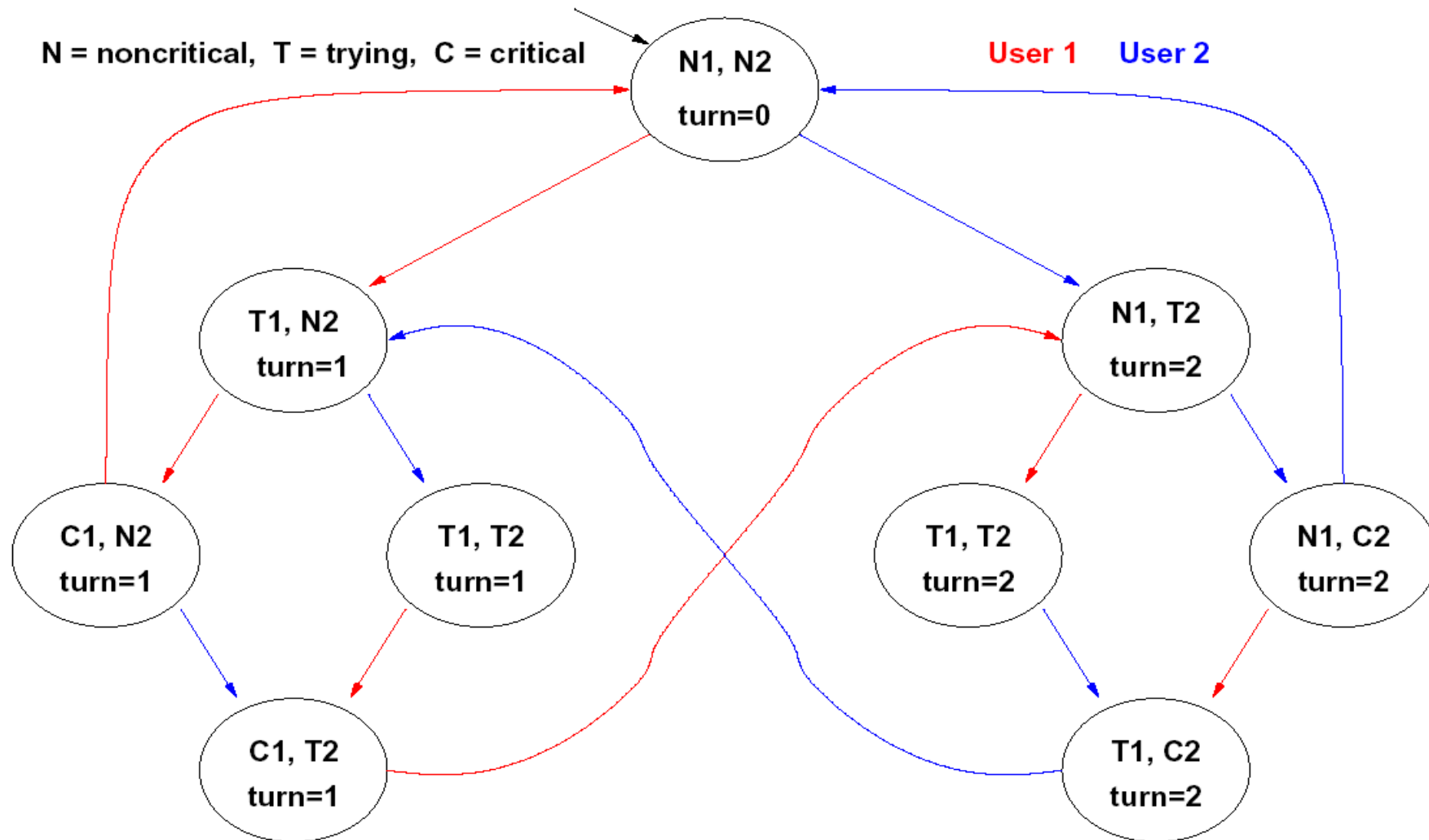
- **Fairness:** “A is *infinitely often* idle”

$$\mathbf{GF} R_A$$

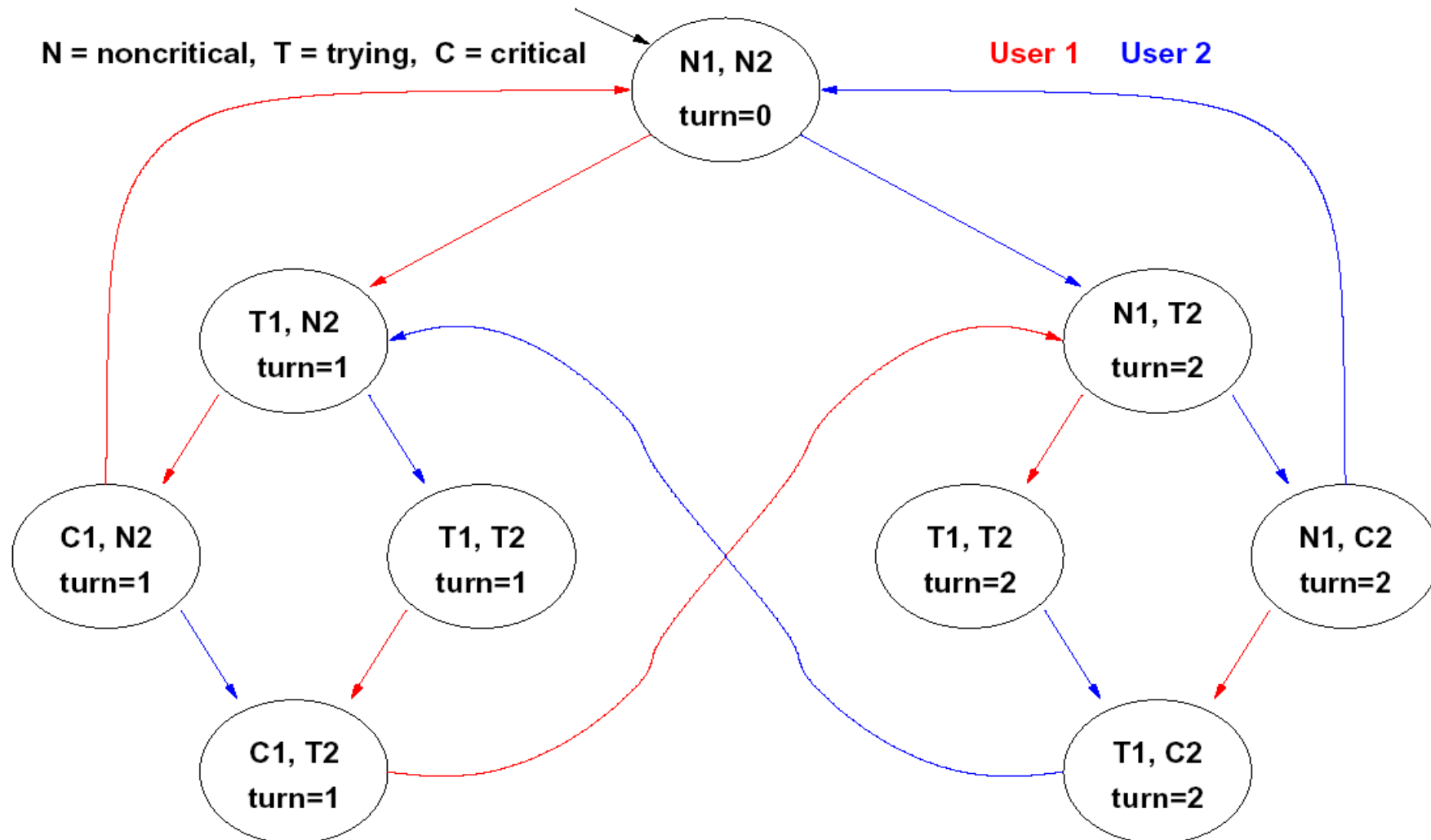
- **Strong fairness:** “if A is *infinitely often* waiting, then it will *infinitely often* printing”

$$\mathbf{GF} W_A \supset \mathbf{GF} P_A$$

# Example: mutual exclusion



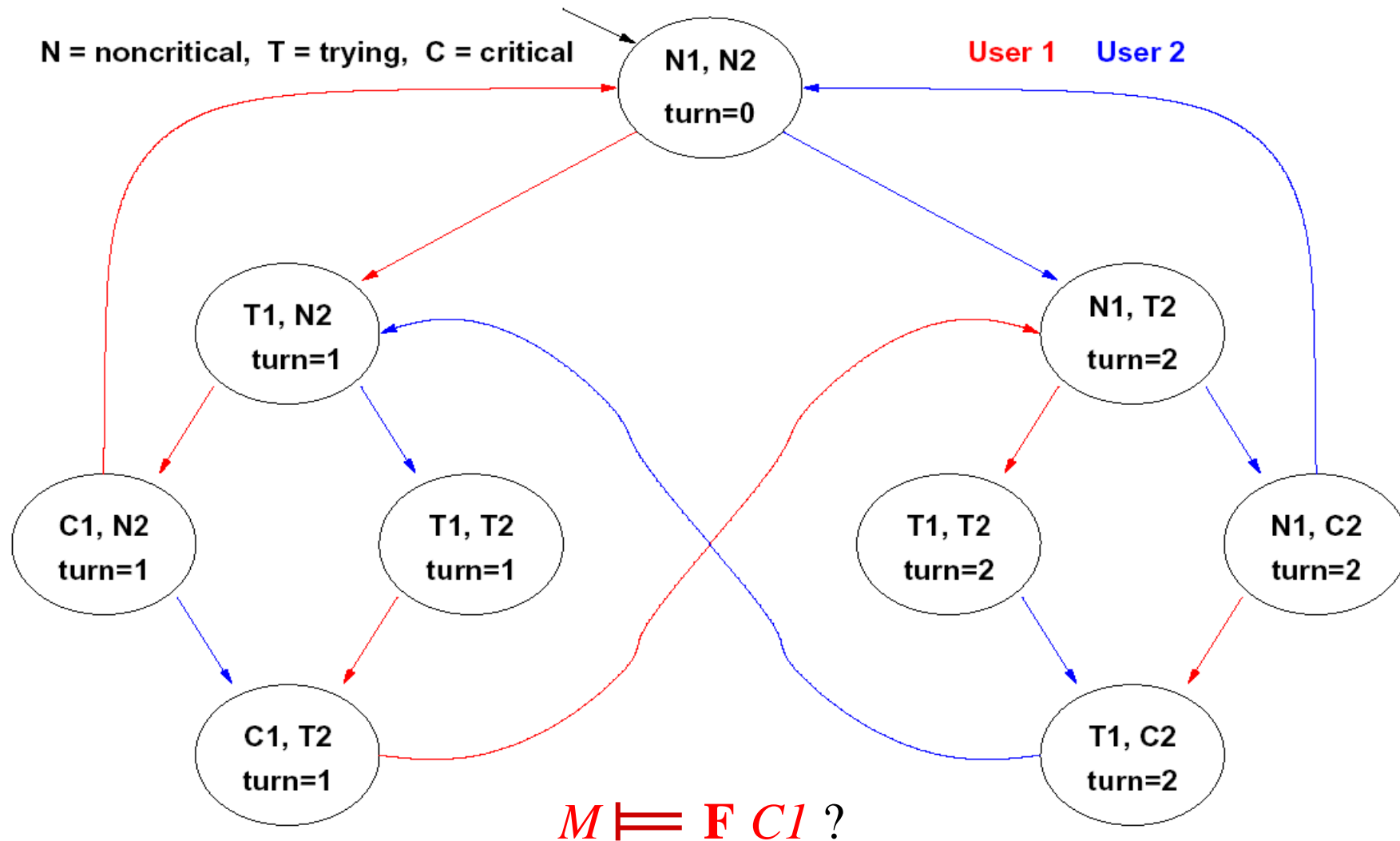
# Example: mutual exclusion (safety)



$M \models G \neg(C1 \wedge C2) ? \quad [M \models \neg F(C1 \wedge C2) ?]$

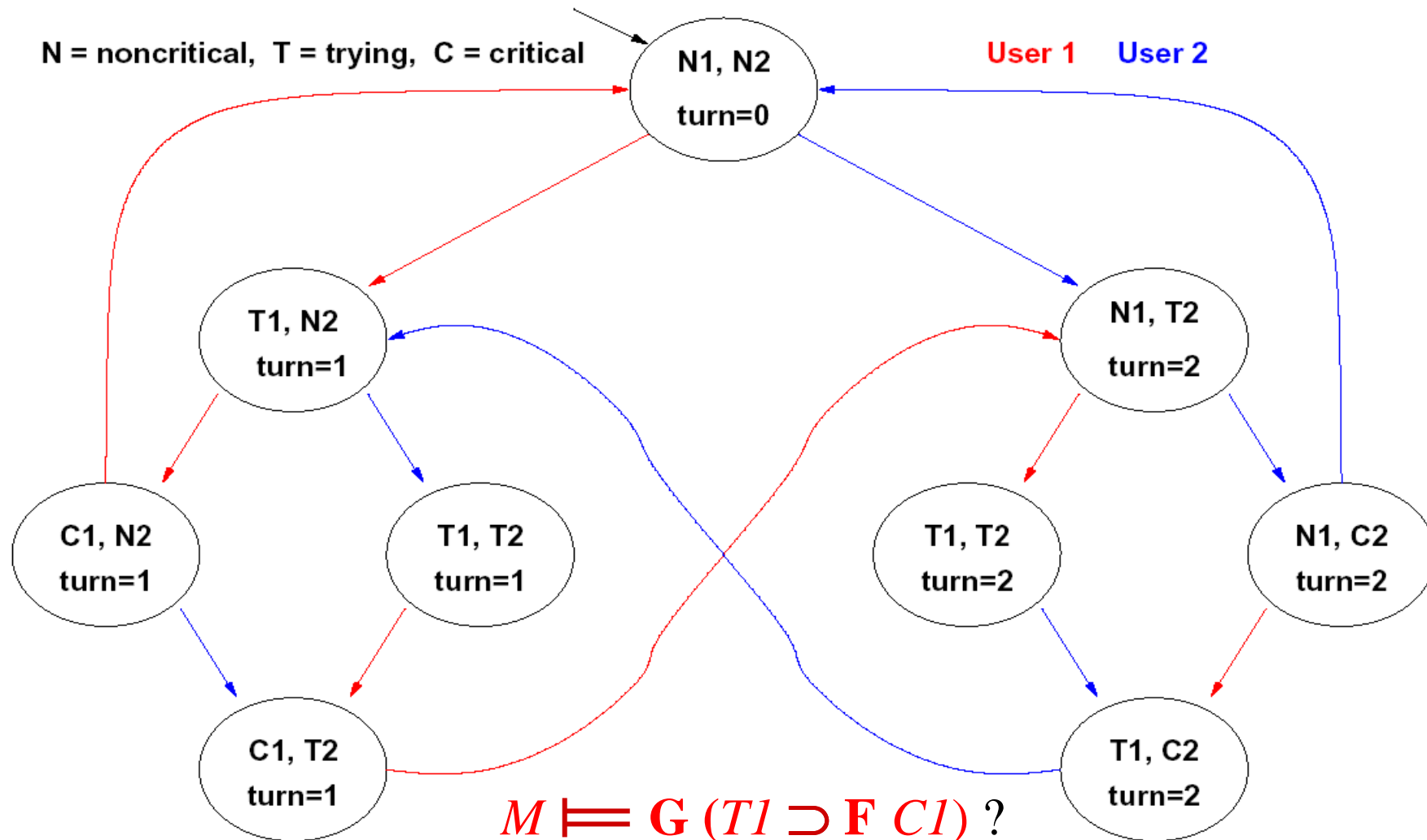
**YES:** There is no reachable state in which both **C1** and **C2** hold!

# Example: mutual exclusion (liveness)



**NO:** there is an infinite (cyclic) solution in which **C1** never holds!

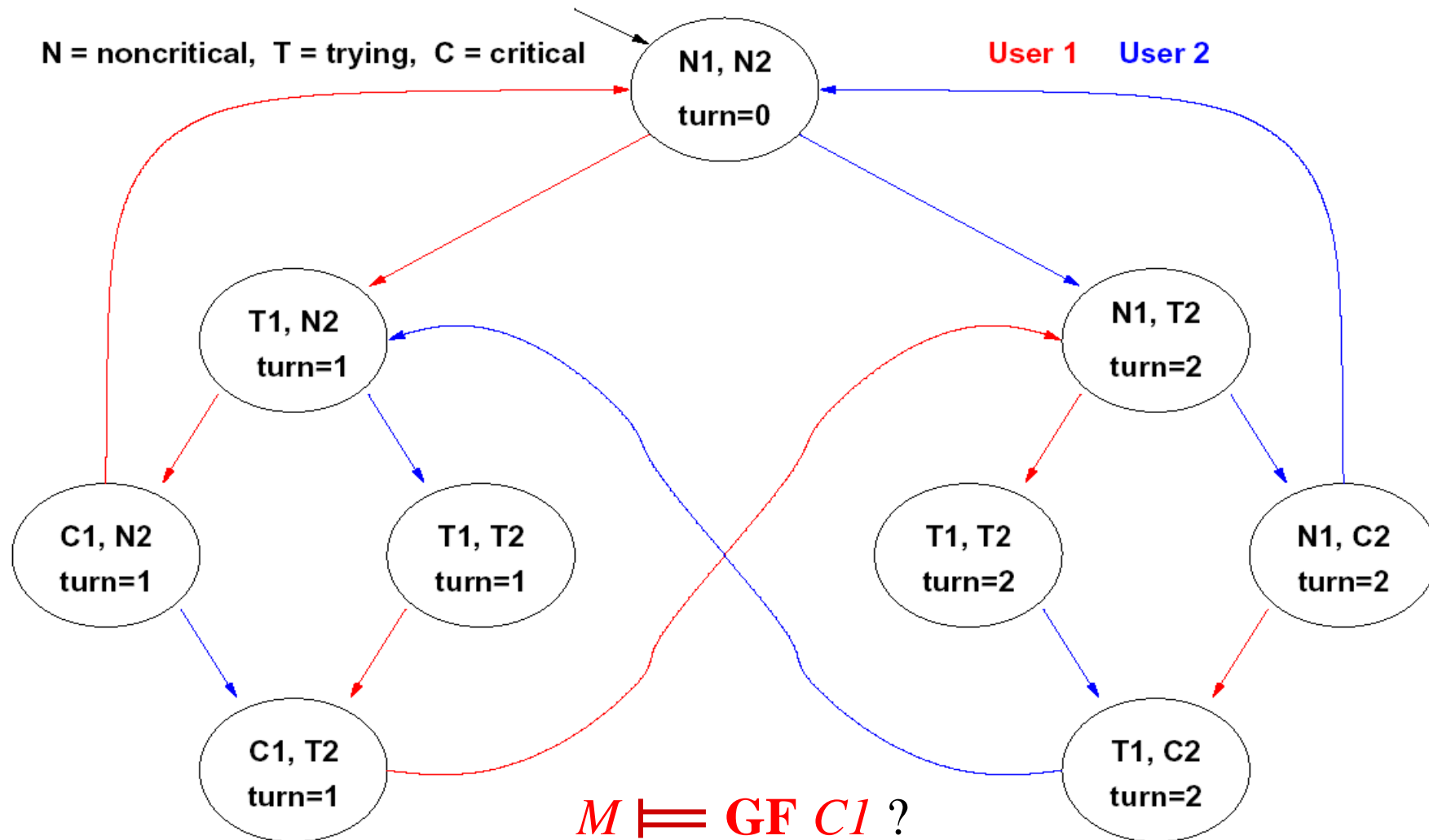
# Example: mutual exclusion (liveness)



**YES:** every path starting from each state where *T1* holds passes through a state where *C1* holds!

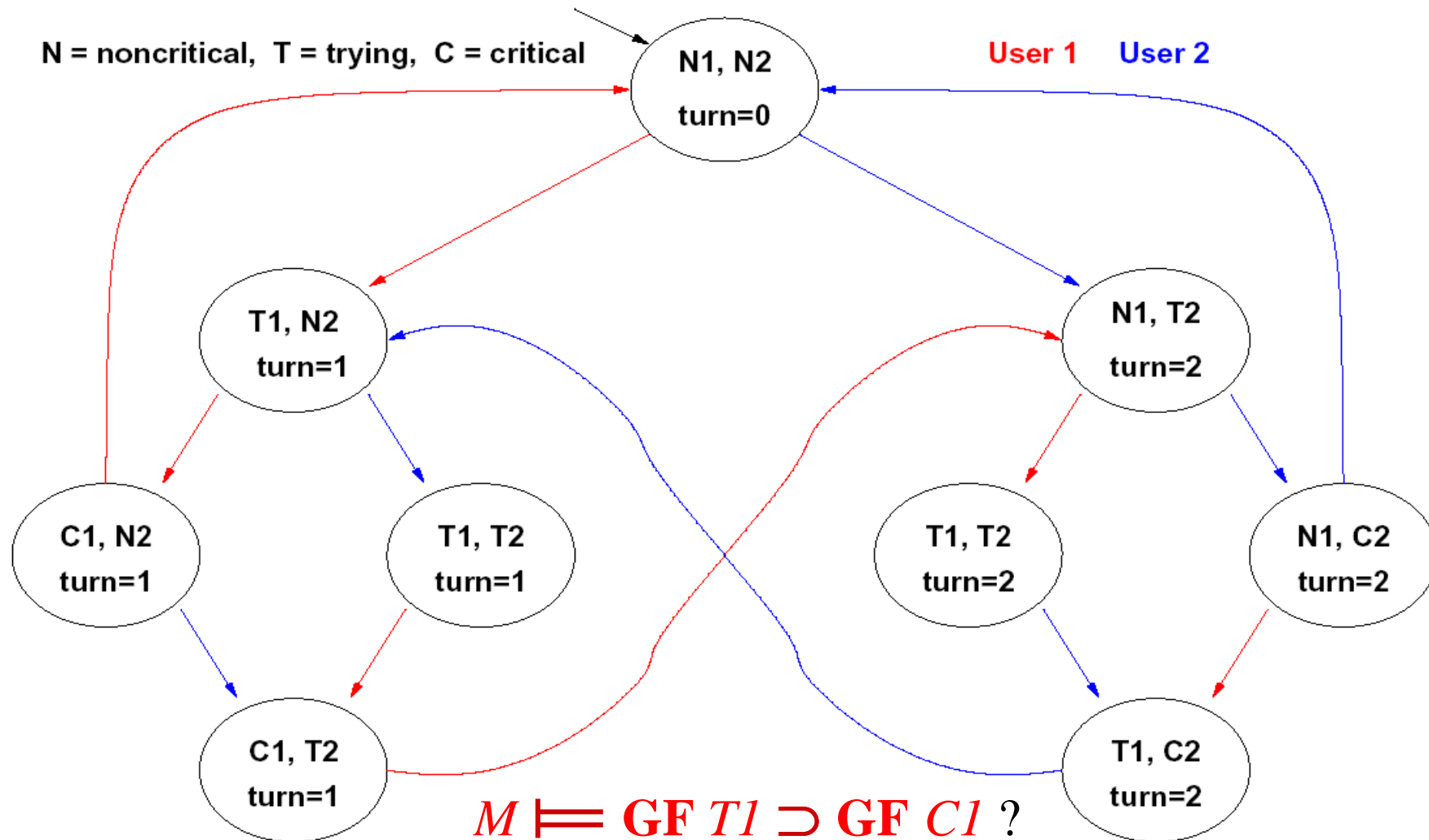


# Example: mutual exclusion (fairness)



**NO:** e.g., in the initial state, there is an infinite (cyclic) solution in which *C1* never holds!

# Example: mutual exclusion (strong fairness)



**YES:** every path which visits *T1* infinitely often also visits *C1* infinitely often (see liveness prop. in previous example)!

# Model Checking

- $\mathbf{K} = (\mathbf{S}, \mathbf{S}_0, \mathbf{R}, \mathbf{AP}, \mathbf{L})$  (the system)
- $\varphi$ , an **LTL** formula. (the property)
- $\mathbf{K} \models \varphi$  *iff* **every AP-computation** of  $\mathbf{K}$  is a model of  $\varphi$ .
- Determining this is the *model checking problem*.
- A solution to this problem can be automated!