Tecniche di Specifica e di Verifica

Linear Time Temporal Logic I

Temporal Logics: The context

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

Linear time temporal logics.

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

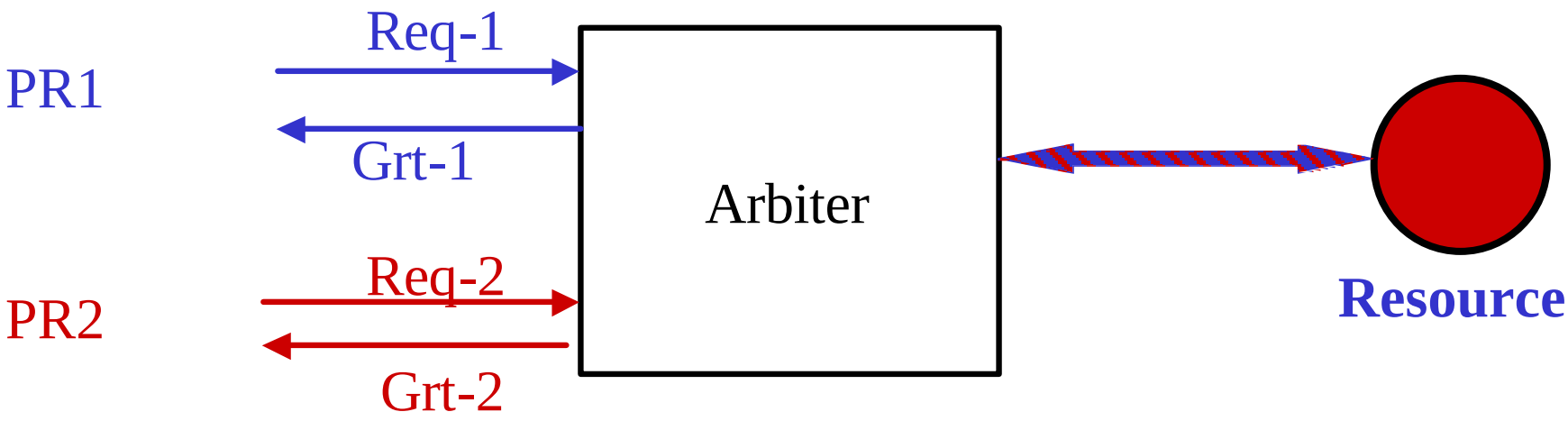
The Application

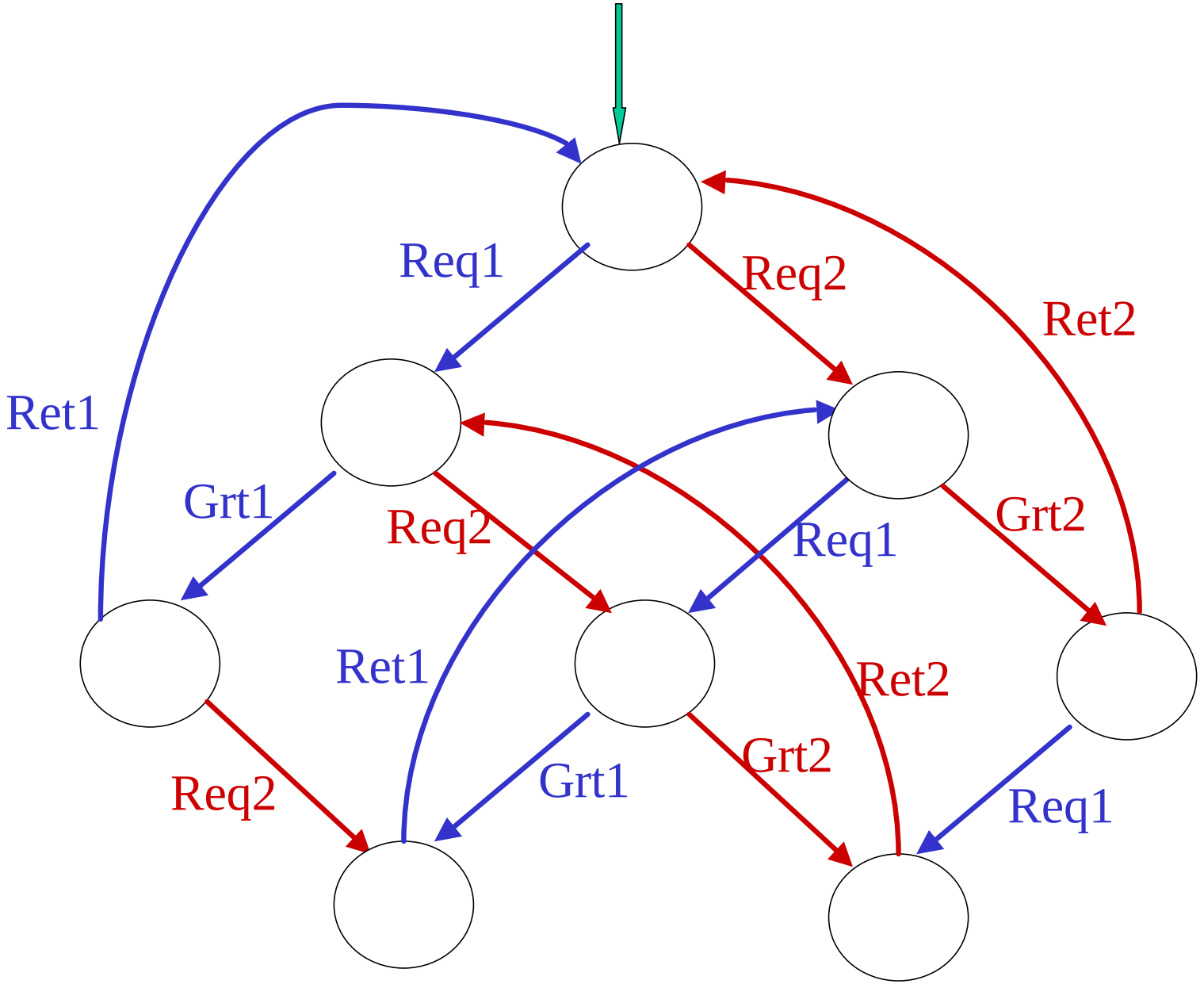
- Model a system to be verified as a Kripke structure:
  - Transition system **TS = (S, S_0, R)**
  - **AP** = A finite set of atomic propositions.
    - Basic assertions about the system
  - **L : S $\longrightarrow$ 2^{AP}** = The set of subsets of **AP**.
  - **p Î L(s)** ---- **p** is true at **s**.
  - **p Ï L(s)** ---- **p** is not true at **s**.
- **K = (S, S_0, R, AP, L)** ---- Kripke structure

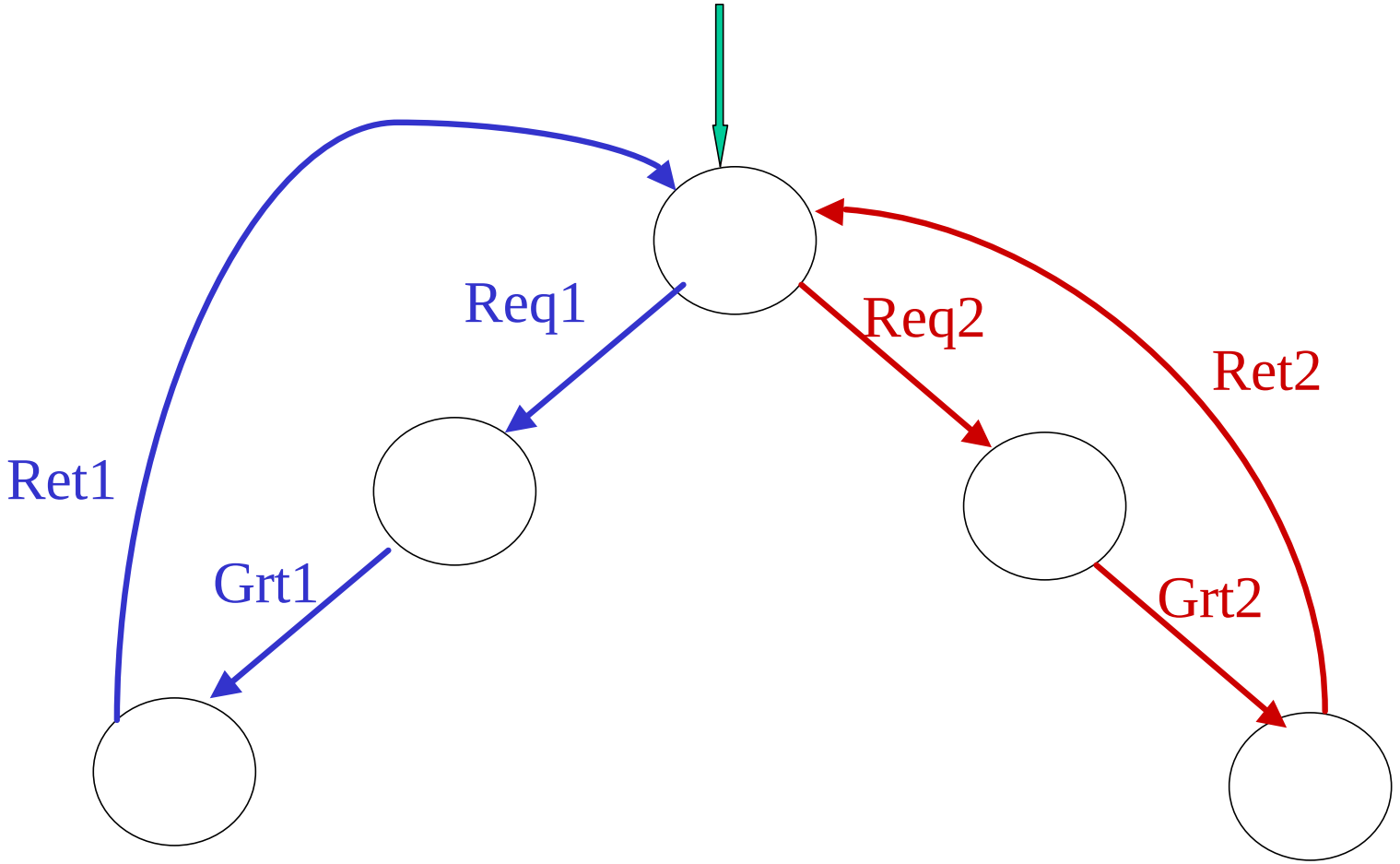
The Application

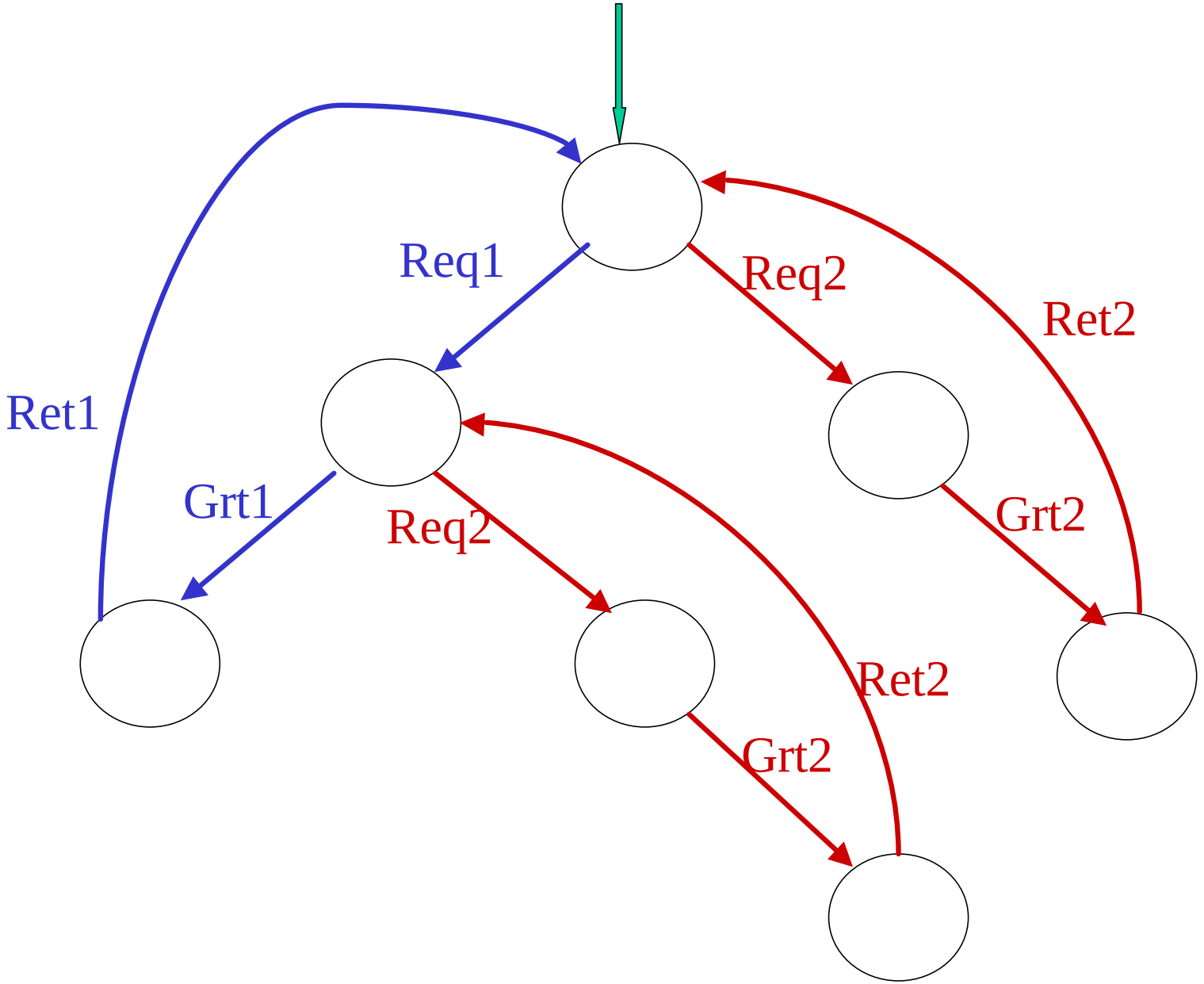
- ***The computations*** of the Kripke structure **K** will be the ***models*** for **LTL** formulas.
- ***The property*** to be verified is captured as an LTL formula φ.
- The modeled system **K** has the property φ ***iff every computation of*** **K** ***is a model of*** φ.
- We need to verify (***model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

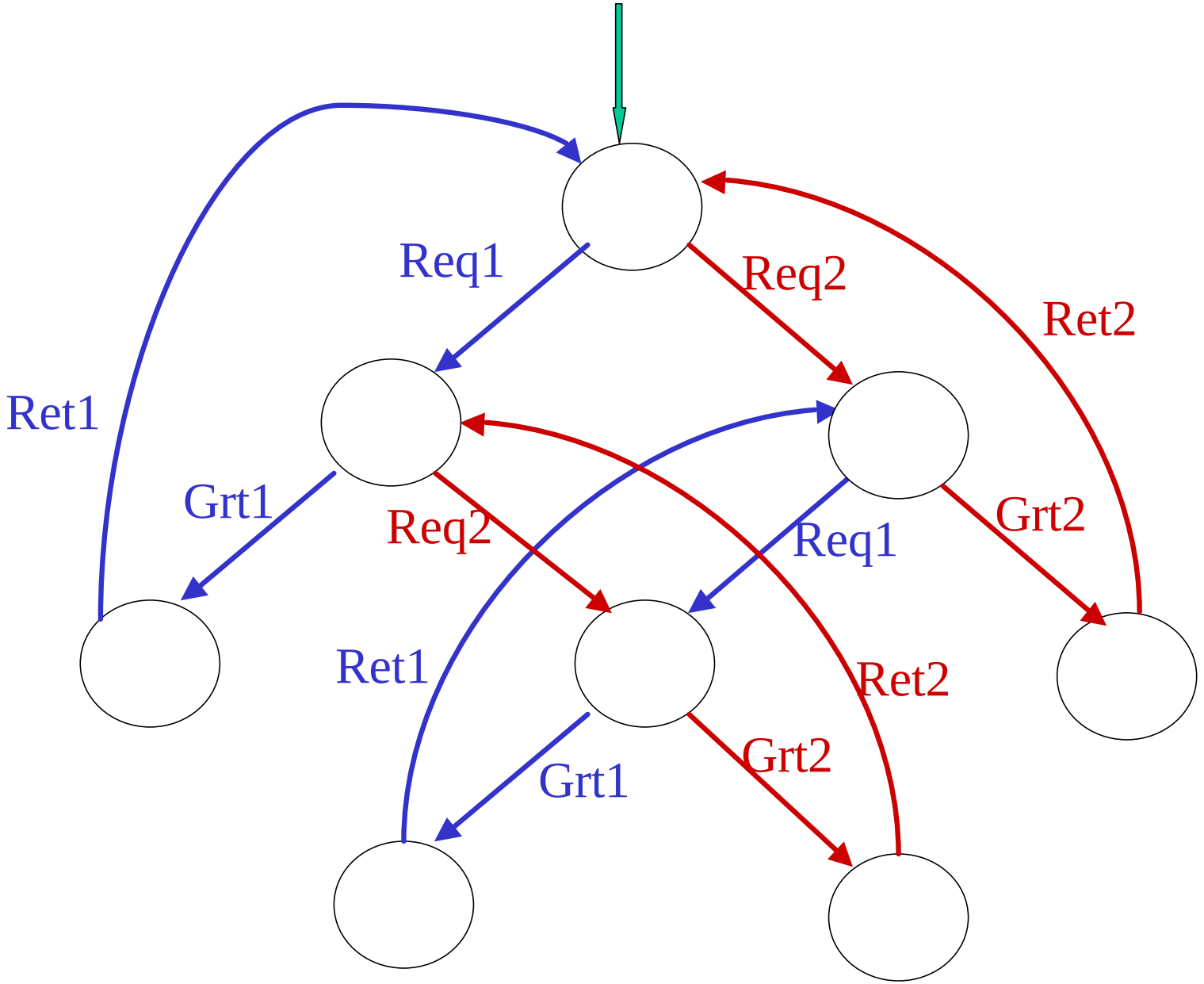
An Example

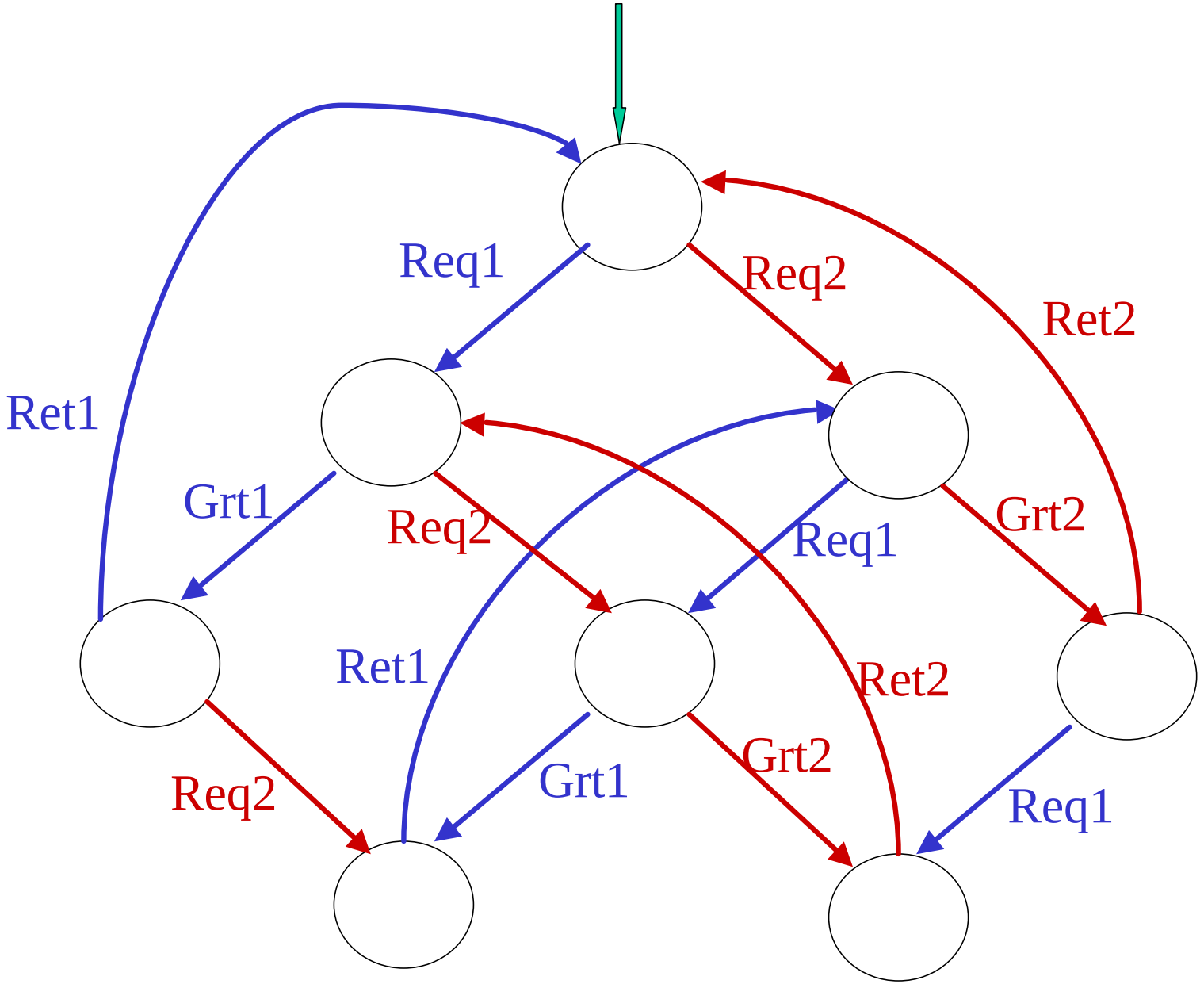



Req1
Req2
Ret2
Ret1
Grt1
Req2
Req1
Grt2
Ret1
Ret2
Req2
Grt1
Grt2
Req1

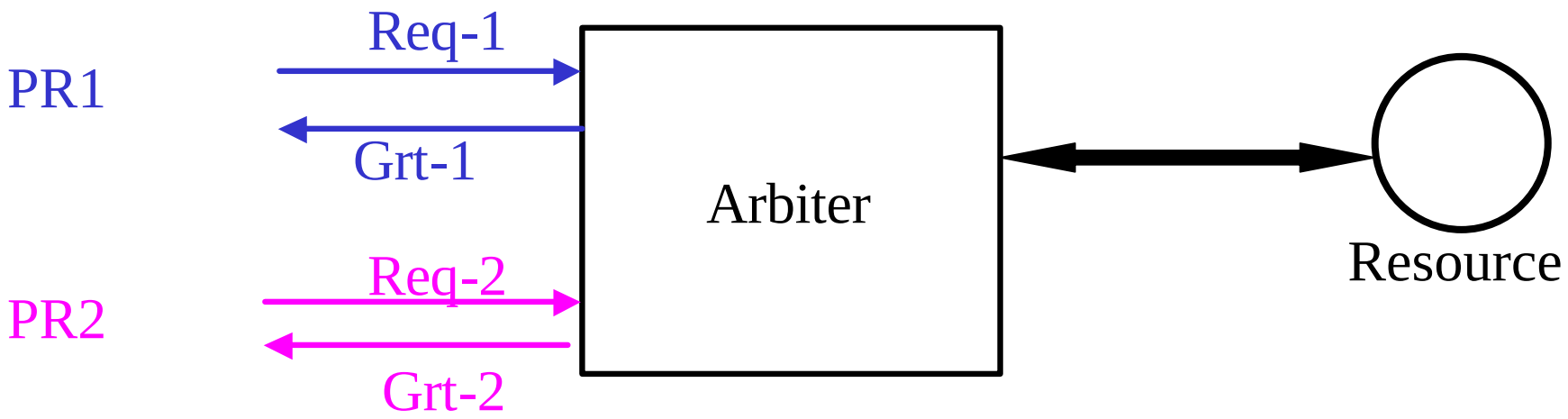

Req1
Req2
Ret2
Ret1
Grt1
Grt2


Req1
Req2
Ret2
Ret1
Grt1
Req2
Grt2
Ret2
Grt2


Req1
Req2
Ret2
Ret1
Grt1
Req2
Req1
Grt2
Ret1
Ret2
Grt2
Grt1


Req1
Req2
Ret2
Ret1
Grt1
Req2
Req1
Grt2
Ret1
Ret2
Req2
Grt1
Grt2
Req1

A set of Atomic Propositions



R1 – Process 1 is ***idle***

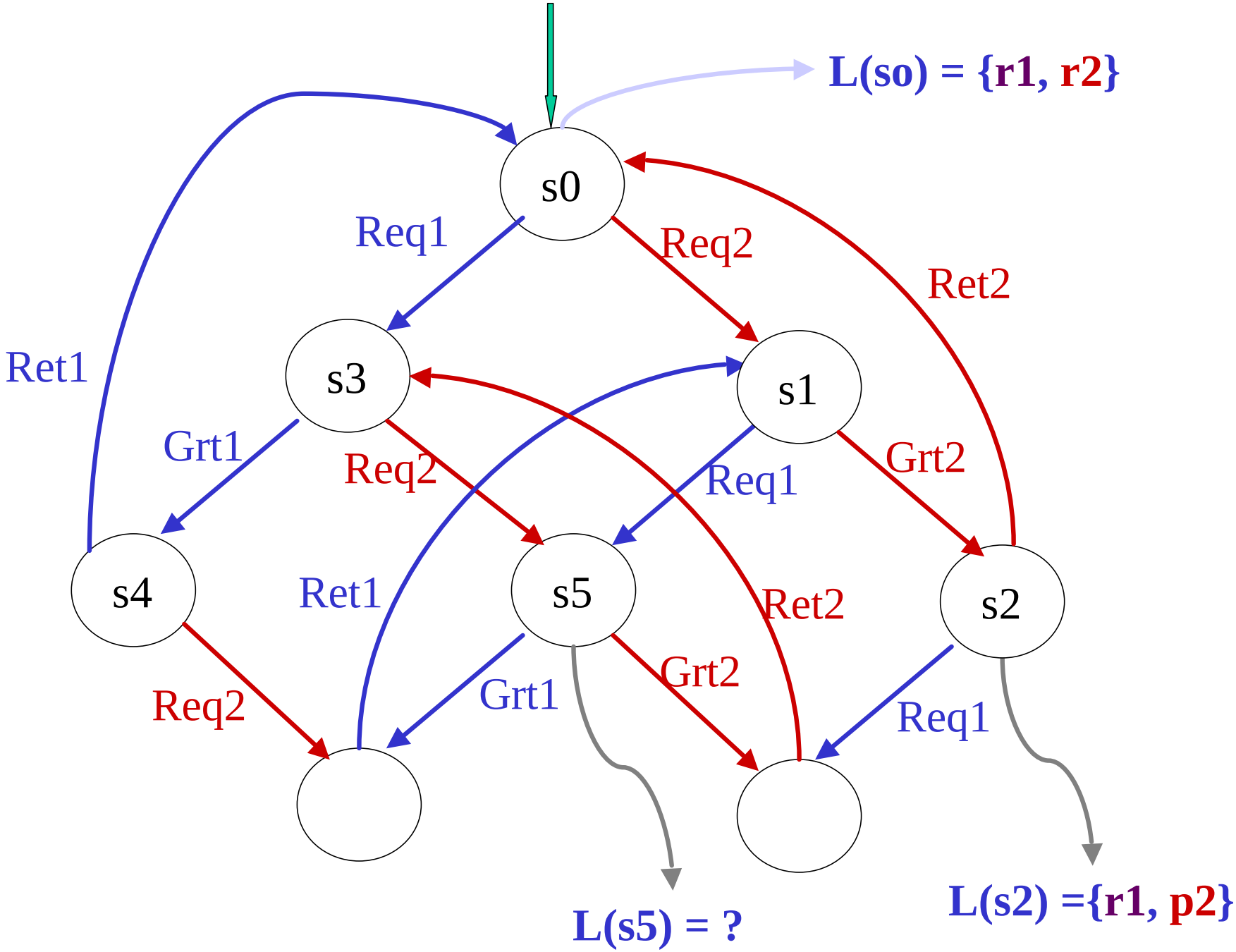
W1– Process 1 is ***waiting***

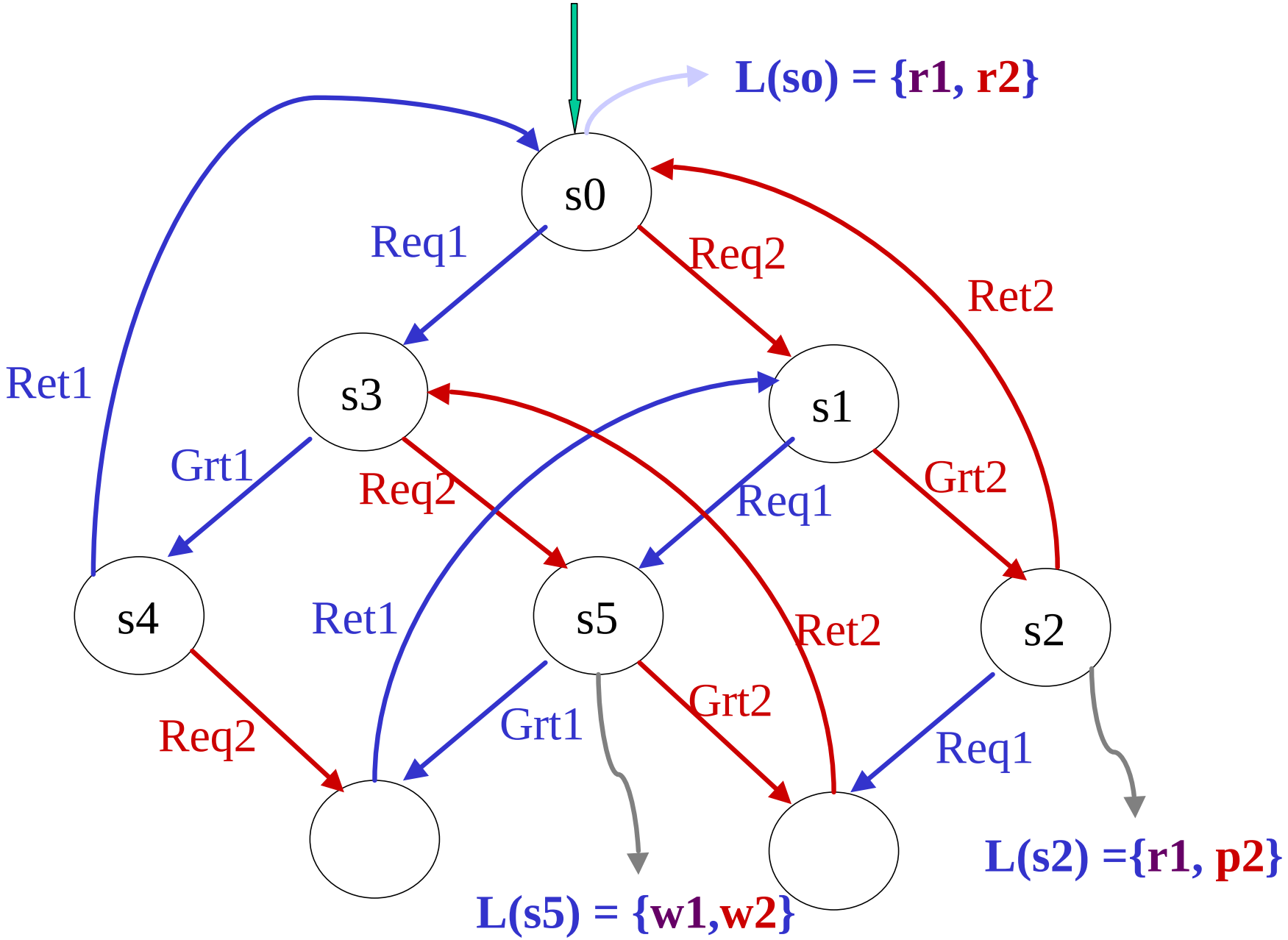
P1 – Process 1 is ***using*** the resource.

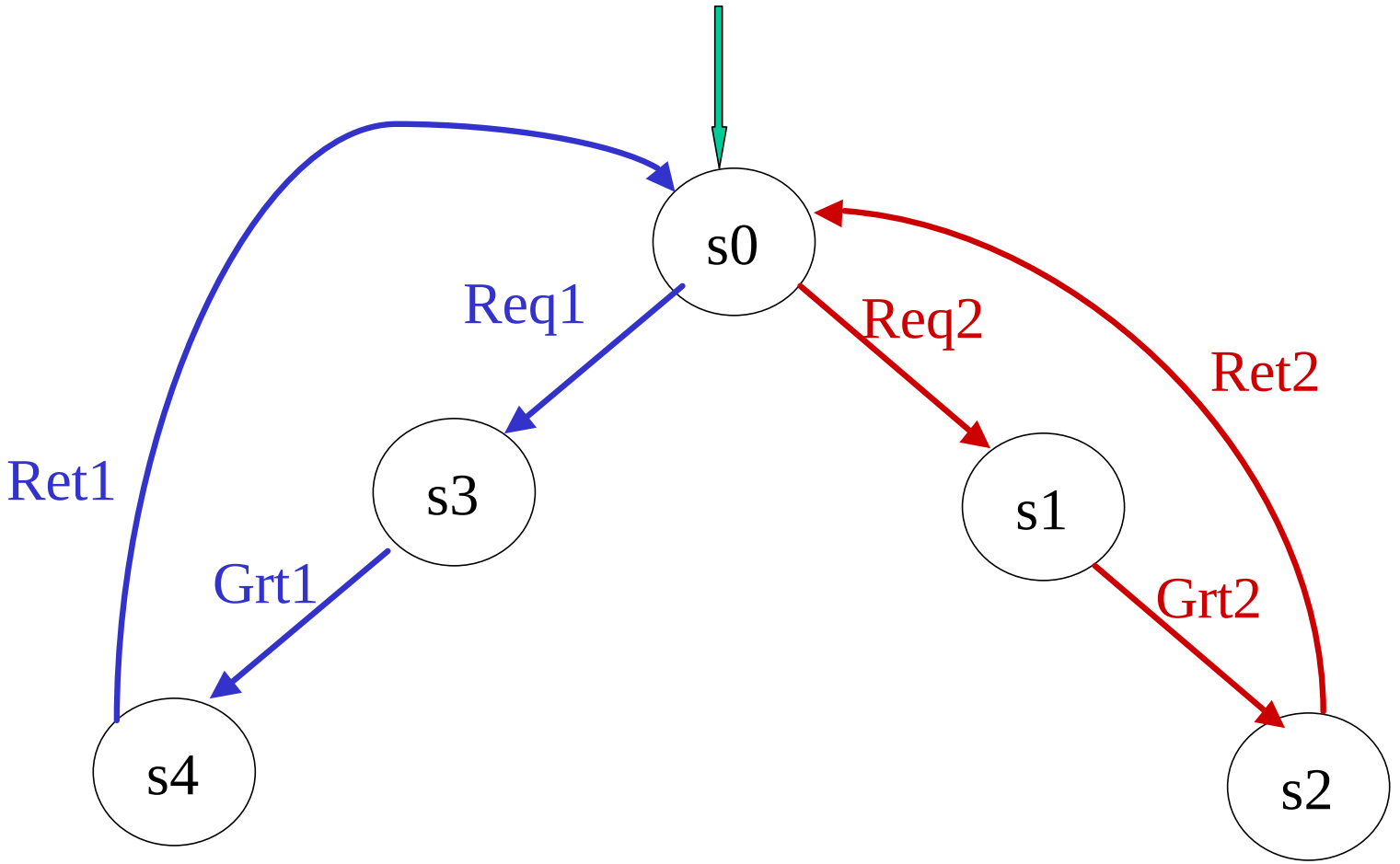
AP = { R1, W1, P1, R2, W2, P2}

The context

- Model a system to be verified as a Kripke structure:
  - Transition system **TS = (S, S_0, R)**
  - AP = A finite set of atomic propositions.
    - Basic assertions about the system
  - **L : S** $\longrightarrow$ $\mathbf{2^{AP}}$ = The set of subsets of AP.
  - **p** Î **L(s)** ---- **p** is true at **s**
  - **p** Ï **L(s)** ---- **p** is not true at **s**.
- **K = (S, S_0, R, AP, L)** ---- Kripke structure


L(so) = {r1, r2}
s0
Req1
Req2
Ret2
Ret1
s3
s1
Grt1
Req2
Req1
Grt2
s4
Ret1
s5
Ret2
s2
Req2
Grt1
Grt2
Req1
L(s5) = ?
L(s2) ={r1, p2}


L(so) = {r1, r2}
s0
Req1
Req2
Ret2
Ret1
s3
s1
Grt1
Req2
Req1
Grt2
s4
Ret1
s5
Ret2
s2
Req2
Grt1
Grt2
Req1
L(s2) ={r1, p2}
L(s5) = {w1,w2}



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

- **K = (S, S_0, R, AP, L)**
- Every computation (sequence of states) can be viewed as a sequence of subsets of **AP**.
- **s_0 s_1 s_2 ….** **---- L(s_0) L(s_1) L(s_2)** ….
- These **AP-computations** will be the models for the formulas of LTL.

- Verification :
  - Every **AP**-computation of **K** is a model of j

Linear Time Temporal Logic (LTL)

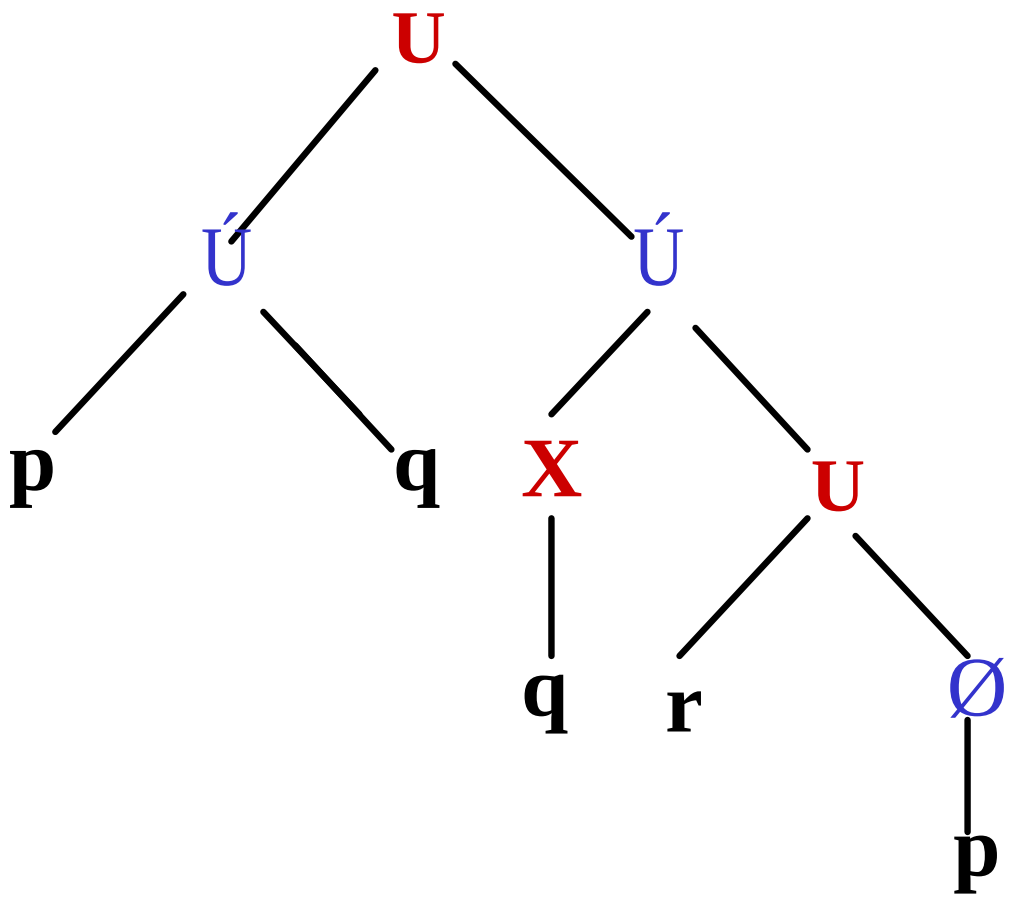
- Syntax :
  - **AP = {p_0, p_1, …p_n}**, a finite set of atomic propositions.
- Formulas :
  - Every **p_i** in AP is a ***LTL formula***.
  - If j is a formula then Ø j is a ***LTL formula***.
  - If j_1 and j_2 are formulas then **(j_1 Ú j_2)** is a ***LTL formula***.
  - If j is a formula then **X** j is a ***LTL formula*** (Next).
  - If j_1 and j_2 are formulas then **(j_1 U j_2)** is a ***LTL formula*** (Until).

Formulas

- **LTL ::= *p*** ½ Ø j ½ j_1 Ú j_2 ½**X** j ½ j_1 **U** j_2

- **p ; p**Ú**q ; (**Ø**p**Ú**q)**Ú Ø**(r**Ú**q)**
- **X q ; X(p**Ú**q) ; X((**Ø**p**Ú**q)** Ú **X**Ø**(r**Ú**q))**
- **(p**Ú**q)U(X r**Ú **(**Ø**q U (X**Ø**p)))**

(p Ú q) U (X q Ú (r U Ø p))



Semantics

- **AP** = A finite set of atomic propositions.
- $S = 2^{AP}$ = The set of subsets of **AP**
- **AP = { p, q, r }**
- S = { f, **{p}, {q}, {r}, {p,q}, {p,r}, {p,r}, {p,q,r}** }

- S^{w} = The set of infinite sequences over S.

Semantics

- **AP = {p, q, r}** S = $\mathbf{2^{AP}}$
- S = **{** f, **{ p }, { q }, ….,{ p, q, r } }**

s : **{ p,r } { q }** Æ **{ p, q, r } { r }**…

path: 0 → 1 → 2 → 3 → 4 …

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

Semantics

- S^w = The set of infinite sequences over Σ.
- $s \in S^w$ --- A model
- $s(i)$ ---- **i**-th position of s
- **{p} {q,r} Æ {r, q} {p, q, r}……..**

  | | | | | |

- **0 1 2 3 4 ……**
- **$s(0) = \{p\}$ $s(2) = Æ$ $s(3) = ?$**

Semantics

- **AP** $S = 2^{AP}$
- S^w = The set of infinite sequences over S.
- $s \in S^w$ --- A model
- $s(i)$ ---- **i**-th position of s
- j , a formula.
- $s(i) \models j$
  - $s(i)$ ***satisfies*** j
  - j is true in the **i**-th position of s

Semantics

- **LTL ::= *p*** ½Ø j ½ j_1 Ú j_2 ½**X** j ½ j_1 **U** j_2
- s = G_0 G_1 G_2 ….. G_i G_{i+1} ……
- Each G_j is a subset of **AP**.
- **s(i)⊨p *iff* p** Î G_i

Semantics

- **LTL ::= *p*** ½Ø j ½ j $_1$ Ú j $_2$ ½**X** j ½ j $_1$ **U** j $_2$
- **AP = {p, q, r}**
- s = **{p,q} {r}** Æ **{q, r} {p, q, r} …..**

0 1 2 3 4

- s**(0)** ***satisfies*** **q**
- s**(1)** ***satisfies*** **r**
- s**(2)** does ***not*** *satisfy* **q !**

Semantics

- **LTL ::= *p*** ½Ø j ½ j_1 Ú j_2 ½**X** j ½ j_1 **U** j_2
- s = G_0 G_1 G_2 ….. G_i G_{i+1} ……
- Each G_j is a subset of **AP**.
- **s(i) ⊨ Ø j** ***iff*** **s(i) ⊭ j**

Semantics

- **LTL ::= *p*** ½Ø j ½ j_1 Ú j_2 ½**X** j ½ j_1 **U** j_2
- s = G_0 G_1 G_2 ….. G_i G_{i+1} ……
- Each G_j is a subset of **AP**.
- **s(i) ⊨ j_1 Ú j_2** ***iff*** **s(i) ⊨ j_1** ***OR*** **s(i) ⊨ j_2**

Semantics

- **LTL ::= *p*** ½Ø j ½ **j** $_1$ Ú **j** $_2$ ½**X** j ½ **j** $_1$ **U j** $_2$
- **AP = {p, q, r}**
- s = **{p, q} {r}** Æ **{q, r} {p, q, r} …..**

  0 1 2 3 4
- s**(0)** ***satisfies*** Ø**r** ; s**(0)** does ***not satisfy*** **r**
- s**(1)** ***satisfies*** **p** Ú **r** ; s**(1)** ***satisfies*** **r**
- s**(2)** ***satisfies*** Ø**(p** Ú **r) ?**

Semantics

- **LTL ::= *p*** ½Ø j ½ j $_1$ Ú j $_2$ ½**X** j ½ j $_1$ **U** j $_2$
- **AP = {p, q, r}**
- s = **{p, q} {r}** Æ **{q, r} {p, q, r) …..**

  0 1 2 3 4
- s**(2)** ***satisfies*** Ø**(p** Ú **r)** ? ***Yes!***
- s**(2)** does ***not satisfy*** **p** Ú **r**

Semantics

- **LTL ::= *p*** ½Ø j ½ j $_1$ Ú j $_2$ ½**X** j ½ j $_1$ **U** j $_2$
- s = G_0 G_1 G_2 ….. G_i G_{i+1} ……..
  X j j

- **s(i) ⊨ X** j ***iff*** **s(i+1) ⊨** j

Semantics

- **LTL ::= *p*** ½Ø j ½ j $_1$ Ú j $_2$ ½**X** j ½ j $_1$ **U** j $_2$
- **AP = {p, q, r}**
- s = **{p,q} {r}** Æ **{q,r} {p,q,r) …..**

  0 1 2 3 4
- s**(2)** ***satisfies*** **X r** ; s**(3)** ***satisfies*** **r**
- s**(0)** ***satisfies*** **X(p** Ú **r)** ; s**(1)** ***satisfies*** **r**
- s**(1)** does ***not satisfy*** **X(p** Ú **r)**
  - s**(2)** does ***not satisfy*** **p** Ú **r**

Semantics

- **LTL ::= *p* ½Ø j ½ j $_1$ Ú j $_2$ ½**X** j ½ j $_1$ **U** j $_2$**
- **AP = {p, q, r}**
- s = **{p,q} {r} Æ {q,r} {p,q,r} …..**

  0 1 2 3 4
- s**(1)** ***satisfies*** **X(X Ø p)** ***iff***
  - s**(2)** ***satisfies*** **X Ø p** ***iff***
  - s**(3)** ***satisfies*** **Ø p** ***iff***
  - s**(3)** does ***not satisfy*** **p**

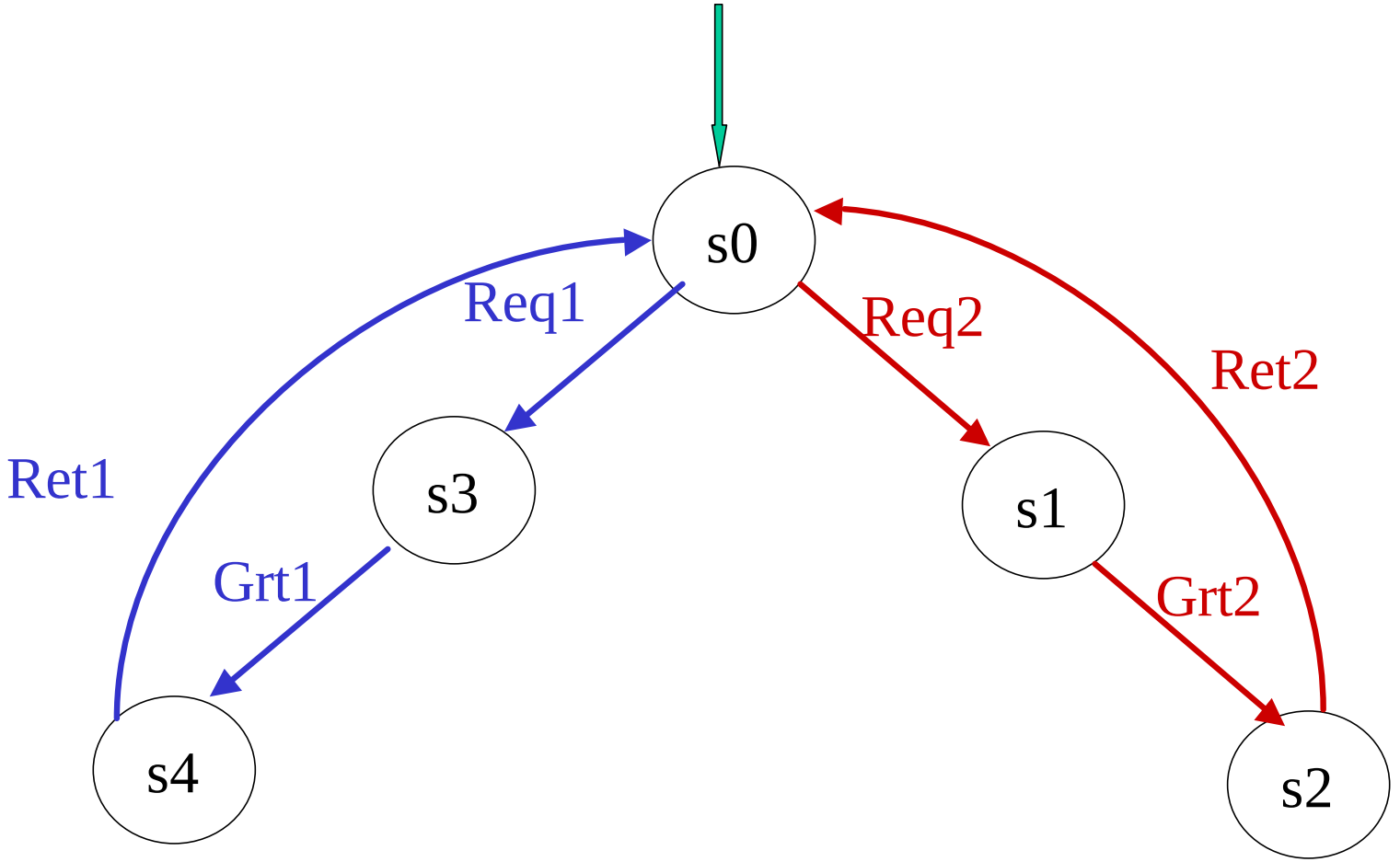
Semantics

- **LTL ::= *p*** ½Ø j ½ j_1 Ú j_2 ½**X** j ½ j_1 **U** j_2
- s = G_0 G_1 … G_i G_{i+1} …. G_{k-1} G_k …
  j_1… j_1 … j_1 j_2

- **s(i)** ⊨ j_1 **U** j_2 ***iff*** there exists **k** ³ **i** s.t.
  - **s(k)** ⊨ j_2
  - **s(j)** ⊨ j_1 for every **i £ j < k**

Semantics

- **LTL ::= *p*** ½Ø j ½ j **1** Ú j **2** ½**X** j ½ j **1** **U** j **2**
- **k** could be arbitrarily greater than **i**.
- **k = i** is allowed and there is **no i** £ **j** < **k**
- s**(i)**⊨ j **1** **U** j **2** ***iff*** there exists **k** ³ **i** s.t.
  - s**(k)**⊨ j **2**
  - s**(j)**⊨ j **1** for every **i** £ **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

- s**(1)** ***satisfies*** **(r2 U w2)** ;
  - s**(4)** ***satisfies*** **w2** and
  - s**(1),** s**(2),** s**(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

- s**(1)** does ***not satisfy*** **(r2 U p2)** ;
  - s**(5)** ***satisfies*** **p2** and
  - s**(1),** s**(2),** s**(3)** ***satisfy*** **r2**.
  - but s**(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

- s**(1)** ***does satisfy*** **((r2** Ú **w2) U p2)** ;
  - s**(5)** ***satisfies*** **p2** and
  - s**(1),** s**(2),** s**(3)** ***satisfy*** **r2, hence also (r2** Ú **w2).**
  - s**(4)** ***satisfies*** **w2, hence also (r2** Ú **w2) !**

Models

- **AP** $S^{AP} = 2$
- S^{ω} = The set of infinite sequences over S.
- $s \hat{I} S^{\omega}$
- j an **LTL** formula.

- A path s is a ***model*** of j **(** $s \models j$ **)** ***iff***
  - s**(0)** $\models$ j

Validity in LTL

- **AP** $S^{AP} = 2$
- S^{ω} = The set of infinite sequences over S.
- $s \in S^{\omega}$
- φ an **LTL** formula.

- φ is ***LTL-valid*** **(** $\models \varphi$ **)** ***iff for every*** $s \in S^{\omega}$ $-$ $s \models \varphi$

Derived Operators

- **LTL ::= *p*** ½Ø j ½ j_1 Ú j_2 ½**X** j ½ j_1 **U** j_2
- j_1 Ù j_2 --- Ø (Ø j_1 Ú Ø j_2) (**And**)
- j_1 É j_2 --- Ø j_1 Ú j_2 (**Implies**)
- j_1 ° j_2 ---- (j_1 É j_2) Ù (j_2 É j_1) (**Iff**)
- **AP = {p_1, p_2, …,p_n}**
- T ---- p_1 Ú Øp_1 (**true**)

- **Fact : In *every model*** s, ***at every*** **i**,
  - s**(i)** ⊨ T

Derived Operators

- **LTL ::= *p*** ½Ø j ½ j $_1$ Ú j $_2$ ½**X** j ½ j $_1$ **U** j $_2$
- **F** j **---- (**⊤ **U** j **)** (future ; diamond: ◇)

- **Fact** :
  - s**(i)** ⊨ **F** j *iff* **there exists k** ³ **i** such that s**(k)** ⊨ j .

Derived Operators

- **Fact** :
  - $s(i) \models F\, j$ *iff* **there exists k** $\geq$ **i** such that $s(k) \models j$.

Proof:

$s(i) \models F\, j$ *iff*

$s(i) \models (\top\ U\ j)$ *iff*

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

$\exists\, k \geq i,\ s(j) \models j$

Derived Operators

- **LTL ::= *p*** ½Ø j ½ j **1** Ú j **2** ½**X** j ½ j **1** **U** j **2**
- **F** j **----** **(**⊤ **U** j **)**
- **G** j **----** Ø**F**Ø j (invariant; box: □)

- **Fact** :
  - s**(i)** ⊨ **G** j *iff* **for every k** ³ **i** ,
    s**(k)** ⊨ j .

Derived Operators

- **LTL ::= *p* ½Ø j ½ j 1 Ú j 2 ½X j ½ j 1 U j 2**
- **(y R j) ---- Ø (Ø y U Ø j)** (Releases)
- **G j ---- (^ R j)**

- **Fact** :
  - s**(i)** ⊨ (y **R** j) ***iff*** **for every k** ³ **i**
    - s**(k)** ⊨ j ***or***
    - **for some i** £ **j < k,** s**(j)** ⊨ y

Derived Operators

- **LTL ::= *p*** ½Ø j ½ j $_1$ Ú j $_2$ ½**X** j ½ j $_1$ **U** j $_2$
- **(** y **W** j **)** **----** **G** y Ú **(** y **U** j **)** (Unless)

Derived Operators

- **LTL ::= *p*** ½Ø j ½ j $_1$ Ú j $_2$ ½**X** j ½ j $_1$ **U** j $_2$
- **(y B j)** **----** **?????** (Before)

Derived Operators

- **LTL ::= *p*** ½Ø j ½ j **1** Ú j **2** ½**X** j ½ j **1** **U** j **2**
- **(y B j)** **----** Ø **(** Ø y **U** j **)** (Before)
- Why???? Prove that it matches the intuition!

Tableau rules

- **(**y **U** j**)** ≡ j Ú (y Ù **X (**y **U** j**)** **)**
- **(**y **R** j**)** ≡ j Ù (y Ú **X (**y **R** j**)** **)** ≡
  ≡ (j Ù y) Ú (j Ù **X (**y **R** j**)** **)**
- **F** j ≡ j Ú **X F** j
- **G** j ≡ j Ù **X G** j

LTL: Some examples

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

$$\mathbf{G}(\varnothing\ (\mathbf{P}_A \wedge \mathbf{P}_B))$$

- **Liveness:** “if A waiting, then it will eventually print”

$$\mathbf{G}(\mathbf{W}_A \rightarrow \mathbf{F}\ \mathbf{P}_A)$$

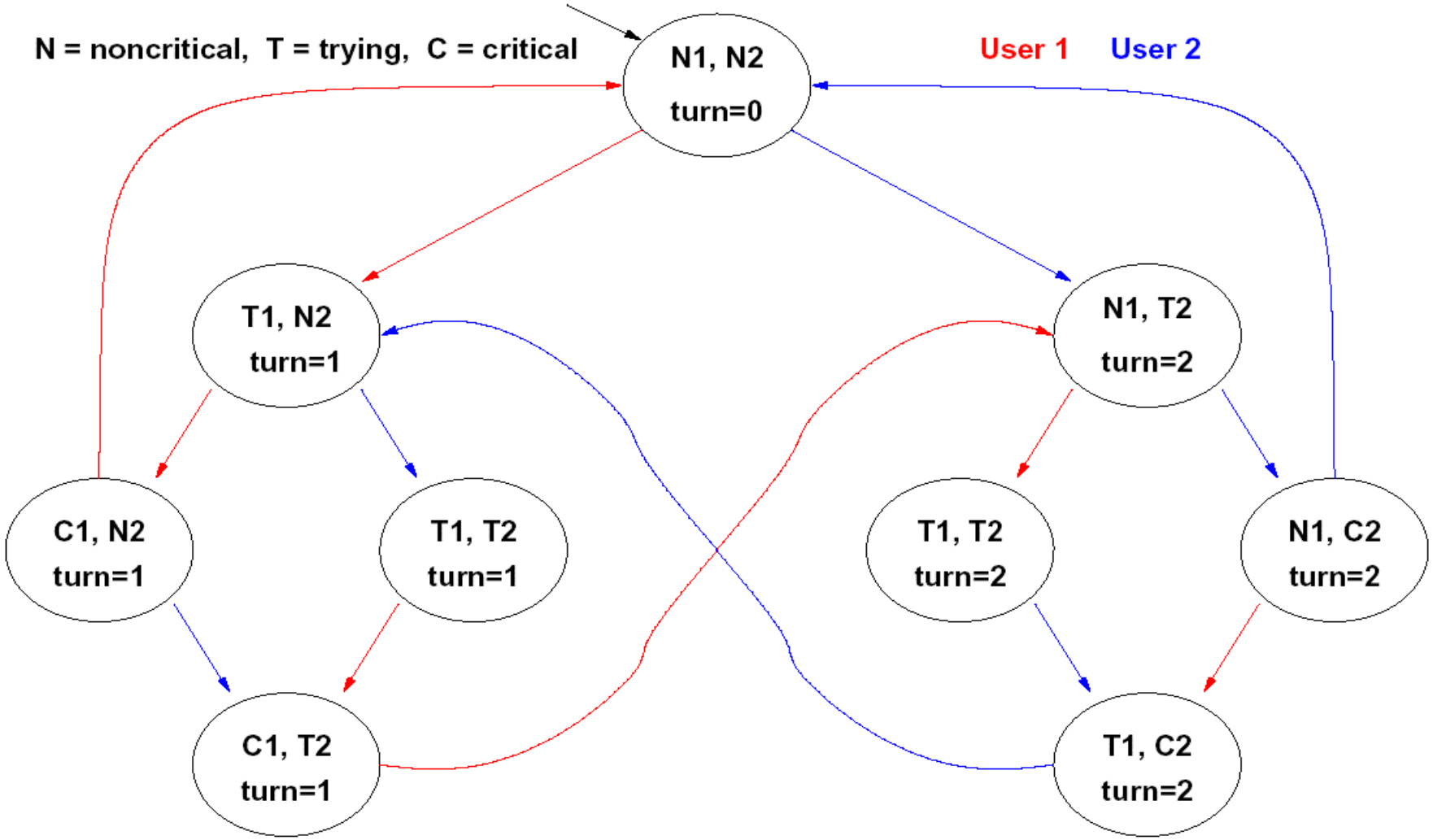
- **Fairness:** “A infinitely often idle”

$$\mathbf{GF}\ \mathbf{R}_A$$

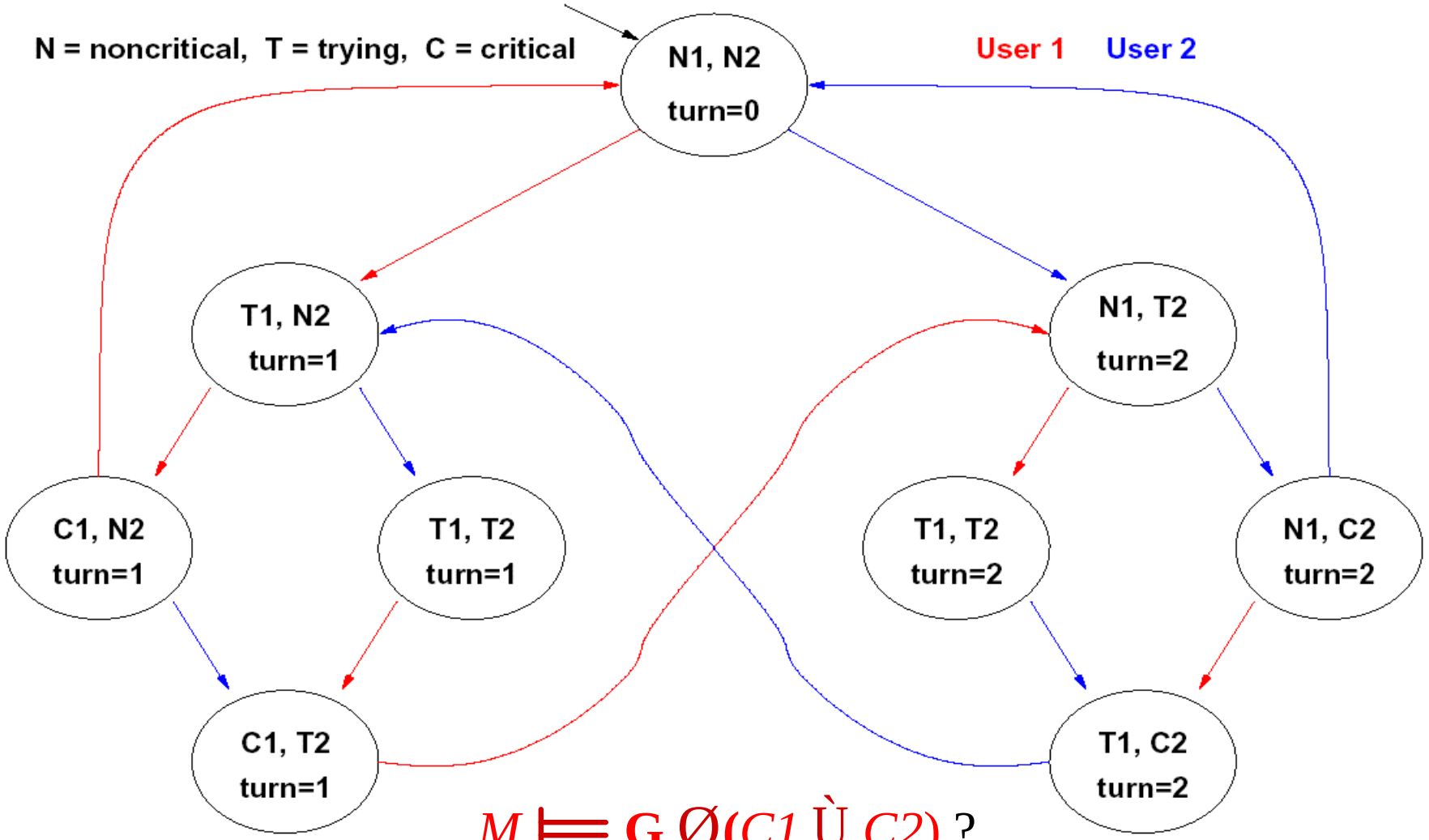
- **Strong fairness:** “if A infinitely often waiting, then it will infinitely often printing”

$$\mathbf{GF}\ \mathbf{W}_A \rightarrow \mathbf{GF}\ \mathbf{P}_A$$

Example: mutual esclusion



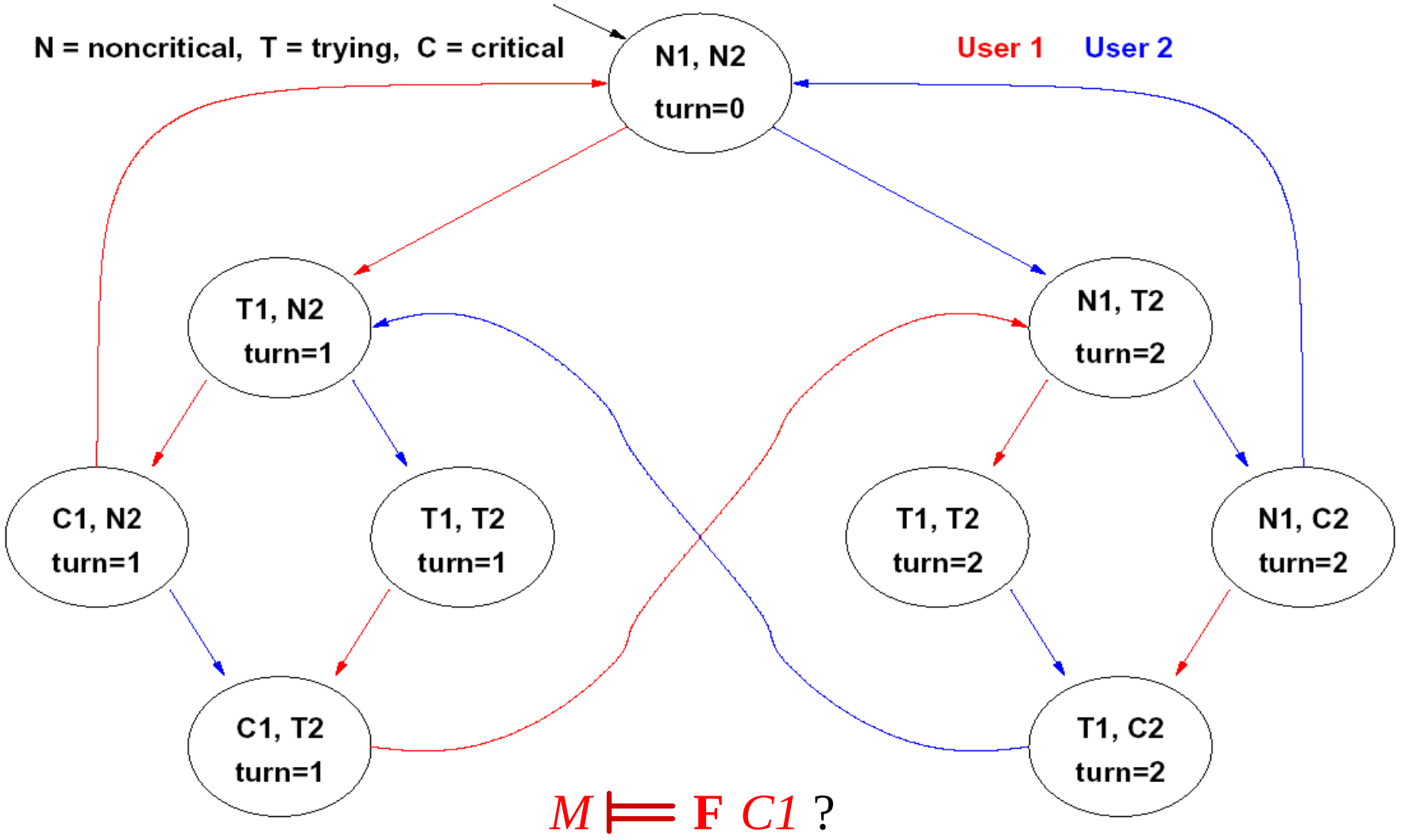
Example: mutual esclusion (safety)



$M \models \mathbf{G}\,\emptyset(C1 \wedge C2)$?

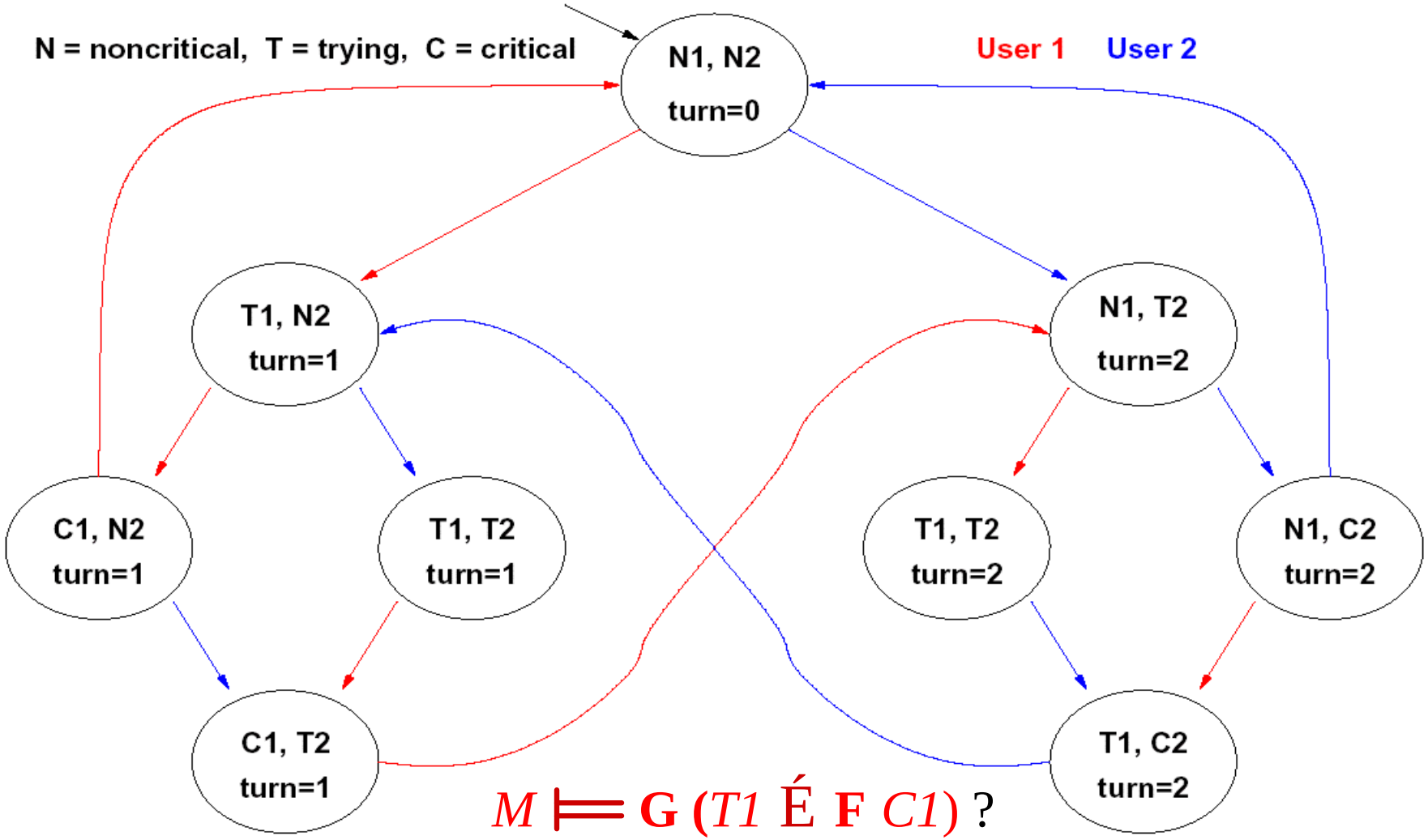
YES: There is no reachable state in which both C1 and C2 hold!

Example: mutual exclusion (liveness)



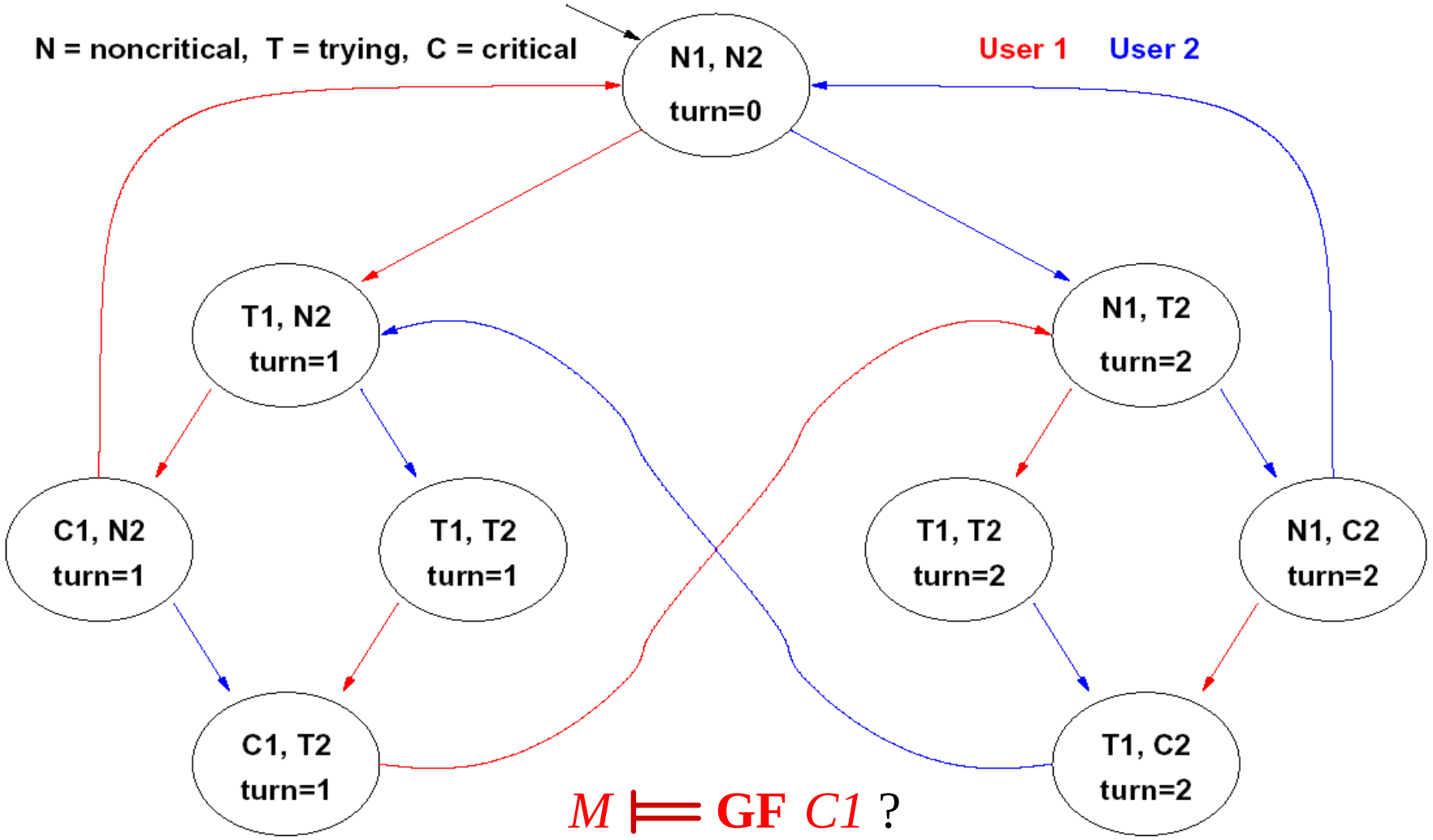
NO: there is an infinite cyclic solution in which C1 never holds!

Example: mutual exclusion (liveness)



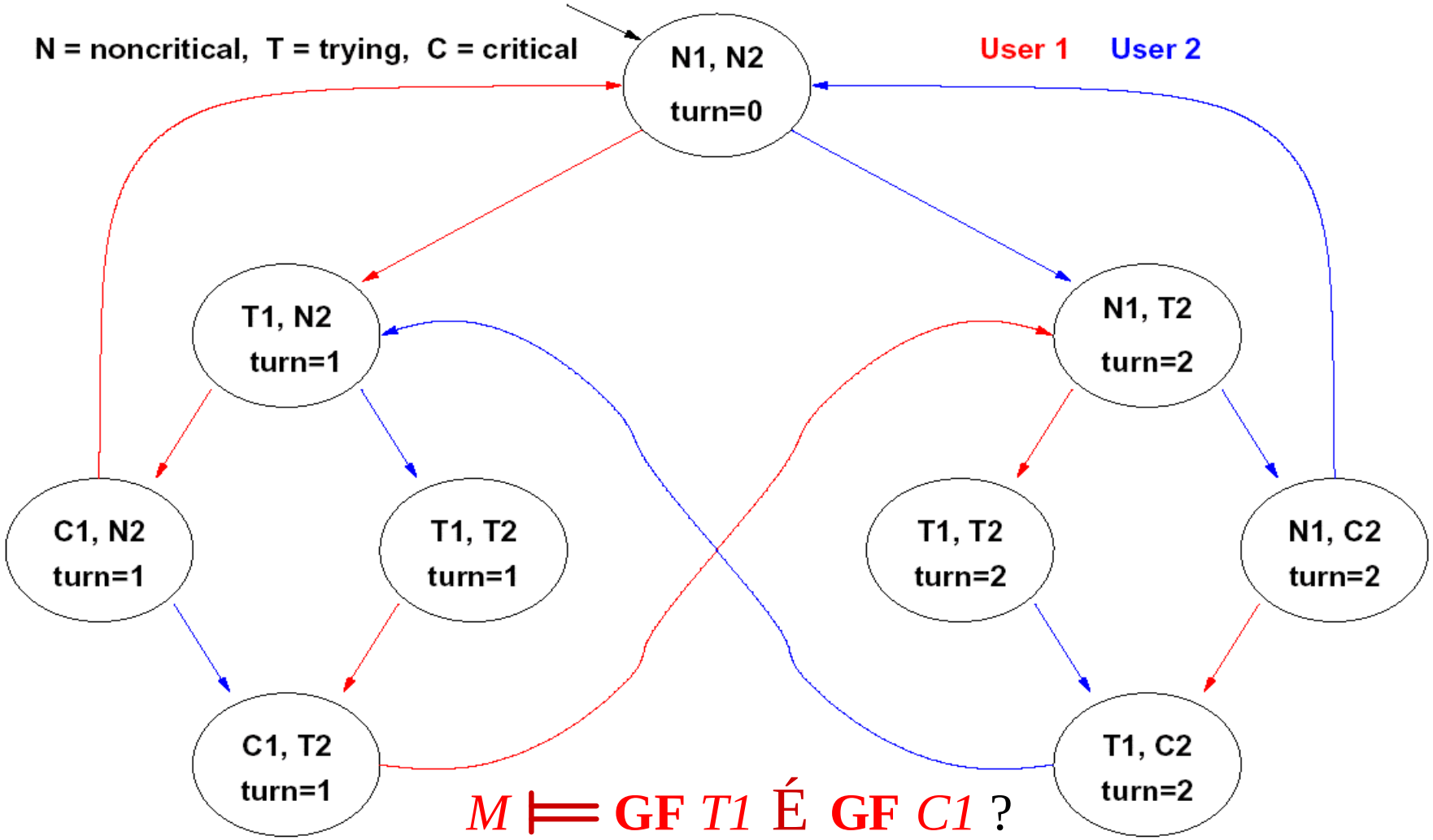
YES: every path starting from each state where *T1* holds passes through a state where *C1* holds!

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

- **K = (S, S_0, R, AP, L)** (the system)
- φ , an **LTL** formula. (the property)
- **K** $\models$ φ ***iff*** **every AP-computation** of **K** is a model of φ .
- Determining this is the ***model checking problem***.
- A solution to this problem can be automated!