

# CONCEPTION ET VÉRIFICATION DE SYSTÈMES CRITIQUES

## LA SPÉCIFICATION DES PROPRIÉTÉS AVEC LA LOGIQUE LTL

🎓 2A Cursus Ingénieurs - ST5 : Modélisation fonctionnelle et régulation

🏛️ CentraleSupélec - Université Paris-Saclay - 2025/2026



**Idir AIT SADOUNE**

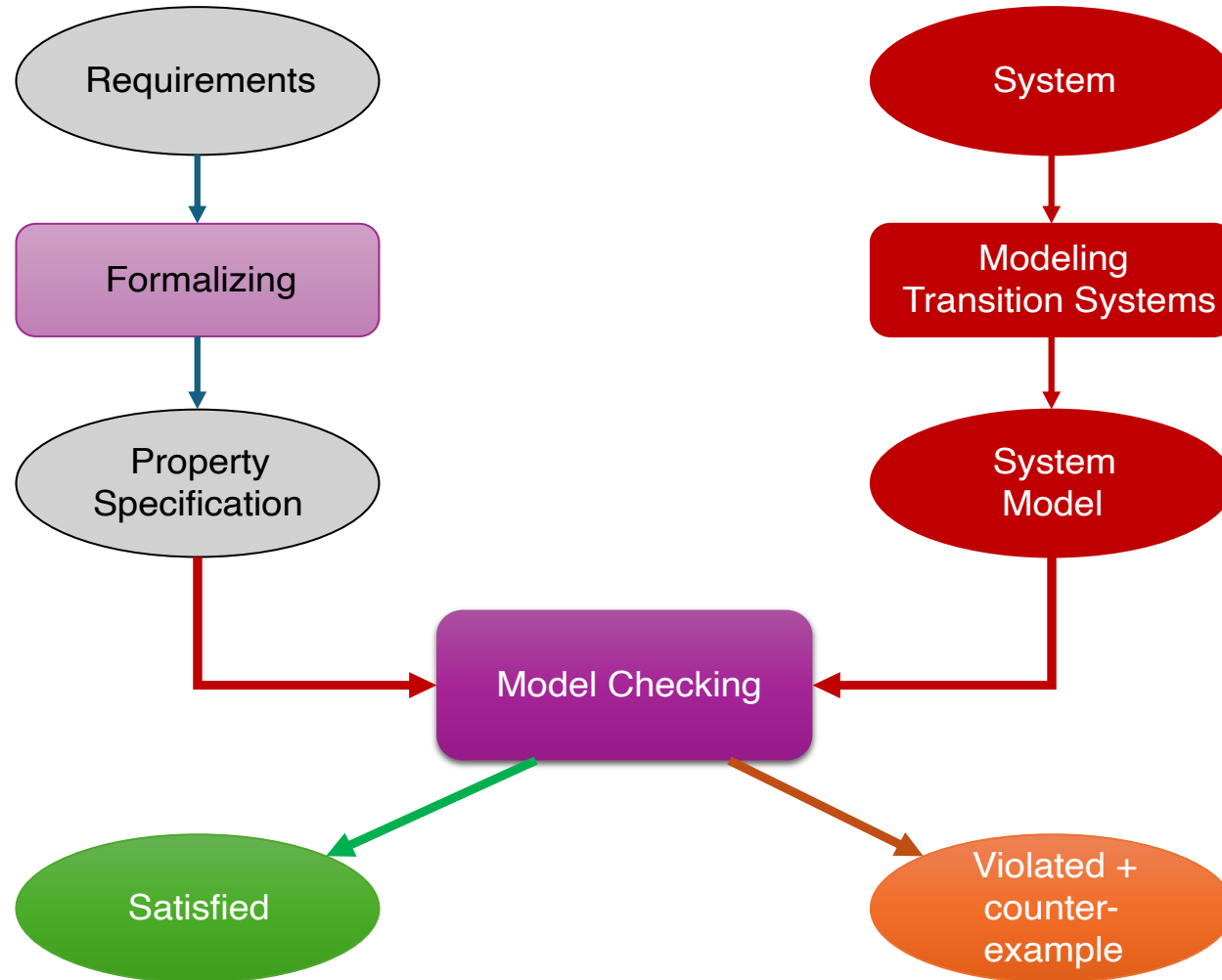
[idir.aitsadoue@centralesupelec.fr](mailto:idir.aitsadoue@centralesupelec.fr)

# PLAN

- La logique temporelle LTL
- Exemples de propriétés LTL
- Spécification de propriétés

[Retour au plan](#) - [Retour à l'accueil](#)

# PRINCIPE DU MODEL-CHECKING



# RAPPEL

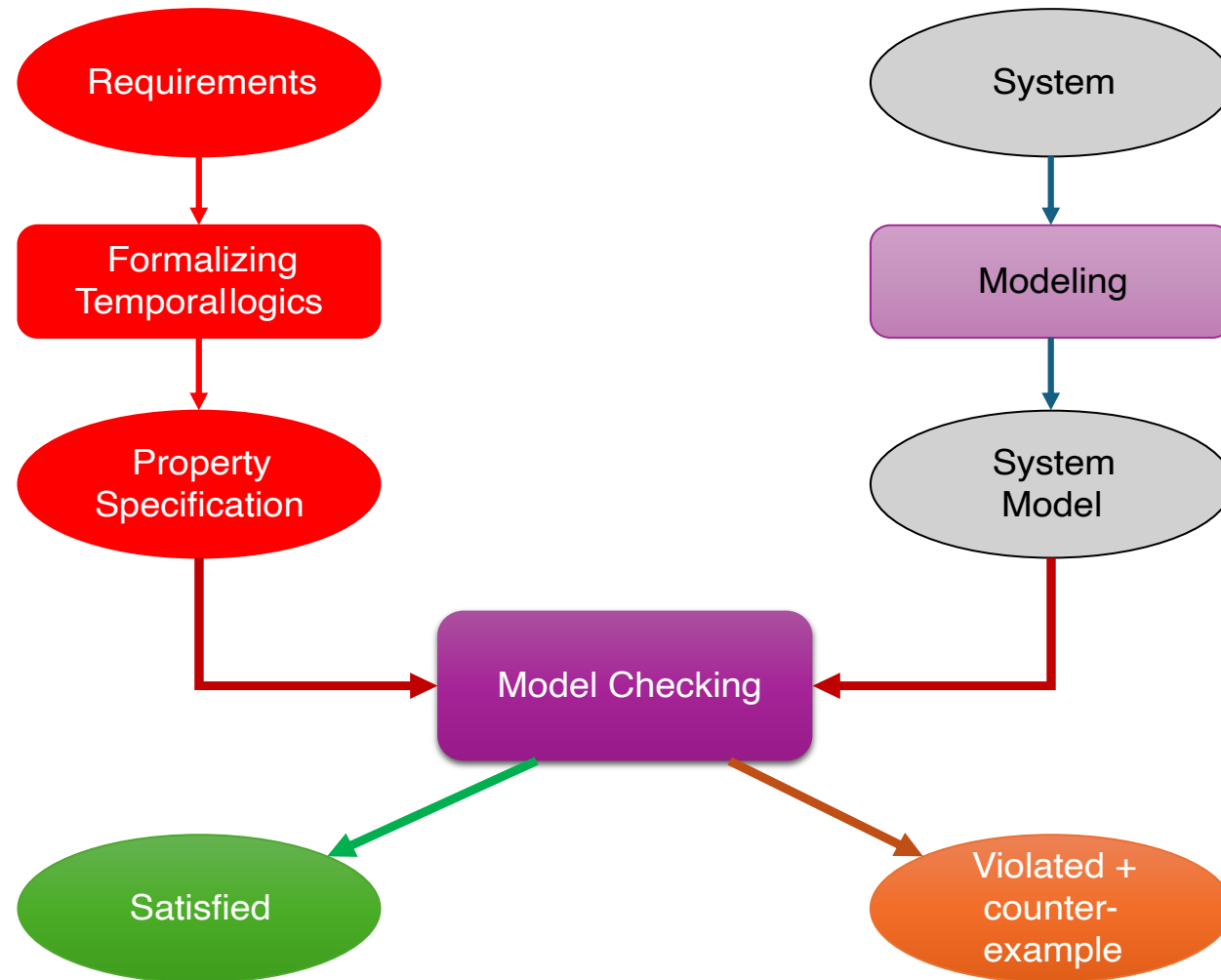
- Un **systèmes de transition**  $TS$  est un tuple  $(S, \delta, I, AP, \mathcal{L})$
- Un **fragment de chemin infini**  $\pi$  est une séquence d'états infinie :  
 $\pi = s_0 s_1 s_2 \dots$  tel que  $\forall i \geq 0, s_i \longrightarrow s_{i+1} \in \delta$
- La **trace du chemin**  $\pi = s_0 s_1 s_2 \dots \in S^\omega$  avec  $\mathcal{L} : S \longrightarrow 2^{AP}$ 
  - $trace(\pi) = \sigma = \mathcal{L}(s_0)\mathcal{L}(s_1)\mathcal{L}(s_2)\dots \in (2^{AP})^\omega$

# PLAN

- > La logique temporelle LTL
- > Exemples de propriétés LTL
- > Spécification de propriétés

[Retour au plan](#) - [Retour à l'accueil](#)

# PRINCIPE DU MODEL-CHECKING



# LOGIQUES TEMPORELLES

## POURQUOI ?

- Permettent d'exprimer des **propriétés** sur des **séquences d'observations**
- Utilisation de **connecteurs temporels** et de **quantificateurs** sur les **chemins**
- On pourrait utiliser **la logique du premier ordre**.

$\phi ::= true \mid a \mid \phi \wedge \phi \mid \neg \phi \mid \exists x. \phi \mid \dots$

- **Exemple** : "toute requête sera un jour satisfaite"
- $\forall t. (requete \rightarrow \exists t' \geq t. (reponse))$

- ✗ difficile à écrire/comprendre
- ✗ vérification peu efficace

# LOGIQUES TEMPORELLES

## POURQUOI ?

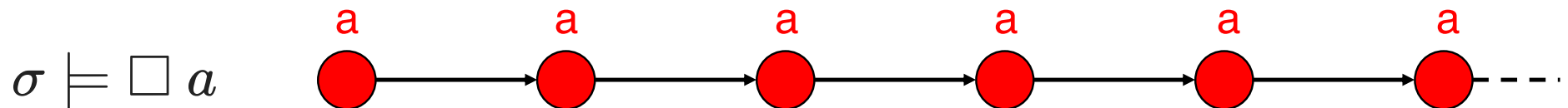
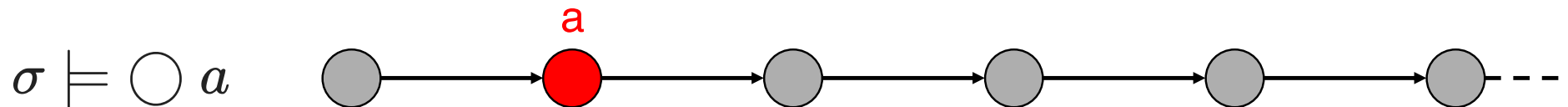
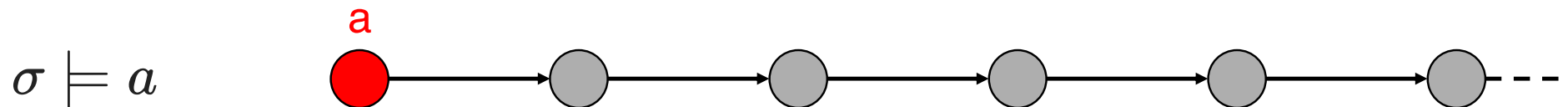
- Pas de variable pour gérer le temps (instants implicites)
- **Temporel**  $\neq$  **temporisé**  
la logiques temporelles ne quantifient pas écoulement du temps.
- Deux approches :
  1. **temps linéaire** : propriétés des séquences d'exécutions (futur déterminé)
  2. **temps arborescent** : propriétés de l'arbre d'exécutions (tous les futurs possibles)

# PROPOSITIONAL LINEAR TEMPORAL LOGIC (LTL)

$\phi ::= \text{true} \mid a \mid \phi \wedge \phi \mid \neg \phi \mid \bigcirc \phi \mid \Box \phi$

avec  $a \in AP$

$\bigcirc \equiv X$  (next)     $\Box \equiv G$  (always)



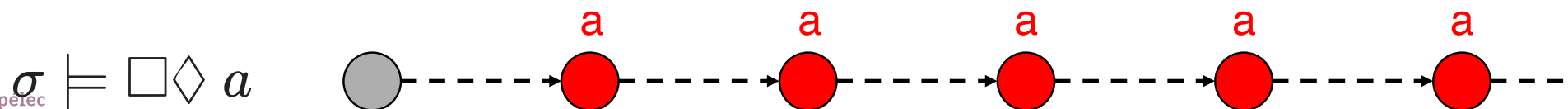
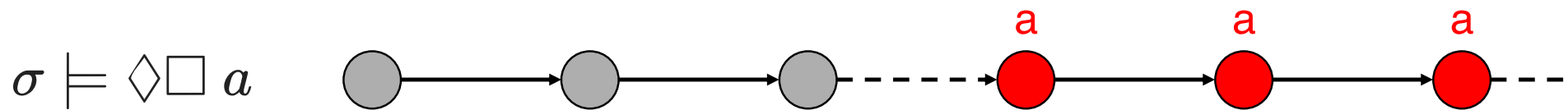
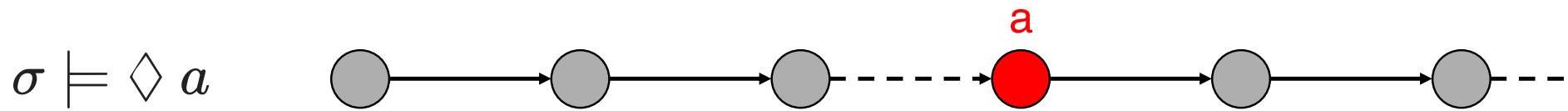
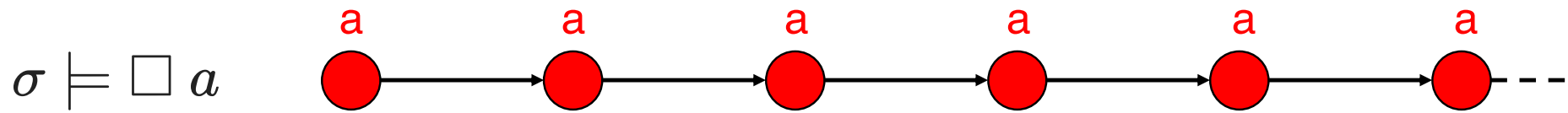
# OPÉRATEURS TEMPORELS DÉRIVÉS

$\Box \phi \equiv G \phi$   
(always)

$\Diamond \phi \equiv F \phi \equiv \neg \Box \neg \phi$   
(eventually)

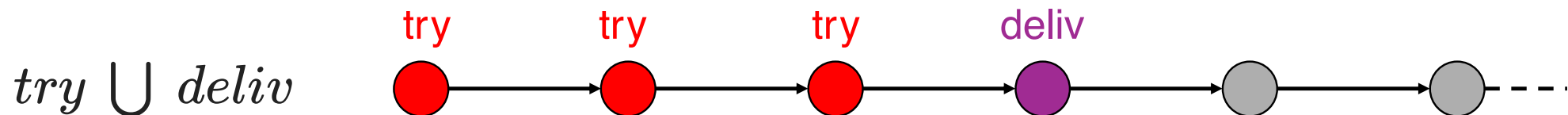
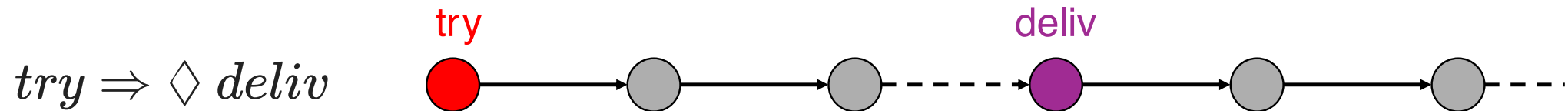
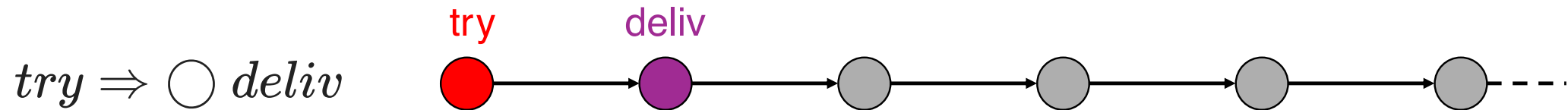
$\Diamond \Box \phi$   
(persistence)

$\Box \Diamond \phi \equiv \neg \Diamond \Box \neg \phi$   
(infinitely many)



# L'OPÉRATEUR UNTIL

$\phi ::= true \mid a \mid \phi \wedge \phi \mid \neg \phi \mid \bigcirc \phi \mid \square \phi \mid \phi \cup \phi$



$\Diamond \phi \equiv true \cup \phi$  et  $\square \phi \equiv \neg \Diamond \neg \phi$

# LA SÉMANTIQUE DES OPÉRATEURS LTL

- **Syntaxe version 1 :**

$$\phi ::= true \mid a \mid \phi_1 \wedge \phi_2 \mid \neg \phi \mid \bigcirc \phi \mid \square \phi \mid \diamond \phi \mid \phi_1 \cup \phi_2$$

- **Syntaxe version 2 : (adoptée dans la suite du cours/TD)**

$$\phi ::= true \mid a \mid \phi_1 \wedge \phi_2 \mid \neg \phi \mid X \phi \mid G \phi \mid F \phi \mid \phi_1 U \phi_2$$

- Pour  $\sigma = A_0 A_1 A_2 \cdots \in (2^{AP})^\omega$  :

$$\sigma \models true$$

$$\sigma \models a \quad \text{iff } a \in A_0$$

$$\sigma \models \phi_1 \wedge \phi_2 \quad \text{iff } \sigma \models \phi_1 \text{ and } \sigma \models \phi_2$$

$$\sigma \models \neg \phi \quad \text{iff } \sigma \not\models \phi$$

$$\sigma \models X \phi \quad \text{iff } A_1 A_2 A_3 \cdots \models \phi$$

$$\sigma \models G \phi \quad \text{iff } \forall i \geq 0, A_i A_{i+1} A_{i+2} \cdots \models \phi$$

$$\sigma \models F \phi \quad \text{iff } \exists i \geq 0, A_i A_{i+1} A_{i+2} \cdots \models \phi$$

$$\sigma \models \phi_1 U \phi_2 \quad \text{iff } \exists j \geq 0, A_j A_{j+1} \cdots \models \phi_2 \text{ et } \forall 0 \leq i < j, A_i A_{i+1} \cdots \models \phi_1$$

# PLAN

- La logique temporelle LTL
- Exemples de propriétés LTL
- Spécification de propriétés

[Retour au plan](#) - [Retour à l'accueil](#)

# PROPRIÉTÉ DU TEMPS LINÉAIRE

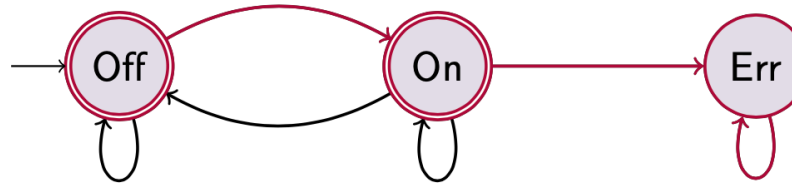
- Les propriétés du temps linéaire spécifient le comportement **admissible** du système considéré
  - La propriété LT spécifie les traces qu'un  $TS$  peut exhiber

## Définition formelle

- Une propriété temporelle linéaire  $P$  sur  $AP$  est un sous-ensemble de  $(2^{AP})^\omega$
- $TS$  **satisfait**  $P$  (sur  $AP$ ) :
  - $TS \models P$  si et seulement si  $Traces(TS) \subseteq P \subseteq (2^{AP})^\omega$

- Nous utiliserons la **logique temporelle linéaire (LTL)** pour formaliser  $P$

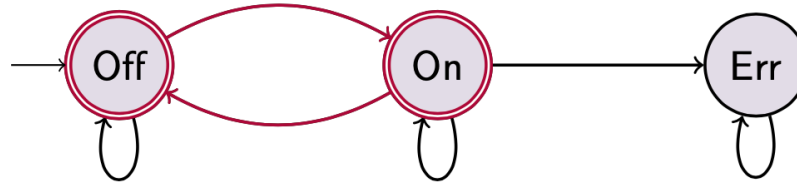
# EXEMPLE I



Prenant la trace  $\sigma = \text{Off On Err Err Err} \dots = \text{Off On Err}^\omega$

- $\sigma \models \text{Off}$  mais  $\sigma \not\models \text{On}$  alors  $\sigma \models \neg \text{On}$
- $\sigma \models \text{X On}$
- $\sigma \models \text{XX Err}$
- $\sigma \models (\text{Off} \vee \text{On}) \cup \text{Err}$
- $\sigma \models \text{G}(\text{Err} \Rightarrow \text{X Err})$
- $\sigma \models \text{G}(\text{Err} \Rightarrow \text{G Err})$
- $\sigma \models \text{FG Err}$
- $\sigma \models \text{XX G Err}$

# EXAMPLE II



Prenant la trace  $\sigma = \text{Off On Off On Off} \dots = (\text{Off On})^\omega$

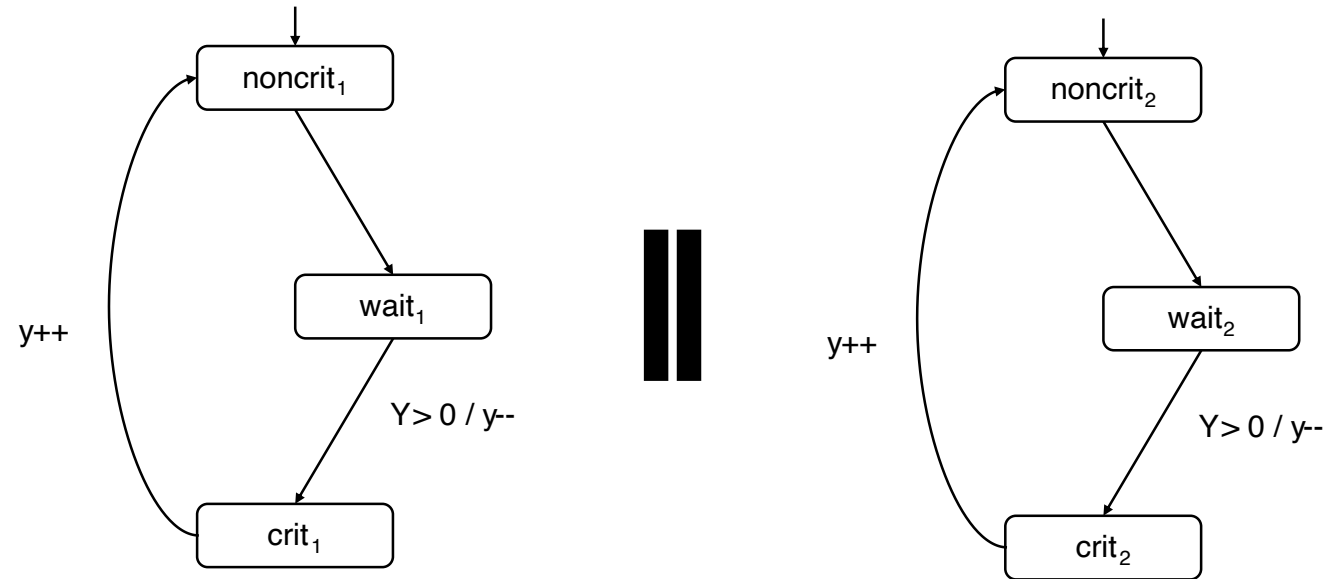
- $\sigma \not\models (\text{Off} \vee \text{On}) \cup \text{Err}$
- $\sigma \models \mathbf{F} \text{Err} \Rightarrow ((\text{Off} \vee \text{On}) \cup \text{Err})$  car  $\sigma \not\models \mathbf{F} \text{Err}$
- $\sigma \models \mathbf{G}(\text{On} \vee \text{Off})$
- $\sigma \models \mathbf{GF} \text{On} \wedge \mathbf{GF} \text{Off}$
- $\sigma \not\models \mathbf{FG} \text{On} \vee \mathbf{FG} \text{Off}$
- $\sigma \models \mathbf{G}(\text{Off} \Rightarrow \mathbf{X} \text{On}) \wedge \mathbf{G}(\text{On} \Rightarrow \mathbf{X} \text{Off})$

# PLAN

- La logique temporelle LTL
- Exemples de propriétés LTL
- **Spécification de propriétés**

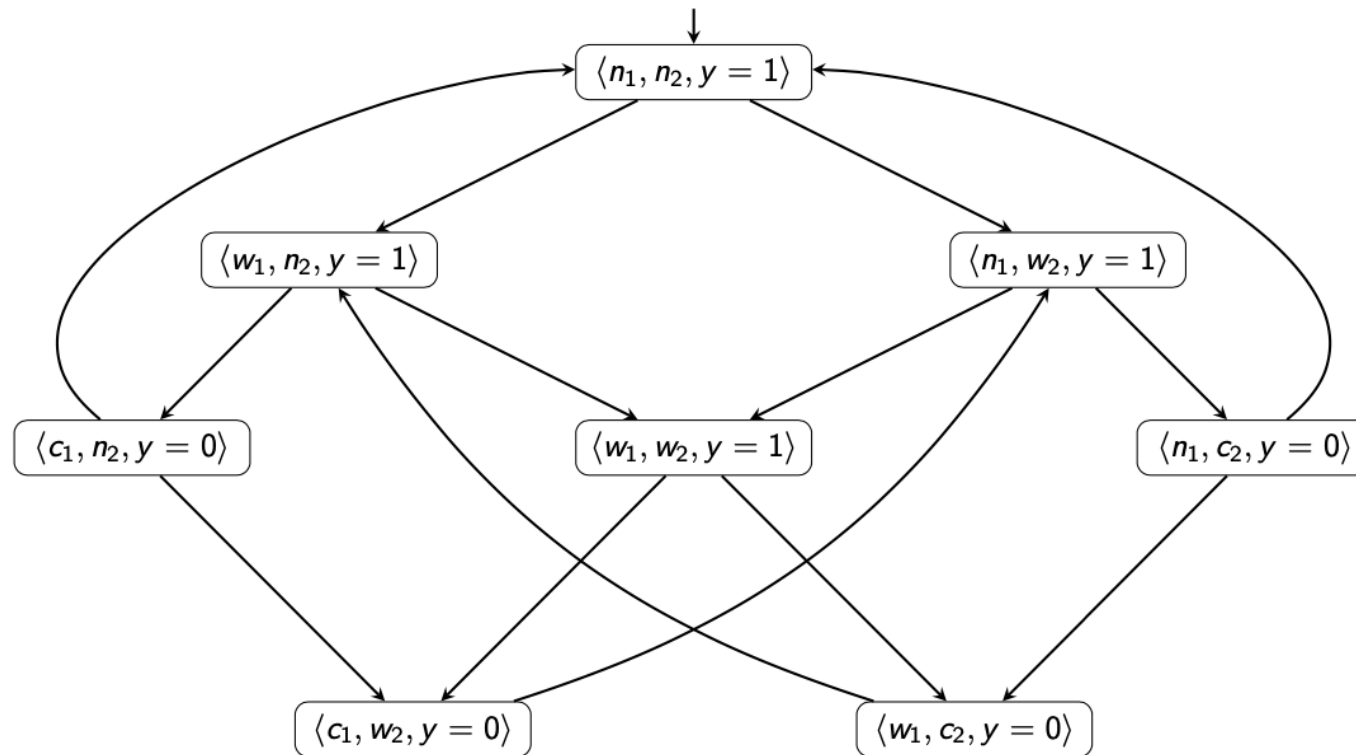
[Retour au plan](#) - [Retour à l'accueil](#)

# RAPPEL DE L'EXEMPLE



$y = 0$  signifie "le verrou est actuellement possédé";  
 $y = 1$  signifie "le verrou est libre"

# RAPPEL DE L'EXEMPLE



$n_i : noncrit_i, \quad w_i : wait_i, \quad c_i : crit_i$

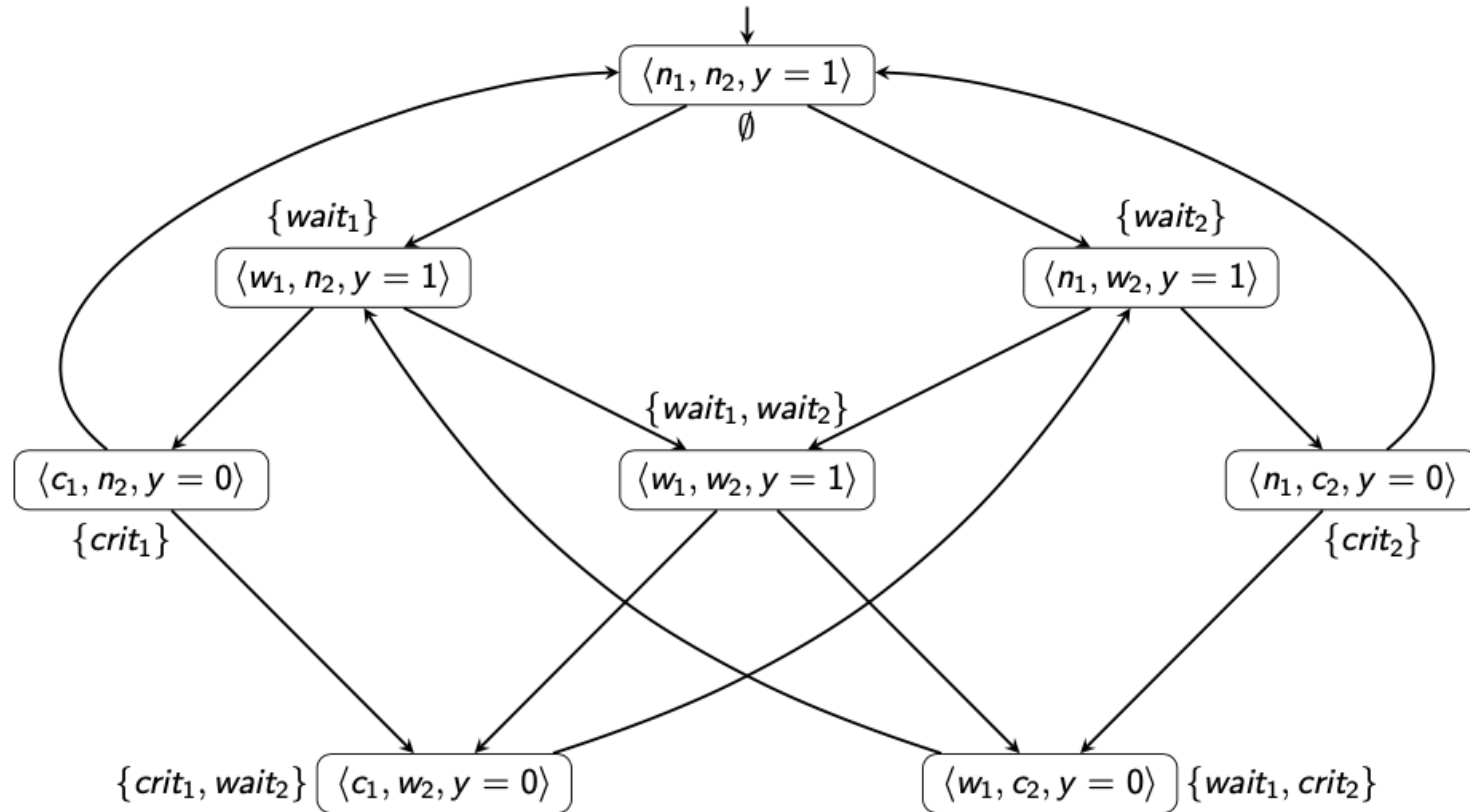
# COMMENT SPÉCIFIER L'EXCLUSION MUTUELLE ?

## L'exclusion mutuelle

Il y a au plus un processus dans la section critique

- Soit  $AP = \{crit_1, crit_2\}$ 
  - les autres propositions atomiques n'ont aucune pertinence pour cette propriété
- Formalisation LTL de la propriété LT
$$P_{mutex} = G \neg (crit_1 \wedge crit_2)$$
- L'algorithme basé sur le sémaphore satisfait-il  $P_{mutex}$  ?

# LA RÉPONSE



**OUI !** car il n'existe aucun état accessible étiqueté avec  $\{crit_1, crit_2\}$

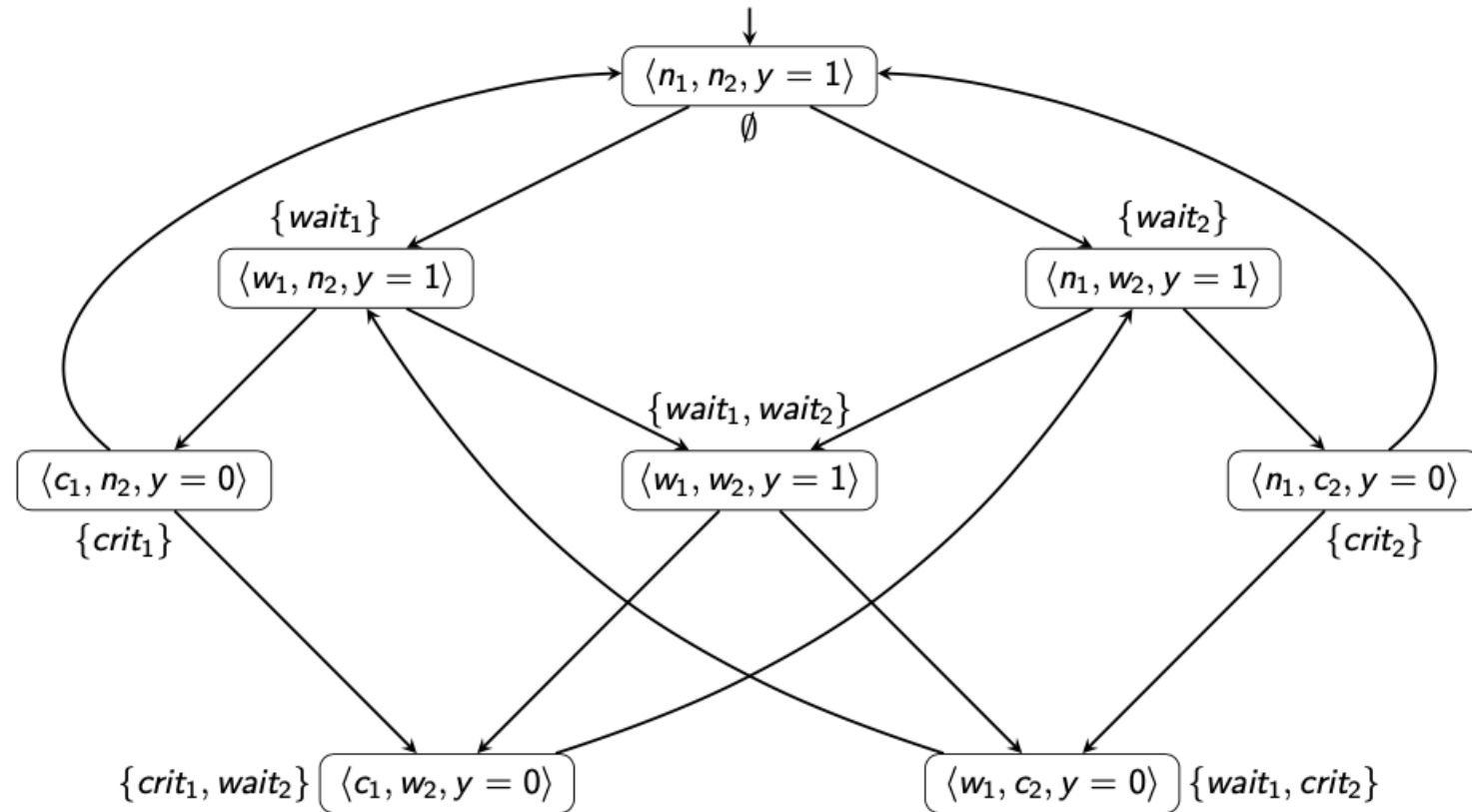
# COMMENT SPÉCIFIER L'ABSENCE DE FAMINE ?

## L'absence de famine

Un processus qui veut entrer dans la section critique est finalement capable de le faire

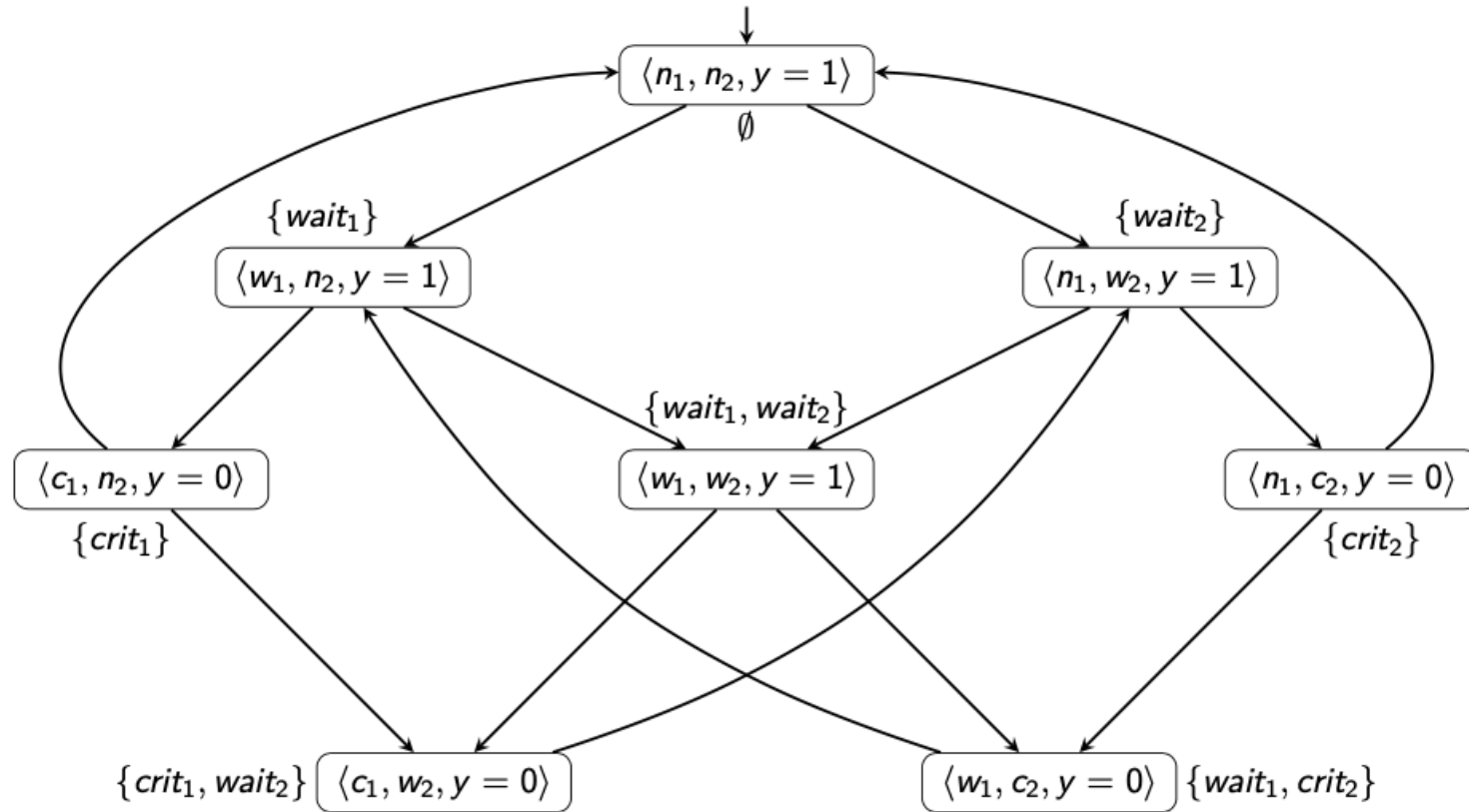
- Soit  $AP = \{wait_1, crit_1, wait_2, crit_2\}$
- Formalisation LTL de la propriété LT
$$P_{nostarve} = G (wait_1 \Rightarrow F crit_1) \wedge G (wait_2 \Rightarrow F crit_2)$$
- L'algorithme basé sur le sémaphore satisfait-il  $P_{nostarve}$  ?

# LA RÉPONSE



**NON !** Le processus un ou le processus deux risquent de mourir de faim !

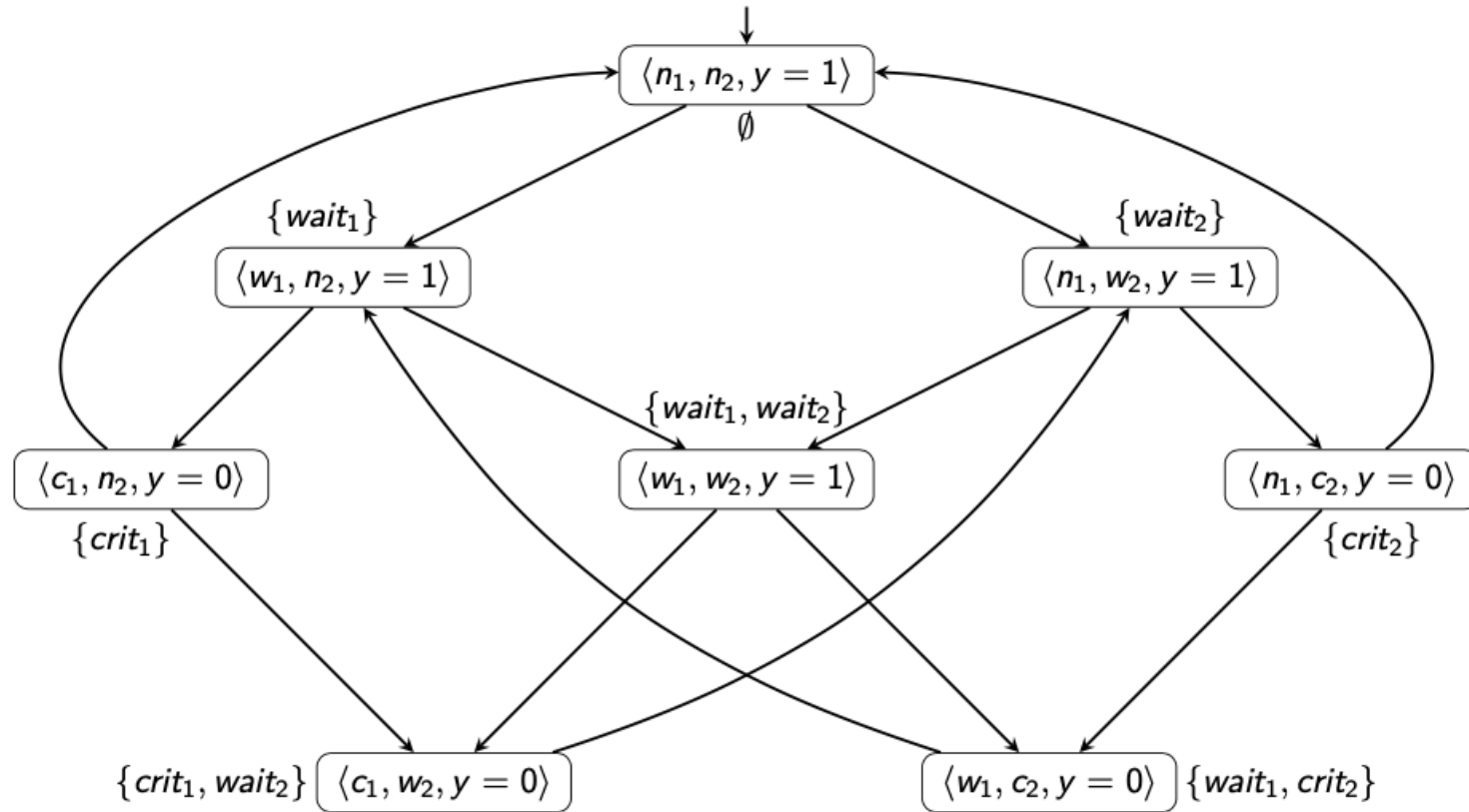
# LA RÉPONSE



$$\sigma = \emptyset(\{wait_1\}\{wait_1, wait_2\}\{wait_1, crit_2\})^\omega \in Traces(TS)$$

$$\sigma \models F(wait_1 \wedge G \neg crit_1) \Rightarrow \sigma \notin P_{nostarve}$$

# LA RÉPONSE



$$\sigma = \emptyset(\{wait_2\}\{wait_1, wait_2\}\{crit_1, wait_2\})^\omega \in Traces(TS)$$

$$\sigma \models F(wait_2 \wedge G\neg crit_2) \Rightarrow \sigma \notin P_{nostarve}$$

# LA PROPRIÉTÉ D'INVARIANTS

- **Propriété de sécurité** typique  $\rightarrow$  propriété d'exclusion mutuelle
  - la **mauvaise chose** (avoir  $> 1$  processus dans la section critique) **ne se produit jamais**
- Une autre **propriété de sécurité** typique  $\rightarrow$  vérifie les limites des variables (dépassement)

Ces propriétés sont des **invariants**

- Un **invariant** est une propriété LT
  - qui est donné par une **condition**  $\phi$  sur  $AP$
  - exige que la **condition**  $\phi$  soit vraie **pour tous les états** (atteignables)
  - **exemple** : la propriété d'exclusion mutuelle  $\phi = \neg(crit_1 \wedge crit_2)$

# LA PROPRIÉTÉ D'INVARIANTS

## DÉFINITION FORMELLE

- Une propriété LT  $P_{inv}$  sur  $AP$  est un **invariant** s'il existe une formule **pure propositionnelle**  $\phi$  sur  $AP$  telle que :

$$P_{inv} = G \phi$$

- $\phi$  est appelé une **condition invariante** de  $P_{inv}$
- Notez que :  
 $TS \models P_{inv}$  si et seulement si  $\forall s \in Reach(TS), \mathcal{L}(s) \models_{prop} \phi$
- $\phi$  doit être satisfait par tous les états initiaux et tous les états atteignables de  $TS$

# PROPRIÉTÉS DE SÉCURITÉ

- **Propriétés de sécurité (safety property)** → "rien de mauvais ne devrait arriver"
  - une **propriété invariante** est une **propriété de sécurité particulière**
- **Les propriétés de sécurité** peuvent imposer des exigences sur **des fragments de chemin finis**
- **Une propriété de sécurité** qui n'est pas un invariant
  - considérant un distributeur de billets
  - propriété "l'argent ne peut être retiré qu'une fois qu'un code PIN correct a été fourni"
  - **pas un invariant**, car ce n'est pas une propriété à vérifier dans tous les états
- Un exemple LTL typique → **Réponse limitée**

$$G(request \Rightarrow \bigvee_{i=n}^m X^i response)$$

# PROPRIÉTÉS DE VIVACITÉ

- Les **propriétés de sécurité** précisent que "quelque chose de mauvais n'arrive jamais"
- Ne rien faire satisfait facilement une propriété de sécurité
  - car cela ne mènera jamais à une "mauvaise" situation
- Les propriétés de sécurité sont complétées par des **propriétés de vivacité** (**Liveness properties**)
  - qui nécessitent des progressions dans l'exécution
  - qui affirment  $\rightarrow$  "quelque chose de bien arrivera éventuellement"
- Un exemple LTL typique  $\rightarrow F \phi$

# PROPRIÉTÉS DE VIVACITÉ

## EXEMPLES

- Revenons à notre algorithme basé sur les sémaphores avec

$$AP = \{wait_1, crit_1, wait_2, crit_2\}$$

- **Finale**ment (**Eventually**) :

$$F \text{ crit}_1 \wedge F \text{ crit}_2$$

- **Finale**ment répété (**Repeated eventually**) :

$$GF \text{ crit}_1 \wedge GF \text{ crit}_2$$

- **Absence de famine** :

$$G (wait_1 \Rightarrow F \text{ crit}_1) \wedge G (wait_2 \Rightarrow F \text{ crit}_2)$$

# MERCI

[PDF version of the slides](#)

[Retour à l'accueil](#) - [Retour au plan](#)