# Logiques temporelles pour la spécification

UE « Modélisation et spécification » — 2017-2018



Modélisation -> Arnaud Sangnier
Spécification -> François Laroussinie

Page web du cours:
https://www.irif.fr/~francoisl/m2modspec.html



## Spécifier un système

système = programme, un algorithme, un protocole, …

spécifier =
- énoncer la « correction » du système
- énoncer les propriétés attendues pour le système.
- énoncer l'absence de bug



## Spécifier un système
### exemple

premier essai

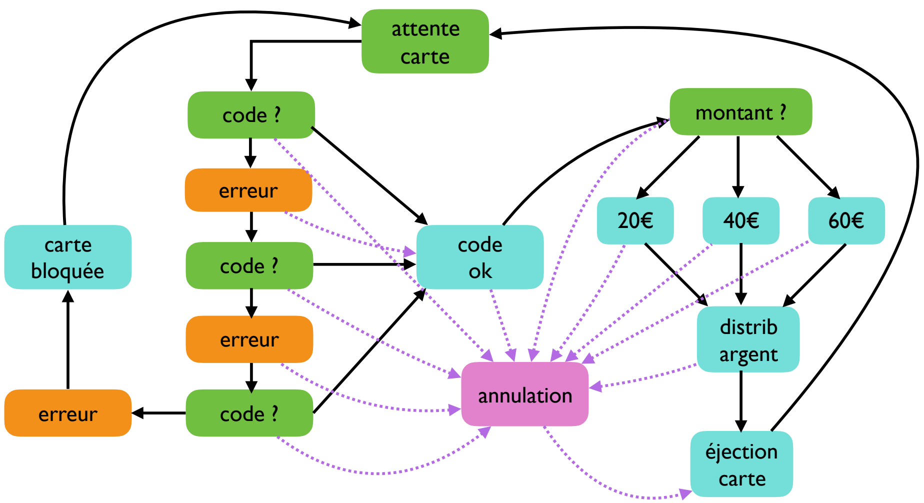
Un distributeur de billets.

Un distributeur de billets « correct » ?

- Si le bon code est donné, on peut choisir une somme et obtenir de l'argent.
- Si on fait trois erreurs de code, la carte est bloquée.
- Après avoir obtenu l'argent, la carte est éjectée.
- A tout moment, si on appuie sur annulation, la carte est éjectée sans donner d'argent.
- Si on demande un ticket, un ticket est donné.
- On éjecte la carte avant de donner l'argent
- …

# Spécifier un système
## exemple

Un distributeur de billets.






# Spécifier un distributeur de billets

premier essai

- Si le bon code est donné, on peut choisir une somme et obtenir de l'argent.
- Si on fait trois erreurs de code, la carte est bloquée.
- Après avoir obtenu l'argent, la carte est éjectée.
- A tout moment, si on appuie sur annulation, la carte est éjectée sans donner d'argent. sauf si 3 erreurs ont été commises… par le même utilisateur !
- Si on demande un ticket, un ticket est donné.
- …



Voir les cours d'Arnaud Sangnier pour la construction de modèles.

C'est une étape cruciale !
Un modèle faux ne sert pas à grand chose…

Ici on supposera toujours que l'on dispose d'un « STE »: un système de transitions étiquetés.

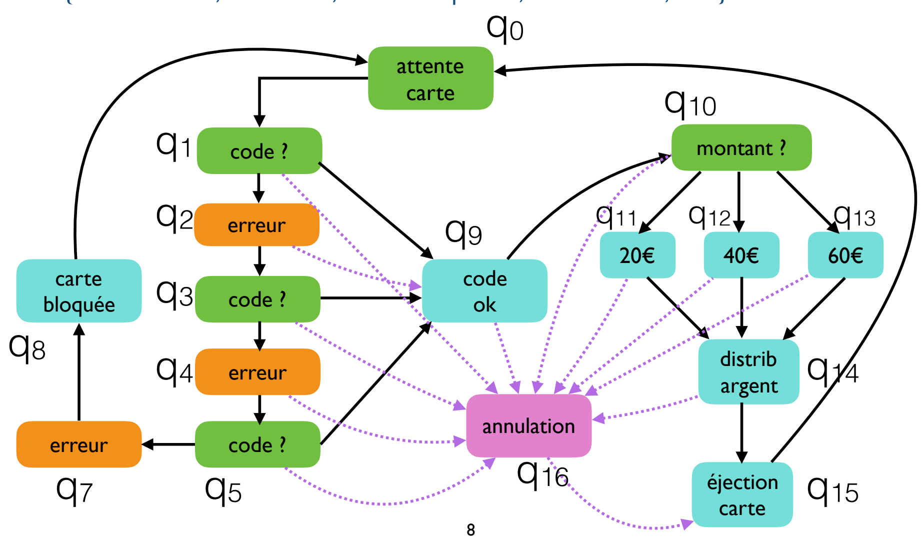
$$\mathbf{S} = (Q, Act, \rightarrow, q_0, AP, L)$$



# Système de transitions étiquetés

$\mathbf{S} = (Q, Act, \rightarrow, q_0, AP, L)$

AP={att. carte, code?, c. bloquée, code ok, …}



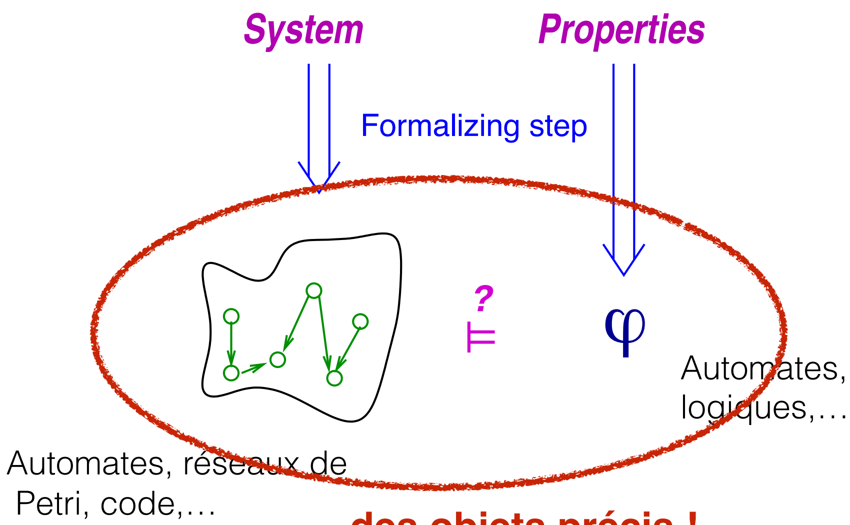
NB: on ne construit pas un STE « à la main » !

On utilise des langages de haut niveau pour décrire le comportement du système étudié (produit synchrone, réseau de Pétri, programmes, etc.).
Et ces langages ont une sémantique définie sous la forme d'un STE.

Même idée pour la spécification:
On utilise des langages de spécification avec une sémantique précise !



# Model-checking




**des objets précis !**



*Exemple* algorithme d'exclusion mutuelle

```
boolean D1:= False
boolean D2 := False
```

| Processus P1: | Processus P2: |
|---|---|
| loop forever: | loop forever: |
| p1: Section NC | p1: Section NC |
| p2: D1 := True | p2: D2 := True |
| p3: await (not D2) | p3: await (not D1) |
| p4: section critique | p4: section critique |
| p5: D1 := False | p5: D2 := False |

+ hypothèse d'atomicité

→ un STE (voir la construction précise avec Arnaud Sangnier !)



*Exemple* algorithme d'exclusion mutuelle

→ un STE avec AP={$D_1$,$D_2$,$SC_1$,$SC_2$}

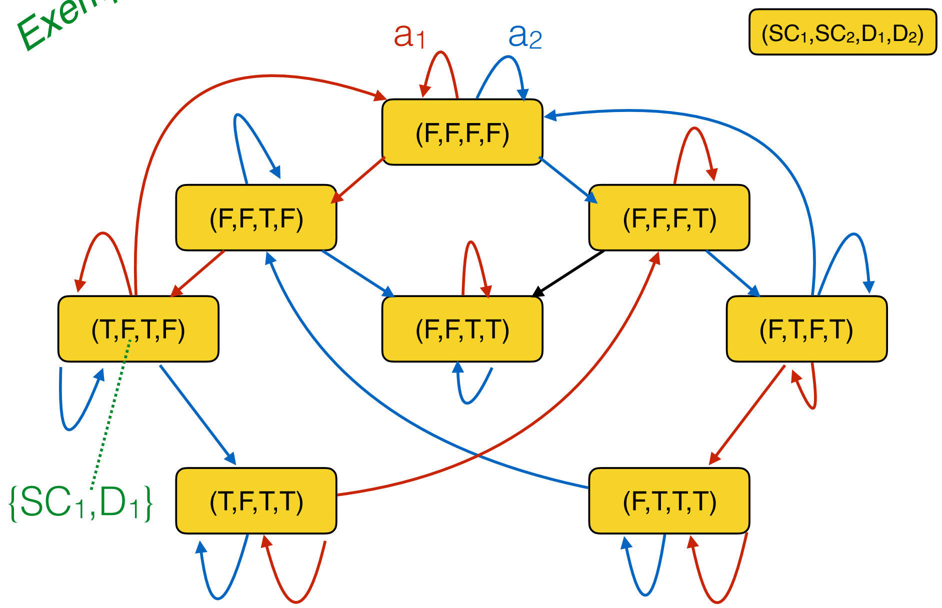
- $D_1$ étiquette les états où la variable D1 est vraie,
- $D_2$ étiquette les états où la variable D2 est vraie,
- $SC_1$ étiquette les états où le processus 1 est en section critique,
- $SC_2$ étiquette les états où processus 2 est en section critique.

Act = { $a_1$, $a_2$ }

Et Q contient 8 états.

*Exemple*

## algorithme d'exclusion mutuelle






## algorithme d'exclusion mutuelle

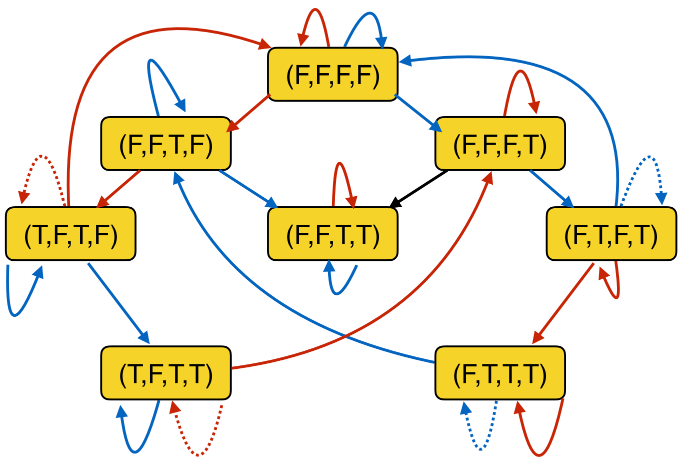
### propriétés attendues

- Exclusion mutuelle: Jamais les deux processus ne peuvent se trouver en SC au même moment.
- Il n'y a jamais pas de blocage.
- Absence de famine: Si un processus demande l'accès à la SC, il y arrivera un jour.
- Attente bornée: Si un processus demande l'accès à la SC, l'autre processus ne peut pas passer avant lui plus d'une fois.

En général, on vérifie ces propriétés sous hypothèses d'équité entre processus et en supposant que chaque section critique se termine.



## algorithme d'exclusion mutuelle

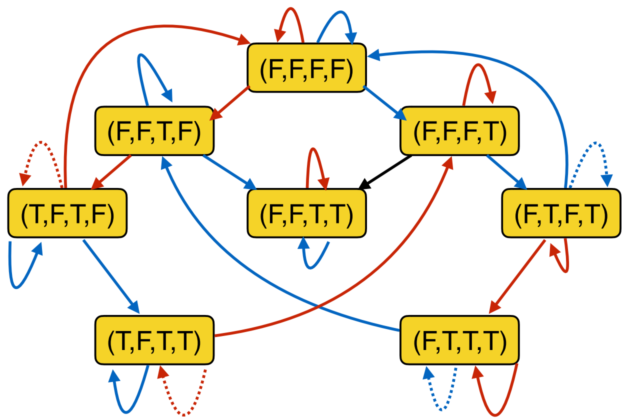



Equité = infinité de transitions bleues *et* de rouges.

SC termine = infinité de transitions bleues *et* de rouges **pleines**.



## algorithme d'exclusion mutuelle




+ équité
+ SC finies

→ un ensemble d'exécutions possibles du système modélisé.

Vérifient-elles les propriétés attendues ?

## algorithme d'exclusion mutuelle

propriétés attendues

- Exclusion mutuelle: Jamais les deux processus ne peuvent se trouver en SC au même moment. ✔
- Il n'y a jamais pas de blocage. ?
- Absence de famine: Si un processus demande l'accès à la SC, il y arrivera un jour. ✘
- Attente bornée: Si un processus demande l'accès à la SC, l'autre processus ne peut pas passer avant lui plus d'une fois. ?

(→ l'algorithme n'est pas bon !)



## Comment énoncer précisément ce genre de propriétés ?

- Exclusion mutuelle: Jamais les deux processus ne peuvent se trouver en SC au même moment.
- Il n'y a jamais pas de blocage.
- Absence de famine: Si un processus demande l'accès à la SC, il y arrivera un jour.
- Attente bornée: Si un processus demande l'accès à la SC, l'autre processus ne peut pas passer avant lui plus d'une fois.



## Spécifier un distributeur de billets

- Si le bon code est donné, on peut choisir une somme et on obtiendra de l'argent.
- Après avoir fait trois erreurs de code, la carte est bloquée.
- Après avoir obtenu l'argent, la carte est éjectée.
- A tout moment, si on appuie sur annulation, la carte est éjectée sans donner d'argent sauf si l'utilisateur a fait trois erreurs avant.



## Spécifier un **système réactif**

système réactif = système qui interagit avec un environnement.

- Il ne calcule pas un résultat en un temps fini.
- Il maintient une interaction avec son environnement.
- Sa correction se base sur l'ordre des actions/évènements tout au long de son exécution.

avant/après/jusqu'à/depuis

+

si alors ($\Rightarrow$), et ($\wedge$), ou ($\vee$)

→ la logique temporelle

## Logique propositionnelle

**Syntaxe:** $P \in AP$

$\phi, \psi \ ::= \ P \mid \neg \phi \mid \phi \vee \psi \mid \phi \wedge \psi$

carte_éjectée $\wedge$ argent_distribué

erreur $\vee$ annulation

$\neg ( CS_1 \wedge CS_2)$

$\neg CS_1 \vee \neg CS_2$ $\qquad (\neg CS_1) \vee (\neg CS_2)$

$D_1 \wedge \neg D_2$

L'évaluation d'une formule dépend de la valeur de vérité de chaque proposition atomique.



## Logiques temporelles

Les logiques temporelles étendent la logique propositionnelle avec:

- des « modalités temporelles » (ou des « opérateurs temporels »)
- sont interprétées dans des modèles munis d'une notion de temps.

Il y a de nombreuses logiques temporelles ! (et encore plus que ça !)

- LTL « Linear-time Temporal Logic »
- CTL « Computation Tree Logic »



## LTL « Linear-time Temporal Logic »

Logique de temps linéaire.
Le comportement d'un système est vu comme l'ensemble de ses exécutions prises séparément.

La notion de temps (avant/après/jusqu'à/depuis) s'interprète donc le long d'une exécution d'un STE.



## LTL- (premier fragment)

**Syntaxe:** $P \in AP$

$\phi, \psi ::= P \mid \neg\phi \mid \phi\vee\psi \mid \phi\wedge\psi \mid \mathbf{X}\,\phi \mid \mathbf{F}\,\phi \mid \mathbf{X}^{-1}\,\phi \mid \mathbf{F}^{-1}\,\phi$

$\mathbf{X}\,\phi$ : demain $\phi$

$\mathbf{F}\,\phi$ : un jour dans le futur $\phi$

$\mathbf{X}^{-1}\,\phi$ : hier $\phi$

$\mathbf{F}^{-1}\,\phi$ : un jour dans le passé $\phi$

## STE

$\mathbf{S} = (Q, Act, \rightarrow, q_0, AP, L)$

NB: ici on n'utilisera pas les actions sur les transitions (Act).

NB: ici supposera que tout état de Q a au moins un successeur par $\rightarrow$.

**Exécutions:**
séquence infinie $\rho = s_0\, s_1\, s_2\, s_3 \ldots$ telle que:

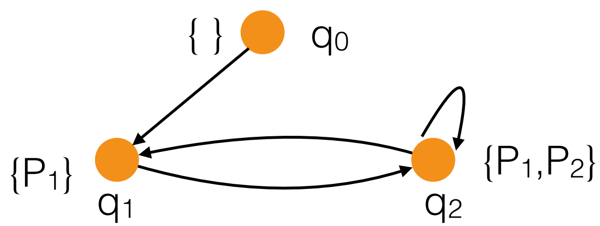
- $s_i \in Q \quad \forall\, i = 0,1,2,\ldots$
- $(s_i,s_{i+1}) \in \rightarrow$ (ou $s_i \rightarrow s_{i+1}$)

Notation: $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots \rightarrow s_k \rightarrow$
et $\rho(i) = s_i$ , $\rho^i = s_i\, s_{i+1}\, s_{i+2} \ldots$ $\rho|_i = s_0 s_1 \ldots s_i$

Exec(q) = ensemble des exécutions issues de q.



## Exemple




$Exec(q_1) = \{\ (q_1q_2q_1q_2\ldots,\ldots,\ q_1q_2q_2q_2\ldots,\ \ldots\}$
ie $Exec(q_1) = (q_1q_2^+)^+q_2^\omega \ \cup\ (q_1q_2^+)^\omega$



## LTL- (premier fragment)

**Syntaxe:** $P \in AP$
$\phi,\psi ::= P \mid \neg\phi \mid \phi\vee\psi \mid \phi\wedge\psi \mid \mathbf{X}\,\phi \mid \mathbf{F}\,\phi \mid \mathbf{X}^{-1}\,\phi \mid \mathbf{F}^{-1}\,\phi$

$\mathbf{X}\,\phi$ : demain $\phi$
$\mathbf{F}\,\phi$ : un jour dans le futur $\phi$
$\mathbf{X}^{-1}\,\phi$ : hier $\phi$
$\mathbf{F}^{-1}\,\phi$ : un jour dans le passé $\phi$

$\rightarrow$ on interprète les formule de LTL- sur une position i le long d'une exécution ρ d'un STE.

+L pour les val. des AP.

$\rho = s_0\, s_1\, s_2\, s_3 \ldots s_i\, s_{i+1} \ldots$



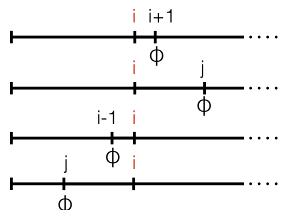

## LTL- (premier fragment)

**Syntaxe:** $P \in AP$
$\phi,\psi ::= P \mid \neg\phi \mid \phi\vee\psi \mid \phi\wedge\psi \mid \mathbf{X}\,\phi \mid \mathbf{F}\,\phi \mid \mathbf{X}^{-1}\,\phi \mid \mathbf{F}^{-1}\,\phi$

**Sémantique:**
soit ρ une exécution d'un STE $\mathbf{S} = (Q, Act, \rightarrow, q_0, AP, L)$
soit i un entier $\geq 0$

$\rho,i \models P$ ssi $P \in L(\rho(i))$
$\rho,i \models \neg\phi$ ssi $\rho,i \not\models \phi$
$\rho,i \models \phi\wedge\psi$ ssi ( $\rho,i \models \phi$ et $\rho,i \models \psi$)
$\rho,i \models \phi\vee\psi$ ssi ( $\rho,i \models \phi$ ou $\rho,i \models \psi$)

$\rho,i \models \mathbf{X}\,\phi$ ssi $\rho,i{+}1 \models \phi$
$\rho,i \models \mathbf{F}\,\phi$ ssi ( $\exists\, j \geq i.\ \ \rho,j \models \phi$ )
$\rho,i \models \mathbf{X}^{-1}\,\phi$ ssi (i>0 et $\rho,i{-}1 \models \phi$ )
$\rho,i \models \mathbf{F}^{-1}\,\phi$ ssi ( $\exists\, j \leq i.\ \ \rho,j \models \phi$)



# LTL-

$\mathbf{S} = (Q, Act, \rightarrow, q_0, AP, L)$

$\rho, i \models \phi$ est désormais défini !

Et $\mathbf{S} \models \phi$ ?

Rappel:
Avec les logiques de temps linéaire, le comportement d'un système est vu comme l'ensemble de ses exécutions prises séparément.

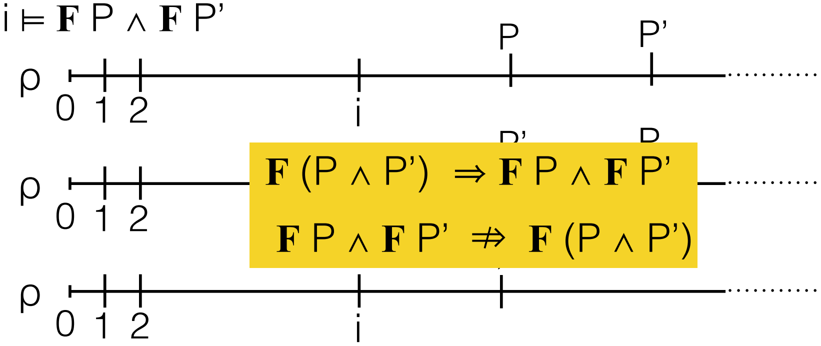
$\mathbf{S} \models \phi$ si et seulement si $\rho, 0 \models \phi \quad \forall \rho \in Exec(q_0)$



# Exemples de formules

Comparer $\mathbf{F}\, P \wedge \mathbf{F}\, P'$ et $\mathbf{F}\,(P \wedge P')$

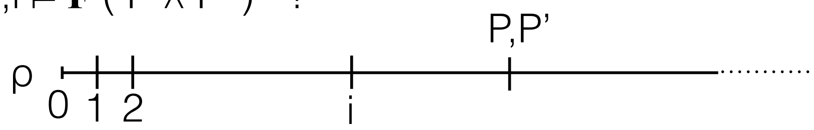
$\rho, i \models \mathbf{F}\, P \wedge \mathbf{F}\, P'$




$\mathbf{F}\,(P \wedge P') \Rightarrow \mathbf{F}\, P \wedge \mathbf{F}\, P'$

$\mathbf{F}\, P \wedge \mathbf{F}\, P' \not\Rightarrow \mathbf{F}\,(P \wedge P')$

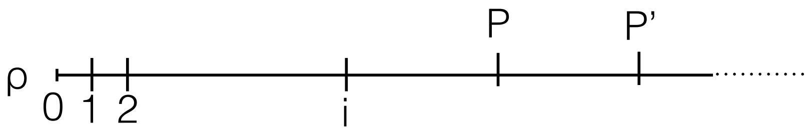
$\rho, i \models \mathbf{F}\,(P \wedge P')$ ?

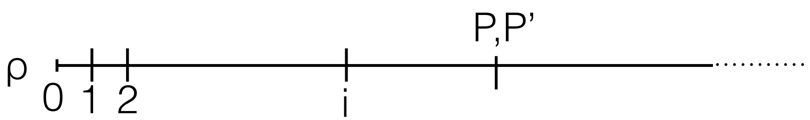




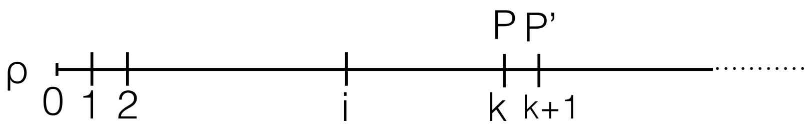

# Exemples de formules

$\rho, i \models \mathbf{F}\,(P \wedge \mathbf{F}\, P')$ ?



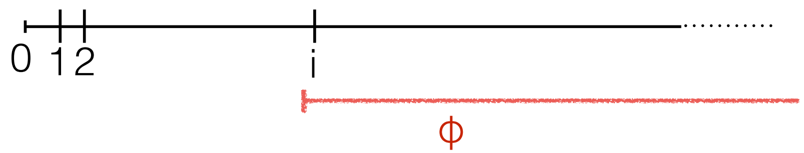

$\rho, i \models \mathbf{F}\,(P \wedge \mathbf{X}\, P')$ ?






# Exemples de formules

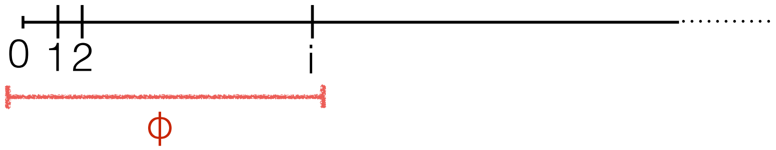
$\rho, i \models \neg\, \mathbf{F} \neg\phi$ ??




→ $\phi$ est vrai pour tous les états i, i+1, i+2,…
« toujours dans le futur »
On le note: $\mathbf{G}\, \phi = \neg\, \mathbf{F} \neg\phi$

## Exemples de formules

$\rho,i \models \neg \mathbf{F}^{-1} \neg \phi$ ??




→ $\phi$ est vrai pour tous les états du passé: i, i-1, i-2…
« toujours dans le passé »
On le note: $\mathbf{G}^{-1}\,\phi = \neg\,\mathbf{F}^{-1}\,\neg\phi$



## Exemples de formules

$\rho,i \models \neg\,\mathbf{X}\,\phi$

$\Leftrightarrow\ \rho,i \models \mathbf{X}\,\neg\,\phi$ (car exécutions infinies)

( sans cette hypothèse, on aurait:
$\rho,i \models \mathbf{X}\,\neg\,\phi \quad \Rightarrow \quad \rho,i \models \neg\,\mathbf{X}\,\phi$ )