Laboratoire Méthodes Formelles

CentraleSupélec | université PARIS-SACLAY

DÉVELOPPEMENT DE SYSTÈMES CRITIQUES AVEC LA MÉTHODE EVENT-B

LA VALIDATION D'UN MODÈLE EVENT-B AVEC PROB

🎓 3A cursus ingénieurs - Mention Sciences du Logiciel
🏛 CentraleSupelec - Université Paris-Saclay - 2025/2026

Idir AIT SADOUNE
idir.aitsadoune@centralesupelec.fr

PLAN

Retour au plan - Retour à l'accueil

CentraleSupélec
université PARIS-SACLAY

PLAN

Retour au plan - Retour à l'accueil

CentraleSupélec
université PARIS-SACLAY

THE PROOF WITH ATELIER-B

- There are two main proof activities in the **Event-B** method:
  1. the proof of consistency $\longrightarrow$ the events of a machine preserve the invariant
  2. the proof of refinement $\longrightarrow$ one machine is a valid refinement of another

- In the **Rodin platform**, proof activities are supported by tools, such as the **Atelier-B plugin**
  - the **Rodin platform** generates the list of proof obligations (**PO**)
  - the **Atelier-B plugin** is an automatic prover

- In some cases, the most complex **POs** are not proved automatically and *must be proved interactively*.

CentraleSupélec
université PARIS-SACLAY

HISTORY OF FORMAL VERIFICATION METHODS

Before . . .

- Software code was sequential
- Properties were expressed in **First-Order Predicate Logic**
- **Theorem provers** $\longrightarrow$ partial/total correctness
- Hardly automated $\longrightarrow$ **semi-decidable** (e.g. B or Event-B methods)

After 80's

- Software is **concurrent** and reactive
- Properties are expressed in **Temporal Logic**
- Solving accurate properties like safety, liveness, fairness . . .
- Push-Button $\longrightarrow$ **decidable** (e.g. Model Checking)

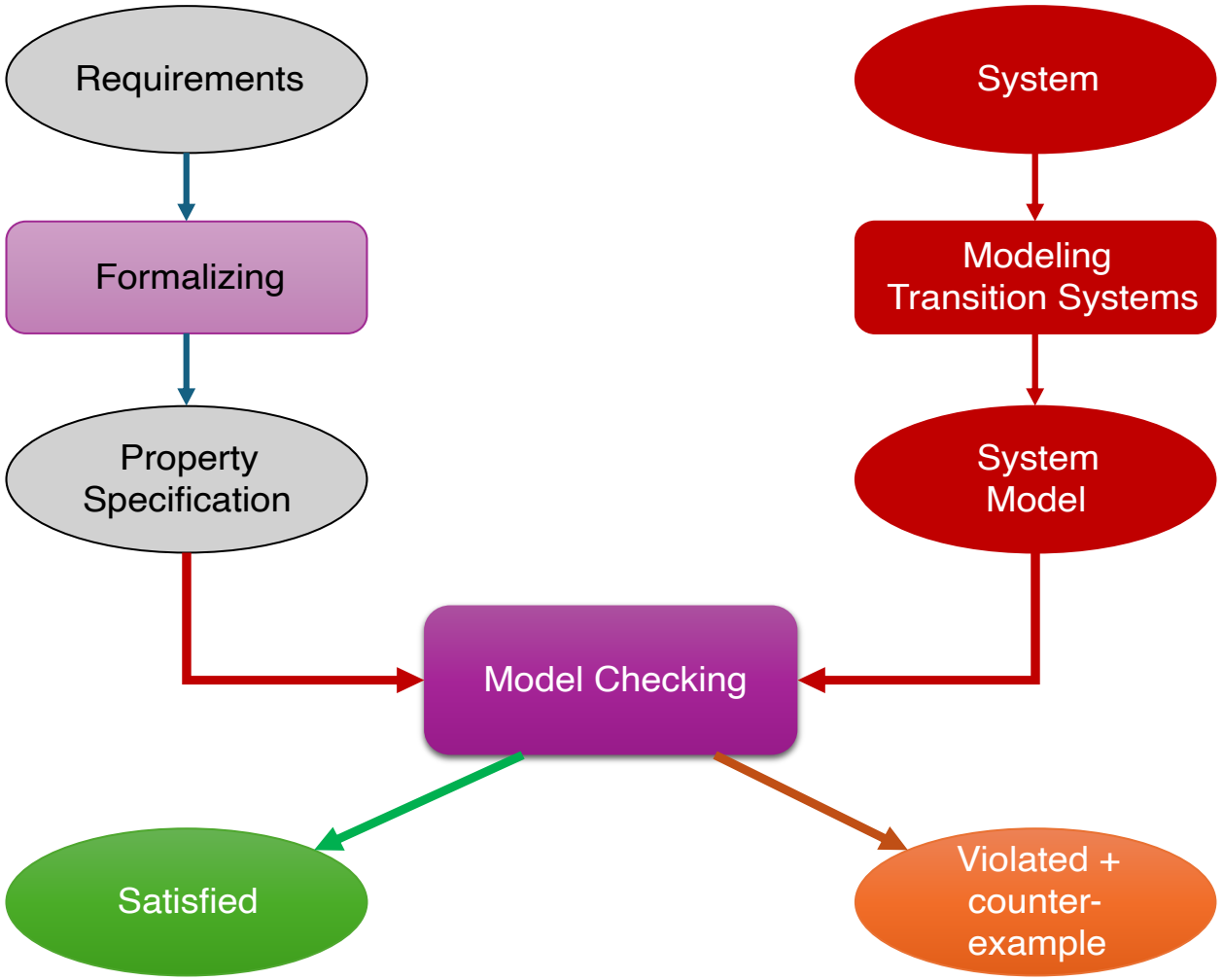
CentraleSupélec
université PARIS-SACLAY

PLAN

Retour au plan - Retour à l'accueil

CentraleSupélec
université PARIS-SACLAY

PRINCIPE DU MODEL-CHECKING



CentraleSupélec
université PARIS-SACLAY

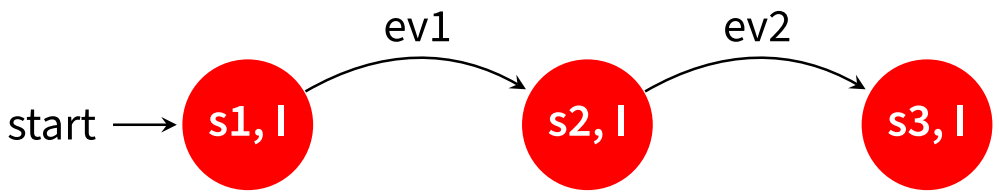
SYSTÈMES DE TRANSITION

- **modèle** pour décrire le comportement des systèmes
- **digraphes** où les **nœuds** représentent les états et les **arêtes** représentent les transitions
- **états** :
  - la couleur actuelle d'un feu de circulation : rouge, vert, orange.
  - **software :** les valeurs actuelles de toutes les variables du programme...
  - **hardware :** la valeur actuelle des registres ainsi que les valeurs des bits d'entrée
- **transitions** : ("changement d'états")
  - un passage d'une couleur à une autre
  - **software :** l'exécution d'une instruction de programme
  - **hardware :** le changement des registres et des bits de sortie pour une nouvelle entrée

CentraleSupélec
université PARIS-SACLAY

MODELLING WITH EVENT-B

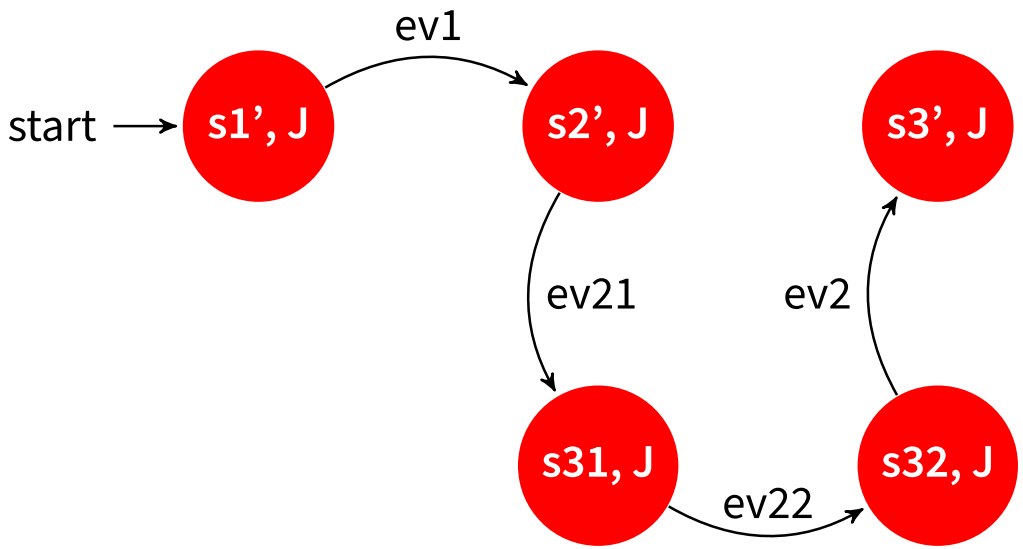
- An **Event-B** specification contains:
  - a state (data, sets, relationships, ...)
  - invariants, properties (first order predicates logic)
  - transitions (intialisation and events) to update the state (substitutions)



CentraleSupélec
université PARIS-SACLAY

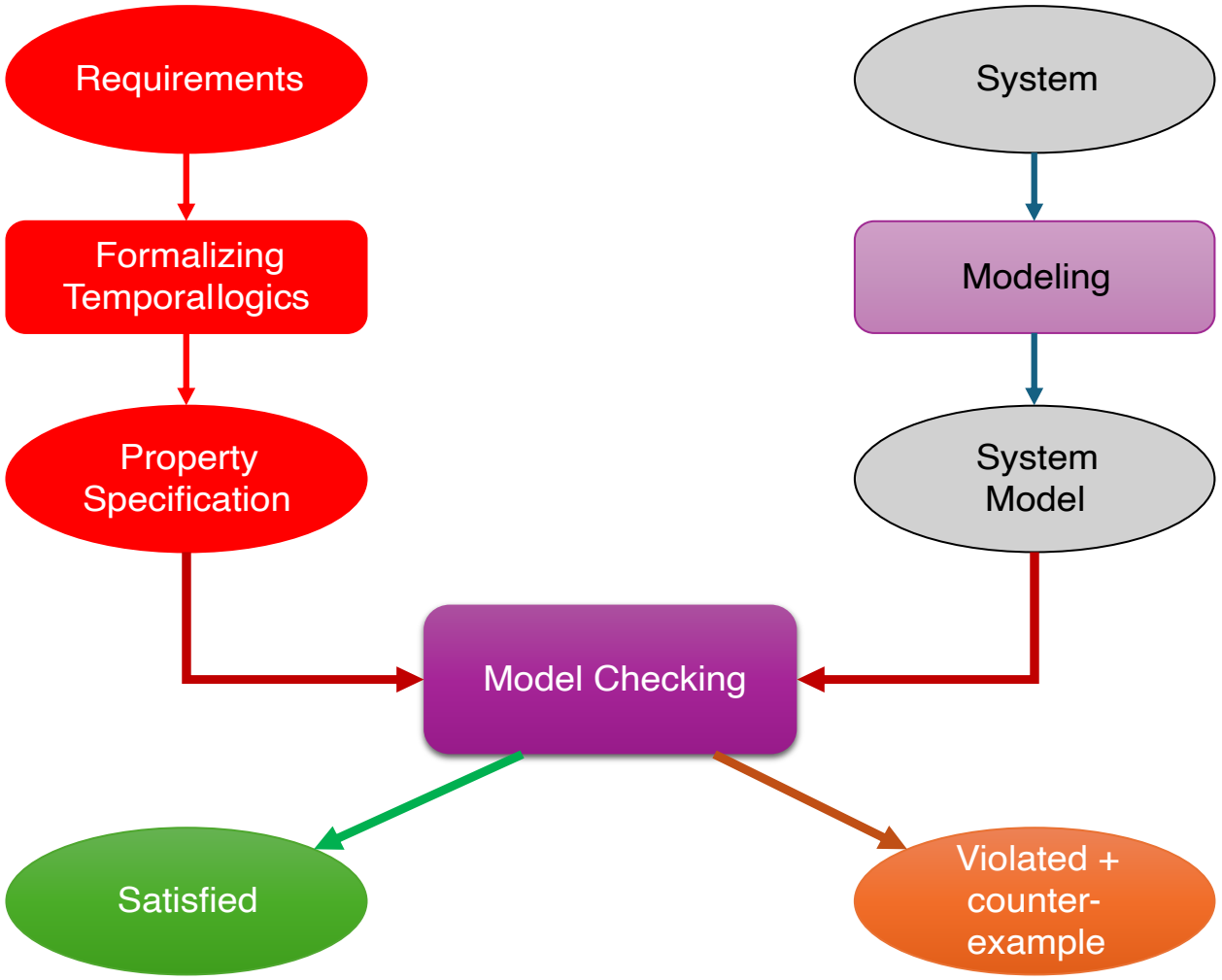
THE REFINMENT OF AN EVENT-B MODEL

- Refining a specification consists of enriching it and reformulating it with another more concrete specification.



CentraleSupélec
université PARIS-SACLAY

PRINCIPE DU MODEL-CHECKING



CentraleSupélec
université PARIS-SACLAY

LOGIQUES TEMPORELLES

- 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**.

$$\phi \quad ::= \quad true \quad | \quad a \quad | \quad \phi \quad \wedge \quad \phi \quad | \quad \neg\, \phi \quad | \quad \exists x.\ \phi \quad | \quad \dots$$

  - **Exemple** : "toute requête sera un jour satisfaite"
  - $\forall t.\, (\text{requete} \rightarrow \exists t' \geq t.\, (\text{reponse}))$
  - ✘ difficile à écrire/comprendre
  - ✘ vérification peu efficace

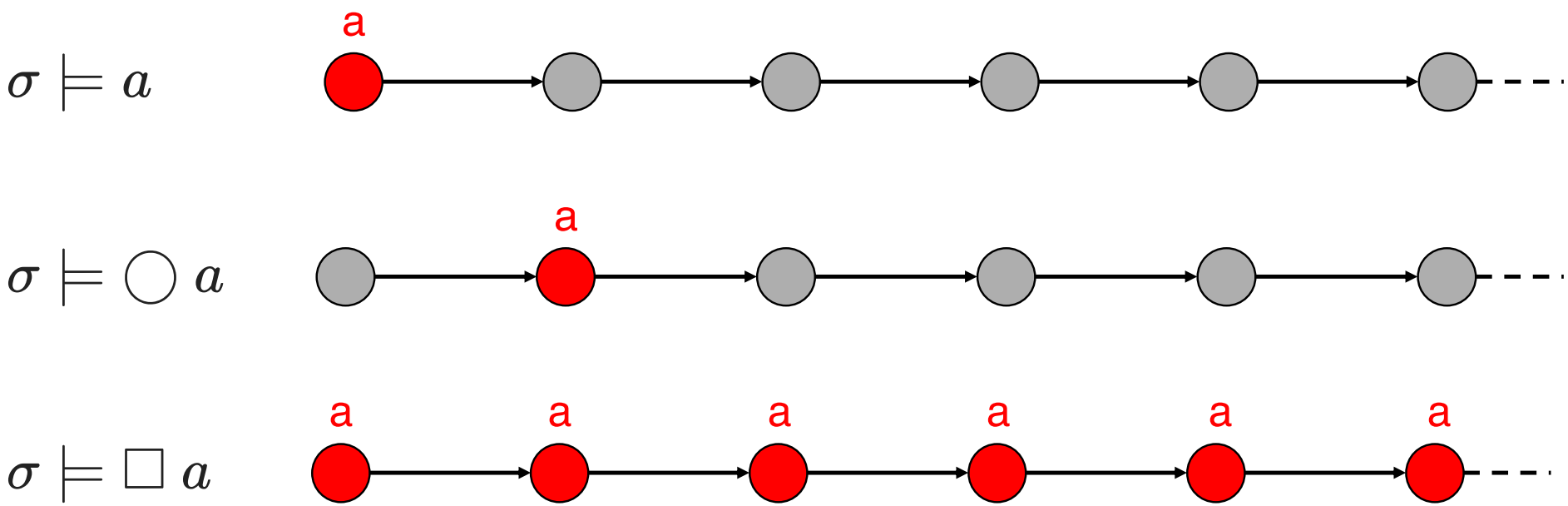
- Pas de variable pour gérer le temps (instants implicites)
- **Temporel $\neq$ temporisé**
  la logiques temporelles ne quantifient pas écoulement du temps.

CentraleSupélec
université PARIS-SACLAY

PROPOSITIONAL LINEAR TEMPORAL LOGIC (LTL)

$$\phi \quad ::= \quad true \quad | \quad a \quad | \quad \phi \quad \wedge \quad \phi \quad | \quad \neg \phi \quad | \quad \bigcirc \phi \quad | \quad \Box \phi$$

avec $a \in AP$ $\qquad$ $\bigcirc \equiv \mathrm{X}$ (next) $\qquad$ $\Box \equiv \mathrm{G}$ (always)



CentraleSupélec
université PARIS-SACLAY

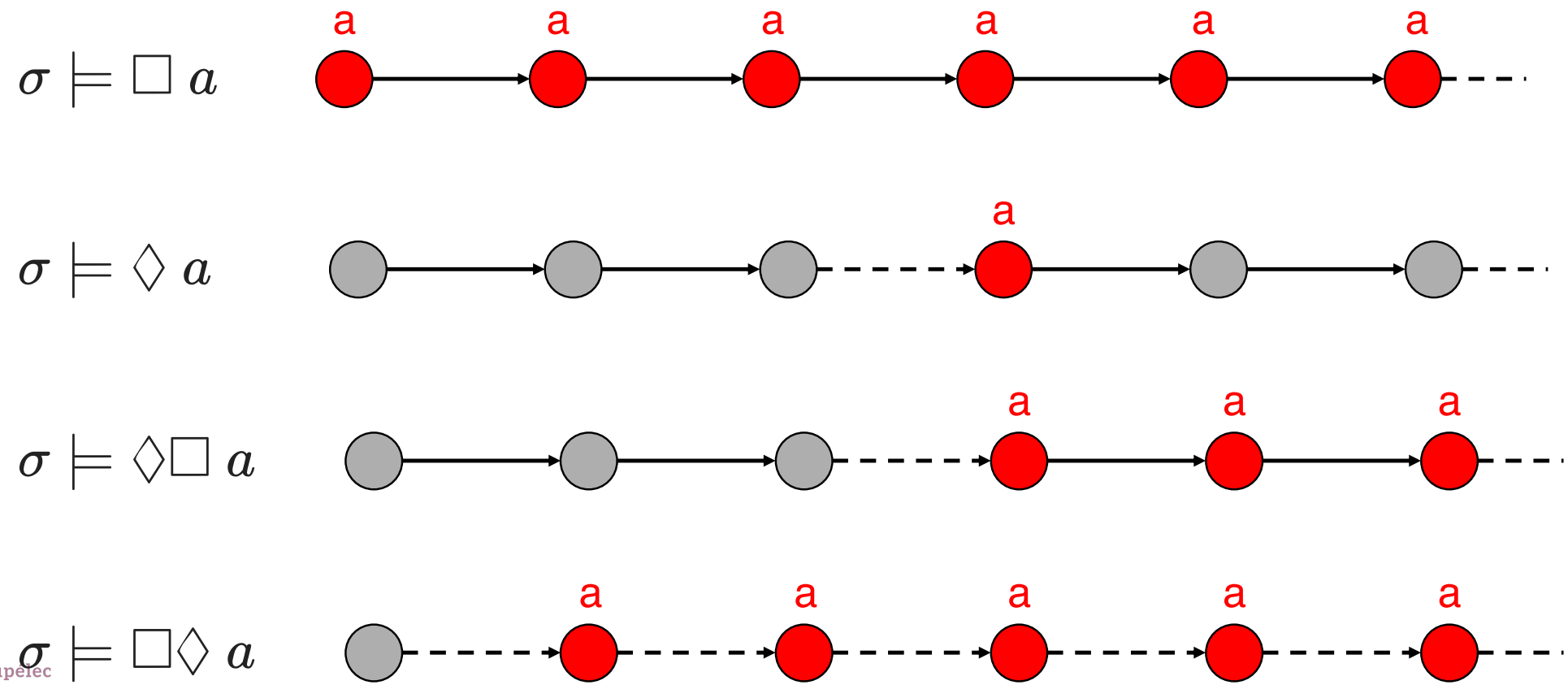
OPÉRATEURS TEMPORELS DÉRIVÉS

$\Box\,\phi \equiv \mathrm{G}\,\phi$ (always)

$\Diamond\,\phi \equiv \mathrm{F}\,\phi \equiv \neg\Box\neg\phi$ (eventually)

$\Diamond\Box\,\phi$ (persistence)

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

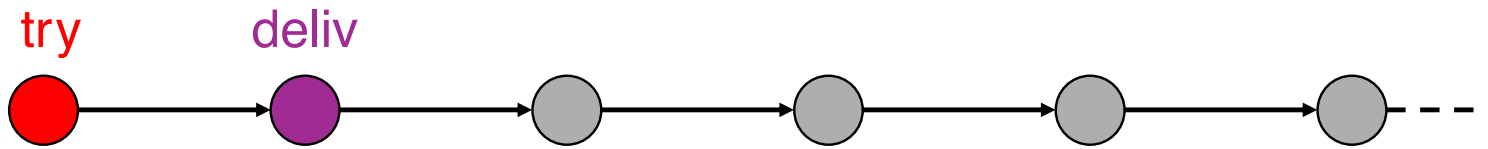


CentraleSupélec
université PARIS-SACLAY

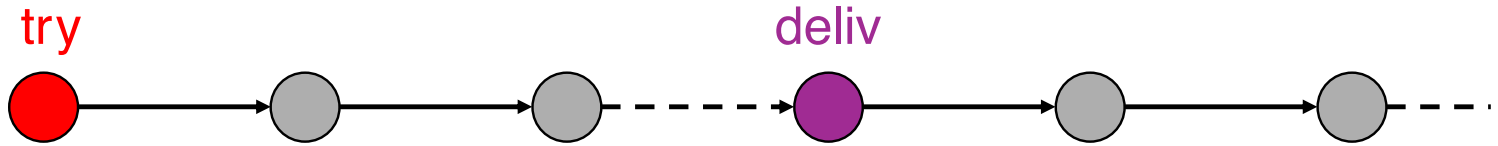
L'OPÉRATEUR UNTIL

$$\phi \quad ::= \quad true \mid a \mid \phi \wedge \phi \mid \neg \phi \mid \bigcirc \phi \mid \Box \phi \mid \phi \bigcup \phi$$

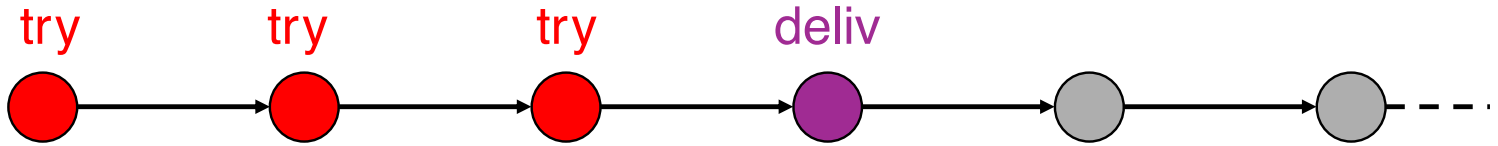
$try \Rightarrow \bigcirc deliv$



$try \Rightarrow \Diamond deliv$



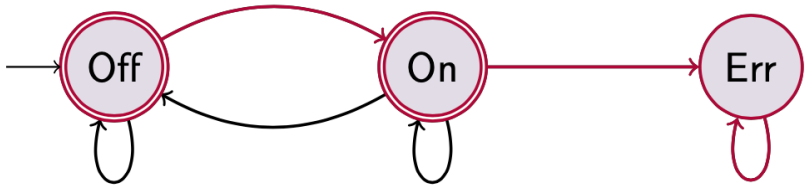
$try \bigcup deliv$



$$\Diamond \phi \equiv true \cup \phi \quad \text{et} \quad \Box \phi \equiv \neg \Diamond \neg \phi$$

CentraleSupélec
université PARIS-SACLAY

EXEMPLE I

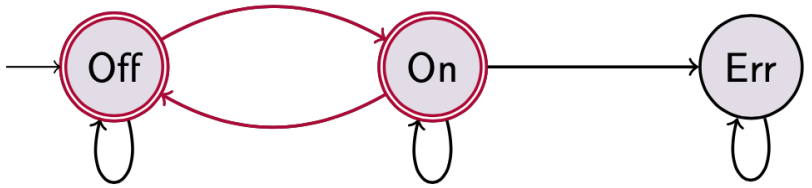


Prenant la trace $\sigma = \text{Off On Err Err Err } \ldots = \text{Off On Err}^{\omega}$

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

CentraleSupélec
université PARIS-SACLAY

EXEMPLE II



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

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

CentraleSupélec
université PARIS-SACLAY

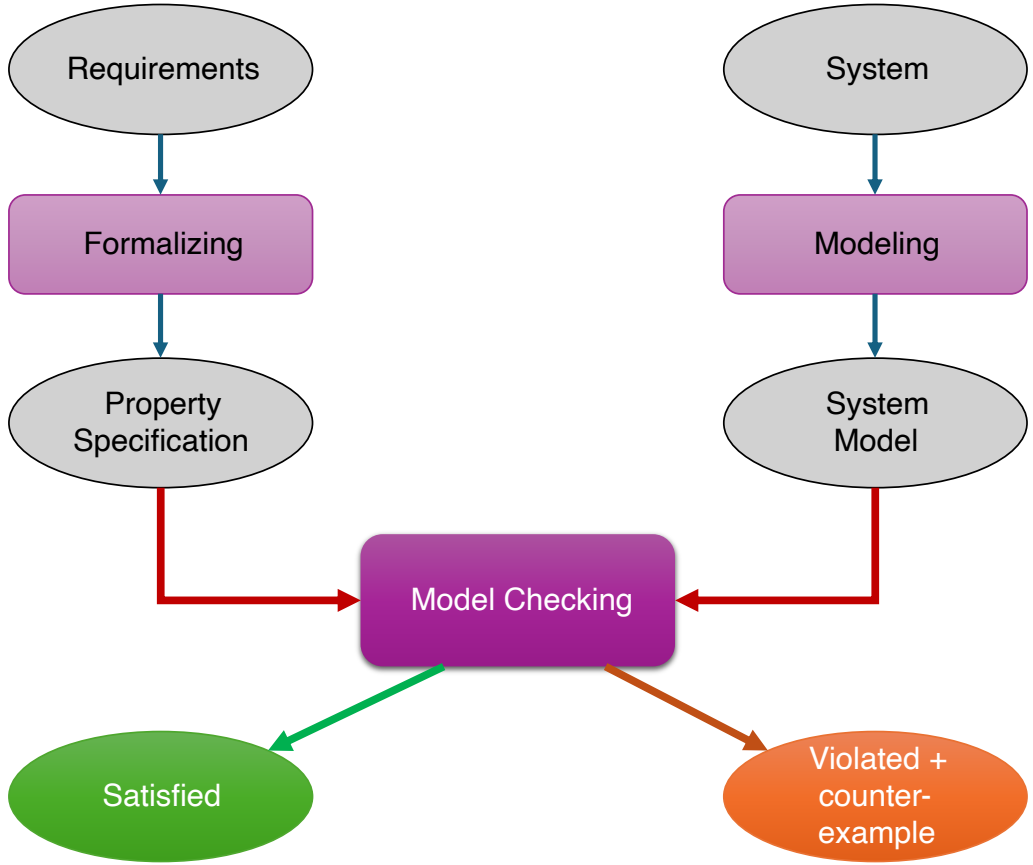
INVARIANTS, SAFETY AND LIVENESS PROPERTIES

- **Safety properties** $\rightarrow$ "nothing bad should happen"
  - typical safety property: mutual exclusion property
  - the bad thing (having > 1 processes in the critical section) never occurs
- An **invariant property** is a particular safety property
  - requires that condition ϕ holds for all states (reachable ones $\rightarrow$ Invariant: $G\ \phi$)
  - e.g. for mutual exclusion property $\phi = \neg(crit_1 \wedge crit_2)$
- Safety properties are complemented by **Liveness properties**
  - that require some progress
  - that assert: "something good" will happen eventually
  - e.g. Eventually : $F\ crit_1 \wedge F\ crit_2$

CentraleSupélec
université PARIS-SACLAY

PRINCIPE DU MODEL-CHECKING

1. **Modeling phase**
   - model the system under consideration into a formal representation
   - formalize the property to check unsing a temporal logic
2. **Running phase**
   - run automatically the model checker to check the validity of the property
3. **Analysis phase** (3 cases)
   1. property satisfied: check next property
   2. property violated:
      - analyze generated counterexample
      - modify the model and repeat
   3. out of memory: try to reduce the model



CentraleSupélec
université PARIS-SACLAY

THE PROS OF MODEL-CHECKING

- ✔ widely applicable (hardware, software, protocol systems, ...)
- ✔ potential "push-button" technology (software-tools)
- ✔ rapidly increasing industrial interest
- ✔ in case of property violation, a counter-example is provided
- ✔ sound and interesting mathematical foundations
- ✔ not baised to the most possible scenarios (such as testing)

CentraleSupélec
université PARIS-SACLAY

THE CONS OF MODEL-CHECKING

✘ mainly focused on control-intensive systems

→ state explosion problem must be addressed to apply to data-oriented systems

✘ model checking is based on two error-prone activities:

→ system Modeling

→ property specification

doing things right $\not\Rightarrow$ doing the right thing

CentraleSupélec

université PARIS-SACLAY

PLAN

Retour au plan - Retour à l'accueil

CentraleSupélec
université PARIS-SACLAY

THE PROB ANIMATOR AND MODEL CHECKER

- **ProB** is an animator, constraint solver and model checker for the **Event-B Method**.
- **ProB**'s animation features allow developers to control and validate the behavior of their specifications.
- Animation features are useful for infinite state machines, not for verification, but for debugging and testing.

ProB main web page

CentraleSupélec
université PARIS-SACLAY

MODEL-CHECKING WITH PROB

- The **ProB plugin** allows automatic verification of the consistency of **Event-B** machines through animation and model checking.
- For exhaustive model verification, the given sets must be limited to finite sets.
  - ➠ allows ProB to browse through the reachable states of the machine.
- The **ProB plugin** graphically displays a counterexample when it discovers a property violation.

CentraleSupélec
université PARIS-SACLAY

THE RODIN PLATFORM

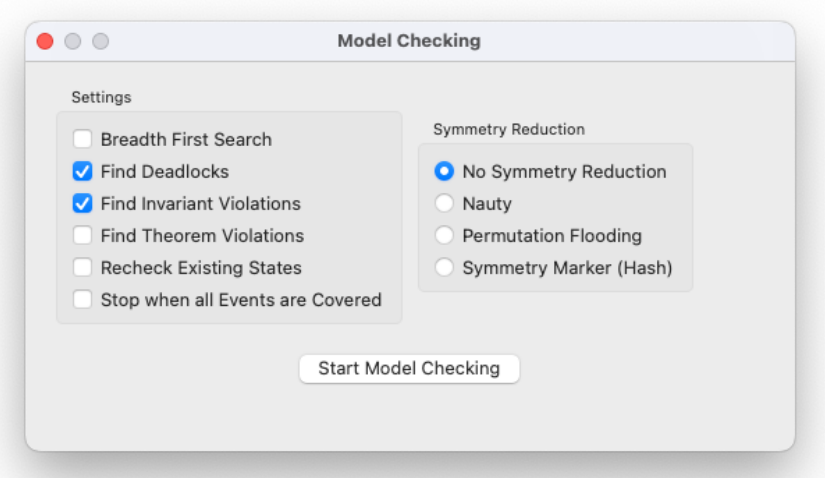
Required plugins for this tutorial :

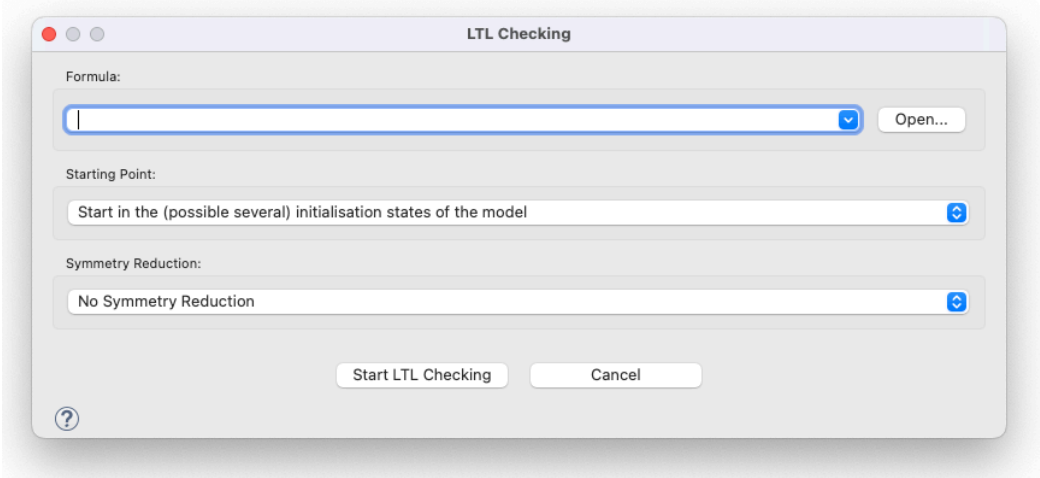
menu : `Help -> Install New Software ...`

- the **Atelier B Provers plugin** from the **Atelier B Provers** Update site.
  `https://www.atelierb.eu/update_site/atelierb_provers`
- the **ProB plugin** from the **ProB** Update site.
  `https://stups.hhu-hosting.de/rodin/prob1/release/`
- the **Theory plugin** from the **Rodin Plug-ins (archive)** Update site.
  `https://rodin-b-sharp.sourceforge.net/updates-archive`

CentraleSupélec
université PARIS-SACLAY

THE PROB PLUGIN





- Tutorial Rodin First Step
- Tutorial First Model Checking
- LTL/CTL Model Checking

CentraleSupélec
université PARIS-SACLAY

PLAN

Retour au plan - Retour à l'accueil

CentraleSupélec
université PARIS-SACLAY

CONCLUSION ABOUT PROB

- The **ProB plugin** is useful in addition to the proof tools.
- As the interactive proof process can be quite long, the **ProB plugin** can be used as a complement to the interactive proof.
- Some errors will be discovered sooner and designers will waste less effort proving incorrect POs.

CentraleSupélec
université PARIS-SACLAY

MERCI

Version PDF des slides

Retour à l'accueil - Retour au plan

CentraleSupélec

université PARIS-SACLAY