LMF | CentraleSupélec | université PARIS-SACLAY

CentraleSupélec

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 - 2024/2025

Idir AIT SADOUNE
idir.aitsadoune@centralesupelec.fr

OUTLINE

Back to the begin - Back to the outline

OUTLINE

Back to the begin - Back to the outline

THE PROOF WITH ATELIER-B

- There are two main proof activities in the **Event-B** method :
  1. the proof of consistency used to show that the events of a machine preserve the invariant,
  2. the proof of refinement used to show that one machine is a valid refinement of another.

THE PROOF WITH ATELIER-B

- There are two main proof activities in the **Event-B** method :
  1. the proof of consistency used to show that the events of a machine preserve the invariant,
  2. the proof of refinement used to show that 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 PROOF WITH ATELIER-B

- There are two main proof activities in the **Event-B** method :
  1. the proof of consistency used to show that the events of a machine preserve the invariant,
  2. the proof of refinement used to show that 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 PROOF WITH ATELIER-B

- There are two main proof activities in the **Event-B** method :
  1. the proof of consistency used to show that the events of a machine preserve the invariant,
  2. the proof of refinement used to show that 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

THE PROOF WITH ATELIER-B

- There are two main proof activities in the **Event-B** method :
  1. the proof of consistency used to show that the events of a machine preserve the invariant,
  2. the proof of refinement used to show that 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

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/Event-B Method)

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/Event-B Method)

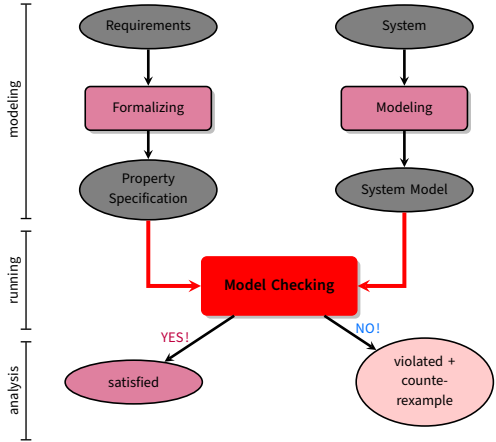
After 80's

- 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)

OUTLINE

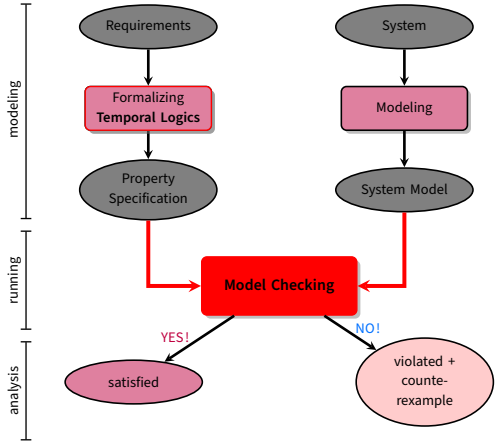
Back to the begin - Back to the outline

PRINCIPLE OF MODEL-CHECKING



CentraleSupélec

PRINCIPLE OF MODEL-CHECKING



CentraleSupélec

PROPOSITIONAL LOGIC

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

where $a \in AP$

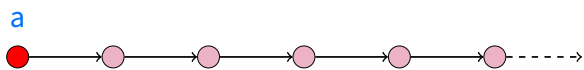
PROPOSITIONAL LINEAR TEMPORAL LOGIC

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

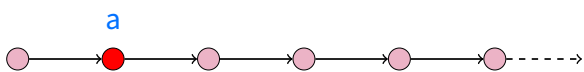
where $a \in AP$

$\bigcirc$
(next)

$\sigma \models a$



$\sigma \models \bigcirc a$



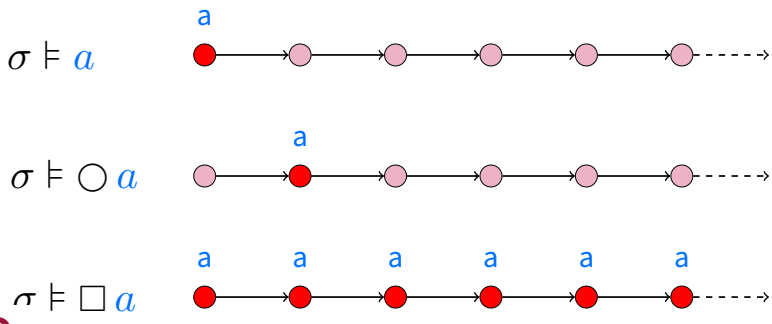
CentraleSupélec

PROPOSITIONAL LINEAR TEMPORAL LOGIC

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

where $a \in AP$

$\bigcirc$ (next) $\Box$ (always)



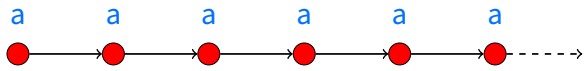
CentraleSupélec

LTL : DERIVED TEMPORAL OPERATORS

$\Box\phi$

(always)

$\sigma \models \Box\, a$

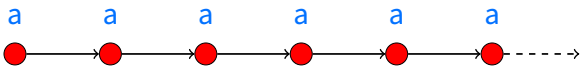


LTL : DERIVED TEMPORAL OPERATORS

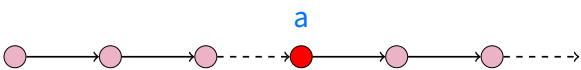
$\Box\phi$ (always)

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

$\sigma \models \Box\, a$



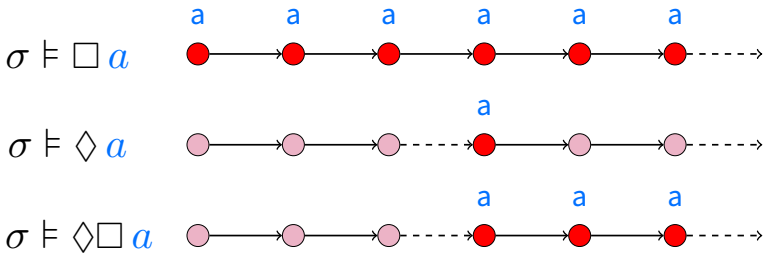
$\sigma \models \Diamond\, a$



CentraleSupélec

LTL : DERIVED TEMPORAL OPERATORS

$\Box\phi$ (always) $\Diamond\phi \equiv \neg\Box\neg\phi$ (eventually) $\Diamond\Box\phi$ (persistence)



CentraleSupélec

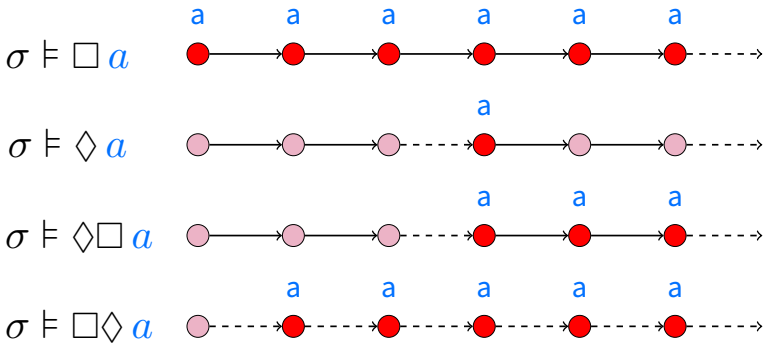
LTL : DERIVED TEMPORAL OPERATORS

$\Box\phi$ (always)

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

$\Diamond\Box\phi$ (persistence)

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



$\{\, i \mid a \in \sigma(i) \,\}$ is infinite

CentraleSupélec

EXAMPLE OF TEMPORAL PROPERTIES

- **Safety** :
  - mutual exclusion : $\Box\neg(crit_1 \wedge crit_2)$
  - elevator : $\Box(moving \Rightarrow doors_{closed})$
  - traffic light : $\Box(yellow \Rightarrow \bigcirc red)$

EXAMPLE OF TEMPORAL PROPERTIES

- **Safety** :
  - mutual exclusion : $\Box\neg(crit_1 \wedge crit_2)$
  - elevator : $\Box(moving \Rightarrow doors_{closed})$
  - traffic light : $\Box(yellow \Rightarrow \bigcirc red)$
- **Liveness** :
  - progress : $\Diamond\, progress$
  - response : $\Box(try\_to\_send \Rightarrow \Diamond delivered)$
  - termination : $\Diamond\Box\, terminated$

CentraleSupélec

EXAMPLE OF TEMPORAL PROPERTIES

- **Safety** : nuclear plant
  - cooling : $\Box\neg(temp_{high} \wedge cooling_{low})$
  - alarm : $\Box(temp_{high} \Rightarrow alarm)$
  - saving : $\Box(temp_{high} \Rightarrow \bigcirc react_{low})$

EXAMPLE OF TEMPORAL PROPERTIES

- **Safety** : nuclear plant
  - cooling : $\Box\neg(temp_{high} \wedge cooling_{low})$
  - alarm : $\Box(temp_{high} \Rightarrow alarm)$
  - saving : $\Box(temp_{high} \Rightarrow \bigcirc react_{low})$

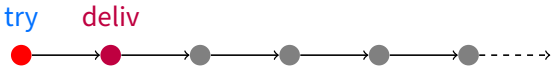
- **Liveness** : nuclear plant
  - reactivity : $\Box\Diamond\, react_{high}$
  - temperature : $\Box(react_{low} \Rightarrow \Diamond temp_{low})$

CentraleSupélec

LTL : UNTIL OPERATOR

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

$try \Rightarrow \bigcirc deliv$

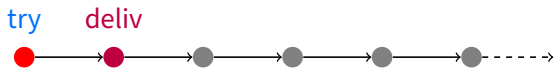


$try \Rightarrow \Diamond deliv$

LTL : UNTIL OPERATOR

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

$try \Rightarrow \bigcirc deliv$



$try \Rightarrow \Diamond deliv$

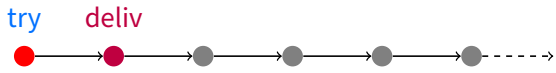
$try \bigcup deliv$

CentraleSupélec

LTL : UNTIL OPERATOR

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

$try \Rightarrow \bigcirc deliv$



$try \Rightarrow \Diamond deliv$

$try \bigcup deliv$

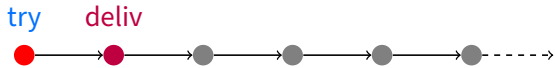
$$\Diamond\,\phi \;\equiv\; \textbf{true} \;\cup\; \phi$$

CentraleSupélec

LTL : UNTIL OPERATOR

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

$try \Rightarrow \bigcirc deliv$



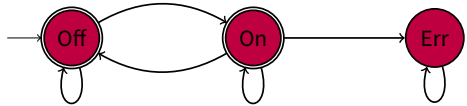
$try \Rightarrow \Diamond deliv$

$try \bigcup deliv$

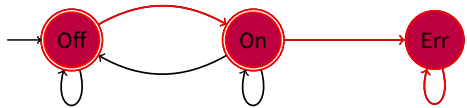
$\Diamond \phi \equiv$ **true** $\bigcup \phi$ and $\Box \phi \equiv \neg \Diamond \neg \phi$

CentraleSupélec

PROPERTIES OF A TRACE



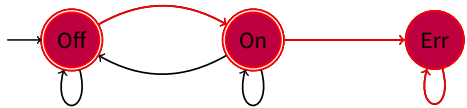
PROPERTIES OF A TRACE



have a path $\pi = Off\, On\, Err\, Err\, Err \ldots = Off\, On\, Err^{\omega}$

- $\pi \models Off$

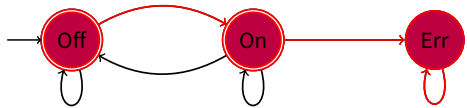
PROPERTIES OF A TRACE



have a path $\pi = Off\,On\,Err\,Err\,Err\,... = Off\,On\,Err^{\omega}$

- $\pi \models Off$, but $\pi \not\models On$

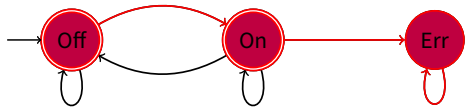
PROPERTIES OF A TRACE



have a path $\pi = Off\,On\,Err\,Err\,Err\,... = Off\,On\,Err^{\omega}$

- $\pi \models Off$, but $\pi \not\models On$, so $\pi \models \neg On$

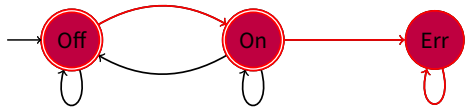
PROPERTIES OF A TRACE



have a path $\pi = Off\,On\,Err\,Err\,Err\,... = Off\,On\,Err^{\omega}$

- $\pi \models Off$, but $\pi \not\models On$, so $\pi \models \neg On$
- $\pi \models \bigcirc On$

PROPERTIES OF A TRACE

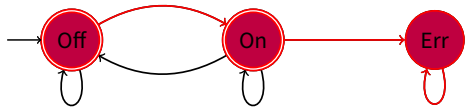


have a path $\pi = Off\,On\,Err\,Err\,Err\,... = Off\,On\,Err^{\omega}$

- $\pi \models Off$, but $\pi \not\models On$, so $\pi \models \neg On$
- $\pi \models \bigcirc On$
- $\pi \models \bigcirc \bigcirc Err$

CentraleSupélec

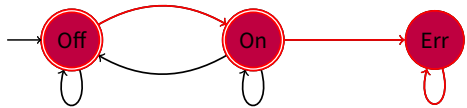
PROPERTIES OF A TRACE



have a path $\pi = Off\, On\, Err\, Err\, Err \ldots = Off\, On\, Err^{\omega}$

- $\pi \models Off$, but $\pi \not\models On$, so $\pi \models \neg On$
- $\pi \models \bigcirc On$
- $\pi \models \bigcirc \bigcirc Err$
- $\pi \models (Off \vee On) \cup Err$

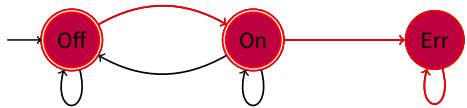
PROPERTIES OF A TRACE



have a path $\pi = Off\, On\, Err\, Err\, Err \ldots = Off\, On\, Err^{\omega}$

- $\pi \models Off$, but $\pi \not\models On$, so $\pi \models \neg On$
- $\pi \models \bigcirc On$
- $\pi \models \bigcirc \bigcirc Err$
- $\pi \models (Off \vee On) \cup Err$
- $\pi \models \Box(Err \Rightarrow \bigcirc Err)$

PROPERTIES OF A TRACE

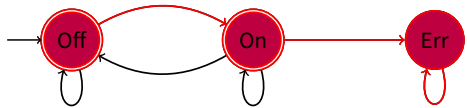


have a path $\pi = Off\,On\,Err\,Err\,Err\,... = Off\,On\,Err^{\omega}$

- $\pi \models Off$, but $\pi \not\models On$, so $\pi \models \neg On$
- $\pi \models \bigcirc On$
- $\pi \models \bigcirc \bigcirc Err$
- $\pi \models (Off \vee On) \cup Err$
- $\pi \models \Box(Err \Rightarrow \bigcirc Err)$
- $\pi \models \Box(Err \Rightarrow \Box Err)$

CentraleSupélec

PROPERTIES OF A TRACE

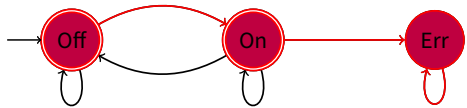


have a path $\pi = Off\,On\,Err\,Err\,Err\,... = Off\,On\,Err^{\omega}$

- $\pi \models Off$, but $\pi \not\models On$, so $\pi \models \neg On$
- $\pi \models \bigcirc On$
- $\pi \models \bigcirc \bigcirc Err$
- $\pi \models (Off \vee On) \cup Err$
- $\pi \models \Box(Err \Rightarrow \bigcirc Err)$
- $\pi \models \Box(Err \Rightarrow \Box Err)$
- $\pi \models \Diamond\Box Err$ $(persistence)$

CentraleSupélec

PROPERTIES OF A TRACE

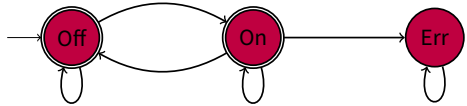


have a path $\pi = Off\,On\,Err\,Err\,Err\,... = Off\,On\,Err^{\omega}$

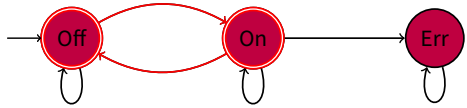
- $\pi \models Off$, but $\pi \not\models On$, so $\pi \models \neg On$
- $\pi \models \bigcirc On$
- $\pi \models \bigcirc \bigcirc Err$
- $\pi \models (Off \vee On) \cup Err$
- $\pi \models \Box(Err \Rightarrow \bigcirc Err)$
- $\pi \models \Box(Err \Rightarrow \Box Err)$
- $\pi \models \Diamond\Box Err$ $(persistence)$
- $\pi \models \bigcirc \bigcirc \Box Err$

CentraleSupélec

PROPERTIES OF A TRACE



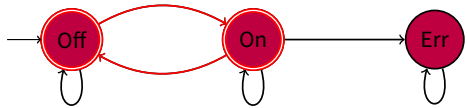
PROPERTIES OF A TRACE



have a path $\pi = Off\,On\,Off\,On\,Off\,... = (Off\,On)^{\omega}$

- $\pi \overset{?}{\models} (Off \vee On) \cup Err$

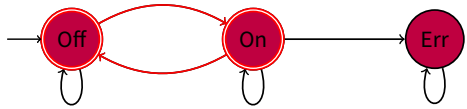
PROPERTIES OF A TRACE



have a path $\pi = Off On Off On Off ... = (Off On)^{\omega}$

- $\pi \not\models (Off \vee On) \cup Err$

PROPERTIES OF A TRACE

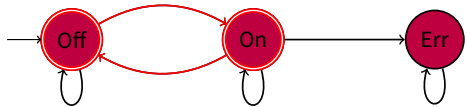


have a path $\pi = Off\,On\,Off\,On\,Off\ldots = (Off\,On)^{\omega}$

- $\pi \not\models (Off \vee On) \cup Err$
- $\pi \overset{?}{\models} \Diamond Err \Rightarrow ((Off \vee On) \cup Err)$

CentraleSupélec

PROPERTIES OF A TRACE

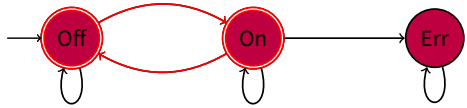


have a path $\pi = Off\,On\,Off\,On\,Off... = (Off\,On)^{\omega}$

- $\pi \not\models (Off \vee On) \cup Err$
- $\pi \models \Diamond Err \Rightarrow ((Off \vee On) \cup Err)$ as $\pi \not\models \Diamond Err$

CentraleSupélec

PROPERTIES OF A TRACE

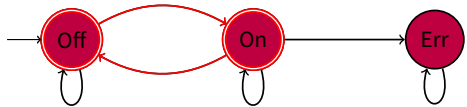


have a path $\pi = Off\,On\,Off\,On\,Off... = (Off\,On)^{\omega}$

- $\pi \not\models (Off \vee On) \cup Err$
- $\pi \models \Diamond Err \Rightarrow ((Off \vee On) \cup Err)$ as $\pi \not\models \Diamond Err$
- $\pi \overset{?}{\models} \Box(On \vee Off)$

CentraleSupélec

PROPERTIES OF A TRACE

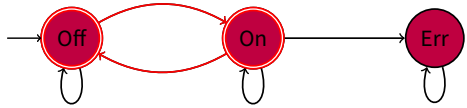


have a path $\pi = Off On Off On Off ... = (Off On)^{\omega}$

- $\pi \not\models (Off \vee On) \cup Err$
- $\pi \models \Diamond Err \Rightarrow ((Off \vee On) \cup Err)$ as $\pi \not\models \Diamond Err$
- $\pi \models \Box(On \vee Off)$

CentraleSupélec

PROPERTIES OF A TRACE

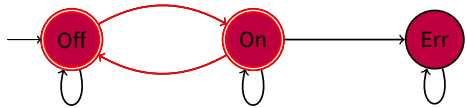


have a path $\pi = Off\,On\,Off\,On\,Off... = (Off\,On)^{\omega}$

- $\pi \not\models (Off \vee On) \cup Err$
- $\pi \models \Diamond Err \Rightarrow ((Off \vee On) \cup Err)$ as $\pi \not\models \Diamond Err$
- $\pi \models \Box(On \vee Off)$
- $\pi \overset{?}{\models} \Box\Diamond On \wedge \Box\Diamond Off$ $(infinitely\,many)$

CentraleSupélec

PROPERTIES OF A TRACE

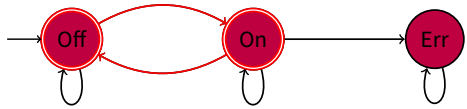


have a path $\pi = Off On Off On Off ... = (Off On)^{\omega}$

- $\pi \not\models (Off \vee On) \cup Err$
- $\pi \models \Diamond Err \Rightarrow ((Off \vee On) \cup Err)$ as $\pi \not\models \Diamond Err$
- $\pi \models \Box(On \vee Off)$
- $\pi \models \Box\Diamond On \wedge \Box\Diamond Off$ $(infinitely\, many)$

CentraleSupélec

PROPERTIES OF A TRACE

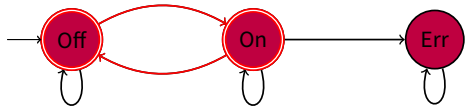


have a path $\pi = Off\,On\,Off\,On\,Off... = (Off\,On)^{\omega}$

- $\pi \not\models (Off \vee On) \cup Err$
- $\pi \models \Diamond Err \Rightarrow ((Off \vee On) \cup Err)$ as $\pi \not\models \Diamond Err$
- $\pi \models \Box(On \vee Off)$
- $\pi \models \Box\Diamond On \wedge \Box\Diamond Off$ $(infinitely\,many)$
- $\pi \overset{?}{\models} \Diamond\Box On \vee \Diamond\Box Off$ $(persistence)$

CentraleSupélec

PROPERTIES OF A TRACE

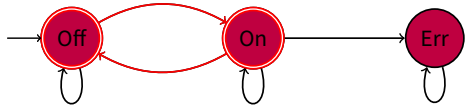


have a path $\pi = Off\,On\,Off\,On\,Off\,... = (Off\,On)^{\omega}$

- $\pi \not\models (Off \vee On) \cup Err$
- $\pi \models \Diamond Err \Rightarrow ((Off \vee On) \cup Err)$ as $\pi \not\models \Diamond Err$
- $\pi \models \Box(On \vee Off)$
- $\pi \models \Box\Diamond On \wedge \Box\Diamond Off$ $(infinitely\,many)$
- $\pi \not\models \Diamond\Box On \vee \Diamond\Box Off$ $(persistence)$

CentraleSupélec

PROPERTIES OF A TRACE

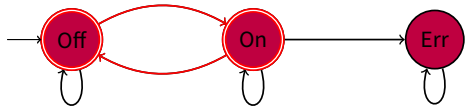


have a path $\pi = Off\,On\,Off\,On\,Off... = (Off\,On)^{\omega}$

- $\pi \not\models (Off \vee On) \cup Err$
- $\pi \models \Diamond Err \Rightarrow ((Off \vee On) \cup Err)$ as $\pi \not\models \Diamond Err$
- $\pi \models \Box(On \vee Off)$
- $\pi \models \Box\Diamond On \wedge \Box\Diamond Off$ $(infinitely\,many)$
- $\pi \not\models \Diamond\Box On \vee \Diamond\Box Off$ $(persistence)$
- $\pi \overset{?}{\models} \Box(Off \Rightarrow \bigcirc On) \wedge \Box(On \Rightarrow \bigcirc Off)$

CentraleSupélec

PROPERTIES OF A TRACE

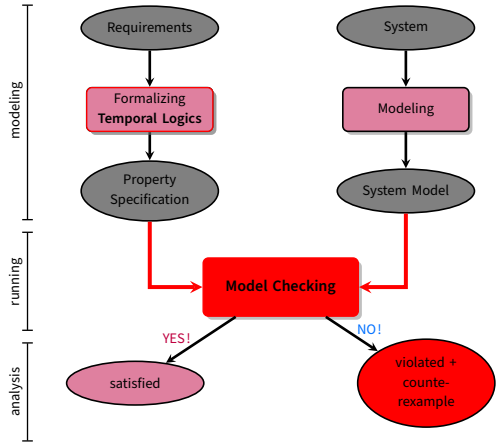


have a path $\pi = Off On Off On Off ... = (Off On)^{\omega}$

- $\pi \not\models (Off \vee On) \cup Err$
- $\pi \models \Diamond Err \Rightarrow ((Off \vee On) \cup Err)$ as $\pi \not\models \Diamond Err$
- $\pi \models \Box(On \vee Off)$
- $\pi \models \Box\Diamond On \wedge \Box\Diamond Off$ $(infinitely\, many)$
- $\pi \not\models \Diamond\Box On \vee \Diamond\Box Off$ $(persistence)$
- $\pi \models \Box(Off \Rightarrow \bigcirc On) \wedge \Box(On \Rightarrow \bigcirc Off)$

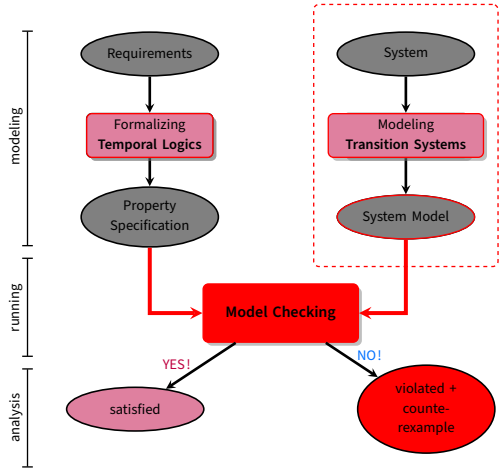
CentraleSupélec

SYSTEM MODELING



CentraleSupélec

SYSTEM MODELING



CentraleSupélec

TRANSITION SYSTEMS

- model to describe the behaviour of systems
- **digraphs** where nodes represent states, and edges represent transitions
- states
  - the current colour of a traffic light : red, green, orange.
- transitions (“state change”)
  - a switch from one colour to another

TRANSITION SYSTEMS

- model to describe the behaviour of systems
- **digraphs** where nodes represent states, and edges represent transitions
- states
  - the current colour of a traffic light : red, green, orange.
  - **software :** the current values of all program variables + the program counter
- transitions (“state change”)
  - a switch from one colour to another
  - **software :** the execution of a program statement

TRANSITION SYSTEMS

- model to describe the behaviour of systems
- **digraphs** where nodes represent states, and edges represent transitions
- states
  - the current colour of a traffic light : red, green, orange.
  - **software :** the current values of all program variables + the program counter
  - **hardware :** the current value of the registers + the input bits
- transitions (“state change”)
  - a switch from one colour to another
  - **software :** the execution of a program statement
  - **hardware :** the change of the registers and output bits for a new input

MODELLING WITH EVENT-B

MODELLING WITH EVENT-B

- An **Event-B** specification contains :

MODELLING WITH EVENT-B

- An **Event-B** specification contains :
  - a state (data, sets, relationships, ...)

CentraleSupélec

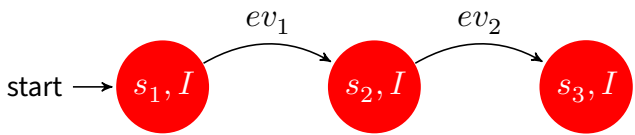
MODELLING WITH EVENT-B

- An **Event-B** specification contains :
  - a state (data, sets, relationships, ...)
  - invariant properties (first order predicates logic)

CentraleSupélec

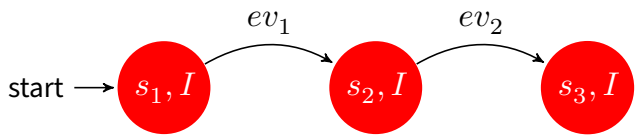
MODELLING WITH EVENT-B

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



CentraleSupélec

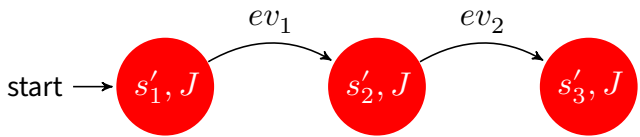
THE REFINEMENT OF AN EVENT-B MODEL



CentraleSupélec

THE REFINEMENT OF AN EVENT-B MODEL

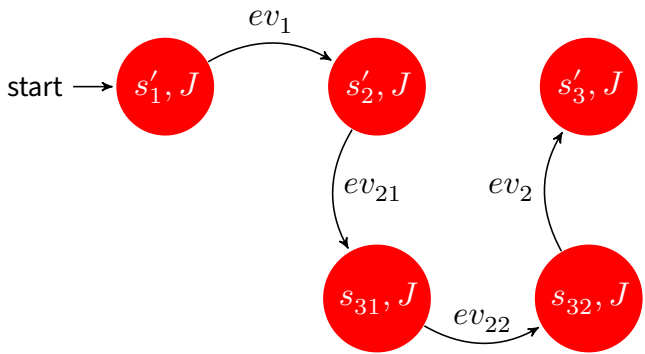
- Refining a specification consists of enriching it and reformulating it with another more concrete specification.



Data refinement (states)

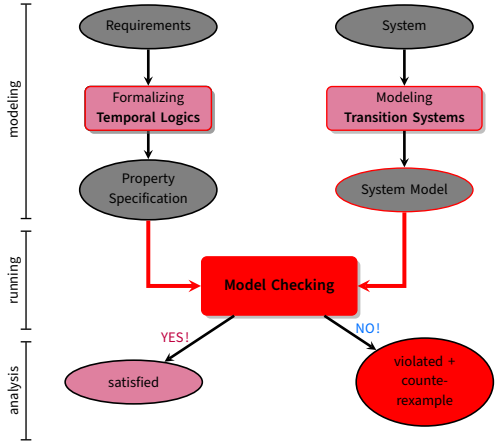
THE REFINEMENT OF AN EVENT-B MODEL

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



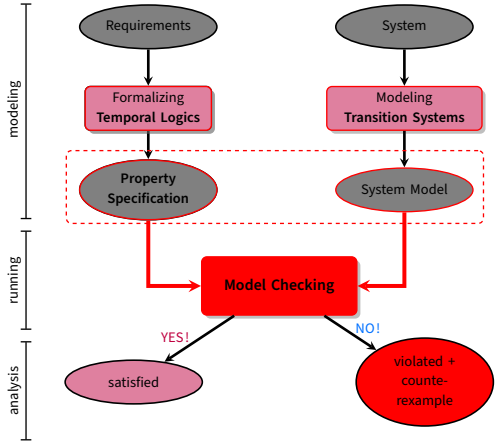
Behavior refinement (events)

PROPERTY SPECIFICATION



CentraleSupélec

PROPERTY SPECIFICATION



CentraleSupélec

INVARIANTS, SAFETY AND LIVENESS PROPERTIES

- **Safety properties** $\longrightarrow$ "nothing bad should happen"
  - Typical safety property : mutual exclusion property
  - the bad thing (having > 1 process in the critical section) never occurs

INVARIANTS, SAFETY AND LIVENESS PROPERTIES

- **Safety properties** $\longrightarrow$ "nothing bad should happen"
  - Typical safety property : mutual exclusion property
  - the bad thing (having > 1 process in the critical section) never occurs

- An **Invariant property** is a particular safety property
  - that is given by a condition ϕ over AP
  - requires that condition ϕ holds for all states (reachable ones)
  - e.g. for mutual exclusion property $\phi = \neg(crit_1 \wedge crit_2)$

INVARIANTS, SAFETY AND LIVENESS PROPERTIES

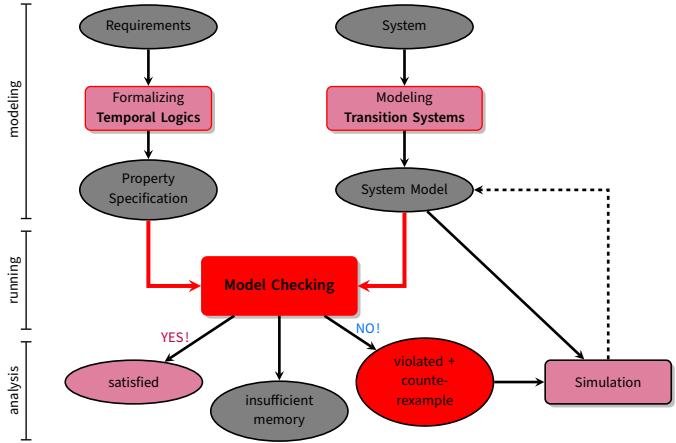
- **Safety properties** $\longrightarrow$ "nothing bad should happen"
  - Typical safety property : mutual exclusion property
  - the bad thing (having > 1 process in the critical section) never occurs

- An **Invariant property** is a particular safety property
  - that is given by a condition ϕ over AP
  - requires that condition ϕ holds for all states (reachable ones)
  - 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 : $\Diamond\, crit_1 \wedge \Diamond\, crit_2$

CentraleSupélec

MODEL CHECKING PROCESS



CentraleSupélec

MODEL CHECKING PROCESS

1. **Modeling phase**
   - Model the system under consideration into a formal representation
   - Formalize the property to check using a temporal logic

MODEL CHECKING PROCESS

1. **Modeling phase**
   - Model the system under consideration into a formal representation
   - Formalize the property to check using a temporal logic

2. **Running phase**
   - run automatically the model checker to check the validity of the property in the model

MODEL CHECKING PROCESS

1. **Modeling phase**
   - Model the system under consideration into a formal representation
   - Formalize the property to check using a temporal logic

2. **Running phase**
   - run automatically the model checker to check the validity of the property in the model

3. **Analysis phase** (3 cases)
   - property satisfied : check next property (if any)
   - property violated :
     - analyze generated counterexample by simulation
     - modify the model and repeat the entire procedure
   - out of memory : try to reduce the model (abstraction) and try again

THE PROS OF MODEL CHECKING

THE PROS OF MODEL CHECKING

- ✔ widely applicable (hardware, software, protocol systems, ...)

THE PROS OF MODEL CHECKING

- ✔ widely applicable (hardware, software, protocol systems, ...)
- ✔ potential “push-button” technology (software-tools)

THE PROS OF MODEL CHECKING

- ✔ widely applicable (hardware, software, protocol systems, ...)
- ✔ potential “push-button” technology (software-tools)
- ✔ rapidly increasing industrial interest

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

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

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 biased to the most possible scenarios (such as testing)

THE CONS OF MODEL CHECKING

THE CONS OF MODEL CHECKING

- ✘ mainly focused on control-intensive systems
  - state explosion problem must be addressed to apply to data-oriented systems

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

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

OUTLINE

Back to the begin - Back to the outline

THE PROB
ANIMATOR AND MODEL CHECKER

ProB Main Page

THE PROB ANIMATOR AND MODEL CHECKER

- **ProB** is an animator, constraint solver and model checker for the **Event-B Method**.

ProB Main Page

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.

ProB Main Page

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 Page

CentraleSupélec

MODEL CHECKING WITH PROB

MODEL CHECKING WITH PROB

- The **ProB plugin** allows automatic verification of the consistency of **Event-B** machines through animation and model checking.

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.

MODEL CHECKING WITH PROB

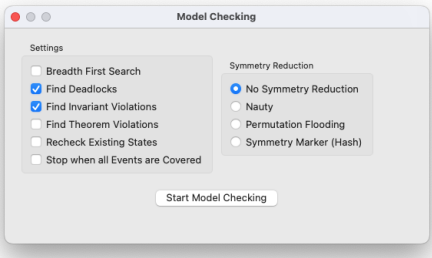
- The **ProB plugin** allows automatic verification of the consistency of **Event-B** machines through animation and model checking.
- For exhaustive model verification, the given sets must be limited to finite sets.
  - ➠ allows ProB to browse through the reachable states of the machine.

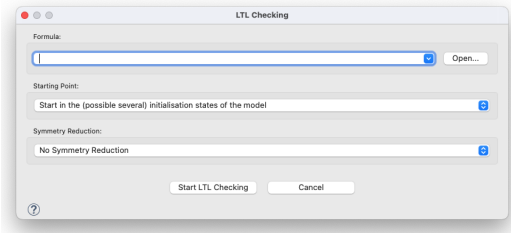
MODEL CHECKING WITH PROB

- 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

THE PROB PLUGIN





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

CentraleSupélec

OUTLINE

Back to the begin - Back to the outline

CONCLUSION ABOUT PROB

CONCLUSION ABOUT PROB

- The **ProB plugin** is useful in addition to the proof tools.

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.

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.

THANK YOU

Back to the begin - Back to the outline