Analyse prévisionnelle des fautes des systèmes embarques discrets par vérification model-based

Matthieu PERIN, Jean-Marc FAURE

Laboratoire Universitaire de Recherche en Production Automatisée

ENS de CACHAN

Plan de l'exposé

- Objectif des travaux
- Principe de la méthode d'analyse
- Modélisation du processus
- Interprétation des résultats de preuve
- Conclusions et perspectives

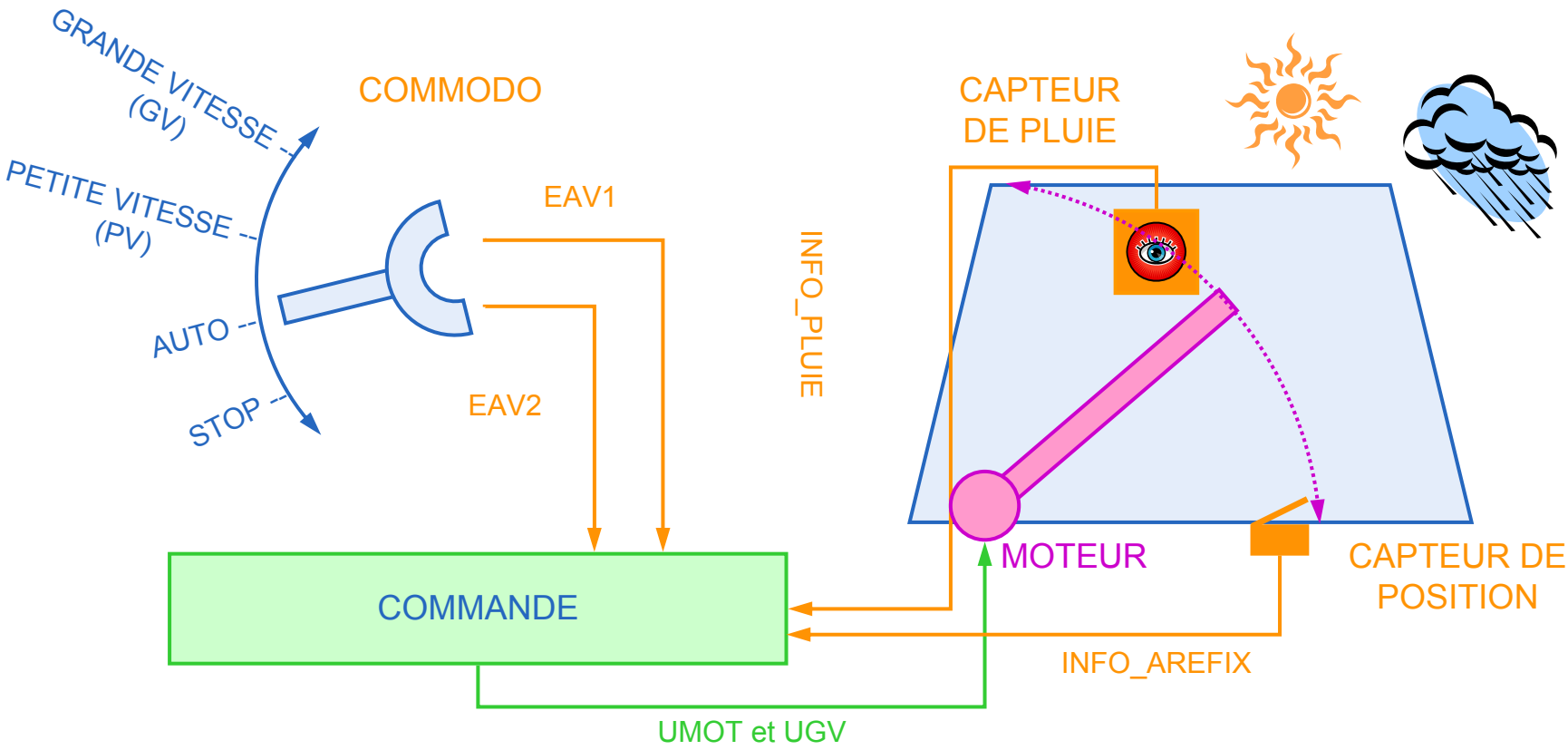
Apport escompté

- **Méthodes conventionnelles d'analyse de sûreté pour un système embarqué**
  - Analyse prévisionnelle des fautes, par exemple par arbre de défaillances, pour la partie matérielle (seules les fautes aléatoires sont considérées)
  - Analyse prévisionnelle par vérification formelle pour la partie logicielle (seules les fautes systématiques sont considérées)
  - Analyse du système complet, suite à l' implantation, par des tests

- **Notre apport : analyse de la propagation par la commande des fautes aléatoires du processus**
  - Définition et mise en œuvre d'une méthode d'analyse prévisionnelle exhaustive qui vise à prouver que le logiciel de commande propage (ou ne propage pas) une faute aléatoire du processus commandé

- **Hypothèse**
  - Le logiciel de commande ne comporte pas de fautes de conception en l'absence de fautes aléatoires.

Exemple illustratif : système d'essuyage



Exemple de comportement souhaité (pas de propagation de faute d'un composant matériel)

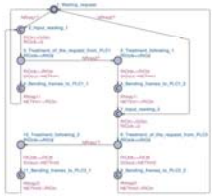
Même en présence d'une faute du capteur de position basse du balai (variable INFO_AREFIX toujours fausse)

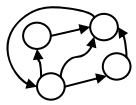
- lorsque le commodo est mis en position auto et y reste, et qu'il pleut, le balayage s'effectue ;
- lorsque le commodo est mis en position stop et y reste, le balayage finit toujours par s'arrêter.

Analyse exhaustive par vérification formelle

Principe du model-checking

Modèle formel d'un système (sous forme de structures de Kripke, d'automates temporisés, …)





AG(APB→AF ~horn)

AG(~d1→AF ~lig)

Propriétés à prouver (expressions dans une logique temporelle, telle que LTL, CTL)

Model checker

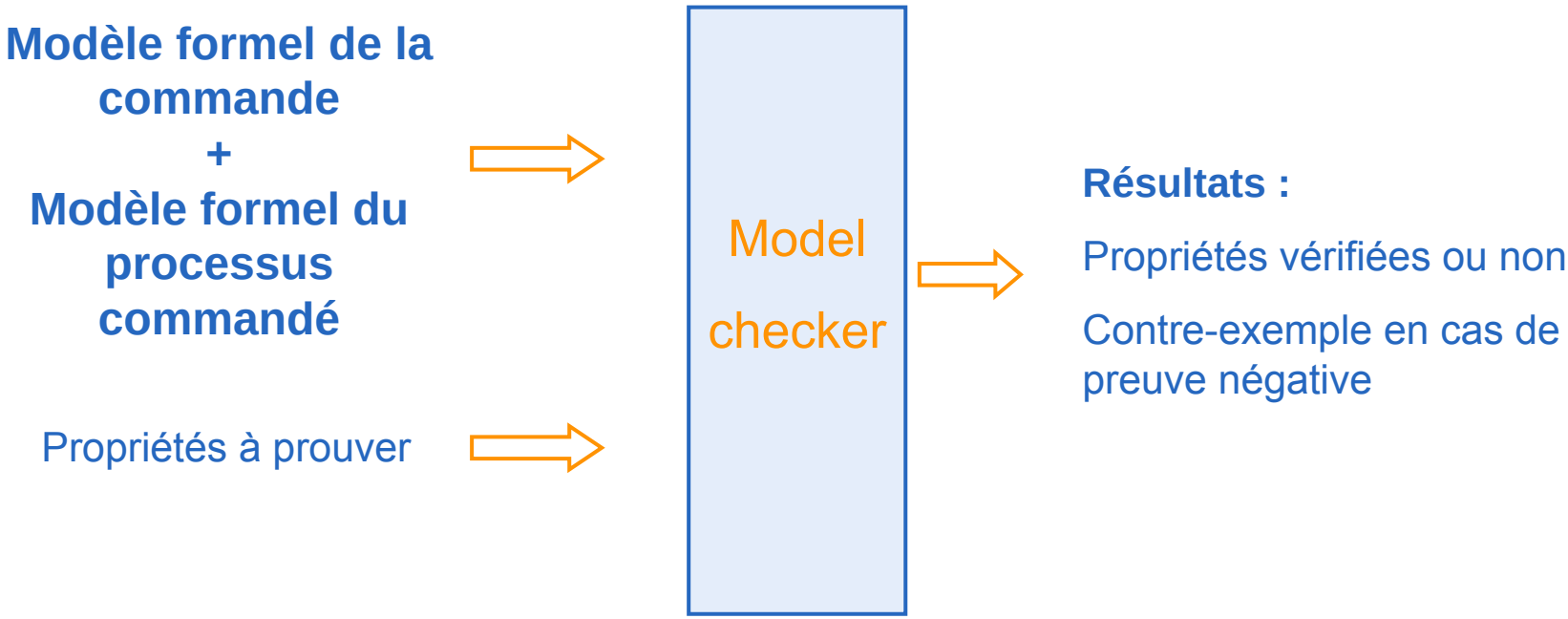
Résultats :

Propriétés prouvées ou non

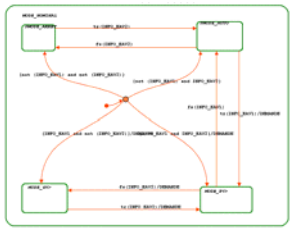
Contre-exemple en cas de preuve négative

Vérification formelle model-based [FRE 00]

- La preuve porte sur un modèle formel du SYSTEME BOUCLE (commande couplée au processus).



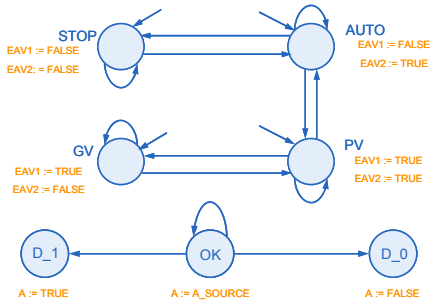
Méthode d'analyse proposée



Modèle de la commande
(en state-charts, car issu d'un processus de développement industriel)

```
MODULE main
 VAR
  ..............
 DEFINE
  ......
 ASSIGN
  .......
```

Modèle formel de la commande (dans la syntaxe NuSMV)



Modèle formel du processus commandé sujet à fautes
(sous forme de Machines de Moore Non-déterministes (MMN))

```
MODULE main

-- sources physiques
 VAR
   pos_commodo : {stop,auto,pv,gv} ;
   meteo : {pluie,soleil} ;
   pos_balai : {bas,montee,haut,descente} ;

-- Defaillances
   etat_capt_pluie  : {OK,D_1,D_0} ;
   etat_capt_arefix : {OK,D_1,D_0} ;
   etat_capt_eav1   : {OK,D_1,D_0} ;
   etat_capt_eav2   : {OK,D_1,D_0} ;

-- Sources internes
 DEFINE
   capt_eav1 := (pos_commodo = pv | pos_commodo = gv);
   capt_eav2 := (pos_commodo = auto | pos_commodo = pv);
   capt_arefix := pos_balai = bas ;

 VAR
   capt_pluie : boolean ;

 ASSIGN
   init(capt_pluie) := (meteo = pluie);
   next(capt_pluie) := next(meteo = pluie);
```

Modèle formel du processus (dans la syntaxe NuSMV)

En présence d'une faute du capteur, …

Définition des comportements souhaités

AG (p1 -> EF p2)

Propriétés formelles

NuSMV (model-checker non temporisé)

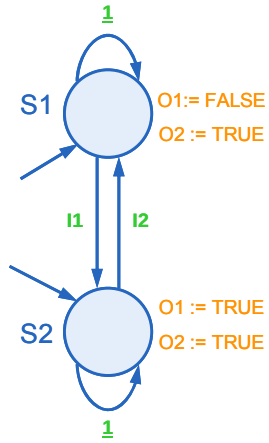
Résultats de preuve

```
-- specification AG (p1 -> EF p2)
is false
-- as demonstrated by the
following execution sequence
Trace Description: CTL
Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  pos_balai = bas
  ETAT_eav2 = OK
  eav2 = 1
  commodo = auto
  ETAT_info_pluie = OK
  info_pluie = 0
  …
```

Règles générales

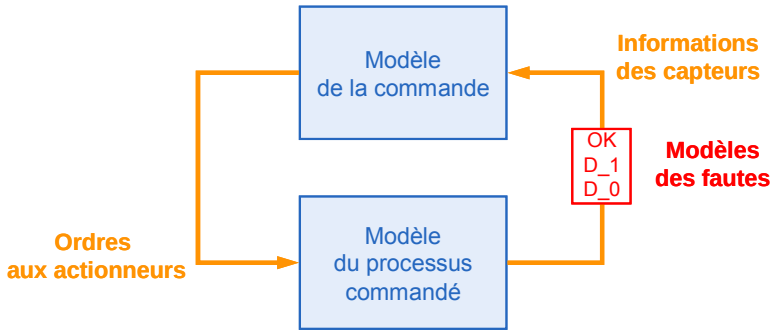
Formalisme retenu : Machines de Moore Non-déterministes (MMN)

- Σ_in : alphabet d'entrée, non vide
- Σ_out : alphabet de sortie, non vide
- S : ensemble, non vide, des états de la machine
- S_0 : ensemble, non vide, des états initiaux
- T : fonction de transition
- Λ : fonction d'assignation

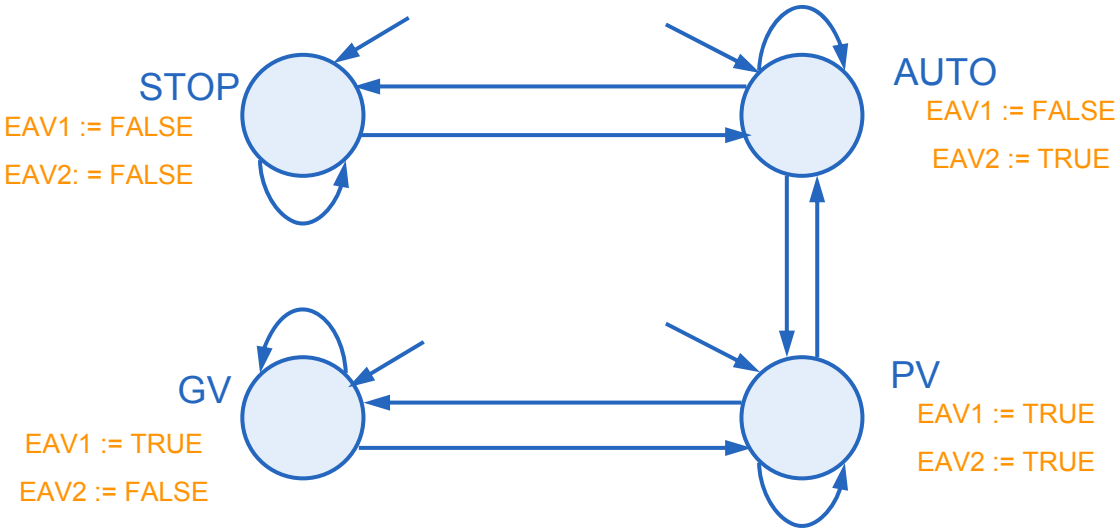


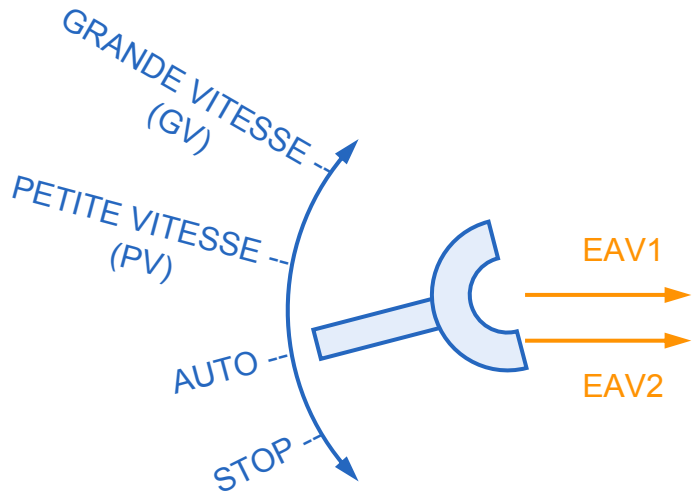
Remarque :  représente une transition franchissable sans condition particulière sur la valeur des éléments de l'alphabet d'entrée (1 peut être omis)

Modèles de fautes implantés entre le processus et la commande



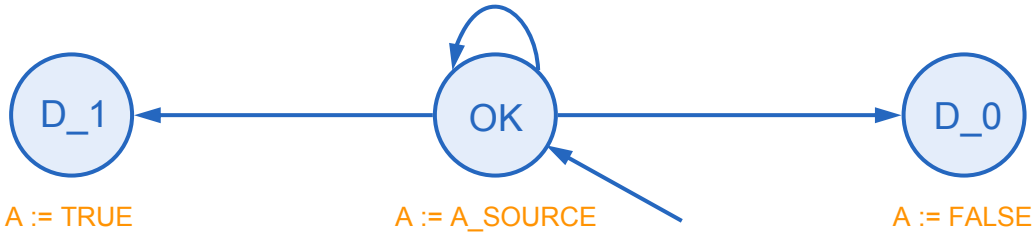
Exemple de modèle de composant : modèle du commodo

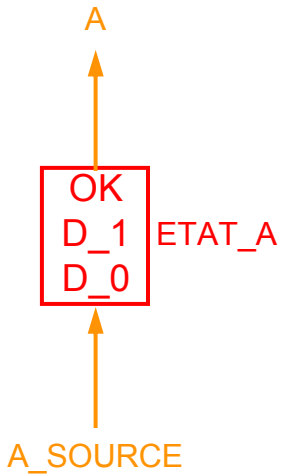




- S = S_0 = {STOP, AUTO, PV, GV}
- Σ_in = ∅
  (le conducteur peut modifier la position à tout instant)
- Σ_out = {EAV1, EAV2}

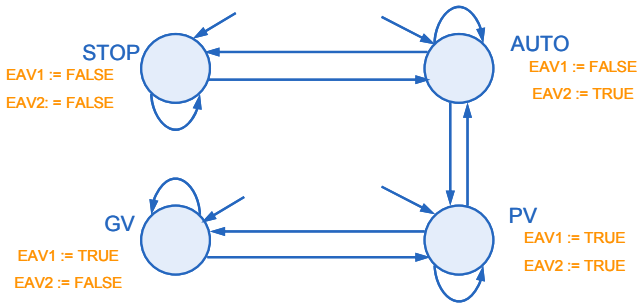
Exemple de modèle de fautes à trois états

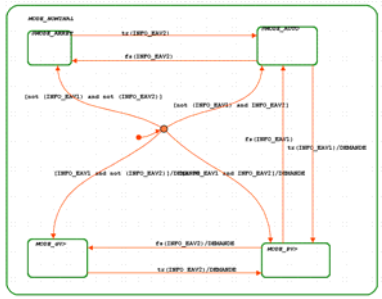


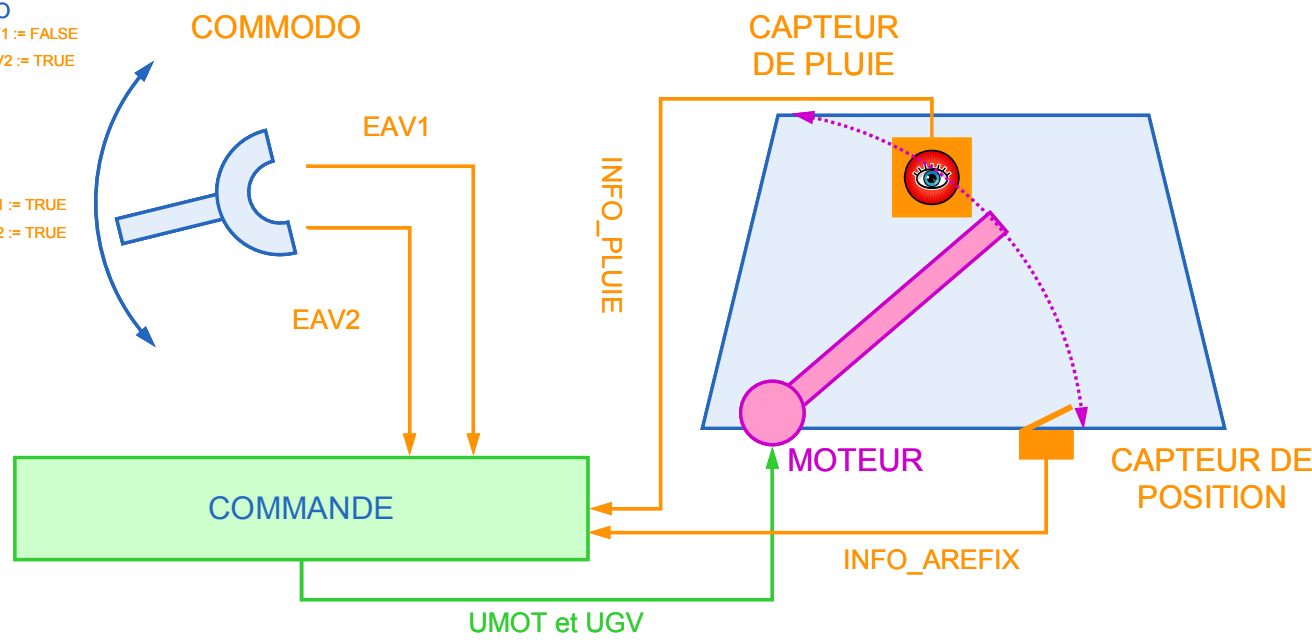


- S = {OK,D_1,D_0}
- S_0 = {OK}
- Σ_in = {A_SOURCE}
- Σ_out = {A}

Synchronisation avec le modèle de commande - 1

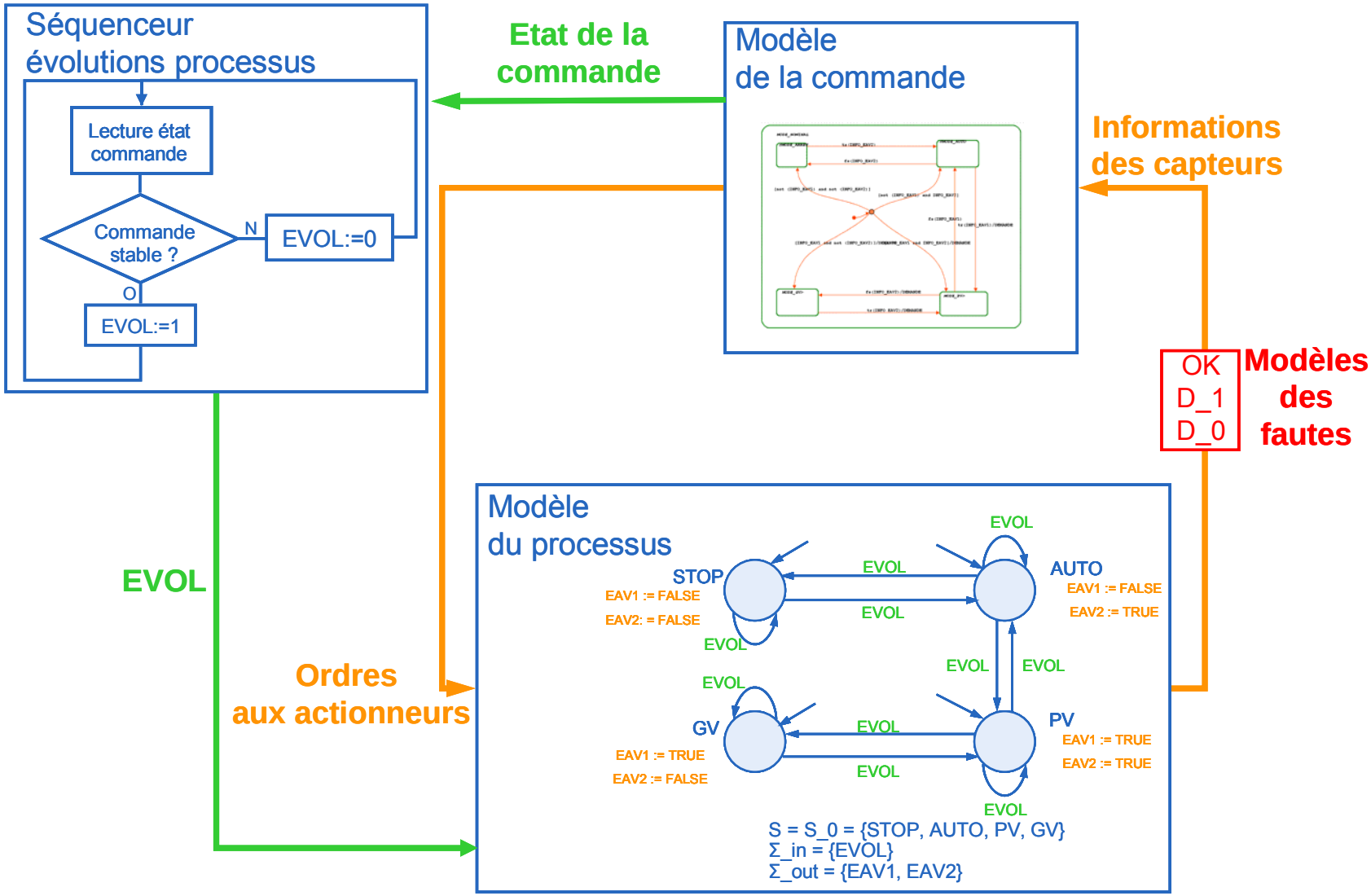




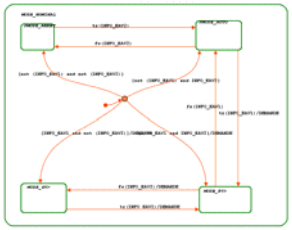


- **En réalité, la commande est beaucoup plus réactive que le processus.**
- **La modélisation de ces deux éléments dans un formalisme unique peut conduire à des états ou à des séquences d'états du système non significatifs.**
- **Il est nécessaire d'introduire un séquenceur des évolutions du processus.**

Synchronisation avec le modèle de la commande - 2



Principe général



Modèle informel de la commande

MODULE main
VAR
.....
DEFINE
.....
ASSIGN
.....

Modèle formel de la commande

Modèle formel du processus sujet à fautes

AG (p1 -> EF p2)

Propriétés formelles

NuSMV

(model-checker non temporisé)

La commande ne propage pas la faute.

La commande propage la faute.

-- specification AG (p1 -> EF p2)
is false
-- as demonstrated by the
following execution sequence
Trace Description: CTL
Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
pos_balai = bas
ETAT_eav2 = OK
eav2 = 1
commodo = auto
ETAT_info_pluie = OK
info_pluie = 0
...

Analyse de la trace et correction du modèle de commande

Exemple de propriété vérifiée avec le modèle de commande initial

- Il y a balayage lorsque le commodo est en position auto et y reste, et que le pluie commence à tomber, même en présence d'une faute du capteur de position basse du balai.

- **AG !EG (!UMOT & !EAV1 & EAV2 & INFO_PLUIE)**
  - « Il n'existe aucun état duquel parte un chemin tel que, pour tous les états, la proposition logique !UMOT & !EAV1 & EAV2 & INFO_PLUIE (le moteur n'est pas alimenté bien que le commodo soit en position AUTO et qu'il pleuve) est vérifiée. »

- La preuve positive montre que la faute du capteur n'affecte pas cette fonction (commande insensible à la faute).

Exemple de propriété non vérifiée avec le modèle de commande initial

- Lorsque le commodo est mis en position stop et y reste, le balayage finit toujours par s'arrêter, même en présence d'une faute du capteur de position basse du balai.

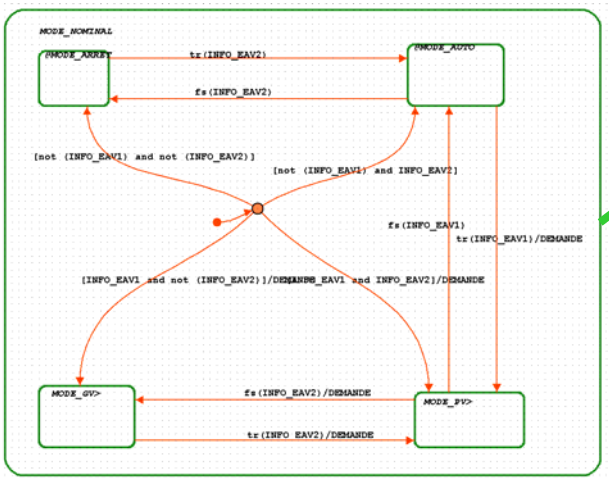
- **AG !EG (UMOT & !EAV1 & !EAV2)**
  - « Il n'existe aucun état duquel parte un chemin tel que, pour tous les états, la proposition logique UMOT & !EAV1 & !EAV2 (le moteur est alimenté bien que le commodo soit en position STOP) est vérifiée. »

- L'analyse d'un contre exemple montre que, si le capteur est défaillant bloqué à faux (non détection de la position basse du balai), la commande n'arrête jamais le balayage.

Exemple de correction du modèle de commande

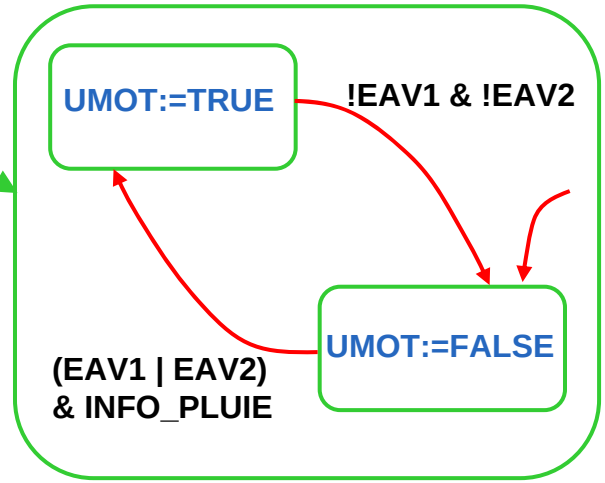
- La commande a été modifiée, en ajoutant une détection de la faute du capteur de position basse (via une temporisation) qui provoque un passage en mode dégradé arrêtant le balayage.
- Cette commande modifiée ne propage plus la faute.

NORMAL



UMOT & Tempo_STOP

DEGRADE



Tempo_STOP : variable booléenne vraie s'il s'est écoulé plus de n unités de temps depuis l'activation de l'état STOP du modèle de commodo

Conclusions et perspectives

- Intérêt des méthodes de vérification formelle pour l'analyse prévisionnelle de sûreté des systèmes embarqués discrets (approche model-based safety analysis)
- Modélisation du processus prenant en compte la différence de dynamique entre commande et processus
- Etude des possibilités de passage à l'échelle
- Prise en compte du temps physique (model-checking temporisé)

MERCI DE VOTRE ATTENTION