Méthodes Formelles dans l'Ingénierie du Logiciel

Alain Griffault
LaBRI/MVTsi
Alain.Griffault@labri.u-bordeaux.fr

ENSERB
3 ème année
Filière Informatique
Option Génie Logiciel

Bordeaux
27 janvier & 3 février 1998

PLAN

Problématique de l'équipe MVTsi

L'équipe MVTsi

- André Arnold
- Didier Bégay
- Richard Castanet
- Marc-Michel Corsini
- Jean-Michel Couvreur
- Paul Crubillé
- Anne Dicky
- Jérôme Dormoy
- Patrick Félix
- Alain Griffault
- Patrice Laurençot
- Maria Nikolskaia
- Gérald Point
- Antoine Rauzy
- Philippe Thomas

et aussi

- Paul Crubillé

Le cadre formel

Le modèle Arnold-Nivat

Un modèle synchrone pour décrire des processus communicants.

- Les systèmes de transition.
- Les systèmes de synchronisation.
- Le produit de synchronisation

La logique de Dicky

Elle comprend :

- Des constantes `{}` et `*`.
- Des fonctions élémentaires `src, tgt, rsrc, rtgt`.
- Des opérateurs ensembliste `\/, /\` et `-`.
- Les fonctions point-fixe avec un critère de monotonicité.

L'outil Mec

Mec, développé par l'équipe MVTsi du LaBRI, commercialisé par le CNRS, est un model-checker qui implémente :

- Le modèle Arnold-Nivat.
- La logique de Dicky.
- Le calcul de fonctions point-fixe définies par l'utilisateur.
- Quelques fonctions spécifiques (loop, trace, projections).
- Des fonctions d'éditions et d'archivages des résultats.
- Une représentation explicite et implicite en Arbres Partagés.

Un exemple

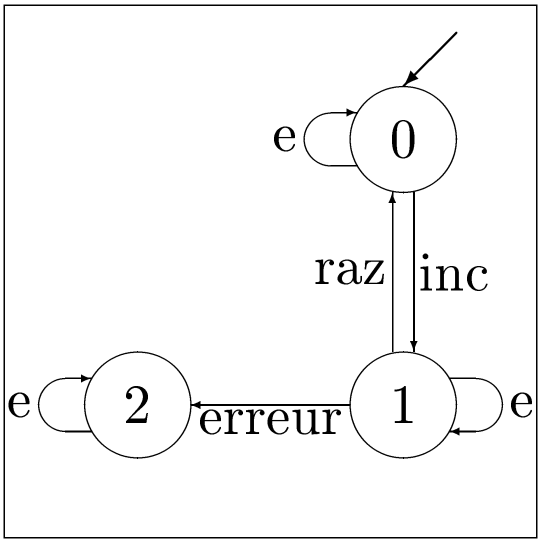

Figure 1: ST variable

synchronisation	
variable	variable
e	inc
inc	raz
erreur	e
e	erreur
raz	raz

```
dead := * - src(*);
boucle := loop(*,*);

function inevitable(Y:trans ; X:state)
    return Z:state;
begin
Z=X∪(src(Y∩rtgt(Z))-src(Y∩rtgt(*-Z)))
end.

deadlock := inevitable(*,dead);
```

Un exemple (suite)

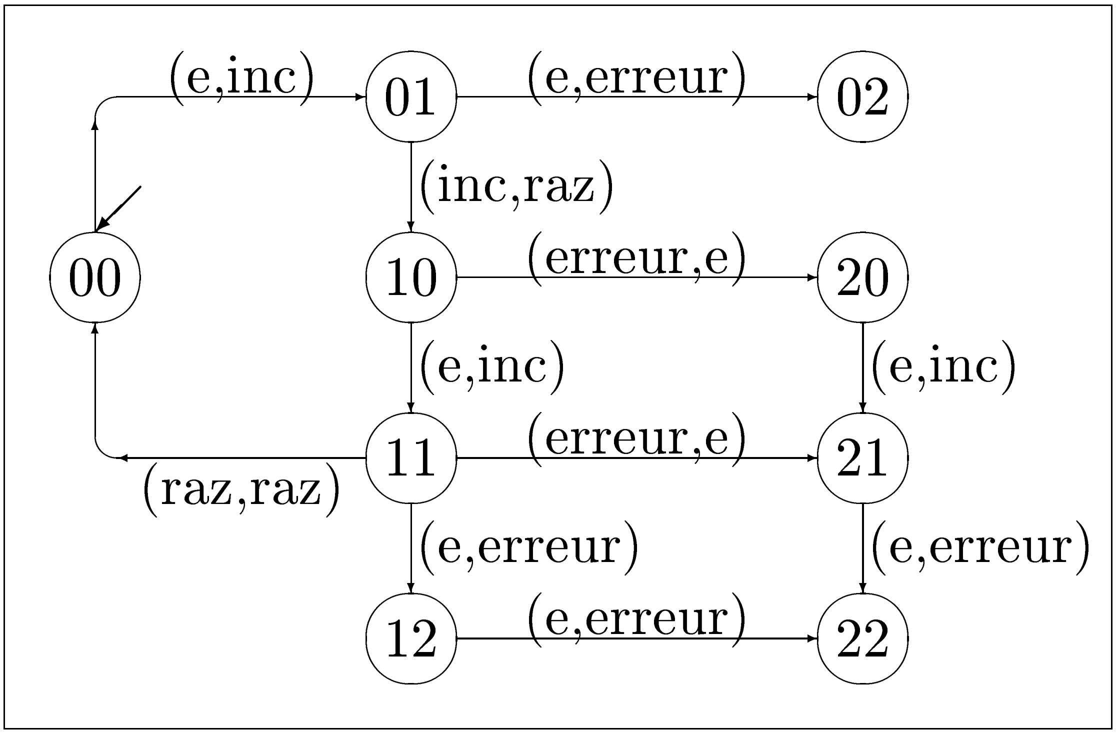

Figure 2: Le produit synchronisé

dead	02 , 22
boucle	00-01 , 01-10 , 10-11 , 11-00
deadlock	02 , 12 , 20 , 21 , 22

10 ans avec MEC

Références

- Académiques :
  - Des algorithmes d'exclusion mutuelle.
  - Des algorithmes d'élections.
  - Le protocole du bit alterné.
  - Le protocole de Stenning.
  - Le jeu de Nim.
  - Le Tic-Tac-Toe.
- Industrielles :
  - Schlumberger.
  - C.N.E.T.
  - I.S.D.F.
  - ELF Aquitaine.
  - Schneider (Merlin Gérin).
  - Marben.
  - SNECMA.

Mec et le cycle de vie.

- Description informelle.

  $\downarrow$ OUI
- Spécification formelle.

  $\downarrow$ oui
- Prototypage.

  $\downarrow$ NON
- Conception détaillée.

  $\downarrow$ NON
- Implémentation.
- Test. OUI si ...
- Maintenance. OUI si ...
- Correction de "bug". OUI/NON

Quelques définitions

modélisation L'activité consiste à définir un modèle abstrait.

validation L'activité consiste à *montrer* que le modèle abstrait est un bon candidat pour représenter le système réel.

vérification L'activité consiste à *prouver* que le modèle abstrait *donc certainement le système réel* vérifie les spécifications.

Quelques remarques

modélisation une étape informelle.

validation une étape informelle.

vérification une étape formelle.

Bilan Académique/Industrielle

	Exemples Académiques	Exemples Industriels
Description	algorithmes non ambigües	textes diagrammes ambigües incompletes
Modélisation	Facile	Très difficile
Validation	Inutile	Très difficile Incomplete
Vérification	Très difficile	Facile
Solutions classiques		
Modélisation		Méthodes
Validation		Tests Simulation
Vérification	Une autre logique	
Solutions avec Mec		
Modélisation		hypothèses
Validation		Propriétés
Vérification		Propriétés

Méthodologie

Les différentes visions du problème

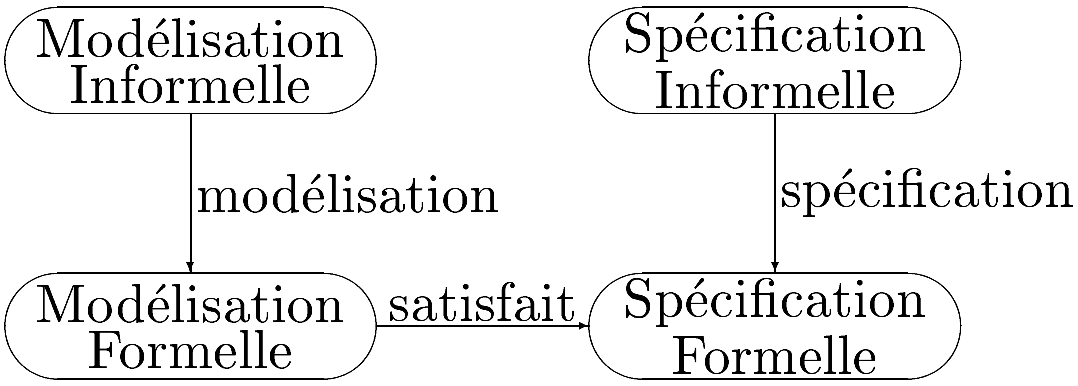

Figure 3: Vision 1

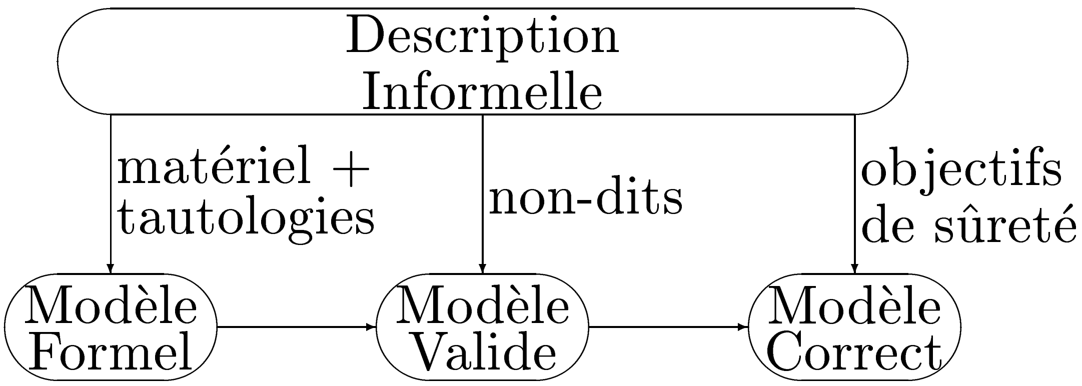

Figure 4: Vision 2

Processus de travail

1. Concevoir une premier modèle avec beaucoup d'abstractions.
2. Valider le modèle.
3. Itérer 1 et 2 en augmentant la liste des hypothèses, jusqu'à ce que la validation soit satisfaisante.
4. Vérifier le modèle
5. Itérer 1, 2, 3 et 4 en augmentant la liste des hypothèses, jusqu'à ce que la vérification soit satisfaisante.
6. Itérer 1, 2, 3, 4 et 5 en :
   - raffinant le modèle,
   - augmentant la liste des propriétés vérifiées,

   jusqu'à ce que l'une des conditions suivantes soit vraie :
   - Le temps imparti pour la spécification est terminé.
   - Le modèle est une implémentation.
   - Le système global est trop important pour la machine utilisée.

Validation du modèle

Des Validations syntaxiques

- Tous les états décrits sont-ils accessibles ?
- Toutes les transitions décrites sont-elles utilisées ?
- Tous les vecteurs de synchronisations décrits sont-il utilisés ?

Les propriétés usuelles

1. le blocage.
2. le blocage généralisé.
3. la possibilité de ne pas revenir dans l'état initial.
4. l'impossibilité de revenir dans l'état initial.
5. le respect des événements incontrôlables.
6. le respect des événements impossibles.

Vérification du modèle

Problème de protocoles

- le protocole est il correct avec un médium parfait ?
- le protocole resiste-il aux pertes, duplications, déséquencements et erreurs de transmissions ?

Problème de circuits logiques Deux circuits sont-ils équivalents ?

Problème d'exclusion mutuelle

- l'exclusion est-elle respectée ?
- l'absence de famine est-elle verifiée ?
- l'équité est-elle verifiée ?

Problème de jeux Existance de stratégies pour des jeux à plusieurs joueurs ?

Problème de contrôle-commande

- une commande peut-elle être sans effet ?
- une commande peut-elle engendrée des effets indésirées ?
- la réponse à une commande peut-elle être retardée indéfiniment ?

Le régulateur de vitesse

Cet exemple est tiré de l'article :
State-Based Model Checking of
Event-Driven System Requirements

Joanne M. Atlee and John Gannon

IEEE Transactions on Software Engineering

Vol 19, No 1, January 1993.

Description informelle

Spécifications des besoins La vitessse de croisière d'un véhicule est contrôlée par logiciel :

OFF : Le contact n'est pas mis.

INACTIVE : Le contact est mis, mais la vitesse n'est pas controlée.

CRUISE : Le contact est mis, et la vitesse est effectivement controlée.

OVERRIDE : Le contact est mis, la vitesse est controlée, mais le contrôle est momentanément suspendu.

L'état initial du contrôleur est **OFF**.

Les événements sont : soient des actions du conducteur, soient générés par des capteurs.

Le système est de type réactif, les changements d'états sont synchronisés aux fronts montants et descendants des signaux.

Description informelle (suite)

Les changements d'états du contrôleur

Etat	Cont.	Rou.	Vit. élevée	Fre.	Act.	Dés.	Rep.	Etat
OFF	toT							INACTIVE
INACTIVE	toF							OFF
INACTIVE	isT	isT		isF	toT			CRUISE
CRUISE	toF							OFF
CRUISE		toF						INACTIVE
CRUISE			toT					INACTIVE
CRUISE				toT				OVERRIDE
CRUISE						toT		OVERRIDE
OVERRIDE	toF							OFF
OVERRIDE		toF						INACTIVE
OVERRIDE	isT	isT		isF	toT			CRUISE
OVERRIDE	isT	isT		isF			toT	CRUISE

Description informelle (fin)

Spécifications de sûreté Les propriétés suivantes doivent être prouvées :

1. Lorsque le contrôleur est dans l'état **OFF**, le contact n'est pas mis.
2. Lorsque le contrôleur n'est pas dans l'état **OFF**, le contact est mis.
3. Lorsque le contrôleur est dans l'état **INACTIVE**, le contact est mis et soit la voiture est arrêtée, soit le contrôleur n'est pas activé.
4. Lorsque le contrôleur est dans l'état **CRUISE**, le contact est mis, la voiture roule, et le conducteur n'appuye pas sur la pédale de frein.
5. Lorsque le contrôleur est dans l'état **OVERRIDE**, le contact est mis et la voiture roule.

Résultats parus dans l'article

La traduction en CTL est aisée.

Les propriétés de sûreté	Observation
propriété 1	fausse
propriété 2	fausse
propriété 3	fausse
propriété 4	fausse
propriété 5	fausse

L'objectif

La méthode permet-elle de construire un modèle valide qui satisfait les spécifications de sûreté ?

Les événements simultanés

1. Ils sont traités : Dans ce cas, il est nécessaire de spécifier pour chacun des sous ensembles d'événements possibles simultanément, l'effet produit.
2. Ils sont séquentialisés : Cela signifie que le logiciel ne traitera qu'un seul événement à la fois. Différéntes possibilités :
   - 2.1. Chaque événement génère un message qui est mis dans un file d'attente unique. Deux événements simultanés seront placés dans la file dans un ordre quelconque.
   - 2.2. Chaque événement génère un message qui est mis dans sa file d'attente. Le logiciel scrute les files d'attente et choisi les événements en fonction de priorités.

Le graphe du cas 1 contient celui du cas 2.1 qui contient lui-même celui du cas 2.2.

Hypothèse 1: cas 2.1

De l'informel au formel

Les composants du système global.

Des signaux : liés aux capteurs :

- Un indique si la voiture avance.
- Un indique si la vitesse est trop élevée.

liés aux actions du conducteur :

- Un indique si le contact est mis.
- Un indique si la pédale de freins est appuyée.
- Un indique lorsque le bouton *activate* est appuyé.
- Un indique lorsque le bouton *desactivate* est appuyé.
- Un indique lorsque le bouton *reprise* est appuyé.

Un contrôleur : change le mode du régulateur.

Un conducteur: il représente l'environnement.

Le premier modèle

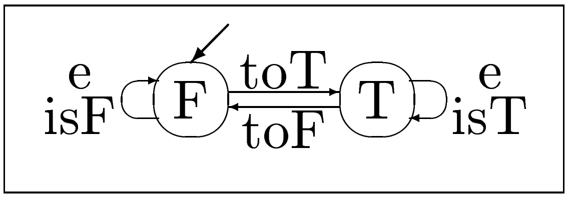

Figure 5: Un signal

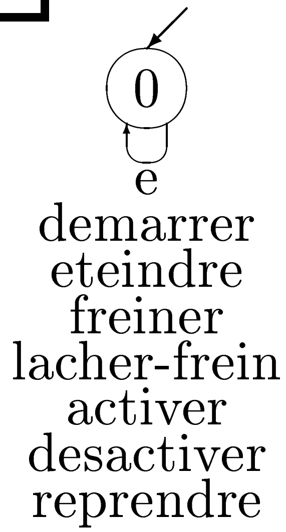

Figure 6: conducteur

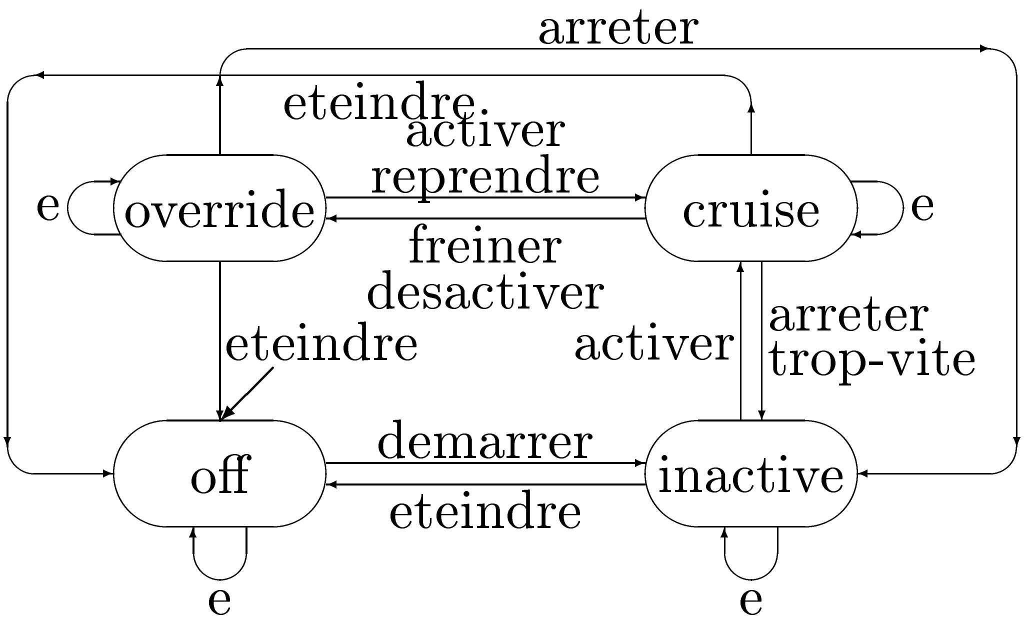

Figure 7: Première version du contrôleur

Le système de synchronisation

conducteur	sig. con.	sig. rou.	sig. vite	sig. fre.	sig. act.	sig. des.	sig. rep.	controleur
demarrer	toT	e	e	e	e	e	e	demarrer
eteindre	toF	e	e	e	e	e	e	eteindre
freiner	e	e	e	toT	e	e	e	freiner
lacher-frein	e	e	e	toF	e	e	e	e
activer	isT	isT	e	isF	toT	e	e	activer
desactiver	e	e	e	e	e	toT	e	desactiver
reprendre	isT	isT	e	isF	e	e	toT	reprendre
e	e	toF	e	e	e	e	e	arreter
e	e	e	toT	e	e	e	e	trop-vite

Validation du premier modèle

Les marques d'états		Observation
total	2	Semble petit
initial	1	Normal
dead	0	bon
deadlock	0	bon
controlable(demarrer)	1	KO
controlable(eteindre)	1	KO
controlable(freiner)	2	KO
controlable(lacher frein)	2	KO
controlable(activer)	2	KO
controlable(desactiver)	2	KO
controlable(reprendre)	2	KO
controlable(roule)	2	KO
controlable(arrete)	0	bon
controlable(trop vite)	2	KO
controlable(vitesse normale)	0	bon

Les marques de transitions		Observation
total	2	Semble petit
boucle	2	bon
boucleinitial	2	bon

Quelques problèmes

incomplétude : Beaucoup d'implicite dans l'informel.

contraintes physiques : Difficiles car non dites, mais elle doivent être décrites pour que le modèle soit réaliste.

...

un exemple Faire l'hypothèse que les boutons n'émettent des signaux que lorsque le contact est mis ne nous semble pas restrictif.

1. Cette contrainte est une contrainte constructeur qui doit prévoir un mécanisme physique qui empêche l'accès aux boutons lorsque le contact n'est pas mis.
2. Cette contrainte est une contrainte constructeur qui doit prévoir un mécanisme électrique qui empêche l'envoi de signaux à partir des boutons lorsque le contact n'est pas mis.

3. Cette contrainte est une contrainte logicielle pour le développeur qui doit prévoir un mécanisme pour ne pas générer de message lors de reception des signaux en provenance des boutons lorsque le contact n'est pas mis.

D'une manière générale, cet exemple montre comment doit être introduit une contrainte supplémenatire dans une modélisation formellle.

1. Décrire en langage informelle la nouvelle contrainte en la justifiant si possible.
2. Décrire formellement la contrainte.
3. Indiquer s'il s'agit d'une contrainte matérielle ou logicielle, et qui devra la traiter.

…Le neuvième modèle

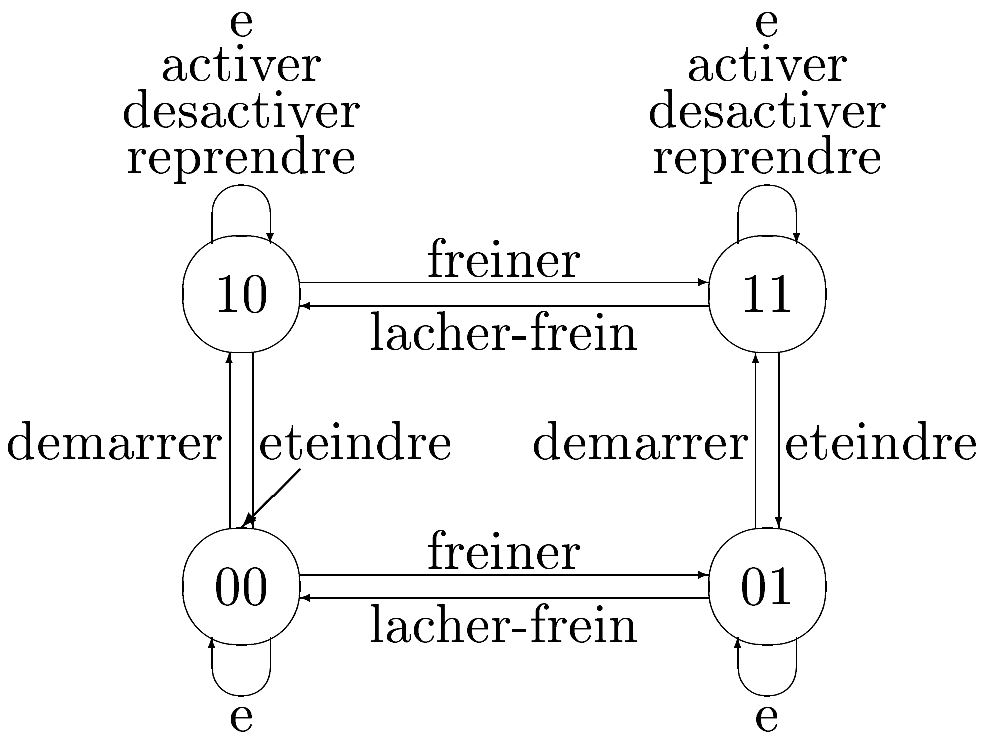

Figure 8: Troisième version du conducteur

…Le neuvième modèle (suite)

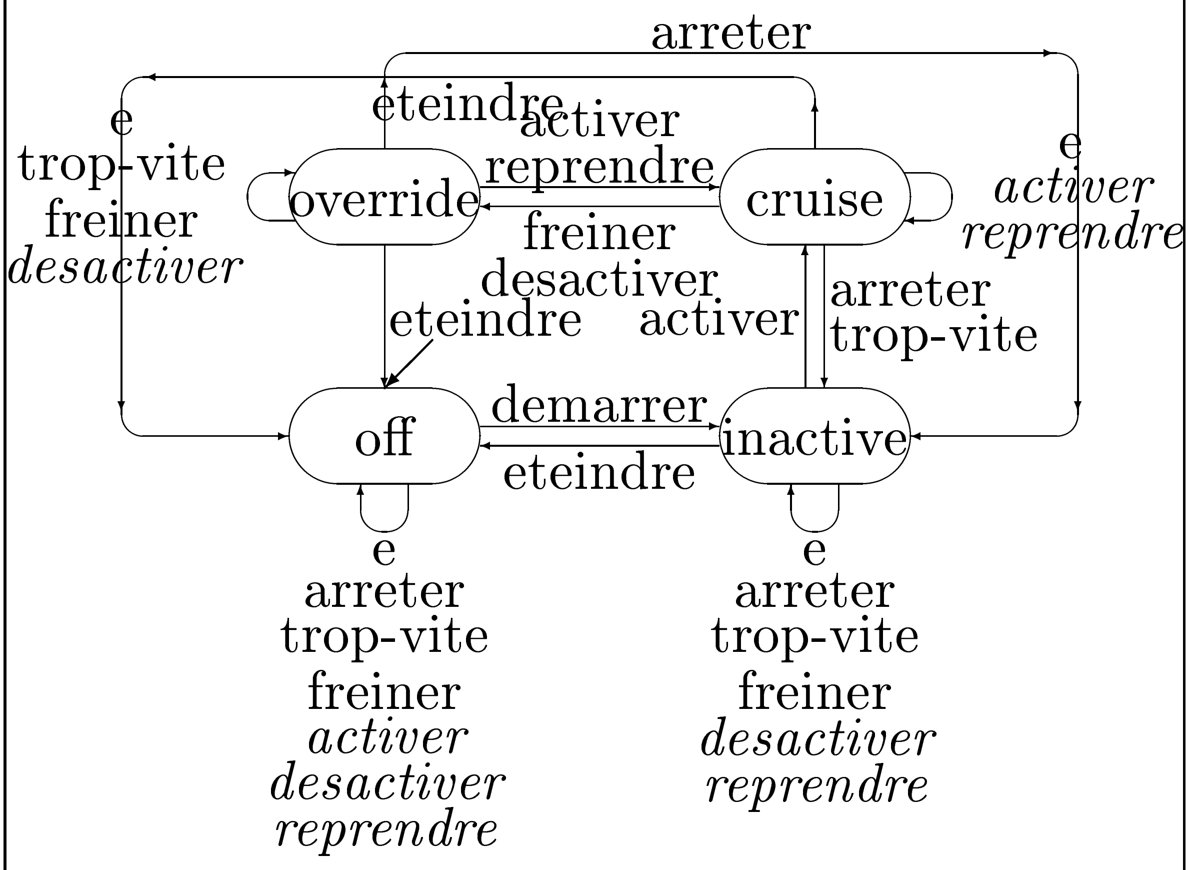

Figure 9: Troisième version du contrôleur

Validation du neuvième modèle

Les marques d'états		Observation
total	56	correct
initial	1	Normal
dead	0	bon
deadlock	0	bon
controlable(demarrer)	0	bon
controlable(eteindre)	0	bon
controlable(freiner)	0	bon
controlable(lacher frein)	0	bon
controlable(activer)	0	bon
controlable(desactiver)	0	bon
controlable(reprendre)	0	bon
controlable(roule)	0	bon
controlable(arrete)	0	bon
controlable(trop vite)	0	bon
controlable(vitesse normale)	0	bon

Les marques de transitions		Observation
total	400	correct
boucle	400	bon
boucleinitial	400	bon

Vérification du modèle

off ⇒ contact non mis

```
off˜:= !state[9]='off';
contact_mis := !state[2]='T';
propriete1 := off - (*-contact_mis);
```

non off ⇒ contact mis

```
propriete2 := (*-off) - contact_mis;
```

inactive ⇒ contact mis et (voiture arrêtee ou contrôleur non activé)

```
inactive := !state[9]='inactive';
roule := !state[3]='T';
non_active := !state[6]='F';
propriete3 := inactive -
(contact_mis/\((*-roule)\/non_active));
```

cruise ⇒ contact mis et (voiture roule et freins non appuyés)

```
cruise := !state[9]='cruise';
frein_appuye := !state[5]='T';
propriete4 := cruise -
(contact_mis/\(roule/\(*frein_appuye)));
```

override ⇒ contact mis et voiture roule

```
propriete5:=override-(contact_mis/\roule);.
```

Résultats

Les propriétés de sûreté		Observation
propriété1	0	bon
propriété2	0	bon
propriété3	4	KO
propriété4	0	bon
propriété5	0	bon

Plusieurs versions plus tard

- Utilisation de l'information sur la vitesse.
- Elimination des états incohérents.
```
ScienceFiction :=
     !state[3]='F' /\ !state[4]='T'
```
- Arrêt de l'activation en cas d'arrêt du véhicule.
- Arrêt de l'activation en cas de vitesse trop élevée.
- ...

Résultats de validation

Les marques d'états		Observation
total	40	correct
initial	1	Normal
incoherent	0	bon
dead	0	bon
deadlock	0	bon
controlable(demarrer)	0	bon
controlable(eteindre)	0	bon
controlable(freiner)	0	bon
controlable(lacher frein)	0	bon
controlable(activer)	0	bon
controlable(desactiver)	0	bon
controlable(reprendre)	0	bon
controlable(roule)	0	bon
controlable(arrete)	0	bon
controlable(trop vite)	0	bon
controlable(vitesse normale)	0	bon

Les marques de transitions		Observation
total	267	correct
boucle	267	bon
boucleinitial	267	bon

Résultats de vérification

Les propriétés de sûreté		Observation
propriété1	0	bon
propriété2	0	bon
propriété3	0	bon
propriété4	0	bon
propriété5	0	bon

Synthèse du modèle

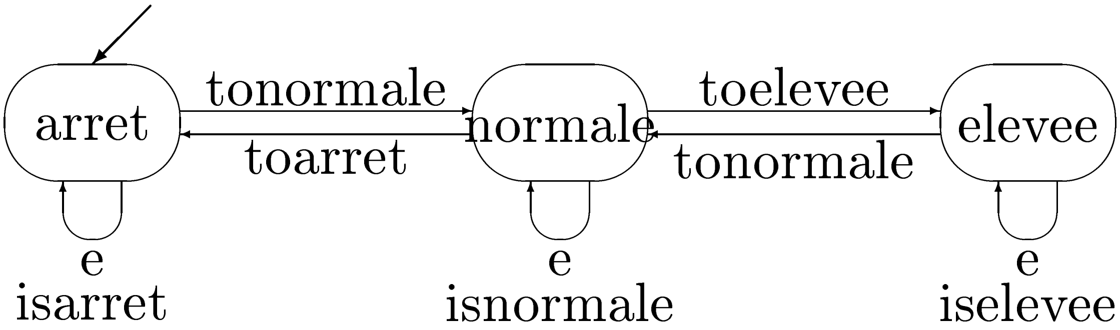

Figure 10: Un capteur vitesse

CONCLUSION

Les modèles

Les systèmes de transition et de synchronisation semblent un bon formalisme pour la description comportementale des systèmes complexes, sous réserve que les données manipulées peuvent par abstraction, être assimilées à des valeurs entières dont le domaine est réduit à peu de valeurs.

La méthodologie

La méthodologie proposée comme accompagnatrice de l'outil MEC semble :

- simple, (issue de l'enseignement et d'une confrontation à des exemples industriels)
- applicable dans de nombreux domaines : systèmes embarqués, contrôle-commande, protocoles, algorithmique distribuée.
- contenir suffisemment de garde-fous, pour éviter des conclusions hâtives sur un modèle qui ne serait pas réaliste.