

JFPDA'2010

# Synthèse de contrôleur simplement valide dans le cadre de la programmation par contraintes

Michel Lemaître  
Gérard Verfaillie  
Cédric Pralet  
Guillaume Infantès

ONERA/DCSD

Besançon, 3 Juin 2010

# Objectifs

- synthétiser un contrôleur d'un système à contrôler
- d'après une description déclarative (contraintes) du système et des exigences à satisfaire
- par une approche qui se veut pratique (économe en ressources de calcul).

# Objectifs

- synthétiser un contrôleur d'un système à contrôler
- d'après une description déclarative (contraintes) du système et des exigences à satisfaire
- par une approche qui se veut pratique (économe en ressources de calcul).

Sacrifice : approche non complète (peut échouer).

# Le besoin : informatique embarquée

- **petits calculateurs dédiés domestiques**

TV, téléphones, organiseurs, GPS, ...

contrôleur de machine à laver, réfrigérateur, chaudière ...

- **domaines industriels**

ascenseurs

automobile (ABS, allumage électronique ...)

trains (sécurité ...)

avionique : commandes de vol

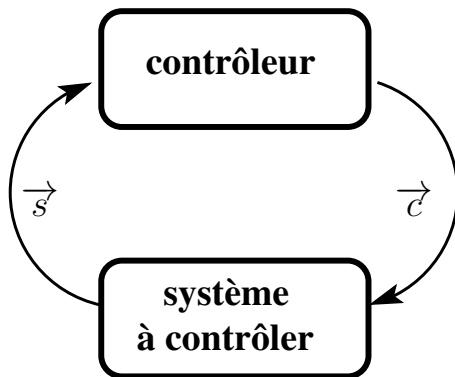
drones

satellites ...

# Le cadre : systèmes à événements discrets (SED)

- la dynamique du système est régie des **événements** et non pas simplement par des équations différentielles ...
- événements discrets = il adviennent de manière instantanée
- entre deux événements le système peut évoluer continûment

## Système avec contrôle discret, en boucle fermée



$\vec{s} = (s_1, s_2, \dots)$ ,  $s_i \in \mathcal{S}$  (états)

$\vec{c} = (c_1, c_2, \dots)$ ,  $c_i \in \mathcal{C}$  (commandes)

$\mathcal{S}$  et  $\mathcal{C}$  : domaines “combinatoires”

(produits cartésiens de domaines simples pour des attributs)

# Contrôleur réactif temps-réel

En informatique on est obligatoirement discret :

- ① pas de vrais réels : la représentation des nombres est bornée
- ② les calculateurs digitaux fonctionnent en cycles et chaque cycle prend un temps non nul (pendant lequel le dispositif continue à évoluer)

# Contrôleur réactif temps-réel

En informatique on est obligatoirement discret :

- ① pas de vrais réels : la représentation des nombres est bornée
- ② les calculateurs digitaux fonctionnent en cycles et chaque cycle prend un temps non nul (pendant lequel le dispositif continue à évoluer)

Un **contrôleur réactif temps-réel** doit réagir continuellement et rapidement à ses entrées par des sorties appropriées, à la vitesse imposée par le dispositif et son environnement

→ **nécessité d'un temps de réponse et d'une mémoire bornés**

# Le problème de la synthèse de contrôleur (informellement)

## Données :

- description  $T$  du système à contrôler
- description  $R$  (*requirements*) des propriétés qu'on exige d'être satisfaites par le système contrôlé ( contrôleur + système à contrôler )

# Le problème de la synthèse de contrôleur (informellement)

## Données :

- description  $T$  du système à contrôler
- description  $R$  (*requirements*) des propriétés qu'on exige d'être satisfaites par le système contrôlé ( contrôleur + système à contrôler )

**Solution** : s'il existe, un contrôleur (= une *politique*)  $\Pi : \mathcal{S} \rightarrow \mathcal{C}$  **valide** c'est-à-dire tel que, en tout état  $s$  atteignable, la commande  $\Pi(s)$

- est définie (**existence**)
- peut s'appliquer (compatible avec  $T$ ) (**applicabilité**)
- respecte les exigences  $R$  (**conformité**)

# Systèmes à contrôle « continu »

On se restreint dans cette étude  
aux systèmes qui n'ont pas d'état but à atteindre :  
**les exigences sont uniquement des propriétés de sûreté.**

Autrement dit : on ne traite pas ici un problème de planification.

La plupart des systèmes temps-réel s'inscrivent dans ce cadre.

## Description du système à contrôler : transitions $T$

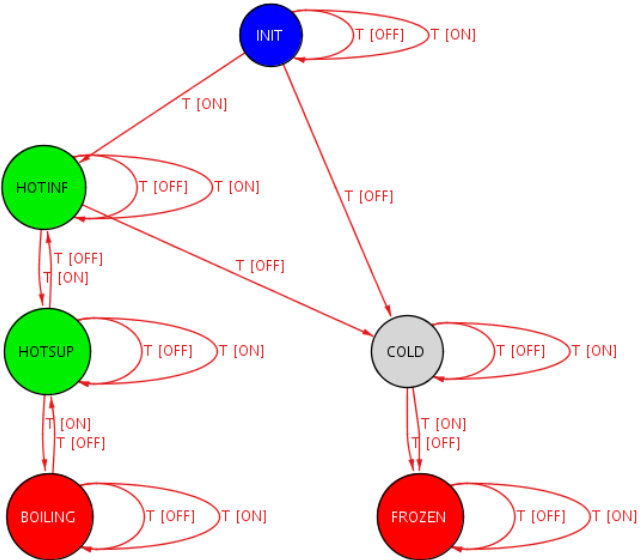
Le système à contrôler est décrit par ses transitions possibles  $T$  :

$$T : \mathcal{S} \times \mathcal{C} \times \mathcal{S} \rightarrow \mathbb{B}$$

$\forall s, c, s' : T(s, c, s') = \text{vrai ssi}$

une transition du système à contrôler est faisable  
à partir de  $s$  vers  $s'$  avec la commande  $c$

# Exemple – contrôler un chauffe-eau : transitions



## Description des exigences $R$ (*requirements*)

Le prédicat  $R$  décrit les exigences, c'est-à-dire le comportement souhaité du système contrôlé :

$$R : \mathcal{S} \times \mathcal{C} \times \mathcal{S} \rightarrow \mathbb{B}$$

$\forall s : R(s, c, s') = \text{vrai}$  ssi  $(s, c, s')$  est une transition qui respecte les exigences.

## Exemple – contrôler un chauffe-eau : exigences

$$\forall s : OK(s) = s \in \{\text{INIT}, \text{COLD}, \text{HOTINF}, \text{HOTSUP}\}$$
$$\forall s, c, s' : [R(s, c, s') = [OK(s) \implies OK(s')]]$$

## Un contrôleur ou *politique* $\Pi$

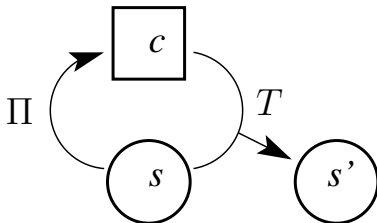
$$\Pi : \mathcal{S} \times \mathcal{C} \rightarrow \mathbb{B}$$

$\forall s, c : \Pi(s, c) = \text{vrai}$  ssi la commande  $c$  est possible lorsque l'état du système est  $s$  avec la politique  $\Pi$   
Noter la forme non déterministe.

## Un contrôleur ou *politique* $\Pi$

$$\Pi : \mathcal{S} \times \mathcal{C} \rightarrow \mathbb{B}$$

$\forall s, c : \Pi(s, c) = \text{vrai}$  ssi la commande  $c$  est possible lorsque l'état du système est  $s$  avec la politique  $\Pi$   
Noter la forme non déterministe.



# Validité totale ou simple d'une politique

(revoir la description informelle d'une politique valide)

# Validité totale ou simple d'une politique

(revoir la description informelle d'une politique valide)

**Validité totale** : atteignabilité à partir d'un état initial

Difficulté : propriété **globale**, qui doit être calculée explicitement (chemins dans un graphe qui dépend de  $\Pi$ ) : calculs lourds en général

# Validité totale ou simple d'une politique

(revoir la description informelle d'une politique valide)

**Validité totale** : atteignabilité à partir d'un état initial

Difficulté : propriété **globale**, qui doit être calculée explicitement (chemins dans un graphe qui dépend de  $\Pi$ ) : calculs lourds en général

**Validité simple** : l'atteignabilité est remplacée par une propriété **locale** plus faible (plus souvent vérifiée)

mais beaucoup plus facile à calculer.

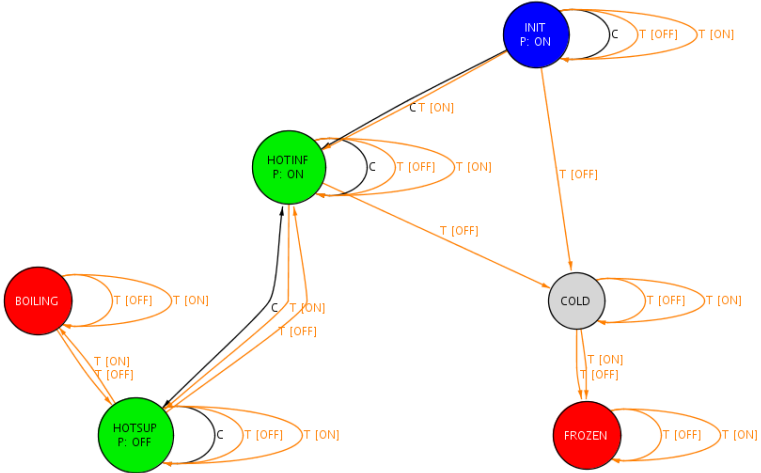
Ici l'atteignabilité est remplacée par la « **contrôlabilité** »  $T(s)$  :

$$T(s) =_{\text{def}} \exists c, s' : T(s, c, s')$$

Va bien avec le contrôle « continu ».

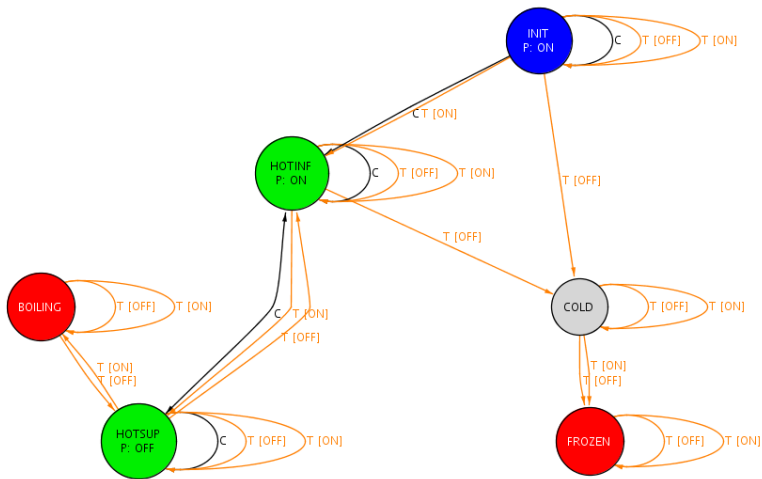
# Exemple : le contrôle du chauffe-eau (1)

Il existe une politique totalement valide (calculée avec Alloy) ...



# Exemple : le contrôle du chauffe-eau (1)

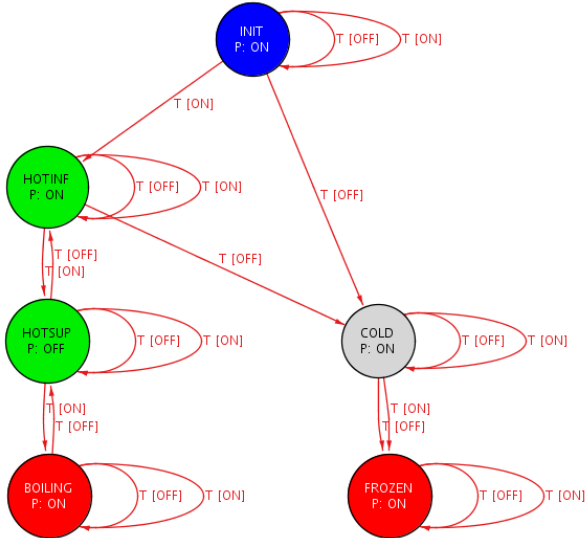
Il existe une politique totalement valide (calculée avec Alloy) ...



... mais pas de politique simplement valide ...

# Exemple : le contrôle du chauffe-eau (2)

... sauf si on interdit COLD :



# Calcul de politiques simplement valides

On caractérise de manière constructive et relativement simple les politiques simplement valides à partir de  $T$  et  $R$  (voir le papier).

## Calcul de politiques simplement valides

On caractérise de manière constructive et relativement simple les politiques simplement valides à partir de  $T$  et  $R$  (voir le papier).

On utilise le cadre de la **programmation par contraintes** pour

- exprimer  $T$  et  $R$  (la PPC est un cadre très expressif)
- calculer effectivement une politique simplement valide.

# Calcul de politiques simplement valides (suite)

Opérations :

- construction du CSP  $T(s, c, s')$ ,  
résolution (toutes solutions), projection sur  $(s, c)$   
→ couples  $(s, c)$  applicables

# Calcul de politiques simplement valides (suite)

Opérations :

- construction du CSP  $T(s, c, s')$ ,  
résolution (toutes solutions), projection sur  $(s, c)$   
→ couples  $(s, c)$  applicables
- construction du CSP  $T(s, c, s') \wedge \neg R(s, c, s')$ ,  
résolution (toutes solutions), projection sur  $(s, c)$   
→ couples  $(s, c)$  non conformes (interdits)

# Calcul de politiques simplement valides (suite)

Opérations :

- construction du CSP  $T(s, c, s')$ ,  
résolution (toutes solutions), projection sur  $(s, c)$   
→ couples  $(s, c)$  applicables
- construction du CSP  $T(s, c, s') \wedge \neg R(s, c, s')$ ,  
résolution (toutes solutions), projection sur  $(s, c)$   
→ couples  $(s, c)$  non conformes (interdits)
- applicables moins non conformes = politique potentielle

# Calcul de politiques simplement valides (suite)

Opérations :

- construction du CSP  $T(s, c, s')$ ,  
résolution (toutes solutions), projection sur  $(s, c)$   
→ couples  $(s, c)$  applicables
- construction du CSP  $T(s, c, s') \wedge \neg R(s, c, s')$ ,  
résolution (toutes solutions), projection sur  $(s, c)$   
→ couples  $(s, c)$  non conformes (interdits)
- applicables moins non conformes = politique potentielle
- reste à vérifier l'existence d'une commande pour tout état contrôlable.

## Calcul de politiques simplement valides (suite)

Opérations :

- construction du CSP  $T(s, c, s')$ ,  
résolution (toutes solutions), projection sur  $(s, c)$   
→ couples  $(s, c)$  applicables
- construction du CSP  $T(s, c, s') \wedge \neg R(s, c, s')$ ,  
résolution (toutes solutions), projection sur  $(s, c)$   
→ couples  $(s, c)$  non conformes (interdits)
- applicables moins non conformes = politique potentielle
- reste à vérifier l'existence d'une commande pour tout état contrôlable.

Calculs ensemblistes simples (avec des tables de hachage).

Donne la politique sous forme explicite (ensemble de tuples).

# Conclusion

Un cadre pour la synthèse de contrôle « continu »

# Conclusion

Un cadre pour la synthèse de contrôle « continu »

Une méthode constructive basée sur

- le remplacement de la propriété d'atteignabilité (globale) par une propriété de faisabilité (locale) plus facile à calculer

# Conclusion

Un cadre pour la synthèse de contrôle « continu »

Une méthode constructive basée sur

- le remplacement de la propriété d'atteignabilité (globale) par une propriété de faisabilité (locale) plus facile à calculer
- l'utilisation du cadre de la programmation par contraintes pour l'expression des données du problème et le calcul effectif du contrôle.

Deux exemples traités (chauffe-eau, robot exploreur).

# Perspectives

# Perspectives

- traiter d'autres exemples (en cours)

# Perspectives

- traiter d'autres exemples (en cours)
- comparer avec d'autres approches (*Simulate and Branch*, Anzu)

# Perspectives

- traiter d'autres exemples (en cours)
- comparer avec d'autres approches (*Simulate and Branch*, Anzu)
- résolution : autres structures de données pour les calculs relationnels (ex : BDDs), autres cadres ou méthodes (ex : SAT, recherche locale ...)

# Perspectives

- traiter d'autres exemples (en cours)
- comparer avec d'autres approches (*Simulate and Branch*, Anzu)
- résolution : autres structures de données pour les calculs relationnels (ex : BDDs), autres cadres ou méthodes (ex : SAT, recherche locale ...)
- autres affaiblissements que  $T(s)$

# Perspectives

- traiter d'autres exemples (en cours)
- comparer avec d'autres approches (*Simulate and Branch*, Anzu)
- résolution : autres structures de données pour les calculs relationnels (ex : BDDs), autres cadres ou méthodes (ex : SAT, recherche locale ...)
- autres affaiblissements que  $T(s)$
- comment réparer si échec ?

# Perspectives

- traiter d'autres exemples (en cours)
- comparer avec d'autres approches (*Simulate and Branch*, Anzu)
- résolution : autres structures de données pour les calculs relationnels (ex : BDDs), autres cadres ou méthodes (ex : SAT, recherche locale ...)
- autres affaiblissements que  $T(s)$
- comment réparer si échec ?
- exploitation en ligne.

# Perspectives

- traiter d'autres exemples (en cours)
- comparer avec d'autres approches (*Simulate and Branch*, Anzu)
- résolution : autres structures de données pour les calculs relationnels (ex : BDDs), autres cadres ou méthodes (ex : SAT, recherche locale ...)
- autres affaiblissements que  $T(s)$
- comment réparer si échec ?
- exploitation en ligne.

MERCI POUR VOTRE ATTENTION