T4: Validation formelle de propriétés quantitatives: approche par automates

Projet ANR VACSIM

30 septembre 2011

Plan

Plan

Analyse quantitative des automates temporisés

Défis scientifiques :

Évaluer les performances des systèmes temps-réel, e.g.
calcul de temps moyen, coût d'exécution, probabilité d'erreur, etc

- Probabilités
- Volume
- Topologie
- Fréquence

Objectifs

- Élaboration d'algorithmes de décision, de calcul quantitatif.
- Application à la validation
  - test, contrôle et diagnostic
  - validation sur des exemples via un prototype

Probabilités

Principe : randomisation des délais et des choix d'actions pour

- négliger les comportements marginaux potentiellement défaillants,
- quantifier le risque d'occurrences de certaines exécutions.

État de l'art : approximation de la probabilité d'une propriété pour les TA à une horloge

Pistes :

- extension à plusieurs horloges
- calculs de temps moyens
- représentation d'un testeur choisissant aléatoirement les délais (notion de complétude en probabilité du test)

Volume

Principe : associer un volume à un langage temporisé, pour comparer l'importance de langages temporisés

État de l'art : calcul de volumes (analytiquement et par discrétisation)

Pistes :

- calculs avec des techniques similaires à celles du cadre probabiliste
- comparaison de sur-approximations, estimation de l'importance d'un sous-langage invalidant une propriété (lien avec les probabilités)

Topologie

Principe : topologie sur les exécutions des automates temporisés pour

- mesurer l'écart à un comportement normal
- recouvrement par des boules de diamètre fixé

État de l'art : définition d'une topologie

Pistes :

- définir une distance induisant cette topologie
- lien avec les volumes
- définir un critère de couverture pour le test de TA

Fréquence

Principe : mesurer la proportion de temps passé dans une localité, pour estimer l'importance d'une erreur, la fréquence des états critiques

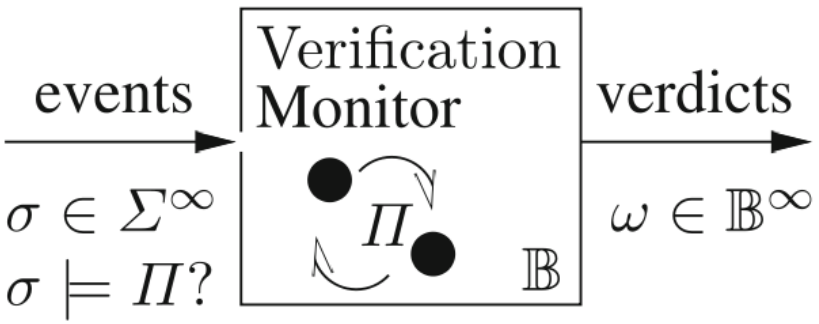
État de l'art : définition de la fréquence

Pistes :

- évaluation des défaillances comme la fréquence d'un ensemble d'états critiques
- algorithmes de calcul d'une fréquence
- méthodes pour décider si une fréquence est > 0, $= 1$ ou $< \epsilon$

Plan

Vérification à l'exécution : principes

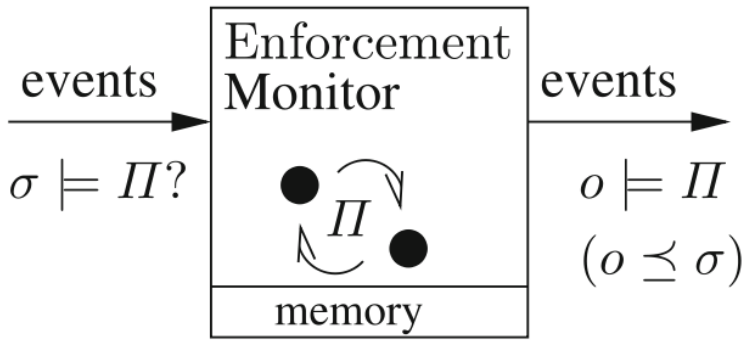


Monitoring

- S'assurer qu'une propriété Π est vérifiée **à l'exécution**
- On-line/off-line: analyse incrémentale/après mémorisation.
- Non-intrusif: exécution du programme non modifiée.
- Pas de spécification du système, seulement de la propriété.

$\rightarrow$ Moniteur d'exécution (oracle)

Enforcement à l'exécution : principes



Enforcement

- Extension du monitoring
- Moniteur + mémoire: retarder des entrées puis les relâcher quand la propriété est vérifiée
- Séquence de sorties = un préfixe de la séquence d'entrée

$\rightarrow$ Moniteur d'enforcement

Enforcement à l'exécution : problématique et résultats dans le cadre non temporisé

Propriétés à respecter :

- Correction : séquences de sorties doivent satisfaire Π
- Transparence : séquences d'entrées correctes préservées

Synthèse du moniteur d'enforcement

Génération d'une machine à états finie émettant les commandes **store**, **dump** et **halt**.

Classe des Propriétés

Caractérisation des ensembles de propriétés monitorables, enforçables et testables : classification *safety-progress*

Enforcement de système temporisé

Extension de la technique d'enforcement au temporisé?

Nouveaux éléments à prendre en compte :

- Les séquences d'entrées/sorties sont des mots temporisés : $\sigma = (\delta_1, a_1), (\delta_2, a_2), ...(\delta_n, a_n)$, $\delta_i \in \mathbb{R}_{\geq}$, $a_i \in \Sigma$
- La propriété Π est décrite par un automate temporisé ou une logique temporisée.

Synthèse d'un enforceur temporisé

- Classe de propriétés enforçables ?
- Nouvelle notion de transparence à définir
- Modèle de l'enforceur ?

Quelques pistes:

Classe des Propriétés

- Sous-classe de TLTL
- *safety-progress*: Safety & guarantee

Notions de transparence possibles

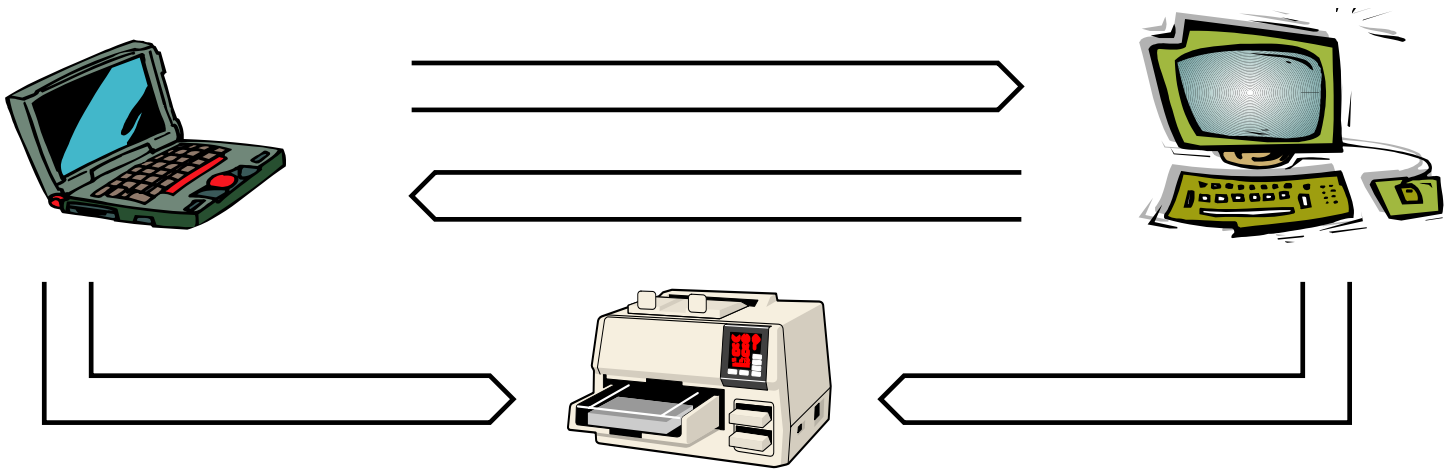
- L'enforceur ne peut qu'augmenter le délai entre événements consécutifs (retard), ou
- L'enforceur peut réduire le délai entre deux événements dans le but de rattraper un retard, ou
- L'enforceur relâche les événements au plus tôt, dès que la séquence appartient au langage.

Enforceur

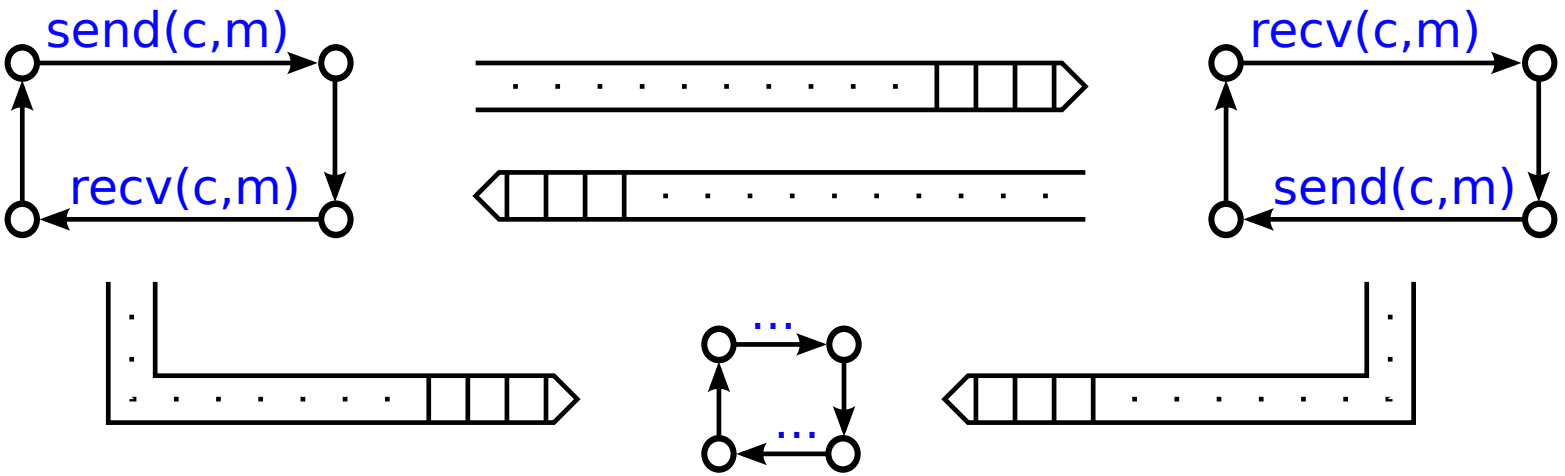
- Automate temporisé + mémoire
- commandes : **Store(t)**, **Dump**, **Halt**

Plan

Automates communicants

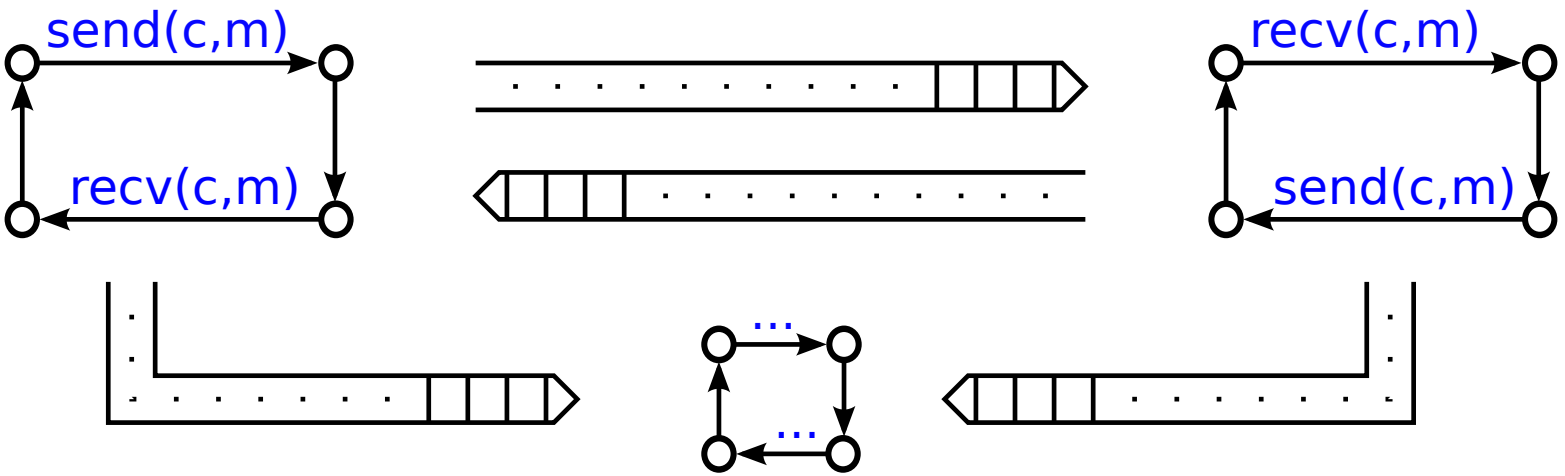


Automates communicants



- Processus locaux : automates finis
- Communication asynchrone, par envoi/réception de messages
  - canaux FIFO a priori non-bornés
- Vérification de propriétés de *sûreté* sur le modèle
  - Exemple : absence de blocage

Automates communicants temporisés



- Processus locaux : automates finis temporisés
  - horloges locales
- Communication asynchrone, par envoi/réception de messages
  - canaux FIFO a priori non-bornés
- Vérification de propriétés de *sûreté* sur le modèle
  - Exemple : absence de blocage

Vérification des automates communicants

Indécidabilité

La vérification algorithmique des automates communicants est impossible en général. [Brand&Zafiropulo'83]

Sous-classes décidables

- Restriction des contenus admissibles de canaux
- Pertes (incontrôlées) de messages
- Restrictions sur la topologie du réseau d'automates

Outils

- SPIN [Bell Labs] Canaux fiables, bornés (Promela)
- TReX [LIAFA] Canaux avec pertes, non-bornés, horloges
- McScM [LaBRI] Canaux fiables, non-bornés

Vérification des automates temporisés

Décidabilité

La vérification algorithmique des automates temporisés est réalisable en espace polynomial (optimal). [Alur&Dill'94]

Caractéristiques principales

- Horloges parfaites
- Réseau d'automates à communications synchrones
- Temps global (horloges synchrones)

Outils

- UPPAAL [Aalborg] Sûreté, sous-ensemble de CTL
- Kronos [Verimag] Sûreté, vivacité, TCTL
- RED [Taiwan] Symbolique, TCTL
- TChecker [LaBRI] Sûreté, vivacité, vérification à la volée

Vérification des automates communicants + temporisés

Pourquoi s'y intéresser ?

- Aspects temps-réel locaux dans un contexte asynchrone
- Modélisation fine d'algorithmes distributés, systèmes GALS
- Peu de résultats théoriques connus sur ce modèle

Objectifs

- Relâcher l'hypothèse usuelle d'évolution parfaitement synchrone des horloges
- Identifier la frontière entre décidabilité et indécidabilité
  - restrictions sur la topologie du réseau, le nombre d'horloges, la fiabilité des canaux, ...
- Complexité algorithmique de la vérification (lorsque décidable)