Problèmes indécidables dans la vérification

Rappel: Pour un TA A on sait tester si L(A)=vide. Méthode: on construit un automate (fini) de régions, on teste le vide.
C'est une clé pour des nombreux problèmes de vérification

Mais des nombreux problèmes de vérification des systèmes cyber-physiques sont indécidables:

A. Reachability (=non-empty langauage) pour des classes plus riches que TA, donc tous les problèmes de vérif pour ces classes!
B. Questions plus sophistiquées pour les TA, p.ex. L(A) est -il universel?

1 Stopwatch automata

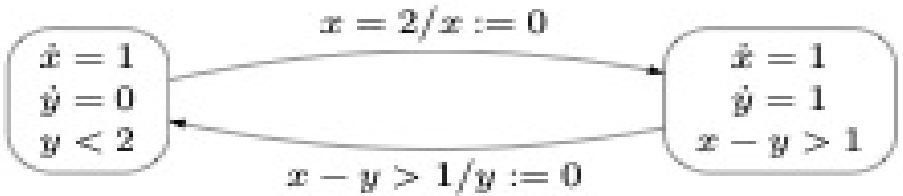


Figure 1: An example of an SA

Object of study. A stopwatch is a real variable which can have one of two dynamics: in some states it is $\dot{x} = 1$, in other states $\dot{x} = 0$. Intuitively it is a clock that can be stopped.

Stopwatch automata (SA) are hybrid automata where

- all continuous variables are stopwatches, there are finitely many of them;
- guards and invariants are boolean combinations of constraints $x < c$, $x \leq c$, $x - y < c$, $x - y \leq c$, where x, y are stopwatches, and c - integer constants;
- resets are as in timed automata: at a transition some stopwatches are reset to 0, while others stay unchanged.

On corrige exo 1 (stopwatch), on gère d'abord 1 compteur

1 cpt, codage

$C = n$ — $x - y = n$

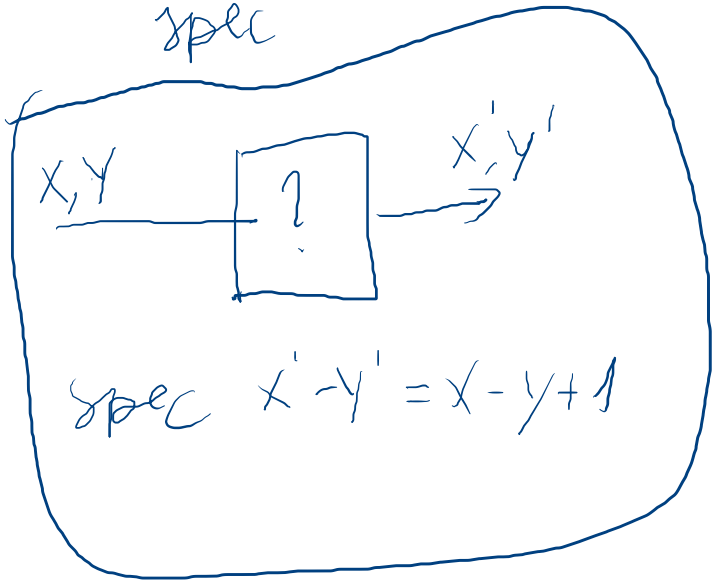
gadgets pour

$n \to n+1$ C++

$n \to n-1$ C--

$C > 0$

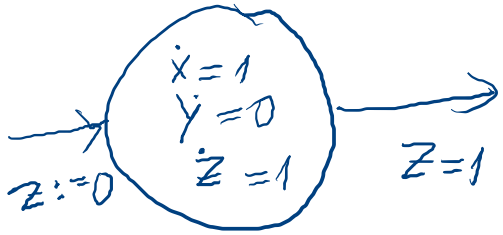
spec



pour $C > 0$?

gadget

pour C++



on a $x' = x+1$, $y' = y$ $\Rightarrow$ spec Ok

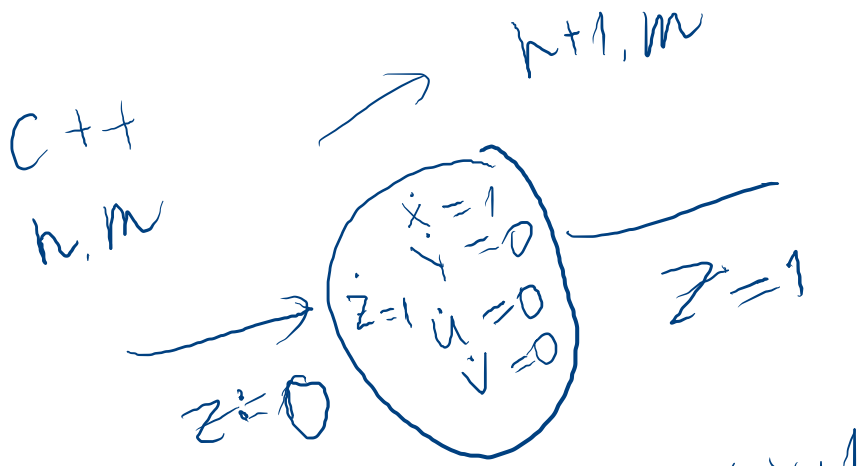
pour C--

Exo 1 - on passe au 2 compteurs

2 cpt $C = n$ $D = m$

$n = x - y$

$m = u - v$



spec $x' - y' = x - y + 1$

$u' - v' = u - v$

respectée

ou le fait pour

C++ D++

C-- D--

C > 0 ? D > 0

6 opérations

Exo1 - comment simuler une machine de Minsky

pour une MM, on fait un SA
donnée comme ceci

5 var
x, y, u, v, z

- pour chaque ligne de MM
  on fait un gadget comme
  décrit plus haut
- on les connecte selon les goto

Exo 1 - lemme sur la simulation

se prouve par induction sur la longueur

MM $(q, m, n) \dots\rightarrow (q', m', n')$

$\Updownarrow$

SA qui on a construit $(q, x=m, y=0, u=n, v=0, z=0)$

$\downarrow$

(q', x', y', u', v', z')

avec $x' - y' = m'$ et $u' - v' = n'$

Exo 1 - fin de preuve d'indécidabilité

pour la réduction on en déduit

MM : $(q, 0, 0) \dots\rightarrow (q', *, *)$.

ssi

SA $(q, 0, 0, 0, 0) \dots\rightarrow (q', * * \dots *)$

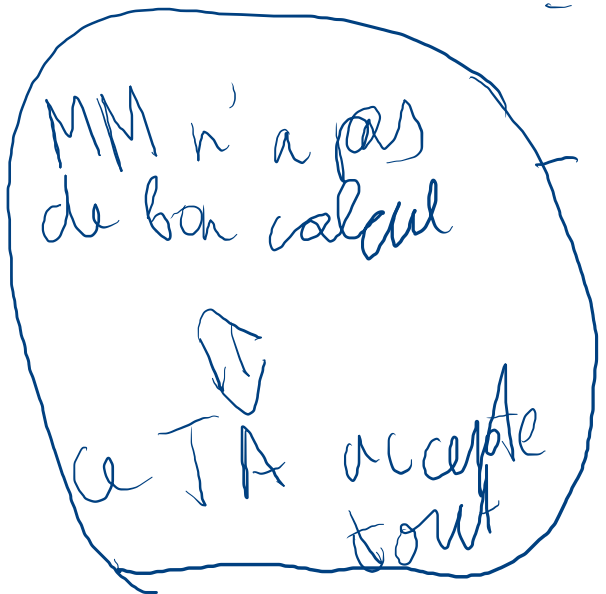
si Reach était décidable pour SA, elle le serait pour MM, or ce n'est pas le cas !

conclu : Reach indécidable pour SA

Théorème d'Alur: universalité indécidable pour TA (énoncé + principe de preuve)

Th d'Alur : pour un TA A
indécidable si $L(A)$ = tous les timed words

Idée : - un TA ne peut pas simuler une MM, hélas
- on va utiliser les TA pour inspecter les "logs" de MM.
- un TA peut reconnaître tous les logs faux



⇔ existe un bon calcul de MM
⇕
ce TA rejette au moins un log

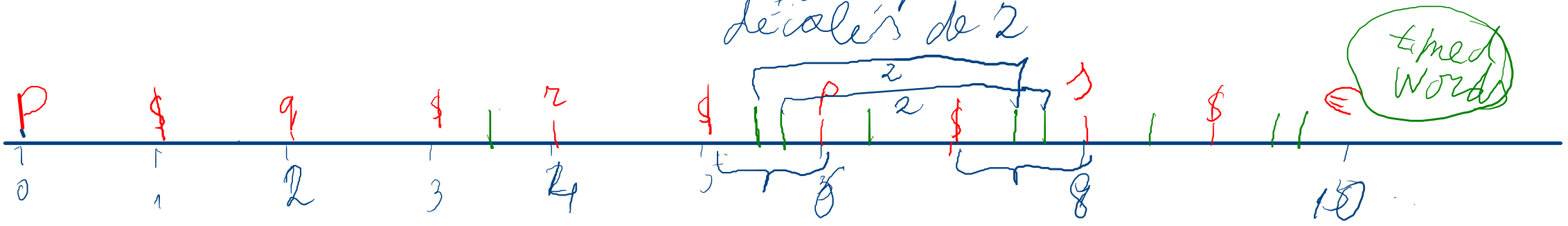
calcul de MM de $(p,0,0) \to (?,*,*)$

Preuve du thm: comment on code les logs

$(p,0,0) \to (q,0,1) \to (r,0,2) \to (p,1,2) \to (?,2,2)$

je le représente par mot temporel

on fera que tous les traits sont décalés de 2



C=0 D=0 C=0 D=1 C=0 D=2 C=1 D=2 C=1 D=2

pour une config $p, c=m, d=h$ on fera donc

comment on code les logs - suite et lemme de régularité

convention : - entre un état et un état suivant tout les 1111 représentant les compteurs sont séparés par 2 unités de temps

- quand on décrémente un cpt on supprime le dernier 1
- quand on incrémente on ajoute un 1 après le dernier

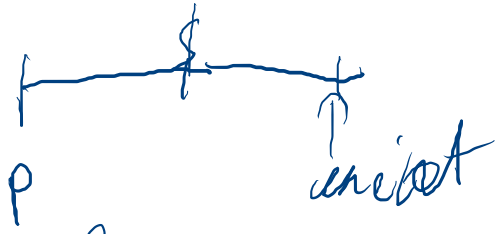
Lemme principal :

l'ensemble des mots tempo qui ne correspondent pas au calculs de la MM donnée (de (p,0,0) vers (s,*,*) est un langage temporisé régulier!!!

Lemme de régularité : idée de preuve

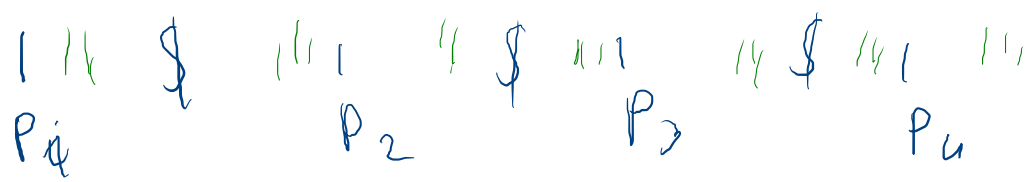
idée de preuve : on fait un petit TA pour reconnaître chaque type de Reg.

- ne pas commencer de



- ne pas finir par

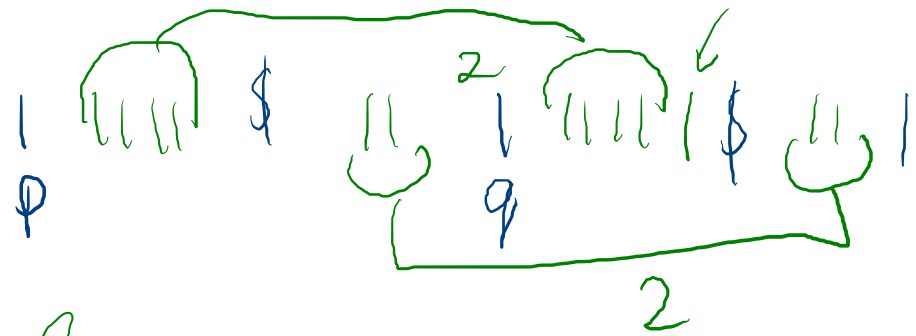
- ne pas avoir la bonne structure



- faire une erreur lors de l'exécution (par une instruction de la MM)

p.ex si on a p: C++ goto q

alors on veut partout



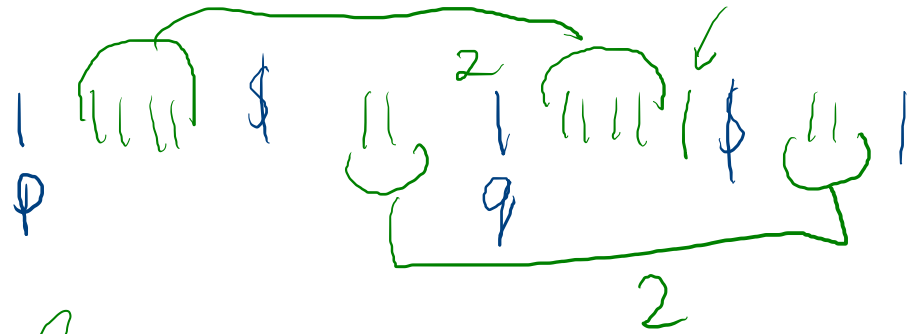
tous les bugs sont détectable par un AT

suite d'explication du lemme de régularité

- faire une erreur lors de l'exécution (par une instruction de la MM)

p.ex si on a p: c++ goto q

alors on veut partout



tous les loops sont détectable par un AT

fin de preuve : universalité indécidable

ainsi on fait un nbre fini de TA qui détectent tous les bugs.

soit A leur union

(1) L(A) est universel

(2) tout log de MM est bogué

(3) MM n'a pas de calcul $(p,0,0) \vdash \cdots \vdash (1,*,*)$

⇒ on saurait décider Lu(1) on saurait 6(3)

⇒ (1) indécidable