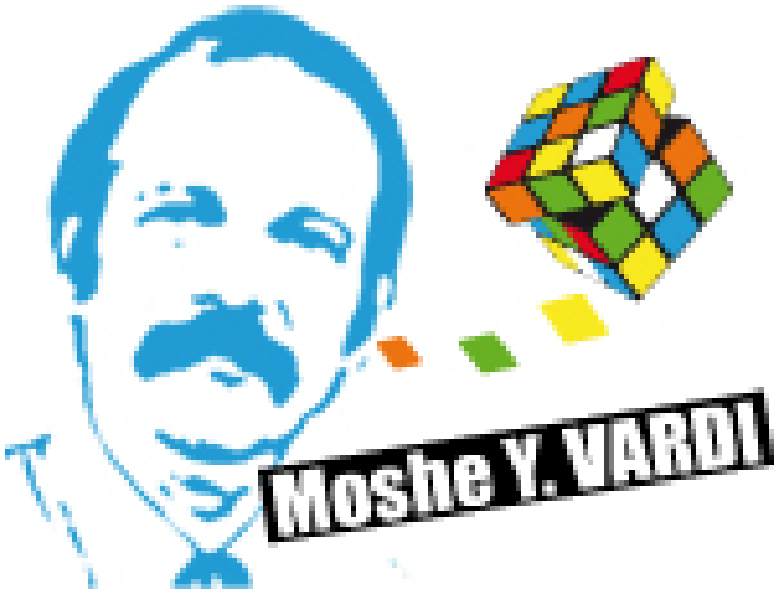
Motivation pour les logiques modales temporelles

François Schwarzentruber
ENS Cachan – Antenne de Bretagne

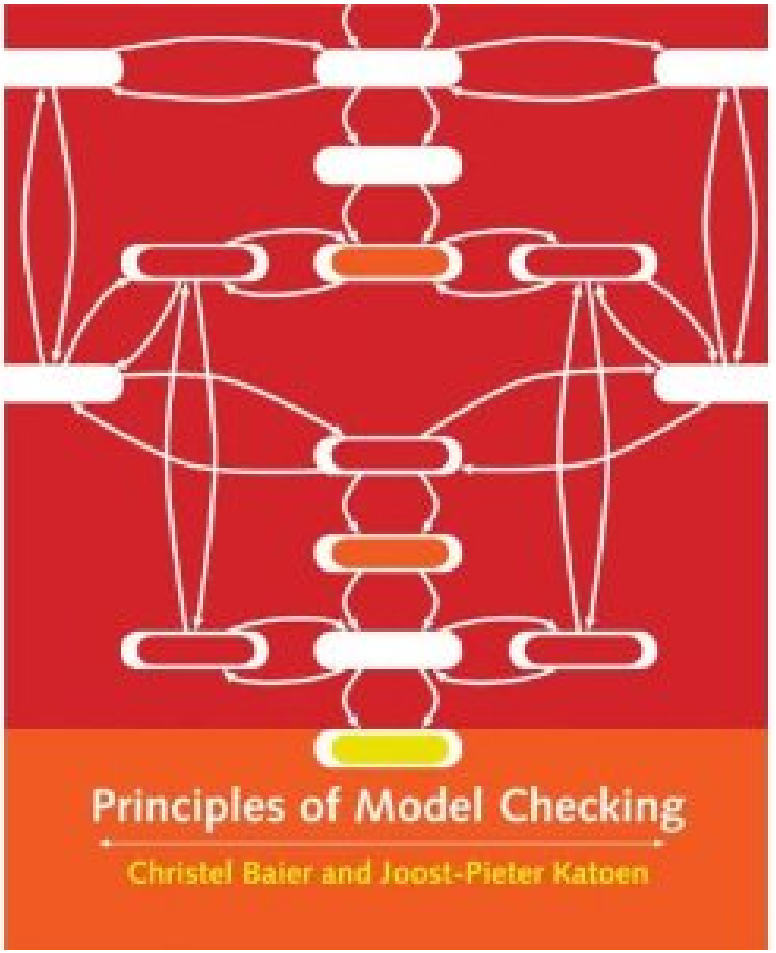
GIPSy 2011

The rise and fall of LTL
Moshe Y. Vardi
mardi 25 octobre de 10h à 12h
Amphitheater G, INRIA/IRISA, Rennes.



And Logic Begat Computer Science:
When Giants Roamed the Earth
Moshe Y. Vardi
jeudi 27 octobre de 14h à 15h30
Amphithéâtre A sur le Campus de Beaulieu, Rennes.

Références


Principles of Model Checking
Christel Baier and Joost-Pieter Katoen


VÉRIFICATION
DE LOGICIELS
Techniques et outils du model-checking
Ouvrage collectif - Coordination Philippe Schnoebelen

Que pouvez-vous me dire sur ce programme ?

$$\begin{aligned}
\textbf{proc}\ \mathsf{Inc} &= \textbf{while}\ \mathrm{true}\ \textbf{do}\ \textbf{if}\ x < 200\ \textbf{then}\ x := x + 1\ \textbf{fi}\ \textbf{od} \\
\textbf{proc}\ \mathsf{Dec} &= \textbf{while}\ \mathrm{true}\ \textbf{do}\ \textbf{if}\ x > 0\ \textbf{then}\ x := x - 1\ \textbf{fi}\ \textbf{od} \\
\textbf{proc}\ \mathsf{Reset} &= \textbf{while}\ \mathrm{true}\ \textbf{do}\ \textbf{if}\ x = 200\ \textbf{then}\ x := 0\ \textbf{fi}\ \textbf{od}
\end{aligned}$$

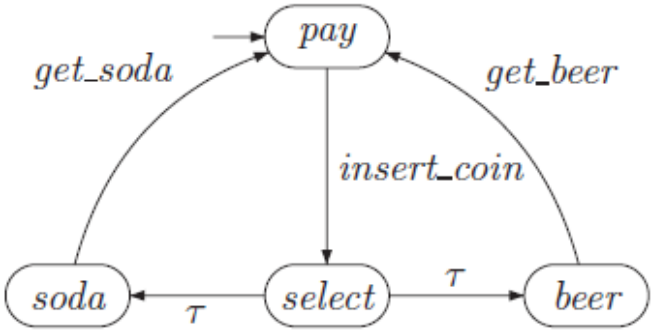
Exécuter Inc, Dec, Reset en parallèle !

Propriétés à vérifier

- Correction (est-ce que le système fait ce qu'on veut ?)
- Accessibilité (peut-on arriver dans un état mauvais ?)
- Sûreté (rien de mauvais n'arrive ~ invariant)
- Vivacité (liveness) : le système arrivera-t-il dans un bon état ?
- Fairness : est-ce qu'un événement arrive une infinité de fois ?
- Propriétés temps-réel : est-ce que le système réagit assez vite ?

Model-checking

Modélisation



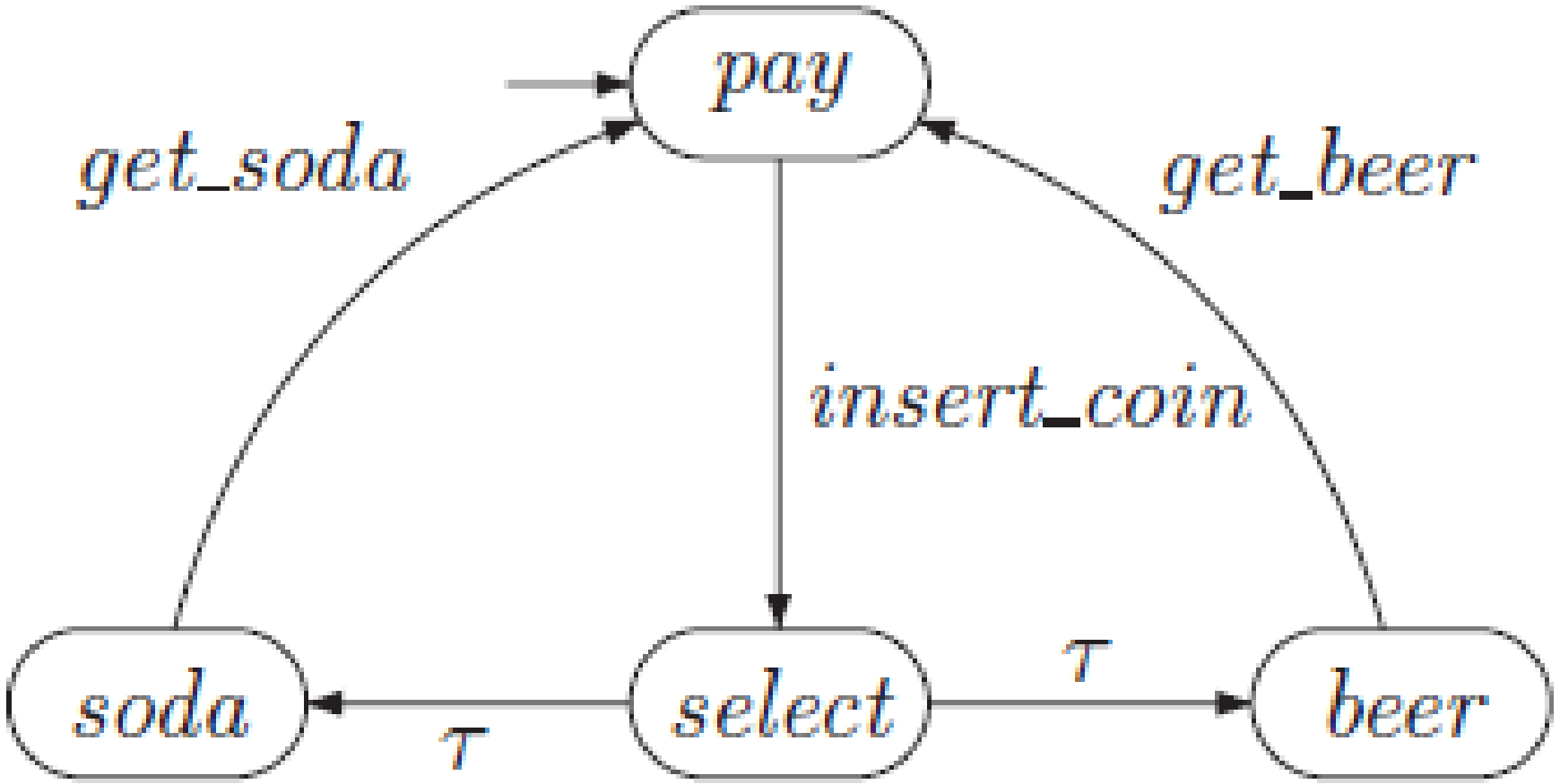
j'aurai une boisson

Model-checking

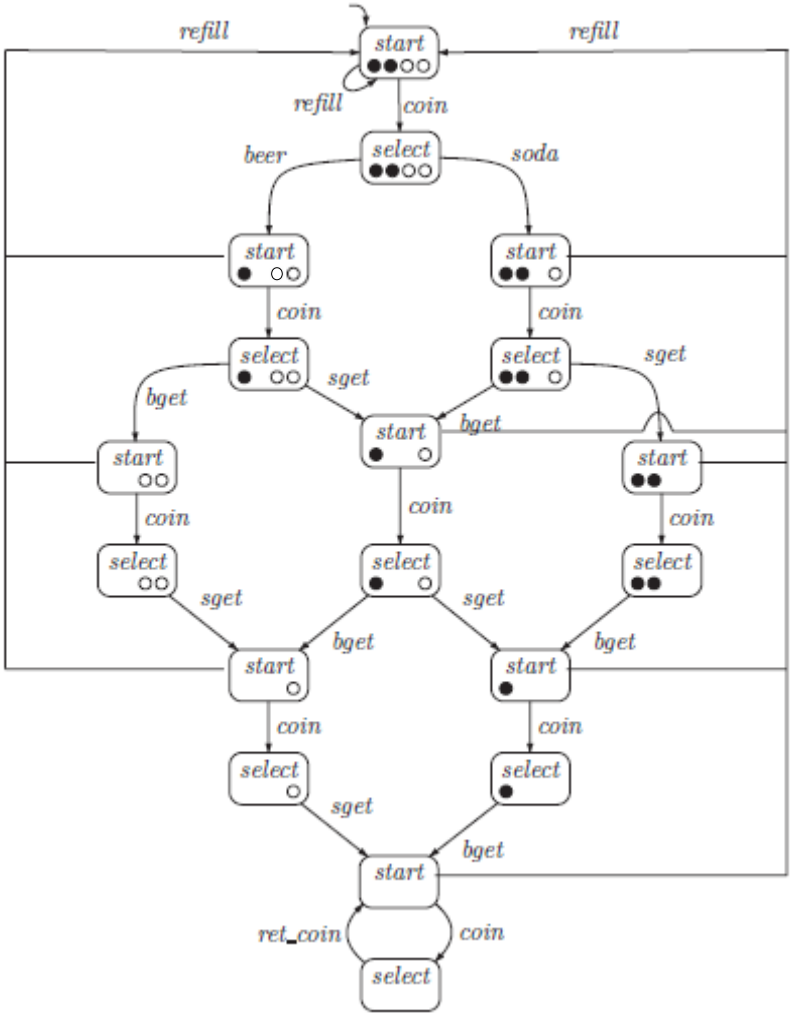
oui

Les modèles

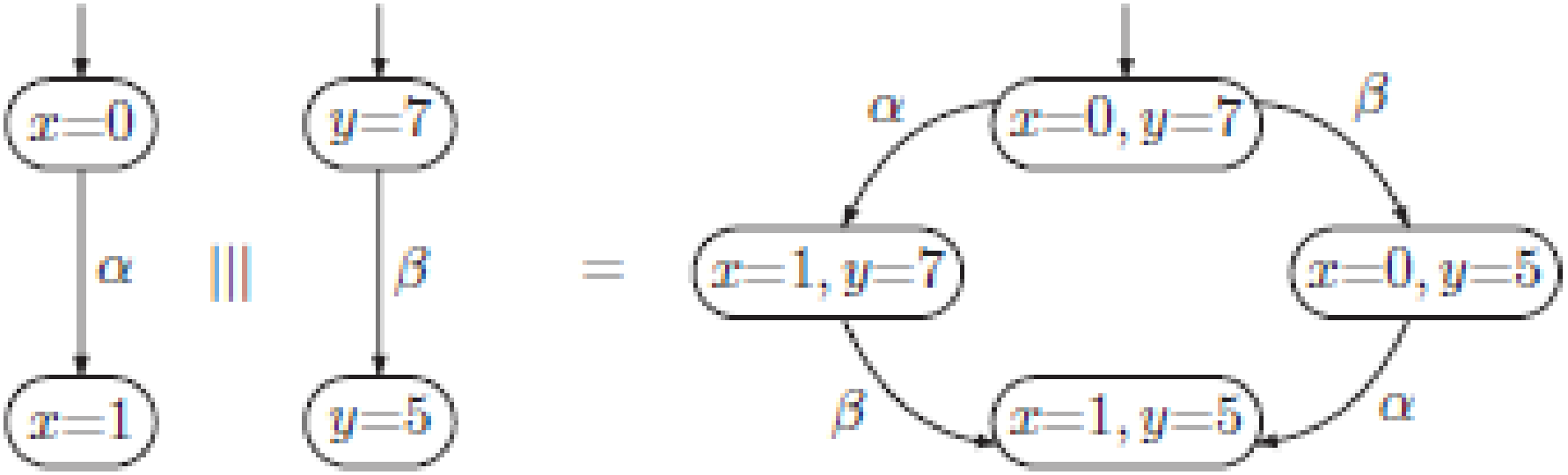
Système de transitions



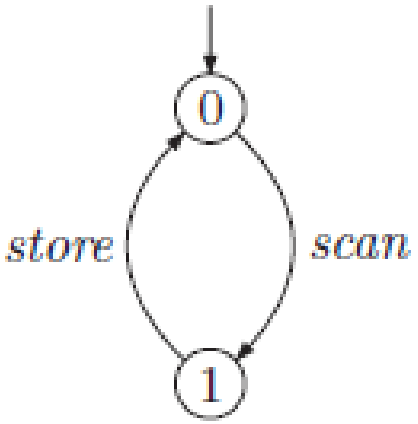
Explosion d'états



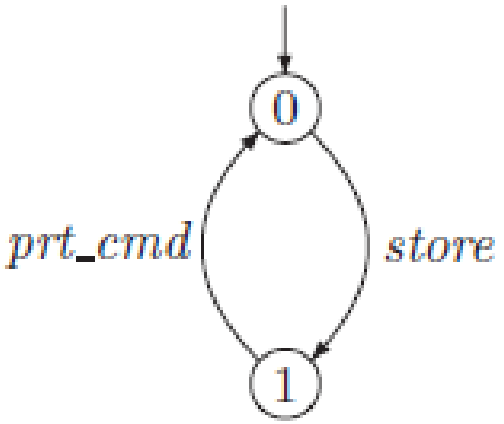
Parallélisme : composition asynchrone...re-explosion d'états



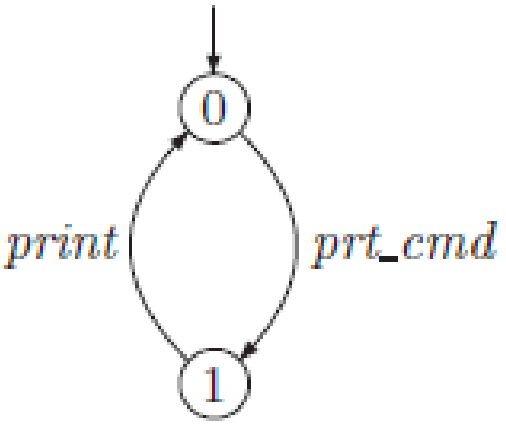
Composition parallèle synchrone



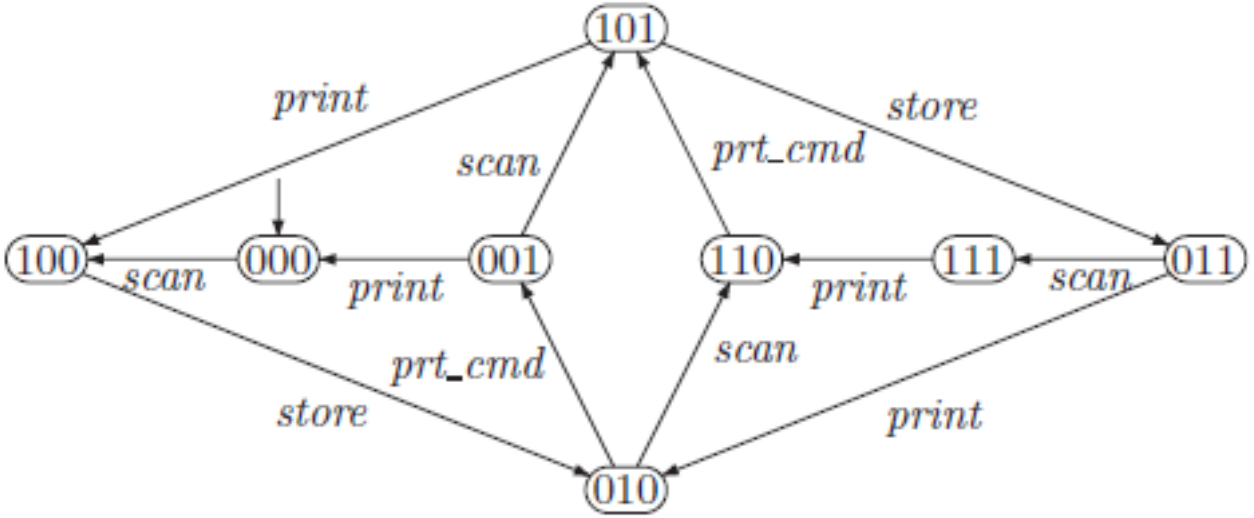
bar code reader



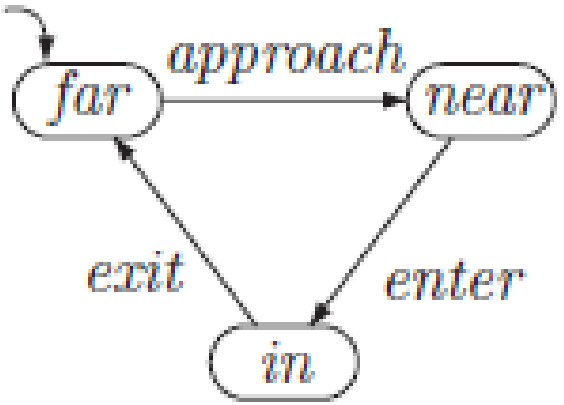
actual booking program



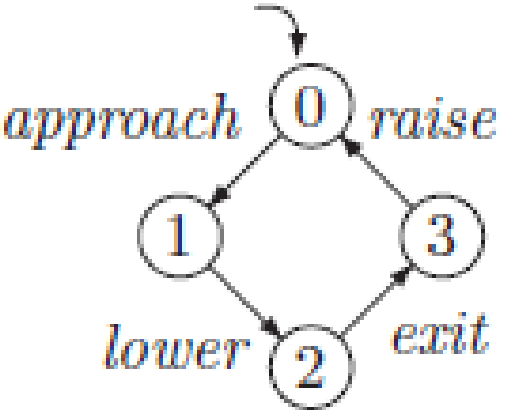
printer



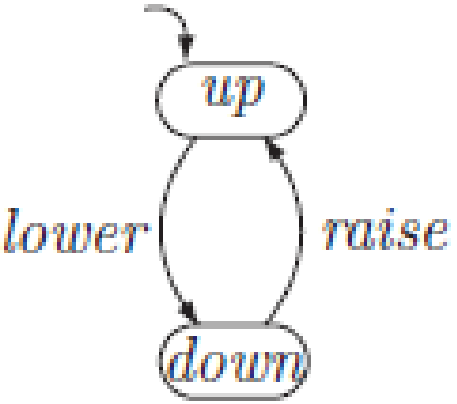
Composition parallèle synchrone et trains



Train

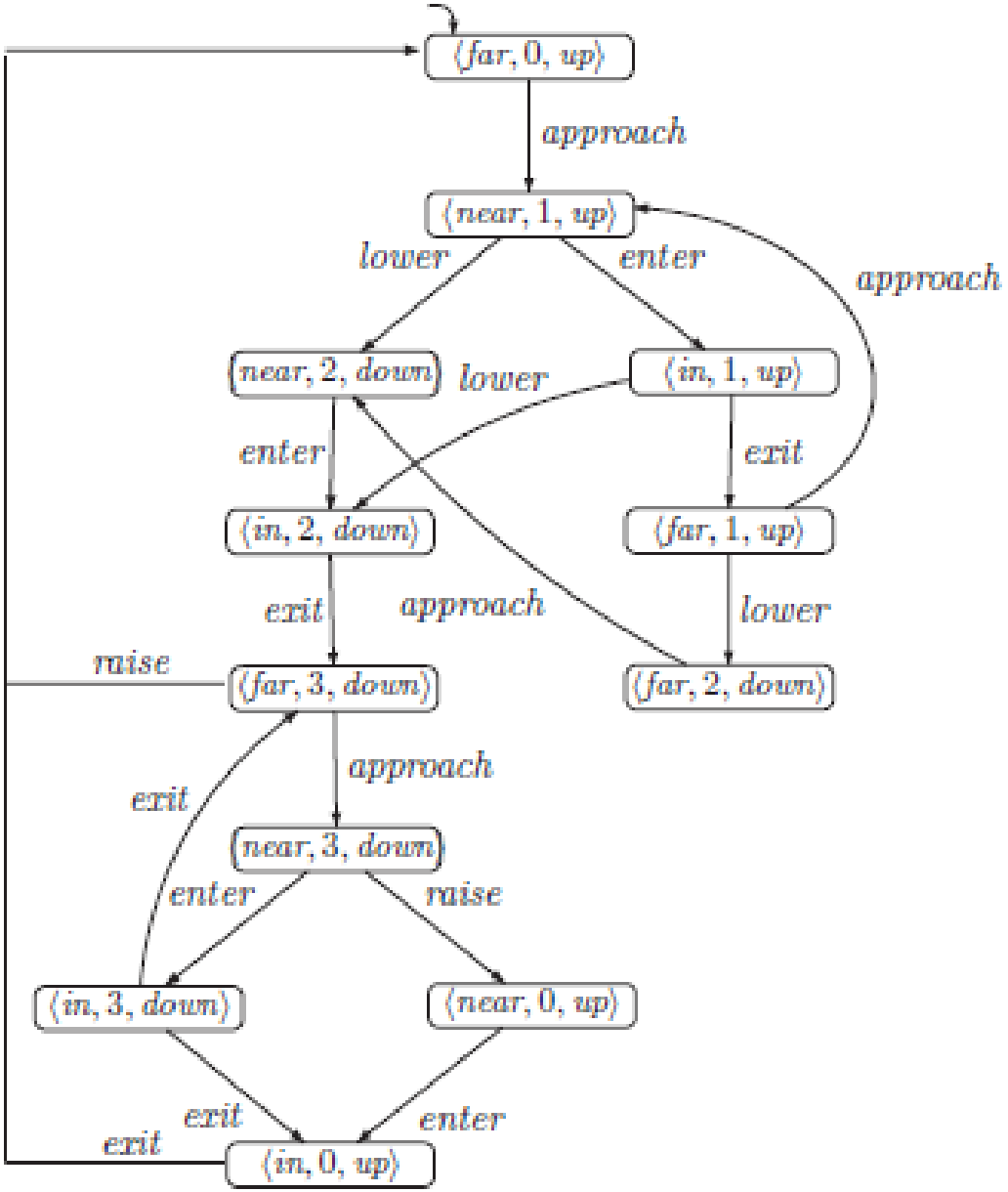


Controller



Gate

Composition : re-explosion d'états



Model-checking

Avantages :

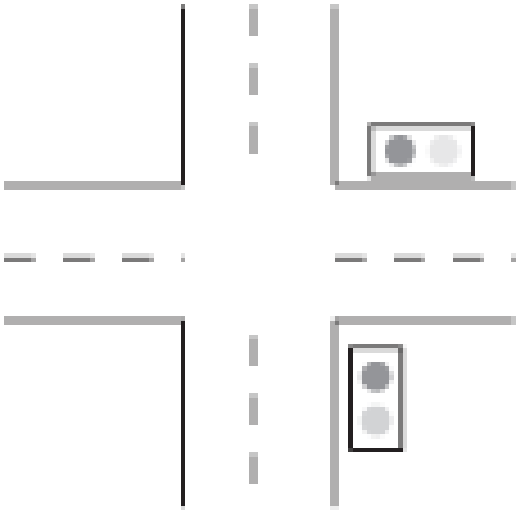
- Automatique
- Exhaustif

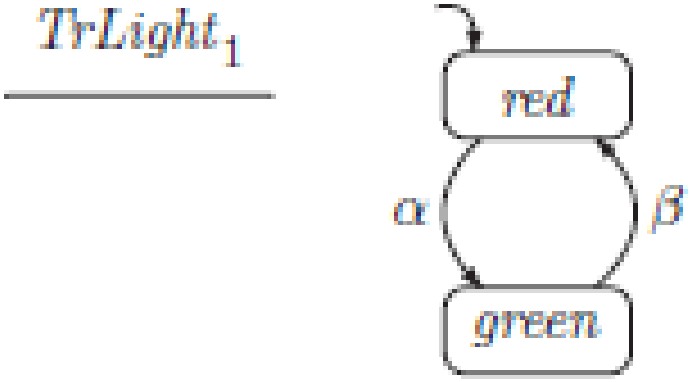
Inconvénients

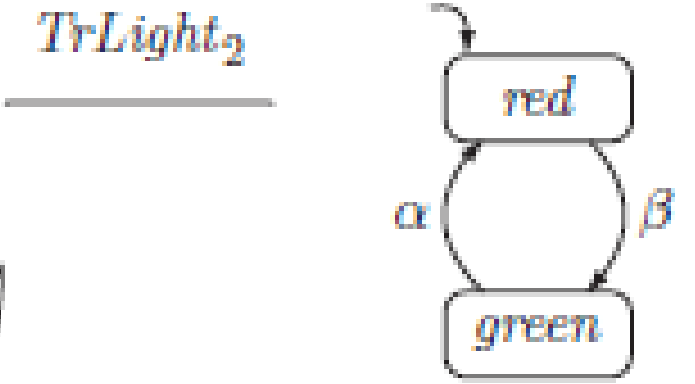
- Nécessite une modélisation... (mais on utilise la modélisation qui a servi à concevoir le système !)
- Modélisation peu précise
- Explosions d'états

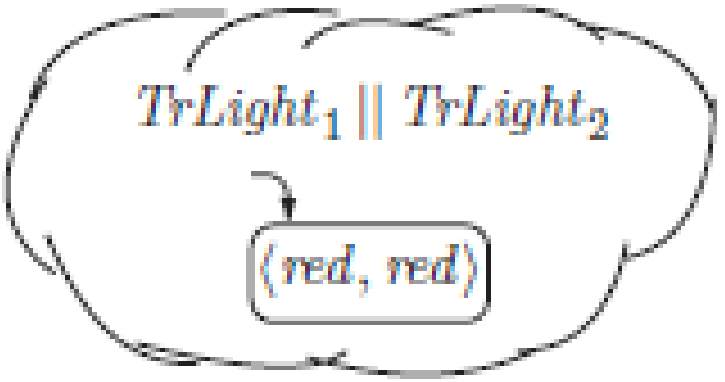
Propriétés à vérifier

Blocage

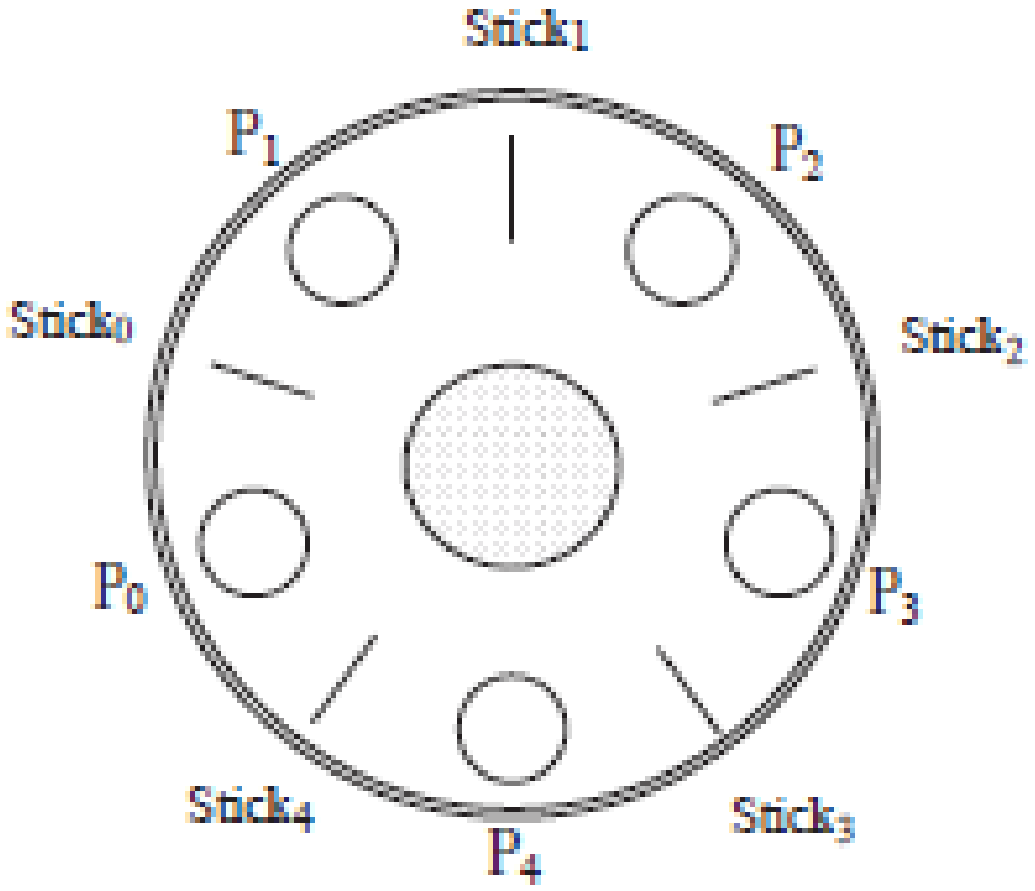



$TrLight_1$
red
α
β
green

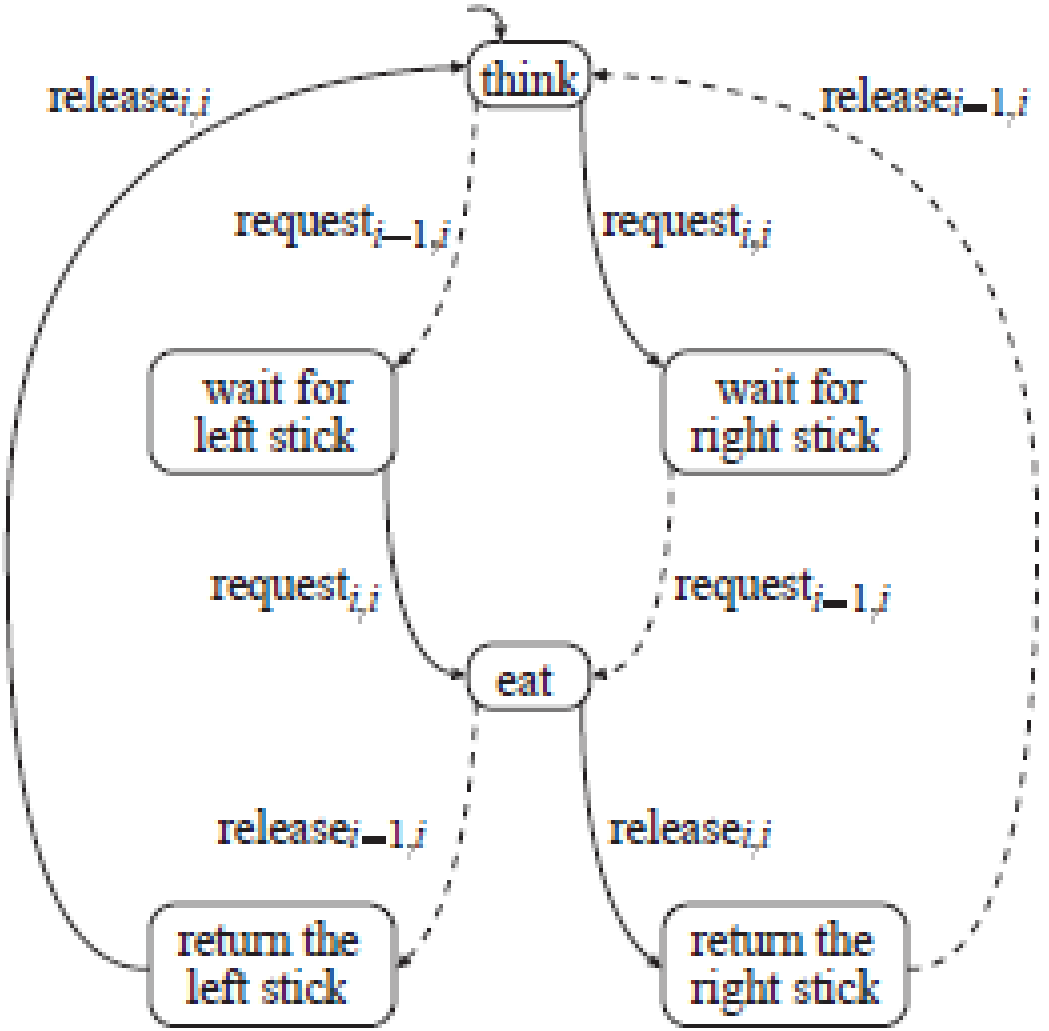

$TrLight_2$
red
α
β
green


$TrLight_1 \,||\, TrLight_2$
$\langle red, red \rangle$

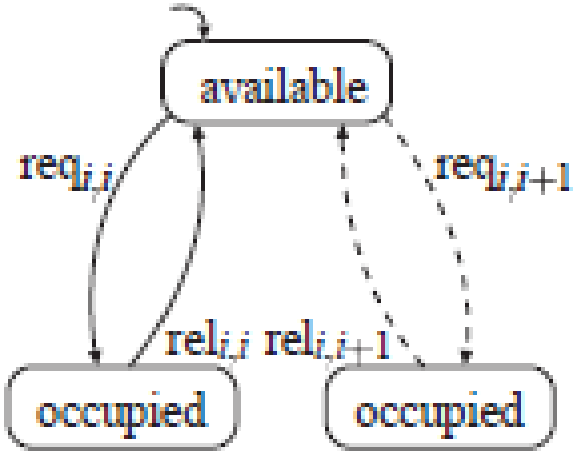
Blocage : dîner des philosophes [Dijkstra]



Les modèles pour chaque philosophe et chaque fourchette



philosophe



fourchette

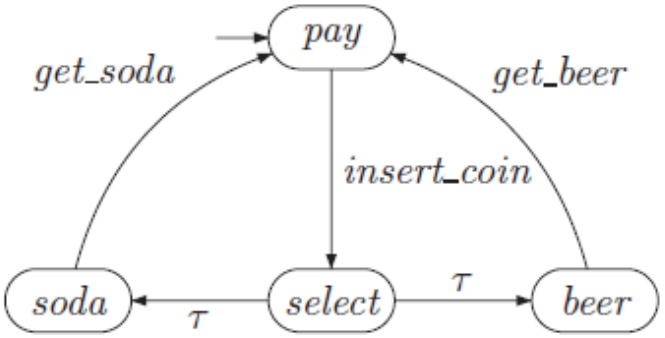
Propriétés

Dans toute exécution :

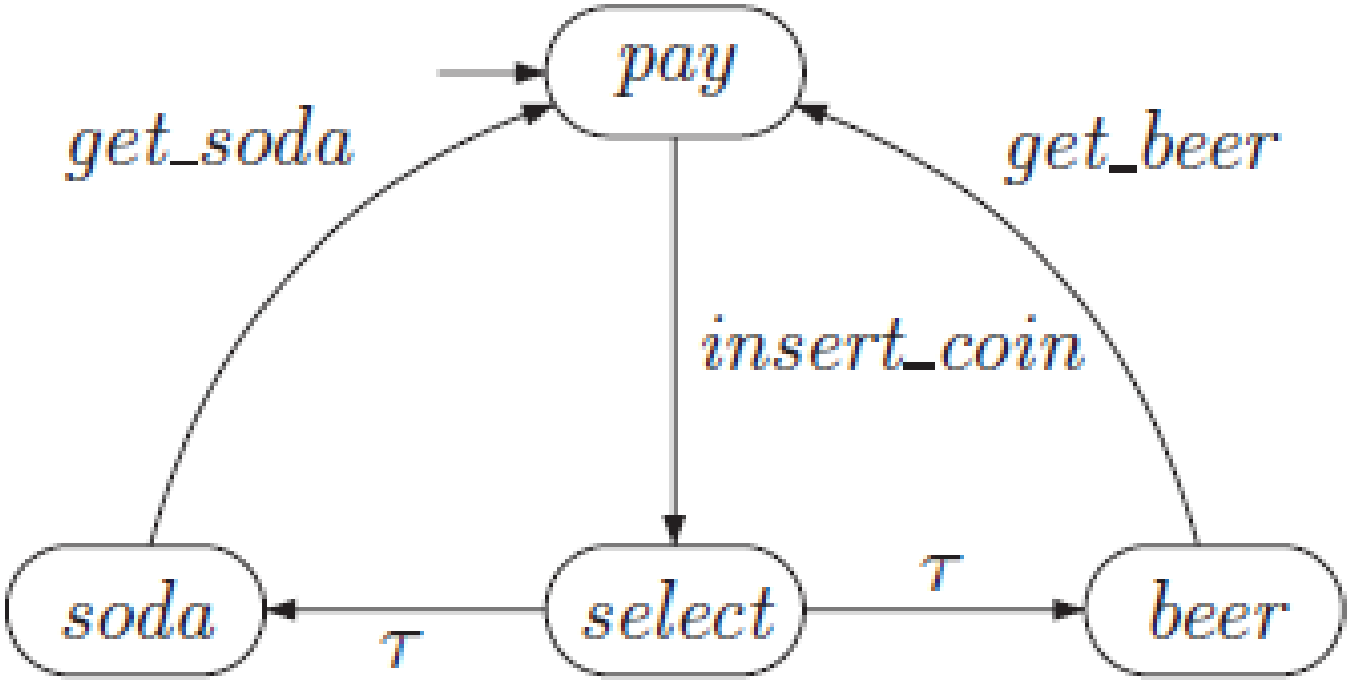
- La machine n'est **jamais** bloqué.
- Pour toute exécution, on peut toujours payer **une infinité de fois**.
- J'**aurai** toujours une boisson.
- Si je paie, **à l'instant d'après** je peux sélectionner.

Il existe une exécution :

- Il y a une exécution où j'**aurai** c soda.



Chemin



pay, select, soda, pay, select, soda, pay select, beer, pay, select, soda...

Logique = problème générique

	4		1					
		3	5				1	9
					6			3
		7			5			8
	8	1				9	6	
9			2			7		
6			9					
8	1				2	4		
					4		9	

Logique propositionnelle

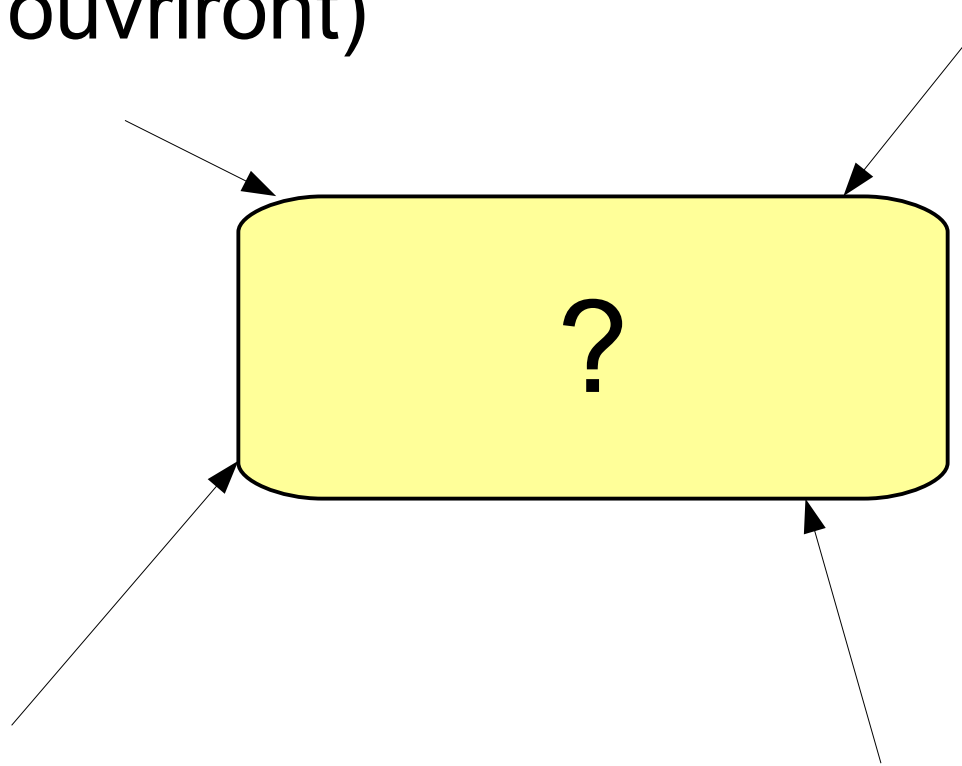
...

planification

Logique où se réduisent les problèmes de vérification

Vivacité (un jour, les portes du métro s'ouvriront)

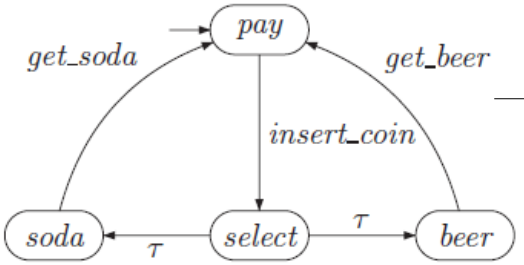
Sureté (invariant)



...

Model-checking avec une logique temporelle linéaire

j'aurai une boisson



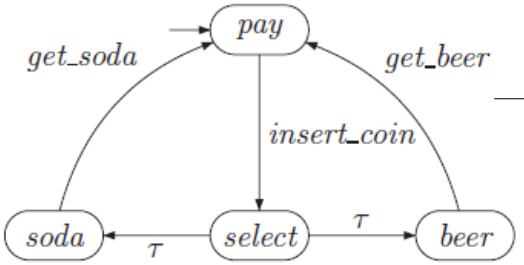
Model-checking

oui, c'est vrai
dans toutes
les exécutions

LTL

Model-checking avec une logique temporelle branché

Dans toute les exécutions, j'aurai une boisson
et il existe une exécution où je me pairai une bière puis un soda

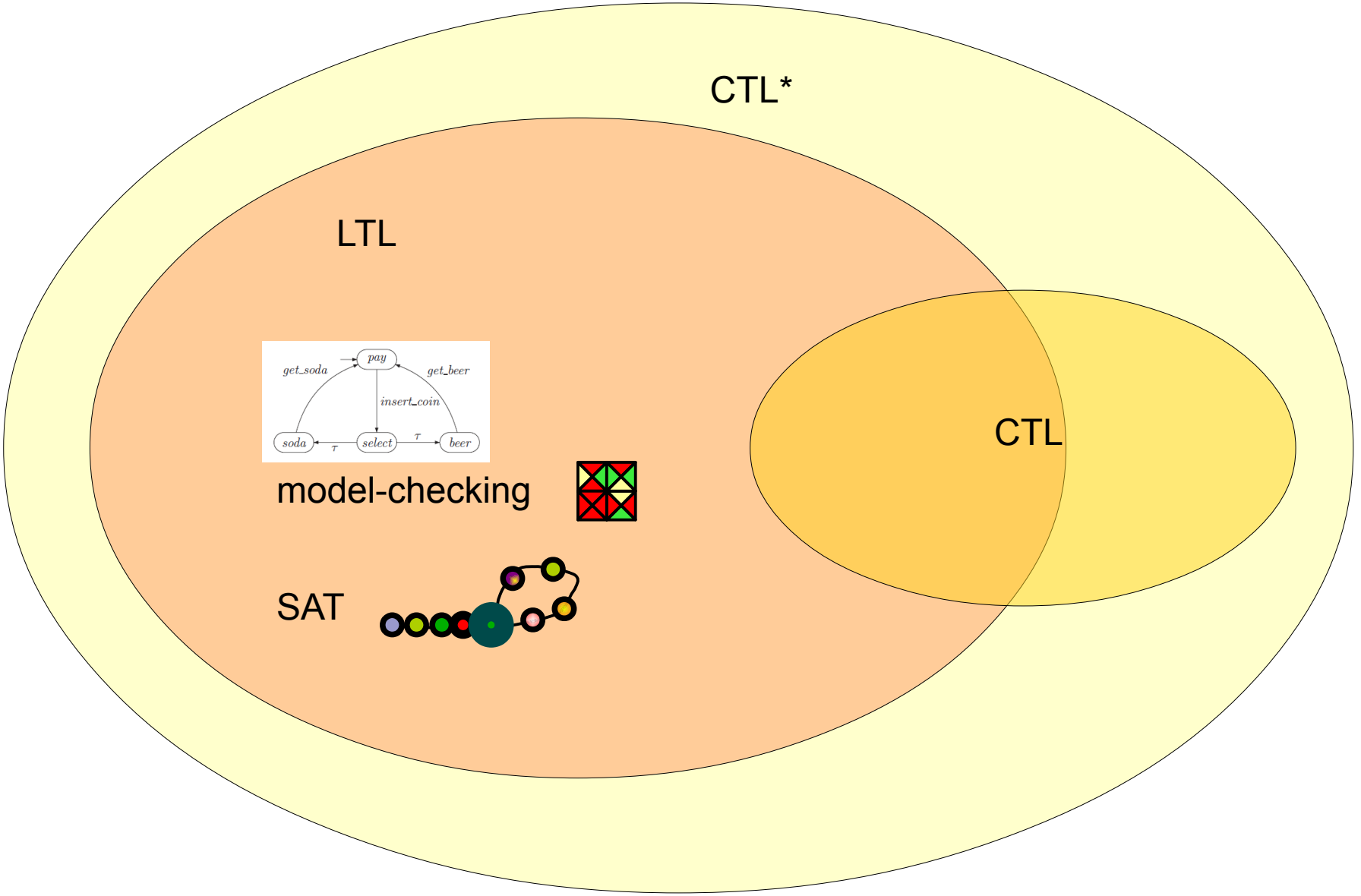


Model-checking

oui, c'est vrai

CTL* et le fragment CTL

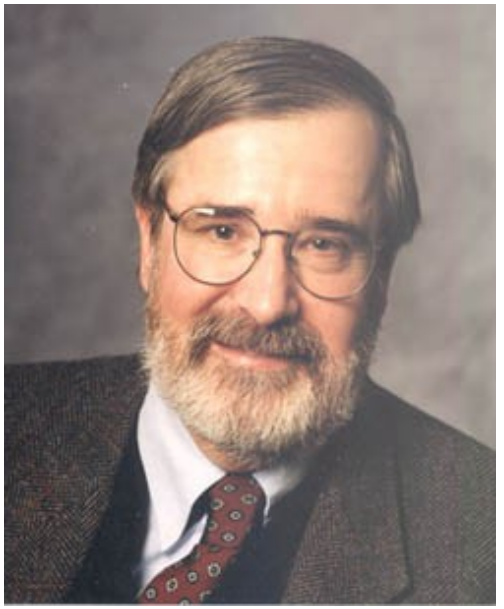
Plan du cours



Un peu d'histoire

Pnueli

A. P. Sistla



Edmund M. Clarke

(Havard university)

1977 :
Création de LTL

1982 : the complexity of LTL