並行分散計算特論

Shoji Yuen

2011/10/4

概要

シラバス 並行計算の概念および体系について修得する。特に、通信プロセス計算モデルの表現と意味論について講義し、従来の逐次計算モデルと比較して、並行計算によって生じる問題について、体系的な意味づけとその対処方法について学ぶ。ネットワークを介した情報システムにおける並行計算の具体的イメージを通して信頼性の高い並行ソフトウェアの構築、時間を含むシステムの振舞い、さらに、具体的な信頼性向上のための形式手法としてモデル検査について述べる。

キーワード 並行計算モデル、プロセス計算、時間オートマトン、モデル検査、様相論理

講義予定

1. 概論
2. 並行計算モデル
3. ラベルつき遷移系
4. 双模倣関係と等価性
5. 双模倣に基づく意味論
6. プロセス計算
7. 時間オートマトン
8. モデル検査
9. 様相論理による振舞の表現

資料は英語でつくる予定。
教科書については入手困難なものはホームページからダウンロード (パスワードつき)

Formal semantics for working programmers

- Rigorous mathematical model

$$P = Q,\ P \leq Q,\quad \Gamma, \mathsf{prop} \models P,\ \Gamma \vdash p : \mathsf{prop}$$

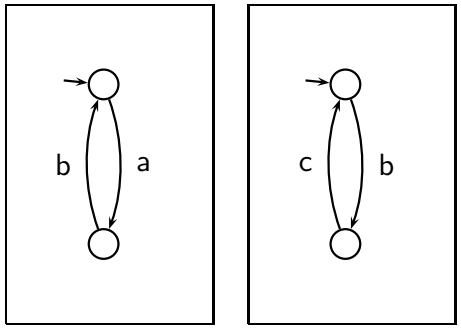
  - Correct Design Principle
  - Compositionality

  Sometimes "over rigorous"
- Obstacles and limitations:
  - Computability
  - Full–abstractness
  - Complexity
  - Scaleability

  Cost/effectiveness ratio

Communicating Processes

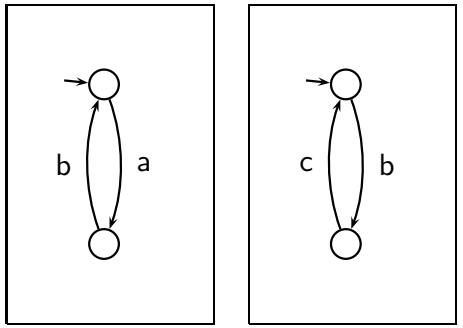
Parallel compositon of transition systems



Synchronization by label matching

Communicating Processes

Parallel compositon of transition systems



Synchronization by label matching

without synchronization: $(ab)^*||(bc)^*$
with synchronization: $a(b(a|c))^*$

Communication structure

- **Basic Mechanism**
  - Handshake (synchronous)
  - Mailbox (asynchronous)
- **Configuration**
  - Fixed (static structure)
  - Variable (dynamic structure)

Communication structure

- **Basic Mechanism**
  - Handshake (synchronous)
  - Mailbox (asynchronous)
- **Configuration**
  - Fixed (static structure)
  - Variable (dynamic structure)

Behavioral semantics

- Semantics of a program (Continuous) Function $f(x)$

  Classical I/O system
  - Input = parameters
  - Output = evaluated value

  Communicating system

  ??Appropriate I/O correspondence??

Domains of 'interactional behaviors?

A program be a function over the behavior domain

Algebraic Approach

Algebraic Characterization

Based on CCS (Milner:1980,1989)

Behavior is an equivalence over terms

$\Rightarrow$ An algebra of behavior

$P = Q$ "P behaves exactly like Q"

For all operations F in the calculus:
$F(P) = F(Q)$ when $P = Q$

Process algebra $(\mathcal{P}, =)$: $\mathcal{P}$:Set of process terms

Processes = communicating (concurrent) programs

Semantics of Programs

Formal semantics
=a mathematical structure corresponding to programs

Equivalence relation $\sim$ over program terms

$$P \sim Q$$

Finest algebra:term algebra
(all programs are distinguished if any syntactic difference.)

$$1 + 1 \neq 2 \qquad \text{Usuallyunacceptable}$$

?Appropirate Equivalence (or preorder)?

Coarsest algebra = singleton
all programs are identical.

Behavior of automata

The (classical) theory of automata

Definition

Automaton

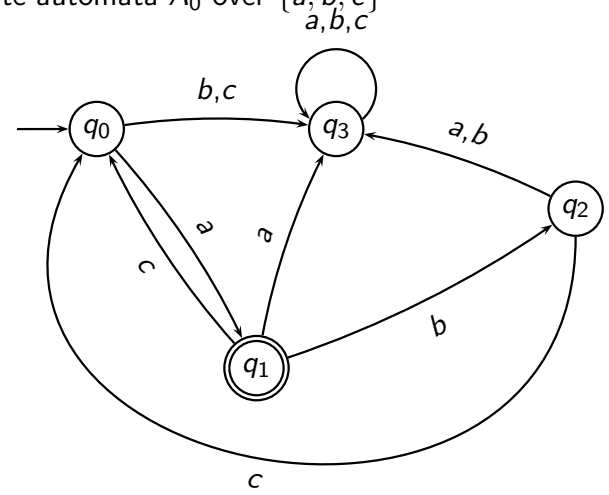
Given a set of actions Act: $A = \langle \mathcal{Q}, q_0, \mathcal{F}, \mathcal{T} \rangle$

- $\mathcal{Q}$*:Set of states;*
- $q_0 \in \mathcal{Q}$*:the start state;*
- $\mathcal{F} \subseteq \mathcal{Q}$*:the accepting states; and*
- $\mathcal{T} \subseteq \mathcal{Q} \times Act \times \mathcal{Q}$*:the transtions*

$(q, a, q') \in \mathcal{T}$ is written as $q \xrightarrow{a} q'$. A is *finite* if $|\mathcal{Q}|$ is finite. A is *deterministic* if $|\{q' \mid q \xrightarrow{a} q'\}| = 1$ for all q, a.

Transition graph

A finite automata A_0 over $\{a, b, c\}$



Semantics of automata

Definition

An automaton: $A = \langle \mathcal{Q}, q_0, \mathcal{F}, \mathcal{T} \rangle$

$\hat{A} = \{w | q_0 \xrightarrow{w} q,\ q \in \mathcal{F}\}$

$\xrightarrow{w} = \xrightarrow{a_1}\xrightarrow{a_2} \cdots \xrightarrow{a_2}$ *where* $w = a_1, \ldots, a_n$

$A_1 \sim A_2$ *if* $\hat{A}_1 = \hat{A}_2$

A pair of automata are equal if they accept the same language.

Regular sets

Operation over sets

$S_1 \cup S_2$ (Union)
$S_1 \cdot S_2 = \{w_1 \cdot w_2 | w_1 \in S_1, w_2 \in S_2\}$
$S^* = \{\varepsilon\} \cup S \cup S \cdot S \cup S \cdot S \cdot S \cup \cdots$

Regular set

Definition

A set of strings over Act is regular *if it is constructed applying following rules:*

- $\emptyset$, $\{a\}$ *are regular;*
- $S_1 \cup S_2$ *is regular if both* S_1 *and* S_2 *are regular;*
- $S_1 \cdot S_2$ *is regular if both* S_1 *and* S_2 *are regular;*
- S^* *is regular if* S *is regular*

Arden's Rule

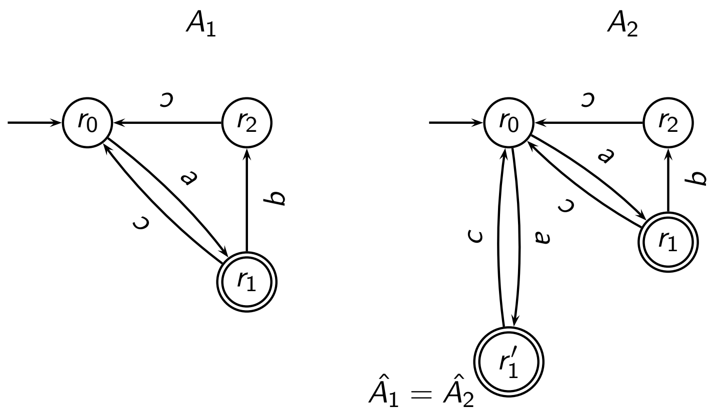
Proposition

For any sets of strings S *and* T*:*
$X = S \cdot X + T$ *has* $X = S^* \cdot T$ *as a solution. Moreover, this solution is unique if* $\varepsilon \notin S$

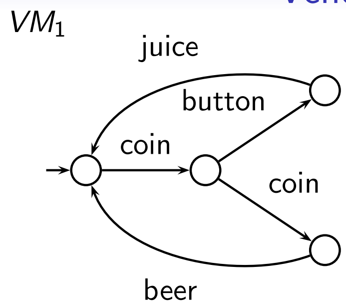
$$\begin{aligned} X_0 &= aX_1 + bX_3 + cX_3 \\ X_1 &= aX_3 + bX_2 + cX_0 + \varepsilon \\ X_2 &= aX_3 + bX_3 + cX_0 \\ X_3 &= aX_3 + bX_3 + cX_3 \end{aligned}$$

Using the Arden's rule:$\hat{A}_0 = a((bc + c)a)^*$

DFA and NFA

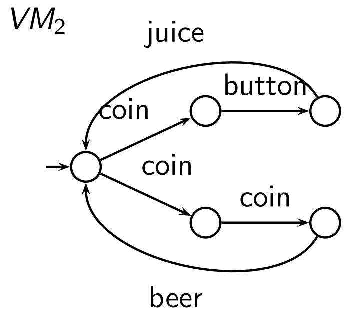


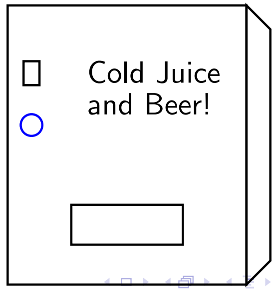
Vending Machine



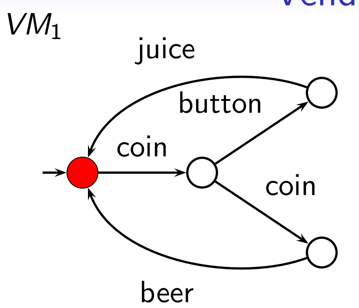
To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer



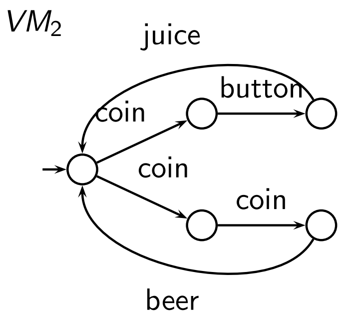


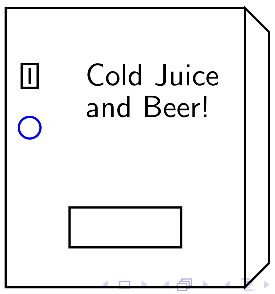
Vending Machine



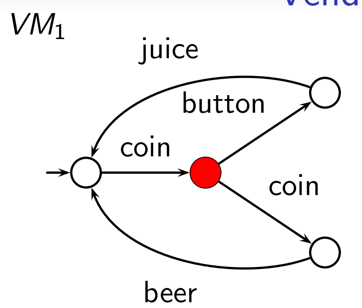
To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer



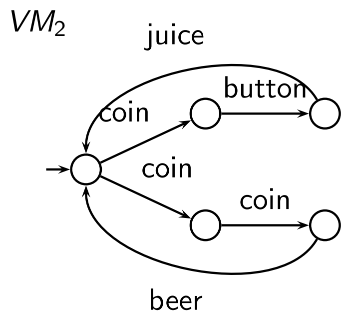


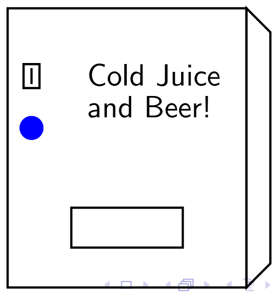
Vending Machine



To get a juice:
Put one coin and push button
To get a beer:
Put two coins

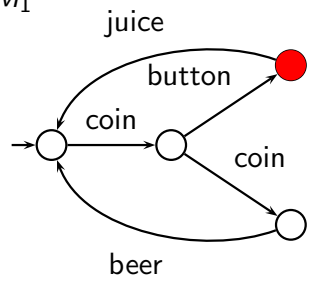
Correctness:
one coin for juice
two coins for beer





Vending Machine

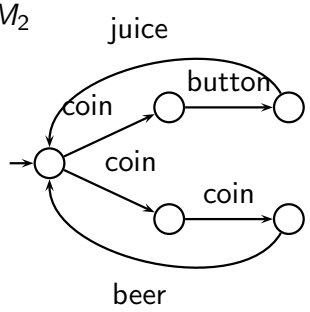
VM_1

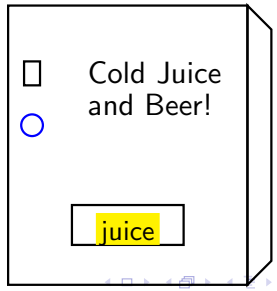


To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer

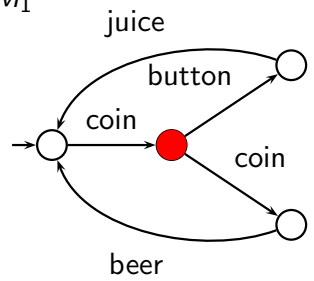
VM_2





Vending Machine

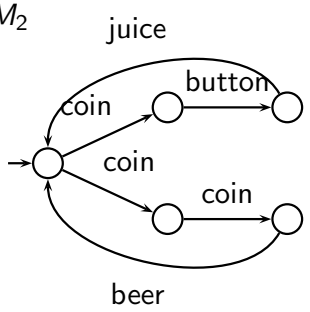
VM_1

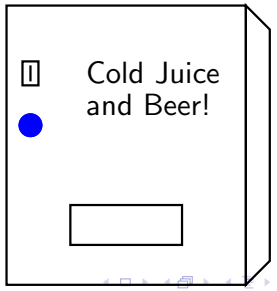


To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer

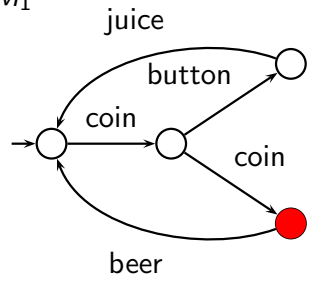
VM_2





Vending Machine

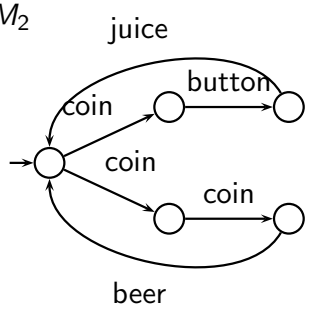
VM_1

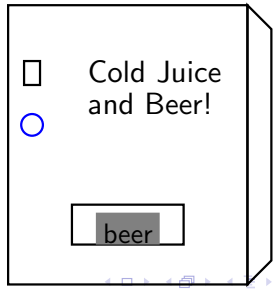


To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer

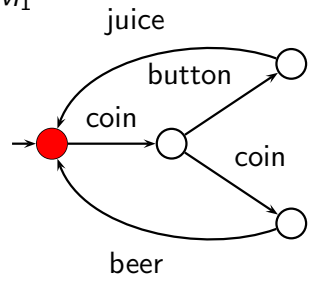
VM_2





Vending Machine

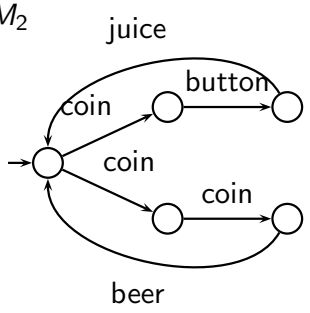
VM_1

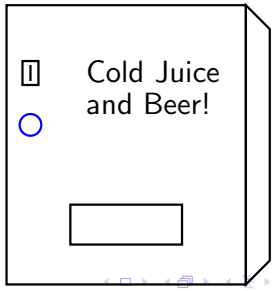


To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer

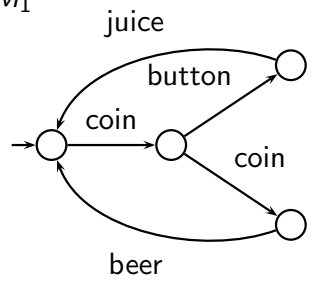
VM_2





Vending Machine

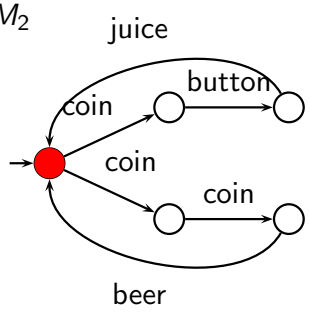
VM_1

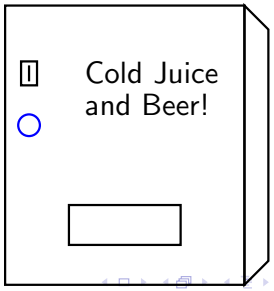


To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer

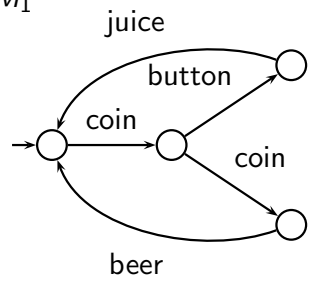
VM_2





Vending Machine

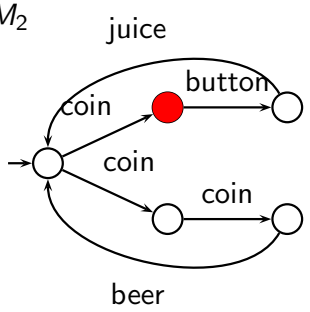
VM_1

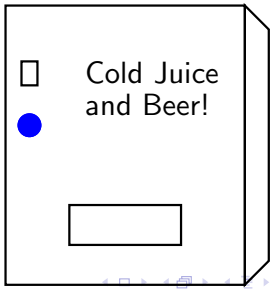


To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer

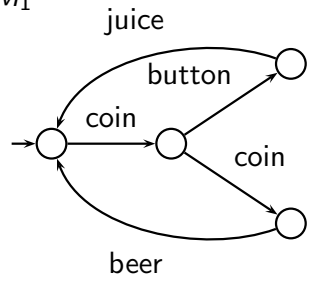
VM_2





Vending Machine

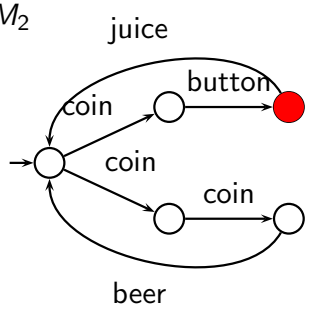
VM_1

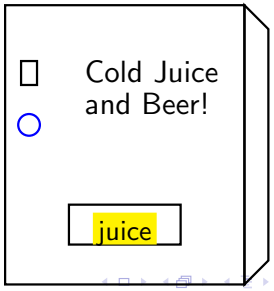


To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer

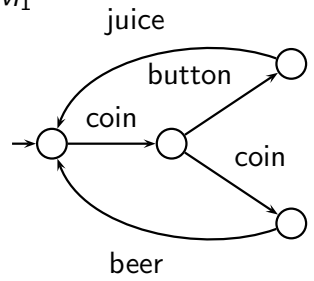
VM_2





Vending Machine

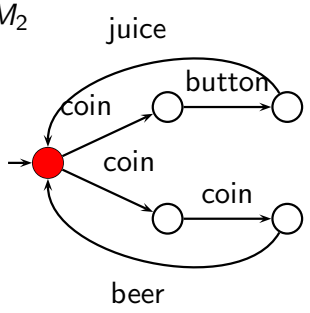
VM_1

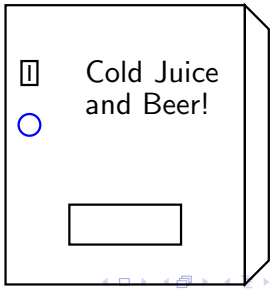


To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer

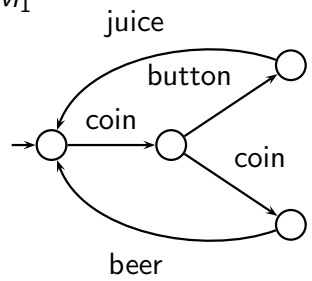
VM_2





Vending Machine

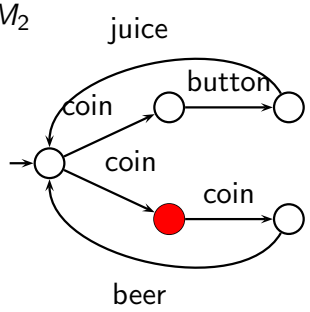
VM_1

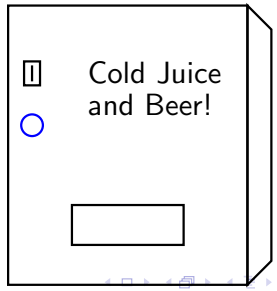


To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer

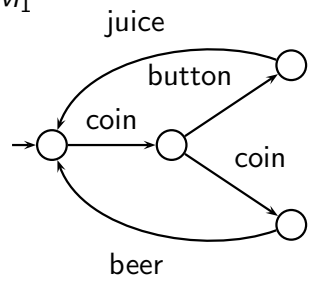
VM_2





Vending Machine

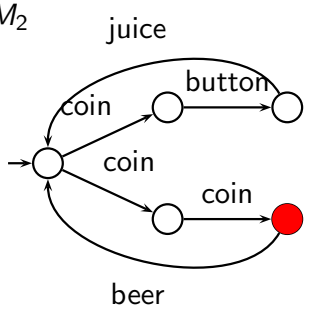
VM_1

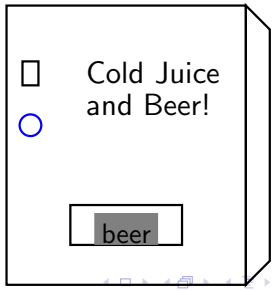


To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer

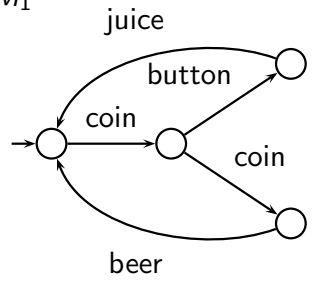
VM_2





Vending Machine

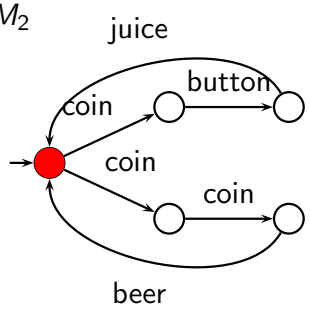
VM_1

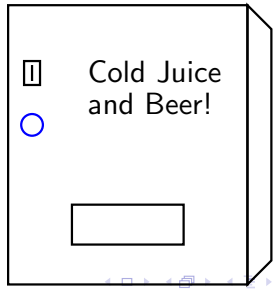


To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer

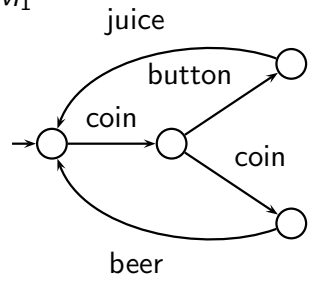
VM_2





Vending Machine

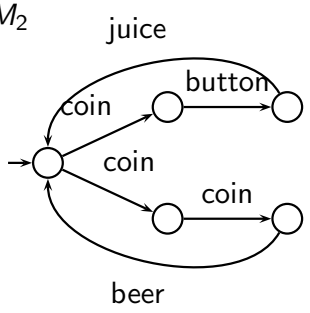
VM_1

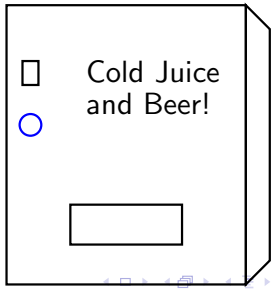


To get a juice:
Put one coin and push button
To get a beer:
Put two coins

Correctness:
one coin for juice
two coins for beer

VM_2





Correct vending machine?

Intuitively VM_1 is correct and VM_2 is broken.

Correct vending machine?

Intuitively VM_1 is correct and VM_2 is broken.

Why?

Correct vending machine?

Intuitively VM_1 is correct and VM_2 is broken.

Why?

A customer has control in VM_1, but a machine has control in VM_2.

Differnce is recognized during the interaction.

Reactive system

A reactive system behavior is not fully characterized by languages!
Necessary to have an alternative equivalence notion to distinguish: