

- ① Prolégomènes
- ② Modélisation de programmes et de systèmes logiciels
- ③ Modélisation relationnelle

- 1 Prolégomènes
- 2 Modélisation de programmes et de systèmes logiciels
- 3 Modélisation relationnelle

① Prolégomènes

② Modélisation de programmes et de systèmes logiciels

③ Modélisation relationnelle

- ▶ \mathcal{R} : exigences du système.

- ▶ \mathcal{R} : exigences du système.
- ▶ \mathcal{D} : domaine du problème.

- ▶ \mathcal{R} : exigences du système.
- ▶ \mathcal{D} : domaine du problème.
- ▶ \mathcal{S} : système répondant aux spécifications.

- ▶ \mathcal{R} : exigences du système.
- ▶ \mathcal{D} : domaine du problème.
- ▶ \mathcal{S} : système répondant aux spécifications.

\mathcal{D}, \mathcal{S} SATISFAIT \mathcal{R}

- ▶ \mathcal{R} : exigences du système.
- ▶ \mathcal{D} : domaine du problème.
- ▶ \mathcal{S} : système répondant aux spécifications.

\mathcal{D}, \mathcal{S} SATISFAIT \mathcal{R}

- ▶ \mathcal{R} : pre/post.
- ▶ \mathcal{D} : entiers, réels, ...
- ▶ \mathcal{S} : code, procédure, programme, ...

A program P satisfies a (pre,post) contract :

- ▶ P transforms a variable v from initial values v_0 and produces a final value $v_f : v_0 \xrightarrow{P} v_f$
- ▶ v_0 satisfies pre : $\text{pre}(v_0)$ and v_f satisfies post : $\text{post}(v_0, v_f)$
- ▶ $\text{pre}(v_0) \wedge v_0 \xrightarrow{P} v_f \Rightarrow \text{post}(v_0, v_f)$
- ▶ \mathbb{D} est le domaine RTE de V

requires $\text{pre}(v_0)$
 ensures $\text{post}(v_0, v_f)$
 variables X

```
begin
  0 :  $P_0(v_0, v)$ 
  instruction0
  ...
  i :  $P_i(v_0, v)$ 
  ...
  instructionf-1
  f :  $P_f(v_0, v)$ 
end
```

- ▶ $\text{pre}(v_0) \wedge v = v_0 \Rightarrow P_0(v_0, v)$
- ▶ $\text{pre}(v_0) \wedge P_f(v_0, v) \Rightarrow \text{post}(v_0, v)$
- ▶ For any pair of labels ℓ, ℓ' such that $\ell \longrightarrow \ell'$, one verifies that, pour any values $v, v' \in \text{MEMORY}$

$$\left(\begin{array}{l} \text{pre}(v_0) \wedge P_\ell(v_0, v) \\ \wedge \text{cond}_{\ell, \ell'}(v) \wedge v' = f_{\ell, \ell'}(v) \\ \Rightarrow P_{\ell'}(v_0, v') \end{array} \right),$$
- ▶ For any pair of labels m, n such that $m \longrightarrow n$, one verifies that, $\forall v, v' \in \text{MEMORY} :$

$$\text{pre}(v_0) \wedge P_m(v_0, v) \Rightarrow \text{DOM}(m, n)(v)$$

Example of an annotation

```
VARIABLES  $X$ 
REQUIRES ...
ENSURES ...
WHILE  $0 < X$  DO
   $X := X - 1$ ;
OD;
```

$\longrightarrow \longrightarrow$

→ → →

→ → → →

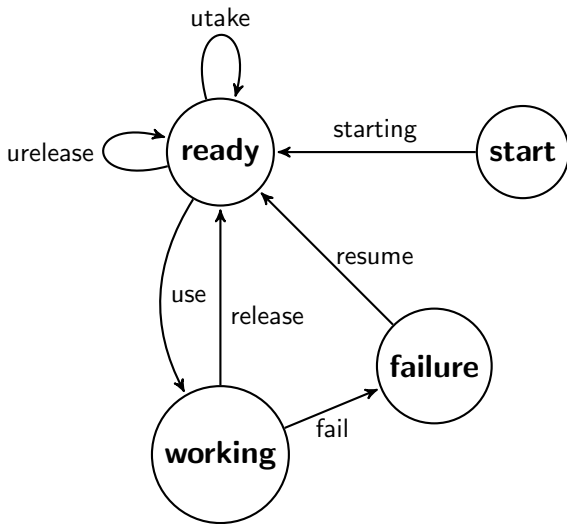
→ → → →

Modélisation, spécification et vérification
CM1 (9 octobre 2024) (Dominique Méry)

① Prolégomènes

② Modélisation de programmes et de systèmes logiciels

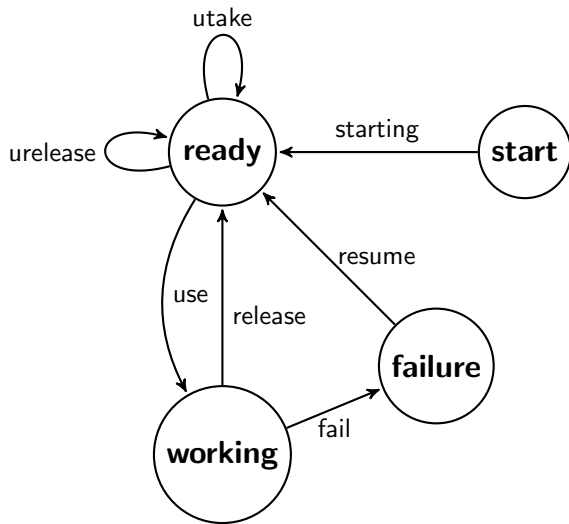
③ Modélisation relationnelle



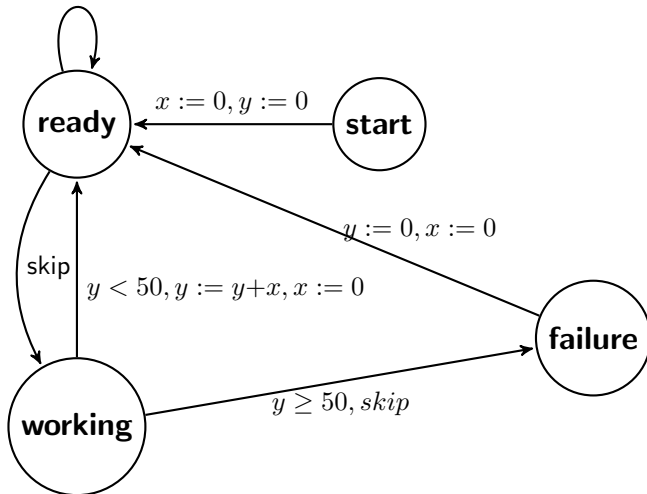
```
ℓ0[Q := 0];  
ℓ1[R := X];  
IF ℓ5[Y > 0]  
    WHILE ℓ2[R ≥ Y]  
        ℓ3[Q := Q+1];  
        ℓ4[R := R-Y]  
    ENDWHILE  
ELSE  
    ℓ6[skip]  
ENDIF
```



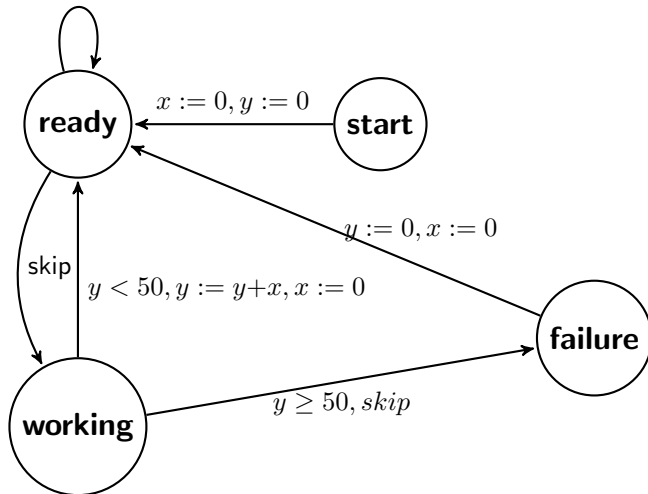
- ▶ Un automate a des états de contrôle : compteur ordinal d'un programme
- ▶ Un automate a des étiquettes : événements, actions, ...
- ▶ Un automate peut aussi avoir des variables explicites qui sont modifiées par des actions
- ▶ Un automate décrit des exécutions possibles qui sont des chemins suivant les informations de l'automate.



$x \leq 5, x := x+1$

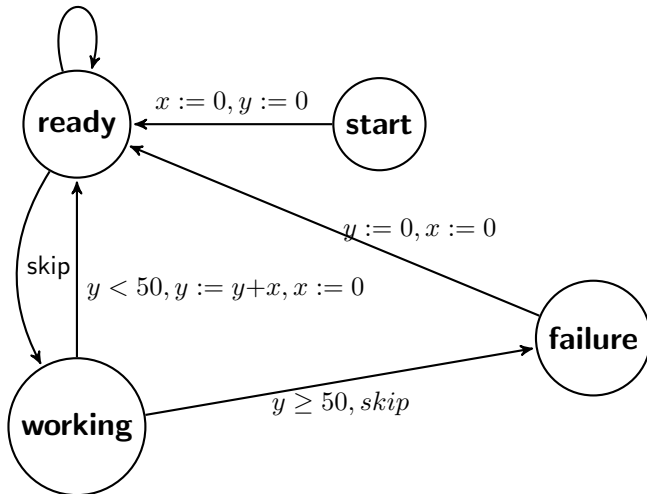


$x \leq 5, x := x+1$



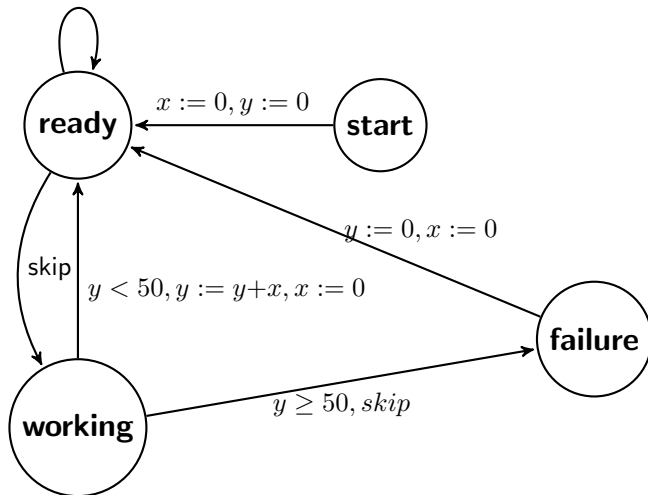
► **safety1** : $0 \leq x \leq 5$

$x \leq 5, x := x+1$



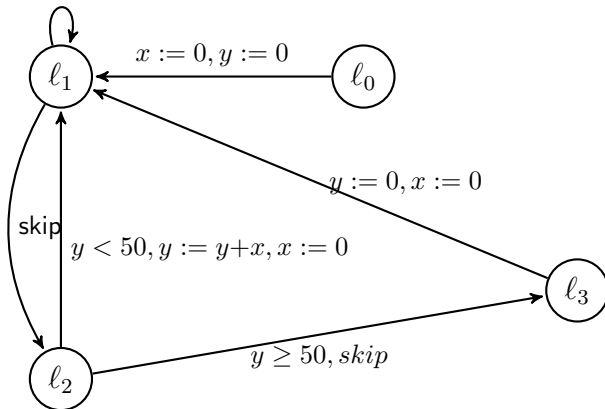
► **safety1** : $0 \leq x \leq 5$ et ...

$x \leq 5, x := x+1$

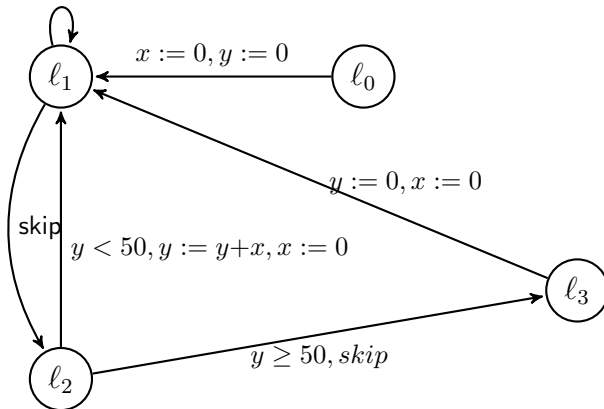


► safety1 : $0 \leq x \leq 5$ et ... safety2 : $0 \leq y \leq 56$

$x \leq 5, x := x+1$

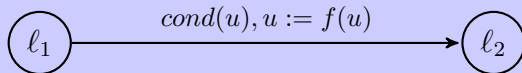


$x \leq 5, x := x+1$

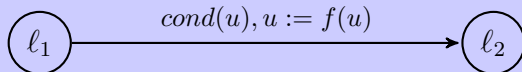


- ▶ $safety1 : 0 \leq x \leq 5$ et $safety2 : 0 \leq y \leq 56$
- ▶ $skip = x := x, y := y$
- ▶ $skip = TRUE, x := x, y := y = TRUE, skip$

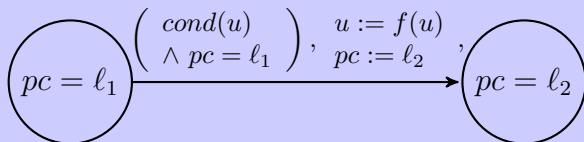
Transition entre deux états de contrôle



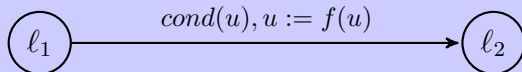
Transition entre deux états de contrôle



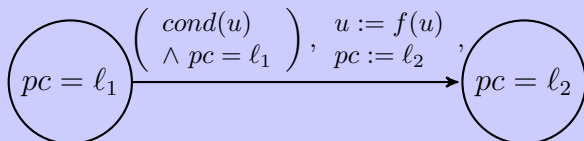
Transition entre deux états de contrôle



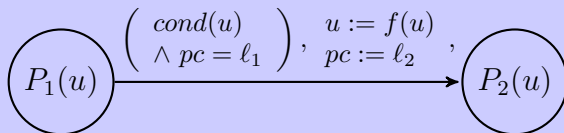
Transition entre deux états de contrôle



Transition entre deux états de contrôle



Transition entre deux prédicats



① Prolégomènes

② Modélisation de programmes et de systèmes logiciels

③ Modélisation relationnelle

Un modèle relationnel \mathcal{MS} pour un système \mathcal{S} est une structure

$$(Th(s, c), X, \text{VALS}, \text{Init}(x), \{r_0, \dots, r_n\})$$

où

- ▶ $Th(s, c)$ est une théorie définissant les ensembles, les constantes et les propriétés statiques de ces éléments.
- ▶ X est une liste de variables flexibles.
- ▶ VALS est un ensemble de valeurs possibles pour X .
- ▶ $\{r_0, \dots, r_n\}$ est un ensemble fini de relations reliant les valeurs avant x et les valeurs après x' .
- ▶ $\text{Init}(x)$ définit l'ensemble des valeurs initiales de X .
- ▶ la relation r_0 est la relation $Id[\text{VALS}]$, identité sur VALS .

☒ Definition

Soit $(Th(s, c), X, VALS, Init(x), \{r_0, \dots, r_n\})$ un modèle relationnel d'un système \mathcal{S} . La relation Next associée à ce modèle est définie par la disjonction des relations r_i :

$$\text{Next} \stackrel{\text{def}}{=} r_0 \vee \dots \vee r_n$$

.....
pour une variable x , nous définissons les valeurs suivantes :

- ▶ x est la valeur courante de la variable X.
- ▶ x' est la valeur suivante de la variable X.
- ▶ x_0 ou \underline{x} sont la valeur initiale de la variable X.
- ▶ \bar{x} ou x_f est la valeur finale de la variable X, quand cette notion a du sens.

.....

☒ Definition(assertion)

Soit $(Th(s, c), X, VALS, Init(x), \{r_0, \dots, r_n\})$ un modèle relationnel M d'un système \mathcal{S} . Une propriété A est une propriété assertionnelle de sûreté pour le système \mathcal{S} , si

$$\forall x_0, x \in VALS. Init(x_0) \wedge Next^*(x_0, x) \Rightarrow A(x).$$

.....

.....

☒ Definition(relation)

Soit $(Th(s, c), X, VALS, Init(x), \{r_0, \dots, r_n\})$ un modèle relationnel M d'un système \mathcal{S} . Une propriété R est une propriété relationnelle de sûreté pour le système \mathcal{S} , si

$$\forall x_0, x \in VALS. Init(x_0) \wedge Next^*(x_0, x) \Rightarrow R(x_0, x).$$

.....

- ▶ P. et R. Cousot développent une étude complète des propriétés d'invariance et de sûreté en mettant en évidence correspondances entre les différentes méthodes ou systèmes proposées par Turing, Floyd, Hoare, Wegbreit, Manna ... et reformulent les principes d'induction utilisés pour définir ces méthodes de preuve (voir les deux cubes des 16 principes).