Chapitre 2

Machines d'état et correction de programmes

Plan

1. Machine d'état
2. Principe d'invariant
3. Correction de programmes
4. Variables dérivées
5. Correction automatique

Lectures conseillées :

- MCS : 5.4, 7.2.
- Leslie Lamport. Computation and state machines. 2008
  `http://research.microsoft.com/en-us/um/people/lamport/pubs/state-machine.pdf`

Principe d'invariant

La démonstration du taquin est basée sur le principe d'invariant.

Idée générale :

- On étudie un processus qui évolue étape par étape
- On souhaite montrer qu'une propriété est vérifiée à chaque étape du processus. On appelle ce type de propriété un invariant

La vérification de l'invariant se fait par induction sur le nombre d'étapes du processus.

Un processus qui évolue étape par étape peut se modéliser par une machine d'état.

Machine d'état

Un machine d'état est un modèle abstrait d'un processus évoluant pas à pas.

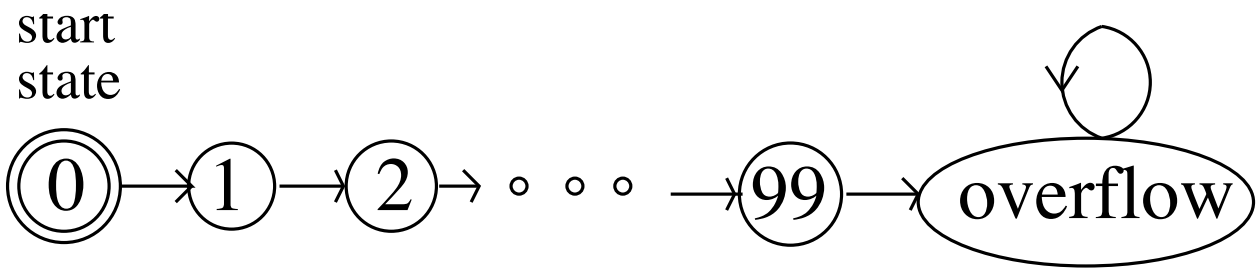
Définition : Une *machine d'état* est un triplet $(\mathcal{Q}, \mathcal{Q}_0, \delta)$ où :

- $\mathcal{Q}$ est un ensemble (non vide) d'états (potentiellement infini),
- $\mathcal{Q}_0 \subseteq \mathcal{Q}$ est l'ensemble (non vide) des états initiaux,
- $\delta \subseteq \mathcal{Q} \times \mathcal{Q}$ est l'ensemble des transitions permises.

Une machine d'état peut être interprétée comme un graphe dirigé dont les sommets sont les états et les arêtes (dirigées) sont les transitions.

Dans la suite, on notera $q \longrightarrow r$ une transition (q, r).

Exemples



Un compteur :

- $\mathcal{Q} = \{0, 1, \ldots, 99, \mathit{overflow}\}$
- $\mathcal{Q}_0 = \{0\}$
- $\delta = \{n \rightarrow n+1 | 0 \leq n < 99\} \bigcup \{99 \rightarrow \mathit{overflow}, \mathit{overflow} \rightarrow \mathit{overflow}\}$

Le taquin :

- $\mathcal{Q}$ =l'ensemble des configurations possibles du taquin, $|\mathcal{Q}| = 16!$
- $\mathcal{Q}_0 = \{$la configuration initiale$\}$
- $\delta = \{\langle$config1, config2$\rangle |$ config2 peut s'obtenir à partir de config1 en un mouvement$\}$

Exécution et déterminisme

Définition : Une *exécution* ou *trajectoire* d'une machine d'état est une séquence (potentiellement infinie) d'états $s_0, s_1, s_2 \ldots$ telle que :

- $s_0 \in \mathcal{Q}_0$
- $(s_i, s_{i+1}) \in \delta, \forall i \geq 0$

Définition : Une machine d'état est *déterministe* si et seulement si il n'y a qu'un état initial ($|\mathcal{Q}_0| = 1$) et la relation δ est une fonction, c'est-à-dire si pour tout état $s \in \mathcal{Q}$, il y a au plus un état $t \in \mathcal{Q}$ tel que $(s, t) \in \delta$.

Exemples :

- Le compteur est déterministe. Une seule exécution possible :

$$0, 1, 2, \ldots, 99, \textit{overflow}, \textit{overflow} \ldots$$

- Le taquin ne l'est pas (plusieurs mouvements possibles par état).

Atteignabilité et invariant

Définition : Un état r est *atteignable* s'il est contenu dans une exécution de longueur finie.

Définition : Un *invariant* est un prédicat $P(s)$ défini sur $\mathcal{Q}$ qui est vrai pour tous les états *atteignables*.

Exemple : le prédicat "la parité du nombre d'inversions est différente de la parité du numéro de la ligne contenant la case vide" est un invariant pour le taquin.

Démonstration d'un invariant

Théorème : Soit une machine d'état $\mathcal{M} = (\mathcal{Q}, \mathcal{Q}_0, \delta)$. Si P est un prédicat sur $\mathcal{Q}$ tel que :

- $P(s)$ est vrai pour tout $s \in \mathcal{Q}_0$
- et $P(s) \Rightarrow P(s')$ est vrai pour tout $(s, s') \in \delta$,

alors P est un invariant pour $\mathcal{M}$.

Démonstration :

- La preuve se fait par induction sur la longueur d'exécution.
- Soit $Q(n)$ le prédicat suivant :[1]

  "$P(s)$ est vrai pour tout s contenu dans une exécution de longueur n".

- Cas de base : $Q(1)$ est vrai puisque toute exécution de longueur 1 contient seulement un état initial et $P(s)$ est vrai pour tout $s \in \mathcal{Q}_0$.

[1] Ne pas confondre P et Q !

- Cas inductif :
  - Supposons $Q(n)$ vrai et montrons que $Q(n+1)$ est vrai également.
  - Soit r' un état contenu dans une exécution de longueur $n+1$. Il suffit de montrer que $P(r')$ est vrai.
  - Si r' est le premier état de l'exécution, alors $P(r')$ est vérifié.
  - Sinon, soit r l'état précédant r' dans l'exécution. r est contenu dans une exécution de longueur n et donc, par hypothèse inductive, $P(r)$ est vrai. Puisque $(r, r') \in \delta$, $P(r)$ implique que $P(r')$ est vrai.
- $P(s)$ est donc vrai pour tout état s contenu dans une exécution de longueur finie, ce qui correspond aux états atteignables de la machine.
- P est donc un invariant pour $\mathcal{M}$. □

Invariant inductif

- Définition : Un invariant qui peut se démontrer par le théorème précédent s'appelle un *invariant inductif*.
- Tous les invariants ne peuvent pas se démontrer de cette manière.

- Pour démontrer un invariant $P(s)$, il suffit de trouver un invariant (inductif) $Q(s)$ tel que :
  - $Q(s)$ est vrai pour tout $s \in \mathcal{Q}_0$
  - $Q(s) \Rightarrow Q(s')$ est vrai pour tout $(s, s') \in \delta$,
  - $Q(s) \Rightarrow P(s)$ est vrai pour tout état s atteignable

- Exemple : Dans l'exemple du taquin :
  - $P(n) =$"Après n mouvements, la grille cible n'est pas atteinte" est un invariant mais pas un invariant inductif ($P(n+1)$ ne peut pas se déduire de $P(n)$).
  - $Q(n) =$"Après n mouvements, la parité du nombre d'inversions est différente de la parité du numéro de la ligne contenant la case vide" est un invariant inductif (tel que pour tout n, $Q(n) \Rightarrow P(n)$).

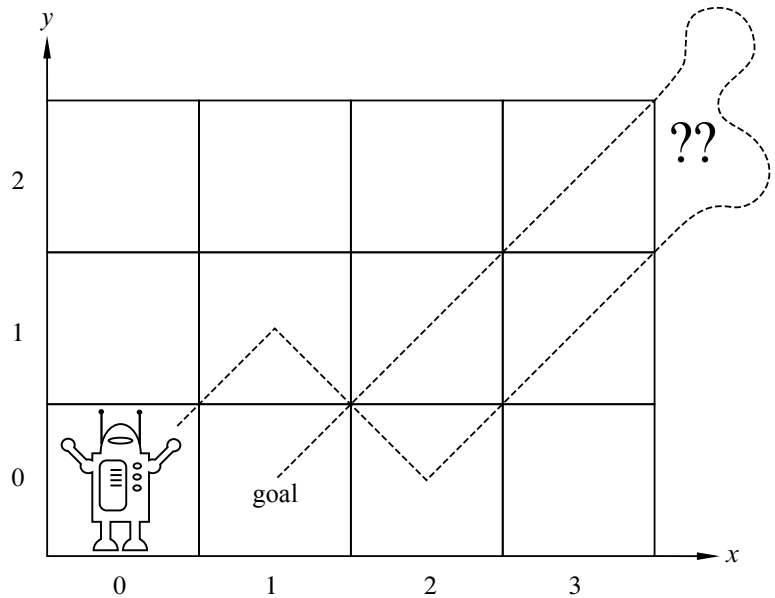
Application

Soit un robot se déplaçant sur une grille entière à deux dimensions :

- L'état du robot est spécifié par les coordonnées entières (x, y) représentant sa position courante sur la grille. $\mathcal{Q} = \mathbb{Z} \times \mathbb{Z}$
- Le robot se trouve initialement à l'origine $\mathcal{Q}_0 = \{(0, 0)\}$
- A chaque pas, le robot peut se déplacer uniquement en diagonale

$$\delta = \{(m, n) \rightarrow (m \pm 1, n \pm 1) | (m, n) \in \mathcal{Q}\}$$

$\Rightarrow$ On cherche à déterminer si le robot peut atteindre l'état $(1, 0)$.



Théorème : La somme des coordonnées de tout état atteignable par le robot est pair.

Démonstration :

- Soit le prédicat $Q((m,n)) =$"$m+n$ est pair". Montrons que $Q((m,n))$ est un invariant inductif.
- Cas de base : $Q((0,0))$ est vrai.
- Cas inductif :
  - Nous devons montrer que $Q((m,n)) \Rightarrow Q((m',n'))$ est vrai pour toute transition $(m,n) \rightarrow (m',n') \in \delta$.
  - Toute transition étant de la forme $(m,n) \rightarrow (m \pm 1, n \pm 1)$, la somme des coordonnées est augmentée de 0, 2 ou -2 à chaque transition.
  - Ajouter 0, 2 ou -2 à un nombre pair donne un nombre pair.
- Par le théorème d'invariant, on peut conclure que P est un invariant, ce qui démontre le théorème. □

Corrolaire : Le robot n'atteindra jamais la position $(1, 0)$.
Démonstration : C'est une conséquence directe du théorème puisque $1 + 0$ est impair. □

Remarque : L'invariant qu'on veut montrer est $P((m, n)) =$"$(m, n) \neq (1, 0)$" qui n'est pas inductif. On montre que c'est une conséquence de l'invariant inductif $Q((m, n)) =$"$m + n$ est pair".

Exercice : Réécrivez la démonstration du problème du taquin en vous appuyant sur le théorème d'invariant.

Le problème des cruches

Soient deux cruches non graduées et initialement vides, de contenances respectives 3 et 6 litres.

Seules les trois actions suivantes sont possibles :

- remplissage d'une cruche via la fontaine,
- vidage d'une cruche dans la fontaine,
- transvasement d'une cruche vers l'autre jusqu'à ce que l'une soit remplie ou que l'autre soit vide.

Question : Est-il possible d'obtenir exactement 4 litres dans la grande cruche ?

Modélisation par une machine d'état

- Un état est défini par la quantité d'eau dans chacune des cruches

$$\mathcal{Q} = \{(a, b) | 0 \leq a \leq 3, 0 \leq b \leq 6, a, b \in \mathbb{R}\}$$

- Initialement, les cruches sont vides

$$\mathcal{Q}_0 = \{(0, 0)\}$$

- L'ensemble des transitions est l'union de
  - $(a, b) \rightarrow (3, b)$ (remplissage petite cruche)
  - $(a, b) \rightarrow (a, 6)$ (remplissage grande cruche)
  - $(a, b) \rightarrow (0, b)$ (vidage petite cruche)
  - $(a, b) \rightarrow (a, 0)$ (vidage grande cruche)
  - $(a, b) \rightarrow (\max(0, a - (6 - b)), \min(a + b, 6))$ (transvasement petite vers grande cruche)
  - $(a, b) \rightarrow (\min(a + b, 3), \max(0, b - (3 - a)))$ (transvasement grande vers petite cruche)

(cette machine est non déterministe)

Réponse à l'énigme

Théorème : Il n'est pas possible de remplir une des deux cruches avec exactement 4 litres

Démonstration :

- Soit le prédicat $P((a, b)) =$"a et b sont des multiples de 3".
- Il suffit de montrer que $P((a, b))$ est un invariant pour la machine d'état :
  - $P((0, 0))$ est vrai
  - "$P((a, b)) \Rightarrow P((c, d))$ pour tout $(a, b) \longrightarrow (c, d) \in \delta$" se montre en traitant au cas par cas les 6 types de transition

    (Laissé comme exercice)
- 4 n'étant pas un multiple de 3, le théorème est vérifié. □

Programme informatique et machine d'état

- Un programme informatique (itératif) peut être modélisé par une machine d'état (déterministe)
- Inversément, un programme informatique peut être vu comme une manière de décrire une machine d'état.

- L'état de la machine est défini par les valeurs des variables utilisées par le programme[2].
- Les transitions sont définies par les instructions du programme.

[2] plus, si nécessaire, certaines variables "cachées" telles que le numéro de l'instruction, le contenu de la pile d'appel des fonctions, etc.

Exemple

Soit le programme suivant (en notation pseudo-code) :

```
1  x = a ; y = b
2  while x ≠ y
3        if x > y
4              x = x − y
5        elseif x < y
6              y = y − x
```

La machine d'état $(\mathcal{Q}, \mathcal{Q}_o, \delta)$ correspondante est définie par :

- $\mathcal{Q} = \{(x, y) | x, y \in \mathbb{Z}\}$
- $\mathcal{Q}_0 = \{(a, b)\}$
- $\delta =$

$$(x, y) \rightarrow \begin{cases} (x - y, y) & \text{si } x > y \\ (x, y - x) & \text{si } x < y \end{cases}$$

Etats terminaux

Définition : Un état d'une machine d'état est *terminal* s'il n'existe aucune transition partant de cet état.

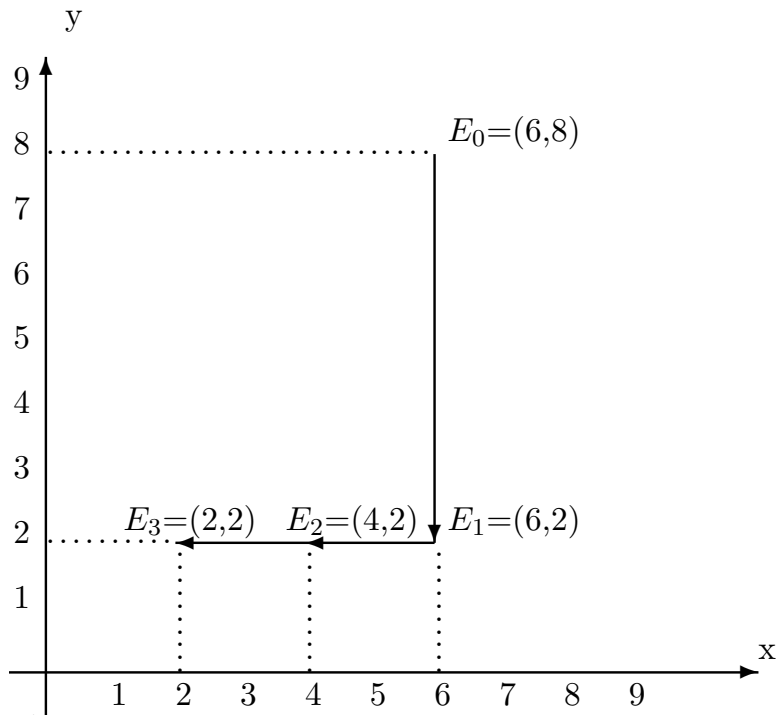
Dans le contexte de l'étude de programmes informatiques, on n'étudiera que deux types d'exécutions :

- Les exécutions de longueur infinie
- Les exécutions de longueur finie dont le dernier état est un état terminal

Les premières correspondent à des boucles infinies, les secondes à des programmes qui se terminent.

La nature de l'exécution varie en général en fonction de l'état initial.

Exemples d'exécution de la machine du transparent 91



(De Marneffe)

- En partant de $(6, 8)$, la machine finit par s'arrêter en $(2, 2)$ (plus de transition possible)
- En partant de $(-4, -2)$, l'exécution (infinie) est $(-4, -2)$, $(-4, 2)$, $(-4, 6)$, $(-4, 10)$,. . ..
- En partant de $(4, 0)$, la machine reste dans cet état et ne s'arrête jamais.

Précondition et postcondition

Parmi les exécutions de la machine qui se terminent, seules certaines sont intéressantes.

Elles sont définies par deux prédicats :

- **Postcondition :** prédicat qui doit être vérifié par les états terminaux solutions du problème posé.
  - Exemple : $Post((x, y)) =$"x est le plus grand commun diviseur de a et b"
- **Précondition :** définit un ensemble d'états[3] qui, étant les états initiaux de la machine, donneront des exécutions finies dont les états terminaux vérifient la postcondition.
  - Exemple : $Pre((x, y)) =$"$x > 0$ et $y > 0$".

Remarque : on appelle *précondition la plus faible* le prédicat qui définit l'ensemble de *tous* les états initiaux qui mèneront à un état terminal vérifiant la postcondition.

[3] ceux qui vérifient le prédicat

Correction de programmes

Un programme est *totalement correct* s'il vérifie deux propriétés :

- **Correction partielle :** Si le programme atteint un état terminal à partir d'un état initial vérifiant la précondition, alors cet état satisfait la postcondition.
- **Terminaison :** Le programme atteint toujours un état terminal à partir d'un état initial vérifiant la précondition

L'un n'implique pas l'autre. Le programme suivant est seulement partiellement correct :

```
1  while x ≠ y
2        y = y + 1
```

$$Pre((x, y)) = \text{“}x, y \in \mathbb{Z}\text{”}$$

$$Post((x, y)) = \text{“}x = y\text{”}$$

La correction partielle se montre généralement par le principe d'invariant. La terminaison se montre par le principe du bon ordre.

Exponentiation

```
1  x = a ; y = 1 ; z = b
2  while z ≠ 0
3        r = REMAINDER(z, 2)
4        z = QUOTIENT(z, 2)
5        if r = 1
6              y = xy
7        x = x^2
```

$Pre((x, y, z)) =$ "$x \in \mathbb{R}, z \in \mathbb{Z}$"

$Post((x, y, z)) =$ "$y = a^b$"

$(\text{REMAINDER}(q, r) = q \bmod r$ et $\text{QUOTIENT}(q, r) = \lfloor \frac{q}{r} \rfloor)$

Machine d'état :

- $\mathcal{Q} = \mathbb{R} \times \mathbb{R} \times \mathbb{N}$
- $\mathcal{Q}_0 = \{(a, 1, b)\}$
- $\delta =$

$$(x, y, z) \rightarrow \begin{cases} (x^2, y, \text{QUOTIENT}(z, 2)) & \text{si } z \neq 0 \text{ et z pair} \\ (x^2, xy, \text{QUOTIENT}(z, 2)) & \text{si } z \neq 0 \text{ et z impair} \end{cases}$$

Exponentiation : correction partielle

Théorème : Si l'algorithme se termine, on a $y = a^b$ pour $a \in \mathbb{R}$ et $b \in \mathbb{Z}$.

Démonstration :

Soit le prédicat :

$$P((x, y, z)) = \text{“}z \in \mathbb{N} \text{ et } yx^z = a^b\text{”}.$$

Montrons que P est un invariant inductif pour la machine d'état.

Cas de base : $P((a, 1, b))$ est vérifié puisque $1 \cdot a^b = a^b$.

Cas inductif : Soit une transition $(x, y, z) \to (x', y', z') \in \delta$. Supposons que $P((x, y, z))$ est vérifié et montrons que $P((x', y', z'))$ est vérifié, c'est-à-dire

$$z' \in \mathbb{N} \text{ et } y'x'^{z'} = a^b.$$

Puisqu'il y a une transition depuis (x, y, z), on a $z \neq 0$ et puisque $z \in \mathbb{Z}$, on peut considérer deux cas :

- Cas 1 : z est pair. On a alors $x' = x^2, y' = y, z' = z/2$. Donc, $z' \in \mathbb{Z}$ et
$$y'x'^{z'} = y(x^2)^{z/2} = yx^{2z/2} = yx^z = a^b$$
(car $P((x, y, z))$ est vérifié).
- Cas 2 : z est impair. On a alors $x' = x^2, y' = xy, z' = (z-1)/2$. Donc, $z' \in \mathbb{Z}$ et
$$y'x'^{z'} = xy(x^2)^{(z-1)/2} = yx^{1+2(z-1)/2} = yx^z = a^b.$$

Par le théorème d'invariant, on conclut que P est vérifié pour tout état atteignable.

Etant donné le gardien de la boucle, les seuls états terminaux atteignables sont ceux pour lesquels $z = 0$. Donc, si un état terminal est atteint, par l'invariant, on a $y = yx^0 = a^b$. □

Interprétation

La preuve précédente suit le schéma de démonstration du transp. 83 :

- On veut montrer que le prédicat :

$$\begin{aligned} Q((x,y,z)) &= \text{"}(x,y,z) \text{ terminal} \Rightarrow Post((x,y,z))\text{"} \\ &= \text{"}z = 0 \Rightarrow y = a^b \text{ "} \end{aligned}$$

  est un invariant pour la machine d'état

- Q n'étant pas un invariant inductif, on passe par un invariant inductif :

$$P((x,y,z)) = \text{“}z \in \mathbb{N} \text{ et } yx^z = a^b\text{"}\,,$$

  qui est tel que pour tout état atteignable $(x,y,z) \in \mathcal{Q}$:

$$P((x,y,z)) \Rightarrow Q((x,y,z))$$

Exponentiation : terminaison

Théorème : L'algorithme d'exponentiation se termine toujours en un état où $z = 0$.

Démonstration (pédantique) :

- Après chaque transition, z devient strictement plus petit (par l'instruction $z = \text{QUOTIENT}(z, 2)$).
- Or par l'invariant, z est un entier naturel. Soit $S \subseteq \mathbb{N}$ l'ensemble des valeurs de z aux états atteignables par la machine.
- Par le principe du bon ordre, S possède une valeur minimale z_0.
- Vu la décroissance stricte de z, la machine finira toujours par atteindre cette valeur et s'arrêtera (par l'absurde, si ce n'était pas le cas, une transition à partir de cet état diminuerait encore z et donc z_0 ne serait pas le minimum).
- La seule manière de s'arrêter est d'avoir $z = 0$. On a donc $z_0 = 0$. □

Exercices

Montrez par induction forte que le nombre de transition de la machine d'état de l'algorithme d'exponentiation s'arrêtera après $\lceil \log_2 n \rceil + 1$ transitions à partir d'un état où $z = n \neq 0 \in \mathbb{N}$.

Montrer que l'algorithme du transparent 91 est totalement correct pour les précondition et postcondition suivantes :

$$Pre((x, y)) = "x > 0 \text{ et } y > 0"$$

$$Post((x, y)) = "x \text{ est le plus grand commun diviseur de } a \text{ et } b"$$

Variables dérivées

Définition : Une *variable dérivée* pour une machine $\mathcal{M} = (\mathcal{Q}, \mathcal{Q}_O, \delta)$ est une fonction $f : \mathcal{Q} \to S$ où S est un ensemble quelconque (typiquement $\mathbb{N}$).

Une variable dérivée est aussi appelée un variant ou une fonction potentielle.

Exemple : La coordonnée x du robot ou le nombre d'inversions du taquin,

Définition : Une variable dérivée $f : \mathcal{Q} \to \mathbb{R}$ est *strictement décroissante* si et seulement si $q \to q' \in \delta$ implique $f(q') < f(q)$. Elle est *faiblement décroissante* si et seulement si $q \to q' \in \delta$ implique $f(q') \leq f(q)$.

Variables dérivées et terminaison

Théorème : Si $f : Q \to \mathbb{N}$ est une variable dérivée strictement décroissante pour une machine d'état, alors le nombre de transitions d'une exécution démarrant dans un état q_0 est au plus $f(q_0)$.
Démonstration : Par contradiction. Supposons qu'il existe une exécution $q_0, q_1, \ldots, q_t, \ldots$ où $t > f(q_0)$. Alors $f(q_t) \leq f(q_0) - t < 0$, puisque la valeur de f décroît d'au moins 1 à chaque transition, ce qui contredit que $f(q) \in \mathbb{N}$. □

Remarques :

- Une fonction faiblement décroissante n'assure évidement pas la terminaison.
- La fonction f est parfois appelée *fonction de terminaison* pour la machine d'état.
- Le théorème peut se généraliser à des variables dérivées prenant leur valeur dans un ensemble bien ordonné.

Un exemple plus compliqué

Soit la machine d'état suivante :

- $\mathcal{Q} \in \mathbb{N} \times \mathbb{N}$
- $\mathcal{Q}_0 = \{(a, b)\}$ avec $a, b \in \mathbb{N}$
- $\delta = \{(x, y) \rightarrow (x - 1, y) | \forall x, y \in \mathbb{N} : x > 0\} \bigcup \{(x, y) \rightarrow (z, y - 1) | \forall x, y, z \in \mathbb{N} : y > 0, z \geq x\}$

qui modélise par exemple les déplacements d'un robot dans un espace discret à deux dimensions.

Théorème : Quel que soit son état initial, la machine finira toujours par atteindre l'état $(0, 0)$.

Démonstration :

- Soit la variable dérivée $v : \mathcal{Q} \to \mathbb{N} + To1$:

$$v((x, y)) = y + \frac{x}{x+1}.$$

- Chaque transition $(x, y) \to (x', y')$ est telle que $v((x', y')) < v((x, y))$ et donc v est une variable strictement décroissante.
- L'ensemble $\mathbb{N} + To1$ étant bien ordonné (voir transp. 41), v finira par atteindre son minimum.
- Le seul état sans transition étant $(0, 0)$, la machine ne peut s'arrêter que dans cet état. □

Vérification formelle de programmes

- La méthode d'invariant et le principe du bon ordre permettent :
  - de prouver formellement la correction et la terminaison de programmes
  - de vérifier que certaines propriétés sont vérifiées tout au long de l'exécution d'un programme
- Cette vérification est cruciale dans de nombreuses applications critiques de l'informatique (domaines médical, spatial, sécurité, etc.)

- Idéalement, on aimerait que ces vérifications formelles puissent se faire de manière automatique
- Des outils d'aide à la vérification existent mais ces outils ne peuvent automatiquement :
  - inventer les invariants de boucle
  - prouver un invariant donné dans les cas non triviaux
  - prouver dans tous les cas la terminaison des programmes

Invariant difficile à trouver

Que fait le programme suivant ?

```
ALGO(A)
1  i = 1
2  j = A.length
3  while i < j
4        if A[i] > A[j]
5              SWAP(A[i], A[i + 1])
6              SWAP(A[i], A[j])
7              i = i + 1
8        else j = j − 1
```

Terminaison difficile à prouver

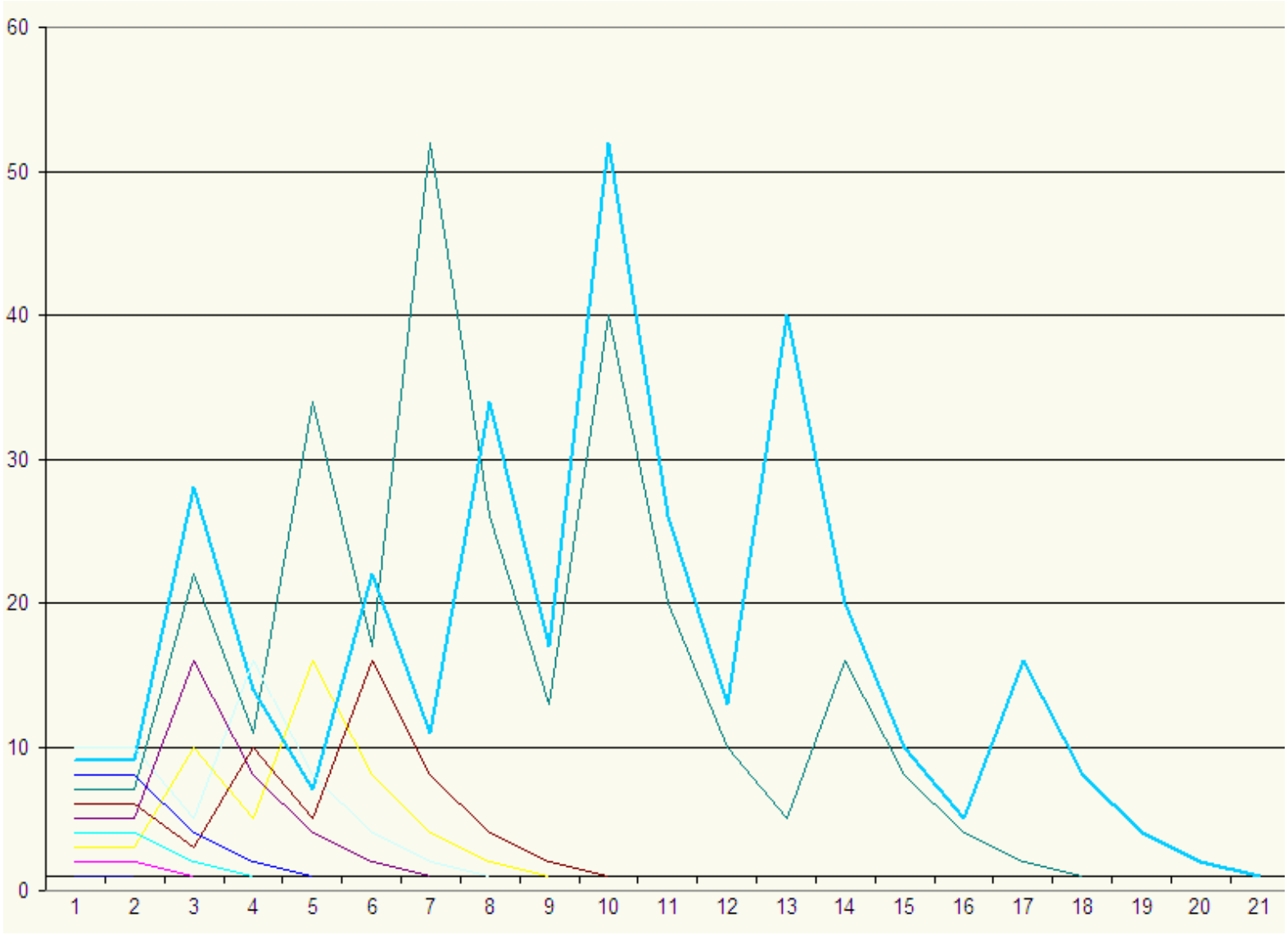
Conjecture de Syracuse : Le programme suivant :

```
ALGO(n)
1  while n ≠ 1
2        if n pair
3              n = n/2
4        else n = 3n + 1
```

se termine pour tout $n \in \mathbb{N}^+$.

On a montré expérimentalement que la conjecture était vraie pour tout $n < 1,25 \cdot 2^{62}$ mais à ce jour elle n'a toujours pas été prouvée.

Exemples d'exécutions



Source : `animalerienumerique.blogspot.be`

Problème de l'arrêt

En théorie de la calculabilité, le problème de l'arrêt consiste à déterminer si un programme informatique se termine ou non.

Ce problème n'est pas décidable : on peut montrer qu'il n'est pas possible d'écrire un programme informatique prenant comme entrée un programme quelconque et renvoyant VRAI si le programme s'arrête, FAUX sinon.

On en donnera ici qu'un preuve informelle. Voir le cours "Introduction à la calculabilité" pour une preuve plus rigoureuse.

Problème de l'arrêt

Théorème : Il n'existe pas d'algorithme CheckHalt qui prend en entrée un programme P et des données D et renvoit "halts" si P s'arrête après un nombre fini d'étapes quand il est appliqué aux données D, "loops forever" sinon.

Démonstration :

- Par contradiction. Supposons qu'un tel algorithme CheckHalt existe.
- Un programme P peut lui-même être considéré comme une entrée (une chaîne de caractères). On peut donc appliquer CheckHalt sur l'entrée (P, P).
- Soit l'algorithme Test prenant en entrée un programme P et défini comme suit :

```
Test(P)
1  if CheckHalt(P, P) == "halts"
2        "exécuter une boucle infinie"
3  else
4        return
```

- Appliquons maintenant Test à lui-même. Deux résultats possibles :
  - Test(Test) se termine : la valeur de CheckHalt(Test, Test) est alors “halts” et donc Test(Test) boucle à l'infini.
  - Test(Test) boucle à l'infini : CheckHalt(Test, Test) renvoie “loops forever” et donc Test(Test) se termine.
- Cette analyse montre que Test(Test) boucle et aussi se termine.
- Cette contradiction montre qu'il n'est pas possible d'écrire une fonction CheckHalt. □

Résumé

- Une machine d'état permet de modéliser un processus calculatoire.
- La technique d'invariant inductif permet de prouver des propriétés d'une machine d'état.
- Un programme informatique peut être modélisé par une machine d'état.
- La correction partielle se prouve par invariant et la terminaison par le principe du bon ordre.

- Les machine d'état sont très utilisées en informatique (compilateur, théorie de la calculabilité, protocoles réseaux, processus de décision etc.).
- Beaucoup de variantes existent (automates, machine de turing, probabilistes etc.).