Jean Goubault-Larrecq

Calculabilité

5. Fonctions récursives

Aujourd'hui

- Fonctions récursives primitives
- La hiérarchie de Grzegorczyk
- La fonction d'Ackermann-Péter

- Fonctions récursives partielles (générales)
- Equivalence avec les machines de Turing
- Basé sur le poly d'Hubert Comon (suite):
  http://www.lsv.fr/~comon/Calculabilite1/rec-primitive.pdf
  http://www.lsv.fr/~comon/Calculabilite1/poly-re.pdf

Fonctions primitives récursives

Fonctions primitives récursives

- **Fonctions primitives récursives:**
  Une classe de fonctions de $\mathbf{N}^k$ vers **N**
  inventée par Kurt Gödel
  pour décrire une notion de calcul
  (avant Turing)

- … de même que la notion plus générale de **fonction récursive** (partielle)

à ne pas confondre avec une fonction qui s'appelle elle-même 😕

Par Auteur inconnu — Familienalbum der Familie Gödel,
Scan from Gianbruno Guerrerio,
Kurt Gödel - Logische Paradoxien und mathematische Wahrheit, S.24,
Domaine public,
https://commons.wikimedia.org/w/index.php?curid=1059569

Fonctions primitives récursives

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $\quad$ $Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)
    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)
    $P_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1 \le i \le k$, $(n_1,\ldots,n_k) \mapsto n_i$)
- et stable par **composition** et **récursion primitive** (transparents suivants)
- Il s'agit d'une famille de fonctions **totales** (applications)

Composition

- Composition de fonctions à **plusieurs** arguments

- Si $g_1, \ldots, g_m : \mathbf{N}^k \to \mathbf{N}$ $(m \geq 0)$,
  et $f : \mathbf{N}^m \to \mathbf{N}$,
  $\mathrm{Comp}_m(f, g_1, \ldots, g_m) : \mathbf{N}^k \to \mathbf{N}$
  $\boldsymbol{n} \mapsto f(g_1(\boldsymbol{n}), \ldots, g_m(\boldsymbol{n}))$
  (On notera $\boldsymbol{n}$ pour $(n_1,\ldots,n_k)$)

- Autre notation: $f \circ (g_1, \ldots, g_m)$

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $\quad Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)
    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)
    $P_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1 \leq i \leq k$, $(n_1,\ldots,n_k) \mapsto n_i$)
  - et stable par **composition** et **récursion primitive**

Récursion primitive

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $\quad Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)
    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)
    $P_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1 \le i \le k$, $(n_1,\ldots,n_k) \mapsto n_i$)
  - et stable par **composition** et **récursion primitive**

- Intuitivement,

```
let rec f(n) = match n with
      | 0 -> b
      | S n' -> e (n', f(n'))
```
(* une expression dépendant de `n'` et `f(n')` *)

- Si $b : \mathbf{N}^k \to \mathbf{N}$, et $e : \mathbf{N}^{k+2} \to \mathbf{N}$,

  Prim(b,e) est la fonction totale $f : \mathbf{N}^{k+1} \to \mathbf{N}$ définie par:

  $f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$

  $f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

$\boldsymbol{m}$ est un k-uplet de **paramètres**

définition par récurence sur n

Boucles for

- « Les fonctions primitives sont celles qu'on peut définir à l'aide de boucles `for` (pas de boucle `while`) »
- Prim(*b*,*e*) s'implémente (aussi) par:

```
def f(m1,…,mk,n):
    res = b(m1,…,mk)
    for i in range(1,n):
        res = e(m1,…,mk,i-1,res)
    return res
```

Exemples: 1. addition

- On a: $m+0 = m$

  $m+S(n) = S(m+n)$

- On peut donc définir _+_ comme

  $\mathrm{Prim}(m \mapsto m,$ *b*

  $m,n,r \mapsto S(r))$ *e*

- = $\mathrm{Prim}(P_1^1, S \circ P_3^3)$

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $\quad Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)
    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)
    $P_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1 \le i \le k$, $(n_1,\ldots,n_k) \mapsto n_i$)
  - et stable par **composition** et **récursion primitive**

Si $b : \mathbf{N}^k \to \mathbf{N}$, et $e : \mathbf{N}^{k+2} \to \mathbf{N}$,
Prim(b,e) est la fonction totale $f : \mathbf{N}^{k+1} \to \mathbf{N}$ définie par:

$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$

$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

Exemples: 2. multiplication

- On a: $m \times 0 = 0$

  $m \times S(n) = (m \times n) + m$

- On peut donc définir _×_ comme

  $\mathrm{Prim}(m \mapsto 0,$ *b*

  $m,n,r \mapsto r+m)$ *e*

- $=$ $\mathrm{Prim}(Z, + \circ (P_3^3, P_3^1))$

  (où $+ \stackrel{\text{def}}{=} \mathrm{Prim}(P_1^1, S \circ P_3^3)$)

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $\quad Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)
    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)
    $P_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1 \le i \le k$, $(n_1,\ldots,n_k) \mapsto n_i$)
  - et stable par **composition** et **récursion primitive**

Si $b : \mathbf{N}^k \to \mathbf{N}$, et $e : \mathbf{N}^{k+2} \to \mathbf{N}$,
Prim(b,e) est la fonction totale $f : \mathbf{N}^{k+1} \to \mathbf{N}$ définie par:

$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$

$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

Exemples: 3. prédécesseur

- On pose: Pred(0) $\stackrel{\text{def}}{=}$ 0

  Pred(S(n)) $\stackrel{\text{def}}{=}$ n

- On définit Pred comme

  Prim($\mapsto 0$, b

  $n,r \mapsto n$) e

- = Prim(0, P_2^1)

Note: pas de récursion = e n'utilise pas r

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $\quad Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)

    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)

    $P_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1 \le i \le k$, $(n_1,\ldots,n_k) \mapsto n_i$)
  - et stable par **composition** et **récursion primitive**

Si $b : \mathbf{N}^k \to \mathbf{N}$, et $e : \mathbf{N}^{k+2} \to \mathbf{N}$,

Prim(b,e) est la fonction totale $f : \mathbf{N}^{k+1} \to \mathbf{N}$ définie par:

$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$

$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

Exemples: 4. soustraction (tronquée)

- On pose: $m \dot{-} n \stackrel{\text{def}}{=} \max(m-n,0)$

  On a: $m \dot{-} 0 \stackrel{\text{def}}{=} m$

  $m \dot{-} \mathrm{S}(n) \stackrel{\text{def}}{=} \mathrm{Pred}(m \dot{-} n)$

- On définit _ ∸ _ comme

  $\mathrm{Prim}(m \mapsto m,$ *b*

  $m,n,r \mapsto \mathrm{Pred}(r))$ *e*

- $=$ $\mathrm{Prim}(\mathrm{P}_1^1, \mathrm{Pred} \circ \mathrm{P}_3^3)$

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $\quad Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)

    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)

    $\mathrm{P}_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1 \le i \le k$, $(n_1,\ldots,n_k) \mapsto n_i$)
- et stable par **composition** et **récursion primitive**

Si $b : \mathbf{N}^k \to \mathbf{N}$, et $e : \mathbf{N}^{k+2} \to \mathbf{N}$,

Prim(b,e) est la fonction totale $f : \mathbf{N}^{k+1} \to \mathbf{N}$ définie par:

$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$

$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

Exemples: 5. test à zéro

- On pose: $Eq_0(n) \stackrel{\text{def}}{=} 1$ si $n=0$

  $Eq_0(n) \stackrel{\text{def}}{=} 0$ sinon

- On définit Eq_0 comme

  $Prim(\ \mapsto 1,$ *b*

  $n,r \mapsto 0)$ *e*

- $=$ $Prim(S \circ 0, Z \circ P_2^1)$

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)
    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)
    $P_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1 \le i \le k$, $(n_1,\ldots,n_k) \mapsto n_i$)
  - et stable par **composition** et **récursion primitive**

Si $b : \mathbf{N}^k \to \mathbf{N}$, et $e : \mathbf{N}^{k+2} \to \mathbf{N}$,
Prim(b,e) est la fonction totale $f : \mathbf{N}^{k+1} \to \mathbf{N}$ définie par:

$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$

$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

Exemples: 6. tests ≤, ≥, =

- On pose: Leq(m,n) $\stackrel{\text{def}}{=}$ 1 si $m \leq n$
  Leq(m,n) $\stackrel{\text{def}}{=}$ 0 sinon

- On définit Leq comme
  $\text{Eq}_0(\dot{-} \circ (P_2^1, P_2^2))$

- … et Geq comme
  $\text{Eq}_0(\dot{-} \circ (P_2^2, P_2^1))$

- … et Eq comme
  $\text{Eq}_0(+ \circ (\dot{-}, \dot{-} \circ (P_2^2, P_2^1)))$ (par exemple)

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)
    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)
    $P_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1 \leq i \leq k$, $(n_1,\ldots,n_k) \mapsto n_i$)
  - et stable par **composition** et **récursion primitive**

Si $b : \mathbf{N}^k \to \mathbf{N}$, et $e : \mathbf{N}^{k+2} \to \mathbf{N}$,
Prim(b,e) est la fonction totale $f : \mathbf{N}^{k+1} \to \mathbf{N}$ définie par:

$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$

$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

Exemples: 7. if then else

- On pose: If(m,n,p) $\stackrel{\text{def}}{=}$ n si $m\neq 0$
  If(m,n,p) $\stackrel{\text{def}}{=}$ p sinon

- On définit If' comme
  Prim($n,p \mapsto p$, *b*
  $n,p,m,r \mapsto n$) *e*
  (i.e., If'(n,p,m)= If(m,n,p):
  on doit mettre l'argument de récursion en dernier)

- … et If(m,n,p) $\stackrel{\text{def}}{=}$ If'(n,p,m), c'est-à-dire:
  If $\stackrel{\text{def}}{=}$ If' o (P_3^2,P_3^3,P_3^1)

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)
    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)
    $P_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1\leq i\leq k$, $(n_1,\ldots,n_k) \mapsto n_i$)
  - et stable par **composition** et **récursion primitive**

Si $b : \mathbf{N}^k \to \mathbf{N}$, et $e : \mathbf{N}^{k+2} \to \mathbf{N}$,
Prim(b,e) est la fonction totale $f : \mathbf{N}^{k+1} \to \mathbf{N}$ définie par:
$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$
$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

Exemples: 8. reste euclidien

- On pose: $\mathrm{Mod}(m,n) \stackrel{\text{def}}{=} m \bmod n$ (0 si n=0)

  $\mathrm{Mod}(0,n) = 0$

  $\mathrm{Mod}(\mathrm{S}(m),n) = 0$ si $\mathrm{Mod}(m,n)=n-1$

  $= \mathrm{Mod}(m,n)+1$ sinon

- On définit Mod' par:

  $\mathrm{Prim}(n \mapsto 0,$

  $n,m,r \mapsto \mathrm{If}\ (\mathrm{Eq}(\mathrm{S}(r),n),$

  $0,$

  $\mathrm{S}(r))$

- et $\mathrm{Mod} \stackrel{\text{def}}{=} \mathrm{Mod}' \circ (P_2^2,P_2^1)$

Si $b : \mathbf{N}^k \to \mathbf{N}$, et $e : \mathbf{N}^{k+2} \to \mathbf{N}$,
Prim(b,e) est la fonction totale $f : \mathbf{N}^{k+1} \to \mathbf{N}$ définie par:

$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$

$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

Ici r vaudra $\mathrm{Mod}'(n,m) = \mathrm{Mod}(m,n)$

Exemples: 9. division euclidienne

- On pose: $\mathrm{Div}(m,n) \stackrel{\text{def}}{=} \lfloor m/n \rfloor$ (m si n=0)

  $\mathrm{Div}(0,n) = 0$

  $\mathrm{Div}(\mathrm{S}(m),n) = \mathrm{Div}(m,n)+1$ si si $\mathrm{Mod}(m,n)=n-1$

  $= \mathrm{Div}(m,n)$ sinon

- On définit Div' par:

  $\mathrm{Prim}(n \mapsto 0,$

  $n,m,r \mapsto \mathrm{If}\ (\mathrm{Eq}(\mathrm{S}(\mathrm{Mod}(m,n)),n),$

  $\mathrm{S}(r),$

  $r)$

- et $\mathrm{Div} \stackrel{\text{def}}{=} \mathrm{Div}' \circ (P_2^2,P_2^1)$

Si $b : \mathbf{N}^k \to \mathbf{N}$, et $e : \mathbf{N}^{k+2} \to \mathbf{N}$,

$\mathrm{Prim}(b,e)$ est la fonction totale $f : \mathbf{N}^{k+1} \to \mathbf{N}$ définie par:

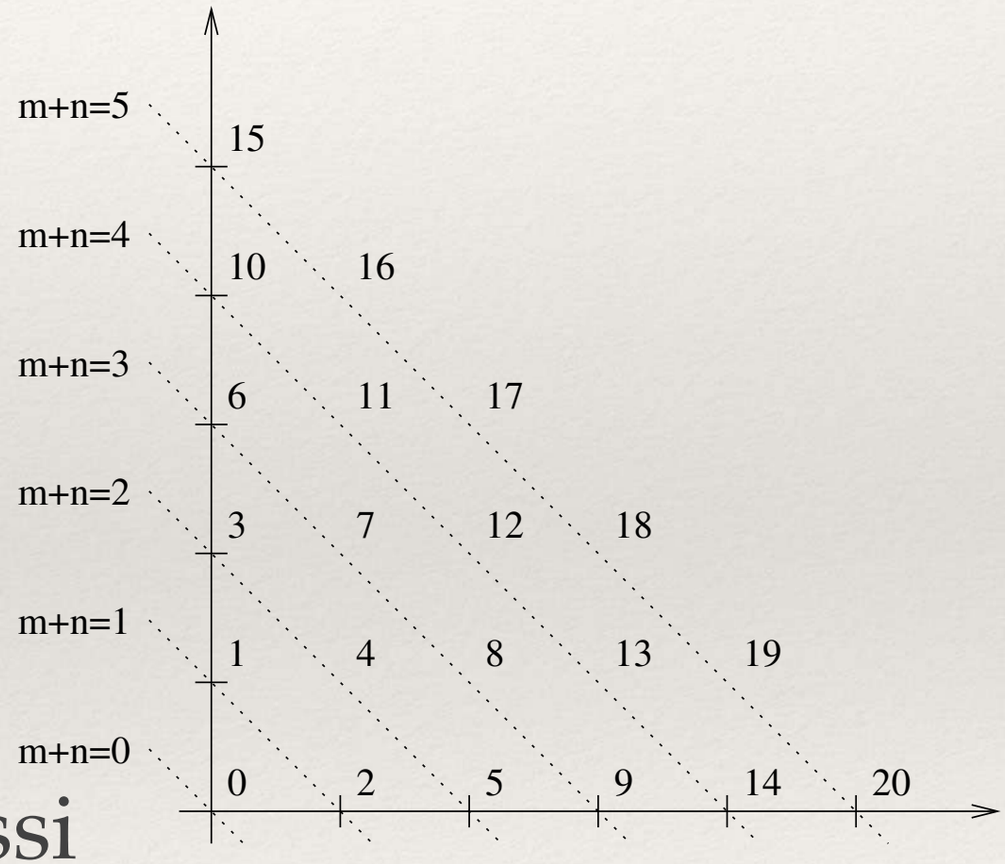
$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$

$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

Ici r vaudra $\mathrm{Div}'(n,m) = \mathrm{Div}(m,n)$

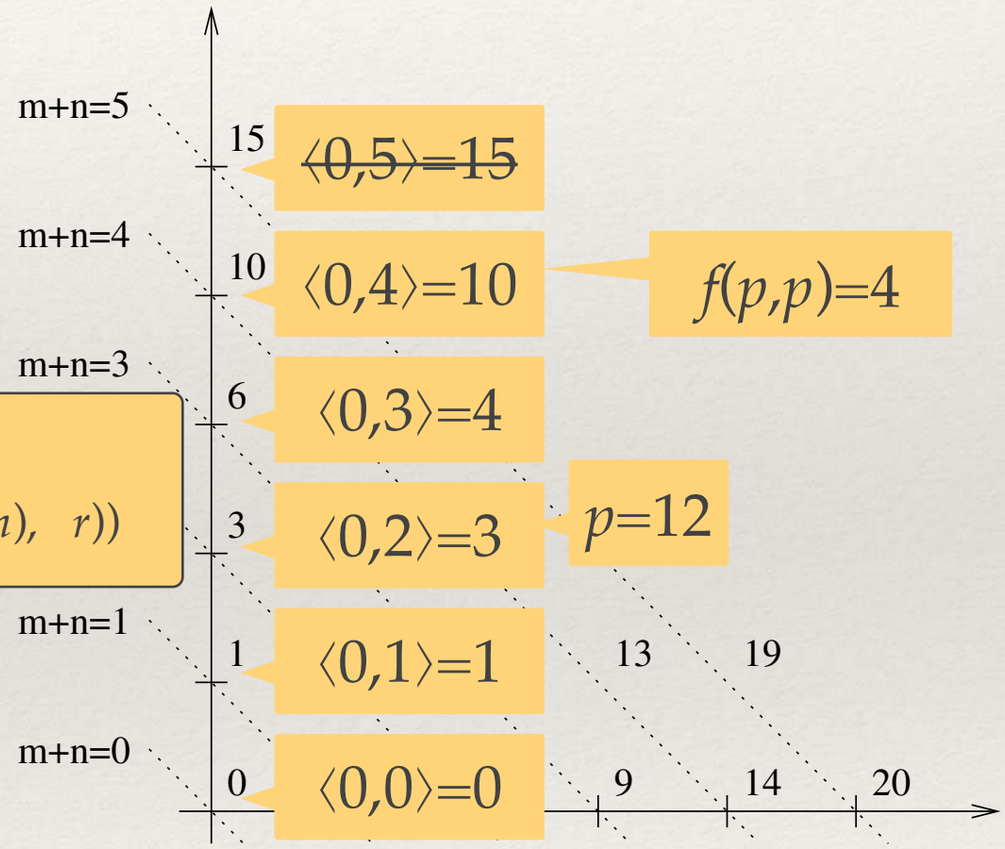
Exemples: 10. couples

- On pose: $\langle m,n\rangle \stackrel{\text{def}}{=} (m+n)(m+n+1)/2+m$
- bijection : $\mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N}$, primitive récursive
- Inverse: si $p=\langle m,n\rangle$, alors
  $q \stackrel{\text{def}}{=} \lfloor(-1+\text{sqrt}(1+8p))/2\rfloor$ vaut $m+n$
  donc $m = p-q(q+1)/2$, $n = q-m$
- L'inverse est **primitif récursif** aussi
  (en principe, calculer une racine carrée, mais on peut faire plus simple…)



Exemples: 10. couples: projections

- $\langle m,n\rangle \stackrel{\text{def}}{=} (m+n)(m+n+1)/2+m$
- Soit $f(p,N) \stackrel{\text{def}}{=} \max\{i \le N \mid \langle 0,i\rangle \le p\}$
  alors si $p=\langle m,n\rangle$, on a $m+n=f(p,p)$,
  $m=p-\langle 0,f(p,p)\rangle$, $n=f(p,p)-m$



$f \stackrel{\text{def}}{=} \text{Prim}\ (p \mapsto 0,\ p,n,r \mapsto \text{If}(\text{Leq}(\langle 0,\text{S}(n)\rangle,p),\ \text{S}(n),\ r))$

- $f(p,0) = 0$
  $f(p,N+1) =$ si $\langle 0,N+1\rangle \le p$ alors $N+1$
  sinon $f(p,N)$
- $\text{proj}_1 \stackrel{\text{def}}{=} \div \circ (\text{P}_1^1, \langle \_,\_\rangle \circ (\text{Z}, f \circ (\text{P}_1^1,\text{P}_1^1)))$
  $\text{proj}_2 \stackrel{\text{def}}{=} \div \circ (f \circ (\text{P}_1^1,\text{P}_1^1), \text{proj}_1)$

Exemples: 11. listes

- Listes à la Caml: $[n_0;n_1;\ldots;n_{k-1}] \stackrel{\text{def}}{=} n_0::(n_1::(\ldots::(n_{k-1}::\text{nil})))$
- On pose: $\text{nil} \stackrel{\text{def}}{=} 0 \qquad n :: \ell \stackrel{\text{def}}{=} \langle n,\ell\rangle+1$

le +1 est de sorte que $0\neq(n :: \ell)$

- Donc: $\text{consp}\ \ell \stackrel{\text{def}}{=} \text{Geq}(\ell,1)$ (* consp $(n :: \ell)=1$, consp nil=0 *)

  $\text{hd}\ \ell \stackrel{\text{def}}{=} \text{proj}_1(\text{Pred}(\ell))$ (* $\text{hd}(n :: \ell) = n$ *)

  $\text{tl}\ \ell \stackrel{\text{def}}{=} \text{proj}_2(\text{Pred}(\ell))$ (* $\text{tl}(n :: \ell) = \ell$ *)

- Aussi: $\text{nth\_tl}\ (\ell, 0) \stackrel{\text{def}}{=} \ell$ (* nth_tl $([n_0;n_1;\ldots;n_{k-1}], i)$*)

  $\text{nth\_tl}\ (\ell, i+1) \stackrel{\text{def}}{=} \text{tl}\ (\text{nth\_tl}\ (\ell, i))$ (* $= [n_i;n_{i+1};\ldots;n_{k-1}]$ si $i\leq k$ *)

- $\text{nth}\ (\ell, i) \stackrel{\text{def}}{=} \text{hd}\ (\text{nth\_tl}\ (\ell, i))$ (* nth $([n_0;n_1;\ldots;n_{k-1}], i) = n_i$ si $i<k$ *)

Exemples: 12. « course-of-values » recursion

- Le principe de récurrence primitive ne permet de définir $f(n)$ qu'en fonction de $f(n–1)$ (si $n \geq 1$)
- On aimerait définir $f(n)$ en fonction de n'importe quels $f(i)$ avec $i<n$
- Par exemple, comptage binaire

$$\text{inc}\ (\text{nil}) = [1]$$
$$\text{inc}\ (0 :: \ell) = 1 :: \ell$$
$$\text{inc}\ (1 :: \ell) = 0 :: \text{inc}\ \ell$$

($\ell < 1 :: \ell$: on ne s'appelle récursivement que sur des nombres plus petits)

Exemples: 12. « course-of-values » recursion

- « Course-of-values » recursion: principe en apparence plus général, permettant de définir

$$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$$

$$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,[f(\boldsymbol{m},n);\ldots;f(\boldsymbol{m},0)])$$

Récurrence primitive:

$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$

$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

- Une telle fonction est **déjà** définissable avec la récursion primitive:

$$g(\boldsymbol{m},0) \stackrel{\text{def}}{=} [b(\boldsymbol{m})] \qquad (* \; g(\boldsymbol{m},n) = [f(\boldsymbol{m},n);\ldots;f(\boldsymbol{m},0)] \; *)$$

$$g(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,g(\boldsymbol{m},n)) :: g(\boldsymbol{m},n)$$

- Puis: $f(\boldsymbol{m},n) \stackrel{\text{def}}{=} \text{hd}\,(g(\boldsymbol{m},n))$

Exemples: 13. autres

- Factorielle, test de primalité, etc.
- Toutes les fonctions p.r. sont **calculables** (on le démontrera plus tard)
- Mais existe-t-il des fonctions calculables **non primitive récursives**?

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $\quad Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)
    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)
    $P_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1 \le i \le k$, $(n_1,\ldots,n_k) \mapsto n_i$)
  - et stable par **composition** et **récursion primitive**

Si $b : \mathbf{N}^k \to \mathbf{N}$, et $e : \mathbf{N}^{k+2} \to \mathbf{N}$,
Prim(b,e) est la fonction totale $f : \mathbf{N}^{k+1} \to \mathbf{N}$ définie par:

$$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$$
$$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$$

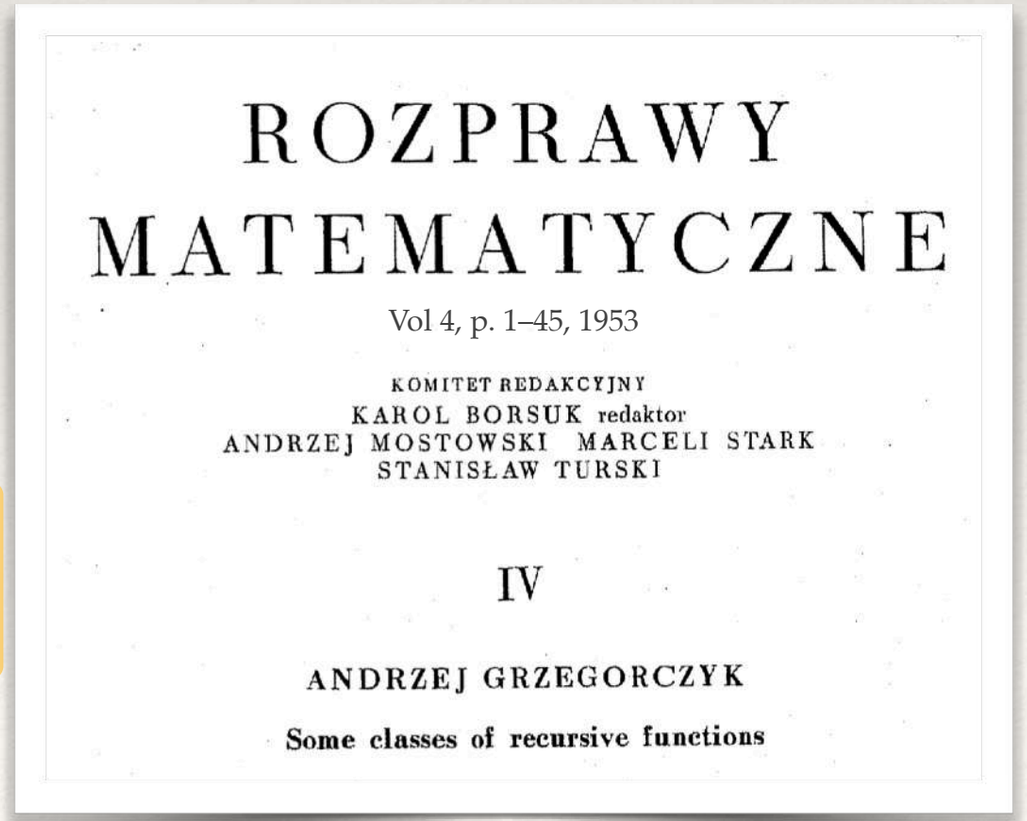
La hiérarchie de Grzegorczyk, la fonction d'Ackermann-Péter

Andrzej Grzegorczyk

Par AcademiconTV — Wywiady z Nestorami Filozofii Polskiej -- Wywiad 1: Prof. Andrzej Grzegorczyk, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=54817687

- $\psi_{n+1}(m) \stackrel{\text{def}}{=} \psi_n$ itérée m fois sur $\psi_n(1)$
- On montre que toute fonction primitive récursive est majorée par une fonction ψ_n
- $\psi(m,n) \stackrel{\text{def}}{=} \psi_n(m)$ est calculable, donc aussi $n \mapsto \psi(n,n)$ … qui ne sera pas p.r.

Fonction d'**Ackermann-Péter**



ROZPRAWY MATEMATYCZNE

Vol 4, p. 1–45, 1953

KOMITET REDAKCYJNY
KAROL BORSUK redaktor
ANDRZEJ MOSTOWSKI MARCELI STARK
STANISŁAW TURSKI

IV

ANDRZEJ GRZEGORCZYK

Some classes of recursive functions

Wilhelm Ackermann, Péter Rózsa

- $\psi_{n+1}(m) \stackrel{\text{def}}{=} \psi_n$ itérée m fois sur $\psi_n(1)$
- On montre que toute fonction primitive récursive est majorée par une fonction ψ_n
- $\psi(m,n) \stackrel{\text{def}}{=} \psi_n(m)$ est calculable, donc aussi $n \mapsto \psi(n,n)$ … qui ne sera pas p.r.

Wilhelm Friedrich Ackermann
Par Auteur inconnu — Steinfurt, Domaine public, https://commons.wikimedia.org/w/index.php?curid=9379449

Péter (née Politzer) Rózsa
Par JOC/EFR March 2006 — http://www-history.mcs.st-andrews.ac.uk/history/PictDisplay/Peter.html, Domaine public, https://commons.wikimedia.org/w/index.php?curid=1991781

Zum Hilbertschen Aufbau der reellen Zahlen. *Mathematische Annalen*, vol. 99, 1928, p. 118-133

Fonction d'**Ackermann-Péter**

$$A(0,m) \stackrel{\text{def}}{=} m+1$$
$$A(n+1,0) \stackrel{\text{def}}{=} A(n,1)$$
$$A(n+1,m+1) \stackrel{\text{def}}{=} A(n,A(n+1,m))$$

La hierarchie de Grzegorczyk

- $\psi_{n+1}(m) \stackrel{\text{def}}{=} \psi_n$ itérée m fois sur $\psi_n(1)$
- Formellement:

  $\psi_0(m) \stackrel{\text{def}}{=} m+1$

  $\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$

  $\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

Comparer avec la fonction d'Ackermann-Péter:

$A(0,m) \stackrel{\text{def}}{=} m+1$
$A(n+1,0) \stackrel{\text{def}}{=} A(n,1)$
$A(n+1,m+1) \stackrel{\text{def}}{=} A(n,A(n+1,m))$

- **Lemme.** Pour tout n, ψ_n est primitive récursive.

Preuve. $\psi_0 = S$

$$\psi_{n+1} = \text{Prim}(\mapsto \psi_n(1),\ m,r \mapsto \psi_n(r)). \quad \square$$

La hierarchie de Grzegorczyk

	0	1	2	3	4	m
ψ_0	1	2	3	4	5	m+1
ψ_1	2	3	4	5	6	m+2
ψ_2	3	5	7	9	11	2m+3
ψ_3	5	13	29	61	125	2^{m+3}-3
ψ_4	13	65 533	$2^{65\,536}$–3	…	…	2^(2^…^2) [m+2 termes]
ψ_5	65 533	ψ_4(65533)	…			
ψ_6	ψ_5(1)	…				

Propriétés de ψ_n

On a même montré $\psi_n(m) > m+1$ si $n \geq 1$

- **Lemme 1.** $\psi_n(m) \geq m+1$ (pour tous m, n)

$\psi_0(m) \stackrel{\text{def}}{=} m+1$
$\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$
$\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

- Par récurrence sur n, on montre: pour tout m, $\psi_n(m) \geq m+1$
  - Base: Si $n=0$, évident
  - Réc: on montre $\psi_{n+1}(m) > m+1$ par une 2ème réc. sur m,
    sachant par hyp. réc.: (*) pour tout m', $\psi_n(m') \geq m'+1$
    - Base: si $m=0$, $\psi_{n+1}(0) = \psi_n(1) \geq 1+1$ [par (*)] $> m+1$
    - Réc: $\psi_{n+1}(m+1) = \psi_n(\psi_{n+1}(m))$
      $\geq \psi_{n+1}(m)+1$ [par (*)]
      $> (m+1)+1$ [hyp. réc.] □

On raisonne sur ψ_n en premier,
pas sur ψ_{n+1},
car on ne sait pas (encore)
que ψ_n est croissante

Propriétés de ψ_n

- **Lemme 2.** ψ_n est strictement croissante.
- Si n=0, évident: $\psi_0(m) \stackrel{\text{def}}{=} m+1$
- Sinon, $\psi_{n+1}(m+1) = \psi_n(\psi_{n+1}(m))$
  $\geq \psi_{n+1}(m)+1$ [Lemme 1]. □

$\psi_0(m) \stackrel{\text{def}}{=} m+1$
$\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$
$\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

Lemme 1. $\psi_n(m) \geq m+1$
($>$ si $n \geq 1$)

Propriétés de ψ_n

- **Lemme 3.** Pour tout m, $n \mapsto \psi_n(m)$ est strictement croissante.

$\psi_0(m) \stackrel{\text{def}}{=} m+1$
$\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$
$\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

Lemme 1. $\psi_n(m) \geq m+1$ ($>$ si $n \geq 1$)

Lemme 2. ψ_n strict. croissante.

- Si $m=0$, $\psi_{n+1}(m) = \psi_n(1) > \psi_n(0)$ [Lemme 2]
  $= \psi_n(m)$
- Sinon, $\psi_{n+1}(m+1) = \psi_n(\psi_{n+1}(m))$
  $> \psi_n(m+1)$ [Lemme 1: $\psi_{n+1}(m) > m+1$
  + Lemme 2]. □

Propriétés de ψ_n

- **Lemme 4.** $\psi_1(m)=m+2$.
- On l'a déjà vu.
  Si vous voulez faire la récurrence sur m, exercice. □

$\psi_0(m) \stackrel{\text{def}}{=} m+1$
$\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$
$\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

Lemme 1. $\psi_n(m) \geq m+1$ ($>$ si $n \geq 1$)

Lemme 2. ψ_n strict. croissante.

Lemme 3. $n \mapsto \psi_n(m)$ str. croissante.

	0	1	2	3	4	m
ψ_0	1	2	3	4	5	$m+1$
ψ_1	2	3	4	5	6	$m+2$
ψ_2	3	5	7	9	11	$2m+3$
ψ_3	5	13	29	61	125	$2^{m+3}-3$
ψ_4	13	65 533	$2^{65\,536}-3$	…	…	2^(2^…^2) [$m+2$ termes] – 3
ψ_5	65 533	$\psi_4(65533)$	…			
ψ_6	$\psi_5(1)$	…				

Propriétés de ψ_n: composition

- **Lemme 5.** $\psi_p(\psi_n(m)) \le \psi_{\max(p,n)+2}(m)$.

- Soit $N \stackrel{\text{def}}{=} \max(p,n)$

  $\psi_p(\psi_n(m)) < \psi_N(\psi_{N+1}(m))$ [Lemme 3+Lemme 2]

  $= \psi_{N+1}(m+1)$ [déf. de ψ_{N+1}]

- (Si $m>0$) $\ldots = \psi_{N+1}(\psi_1(m-1))$ [Lemme 4]

  $\le \psi_{N+1}(\psi_{N+2}(m-1))$ [Lemme 3+Lemme 2]

  $= \psi_{N+2}(m)$ [déf. de ψ_{N+2}]

- (Si $m=0$) $\ldots = \psi_{N+1}(1) = \psi_{N+2}(0)$ [déf. de ψ_{N+2}]

  $= \psi_{N+2}(m)$. □

$\psi_0(m) \stackrel{\text{def}}{=} m+1$

$\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$

$\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

Lemme 1. $\psi_n(m) \ge m+1$ ($>$ si $n \ge 1$)

Lemme 2. ψ_n strict. croissante.

Lemme 3. $n \mapsto \psi_n(m)$ str. croissante.

Lemme 4. $\psi_1(m)=m+2$.

Propriétés de ψ_n: Prim

- Pour $\boldsymbol{n}=(n_1,\ldots,n_k)$, on note
  $\max \boldsymbol{n} \stackrel{\text{def}}{=} \max(n_1,\ldots,n_k)$ si $k\geq 1$, 0 sinon

- **Lemme 6.** Si $b(\boldsymbol{m})\leq\psi_k(\max \boldsymbol{m})$ et
  $e(\boldsymbol{m},n,p)\leq\psi_k(\max(\boldsymbol{m},n,p))$ $(\forall \boldsymbol{m},n,p)$,
  alors $\mathrm{Prim}(b,e)(\boldsymbol{m},n)\leq\psi_{k+1}(\max \boldsymbol{m}+n)$ $(\forall \boldsymbol{m},n)$.

- Récurrence sur n (en posant $f\stackrel{\text{def}}{=}\mathrm{Prim}(b,e)$).

- Si $n=0$, $f(\boldsymbol{m},n)=b(\boldsymbol{m})\leq\psi_k(\max \boldsymbol{m})$
  $\leq\psi_{k+1}(\max \boldsymbol{m}+n))$ [Lemme 2+3]

- Sinon, $f(\boldsymbol{m},n+1)=\ldots$ (voir transparent suivant)

$\psi_0(m) \stackrel{\text{def}}{=} m+1$
$\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$
$\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

Lemme 1. $\psi_n(m) \geq m+1$ ($>$ si $n\geq 1$)

Lemme 2. ψ_n strict. croissante.

Lemme 3. $n\mapsto\psi_n(m)$ str. croissante.

Lemme 4. $\psi_1(m)=m+2$.

Lemme 5. $\psi_p(\psi_n(m))\leq\psi_{\max(p,n)+2}(m)$

$f\stackrel{\text{def}}{=}\mathrm{Prim}(b,e) : \mathbf{N}^{k+1} \to \mathbf{N}$
$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$
$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

Propriétés de ψ_n: Prim

- **Lemme 6.** Si $b(\boldsymbol{m}) \le \psi_k(\max \boldsymbol{m})$ et
  $e(\boldsymbol{m},n,p) \le \psi_k(\max(\boldsymbol{m},n,p))$ $(\forall \boldsymbol{m},n,p)$,
  alors $\mathrm{Prim}(b,e)(\boldsymbol{m},n) \le \psi_{k+1}(\max \boldsymbol{m}+n))$ $(\forall \boldsymbol{m},n)$.
- … $f(\boldsymbol{m},n+1) = e(\boldsymbol{m},n,f(\boldsymbol{m},n))$
  $\le \psi_k(\max(\boldsymbol{m},n,f(\boldsymbol{m},n)))$
- $\le \psi_k(\max(\boldsymbol{m},n,\psi_{k+1}(\max \boldsymbol{m}+n)))$
  [hyp.réc.+Lemme 2]
- $= \psi_k(\psi_{k+1}(\max \boldsymbol{m}+n))$ [Lemme 1]
- $= \psi_{k+1}(\max \boldsymbol{m}+n+1)$ [déf. de ψ_{k+1}] □

$\psi_0(m) \stackrel{\text{def}}{=} m+1$
$\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$
$\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

Lemme 1. $\psi_n(m) \ge m+1$ (> si $n \ge 1$)

Lemme 2. ψ_n strict. croissante.

Lemme 3. $n \mapsto \psi_n(m)$ str. croissante.

Lemme 4. $\psi_1(m) = m+2$.

Lemme 5. $\psi_p(\psi_n(m)) \le \psi_{\max(p,n)+2}(m)$

$f \stackrel{\text{def}}{=} \mathrm{Prim}(b,e) : \mathbf{N}^{k+1} \to \mathbf{N}$
$f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$
$f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$

Majoration des fonctions p.r.

- **Prop.** $\forall f$ p.r., $\exists k$ / $\forall \boldsymbol{m}, f(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})$.
- *Preuve.* Soit $A \stackrel{\text{def}}{=} \{f \mid \exists k$ / $\forall \boldsymbol{m}, f(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})\}$
  On montre que A contient 0, Z, S, P_k^i,
  et est stable par composition et par récursion primitive
- La famille **PR** des fonctions p.r. est la plus petite ayant ces propriétés, donc **PR** $\subseteq A$.

$\psi_0(m) \stackrel{\text{def}}{=} m+1$
$\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$
$\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

Lemme 1. $\psi_n(m) \geq m+1$ ($>$ si $n \geq 1$)

Lemme 2. ψ_n strict. croissante.

Lemme 3. $n \mapsto \psi_n(m)$ str. croissante.

Lemme 4. $\psi_1(m) = m+2$.

Lemme 5. $\psi_p(\psi_n(m)) \leq \psi_{\max(p,n)+2}(m)$

Lemme 6. $b(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})$,
$e(\boldsymbol{m},n,p) \leq \psi_k(\max(\boldsymbol{m},n,p))$,
$\Rightarrow \text{Prim}(b,e)(\boldsymbol{m},n) \leq \psi_{k+1}(\max \boldsymbol{m}+n)$

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $\quad Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)
    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)
    $P_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1 \leq i \leq k$, $(n_1,\ldots,n_k) \mapsto n_i$)
  - et stable par **composition** et **récursion primitive**

Majoration des fonctions p.r.

- **Prop.** $\forall f$ p.r., $\exists k$ / $\forall \boldsymbol{m}, f(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})$.
- *Preuve.* Soit $A \stackrel{\text{def}}{=} \{f \mid \exists k$ / $\forall \boldsymbol{m}, f(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})\}$
- Cas 1: 0, $Z \in A$, car $0 \leq \psi_0(0)$
- Cas 2: $S \in A$, car $S(m) = m+1 = \psi_0(m)$
- Cas 3: $P_k^i \in A$, car $P_k^i(\boldsymbol{m}) \leq \max \boldsymbol{m} < \psi_0(\max \boldsymbol{m})$
- Restent composition et récursion primitive…

- **Définition.** La famille des **fonctions primitives récursives** est la plus petite contenant:
  - $0 : \mathbf{N}^0 \to \mathbf{N}$ $\quad Z : \mathbf{N}^1 \to \mathbf{N}$ (zero)
    $S : \mathbf{N}^1 \to \mathbf{N}$ (successeur, $n \mapsto n+1$)
    $P_k^i : \mathbf{N}^k \to \mathbf{N}$ (projections, $1 \leq i \leq k$, $(n_1,\ldots,n_k) \mapsto n_i$)
  - et stable par **composition** et **récursion primitive**

$\psi_0(m) \stackrel{\text{def}}{=} m+1$
$\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$
$\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

Lemme 1. $\psi_n(m) \geq m+1$ ($>$ si $n \geq 1$)

Lemme 2. ψ_n strict. croissante.

Lemme 3. $n \mapsto \psi_n(m)$ str. croissante.

Lemme 4. $\psi_1(m) = m+2$.

Lemme 5. $\psi_p(\psi_n(m)) \leq \psi_{\max(p,n)+2}(m)$

Lemme 6. $b(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})$,
$e(\boldsymbol{m},n,p) \leq \psi_k(\max(\boldsymbol{m},n,p))$,
$\Rightarrow \mathrm{Prim}(b,e)(\boldsymbol{m},n) \leq \psi_{k+1}(\max \boldsymbol{m}+n)$

Majoration des fonctions p.r.

- **Prop.** $\forall f$ p.r., $\exists k$ / $\forall \boldsymbol{m}, f(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})$.
- *Preuve.* Soit $A \stackrel{\text{def}}{=} \{f \mid \exists k$ / $\forall \boldsymbol{m}, f(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})\}$
- Composition: Lemme 5 (+ Lemme 2)

$\psi_0(m) \stackrel{\text{def}}{=} m+1$
$\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$
$\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

Lemme 1. $\psi_n(m) \geq m+1$ ($>$ si $n \geq 1$)

Lemme 2. ψ_n strict. croissante.

Lemme 3. $n \mapsto \psi_n(m)$ str. croissante.

Lemme 4. $\psi_1(m) = m+2$.

Lemme 5. $\psi_p(\psi_n(m)) \leq \psi_{\max(p,n)+2}(m)$

Lemme 6. $b(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})$,
$e(\boldsymbol{m},n,p) \leq \psi_k(\max(\boldsymbol{m},n,p))$,
$\Rightarrow \text{Prim}(b,e)(\boldsymbol{m},n) \leq \psi_{k+1}(\max \boldsymbol{m}+n)$

Majoration des fonctions p.r.

- **Prop.** $\forall f$ p.r., $\exists k$ / $\forall \boldsymbol{m}, f(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})$.
- *Preuve.* Soit $A \stackrel{\text{def}}{=} \{f \mid \exists k \,/\, \forall \boldsymbol{m}, f(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})\}$
- Récursion primitive:
  si $\forall \boldsymbol{m}, b(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})$
  $\forall \boldsymbol{m},n,p, e(\boldsymbol{m},n,p) \leq \psi_{k'}(\max(\boldsymbol{m},n,p))$,
  on peut supposer $k'=k$
  (sinon remplacer k,k' par $\max(k,k')$, grâce au Lemme 3)
- alors $\forall \boldsymbol{m},n$, $\mathrm{Prim}(b,e)(\boldsymbol{m},n)$
  $\leq \psi_{k+1}(\max \boldsymbol{m}+n))$ **[Lemme 6]**
  $\leq \psi_{k+1}(2(\max(\boldsymbol{m},n)))$ [+≤max, et Lemme 2]
  $\leq \psi_{k+1}(\psi_2(\max(\boldsymbol{m},n)))$ [$\psi_2(j)=2j+3$, et encore Lemme 2]
  $\leq \psi_{\max(k,1)+3}(\max(\boldsymbol{m},n))$ [Lemme 5] □

$\psi_0(m) \stackrel{\text{def}}{=} m+1$
$\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$
$\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

Lemme 1. $\psi_n(m) \geq m+1$ ($>$ si $n \geq 1$)

Lemme 2. ψ_n strict. croissante.

Lemme 3. $n \mapsto \psi_n(m)$ str. croissante.

Lemme 4. $\psi_1(m)=m+2$.

Lemme 5. $\psi_p(\psi_n(m)) \leq \psi_{\max(p,n)+2}(m)$

Lemme 6. $b(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})$,
$e(\boldsymbol{m},n,p) \leq \psi_k(\max(\boldsymbol{m},n,p))$,
$\Rightarrow \mathrm{Prim}(b,e)(\boldsymbol{m},n) \leq \psi_{k+1}(\max \boldsymbol{m}+n)$

La fonction d'Ackermann-Péter

- On pose $A(m,n) \stackrel{\text{def}}{=} \psi_n(m)$
- **Théorème.** A n'est pas primitive récursive.
- Sinon, $A \circ (P_1^1, P_1^1) : n \mapsto A(n,n)$ p.r.,
- donc $\exists k \ / \ \forall n,\ A(n,n) \leq \psi_k(n)$
- donc $\psi_{k+1}(k+1) = A(k+1,k+1) \leq \psi_k(k+1)$
- impossible par Lemme 3. □
- … et A est récursive (pas si facile, voir suite).

$\psi_0(m) \stackrel{\text{def}}{=} m+1$
$\psi_{n+1}(0) \stackrel{\text{def}}{=} \psi_n(1)$
$\psi_{n+1}(m+1) \stackrel{\text{def}}{=} \psi_n(\psi_{n+1}(m))$

$A(0,m) \stackrel{\text{def}}{=} m+1$
$A(n+1,0) \stackrel{\text{def}}{=} A(n,1)$
$A(n+1,m+1) \stackrel{\text{def}}{=} A(n,A(n+1,m))$

Lemme 3. $n \mapsto \psi_n(m)$ str. croissante.

Prop. $\forall f$ p.r., $\exists k$ /
$\forall \boldsymbol{m}, f(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})$.

La fonction d'Ackermann-Péter est récursive (totale)

A est récursive totale

- **Théorème.** A est récursive et totale.

- **Totalité:** on montre que A(n,m) est définie
  par réc. sur (n,m) ordonné lexicographiquement
  = récurrence **double** sur (n,m)
  = récurrence sur n d'abord, avec une sous-récurrence sur m

$$\begin{aligned} A(0,m) &\stackrel{\text{def}}{=} m+1 \\ A(n+1,0) &\stackrel{\text{def}}{=} A(n,1) \\ A(n+1,m+1) &\stackrel{\text{def}}{=} A(n,A(n+1,m)) \end{aligned}$$

- **Récursivité:** pas si facile, trois démonstrations possibles. La première:
  - On montre qu'on peut implémenter les **appels de fonction**
    (en particulier les appels récursifs—au sens informatique)
    sur une mT, en sauvegardant les valeurs des bandes sur une **pile**
    lors de l'appel et en les rétablissant au retour;
  - la pile est simplement une bande de travail supplémentaire
  - On utilise ensuite le théorème « calculable=récursif » qu'on verra dans la suite

A est récursive totale

- **Théorème.** A est récursive et totale.

- **Totalité:** on montre que A(n,m) est définie par réc. sur (n,m) ordonné lexicographiquement = récurrence **double** sur (n,m) = récurrence sur n d'abord, avec une sous-récurrence sur m

$$A(0,m) \stackrel{\text{def}}{=} m+1$$
$$A(n+1,0) \stackrel{\text{def}}{=} A(n,1)$$
$$A(n+1,m+1) \stackrel{\text{def}}{=} A(n,A(n+1,m))$$

- **Récursivité:** pas si facile, trois démonstrations possibles. La deuxième:

  - on utilise une construction de H. G. Rice (*Recursion and Iteration*, Comm. ACM, vol. 8, 1965, pp. 114-115), simplifiée par J. Arsac en 1986 (je vais juste en donner un aperçu)

JACQUES ARSAC

La fonction d'Ackermann : un nouveau mode de dérécursivation

Informatique théorique et applications, tome 20, n° 2 (1986), p. 149-156

<http://www.numdam.org/item?id=ITA_1986__20_2_149_0>

La construction d'Arsac

- $A(n+1,m+1) = A(n,A(n,\ldots A(n,A(n+1,0))\ldots))$
  $= A(n,A(n,\ldots A(n,A(n,1))\ldots))$

$A(0,m) \stackrel{\text{def}}{=} m+1$
$A(n+1,0) \stackrel{\text{def}}{=} A(n,1)$
$A(n+1,m+1) \stackrel{\text{def}}{=} A(n,A(n+1,m))$

- A la place, on définit $A'([n_k;\ldots;n_1], m) = A(n_1,A(n_2,\ldots A(n_{k-1},A(n_k,m))\ldots))$

C'est une **pile**

- (1) $A'(\text{nil},m) \stackrel{\text{def}}{=} m$ (fin du calcul; nil=0)
  (2) $A'(0::\ell, m) \stackrel{\text{def}}{=} A'(\ell, m+1)$

  (3) $A'((n+1)::\ell, 0) \stackrel{\text{def}}{=} A'(n::\ell, 1)$

  (4) $A'((n+1)::\ell, m+1) \stackrel{\text{def}}{=} A'((n+1)::n::\ell, m)$

  puis $A(n,m) \stackrel{\text{def}}{=} A'(n::\text{nil}, m)$

La construction d'Arsac

$A(0,m) \stackrel{def}{=} m+1$
$A(n+1,0) \stackrel{def}{=} A(n,1)$
$A(n+1,m+1) \stackrel{def}{=} A(n,A(n+1,m))$

- Rappel: (1) $A'(0,m) \stackrel{def}{=} m \qquad (0=[])$

  (2) $A'(\ell:0, m) \stackrel{def}{=} A'(\ell, m+1)$

  (3) $A'(\ell:(n+1), 0) \stackrel{def}{=} A'(\ell:n, 1)$

  (4) $A'(\ell:(n+1), m+1) \stackrel{def}{=} A'((\ell:n):n+1, m)$

  puis $A(n,m) \stackrel{def}{=} A'(0:n, m)$

- qu'on pourrait coder en Python (« dérécursivation d'une fonction récursive terminale ») comme:

```
def A (n,m):
    ell = 0:n
    while ell!=0:
        r,a = rest(ell),last(ell)
        if a==0:
            ell, m = r, m+1                  # cas (2)
        else:
            if m==0:
                ell, m = (r:(a-1)), 1        # cas (3)
            else:
                ell, m = ((r:a-1):a), m-1 # cas (4)
    return m                                 # cas (1)
```

A est récursive totale

- **Théorème.** A est récursive et totale.

- **Totalité:** on montre que A(n,m) est définie
  par réc. sur (n,m) ordonné lexicographiquement
  = récurrence **double** sur (n,m)
  = récurrence sur n d'abord, avec une sous-récurrence sur m

$$\begin{aligned} A(0,m) &\stackrel{\text{def}}{=} m+1 \\ A(n+1,0) &\stackrel{\text{def}}{=} A(n,1) \\ A(n+1,m+1) &\stackrel{\text{def}}{=} A(n,A(n+1,m)) \end{aligned}$$

- **Récursivité:** pas si facile, trois démonstrations possibles. La troisième:

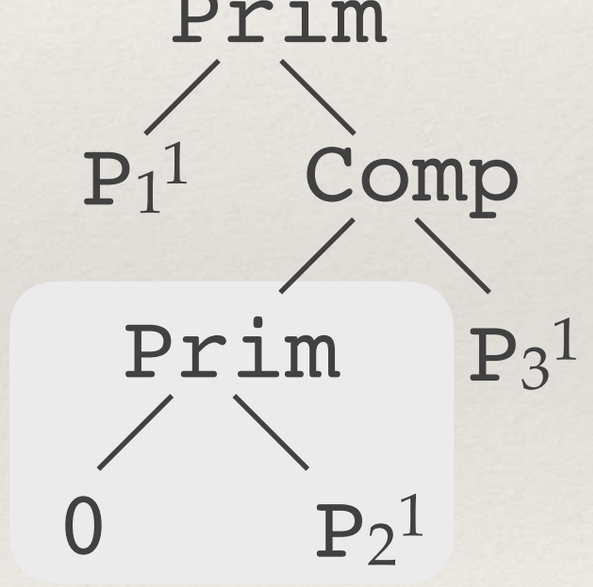
  - sera vue à la toute fin de ces transparents,
    comme conséquence d'autres théorèmes

Une parenthèse: termes p.r., ou: syntaxe et sémantique

Termes p.r.

- Une **syntaxe** pour décrire les fonctions p.r.:
  les **termes p.r.** Ce sont des **arbres finis**

- $T ::= \mathtt{0}$ (arité 0)
  - $\mid \mathtt{Z}$ (arité 1)
  - $\mid \mathtt{S}$ (arité 1)
  - $\mid \mathtt{P}_k^i$ (arité k, $1 \le i \le k$)
  - $\mid \mathtt{Comp}(T, T_1, \ldots, T_m)$
    (arité k, si T d'arité m et $T_1, \ldots, T_m$ d'arité k)
  - $\mid \mathtt{Prim}(T_b, T_e)$
    (arité k+1, si T_b d'arité k et T_e d'arité k+2)



(Ici, le terme p.r.
pour la soustraction;
celui du prédécesseur
sur fond blanc)

Sémantique des termes p.r.

- Sémantique $[\![T]\!]$ d'un terme p.r. T (d'arité k) = fonction de $\mathbf{N}^k$ vers $\mathbf{N}$ définie par:

$T ::= \mathtt{0}$
$\quad | \ \mathtt{Z}$
$\quad | \ \mathtt{S}$
$\quad | \ \mathtt{P}_k^i$
$\quad | \ \mathtt{Comp}(T, T_1, \ldots, T_m)$
$\quad | \ \mathtt{Prim}(T_b, T_e)$

- $[\![\mathtt{0}]\!] \stackrel{\text{def}}{=} 0 \quad [\![\mathtt{Z}]\!] \stackrel{\text{def}}{=} Z \quad [\![\mathtt{S}]\!] \stackrel{\text{def}}{=} S \quad [\![\mathtt{P}_k^i]\!] \stackrel{\text{def}}{=} \mathrm{P}_k^i$

  $[\![\mathtt{Comp}(T, T_1, \ldots, T_m)]\!] \stackrel{\text{def}}{=} [\![T]\!] \circ ([\![T_1]\!], \ldots, [\![T_m]\!])$

  $[\![\mathtt{Prim}(T_b, T_e)]\!] \stackrel{\text{def}}{=} \mathrm{Prim}([\![T_b]\!], [\![T_e]\!])$

- **Prop.** Les fonctions p.r. sont exactement les sémantiques de termes p.r.

Sémantique des termes p.r.

- **Prop.** Les fonctions p.r. sont exactement les sémantiques de termes p.r.

$$
\begin{aligned}
T ::= \ & \mathtt{0} \\
\mid \ & \mathtt{Z} \\
\mid \ & \mathtt{S} \\
\mid \ & \mathtt{P}_k^i \\
\mid \ & \mathtt{Comp}(T, T_1, \ldots, T_m) \\
\mid \ & \mathtt{Prim}(T_\mathrm{b}, T_\mathrm{e})
\end{aligned}
$$

- *Preuve.* Pour tout terme p.r. T, $[\![T]\!]$ est une fonction p.r.: récurrence sur la taille de T
- Réciproquement, l'ensemble $A \stackrel{\text{def}}{=} \{[\![T]\!] \mid T \text{ terme p.r.}\}$ contient 0, Z, S, les projections, et est stable par composition et récursion primitive
- … donc il contient **PR** (qui est le plus petit tel ensemble). □

Intérêt des termes p.r.

- **Prop.** Les fonctions p.r. sont exactement les sémantiques de termes p.r.

$T ::= \mathtt{0}$
$\quad | \ \mathtt{Z}$
$\quad | \ \mathtt{S}$
$\quad | \ \mathtt{P}_k^i$
$\quad | \ \mathtt{Comp}(T, T_1, \ldots, T_m)$
$\quad | \ \mathtt{Prim}(T_b, T_e)$

- Pourquoi introduire les termes p.r.?

- 1. permettent de représenter les fonctions p.r. sur une machine de Turing
  (par ex., le mot « $\mathtt{Prim}\ (\mathtt{P}_1^1, \mathtt{Comp}\ (\mathtt{Prim}\ (\mathtt{0}, \mathtt{P}_2^1), \mathtt{P}_3^1))$ »)

- 2. permettent de faire des **récurrences** sur la taille des termes p.r.
  (par ex., refaire la démonstration de **Prop.** $\forall f$ p.r., $\exists k \ / \ \forall \boldsymbol{m}, f(\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})$.
  en démontrant: $\forall T$ terme p.r., $\exists k \ / \ \forall \boldsymbol{m}, [\![T]\!](\boldsymbol{m}) \leq \psi_k(\max \boldsymbol{m})$
  par récurrence sur la taille de T)

Fonctions récursives (générales)

Fonctions récursives

- Définies comme les fonctions primitives récursives, mais avec une construction en plus: la **minimisation**
- Si $f : \mathbf{N}^{k+1} \to \mathbf{N}$, [en première approche]

  Min(f) est la fonction : $\mathbf{N}^k \to \mathbf{N}$ qui à m associe le premier (le plus petit) n / $f(m,n)=0$
- C'est une **boucle while**: en Python,

```
def min_f(m1,…,mk):
    n = 0
    while f(m1,…,mk,n)!=0:
        n += 1
    return n
```

Un tel n peut ne pas exister;
dans ce cas Min(f)(m) est **indéfini**
(= une boucle while peut
ne pas terminer)

Mais du coup on raisonne avec des
fonctions f **partielles**
désormais, et la définition de Min(f) change…

Minimisation

- Si $f : \mathbf{N}^{k+1} \to \mathbf{N}$ (partielle),
  Min(f) est la fonction (partielle) : $\mathbf{N}^k \to \mathbf{N}$ telle que pour tout $\boldsymbol{m} \in \mathbf{N}^k$,
  - Min(f)($\boldsymbol{m}$) $\stackrel{\text{def}}{=}$ (l'unique) $n \in \mathbf{N}$ /
    — $f(\boldsymbol{m},n)$ est défini et vaut 0
    — $f(\boldsymbol{m},0)$, …, $f(\boldsymbol{m},n-1)$ sont **définis** et $\neq 0$
    si un tel n existe
  - Min(f)($\boldsymbol{m}$) est **indéfini** sinon

Donc, par exemple, si
$f(\boldsymbol{m},0)=4$, $f(\boldsymbol{m},1)=3$, $f(\boldsymbol{m},2)$ indéfini et $f(\boldsymbol{m},3)=0$,
Min(f)($\boldsymbol{m}$) ne vaut pas 3: il est **indéfini**

R-termes

- Une **syntaxe**: les **R-termes**
- $T ::= \mathtt{0}$ (arité 0)
  - $\mid \mathtt{Z}$ (arité 1)
  - $\mid \mathtt{S}$ (arité 1)
  - $\mid \mathtt{P}_k^i$ (arité k, $1 \leq i \leq k$)
  - $\mid \mathtt{Comp}(T, T_1, \ldots, T_m)$
    (arité k, si T d'arité m et $T_1, \ldots, T_m$ d'arité k)
  - $\mid \mathtt{Prim}(T_b, T_e)$
    (arité k+1, si T_b d'arité k et T_e d'arité k+2)
  - $\mid \mathtt{Min}(T)$
    (arité k, si T d'arité k+1)

Nouveau

Sémantique des R-termes

- Sémantique $[\![T]\!]$ d'un R-terme T (d'arité k) = fonction de $\mathbf{N}^k$ vers $\mathbf{N}$ définie par:

$T ::= \mathtt{0}$
| $\mathtt{Z}$
| $\mathtt{S}$
| $\mathtt{P}_k^i$
| $\mathtt{Comp}(T, T_1, \ldots, T_m)$
| $\mathtt{Prim}(T_b, T_e)$
| $\mathtt{Min}(T)$

- $[\![\mathtt{0}]\!] \stackrel{\text{def}}{=} 0 \quad [\![\mathtt{Z}]\!] \stackrel{\text{def}}{=} Z \quad [\![\mathtt{S}]\!] \stackrel{\text{def}}{=} S \quad [\![\mathtt{P}_k^i]\!] \stackrel{\text{def}}{=} P_k^i$

  $[\![\mathtt{Comp}(T, T_1, \ldots, T_m)]\!] \stackrel{\text{def}}{=} [\![T]\!] \circ ([\![T_1]\!], \ldots, [\![T_m]\!])$

  $[\![\mathtt{Prim}(T_b, T_e)]\!] \stackrel{\text{def}}{=} \mathrm{Prim}([\![T_b]\!], [\![T_e]\!])$

- et $[\![\mathtt{Min}(T)]\!] \stackrel{\text{def}}{=} \mathrm{Min}([\![T]\!])$

… à condition d'**étendre les définitions** de Comp_m et Prim aux fonctions **partielles**

Comp_m, Prim et fonctions partielles

- Si $g_1, \ldots, g_m : \mathbf{N}^k \to \mathbf{N}$ (*partielles*),
  et $f : \mathbf{N}^m \to \mathbf{N}$ (*partielle*),
  $\mathrm{Comp}_m(f, g_1, \ldots, g_m) : \mathbf{N}^k \to \mathbf{N}$
  $\boldsymbol{n} \mapsto f(g_1(\boldsymbol{n}), \ldots, g_m(\boldsymbol{n}))$
  si [$g_1(\boldsymbol{n}), \ldots, g_m(\boldsymbol{n})$ définis et]
  $f(g_1(\boldsymbol{n}), \ldots, g_m(\boldsymbol{n}))$ défini
  (indéfini sinon)
- Si $b : \mathbf{N}^k \to \mathbf{N}$, et $e : \mathbf{N}^{k+2} \to \mathbf{N}$,
  Prim(b,e) est la fonction (*partielle*) $f : \mathbf{N}^{k+1} \to \mathbf{N}$ définie par:
  $f(\boldsymbol{m},0) \stackrel{\text{def}}{=} b(\boldsymbol{m})$ si défini
  $f(\boldsymbol{m},n+1) \stackrel{\text{def}}{=} e(\boldsymbol{m},n,f(\boldsymbol{m},n))$
  si $f(\boldsymbol{m},n)$ et $e(\boldsymbol{m},n,f(\boldsymbol{m},n))$ définis (indéfini sinon)

Les fonctions récursives (générales)

- **Définition.** Les fonctions récursives sont les sémantiques des R-termes p.r.

$$
\begin{aligned}
T ::= \ & \mathtt{0} \\
\mid \ & \mathtt{Z} \\
\mid \ & \mathtt{S} \\
\mid \ & \mathtt{P}_k^i \\
\mid \ & \mathtt{Comp}(T, T_1, \ldots, T_m) \\
\mid \ & \mathtt{Prim}(T_\mathrm{b}, T_\mathrm{e}) \\
\mid \ & \mathtt{Min}(T)
\end{aligned}
$$

- En particulier, toutes les fonctions p.r. sont récursives
- La fonction d'Ackermann-Péter est récursive (et totale), mais pas p.r.

« Calculable=récursif »

(Semi-)calculable=récursif (partiel)

- **Théorème.** Les fonctions **récursives partielles** sont exactement les fonctions **semi-calculables**.
Les fonctions **récursives totales** sont exactement les fonctions **calculables**.

$T ::= \mathtt{0}$
| $\mathtt{Z}$
| $\mathtt{S}$
| $\mathtt{P}_k^i$
| $\mathtt{Comp}(T, T_1, \ldots, T_m)$
| $\mathtt{Prim}(T_b, T_e)$
| $\mathtt{Min}(T)$

- C'est ce que nous allons montrer dans la suite.
Nous commençons par la direction récursif (partiel) ⇒ (semi-)calculable:

- **Proposition 1.** Pour tout R-terme T (d'arité k), en posant $f \stackrel{\text{def}}{=} [\![T]\!]$,
il existe une machine de Turing M_T telle que, pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si $f(\boldsymbol{n})$ défini, alors M_T termine sur l'entrée $\boldsymbol{n}$, et produit $f(\boldsymbol{n})$
— sinon, M_T ne termine pas sur l'entrée $\boldsymbol{n}$.

codé en binaire,
chaque entrée séparée par un #

- *Preuve*: on construit une **machine I/O** M_T par récurrence sur la taille de T…

Récursif partiel⇒semi-calculable

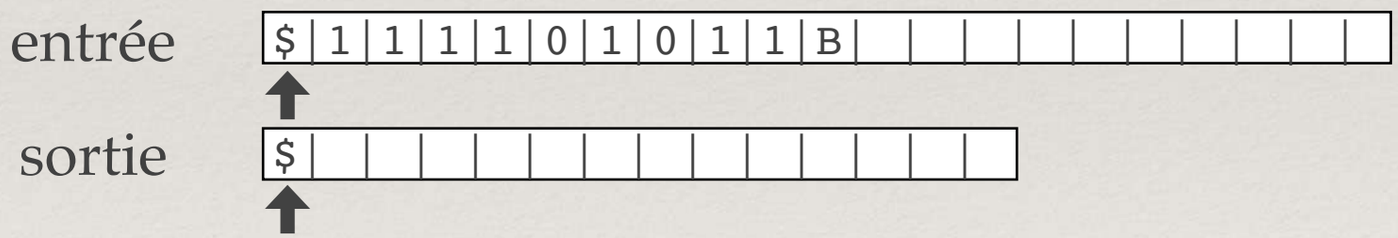
Récursif partiel⇒semi-calculable

Proposition 1. Pour tout R-terme T, soit $f \stackrel{\text{def}}{=} [\![T]\!]$, il existe une mT M_T / pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si $f(n)$ défini, $M_T(\boldsymbol{n})$ termine et produit $f(\boldsymbol{n})$
— sinon, $M_T(\boldsymbol{n})$ ne termine pas.

n codé en binaire,
chaque entrée séparée par un #

$T ::= \mathtt{0}$
| $\mathtt{Z}$
| $\mathtt{S}$
| $\mathtt{P}_k^i$
| $\mathtt{Comp}(T, T_1, \ldots, T_m)$
| $\mathtt{Prim}(T_b, T_e)$
| $\mathtt{Min}(T)$

- Cas `0`, `Z`, $\mathtt{P}_k^i$: exercice
- Cas `S`: pour incrémenter un nombre en binaire, lire l'entrée et:
- tant qu'on lit `1`, écrire `0` en sortie…



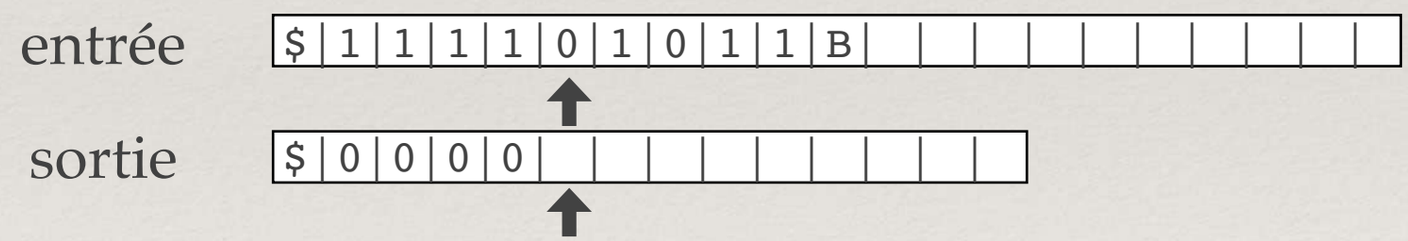
Récursif partiel⇒semi-calculable

Proposition 1. Pour tout R-terme T, soit $f \stackrel{\text{def}}{=} [\![T]\!]$, il existe une mT M_T / pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si $f(n)$ défini, $M_T(\boldsymbol{n})$ termine et produit $f(\boldsymbol{n})$
— sinon, $M_T(\boldsymbol{n})$ ne termine pas.

n codé en binaire,
chaque entrée séparée par un #

$T ::= \mathtt{0}$
| $\mathtt{Z}$
| $\mathtt{S}$
| $\mathtt{P}_k^i$
| $\mathtt{Comp}(T, T_1, \ldots, T_m)$
| $\mathtt{Prim}(T_b, T_e)$
| $\mathtt{Min}(T)$

- Cas `0`, `Z`, $\mathtt{P}_k^i$: exercice
- Cas `S`: pour incrémenter un nombre en binaire, lire l'entrée et:
- tant qu'on lit `1`, écrire `0` en sortie,



- au premier `0` en entrée (ou `B`), écrire un `1`…

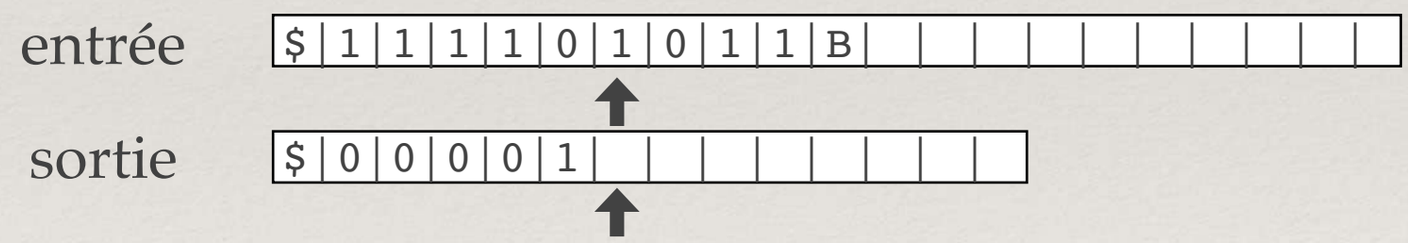
Récursif partiel⇒semi-calculable

Proposition 1. Pour tout R-terme T, soit $f \stackrel{\text{def}}{=} [\![T]\!]$,
il existe une mT M_T / pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si $f(n)$ défini, $M_T(\boldsymbol{n})$ termine et produit $f(\boldsymbol{n})$
— sinon, $M_T(\boldsymbol{n})$ ne termine pas.

n codé en binaire,
chaque entrée séparée par un #

$T ::= \mathtt{0}$
$\quad | \ \mathtt{Z}$
$\quad | \ \mathtt{S}$
$\quad | \ \mathtt{P}_k^i$
$\quad | \ \mathtt{Comp}(T, T_1, \ldots, T_m)$
$\quad | \ \mathtt{Prim}(T_b, T_e)$
$\quad | \ \mathtt{Min}(T)$

- Cas `0`, `Z`, $\mathtt{P}_k^i$: exercice
- Cas `S`: pour incrémenter un nombre en binaire, lire l'entrée et:
- tant qu'on lit `1`, écrire `0` en sortie,



- au premier `0` en entrée (ou `B`), écrire un `1`,
- puis recopier le reste de l'entrée et ajouter un `B`.

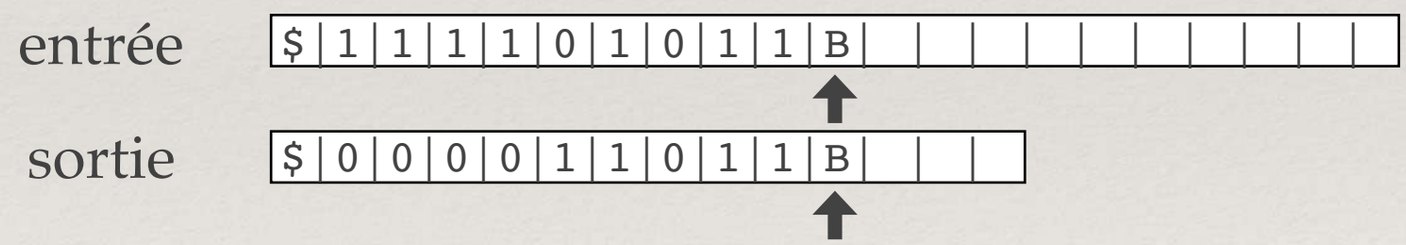
Récursif partiel⇒semi-calculable

Proposition 1. Pour tout R-terme T, soit $f \stackrel{\text{def}}{=} [\![T]\!]$,
il existe une mT M_T / pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si $f(n)$ défini, $M_T(\boldsymbol{n})$ termine et produit $f(\boldsymbol{n})$
— sinon, $M_T(\boldsymbol{n})$ ne termine pas.

n codé en binaire,
chaque entrée séparée par un #

$T ::= \mathtt{0}$
$\quad | \ \mathtt{Z}$
$\quad | \ \mathtt{S}$
$\quad | \ \mathtt{P}_k^i$
$\quad | \ \mathtt{Comp}(T, T_1, \ldots, T_m)$
$\quad | \ \mathtt{Prim}(T_b, T_e)$
$\quad | \ \mathtt{Min}(T)$

- Cas `0`, `Z`, $\mathtt{P}_k^i$: exercice
- Cas `S`: pour incrémenter un nombre en binaire, lire l'entrée et:
- tant qu'on lit `1`, écrire `0` en sortie,



- au premier `0` en entrée (ou `B`), écrire un `1`,
- puis recopier le reste de l'entrée et ajouter un `B`.

Récursif partiel⇒semi-calculable

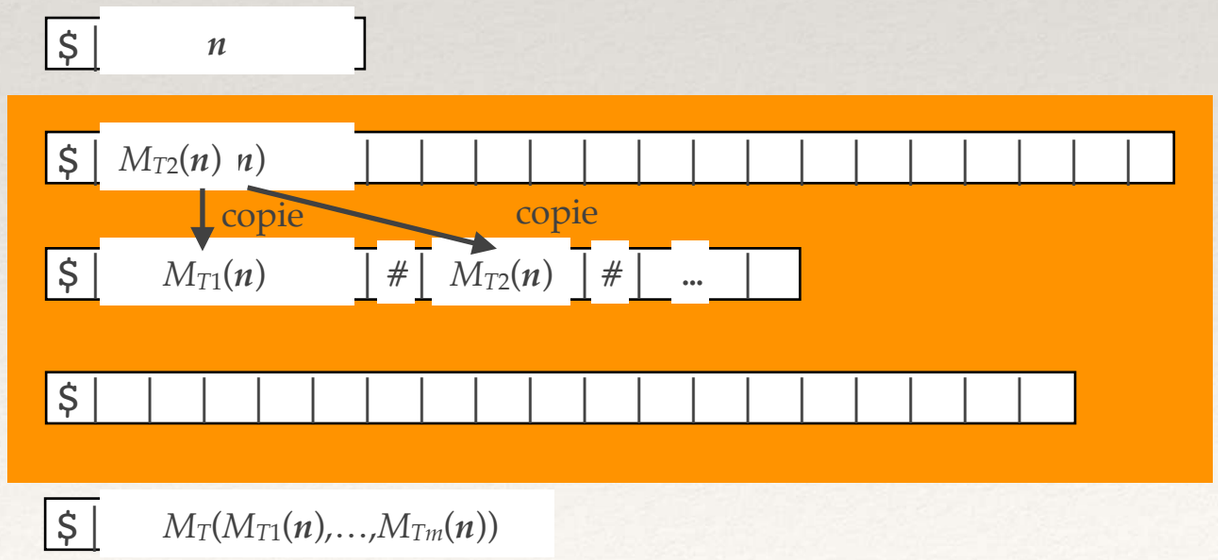
Proposition 1. Pour tout R-terme T, soit $f \stackrel{\text{def}}{=} [\![T]\!]$,
il existe une mT M_T / pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si $f(n)$ défini, $M_T(\boldsymbol{n})$ termine et produit $f(\boldsymbol{n})$
— sinon, $M_T(\boldsymbol{n})$ ne termine pas.

$\boldsymbol{n}$ codé en binaire,
chaque entrée séparée par un #

$T ::= \mathtt{0}$
| $\mathtt{Z}$
| $\mathtt{S}$
| $\mathtt{P}_k^i$
| $\mathtt{Comp}(T, T_1, \ldots, T_m)$
| $\mathtt{Prim}(T_b, T_e)$
| $\mathtt{Min}(T)$

- Cas $\mathtt{Comp}(T, T_1, \ldots, T_m)$:
  on simule $M_{T1}, \ldots, M_{Tm}$
  et on écrit $M_{T1}(\boldsymbol{n})\#\ldots\#M_{Tm}(\boldsymbol{n})$ sur une bande de travail

- puis on simule M_T
  prenant cette bande de travail
  en entrée



Récursif partiel⇒semi-calculable

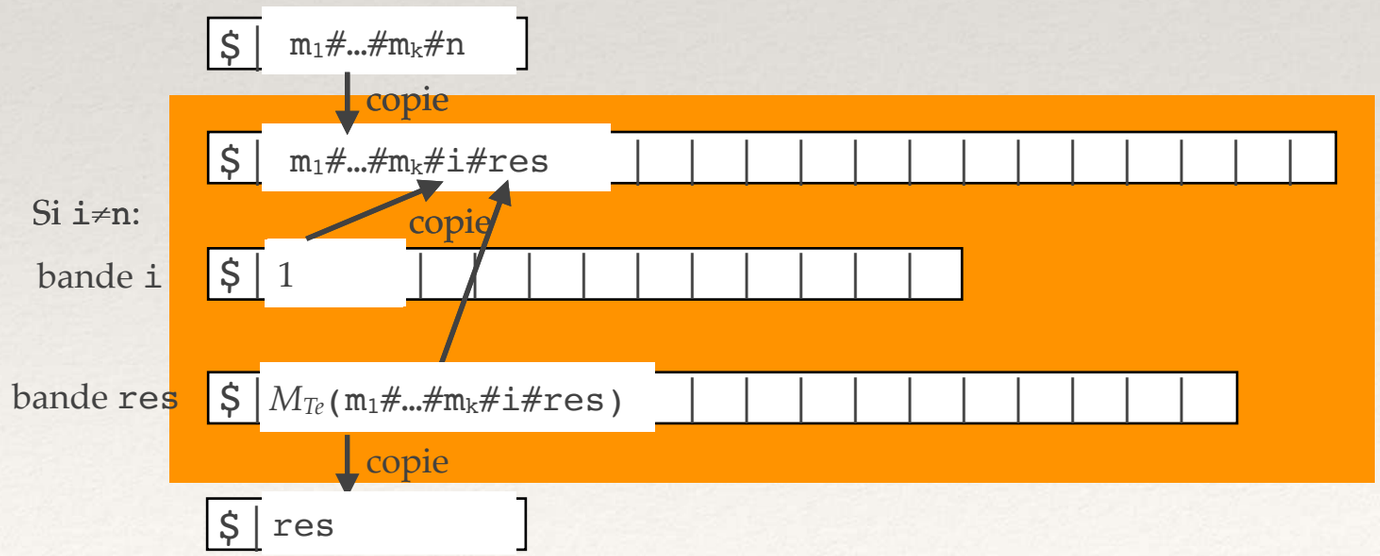
Proposition 1. Pour tout R-terme T, soit $f \stackrel{\text{def}}{=} [\![T]\!]$, il existe une mT M_T / pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si $f(n)$ défini, $M_T(\boldsymbol{n})$ termine et produit $f(\boldsymbol{n})$
— sinon, $M_T(\boldsymbol{n})$ ne termine pas.

n codé en binaire,
chaque entrée séparée par un #

$T ::= \mathtt{0}$
| $\mathtt{Z}$
| $\mathtt{S}$
| $\mathtt{P}_k^i$
| $\mathtt{Comp}(T, T_1, \ldots, T_m)$
| $\mathtt{Prim}(T_b, T_e)$
| $\mathtt{Min}(T)$

- Cas $\mathtt{Prim}(T_b, T_e)$:
  on réserve une bande pour le compteur `i`, et une pour `res`
- on a besoin de tester si `i`≠`n`:
  par parcours en parallèle de `i` et `n`
- on doit incrémenter `i`:
  similaire au cas du successeur `S`.
- etc. Lorsque `i`=`n`, finalement, recopier `res` sur la sortie

```
def f(m1,…,mk,n):
    res = b(m1,…,mk)
    for i in range(0,n-1):
        res = e(m1,…,mk,i,res)
    return res
```



Récursif partiel⇒semi-calculable

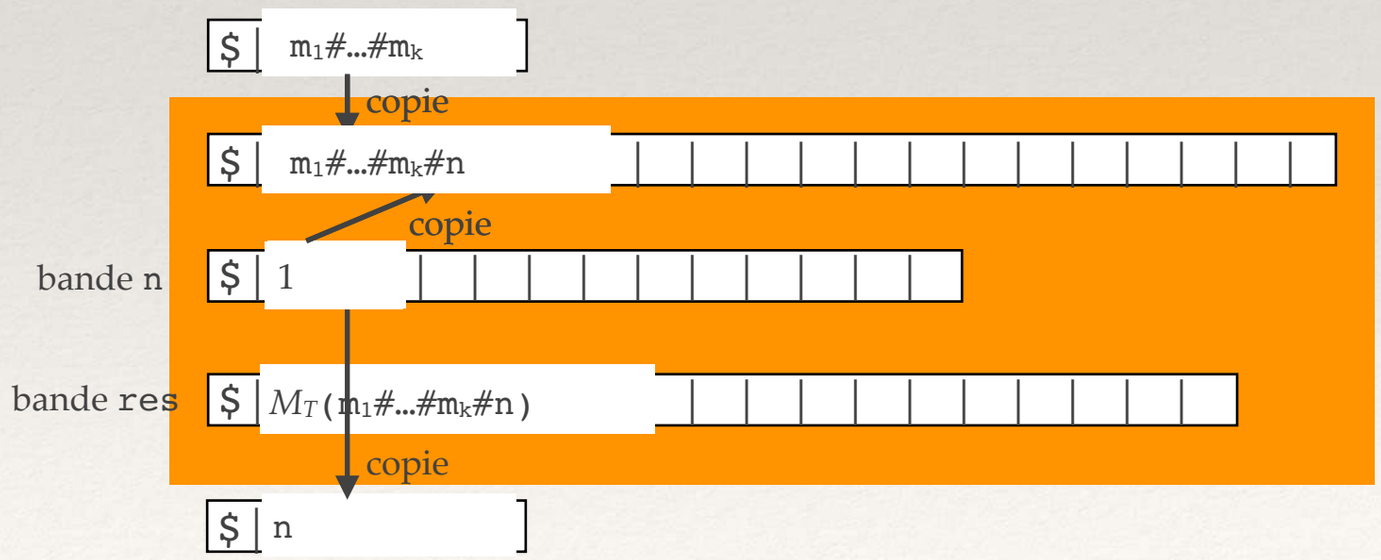
Proposition 1. Pour tout R-terme T, soit $f \stackrel{\text{def}}{=} [\![T]\!]$,
il existe une mT M_T / pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si $f(n)$ défini, $M_T(\boldsymbol{n})$ termine et produit $f(\boldsymbol{n})$
— sinon, $M_T(\boldsymbol{n})$ ne termine pas.

n codé en binaire,
chaque entrée séparée par un #

$T ::= \mathtt{0}$
| $\mathtt{Z}$
| $\mathtt{S}$
| $\mathtt{P}_k^i$
| $\mathtt{Comp}(T, T_1, \ldots, T_m)$
| $\mathtt{Prim}(T_b, T_e)$
| $\mathtt{Min}(T)$

```
def min_f(m1,…,mk):
    n = 0
    while f(m1,…,mk,n)!=0:
        n += 1
    return n
```

- Cas `Min`(T):
  on réserve une bande pour le compteur `n`
- on a besoin de tester si `res`=0:
  i.e., on cherche un bit à 1 dans `res`
- Si `res`≠0, on doit incrémenter `n`:
  similaire au cas du successeur `S`.
- etc. Lorsque `res`=0, finalement, recopier `n` sur la sortie. □



Récursif partiel⇒semi-calculable

Proposition 1. Pour tout R-terme T, soit $f \stackrel{\text{def}}{=} [\![T]\!]$,
il existe une mT M_T / pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si $f(n)$ défini, $M_T(\boldsymbol{n})$ termine et produit $f(\boldsymbol{n})$
— sinon, $M_T(\boldsymbol{n})$ ne termine pas.

n codé en binaire,
chaque entrée séparée par un #

$T ::= \mathtt{0}$
| $\mathtt{Z}$
| $\mathtt{S}$
| $\mathtt{P}_k^i$
| $\mathtt{Comp}(T, T_1, \ldots, T_m)$
| $\mathtt{Prim}(T_b, T_e)$
| $\mathtt{Min}(T)$

- **Note**: la fonction qui à T associe $\langle M_T \rangle^{\text{bin}}$
  est elle-même **calculable**

Semi-calculable⇒récursif partiel

Codage des mots

- Passons à (une forme de) réciproque.
- On se fixe une bijection num : $Q^+ \uplus \sum \to \{0, \ldots, B-1\}$ ($B \geq 3$)
  On code les mots $w \in (Q^+ \uplus \sum)^*$ comme des nombres, écrits en base B
- Plus précisément, si $w = a_1 \ldots a_m$, on pose

  $$C(w) \stackrel{\text{def}}{=} 1\ \text{num}(a_1) \ldots \text{num}(a_m),$$

  autrement dit:

  $$C(w) \stackrel{\text{def}}{=} B^m + \text{num}(a_1)\, B^{m-1} + \ldots + \text{num}(a_{m-1})\, B^1 + \text{num}(a_m)\, B^0$$

- On note w^{-1} le mot renversé, $a_m \ldots a_1$
- On code les configurations (w,q,w')…

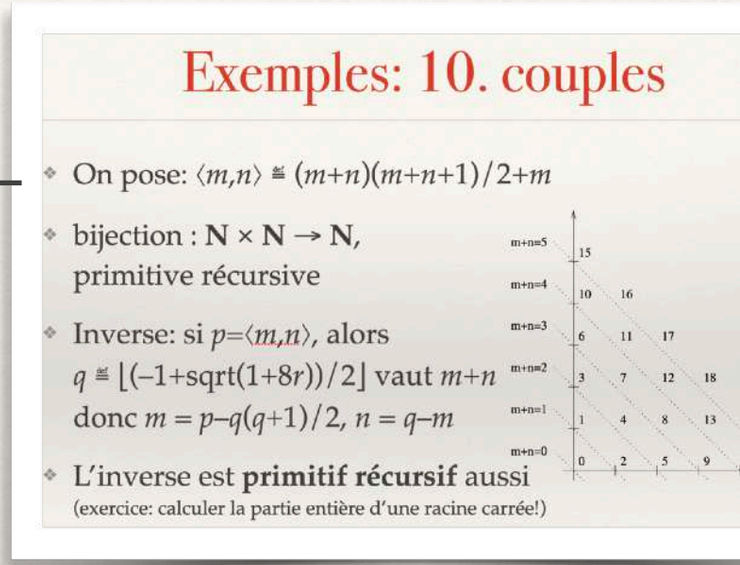
Le 1 en tête sert à rendre C **bijectif** : $(Q^+ \uplus \sum)^* \to \mathbf{N}-\{0\}$
(sans le 1, `23` et `0023` auraient le même code)

Codage des configurations

- On numérote les états étendus (de Q^+) q_0, …, q_{s-1} $(3 \le s < B)$
  avec **accept** $\stackrel{\text{def}}{=} q_{s-2}$, **reject** $\stackrel{\text{def}}{=} q_{s-1}$

- On code une configuration (w,q_i,w') par:

$$C(w,q_i,w') \stackrel{\text{def}}{=} i+s.\langle C(w),C(w'^{-1})\rangle$$



- Si $n \stackrel{\text{def}}{=} C(w,q_i,w')$, les fonctions suivantes sont **p.r.** en n:
  - l'état q_i: $i = n \bmod s$
  - $C(w) = \text{proj}_1(n \text{ div } s)$, $C(w'^{-1}) = \text{proj}_2(n \text{ div } s)$
  - la première lettre de w' = la dernière de w'^{-1}
    $= C(w'^{-1}) \bmod B$ si $C(w'^{-1}) \neq 1$ $[w' \neq \varepsilon]$
    … B sinon

$$C(w) \stackrel{\text{def}}{=} B^m + \text{num}(a_1)\, B^{m-1} + \ldots + \text{num}(a_{m-1})\, B^1 + \text{num}(a_m)\, B^0$$

  - $C(\text{reste}(w'^{-1})) = C(w'^{-1}) \text{ div } B$ si $C(w'^{-1}) \neq 1$ $[w' \neq \varepsilon$: $\text{reste}(w''a) \stackrel{\text{def}}{=} w'']$
    … 1 sinon $[\text{reste}(\varepsilon) \stackrel{\text{def}}{=} \varepsilon]$
  - $C(wb) = C(w).B+\text{num}(b)$, pour toute lettre b; pareil pour $C((bw')^{-1})$

La fonction de transition

- On code une configuration (w,q_i,w') par:
  $$C(w,q_i,w') \stackrel{\text{def}}{=} i+s.\langle C(w),C(w'^{-1})\rangle$$
- **Lemme 1.** Il existe une fonction p.r. g_M / si $\gamma \vdash_M \gamma'$ alors $g_M(C(\gamma))=C(\gamma')$.
- Si $\gamma=(w,q_i,w')$, on calcule q_i [en fait i], la première lettre a [num(a)], de façon **p.r.**, et on forme une table de tous les cas [en utilisant If]
- Pour chaque cas (q_i,a), on ne traite que des cas $i<s-2$
  ($\gamma \vdash_M \gamma'$, donc $q_i \neq$ **accept**, **reject**; sinon, on retourne ce qu'on veut)
- Soit $(q_j,b,d) \stackrel{\text{def}}{=} \delta(q_i,a)$, on forme:
  - $j+s.\langle C(wb),C(\text{reste}(w'^{-1}))\rangle$ si $d=\rightarrow$ [$=C(wb,q_j,w'')$ où $w'=aw''$]
  - $j+s.\langle C(w),C(\text{reste}(w'^{-1})\ b)\rangle$ si $d=\downarrow$ [$=C(w,q_j,bw'')$ où $w'=aw''$]
  - $j+s.\langle C(\text{reste}(w)),B.C(\text{reste}(w'^{-1})\ b) + (C(w) \bmod B)\rangle$ si $d=\leftarrow$. □

Calcul de la $m$ième configuration atteignable

- On code une configuration (w,q_i,w') par:

$$C(w,q_i,w') \stackrel{\text{def}}{=} i+s.\langle C(w),C(w'^{-1})\rangle$$

- **Lemme 1.** Il existe une fonction p.r. g_M / si $\gamma \vdash_M \gamma'$ alors $g_M(C(\gamma))=C(\gamma')$.

- **Lemme 2.** Il existe une fonction p.r. g_M^* /
  si $\gamma_0 \vdash_M \ldots \vdash_M \gamma_m$ alors $g_M^*(C(\gamma_0),m)=C(\gamma_m)$.

- On pose $g_M^*(n,0) \stackrel{\text{def}}{=} n$

$$g_M^*(n,m+1) \stackrel{\text{def}}{=} g_M(g_M^*(n,m))$$

- i.e., $g_M^* \stackrel{\text{def}}{=} \text{Prim}(n \mapsto n,$

$$n,m,r \mapsto g_M(r)) \quad \square$$

Test d'arrêt

- On code une configuration (w,q_i,w') par:

$$C(w,q_i,w') \stackrel{\text{def}}{=} i+s.\langle C(w),C(w'^{-1})\rangle$$

- **Lemme 3.** Il existe une fonction p.r. h_M / γ finale ssi $h_M(C(\gamma))=0$.

- Si $\gamma=(w,q_i,w')$, on calcule q_i [en fait i],
  de façon **p.r.**,
  - Si $i=s-1$ ou $s-2$ (**accept**, **reject**), retourner 0
  - Sinon, retourner 1. □

Temps d'exécution

- On code une configuration (w,q_i,w') par:

$$C(w,q_i,w') \stackrel{\text{def}}{=} i+s.\langle C(w),C(w'^{-1})\rangle$$

- **Lemme 3.** Il existe une fonction p.r. h_M / γ finale ssi $h_M(C(\gamma))=0$.

- **Lemme 4.** Il existe une fonction récursive t_M / $t_M(C(\gamma_0))$ = temps d'exécution de M partant de la configuration initiale γ_0, si M termine à partir de γ_0, et est indéfini sinon.

- $t_M \stackrel{\text{def}}{=} \text{Min}\,(n,m \mapsto h_M\,(g_M^*(n,m)))$, c'est-à-dire $\text{Min}\,(h_M \circ g_M^*)$. □

Simulation du calcul

- On code une configuration (w,q_i,w') par:

$$C(w,q_i,w') \stackrel{\text{def}}{=} i+s.\langle C(w),C(w'^{-1})\rangle$$

- **Lemme 3.** Il existe une fonction p.r. h_M / γ finale ssi $h_M(C(\gamma))=0$.

- **Lemme 4.** Il existe une fonction récursive t_M / $t_M(C(\gamma_0))$ = temps d'exécution de M partant de la configuration initiale γ_0, si M termine à partir de γ_0, et est indéfini sinon.

- **Lemme 5.** Il existe une fonction récursive simul_M / $\text{simul}_M(C(\gamma_0)) = C(\gamma_m)$ si M termine sur la config. γ_m à partir de γ_0, et est indéfini sinon.

- $\text{simul}_M(n) \stackrel{\text{def}}{=} g_M^*(n,t_M(n))$. □

Semi-calculable⇒récursif partiel

- Nous pouvons maintenant (presque) montrer:

- **Proposition 2.** Pour toute mT M, pour tout $k \in \mathbf{N}$, il existe une fonction récursive f telle que pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si M termine et calcule m sur l'entrée $\boldsymbol{n}$ (codée en binaire, séparateur #), alors $f(\boldsymbol{n})$ est défini et vaut m
— sinon, $f(\boldsymbol{n})$ est indéfini.

- $f(\boldsymbol{n}) \stackrel{\text{def}}{=} \text{decode}(\text{simul}_M(\text{code}_k(\boldsymbol{n}))$, où:

  - $\text{code}_k(n_1,\ldots,n_k) \stackrel{\text{def}}{=} C(\varepsilon, q_0, \$\text{bin}(n_1)\#\ldots\#\text{bin}(n_k))$
    $= s.\langle 1, C((\$\text{bin}(n_1)\#\ldots\#\text{bin}(n_k))^{-1})\rangle$

  - $\text{decode}(C(w,q,w')) \stackrel{\text{def}}{=}$ l'unique r / $ww' = \$\text{bin}(r)\texttt{0}\ldots\texttt{0}[\texttt{B}\ldots]$

Il nous reste donc un peu de travail

et pour ça aussi

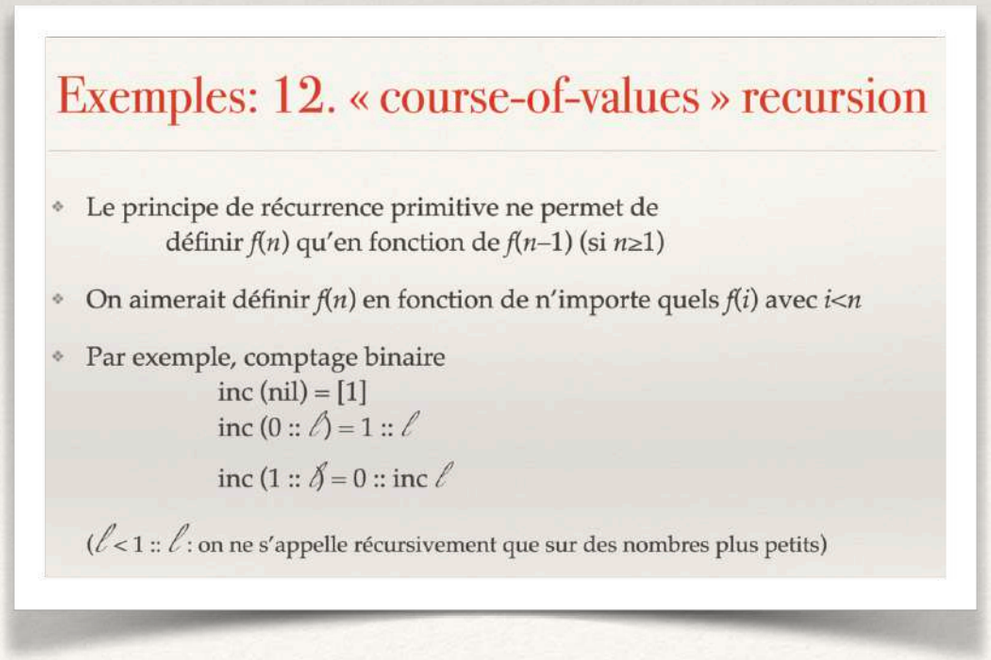
codages/décodages de codages binaires

Codage des codages binaires de nombres

- Si $n = b_{k-1}\,2^{k-1}+\ldots+b_1.2^1+b_0.2^0$, alors bin($n$) est le mot $b_0\ldots b_{k-1}$ sur $\{0,1\} \subseteq \Sigma$
- donc $C(\text{bin}(n)^{-1}) = B^k + \text{num}(b_{k-1})\,B^{k-1} + \ldots + \text{num}(b_1)\,B^1 + \text{num}(b_0)\,B^0$
- **Fait 1:** il existe une fonction p.r. inc : $C(\text{bin}(n)^{-1}) \mapsto C(\text{bin}(n+1)^{-1})$
  (peu importe ce qu'elle calcule sur les autres entrées)

- On a inc(m)=m+1 si m mod B=0 [$k\geq 1$ et b_0=0]
  inc(m div B).B si m mod $B\neq 0$ et $m\neq 1$ [$k\geq 1$ et b_0=1]
  B+1 si m=1 [k=0]

pas en format p.r.



Codage des codages binaires de nombres

- Si $n = b_{k-1}\,2^{k-1}+\ldots+b_1.2^1+b_0.2^0$, alors bin($n$) est le mot $b_0\ldots b_{k-1}$ sur $\{0,1\} \subseteq \Sigma$
- donc $C(\text{bin}(n)^{-1}) = B^k + \text{num}(b_{k-1})\,B^{k-1} + \ldots + \text{num}(b_1)\,B^1 + \text{num}(b_0)\,B^0$
- **Fait 2:** il existe une fonction p.r. Cbin : $n \mapsto C(\text{bin}(n)^{-1})$

- On a Cbin(0)=1
  Cbin(n+1)=inc(Cbin(n)) [primitif récursif]

Codage des codages binaires de nombres

- Si $n = b_{k-1}\,2^{k-1}+\ldots+b_1.2^1+b_0.2^0$, alors bin($n$) est le mot $b_0\ldots b_{k-1}$ sur $\{0,1\} \subseteq \Sigma$
- donc $C(\text{bin}(n)^{-1}) = B^k + \text{num}(b_{k-1})\,B^{k-1} + \ldots + \text{num}(b_1)\,B^1 + \text{num}(b_0)\,B^0$
- **Fait 3:** il existe une fonction p.r. inc' : $C(w\#\text{bin}(n)^{-1}) \mapsto C(w\#\text{bin}(n+1)^{-1})$

- On a inc'(m)=m+1 si m mod B=0 [$k\geq 1$ et b_0=0]
  inc'(m div B).B si m mod B=1 [$k\geq 1$ et b_0=1]
  $m.B$+1 sinon [k=0]

pas en format p.r.

Exemples: 12. « course-of-values » recursion

- Le principe de récurrence primitive ne permet de définir $f(n)$ qu'en fonction de $f(n-1)$ (si $n\geq 1$)
- On aimerait définir $f(n)$ en fonction de n'importe quels $f(i)$ avec $i<n$
- Par exemple, comptage binaire

  inc (nil) = [1]
  inc (0 :: ℓ) = 1 :: ℓ
  inc (1 :: ℓ) = 0 :: inc ℓ

  (ℓ < 1 :: ℓ : on ne s'appelle récursivement que sur des nombres plus petits)

Codage des codages binaires de nombres

- Si $n = b_{k-1}\,2^{k-1}+\ldots+b_1.2^1+b_0.2^0$, alors bin($n$) est le mot $b_0\ldots b_{k-1}$ sur $\{0,1\} \subseteq \Sigma$
- donc $C(\text{bin}(n)^{-1}) = B^k + \text{num}(b_{k-1})\,B^{k-1} + \ldots + \text{num}(b_1)\,B^1 + \text{num}(b_0)\,B^0$
- **Fait 4:** il existe une fonction p.r. Ccat : $C(w),n \mapsto C(w \,\#\, \text{bin}(n)^{-1})$

- On a $\text{Ccat}(m,0)=m.B^2+\text{num}(\#).B+1$
  $\text{Ccat}(m,n+1)=\text{inc}'(\text{Ccat}(m,n))$ [primitif récursif]
- **Fait 5:** pour tout entier k, il existe une fonction p.r.
  $\text{Cbin}_k : (n_1,\ldots,n_k) \mapsto C((\$\text{bin}(n_1)\#\ldots\#\text{bin}(n_k))^{-1})$
- $\text{Cbin}_k(n_1,\ldots,n_k) \stackrel{\text{def}}{=} \text{Ccat}(\text{Ccat}(\ \ldots\ \text{Ccat}(\text{Cbin}(n_k),n_{k-1}),\ \ldots\ n_2),n_1).B+\text{num}(\$)$

$$\begin{cases} \text{si } k\geq 2 \\ \text{Cbin}(n_1).B+\text{num}(\$) & \text{si } k=1 \\ \text{num}(\$) & \text{si } k=0 \end{cases}$$

Codage des codages binaires de nombres

- **Fait 5:** pour tout entier k, il existe une fonction p.r.
  $\mathrm{Cbin}_k : (n_1,\ldots,n_k) \mapsto \mathrm{C}((\$\mathrm{bin}(n_1)\#\ldots\#\mathrm{bin}(n_k))^{-1})$

- Donc:

- **Prop.** Il existe une fonction p.r. code_k telle que
  $\mathrm{code}_k(n_1,\ldots,n_k) \stackrel{\text{def}}{=} \mathrm{C}(\varepsilon, q_0, \$\mathrm{bin}(n_1)\#\ldots\#\mathrm{bin}(n_k))$
  $= s.\langle 1, \mathrm{C}((\$\mathrm{bin}(n_1)\#\ldots\#\mathrm{bin}(n_k))^{-1})\rangle$

Décodage des codages binaires de nombres

On ne se fatigue pas à éliminer
les zéros de poids fort:
$\mathrm{dec}(\mathrm{C}(w\mathtt{10})) = \mathrm{dec}(\mathrm{C}(w\mathtt{1})).B+1$
$= \mathrm{C}(w\mathtt{0}).B+1 = \mathrm{C}(w\mathtt{01})$

- Si $n = b_{k-1}\,2^{k-1}+\ldots+b_1.2^1+b_0.2^0$,
  $\mathrm{C}(\mathrm{bin}(n)^{-1}) = B^k + \mathrm{num}(b_{k-1})\,B^{k-1} + \ldots + \mathrm{num}(b_1)\,B^1 + \mathrm{num}(b_0)\,B^0$
- **Fait 6:** il y a une fonction p.r. dec: $\mathrm{C}(w\mathtt{0}\ldots\mathtt{0}\mathrm{bin}(n+1)^{-1}) \mapsto \mathrm{C}(w\mathtt{0}\ldots\mathtt{0}\mathrm{bin}(n)^{-1})$

(où w ne se termine ni par 0 ni par 1; peu importe ce que dec calcule sur les autres entrées)

- On a $\mathrm{dec}(m)=m-1$ si $m \bmod B=1$ [$k\geq 1$ et $b_0=1$]
  $\mathrm{dec}(m \operatorname{div} B).B+1$ si $m \bmod B=0$ [$k\geq 1$ et $b_0=0$]
  m sinon [cas w, notamment]

La décrémentation reboucle à $w\mathtt{1}\ldots\mathtt{1}$
sur les codages de 0:
$\mathrm{dec}(\mathrm{C}(w\mathtt{00})) = \mathrm{dec}(\mathrm{C}(w\mathtt{0})).B+1$
$= (\mathrm{dec}(\mathrm{C}(w)).B+1).B+1$
$= \mathrm{C}(w\mathtt{11})$
… mais peu importe

Décodage des codages binaires de nombres

- Si $n = b_{k-1}\,2^{k-1}+\ldots+b_1.2^1+b_0.2^0$,
  $C(\mathrm{bin}(n)^{-1}) = B^k + \mathrm{num}(b_{k-1})\,B^{k-1} + \ldots + \mathrm{num}(b_1)\,B^1 + \mathrm{num}(b_0)\,B^0$
- **Fait 7:** il existe une fonction p.r. test_0: $C(w0\ldots0\mathrm{bin}(n)^{-1}) \mapsto 1$ si n=0, 0 sinon
  (où w ne se termine ni par 0 ni par 1; peu importe ce que test_0 calcule sur les autres entrées)
- On a $\mathrm{test}_0(m)$=1 si m–1 si m mod $B\neq0,1$ [cas w]
  $\mathrm{test}_0(m$ div $B)$ si m mod B=0 [$k\geq1$ et b_0=0]
  0 sinon [cas w...1, notamment]

Décodage des codages binaires de nombres

- Si $n = b_{k-1}\,2^{k-1}+\ldots+b_1.2^1+b_0.2^0$,
  $\mathrm{C}(\mathrm{bin}(n)^{-1}) = B^k + \mathrm{num}(b_{k-1})\,B^{k-1} + \ldots + \mathrm{num}(b_1)\,B^1 + \mathrm{num}(b_0)\,B^0$

- **Fait 8:** il existe une fonction p.r. Dbin: $\mathrm{C}(w\mathtt{0}\ldots\mathtt{0}\mathrm{bin}(n)^{-1}) \mapsto n$
  (où w ne se termine ni par `0` ni par `1`; peu importe ce que Dbin calcule sur les autres entrées)

- $\mathrm{Dbin}(m) = 0$ si $\mathrm{test}_0(m)=1$
  $\mathrm{Dbin}(\mathrm{dec}(m))+1$ sinon

- Astuce (« **budget** »): on définit une fonction $\mathrm{Dbin}(m,budget)$ par:
  $\mathrm{Dbin2}(m,0) \stackrel{\text{def}}{=} 0$ [arbitraire]
  $\mathrm{Dbin2}(m,\boldsymbol{budget+1}) \stackrel{\text{def}}{=} 0$ si $\mathrm{test}_0(m)=1$
  $\mathrm{Dbin2}(\mathrm{dec}(m),\boldsymbol{budget})+1$ sinon

- Finalement, $\mathrm{Dbin}(m) \stackrel{\text{def}}{=} \mathrm{Dbin2}(m,\mathrm{S}(m))$
  [car $\mathrm{C}(\mathrm{bin}(n)^{-1}) \geq n$ pour tout n:
  la profondeur de récursion est donc majorée par m]

En général, il suffit de **majorer** la profondeur de récursion **par une fonction p.r.**

Décodage des codages binaires de nombres

- Si $n = b_{k-1}\,2^{k-1}+\ldots+b_1.2^1+b_0.2^0$,
  $\mathrm{C}(\mathrm{bin}(n)^{-1}) = B^k + \mathrm{num}(b_{k-1})\,B^{k-1} + \ldots + \mathrm{num}(b_1)\,B^1 + \mathrm{num}(b_0)\,B^0$

- **Fait 8:** il existe une fonction p.r. Dbin: $\mathrm{C}(w\texttt{0...0}\mathrm{bin}(n)^{-1}) \mapsto n$
  (où w ne se termine ni par `0` ni par `1`; peu importe ce qu'elle calcule sur les autres entrées)

- Donc:

  Prop. Il existe une fonction p.r. decode telle que
  $\mathrm{decode}(\mathrm{C}(w,q,w')) \stackrel{\text{def}}{=}$ l'unique r / ww'=`$`$\mathrm{bin}(r)$`0...0[B`...]
  … lorsque $w=\varepsilon$ [sinon, c'est plus compliqué]

- Comme $\mathrm{C}(w,q_i,w') \stackrel{\text{def}}{=} i+s.\langle \mathrm{C}(w),\mathrm{C}(w'^{-1})\rangle$,
  $\mathrm{decode}(m) = \mathrm{Dbin}(m' \text{ div } B)$
  où $m' \stackrel{\text{def}}{=} \mathrm{proj}_2(m \text{ div } s)$ $[= \mathrm{C}(w'^{-1}) = \mathrm{C}([\ldots\texttt{B}]\texttt{0...0}\mathrm{bin}(r)^{-1}\texttt{\$})]$

Semi-calculable=récursif partiel: fin de l'argument

Semi-calculable⇒récursif partiel

- **Proposition 2.** Pour toute mT M, pour tout $k \in \mathbf{N}$, il existe une fonction récursive f telle que pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si M termine et calcule m sur l'entrée $\boldsymbol{n}$ (codée en binaire, séparateur #), alors $f(\boldsymbol{n})$ est défini et vaut m
— sinon, $f(\boldsymbol{n})$ est indéfini.

- On produit d'abord une mT M' (à une bande) qui simule M, et ramène sa tête en **début de bande** en cas d'arrêt

- On pose $f(\boldsymbol{n}) \stackrel{\text{def}}{=} \text{decode}(\text{simul}_{M'}(\text{code}_k(\boldsymbol{n})))$:

Prop. Il existe une fonction p.r. code_k telle que
$\text{code}_k(n_1,\ldots,n_k) \stackrel{\text{def}}{=} C(\varepsilon,q_0,\$\text{bin}(n_1)\#\ldots\#\text{bin}(n_k))$
$= s.\langle 1, C((\$\text{bin}(n_1)\#\ldots\#\text{bin}(n_k))^{-1})\rangle$

Prop. Il existe une fonction p.r. decode telle que
$\text{decode}(C(w,q,w')) \stackrel{\text{def}}{=}$ l'unique r / $ww'=\$\text{bin}(r)0\ldots0[\text{B}\ldots]$
… lorsque $w=\varepsilon$ [sinon, c'est plus compliqué]

C'est pour ça qu'on demande à M' de ramener sa tête en début de bande à l'arrêt

(Semi-)calculable=récursif (partiel)

- Nous obtenons (enfin):

Théorème. Les fonctions **récursives partielles** sont exactement les fonctions **semi-calculables**.
Les fonctions **récursives totales** sont exactement les fonctions **calculables**.

- De plus, les constructions sont **effectives**:

on peut calculer le code de M_T à partir de T

Proposition 1. Pour tout R-terme T (d'arité k), en posant $f \stackrel{\text{def}}{=} [\![T]\!]$, il existe une machine de Turing M_T telle que, pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si $f(\boldsymbol{n})$ défini, alors M_T termine sur l'entrée $\boldsymbol{n}$, et produit $f(\boldsymbol{n})$
— sinon, M_T ne termine pas sur l'entrée $\boldsymbol{n}$.

on peut calculer un R-terme pour f à partir du code de M

Proposition 2. Pour toute mT M, pour tout $k \in \mathbf{N}$, il existe une fonction récursive f telle que pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si M termine et calcule m sur l'entrée $\boldsymbol{n}$ (codée en binaire, séparateur #), alors $f(\boldsymbol{n})$ est défini et vaut m
— sinon, $f(\boldsymbol{n})$ est indéfini.

Conséquences

Si on remplace « R-terme » par « terme p.r. », tous ces problèmes sont **décidables**: pourquoi?

Indécidabilité

Théorème. Les fonctions **récursives partielles** sont exactement les fonctions **semi-calculables**.
Les fonctions **récursives totales** sont exactement les fonctions **calculables**.
Les constructions sont **effectives** dans les deux sens.

- Les problèmes suivants sont **indécidables**:
  - **Entrée:** un R-terme T (arité k), $(n_1,\ldots,n_k) \in \mathbf{N}^k$
    Question: $[\![T]\!](n_1,\ldots,n_k)$ est-il défini?

    Réduction depuis L_U
  - **Entrée:** un R-terme T d'arité 0
    Question: $[\![T]\!]()$ est-il défini?

    Réduction depuis **Halt**
  - **Entrée:** un R-terme T (arité k)
    Question: $[\![T]\!](n_1,\ldots,n_k)$ est-il défini pour tout $(n_1,\ldots,n_k) \in \mathbf{N}^k$?

    Réduction depuis **UnivHalt**

Théorème de Rice

- Une propriété P de R-termes est **extensionnelle** ssi $[\![T]\!]=[\![T']\!]$ implique $P(T) \Leftrightarrow P(T')$

Théorème. Les fonctions **récursives partielles** sont exactement les fonctions **semi-calculables**.
Les fonctions **récursives totales** sont exactement les fonctions **calculables**.
Les constructions sont **effectives** dans les deux sens.

- P est **non-triviale** ssi il existe
  un R-terme T^+ / $P(T^+)$ et un R-terme T^- / $\neg P(T^-)$

- Si P est extensionnelle et non-triviale, alors:
  - **Entrée:** un R-terme T (arité k)
    Question: $P(T)$?
  - est indécidable.

(Ceci subsume tous les cas du transparent précédent.)

Théorème de Kleene

Proposition 2. Pour toute mT M, pour tout $k \in \mathbf{N}$, il existe une fonction récursive f telle que pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si M termine et calcule m sur l'entrée $\boldsymbol{n}$ (codée en binaire, séparateur #), alors $f(\boldsymbol{n})$ est défini et vaut m
— sinon, $f(\boldsymbol{n})$ est indéfini.

- Dans la preuve de la Prop 2, $f(\boldsymbol{n}) \stackrel{\text{def}}{=} \text{decode}(\text{simul}_{M'}(\text{code}_k(\boldsymbol{n})))$ où:
  - decode est p.r.
  - code_k est p.r.
  - $\text{simul}_{M'}(n) \stackrel{\text{def}}{=} g_{M'}^*(n, t_{M'}(n))$
  - $g_{M'}^*$ est p.r.
  - $t_{M'} \stackrel{\text{def}}{=} \text{Min}\,(n,m \mapsto h_{M'}\,(g_{M'}^*(n,m)))$
  - $h_{M'}$ est p.r.
- Donc f s'écrit $U_{M'}(\text{Min}(n,m \mapsto T_{M'}(n,m))$ où $U_{M'}, T_{M'}$ sont p.r.

on a souffert pour ça (« course-of-values » recursion, budgets)

On doit garder un Min ici.
L'astuce du budget ne fonctionne pas:
on ne sait pas majorer le temps d'exécution par une fonction p.r.

Une seule minimisation
= « on peut réécrire tout programme en un programme équivalent et qui n'a qu'**une seule boucle `while`** »

Théorème de Kleene

et **indépendantes** de f

- **Théorème (forme normale de Kleene).** Pour tout k,
il existe deux fonctions p.r. $U : \mathbf{N} \to \mathbf{N}$ et $T : \mathbf{N}^{k+2} \to \mathbf{N}$ telles que
pour toute fonction récursive partielle f d'arité k,
il existe un entier N_f tel que
Une seule minimisation
pour tout $\boldsymbol{n} \in \mathbf{N}^k$, $f(\boldsymbol{n}) = U(\mathrm{Min}(\boldsymbol{n},m \mapsto T(N_f,\boldsymbol{n},m)))$
(les deux côtés étant soit tous les deux définis, soit tous les deux indéfinis).

- De plus, $f \mapsto N_f$ est calculable (f étant représentée par un R-terme).

- *Preuve* (sketch): f s'écrit $U_{M'}(\mathrm{Min}(n,m \mapsto T_{M'}(n,m))$ où $U_{M'},T_{M'}$ sont p.r.,
et où M' est une mT qui calcule f…

- mais on veut que U et T soient **indépendantes** de f: comment faire?

- Machine de Turing **universelle** ; on pose $N_f \stackrel{\text{def}}{=} \langle M' \rangle^{\text{bin}}$. □

Fonctions calculables en temps p.r.

- **Théorème.** Les fonctions p.r. sont exactement celles qui sont calculables **en temps primitif récursif**.

- *Preuve* (sketch). Dans un sens, si $f=[\![T]\!]$ pour un terme p.r. T, on vérifie que la machine M_T termine en temps p.r. (récurrence sur T).

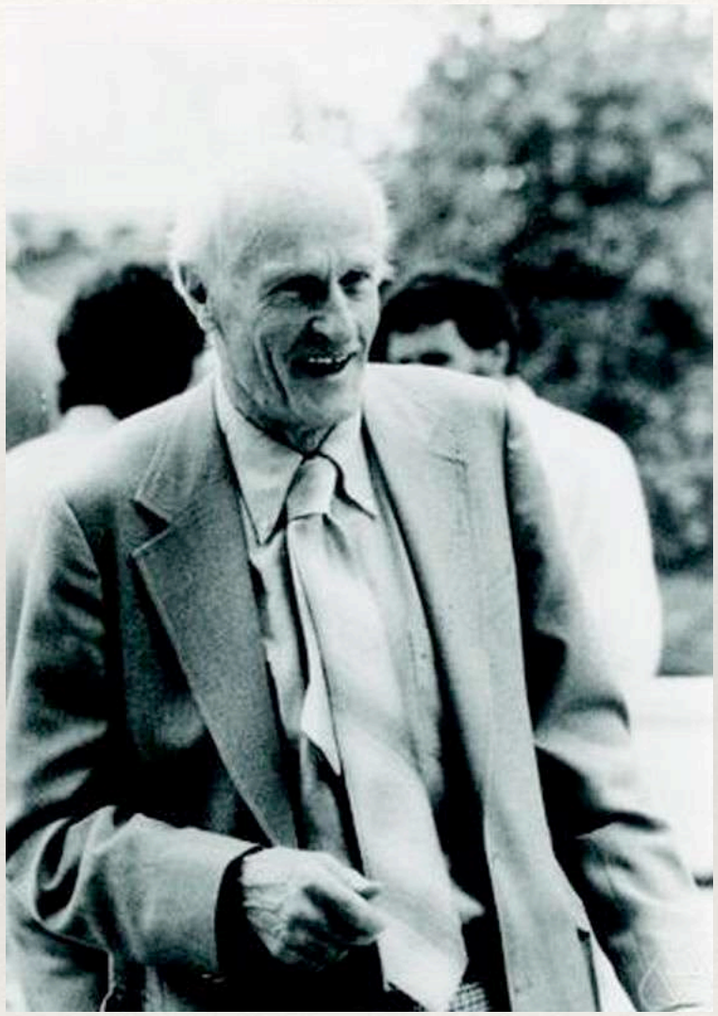
Proposition 1. Pour tout R-terme T (d'arité k), en posant $f \stackrel{\text{def}}{=} [\![T]\!]$, il existe une machine de Turing M_T telle que, pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si $f(\boldsymbol{n})$ défini, alors M_T termine sur l'entrée $\boldsymbol{n}$, et produit $f(\boldsymbol{n})$
— sinon, M_T ne termine pas sur l'entrée $\boldsymbol{n}$.

- Dans l'autre sens, on remplace l'unique minimisation $t_{M'} \stackrel{\text{def}}{=} \text{Min}\,(n,m \mapsto h_{M'}\,(g_{M'}{}^*(n,m)))$ par la fonction p.r. majorant le temps d'exécution de M'. □

Proposition 2. Pour toute mT M, pour tout $k \in \mathbf{N}$, il existe une fonction récursive f telle que pour tout $\boldsymbol{n} \in \mathbf{N}^k$,
— si M termine et calcule m sur l'entrée $\boldsymbol{n}$ (codée en binaire, séparateur #), alors $f(\boldsymbol{n})$ est défini et vaut m
— sinon, $f(\boldsymbol{n})$ est indéfini.

Stephen Cole Kleene

- A établi la plus grande partie des théorèmes fondamentaux de la théorie de la **récursivité**
- … il a aussi créé la théorie des langages rationnels et des **automates**
- … et a inventé la notion de **réalisabilité** en logique (intuitionniste)



Par Konrad Jacobs, Erlangen, Copyright is MFO — Mathematisches Forschungsinstitut Oberwolfach, https://opc.mfo.de/detail?photo_id=2122, CC BY-SA 2.0 de, https://commons.wikimedia.org/w/index.php?curid=12342617

Fonction d'évaluation universelle

Théorème (forme normale de Kleene). Pour tout k,
pour toute fonction récursive partielle f d'arité k,
pour tout $\boldsymbol{n} \in \mathbf{N}^k$, $f(\boldsymbol{n}) = U(\text{Min}(\boldsymbol{n},m \mapsto T(N_f,\boldsymbol{n},m)))$

- Dans la forme normale de Kleene N_f est l'**index** de f.
  Il existe une fonction d'**évaluation** (ou **application**) **universelle**:

- **Théorème.** $\forall k$, il existe une fonction réc. partielle $\mathbf{app}_k$ /
  $\forall f$ réc. partielle d'arité k, $\forall \boldsymbol{n} \in \mathbf{N}^k$, $\mathbf{app}_k(N_f,\boldsymbol{n}) = f(\boldsymbol{n})$

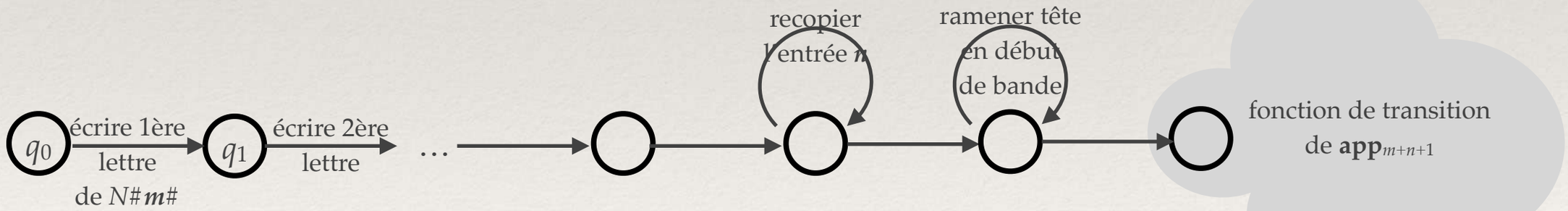
- *Preuve*: $\mathbf{app}_k(N,\boldsymbol{n}) \stackrel{\text{def}}{=} U(\text{Min}(\boldsymbol{n},m \mapsto T(N,\boldsymbol{n},m)))$ □

N_f, c'est ‘f (quote)
$\mathbf{app}_k$, c'est `apply`

Le théorème Smn (… dû à Kleene)

application partielle de N à ses m premiers arguments

- **Théorème.** $\forall\, m,n \in \mathbf{N}$, il existe une fonction p.r. $\mathbf{S}_{mn}$ / $\forall N \in \mathbf{N}, \boldsymbol{m} \in \mathbf{N}^m, \boldsymbol{n} \in \mathbf{N}^n,\quad \mathbf{app}_{m+n+1}(N,\boldsymbol{m},\boldsymbol{n}) = \mathbf{app}_{n+1}(\mathbf{S}_{mn}(N,\boldsymbol{m}),\boldsymbol{n})$
- *Preuve* (sketch). On produit le code $\langle M_{N,\boldsymbol{m}}\rangle^{\text{bin}}$ d'une mT $M_{N,\boldsymbol{m}}$ calculant $\mathbf{app}_{m+n+1}(N,\boldsymbol{m},\boldsymbol{n})$ sur l'entrée $\boldsymbol{n}$. $M_{N,\boldsymbol{m}}$:
- — écrit $N\#\boldsymbol{m}\#$ (stockée dans son contrôle) sur une bande de travail,
  — y recopie son entrée $\boldsymbol{n}$ à la suite
  — et calcule la fonction récursive $\mathbf{app}_{m+n+1}$ sur $N\#\boldsymbol{m}\#\boldsymbol{n}$



- La fonction $N,\boldsymbol{m} \mapsto \langle M_{N,\boldsymbol{m}}\rangle^{\text{bin}}$ est calculable en temps linéaire (donc p.r.)
  Elle est donc non seulement récursive, mais **primitive récursive**. □

Le théorème de récursion (… de Kleene)

- **Théorème.** ∀k, ∀ fonction récursive **totale** f d'arité 1,
  il existe $N \in \mathbf{N}$ / ∀$\boldsymbol{m} \in \mathbf{N}^k$, $\mathbf{app}_{k+1}(f(N),\boldsymbol{m}) = \mathbf{app}_{k+1}(N,\boldsymbol{m})$

- En identifiant les entiers N avec des (index de) fonctions récursives F,
  « `let rec` $F\,(\boldsymbol{m}) = f(F)(\boldsymbol{m})$ »
- La preuve vient de l'imitation des combinateurs de point fixe du lambda-calcul (… au second semestre)

C'est un théorème de **point fixe**:
$f(N) \cong N$,
où $N \cong N'$ ssi N et N' sont des index de la même fonction
($\mathbf{app}_{k+1}(N,\_) = \mathbf{app}_{k+1}(N',\_)$)

- *Preuve*. Une double diagonalisation délirante… transparent suivant.

Le théorème de récursion (… de Kleene)

- **Théorème.** $\forall k$, $\forall$ fonction récursive **totale** f d'arité 1,
  il existe $N \in \mathbf{N}$ / $\forall \boldsymbol{m} \in \mathbf{N}^k$, $\mathbf{app}_{k+1}(f(N),\boldsymbol{m}) = \mathbf{app}_{k+1}(N,\boldsymbol{m})$
- *Preuve.* Soit $g(n,\boldsymbol{m}) \stackrel{\text{def}}{=} \mathbf{app}_{k+1}(f(\mathbf{app}_2(n,n)),\boldsymbol{m})$.
- g est récursive, et a donc un index N_g: $\forall n,\boldsymbol{m}$, $\mathbf{app}_{k+2}(N_g,n,\boldsymbol{m}) = g(n,\boldsymbol{m})$
- Théorème Smn: $\forall n,\boldsymbol{m}$, $g(n,\boldsymbol{m}) = \mathbf{app}_{k+1}(\mathbf{S}_{1(k+1)}(N_g,n),\boldsymbol{m})$ (1)
- Soit $N_\mathbf{S}$ un index de $n \mapsto \mathbf{S}_{1(k+1)}(N_g,n)$: $\forall n$, $\mathbf{S}_{1(k+1)}(N_g,n)=\mathbf{app}_2(N_\mathbf{S},n)$ (2)

  $\mathbf{S}_{1(k+1)}$ est p.r., donc **totale**
- $\forall n,\boldsymbol{m}$, $\mathbf{app}_{k+1}(f(\mathbf{app}_2(n,n)),\boldsymbol{m}) = g(n,\boldsymbol{m})$ [déf. de g]
  $= \mathbf{app}_{k+1}(\mathbf{S}_{1(k+1)}(N_g,n),\boldsymbol{m})$ [par (1)]
  $= \mathbf{app}_{k+1}(\mathbf{app}_2(N_\mathbf{S},n),\boldsymbol{m})$ [par (2)]
- On pose $N \stackrel{\text{def}}{=} \mathbf{S}_{1(k+1)}(N_g,N_\mathbf{S})$: bien défini car $\mathbf{S}_{1(k+1)}$ totale; $f(N)$ aussi car f **totale**
- Par (2), $N = \mathbf{app}_2(N_\mathbf{S},N_\mathbf{S})$; donc, en posant $n \stackrel{\text{def}}{=} N_\mathbf{S}$, pour tout $\boldsymbol{m}$:
  $\mathbf{app}_{k+1}(f(N),\boldsymbol{m}) = \mathbf{app}_{k+1}(f(\mathbf{app}_2(N_\mathbf{S},N_\mathbf{S})),\boldsymbol{m})$
  $= \mathbf{app}_{k+1}(\mathbf{app}_2(N_\mathbf{S},N_\mathbf{S}),\boldsymbol{m}) = \mathbf{app}_{k+1}(N,\boldsymbol{m})$. □

Corollaire: A est récursive (3ème preuve)

Théorème. $\forall k$, $\forall$ fonction récursive **totale** f d'arité 1,
$\exists N \in \mathbf{N} / f(N) \cong N \quad [\forall \boldsymbol{m} \in \mathbf{N}^k, \mathbf{app}_{k+1}(f(N),\boldsymbol{m}) = \mathbf{app}_{k+1}(N,\boldsymbol{m})]$

- « `let rec` $F(\boldsymbol{m}) = f(F)(\boldsymbol{m})$ »
  où $f(F)(0,m) \stackrel{\text{def}}{=} m+1$
  $f(F)(n+1,m) \stackrel{\text{def}}{=} F(n,1)$
  $f(F)(n+1,m+1) \stackrel{\text{def}}{=} F(n,F(n+1,m))$

$A(0,m) \stackrel{\text{def}}{=} m+1$
$A(n+1,0) \stackrel{\text{def}}{=} A(n,1)$
$A(n+1,m+1) \stackrel{\text{def}}{=} A(n,A(n+1,m))$

- *Preuve* (1/2). Soit $g : \mathbf{N}^3 \to \mathbf{N}$ définie par:
  $g(N,0,m) \stackrel{\text{def}}{=} m+1$
  $g(N,n+1,0) \stackrel{\text{def}}{=} \mathbf{app}_3(N,n,1)$
  $g(N,n+1,m+1) \stackrel{\text{def}}{=} \mathbf{app}_3(N,n,\mathbf{app}_3(N,n+1,m))$

et soit N_g un index de g:
$\forall N,n,\boldsymbol{m}, g(N,n,m) = \mathbf{app}_4(N_g,N,n,m)$

- Théorème Smn: $\forall N,n,\boldsymbol{m}, g(N,n,m) = \mathbf{app}_3(\mathbf{S}_{22}(N_g,N),n,m)$
- On applique le théorème de récursion à $f : N \mapsto \mathbf{S}_{22}(N_g,N)$

totale car p.r.

- Soit N_A / $f(N_A) \cong N_A$
  $\forall n,m, \mathbf{app}_3(N_A,n,m) = \mathbf{app}_3(f(N_A),n,m) = g(N_A,n,m) = \ldots$

Corollaire: A est récursive (3ème preuve)

- *Preuve* (2/2). On rappelle que $g : \mathbf{N}^3 \to \mathbf{N}$ est définie par:

  $g(N,0,m) \stackrel{\text{def}}{=} m+1$

  $g(N,n+1,0) \stackrel{\text{def}}{=} \mathbf{app}_3(N,n,1)$

  $g(N,n+1,m+1) \stackrel{\text{def}}{=} \mathbf{app}_3(N,n,\mathbf{app}_3(N,n+1,m))$

$$A(0,m) \stackrel{\text{def}}{=} m+1$$
$$A(n+1,0) \stackrel{\text{def}}{=} A(n,1)$$
$$A(n+1,m+1) \stackrel{\text{def}}{=} A(n,A(n+1,m))$$

- Soit N_A / $f(N_A) \cong N_A$

  $\forall n,m,\ \mathbf{app}_3(N_A,n,m) = \mathbf{app}_3(f(N_A),n,m) = g(N_A,n,m) = \ldots$

  - $\mathbf{app}_3(N_A,0,m) = m+1$
  - $\mathbf{app}_3(N_A,n+1,0) = \mathbf{app}_3(N_A,n,1)$
  - $\mathbf{app}_3(N_A,n+1,m+1) = \mathbf{app}_3(N_A,n,\mathbf{app}_3(N_A,n+1,m))$

- La fonction $n,m \mapsto \mathbf{app}_3(N_A,n,m)$ satisfait donc aux mêmes équations que A: par récurrence double sur n,m, on en déduit $\mathbf{app}_3(N_A,n,m)=A(n,m)$
- Comme $n,m \mapsto \mathbf{app}_3(N_A,n,m)$ est récursive, A **aussi**. □

That’s all folks!