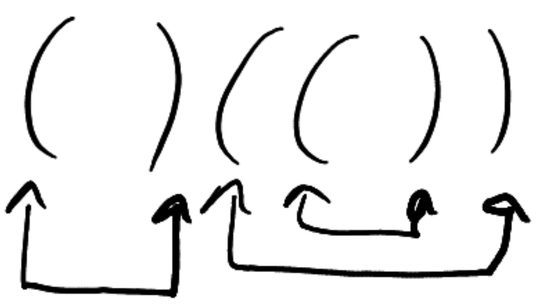
Mots bien parenthésés

Motivation : Exemple très classique permettant de se familiariser avec la notion d'induction.

Exemple : $(5+2)\times(3\times(1+1))$

↳  : l'ordre des parenthèses n'est pas évident à première vue.

Définition : Soit $D \subseteq \Sigma^*$ avec $\Sigma = \{a, b\}$ définie par : $\dfrac{}{\varepsilon \in D}$ ax $\quad \dfrac{u \in D \quad v \in D}{uv \in D}$ conc $\quad \dfrac{u \in D}{a u b \in D}$ par

D est appelé l'ensemble des mots de Dyck.

Exemples : $\dfrac{\dfrac{\dfrac{\varepsilon}{ab} \quad \dfrac{\varepsilon}{aabb}}{abaabb}}{}$ donc $abaabb \in D$.

$abba \notin D$.

↳ $|w|_a = |w|_b \Rightarrow w \in D$

Propriété : $\mathcal{O}_a$ a : $w \in D$ ssi

(1) $|w|_a = |w|_b$

(2) $\forall u \sqsubseteq w$, $|u|_a \geq |u|_b$. ($\sqsubseteq$: préfixe)

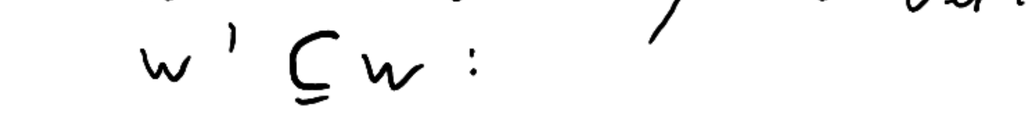
Preuve : $\Rightarrow$ Par induction structurelle sur D :

(ax) : ok

(par) : Soit $u \in D$ vérifiant (1) et (2).

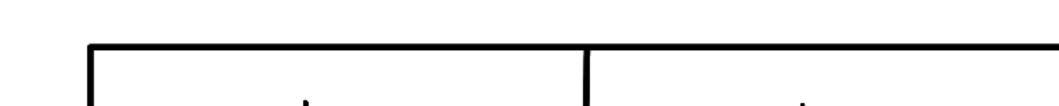
Soit $v = aub$: mq v vérifie (1) et (2).

Soit $v' \sqsubseteq v$:



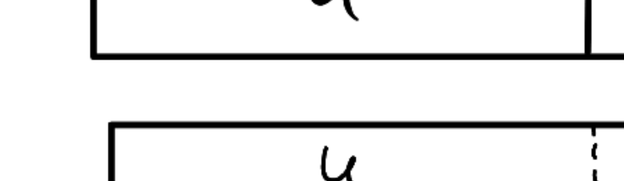
$= v$

1er cas



$= v'$

2nd cas



$= v'$

Par HI, on a dans les deux cas $|v'|_a \geq |v'|_b$.

(Conc) : Soit u et v vérifiant (1) et (2).

$w = uv$: mq w vérifiant (1) et (2).

$w' \sqsubseteq w$:

$= w$

1er cas

$= w'$

2nd cas

$= w'$

Par HI sur u et v, on a $|w'|_a \geq |w'|_b$

$\Leftarrow$ Soit w vérifiant (1) et (2).

On raisonne par récurrence forte sur $n = |w|$.

- Initialisation : $\varepsilon \in D$
- Hérédité : Soit $n \in \mathbb{N}$ tq la propriété est vraie pour $0, 1, \dots, n$. Soit $w \in \Sigma^{n+1}$: $w = w_0 w_1 \cdots w_n$.

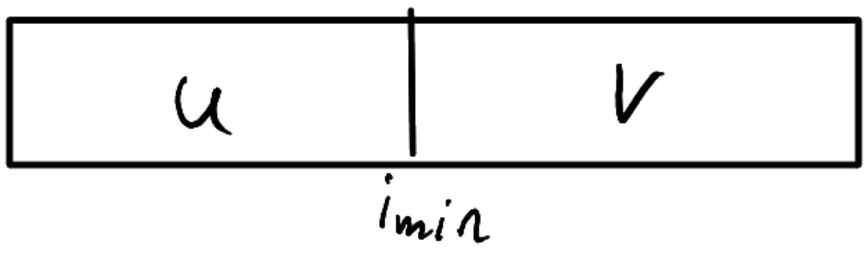
$$\varphi : [\![0; n]\!] \longrightarrow \mathbb{N}$$
$$i \longmapsto |w_0 \cdots w_i|_a - |w_0 \cdots w_i|_b$$

$i_{min} = \min(\{ j \mid \varphi(j) = 0 \})$

(bien défini car $\varphi(n) = 0$)

Si $i_{min} = n$, alors la parenthèse fermante b correspondante à $a = w_0$ est w_n, donc $w = a\,w'\,b \longrightarrow$ HR sur w' puis (par).

Sinon, $w =$



avec $u = w_0 \cdots w_{i_{min}}$ et $v = w_{i_{min}+1} \cdots w_n$

et u et v vérifient (1) et (2) : HR sur u et v. puis (conc).

Programme Python pour reconnaître un mot de Dyck :

```
def est_de_dyck(w):
    pile = 0
    for c in w:
        if c == "a":
            pile += 1
        else:
            pile -= 1
        if pile < 0:
            return False
    return pile == 0
```