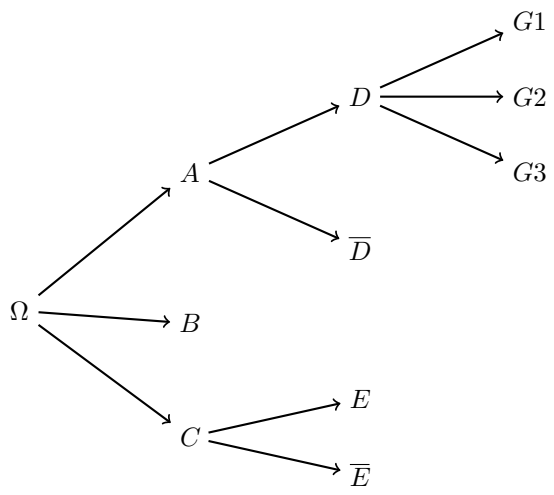
Créer un arbre de probas, en Ti*k*Z, avec python

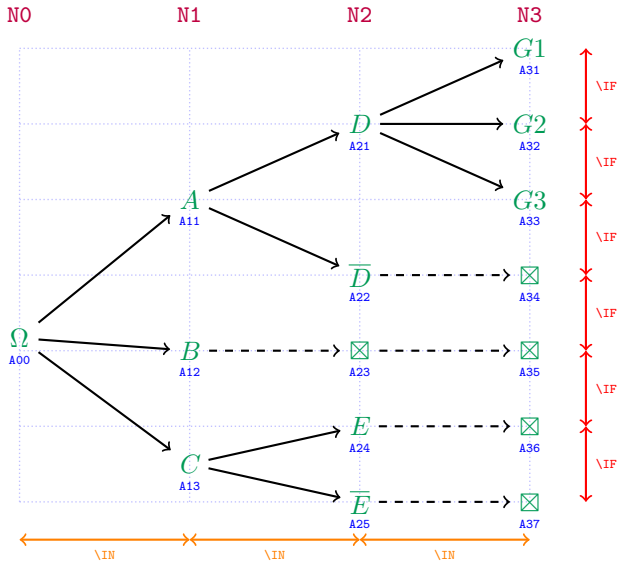
1 Petite introduction avec un arbre « biscornu »

1.1 Ce qu'on veut obtenir

On souhaite tracer l'arbre de probabilité suivant :



On va « compléter » l'arbre afin de visualiser les positions des *nœuds fantômes* :



1.2 Une idée...

Une idée – je pense que le site `http://math.et.info.free.fr/TikZ/Arbre/index.html` fonctionne sur ce principe, en java il me semble – est de partir des feuilles situées sur la fin de l'arbre, et de « remonter » les niveaux pour positionner les nœuds précédents.

Bien évidemment, on doit pouvoir faire mieux et plus rapidement, mais l'idée de faire cela en LaTeX pur m'intéressait... Finalement ça m'a paru compliqué en LaTeX, et j'ai préféré passer par du python pour générer le code, snif...

Je pense même qu'avec beaucoup plus de connaissances, on doit pouvoir optimiser grandement le fonctionnement :

- en utilisant de la récursivité ;
- en utilisant des dictionnaires ;
- ...

2 Présentation du fonctionnement

2.1 Les paramètres

Pour *déclarer* l'arbre, on peut utiliser un paramétrage des branches « niveau par niveau » :

- `N1 := [3]`
- `N2 := [2,1,2]`
- `N3 := [3,1,1,1,1]`

En fait on déclare les branches par « bloc » en fonction des nœuds du niveau précédent, et si le sous-arbre s'arrête, on complète par des sommets et branches fictifs :

- pour `N1`, il y a `3` branches ;
- pour `N2`, il y a `2` puis `1` puis `2` branches ;
- pour `N3`, il y a `3` puis `1` puis `1` puis `1` puis `1` branches.

Pour les sommets, on travaille niveau par niveau, avec des sommets fictifs déclarés par `.` et les contraires par `-X` :

- `S0 := [\Omega]`
- `S1 := [A,B,C]`
- `S2 := [D,-D,.,E,-E]`
- `S3 := [G1,G2,G3,.,.,.,.]`

2.2 Points de repère

Il y a donc homogénéité entre les listes des branches, les `N...` et les listes des sommets, les `S...`, ce qui permet, par une « pseudo-récursivité », de calculer la position verticale des sommets (par une sorte de moyenne).

branches	[3]	[2 ,1,2]	[3 ,1,1,1,1]
sommets	[A,B,C]	[D,-D,.,E,-E]	[G1,G2,G3,.,.,.,.]

Les sommets (ou nœuds) sont déclarés et nommés sous la forme `Aij` avec `i` le niveau et `j` commençant à 0, et en *descendant*, ce qui crée les concordances suivantes :

branches		[3]	[2 ,1 ,2]	[3 ,1 ,1 ,1 ,1]
sommets	Ω	[A ,B ,C]	[D ,-D ,. ,E ,-E]	[G1 ,G2 ,G3 ,. ,. ,. ,.]
nœuds	A00	[A11,A12,A13]	[A21,A22,A23,A24,A25]	[A31,A32,A33,A34,A35,A36,A37]

De ce fait, les flèches peuvent être déclarées grâce à leur `nœud de départ` et leur `nœud d'arrivée`. On peut même les pondérer si besoin :

- pour un arc non, on pourra utiliser `ij/kl` pour spécifier une branche « vierge » `Aij -> Akl` ;
- pour un arc pondéré, on pourra utiliser `ij/kl/p` pour spécifier une branche « avec la proba `p` » `Aij -> Akl` ;
- si la proba est préfixée par `n0,...`, ce sera automatiquement formaté en `\num{0,...}` ;
- si la proba est écrite en `f{a}{b}`, ce sera automatiquement formaté en `\frac{a}{b}`.

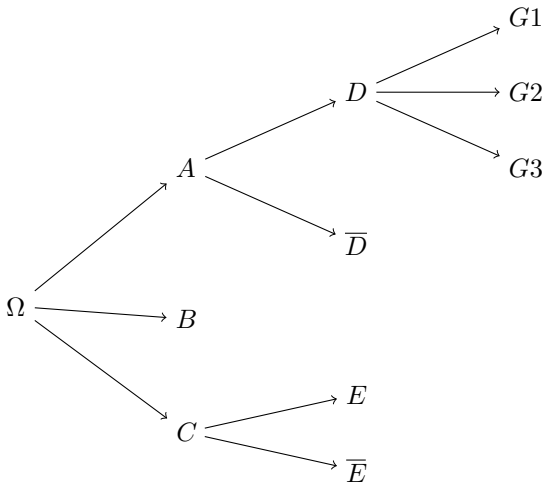
2.3 Le script python

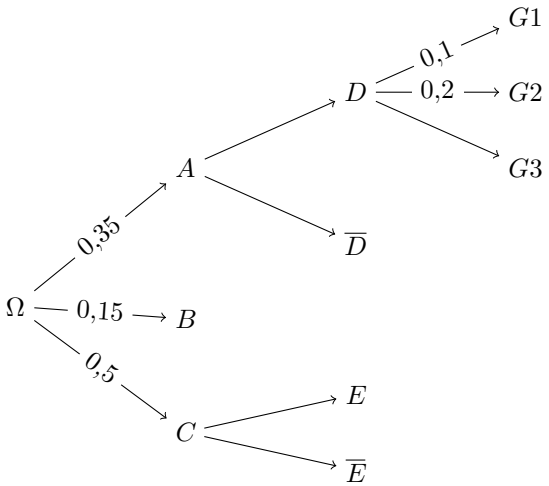
On va maintenant passer sur python pour générer le code Ti*k*Z :

```
# saisie des paramètres
listebranches = [[3],[2,1,2],[3,1,1,1,1]]
listesommets = ['\Omega',['A','B','C'],['D','-D','.','E','-E'],['G1','G2','G3','.','.','.','.']]
espaceniveau = 2.25
espacesommets = 1
# génération du code TiKZ pour un arbre non pondéré
flechesvides = '00/11,00/12,00/13,11/21,11/22,13/24,13/25,21/31,21/32,21/33'
affnoeuds(listebranches,listesommets,espaceniveau,espacesommets,flechesvides)
# génération du code TiKZ pour un arbre pondéré
flechesprobas = '00/11/n0.35,00/12/n0.15,00/13/n0.5,11/21,11/22,13/24,13/25,21/31/n0.1,21/32/n0.2,21/33'
affnoeuds(listebranches,listesommets,espaceniveau,espacesommets,flechesprobas)
```

```
# génération du code TiKZ pour un arbre non pondéré
\begin{tikzpicture}
    %--styles--
    \tikzstyle{noeud}=[]
    \tikzstyle{fleche}=[->]
    \tikzstyle{etiquette}=[sloped,midway,fill=white]
    \def\IN{2.25} %distance interniveau
    \def\IF{1} %distance interfeuilles
    %--Racine--
    \node[noeud] (A00) at (0,-3.8333333333333335) {$\Omega$} ;
    %--Niveau 1--
    \node[noeud] (A11) at ({\IN*(1)},{-\IF*2.0}) {$A$} ;
    \node[noeud] (A12) at ({\IN*(1)},{-\IF*4.0}) {$B$} ;
    \node[noeud] (A13) at ({\IN*(1)},{-\IF*5.5}) {$C$} ;
    %--Niveau 2--
    \node[noeud] (A21) at ({\IN*(2)},{-\IF*1.0}) {$D$} ;
    \node[noeud] (A22) at ({\IN*(2)},{-\IF*3.0}) {$\overline{D}$} ;
    \coordinate (A23) at ({\IN*(2)},{-\IF*4.0}) ;
    \node[noeud] (A24) at ({\IN*(2)},{-\IF*5.0}) {$E$} ;
    \node[noeud] (A25) at ({\IN*(2)},{-\IF*6.0}) {$\overline{E}$} ;
    %--Niveau 3--
    \node[noeud] (A31) at ({\IN*(3)},{-\IF*0}) {$G1$} ;
    \node[noeud] (A32) at ({\IN*(3)},{-\IF*1}) {$G2$} ;
    \node[noeud] (A33) at ({\IN*(3)},{-\IF*2}) {$G3$} ;
    \coordinate (A34) at ({\IN*(3)},{-\IF*3}) ;
    \coordinate (A35) at ({\IN*(3)},{-\IF*4}) ;
    \coordinate (A36) at ({\IN*(3)},{-\IF*5}) ;
    \coordinate (A37) at ({\IN*(3)},{-\IF*6}) ;
    %--branches--
    \draw[fleche] (A00)--(A11) ;
    \draw[fleche] (A00)--(A12) ;
    \draw[fleche] (A00)--(A13) ;
    \draw[fleche] (A11)--(A21) ;
    \draw[fleche] (A11)--(A22) ;
    \draw[fleche] (A13)--(A24) ;
    \draw[fleche] (A13)--(A25) ;
    \draw[fleche] (A21)--(A31) ;
    \draw[fleche] (A21)--(A32) ;
    \draw[fleche] (A21)--(A33) ;
\end{tikzpicture}
```

```
# génération du code TiKZ pour un arbre pondéré
\begin{tikzpicture}
    %--styles--
    \tikzstyle{noeud}=[]
    \tikzstyle{fleche}=[->]
    \tikzstyle{etiquette}=[sloped,midway,fill=white]
    \def\IN{2.25} %distance interniveau
    \def\IF{1} %distance interfeuilles
    %--Racine--
    \node[noeud] (A00) at (0,-3.8333333333333335) {$\Omega$} ;
    %--Niveau 1--
    \node[noeud] (A11) at ({\IN*(1)},{-\IF*2.0}) {$A$} ;
    \node[noeud] (A12) at ({\IN*(1)},{-\IF*4.0}) {$B$} ;
    \node[noeud] (A13) at ({\IN*(1)},{-\IF*5.5}) {$C$} ;
    %--Niveau 2--
    \node[noeud] (A21) at ({\IN*(2)},{-\IF*1.0}) {$D$} ;
    \node[noeud] (A22) at ({\IN*(2)},{-\IF*3.0}) {$\overline{D}$} ;
    \coordinate (A23) at ({\IN*(2)},{-\IF*4.0}) ;
    \node[noeud] (A24) at ({\IN*(2)},{-\IF*5.0}) {$E$} ;
    \node[noeud] (A25) at ({\IN*(2)},{-\IF*6.0}) {$\overline{E}$} ;
    %--Niveau 3--
    \node[noeud] (A31) at ({\IN*(3)},{-\IF*0}) {$G1$} ;
    \node[noeud] (A32) at ({\IN*(3)},{-\IF*1}) {$G2$} ;
    \node[noeud] (A33) at ({\IN*(3)},{-\IF*2}) {$G3$} ;
    \coordinate (A34) at ({\IN*(3)},{-\IF*3}) ;
    \coordinate (A35) at ({\IN*(3)},{-\IF*4}) ;
    \coordinate (A36) at ({\IN*(3)},{-\IF*5}) ;
    \coordinate (A37) at ({\IN*(3)},{-\IF*6}) ;
    %--branches--
    \draw[fleche] (A00)--(A11) node[etiquette] {\num{0.35}} ;
    \draw[fleche] (A00)--(A12) node[etiquette] {\num{0.15}} ;
    \draw[fleche] (A00)--(A13) node[etiquette] {\num{0.5}} ;
    \draw[fleche] (A11)--(A21) ;
    \draw[fleche] (A11)--(A22) ;
    \draw[fleche] (A13)--(A24) ;
    \draw[fleche] (A13)--(A25) ;
    \draw[fleche] (A21)--(A31) node[etiquette] {\num{0.1}} ;
    \draw[fleche] (A21)--(A32) node[etiquette] {\num{0.2}} ;
    \draw[fleche] (A21)--(A33) ;
\end{tikzpicture}
```

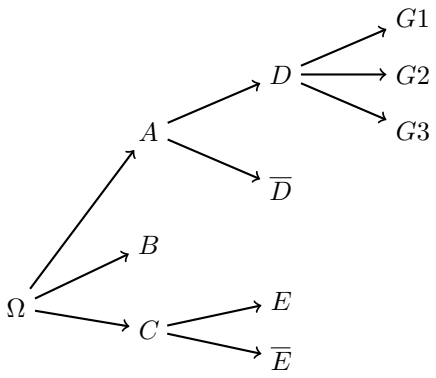




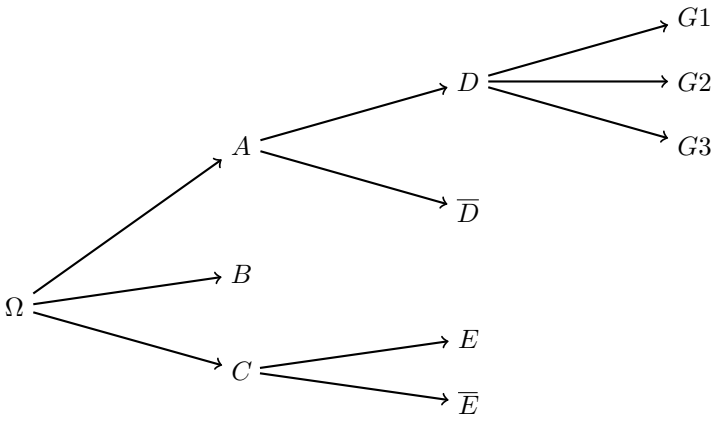
2.4 Paramétrage

Les styles créés ainsi que les dimensions peuvent facilement être modifiées, à sa guise !

- `IN=1.75 et IF=0.75`

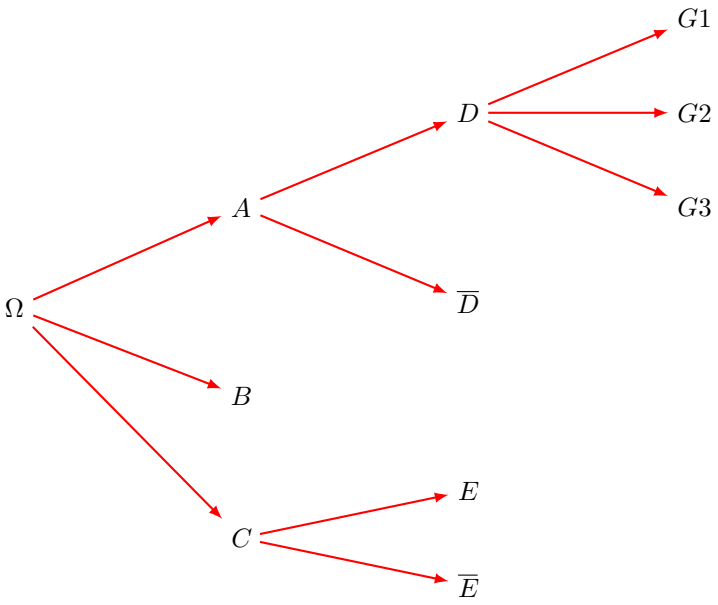


- `IN=3 et IF=0.85`



- `IN=2.25 et IF=1.25`

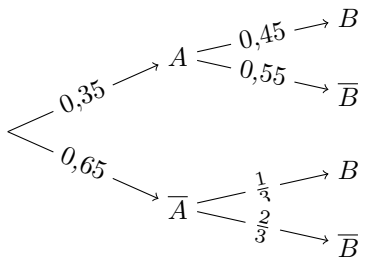
  `\tikzstyle{fleche}=[red,thick,->,>=latex]`



3 Des cas plus « classiques »

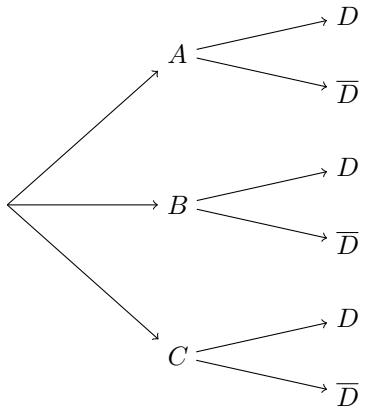
3.1 Un super classique : le 2x2

```
listebranches=[[2],[2,2]]
listesommets=['+',['A','-A'],['B','-B','B','-B']]
espaceniveau = 2.25
espacesommets = 1
flechesprobas = '00/11/n0.35,00/12/n0.65,11/21/n0.45,11/22/n0.55,12/23/f{1}{3},12/24/f{2}{3}'
affnoeuds(listebranches,listesommets,espaceniveau,espacesommets,flechesprobas)
```



3.2 Un autre classique, le 3x2

```
listebranches=[[3],[2,2,2]]
listesommets=['.',['A','B','C'],['D','-D','D','-D','D','-D']]
espaceniveau = 2.25
espacesommets = 1
flechesprobas = '00/11,00/12,00/13,11/21,11/22,12/23,12/24,13/25,13/26'
affnoeuds(listebranches,listesommets,espaceniveau,espacesommets,flechesprobas)
```



4 Pistes pour améliorer

- Meilleure gestion de la déclaration des sommets ? Par dictionnaire ?
- Meilleure gestion de la déclaration des branches ? Idem ?

5 Script python

```
def testbranches(liste) :                # pour tester si la liste des branches est valable !
    valid = True
    for i in range(len(liste)-1) :
        if len(liste[-i-1]) != sum(liste[-i-2]) : valid = False
    return(valid)

def conversion(chaine,car):              # pour transformer une liste, avec délimiteur
    li = list(chaine.split(car))
    return li

def conversionchaineentier(chaine,car):  # pour transformer une liste d'entiers, avec délimiteur
    res = list(chaine.split(car))
    res = [int(i) for i in res]
    return res

def coordniveau(liste) :                 # pour calculer les coordonnées des nœuds
    nbniveau = len(liste)
    listeordonnees = [0] + [ [0] for i in range(nbniveau)]
    listeordonnees[-1] = [i for i in range(sum(liste[-1]))]
    for i in range(2,nbniveau+1) :
        ordoniv,tmpA,tmpB = [0],liste[-i+1],listeordonnees[-i+1]
        while len(tmpA) != 0 :
            calc = tmpA[0]
            ordoniv.append(sum(tmpB[:calc])/calc)
            tmpA,tmpB = tmpA[1:],tmpB[calc:]
        listeordonnees[-i] = ordoniv
    listeordonnees[0] = sum(listeordonnees[1])/len(listeordonnees[1])
    return(listeordonnees)

def affnoeuds(liste,sommets,interniveau,interfeuille,branches) :
    if testbranches(liste) == False :
        print("Erreur de saisie")
    else :
        res = coordniveau(liste)
        print("\\begin{tikzpicture}")
        print(r"    %--styles--")
        print(r"    \tikzstyle{noeud}=[]")
        print(r"    \tikzstyle{fleche}=[->]")
        print(r"    \tikzstyle{etiquette}=[sloped,midway,fill=white]")
        print(f"    \\def\IN{{{interniveau}}}\\def\IF{{{interfeuille}}} %distances niveaux et feuilles")
        print(r"    %--Racine--")
        if sommets[0] == '.' :            # si c'est un sommet "vide"
            print(f"    \\coordinate (A00) at (0,{-res[0]}) ;")
        else :
            print(f"    \\node[noeud] (A00) at (0,{-res[0]}) {{${sommets[0]}$}} ;")
        for i in range(1,len(res)) :
            print(f"    %--Niveau {i}--")
            for j in range(len(res[i])) :
                if sommets[i][j] != "." :          # c'est un "vrai" sommet"
                    if sommets[i][j][0] == "-" :   # c'est un contraire
                        nomsommet = "\\overline{" + sommets[i][j][1:] + "}"
                    else :
                        nomsommet = sommets[i][j]
                    print(f"    \\node[noeud] (A{i}{j+1}) at ({{\IN*({i})}},{{-\IF*{res[i][j]}}}) {{${nomsommet}$}} ;")
                else :
                    print(f"    \\coordinate (A{i}{j+1}) at ({{\IN*({i})}},{{-\IF*{res[i][j]}}}) ;")
        print("    %--branches--")
        fleches = conversion(branches,',')
        for i in range(len(fleches)) :
            donnees = conversion(fleches[i],'/')
            if len(donnees) == 2 :
                print(f"    \draw[fleche] (A{donnees[0]})--(A{donnees[1]}) ;")
            else :
                proba = donnees[2]
                if 'n0' in donnees[2] : proba = '\\num{' + donnees[2][1:] + '}'
                if 'f{' in donnees[2] : proba = '$\\frac' + donnees[2][1:] + '$'
                print(f"    \draw[fleche] (A{donnees[0]})--(A{donnees[1]}) node[etiquette] {{" + proba + "} ;")
        print("\\end{tikzpicture}")
```