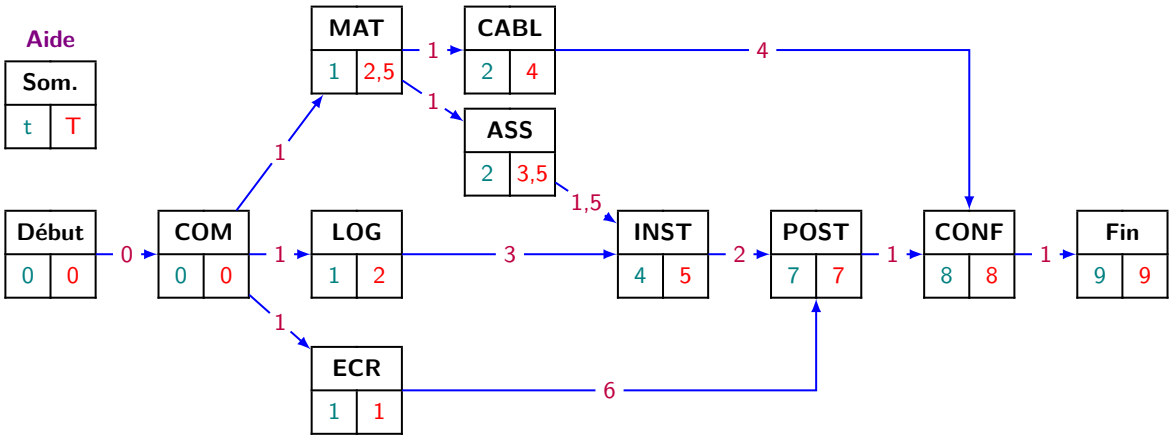
ProfSio [fr]

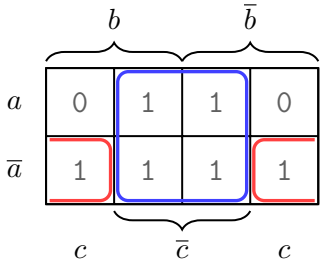
Des outils pour les Maths en BTS SIO.

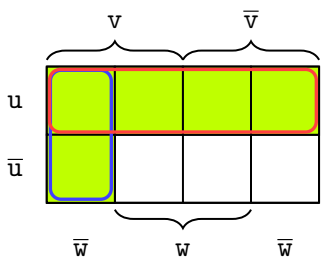
Version 0.1.9 -- 13 décembre 2023

Cédric Pierquet (c pierquet -- at -- outlook . fr)
https://github.com/cpierquet/profsio

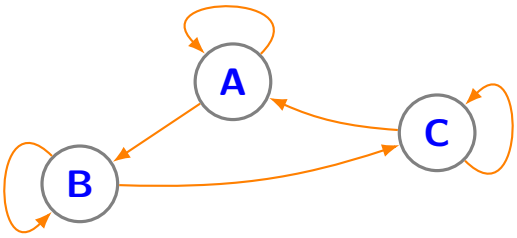
- Commandes spécifiques pour le programme de Mathématiques en BTS SIO [1].
- Créer des diagrammes MPM [2] (Méthode des Potentiels Métra).
- Créer (et simplifier) des tables de Karnaugh avec mise en valeur (manuelle) des regroupements.
- Créer des graphes simples ou des diagrammes sagittaux, travailler sur les matrices.
- Créer des tables de vérité (via LuaLaTeX) grâce au code du package `luatruthtable` [3].

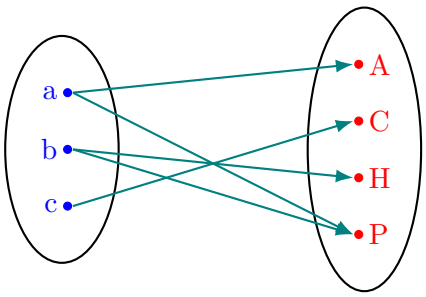






P	Q	$\neg P$	$P \vee Q$	$P \Rightarrow Q$
V	V	F	V	V
V	F	F	F	F
F	V	V	F	V
F	F	V	F	V





1. Brevet de Technicien Supérieur - Services Informatiques aux Organisations : [Lien] sur le site de L'Étudiant
2. Méthode des Potentiels Métra : [Lien] sur le site de Wikipedia
3. Package LaTeX : [Lien] sur le site du CTAN

Table des matières

1 Historique **4**

2 Le package ProfSio **5**
2.1 Introduction 5
2.2 Chargement du package, packages utilisés 5
2.3 Fonctionnement global 5

3 Graphe d'ordonnancement par méthode MPM **6**
3.1 Commande et fonctionnement global 6
3.2 Arguments et clés pour l'environnement 7
3.3 Arguments et clés pour les tâches 8
3.4 Arguments et clés pour les tâches 9
3.5 Exemples 11

4 Tableau de Karnaugh à trois variables **12**
4.1 Commande et fonctionnement global 12
4.2 Arguments et clés pour l'environnement 13
4.3 Arguments et clés pour la commande de remplissage 13
4.4 Arguments et clés pour la commande de regroupement des blocs 15
4.5 Exemples 15

5 Simplification d'une expression booléenne par table de Karnaugh **17**
5.1 Commande et fonctionnement global 17
5.2 Arguments et clés 18
5.3 Exemples 18

6 Écriture et simplification d'une expression booléenne **20**
6.1 Commande et fonctionnement global 20
6.2 Arguments et clés 20
6.3 Exemples 20

7 Graphes *simples* **21**
7.1 Commande et fonctionnement global 21
7.2 Arguments et clés pour l'environnement 22
7.3 Arguments et clés pour la commande de création des sommets 23
7.4 Arguments et clés pour la commande de tracé des arêtes 24
7.5 Exemples 26

8 Matrice d'adjacence, fermeture transitive **27**
8.1 Commandes et fonctionnement global 27
8.2 Matrice d'adjacence, puissance 27
8.3 Chemins de longueur donnée 28
8.4 Matrice de fermeture transitive 29

9 Diagramme sagittal d'une application **31**
9.1 Commande et fonctionnement global 31
9.2 Arguments et clés 31
9.3 Exemples 32

10 Diagramme sagittal d'une composée d'applications **34**
10.1 Commande et fonctionnement global 34
10.2 Arguments et clés 34
10.3 Exemples 36

11 Table de vérité **38**
11.1 Commande et fonctionnement global 38
11.2 Arguments et clés pour la commande 38
11.3 Compléments pour le package luatruthtable 39
11.4 Exemples 39

1 Historique

`v0.1.9` : Travail sur les matrices d'adjacence (chemins, puissances, fermeture)
`v0.1.8` : Possibilité de créer le tableau de Karnaugh via une expression booléenne + Corrections mineures
`v0.1.7` : Possibilité de simplifier une expression booléenne *directement* + amélioration des espaces
`v0.1.6` : Correction dans les simplifications de Karnaugh + Simplification du contraire
`v0.1.5` : Commande pour simplifier une table de Karnaugh à trois variables
`v0.1.4` : Possibilité de remplir une table de Karnaugh sans virgule
`v0.1.3` : Style alternatif et Clé ⟨PoliceTT⟩ pour les tables de Karnaugh
`v0.1.2` : Clé ⟨Offset⟩ pour les diagrammes sagittaux + Diagrammes sagittaux de composées.
: Ajout des tables de vérité (via LuaLaTeX).
`v0.1.1` : Mise à jour de la documentation + Diagrammes sagittaux.
`v0.1.0` : Version initiale.

2 Le package ProfSio

2.1 Introduction

Le package ProfSio propose quelques commandes pour travailler sur des points particuliers de Mathématiques enseignées en BTS SIO :

- graphes d'ordonnancement par la méthode MPM ;
- expressions booléennes et tableaux de Karnaugh pour 3 variables ;
- graphes *simples* orientés ou pondérés, des diagrammes sagittaux ;
- tables de vérité (via LuaLaTeX).

Le code ne propose par de « résolution » du graphe MPM ou de représentation « automatique » d'un graphe, il ne consiste *qu'en* une mise en forme du graphe MPM, du tableau de Karnaugh ou du graphe.

Cependant, il est propose de « simplifier » des expressions booléennes, et pour les tables de vérité, le code se charge de créer le tableau entièrement, grâce aux données du package luatruthtable (légèrement *patchées*.

2.2 Chargement du package, packages utilisés

Le package se charge, de manière classique, dans le préambule.
Il n'existe pas d'option pour le package, et xcolor n'est pas chargé.

```
\documentclass{article}
\usepackage{ProfSio}
```

ProfSio charge les packages suivantes :

- tikz, pgffor, xintexpr, tabularray, simplekv, xstring et listofitems ;
- luacode et nicematrix (uniquement si le compilateur détecté est LuaLaTeX) ;
- les librairies tikz :
  - tikz.positioning, tikz.babel, tikz.calc ;
  - tikz.decorations.pathreplacing et tikz.decorations.markings ;
  - tikz.shapes, tikz.shapes.geometric, tikz.arrows et tikz.arrows.meta.

Il est compatible avec les compilations usuelles en latex, pdflatex, lualatex (obligatoire pour les tables de vérité !) ou xelatex.

2.3 Fonctionnement global

Les environnements sont créés avec TikZ, et la majorité des paramètres des tracés sont personnalisables :

couleurs ; dimensions ; polices.

Le choix a été fait de :

- présenter l'ordonnancement par la méthode MPM, avec présentation des tâches *fixée* ;
- limiter les tableaux de Karnaugh pour 3 variables, avec présentation *fixée* ;
- de ne pas forcément proposer de modification de la présentation *globale*.

3 Graphe d'ordonnancement par méthode MPM

3.1 Commande et fonctionnement global

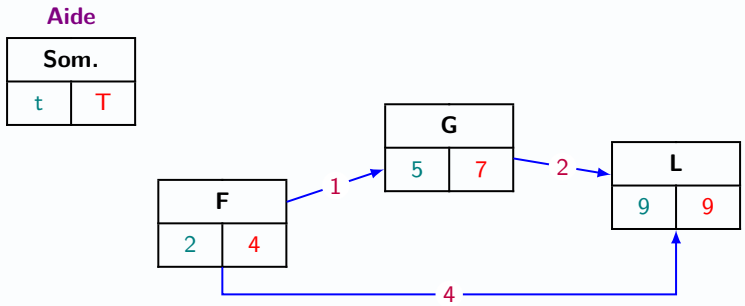
L'environnement dédié à la création du graphe d'ordonnancement est `GrapheMPM`.
C'est en fait un environnement `tikzpicture` personnalisé.

Les commandes à utiliser dans l'environnement sont :

- `\MPMPlaceNotice` ;
- `\MPMPlaceTache` ou `\MPMPlaceTaches` ;
- `\MPMPlaceDuree` ou `\MPMPlaceDurees`.

```
\begin{GrapheMPM}[clés]<options tikz>
    \MPMPlaceNotice(*)(coordonnées)
    \MPMPlaceTache(coordonnées)(Tâche)(Dates)
    \MPMPlaceTaches{ (coordA)(TâcheA)(DatesA) / (coordB)(TâcheB)(DatesB) / ... }
    \MPMPlaceDuree[clés]{TâcheA>TâcheB,durée}<options tikz>
    \MPMPlaceDurees[clés]{TâcheA>TâcheB,durée / TâcheC>TâcheD,durée }<options tikz>
\end{GrapheMPM}
```

```
\begin{GrapheMPM}
    \MPMPlaceNotice(-2,2.15)
    \MPMPlaceTaches{ (0,0)(F)(2,4) / (3,1)(G)(5,7) / (6,0.5)(L)(9,9) }
    \MPMPlaceDurees{F>G,1 / G>L,2}
    \MPMPlaceDuree[Coude,SensCoude=VHV]{F.south>L.south,4}<near start>
\end{GrapheMPM}
```



Les tâches sont créées sous forme de *tableau* et sont associées à des nœuds, nœuds qui servent ensuite à positionner les durées des tâches.

3.2 Arguments et clés pour l'environnement

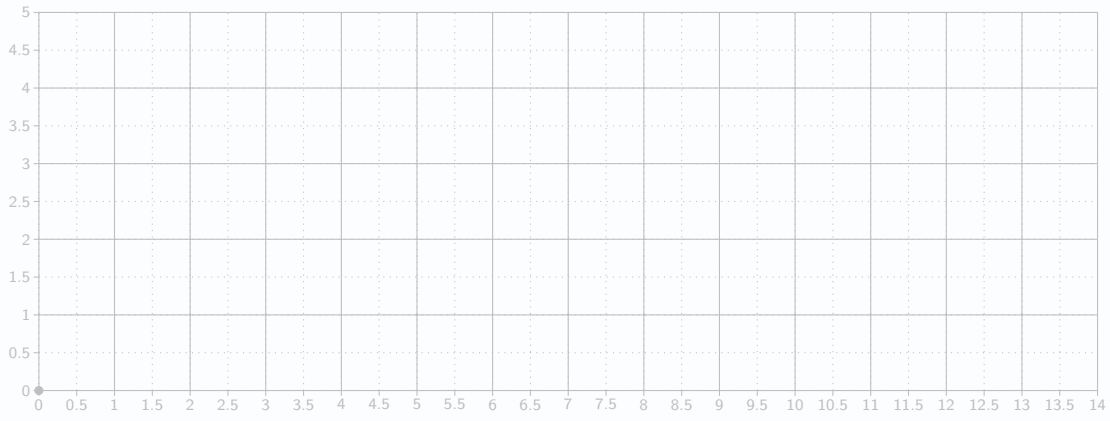
```
\begin{GrapheMPM}[clés]<options tikz>
    %commandes
\end{GrapheMPM}
```

En ce qui concerne la création de l'environnement, les ⟨clés⟩ sont :

- ⟨CouleurDurees⟩ := couleur des durée ; défaut : ⟨purple⟩
- ⟨CouleurFleches⟩ := couleur des arcs ; défaut : ⟨blue⟩
- ⟨LargeurCases⟩ := largeur des cases ; défaut : ⟨0.75cm⟩
- ⟨Epaisseur⟩ := épaisseur des traits (bordures et arcs) ; défaut : ⟨0.75pt⟩
- ⟨Police⟩ := police globale ; défaut : ⟨\footnotesize\sffamily⟩
- ⟨CouleurDates⟩ := couleur des dates, sous la forme ⟨Couleur⟩ ou ⟨Couleur_t/Couleur_T⟩ ; défaut : ⟨teal/red⟩
- ⟨CouleurBords⟩ := couleur des bordures ; défaut : ⟨black⟩
- ⟨NoirBlanc⟩ := booléen pour tout passer en Noir & Blanc ; défaut : ⟨false⟩
- ⟨Grille⟩ := pour afficher une grille d'aide (⟨{xmax,ymax}⟩), entre (0 ; 0) et (xmax ; ymax). défaut : ⟨vide⟩

Le deuxième argument, optionnel et entre `<...>` propose des options, en langage `tikz` à passer à l'environnement.

```
\begin{GrapheMPM}[Grille={14,5}]
    %commandes
\end{GrapheMPM}
```



3.3 Arguments et clés pour les tâches

```
\begin{GrapheMPM}[clés]<options tikz>
    \MPMPlaceNotice(*)(coordonnées)
    \MPMPlaceTache(coordonnées)(Tâche)(Dates)
    \MPMPlaceTaches{ (coordA)(TâcheA)(DatesA) / (coordB)(TâcheB)(DatesB) / ... }
\end{GrapheMPM}
```

La commande \MPMPlaceNotice permet de placer une *notice* :

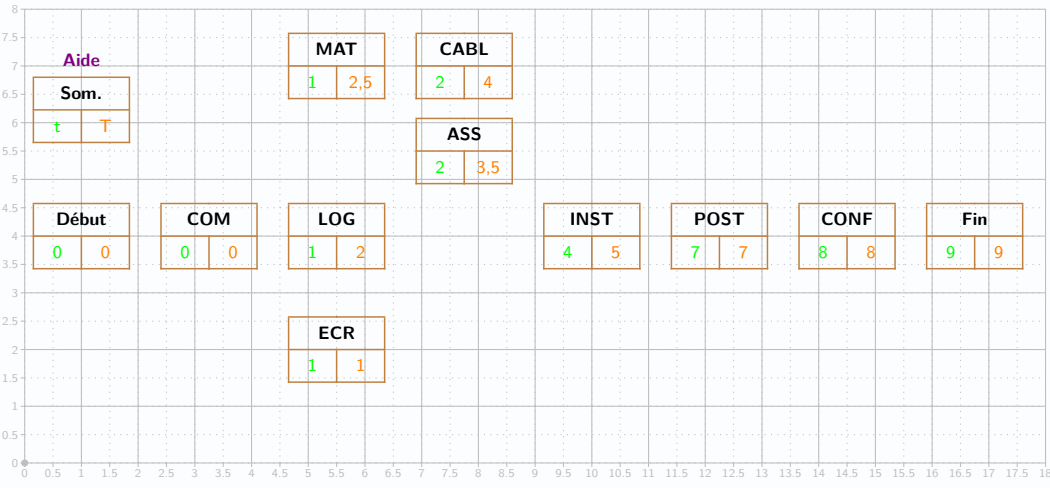
- la version *étoilée* affiche la notice complète, avec les dates et les marges (MT et ML) ;
- les coordonnées sont à donner sous la forme `x,y`.

La commande \MPMPlaceTache permet de placer une tâche :

- argument n°1 := coordonnées sont à donner sous la forme `x,y`.
- argument n°2 := nom de la tâche, qui sera également le nom du nœud ;
- argument n°3 := dates (et marges éventuelles) sous la forme :
  - `t,T` pour une tâche présentée de manière *simple* ;
  - `t,T,MT,ML` pour une tâche présentée de manière *complète* ;

La commande \MPMPlaceTaches permet de placer plusieurs tâches en utilisant la syntaxe de la commande précédente, les éléments de la liste étant séparés par le caractère `/`.

```
\begin{GrapheMPM}[CouleurDates=green/orange,CouleurBords=brown,Grille={18,8}]%
    <scale=0.75,transform shape>
    %NOTICE
    \MPMPlaceNotice(1,6.5)
    %TACHES INDIVIDUELLES
    \MPMPlaceTache(1,4)(Début)(0,0)
    \MPMPlaceTache(3.25,4)(COM)(0,0)
    %TACHES MULTIPLES
    \MPMPlaceTaches{ (5.5,4)(LOG)(1,2) / (5.5,2)(ECR)(1,1) / (5.5,7)(MAT)(1,2{,}5)
    / (7.75,7)(CABL)(2,4) / (7.75,5.5)(ASS)(2,3{,}5) / (10,4)(INST)(4,5) /
    (12.25,4)(POST)(7,7) / (14.5,4)(CONF)(8,8) / (16.75,4)(Fin)(9,9) }
\end{GrapheMPM}
```



3.4 Arguments et clés pour les tâches

```
\begin{GrapheMPM}[clés]<options tikz>
    %DÉCLARATION DES TÂCHES
    \MPMPlaceDuree[clés]{TâcheA>TâcheB,durée}<options tikz>
\end{GrapheMPM}
```

La commande `\MPMPlaceDuree` permet de placer un arc avec la durée de la tâche.

La commande propose les ⟨clés⟩ suivantes :

- ⟨Coude⟩ := booléen pour affiche l'arc sous forme d'un coude ; défaut : ⟨false⟩
- ⟨SensCoude⟩ := permet de préciser le type de coude, parmi ⟨HV / VH / VHV⟩ ; défaut : ⟨HV⟩
- ⟨HauteurCoude⟩ := dans le cas ⟨SensCoude=VHV⟩, permet de préciser le 1er décalage V ; défaut : ⟨10pt⟩
- ⟨DecalHorizDeb⟩ := décalage horizontal du début de l'arc pour la tâche de départ ;
- ⟨DecalVertDeb⟩ := décalage vertical du début de l'arc pour la tâche de départ ;
- ⟨DecalHorizDeb⟩ := décalage horizontal de la fin de l'arc pour la tâche d'arrivée ;
- ⟨DecalVertFin⟩ := décalage vertical de la fin de l'arc pour la tâche d'arrivée. défaut : ⟨0pt⟩

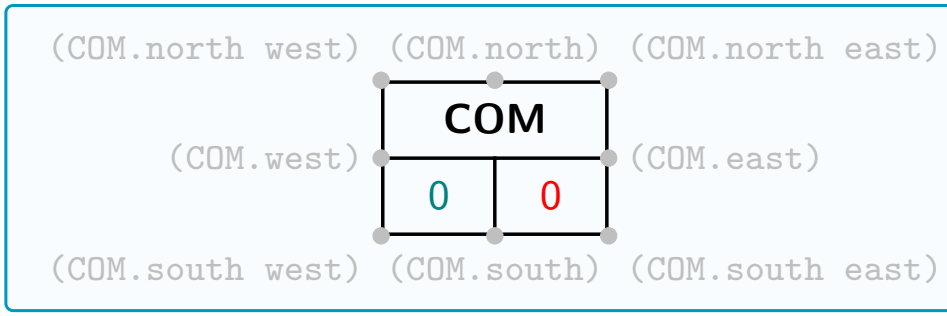
Le second argument, obligatoire et entre `{...}` permet de spécifier les paramètres de l'arc, sous la forme `TâcheDépart>TâcheArrivée,durée`.

Le troisième argument, optionnel et entre `<...>` et valant `midway` par défaut, permet de spécifier une position différente (en langage `tikz`) de la durée (comme par exemple `near start`, `near end` ou `pos=...`).

Les nœuds créés précédemment permettent donc de spécifier les arguments de la commande, et *tout point d'ancrage* des nœuds peuvent être utilisés pour la commande.

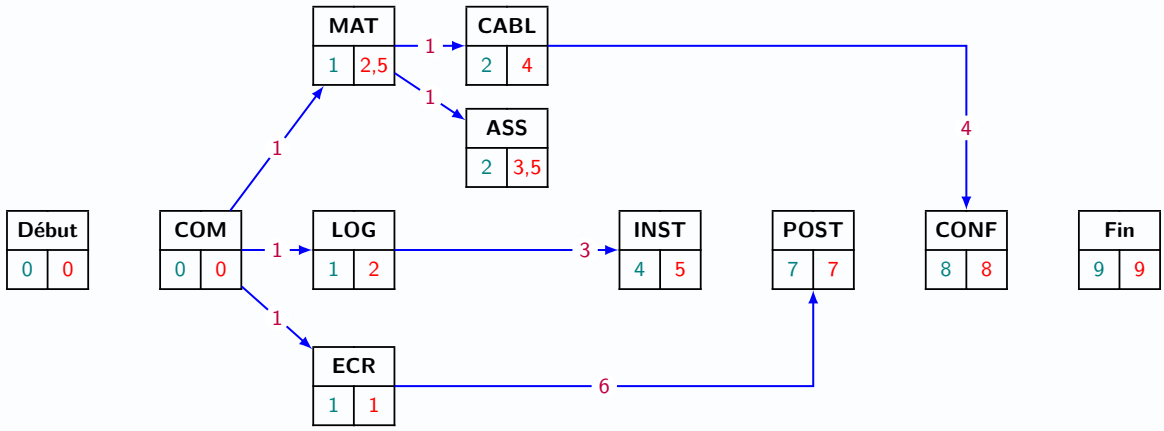
On rappelle que les principaux points d'ancrage d'un nœud `(NOEUD)` Ti*k*Z sont :

- `(NOEUD.north)`, `(NOEUD.east)`, `(NOEUD.south)`, `(NOEUD.west)` ;
- `(NOEUD.north east)`, `(NOEUD.south east)`, `(NOEUD.south west)`, `(NOEUD.north west)`.



Par défaut, les arcs pointent vers le *centre* du nœud, donc dans le cas d'arcs *coudés*, on peut utiliser des points d'ancrage pour une position optimale des arcs.

```
\begin{GrapheMPM}[LargeurCases=0.5cm]<scale=0.9,transform shape>
    %TACHES MULTIPLES
    \MPMPlaceTaches{ (1,4)(Début)(0,0) / (3.25,4)(COM)(0,0) / (5.5,4)(LOG)(1,2) /
    (5.5,2)(ECR)(1,1) / (5.5,7)(MAT)(1,2{,}5) / (7.75,7)(CABL)(2,4) /
    (7.75,5.5)(ASS)(2,3{,}5) / (10,4)(INST)(4,5) / (12.25,4)(POST)(7,7) /
    (14.5,4)(CONF)(8,8) / (16.75,4)(Fin)(9,9) }
    \MPMPlaceDuree{COM>MAT,1}
    \MPMPlaceDuree{COM>LOG,1}\MPMPlaceDuree{COM>ECR,1}
    \MPMPlaceDuree{MAT>CABL,1}\MPMPlaceDuree{MAT>ASS,1}
    \MPMPlaceDuree{LOG>INST,3}<pos=0.85>
    \MPMPlaceDuree[Coude]{ECR>POST,6}<near start>
    \MPMPlaceDuree[Coude]{CABL>CONF,4}<near end>
\end{GrapheMPM}
```



Dans le cas où plusieurs arcs ont les mêmes caractéristiques, on peut utiliser la commande de *placement multiple*, \MPMPlaceDurees, pour laquelle les ⟨clés⟩ et l'argument optionnel entre <...> seront passés à **tous** les arcs.

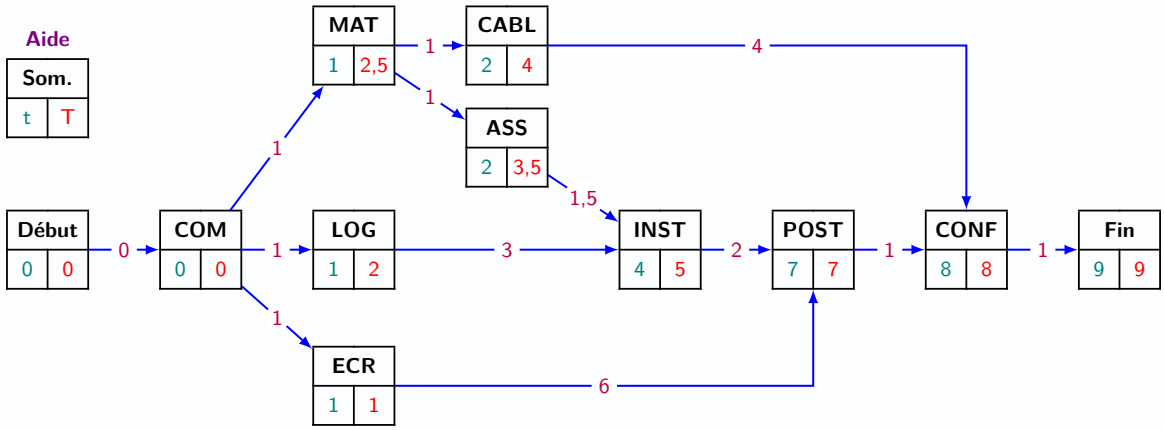
Dans ce cas, les données sont à spécifier sous forme d'une liste, avec le séparateur /.

Cela permet de *condenser* le code, dans le cas où de multiples arcs ont les mêmes caractéristiques.

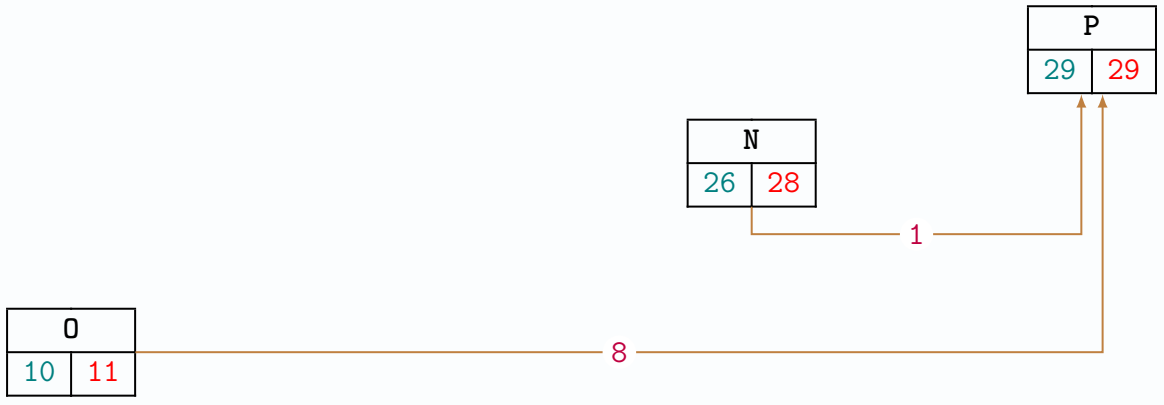
```
\begin{GrapheMPM}[clés]<options tikz>
    %DÉCLARATION DES TÂCHES
    \MPMPlaceDurees%
        [clés globales]%
        {TâcheA>TâcheB,durée / TâcheC>TâcheD,durée / ... }%
        <options tikz globales>
\end{GrapheMPM}
```

3.5 Exemples

```
\begin{GrapheMPM}[LargeurCases=0.5cm]<scale=0.9,transform shape>
    %NOTICE
    \MPMPlaceNotice(1,6.5)
    %TÂCHES
    \MPMPlaceTaches{ (1,4)(Début)(0,0) / (3.25,4)(COM)(0,0) / (5.5,4)(LOG)(1,2) /
    (5.5,2)(ECR)(1,1) / (5.5,7)(MAT)(1,2{,}5) / (7.75,7)(CABL)(2,4) /
    (7.75,5.5)(ASS)(2,3{,}5) / (10,4)(INST)(4,5) / (12.25,4)(POST)(7,7) /
    (14.5,4)(CONF)(8,8) / (16.75,4)(Fin)(9,9) }
    %DURÉES (ARCS DIRECTS)
    \MPMPlaceDurees{Début>COM,0 / COM>MAT,1 / COM>LOG,1 / COM>ECR,1 / MAT>CABL,1 /
    MAT>ASS,1 / LOG>INST,3 / ASS>INST,1{,}5 / INST>POST,2 / POST>CONF,1 /
    CONF>Fin,1}
    %DURÉES (ARCS COUDÉS)
    \MPMPlaceDurees[Coude]{ECR>POST,6 / CABL>CONF,4}<near start>
\end{GrapheMPM}
```




```
%ILLUSTRATION DES CLÉS [Decal...]
\begin{GrapheMPM}[CouleurFleches=brown,CouleurDurees=purple,Police=\large\ttfamily]
    %SOMMETS (EXTRAIT)
    \MPMPlaceTaches{ (6.75,2)(O)(10,11) / (15.75,4.5)(N)(26,28) /
    (20.25,6)(P)(29,29) }
    %ARCS (EXTRAIT)
    \MPMPlaceDuree[Coude,DecalHorizFin=4pt]{O>P.south,8}<near start>
    \MPMPlaceDuree[Coude,SensCoude=VHV,DecalHorizFin=-4pt]{N.south>P.south,1}<near
    start>
\end{GrapheMPM}
```



4 Tableau de Karnaugh à trois variables

4.1 Commande et fonctionnement global

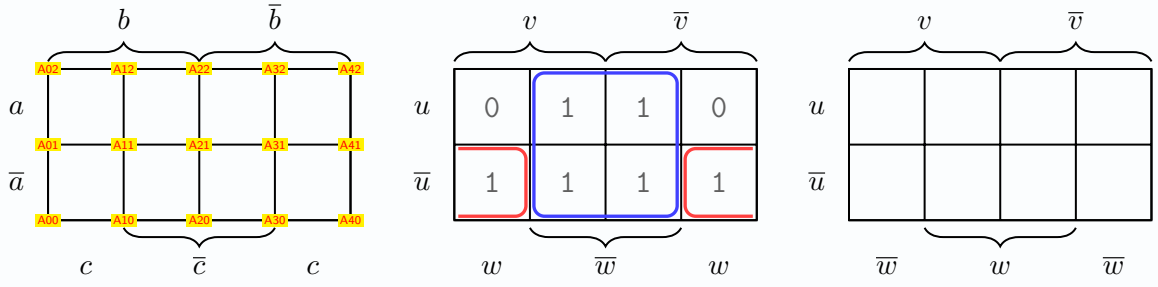
L'environnement dédié à la création du tableau de Karnaugh est `TableKarnaugh`. C'est en fait un environnement `tikzpicture` personnalisé.

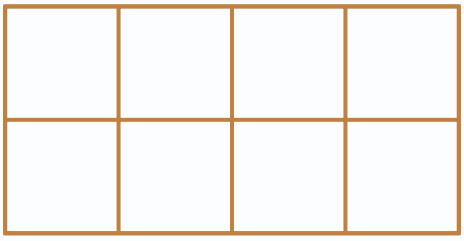
Les commandes à utiliser dans l'environnement sont :

- `\KarnaughCasesResult` pour saisir les cases par codage binaire ;
- `\KarnaughCasesAuto` pour utiliser l'expression booléenne ;
- `\KarnaughBlocRegroup` ;.

```
\begin{TableKarnaugh}[clés]<options tikz>
    \KarnaughCasesResult(*){contenu binaire des cases} %case par case
    \KarnaughCasesAuto(*){expression booléenne} %par une formule
    \KarnaughBlocRegroup[clés]{coinA}{coinB}
\end{TableKarnaugh}
```

```
\begin{TableKarnaugh}[Aide]
\end{TableKarnaugh}
\hspace{0.25cm}
\begin{TableKarnaugh}[Variables=u/v/w]
    \KarnaughCasesResult{0,1,1,0,1,1,1,1}
    \KarnaughBlocRegroup[Type=Centre,Couleur=blue!75,Decalage=-1.5pt]{10}{32}
    \KarnaughBlocRegroup[Type=Gauche,Couleur=red!75,Decalage=-1.5pt]{00}{11}
    \KarnaughBlocRegroup[Type=Droite,Couleur=red!75,Decalage=-1.5pt]{40}{31}
\end{TableKarnaugh}
\hspace{0.25cm}
\begin{TableKarnaugh}[Variables=u/v/w,Swap]
\end{TableKarnaugh}

\begin{center}
    \begin{TableKarnaugh}[Legende=false,Unite=1.5cm,Epaisseur=1.5pt,Couleur=brown]
    \end{TableKarnaugh}
\end{center}
```





Le tableau créé également des nœuds, qui seront utilisés pour effectuer des *regroupements* de cases, afin de simplifier une expression booléenne.

4.2 Arguments et clés pour l'environnement

```
\begin{TableKarnaugh}[clés]<options tikz>
    %commandes
\end{TableKarnaugh}
```

En ce qui concerne la création de l'environnement, les ⟨clés⟩ sont :

- ⟨Couleur⟩ := couleur du tableau ; défaut : ⟨black⟩
- ⟨Unite⟩ := unité de base de la figure ; défaut : ⟨1cm⟩
- ⟨Variables⟩ := nom des variables, sous la forme ⟨Gauche/Haut/Bas⟩ ; défaut : ⟨a/b/c⟩
- ⟨Swap⟩ := booléen pour échanger les variables du *bas* ; défaut : ⟨false⟩
- ⟨Aide⟩ := booléen pour afficher une aide sur les noms des nœuds ; défaut : ⟨false⟩
- ⟨Epaisseur⟩ := épaisseur des tracés ; défaut : ⟨0.75pt⟩
- ⟨CouleurCases⟩ := couleur de remplissage des cases ; défaut : ⟨lightgray⟩
- ⟨CouleurLegende⟩ := couleur de la légende, via ⟨Couleur⟩ ou ⟨CouleurA/CouleurB/CouleurC⟩ ; défaut : ⟨black⟩
- ⟨StyleAlternatif⟩ := booléen pour changer de style ; défaut : ⟨false⟩
- ⟨AideAlt⟩ := booléen pour (dés)activer le label *binaire* des cases ; défaut : ⟨true⟩
- ⟨PoliceTT⟩ := booléen pour forcer les labels en police télétype ; défaut : ⟨false⟩
- ⟨PosVarLaterale⟩ := position de la variable *latérale*. défaut : ⟨Gauche⟩

Le deuxième argument, optionnel et entre `<...>` propose des options, en langage `tikz` à passer à l'environnement.

4.3 Arguments et clés pour la commande de remplissage

```
\begin{TableKarnaugh}[clés]<options tikz>
    \KarnaughCasesResult(*){contenu binaire des cases}
    %ou
    \KarnaughCasesAuto(*){expression booléenne}
\end{TableKarnaugh}
```

En ce qui concerne le remplissage des cases avec la commande `\KarnaughCasesResult` :

- la version *étoilée* permet de *griser* les cases au lieu de les remplir de `0/1` ;
- l'argument obligatoire, et entre `{....}` est la liste des cases, de gauche à droite en partant de la ligne du haut ;
- la couleur de cases est gérée par la clé idoine de l'environnement.

À noter que la liste peut être donnée sous forme ⟨{1,0,1,0,0,0,0,0}⟩ ou ⟨{1010000}⟩.

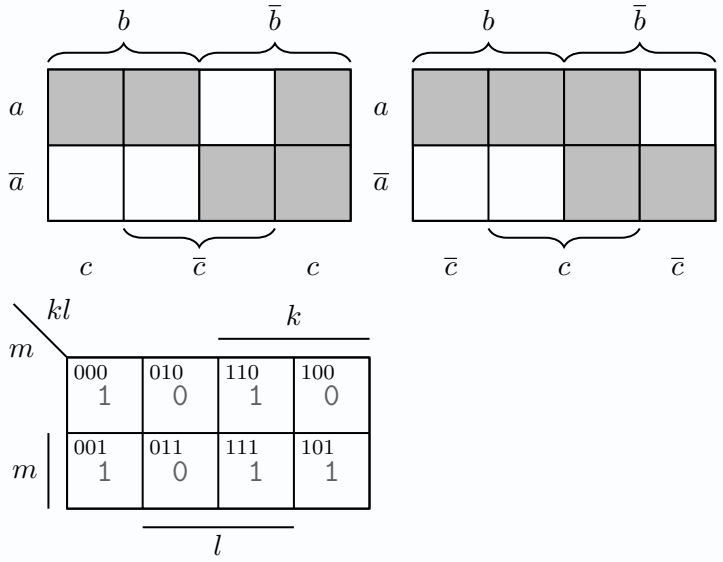
Pour la commande `\KarnaughCasesAuto`, c'est l'expression booléenne à trois variables sous forme de *mintermes* simples, donnés :

— entre parenthèses ;

— les *mintermes* avec les variables dans le même ordre que celui donné en paramètre ;

— avec `*` pour la barre.

```
\begin{TableKarnaugh}
    \KarnaughCasesAuto*{(ab)+(a*b*)+(b*c)}
\end{TableKarnaugh}
\begin{TableKarnaugh}[Swap]
    \KarnaughCasesAuto*{(ab)+(a*b*)+(b*c)}
\end{TableKarnaugh}

\begin{TableKarnaugh}[Variables=k/l/m,StyleAlternatif]
    \KarnaughCasesAuto{(kl)+(k*l*)+(l*m)}
\end{TableKarnaugh}
```



4.4 Arguments et clés pour la commande de regroupement des blocs

```
\begin{TableKarnaugh}[clés]<options tikz>
    %remplissage des cases
    \KarnaughBlocRegroup[clés]{coinA}{coinB}
\end{TableKarnaugh}
```

En ce qui concerne le regroupement des cases par blocs, les ⟨clés⟩ disponibles sont :

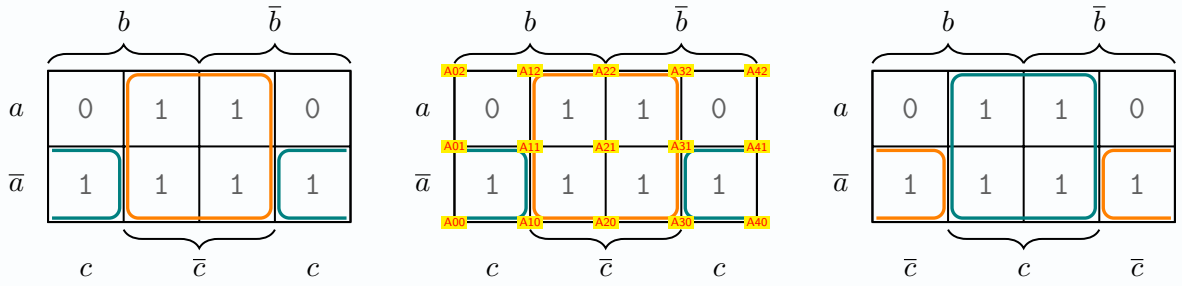
- ⟨Couleur⟩ := couleur du *trait* ; défaut : ⟨red⟩
- ⟨type⟩ := type de regroupement parmi ⟨Centre/Gauche/Droite⟩ ; défaut : ⟨Centre⟩
- ⟨Decalage⟩ := décalage du trait par rapports aux cases. défaut : ⟨2pt⟩

Les deux arguments obligatoires, et entre `{...}`, correspondent aux *coins diagonaux* :

- sans contrainte pour un rectangle ⟨Type=Centre⟩ ;
- du type `{BG}{HD}` pour un rectangle ⟨Type=Gauche⟩ ;
- du type `{BD}{HG}` pour un rectangle ⟨Type=Droite⟩.

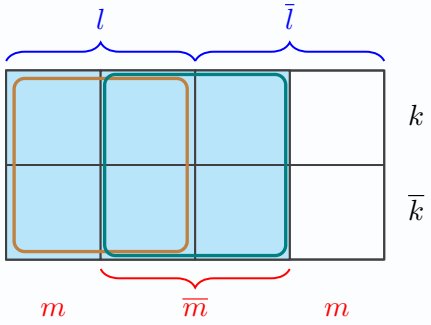
4.5 Exemples

```
\begin{TableKarnaugh}
    \KarnaughCasesResult{0,1,1,0,1,1,1,1}
    \KarnaughBlocRegroup[Type=Centre,Couleur=orange,Decalage=-1.5pt]{10}{32}
    \KarnaughBlocRegroup[Type=Gauche,Couleur=teal,Decalage=-1.5pt]{00}{11}
    \KarnaughBlocRegroup[Type=Droite,Couleur=teal,Decalage=-1.5pt]{40}{31}
\end{TableKarnaugh}
\hspace{5mm}
\begin{TableKarnaugh}[Aide]
    \KarnaughCasesResult{01101111}
    \KarnaughBlocRegroup[Type=Centre,Couleur=orange,Decalage=-1.5pt]{10}{32}
    \KarnaughBlocRegroup[Type=Gauche,Couleur=teal,Decalage=-1.5pt]{00}{11}
    \KarnaughBlocRegroup[Type=Droite,Couleur=teal,Decalage=-1.5pt]{40}{31}
\end{TableKarnaugh}
\hspace{5mm}
\begin{TableKarnaugh}[Swap]
    \KarnaughCasesAuto{(ac)+(a*b)+(a*b*c*)+(b*c)}
    \KarnaughBlocRegroup[Type=Centre,Couleur=teal,Decalage=-1.5pt]{10}{32}
    \KarnaughBlocRegroup[Type=Gauche,Couleur=orange,Decalage=-1.5pt]{00}{11}
    \KarnaughBlocRegroup[Type=Droite,Couleur=orange,Decalage=-1.5pt]{40}{31}
\end{TableKarnaugh}
```

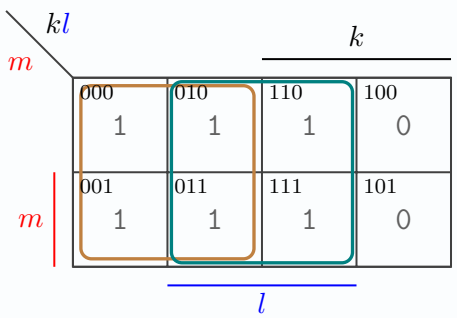


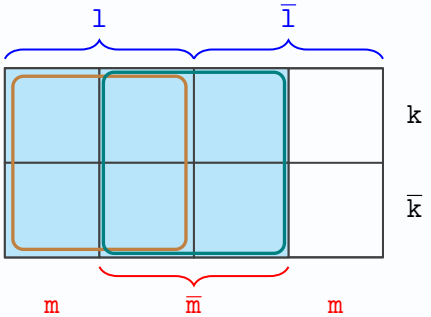
```
On obtient le tableau de Karnaugh suivant :
\begin{TableKarnaugh}
        [Variables=k/l/m,Unite=1.25cm,CouleurCases=cyan!25,Couleur=darkgray,
        PosVarLaterale=Droite,CouleurLegende=black/blue/red]
        <baseline=(current bounding box.center)>
    \KarnaughCasesResult*{1,1,1,0,1,1,1,0}
    \KarnaughBlocRegroup[Type=Centre,Couleur=brown,Decalage=-3pt]{00}{22}
    \KarnaughBlocRegroup[Type=Centre,Couleur=teal,Decalage=-1.5pt]{10}{32}
\end{TableKarnaugh}
```

On obtient le tableau de Karnaugh suivant :




```
\begin{TableKarnaugh}
    [Variables=k/l/m,Unite=1.25cm,Couleur=darkgray,
    PosVarLaterale=Droite,CouleurLegende=black/blue/red,
    StyleAlternatif]
    \KarnaughCasesResult{11101110}
    \KarnaughBlocRegroup[Type=Centre,Couleur=brown,Decalage=-3pt]{00}{22}
    \KarnaughBlocRegroup[Type=Centre,Couleur=teal,Decalage=-1.5pt]{10}{32}
\end{TableKarnaugh}
\hspace{5mm}
\begin{TableKarnaugh}
    [Variables=k/l/m,Unite=1.25cm,CouleurCases=cyan!25,Couleur=darkgray,
    PosVarLaterale=Droite,CouleurLegende=black/blue/red,
    PoliceTT]
    \KarnaughCasesResult*{1,1,1,0,1,1,1,0}
    \KarnaughBlocRegroup[Type=Centre,Couleur=brown,Decalage=-3pt]{00}{22}
    \KarnaughBlocRegroup[Type=Centre,Couleur=teal,Decalage=-1.5pt]{10}{32}
\end{TableKarnaugh}
```





5 Simplification d'une expression booléenne par table de Karnaugh

5.1 Commande et fonctionnement global

L'idée est de proposer une commande pour simplifier une expression booléenne (ou son contraire) à trois variables connaissant sa table de vérité :

— en utilisant une manière *binaire* de déclarer la table de vérité ;

— en adaptant le résultat à la configuration de la table de Karnaugh.

Je remercie mes étudiants de BTS SIO2 (promo 2023/2024 : Léo, Ryad, Mathieu, Adrien, Clément) pour m'avoir aidé à simplifier les 256 (!) tables de Karnaugh possibles, en se répartissant le travail !

Malgré des relectures, il se peut qu'il subsiste malheureusement des coquilles dans les expressions simplifiées, et dans le cas où il existe plusieurs possibilités, la commande n'affichera que l'une d'entre elles !

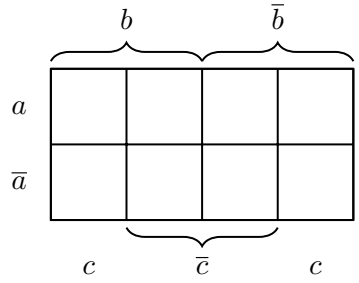
```
\SimplificationKarnaugh[clés]{code binaire de la table}
```

La déclaration *binaire* de la table suit les mêmes règles que pour la commande de création de la table :

— la liste des cases soit être saisie, de gauche à droite en partant de la ligne du haut ;

— elle doit être donnée en binaire, sans séparateur, comme ⟨{1010111}⟩ par exemple.

Par **défaut**, le remplissage des cases est relatif à la configuration suivante, mais il est possible de spécifier un autre type de table :



Un espace *spécial* mathématique a été défini, pour notamment gérer l'espacement horizontal entre des variables booléennes, pour éviter que les barres soient *collées*.
Il s'agit de `\§`, qui vaut `1.5mu`, et qui est utilisé pour écrire des expressions booléennes.

5.2 Arguments et clés

```
\SimplificationKarnaugh[clés]{code binaire de la table}
```

Les clés disponibles sont :

— ⟨Couleurs⟩ := couleurs pour chacune des trois variables booléennes ; défaut : ⟨black/black/black⟩

— ⟨Variables⟩ := variables utilisées ; défaut : ⟨a/b/c⟩

— ⟨Swap⟩ := booléen pour échanger les variables du *bas* ; défaut : ⟨false⟩

— ⟨Contraire⟩ := booléen pour travailler sur le contraire de l'expression booléenne ; défaut : ⟨false⟩

— ⟨Espace⟩ := booléen pour rajouter un petit espace (1.5 mu) dans les produits ; défaut : ⟨true⟩

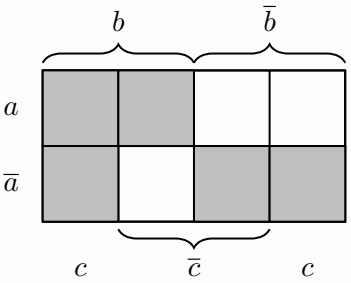
— ⟨StyleAlternatif⟩ := booléen pour changer de style. défaut : ⟨false⟩

L'argument obligatoire est quant à lui la déclaration *binaire* de la table, sous la forme ⟨{xxxxxxxx}⟩.

À noter que le résultat est inséré dans un groupe `ensuremath`.

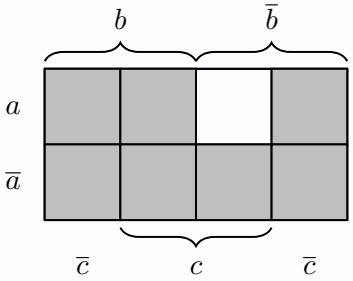
5.3 Exemples

```
La table de Karnaugh :
\begin{TableKarnaugh}<baseline=(current bounding box.center)>
    \KarnaughCasesResult*{11001011}
\end{TableKarnaugh}~
correspond à $E=\SimplificationKarnaugh{11001011}$.
```

La table de Karnaugh : 

correspond à $E = ab + \overline{a}\overline{b} + bc$.

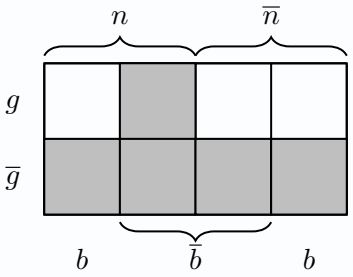
```
La table de Karnaugh :
\begin{TableKarnaugh}[Swap]<baseline=(current bounding box.center)>
    \KarnaughCasesResult*{11011111}
\end{TableKarnaugh}~
correspond à $E=\SimplificationKarnaugh[Swap]{11011111}$.

Et $\overline{E}=\SimplificationKarnaugh[Swap,Contraire]{11011111}$.
```

La table de Karnaugh : 

correspond à $E = \overline{a} + b + \overline{c}$.

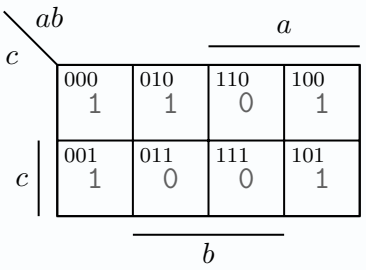
Et $\overline{E} = a\overline{b}c$.

```
La table de Karnaugh :
\begin{TableKarnaugh}[Variables={g/n/b}]<baseline=(current bounding box.center)>
    \KarnaughCasesResult*{01001111}
\end{TableKarnaugh}~
correspond à $E=\SimplificationKarnaugh[Variables={g/n/b}]{01001111}$
```

La table de Karnaugh : 

correspond à $E = \overline{g} + n\overline{b}$

```
La table de Karnaugh :
\begin{TableKarnaugh}[StyleAlternatif]<baseline=(current bounding box.center)>
    \KarnaughCasesResult{11011001}
\end{TableKarnaugh}~
correspond à \SimplificationKarnaugh%
    [StyleAlternatif,Couleurs={purple/blue/orange}]%
    {11011001}
```

La table de Karnaugh : 

correspond à $\overline{b} + \overline{c}\overline{a}$

6 Écriture et simplification d'une expression booléenne

6.1 Commande et fonctionnement global

L'idée est de proposer une commande pour afficher et simplifier une expression booléenne à trois variables connaissant son expression sous forme de *mintermes* simples, donnés :

— entre parenthèses, et avec les variables dans le même ordre que celui donné en paramètre ;

— avec `*` pour la barre.

Par exemple :

— $E = ab + \overline{a}\overline{b}$ sera saisie par `(ab)+(a*b*)` ;

— $E = \overline{a}\overline{c} + \overline{a}\overline{b} + bc$ sera saisie par `(a*c*)+(a*b*)+(bc)`.

6.2 Arguments et clés

```
\SimplificationBooleenne[clés]{expression formatée}
```

Les clés disponibles sont :

— ⟨Enonce⟩ := booléen pour afficher l'expression booléenne brute ; défaut : ⟨true⟩

— ⟨Variables⟩ := variables utilisées ; défaut : ⟨a/b/c⟩

— ⟨Contraire⟩ := booléen pour travailler sur le contraire ; défaut : ⟨false⟩

— ⟨Espace⟩ := booléen pour rajouter un petit espace (1.5 mu) dans les produits. défaut : ⟨true⟩

L'argument obligatoire est quant à lui la déclaration *brute* de l'expression booléenne, donnée avec les règles précédentes. À noter que le résultat est inséré dans un groupe `ensuremath`.

6.3 Exemples

```
$E = \SimplificationBooleenne{(a)+(a*)}$\\
$E = a + \overline{a} = \SimplificationBooleenne[Enonce=false]{(a)+(a*)}$
```

$E = a + \overline{a} = 1$
$E = a + \overline{a} = 1$

```
On considère l'expression booléenne $E= g + g\§b + g\§\overline{b}\§n$ :\\
$E = \SimplificationBooleenne[Variables=g/b/n]{(g)+(gb)+(gb*n)}$\\
$E = \SimplificationBooleenne[Variables=g/b/n,Espace=false]{(g)+(gb)+(gb*n)}$
```

On considère l'expression booléenne $E = g + gb + g\overline{b}n$:
$E = g + gb + g\overline{b}n = g$
$E = g + gb + g\overline{b}n = g$

```
$\overline{E} =
    \SimplificationBooleenne[Contraire,Variables=g/b/n]{(g)+(gb)+(gb*n)}$
```

$\overline{E} = \overline{g + gb + g\overline{b}n} = \overline{g}$

```
On a $F = \SimplificationBooleenne{(a)+(abc*)+(a*b*c)+(abc)+(a*bc)}$.
```

On a $F = a + ab\overline{c} + \overline{a}\overline{b}c + abc + \overline{a}bc = a + c$.

7 Graphes *simples*

7.1 Commande et fonctionnement global

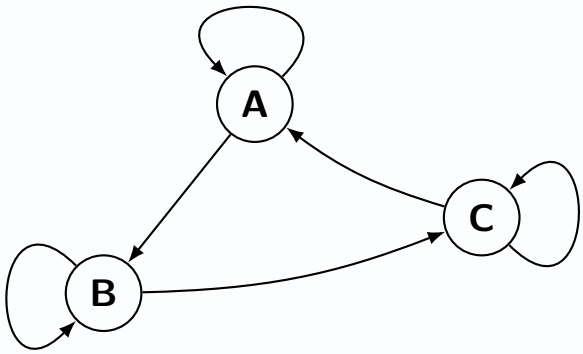
L'environnement dédié à la création d'un graphe *simple* est `GrapheTikz`.
C'est en fait un environnement `tikzpicture` personnalisé.

Les commandes à utiliser dans l'environnement sont :

- `\GrphPlaceSommets` ;
- `\GrphTraceAretes` ;.

```
\begin{GrapheTikz}[clés]<options tikz>
    \GrphPlaceSommets{liste coordonnées/sommet}
    \GrphTraceAretes(*)[type]<options tikz>{liste arêtes}
\end{GrapheTikz}
```

```
\begin{GrapheTikz}
    \GrphPlaceSommets{(2,2.5)/A (0,0)/B (5,1)/C}
    \GrphTraceAretes{A/B}
    \GrphTraceAretes[AngleGauche]{C/A}
    \GrphTraceAretes[AngleDroite]{B/C}
    \GrphTraceAretes[Boucle]{A/45 B/135 C/-45}
\end{GrapheTikz}
```



La majorité des paramètres sont personnalisables, mais le *thème* général est globalement *fixé*, dans le sens où ce sont les éléments *cosmétiques* qui pourront être modifiés.

Au contraire du package `tkz-graph` qui permet beaucoup plus de choses, les commandes de `ProfSio` se veulent beaucoup plus basiques, dans l'optique de travailler avec des graphes en adéquation avec le programme de BTS SIO.

L'utilisateur pourra également redéfinir les styles utilisées par les commandes de `ProfSio` pour refondre le paramétrage global de l'environnement.

```
\begin{GrapheTikz}[clés]<options tikz>
    \tikzset{GrphStyleSommet/.style = {...}}
    \tikzset{GrphStyleArc/.style = {...}}
    \tikzset{GrphStylepoids/.style = {...}}
\end{GrapheTikz}
```

La commande de tracé des arêtes nécessite de travailler avec des nœuds existants, donc tout nœud précédemment défini, que ce soit avec la commande de `ProfSio` ou tout autre commande pourra être utilisé !

7.2 Arguments et clés pour l'environnement

```
\begin{GrapheTikz}[clés]<options tikz>
    %commandes
\end{GrapheTikz}
```

En ce qui concerne la création de l'environnement, les ⟨clés⟩ sont :

- ⟨Police⟩ := police des sommets ; défaut : ⟨\bfseries\Large\sffamily⟩,
- ⟨Poids⟩ := police des éventuels poids ; défaut : ⟨\sffamily⟩,
- ⟨CouleurSommets⟩ := couleur(s) sous la forme ⟨Couleur⟩ ou ⟨CouleurBord/CouleurTexte⟩ des sommets ; défaut : ⟨black⟩
- ⟨CouleurFleches⟩ := couleur des arêtes (et des poids) ; défaut : ⟨black⟩,
- ⟨TypeSommets⟩ := type de forme des sommets ; défaut : ⟨circle⟩
- ⟨Epaisseur⟩ := épaisseur(s) sous la forme ⟨Epaisseur⟩ ou ⟨EpaisseurSommet/EpaisseurArête⟩ des traits ; défaut : ⟨thick⟩
- ⟨Unite⟩ := unité globale de la figure ; défaut : ⟨1cm⟩
- ⟨CouleurFT⟩ := couleur des arêtes de la fermeture transitive (accessible ensuite via ⟨FT⟩) ; défaut : ⟨black⟩
- ⟨Grille⟩ := pour afficher une grille d'aide (⟨{xmax,ymax}⟩), entre (0 ; 0) et (xmax ; ymax) ; défaut : ⟨vide⟩
- ⟨DimensionSommets⟩ := dimension(s) minimale(s) des formes des sommets, sous la forme ⟨Globale⟩ ou ⟨Largeur/Hauteur⟩ ; défaut : ⟨1cm⟩
- ⟨PositionFleches⟩ := position, parmi ⟨Milieu/Fin⟩ pour les flèches ; défaut : ⟨Fin⟩
- ⟨EchelleFleches⟩ := échelle de la flèche ; défaut : ⟨1⟩
- ⟨TypeFleche⟩ := type (en Ti*k*Z) des flèches. défaut : ⟨Latex⟩

Le deuxième argument, optionnel et entre `<...>` propose des options, en langage `tikz` à passer à l'environnement.

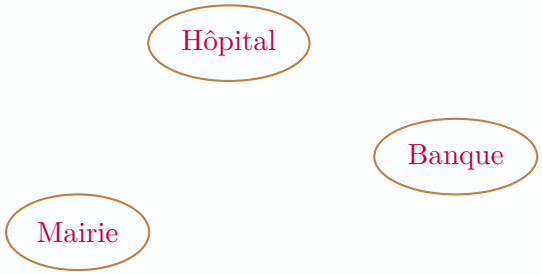
7.3 Arguments et clés pour la commande de création des sommets

```
\begin{GrapheTikz}[clés]<options tikz>
    \GrphPlaceSommets{liste coordonnées/sommet}
\end{GrapheTikz}
```

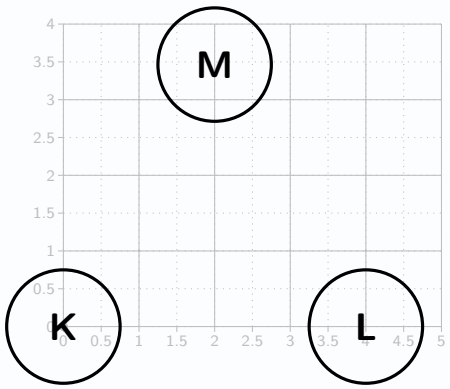
En ce qui concerne la création des sommets, la liste est à donner sous la forme `(xa,ya)/A (xb,yb)/B (xc,yc)/C ...`

Dans le cas de sommets avec des espaces, il faut les *protéger* par des `{...}`.

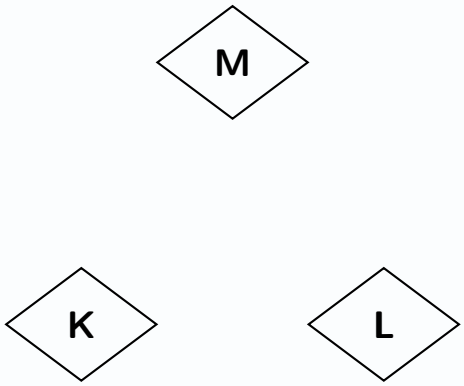
```
\begin{GrapheTikz}[CouleurSommets={brown/purple},TypeSommets=ellipse,Police={}]
    \GrphPlaceSommets{(2,2.5)/Hôpital (0,0)/Mairie (5,1)/Banque}
\end{GrapheTikz}
```




```
\begin{GrapheTikz}[Epaisseur={very thick},Grille={5,4},DimensionSommets=1.5cm]
    \GrphPlaceSommets{(0,0)/K (4,0)/L (60:4)/M}
\end{GrapheTikz}
```




```
\begin{GrapheTikz}[TypeSommets=diamond,DimensionSommets=2cm/1.5cm]
    \GrphPlaceSommets{(0,0)/K (4,0)/L (60:4)/M}
\end{GrapheTikz}
```



7.4 Arguments et clés pour la commande de tracé des arêtes

```
\begin{GrapheTikz}[clés]<options tikz>
    %commandes de placement des sommets
    \GrphTraceAretes(*)[type]<options tikz>{liste arêtes}
\end{GrapheTikz}
```

En ce qui concerne le tracés des arêtes, la commande permet de tracer des arêtes ayant le même style.

La version *étoilée* permet de pondérer l'arête (le poids est, par défaut, situé sur le milieu de l'arête).

Les ⟨type⟩ d'arête, disponible entre `[...]` et valant ⟨Droit⟩ par défaut, de la commande peut valoir :

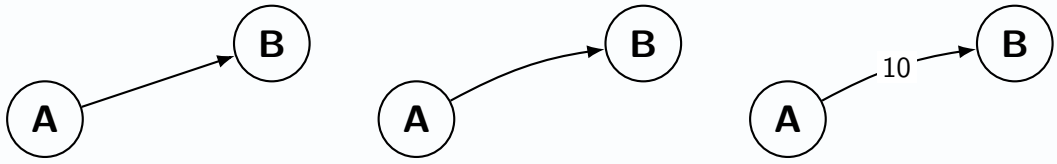
- ⟨Droit⟩ := permet de tracer des arêtes orientées *droites* ;
- ⟨AngleGauche⟩ ou ⟨AngleGauche=...⟩ := permet de tracer des arêtes orientées *courbées vers la gauche*, avec par défaut un angle de 10° ;
- ⟨AngleDroite⟩ ou ⟨AngleDroite=...⟩ := permet de tracer des arêtes orientées *courbées vers la droite*, avec par défaut un angle de 10° ;
- ⟨Boucle⟩ ou ⟨Boucle=...⟩ := permet de tracer une boucle avec un coefficient `looseness` de 6 par défaut.

Dans le cas d'arêtes *classiques*, la liste est à donner sous la forme `Deb/Fin Deb/Fin Deb/Fin ...` ou `Deb/Fin/Poids Deb/Fin/Poids Deb/Fin/Poids ...`

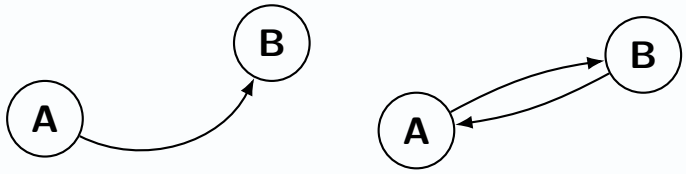
Dans le cas de boucles, la lise est à donner sous la forme `Som/anglesortie Som/anglesortie ...` ou `Som/anglesortie/Poids Som/anglesortie/Poids ...` en sachant que (par défaut) l'angle d'entrée est fixé 90° après dans le sens trigonométrique.

Pour marquer une fermeture transitive, on peut utiliser le style `FT` dans les *options tikz* de la commande.

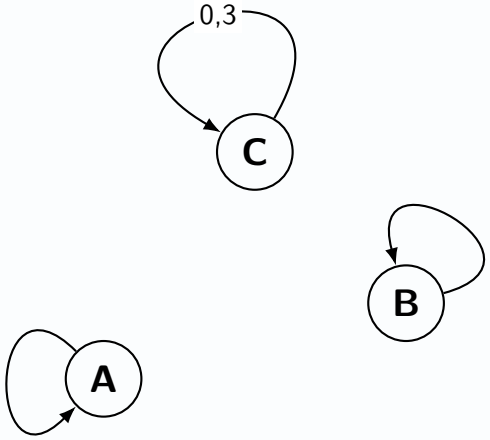
```
\begin{GrapheTikz}
    \GrphPlaceSommets{(0,0)/A (3,1)/B}
    \GrphTraceAretes{A/B}
\end{GrapheTikz}
\hspace{5mm}
\begin{GrapheTikz}
    \GrphPlaceSommets{(0,0)/A (3,1)/B}
    \GrphTraceAretes[AngleGauche]{A/B}
\end{GrapheTikz}
\hspace{5mm}
\begin{GrapheTikz}
    \GrphPlaceSommets{(0,0)/A (3,1)/B}
    \GrphTraceAretes*[AngleGauche]{A/B/10}
\end{GrapheTikz}
```



```
\begin{GrapheTikz}
    \GrphPlaceSommets{(0,0)/A (3,1)/B}
    \GrphTraceAretes[AngleDroite=45]{A/B}
\end{GrapheTikz}
\hspace{5mm}
\begin{GrapheTikz}
    \GrphPlaceSommets{(0,0)/A (3,1)/B}
    \GrphTraceAretes[AngleGauche]{A/B B/A}
\end{GrapheTikz}
```

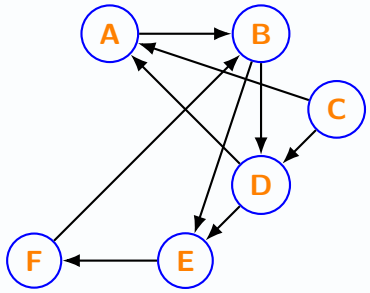



```
\begin{GrapheTikz}
    \GrphPlaceSommets{(0,0)/A (4,1)/B (2,3)/C}
    \GrphTraceAretes[Boucle]{A/135 B/15}
    \GrphTraceAretes*[Boucle=10]{C/60/{0{,}3}}
\end{GrapheTikz}
```

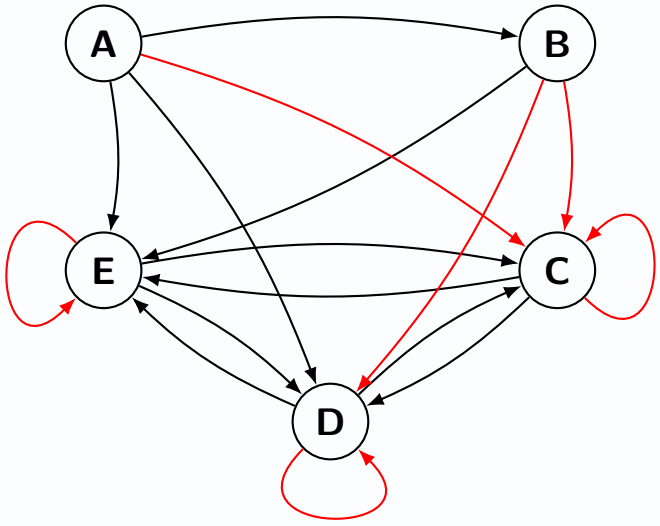


7.5 Exemples

```
\begin{GrapheTikz}
    [DimensionSommets=14pt,Police=\bfseries\sffamily,CouleurSommets={blue/orange}]
    %SOMMETS
    \GrphPlaceSommets{(1,4)/A (3,4)/B (4,3)/C (3,2)/D (2,1)/E (0,1)/F}
    %ARÊTES
    \GrphTraceAretes{A/B B/D B/E C/A C/D D/A D/E E/F F/B}
\end{GrapheTikz}
```




```
\begin{GrapheTikz}
    %SOMMETS
    \GrphPlaceSommets{(1,6)/A (7,6)/B (7,3)/C (4,1)/D (1,3)/E}
    %ARÊTES
    \GrphTraceAretes[AngleGauche]{A/B A/D A/E B/E C/E E/C D/C C/D E/D D/E}
    %FT
    \GrphTraceAretes[AngleGauche]<FT>{A/C B/C B/D}
    \GrphTraceAretes[Boucle]<FT>{C/-45 E/135 D/-135}
\end{GrapheTikz}
```



8 Matrice d'adjacence, fermeture transitive

8.1 Commandes et fonctionnement global

L'idée est de proposer des commandes pour travailler avec l'aspect matriciel des graphes :

- afficher la matrice d'adjacence, avec bordure éventuelle ;
- calculer la puissance n-ième d'une matrice d'adjacence, avec bordure éventuelle ;
- déterminer le nombre de chemins de longueur donnée dans un graphe ;
- déterminer la matrice de la fermeture transitive, avec bordure éventuelle.

Quelques éléments de personnalisations sont disponibles.

La commande est accessible **uniquement** en cas d'une compilation en LuaLaTeX !

Une **double compilation** peut être nécessaire pour le placement correct des filets !

Les calculs matriciels sont effectués par le package `lualinalg` (https://ctan.org/pkg/lualinalg), mais les sorties sont parfois *modifiées* pour une présentation adaptée avec des matrices bordées.

Les matrices sont à déclarer sous une forme particulière, sous la forme `{{ligne1},{ligne2},{...}}` avec `ligne1={e1,e2,...}`.

8.2 Matrice d'adjacence, puissance

La commande dédiée à la l'affichage d'une matrice d'adjacence `\MatriceAdjacence`.

La commande dédiée à la l'affichage d'une puissance d'une matrice d'adjacence `\PuissanceMatrice`.

```
%affichage de la matrice d'adjdacence
\MatriceAdjacence[clés]{matrice}

%affichage de la puissance
\PuissanceMatrice[clés]{matrice}{exposant}
```

Les ⟨clés⟩ (communes) disponibles sont :

- ⟨Bordure⟩ := booléen pour border la matrice ; défaut : ⟨false⟩
- ⟨Sommets⟩ := sommets du graphe (uni-caractères) ; défaut : ⟨ABCDE...⟩
- ⟨Num⟩ := booléen pour formater avec `siunitx` (si chargé !) ; défaut : ⟨false⟩
- ⟨PoliceBordure⟩ := spécifier la police (taille) de la bordure. défaut : ⟨\footnotesize⟩

Le deuxième argument, optionnel et entre `<...>` permet de spécifier la matrice (comme indiqué précédemment) avec laquelle on travaille.

```
\def\MatriceAdj{{{0,1,1,1},{0,0,1,0},{0,1,0,1},{1,0,0,1}}}
On considère la matrice d'adjacence $M = \MatriceAdjacence{\MatriceAdj}$.\par
On considère la matrice d'adjacence $M =
    \MatriceAdjacence[Bordure]{\MatriceAdj}$.\par
On considère la matrice d'adjacence $M =
    \MatriceAdjacence[Bordure,Sommets=JKLM]{\MatriceAdj}$.\par
```

On considère la matrice d'adjacence $M = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$.

On considère la matrice d'adjacence $M = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \end{array}$.

On considère la matrice d'adjacence $M = \begin{array}{c} \\ J \\ K \\ L \\ M \end{array} \begin{array}{c} \begin{array}{cccc} J & K & L & M \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \end{array}$.

8.3 Chemins de longueur donnée

La commande dédiée au calcul du nombre de chemins de longueur donnée dans un graphe est `\NbCheminsLongueur`.

Pour le moment la commande ne calcule *que* le nombre de chemins de longueur donnée entre deux sommets.

```
%calcul du nombre de chemins
\NbCheminsLongueur[Longueur=...,De=...,Vers=...,clés]{matrice}
```

Les ⟨clés⟩ disponibles sont :

- ⟨Sommets⟩ := sommets du graphe (uni-caractères) ; défaut : ⟨ABCDE...⟩
- ⟨Num⟩ := booléen pour formater avec `siunitx` (si chargé !). défaut : ⟨false⟩

Les clés ⟨De⟩, ⟨Vers⟩ et ⟨Longueur⟩ sont *nécessaires* et doivent être cohérentes avec la liste des sommets.

```
\def\MatriceAdjB{{{1,1,0,0,0},{1,0,1,0,0},{1,0,0,1,0},{1,1,0,0,0},{0,0,0,1,0}}}
On donne $M = \MatriceAdjacence[Bordure,Sommets=JKLMN]{\MatriceAdjB}$.\par
On a $M^5 = \PuissanceMatrice[Bordure,Sommets=JKLMN]{\MatriceAdjB}{5}$, il existe
    \NbCheminsLongueur[Longueur=5,De=J,Vers=L,Sommets=JKLMN]{\MatriceAdjB} chemins
    de longueur 5 allant de $J$ vers $L$.
```

On donne $M = \begin{array}{c} \\ J \\ K \\ L \\ M \\ N \end{array} \begin{array}{c} \begin{array}{ccccc} J & K & L & M & N \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$.

On a $M^5 = \begin{array}{c} \\ J \\ K \\ L \\ M \\ N \end{array} \begin{array}{c} \begin{array}{ccccc} J & K & L & M & N \end{array} \\ \begin{pmatrix} 16 & 9 & 5 & 2 & 0 \\ 16 & 9 & 4 & 3 & 0 \\ 16 & 10 & 4 & 2 & 0 \\ 16 & 9 & 5 & 2 & 0 \\ 8 & 5 & 2 & 1 & 0 \end{pmatrix} \end{array}$, il existe 5 chemins de longueur 5 allant de J vers L.

8.4 Matrice de fermeture transitive

La commande dédiée pour travailler sur la matrice de fermeture transitive d'un graphe est \FermetureTransitive.

Quelques options de *présentation* sont disponibles pour cette commande.

```
%matrice de fermeture transitive
\FermetureTransitive[clés]{matrice}
```

Les ⟨clés⟩ disponibles sont :

- ⟨Sommets⟩ := sommets du graphe (uni-caractères) ; défaut : ⟨ABCDE...⟩
- ⟨Formule⟩ := booléen pour afficher la formule ; défaut : ⟨false⟩
- ⟨Brut⟩ := booléen pour afficher le résultat brut de la calculatrice ; défaut : ⟨false⟩
- ⟨Enonce⟩ := booléen pour afficher le nom au début ; défaut : ⟨false⟩
- ⟨NomMatrice⟩ := nom de la matrice d'adjacence ; défaut : ⟨M⟩
- ⟨Bordure⟩ := booléen pour border la matrice ; défaut : ⟨false⟩
- ⟨Sommets⟩ := sommets du graphe (uni-caractères) ; défaut : ⟨ABCDE...⟩
- ⟨PoliceBordure⟩ := spécifier la police (taille) de la bordure. défaut : ⟨\footnotesize⟩

À noter qu'une clé booléenne, ⟨Complet⟩, existe, et qui active ⟨Formule⟩ et ⟨Enonce⟩.

```
\def\MatriceAdjC{{{1,1,1,1},{0,0,0,1},{0,1,0,0},{0,0,1,0}}}

\FermetureTransitive{\MatriceAdjC} et \FermetureTransitive[Brut]{\MatriceAdjC}

On a \FermetureTransitive[Brut,Formule]{\MatriceAdjC}.

Donc \FermetureTransitive[Complet,Bordure,Sommets=XYZF]{\MatriceAdjC}.
```

$$\begin{pmatrix}1&1&1&1\\0&1&1&1\\0&1&1&1\\0&1&1&1\end{pmatrix} \text{ et } \begin{pmatrix}4&10&10&10\\0&1&1&2\\0&2&1&1\\0&1&2&1\end{pmatrix}$$

On a $M+M^2+M^3+M^4=\begin{pmatrix}4&10&10&10\\0&1&1&2\\0&2&1&1\\0&1&2&1\end{pmatrix}$.

Donc $\widehat{M}=M\oplus M^{[2]}\oplus M^{[3]}\oplus M^{[4]}=\begin{array}{c}\\X\\Y\\Z\\F\end{array}\begin{array}{c}\begin{array}{cccc}X&Y&Z&F\end{array}\\\begin{pmatrix}1&1&1&1\\0&1&1&1\\0&1&1&1\\0&1&1&1\end{pmatrix}\end{array}$.

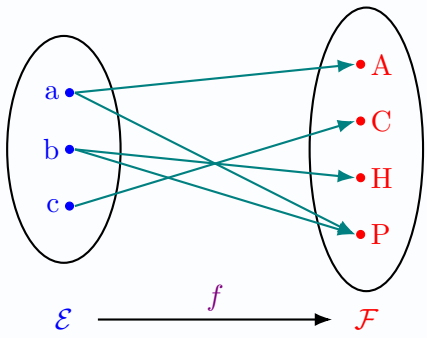
9 Diagramme sagittal d'une application

9.1 Commande et fonctionnement global

La commande dédiée à la création d'un diagramme sagittal pour une application est `\DiagrammeSagittal`.
Le diagramme créé est réalisé avec un environnement `tikzpicture`.

```
%commande autonome
\DiagrammeSagittal[clés]<options tikz>{liaisons}

%commande à insérer dans un environnement tikzpicture
\begin{tikzpicture}
    \DiagrammeSagittal*[clés]{liaisons}
\end{tikzpicture}
```

```
\DiagrammeSagittal[E={a,b,c},F={A,C,H,P}]{a/A,a/P,b/H,b/P,c/C}
```



La majorité des paramètres sont personnalisables, mais le *thème* général est globalement *fixé*, dans le sens où ce sont les éléments *cosmétiques* qui pourront être modifiés.

La commande de création de `ProfSio` est volontairement pour des applications basiques, dans l'optique de travailler avec exemples en adéquation avec le programme de BTS SIO.

9.2 Arguments et clés

```
\DiagrammeSagittal[clés]<options tikz>{liaisons}

\begin{tikzpicture}
    \DiagrammeSagittal*[clés]{liaisons}
\end{tikzpicture}
```

Le code se charge, grâce aux ⟨clés⟩, de positionner et d'aligner les éléments des ensembles et les flèches.
De ce fait, les *écarts* entre les éléments d'un ensemble sont fixées globalement, tout comme le style général des flèches.

La version *étoilé* permet de ne pas créer l'environnement `tikzpicture`, pour d'éventuels rajouts ultérieurs :

- les éléments de l'ensemble de départ sont des nœuds nommés `(E...)` ;
- les éléments de l'ensemble d'arrivée sont des nœuds nommés `(F...)`.

Les ⟨clés⟩ disponibles sont :

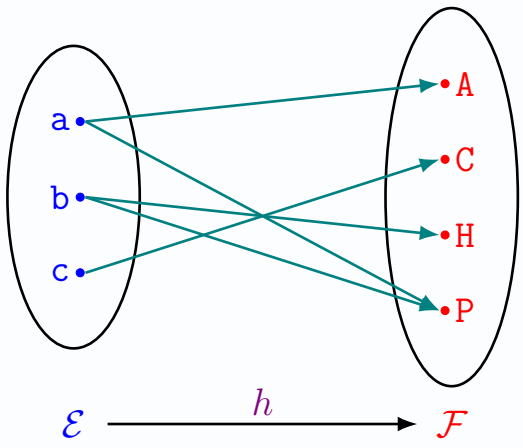
— ⟨DistElem⟩ := distance verticale entre les éléments ; défaut : ⟨0.75⟩
— ⟨DistEns⟩ := distance entre les « patates » ; défaut : ⟨4⟩
— ⟨LargEns⟩ := largeur des « patates » ; défaut : ⟨1.5⟩
— ⟨NomAppli⟩ := nom de l'application ; défaut : ⟨f⟩
— ⟨CouleurE⟩ := couleur de l'ensemble de départ ; défaut : ⟨blue⟩
— ⟨CouleurAppli⟩ := couleur de l'application ; défaut : ⟨violet⟩
— ⟨CouleurF⟩ := couleur de l'ensemble d'arrivée ; défaut : ⟨red⟩
— ⟨CouleurFleches⟩ := couleur des flèches ; défaut : ⟨teal⟩
— ⟨TypeFleche⟩ := type de la flèche ; défaut : ⟨Latex⟩
— ⟨Offset⟩ := décalage entre les flèches et les éléments (points) ; défaut : ⟨2pt⟩
— ⟨Epaisseur⟩ := épaisseur des tracés ; défaut : ⟨0.8pt⟩
— ⟨Police⟩ := police pour les éléments ; défaut : ⟨vide⟩
— ⟨NoirBlanc⟩ := booléen pour forcer l'affichage en N&B ; défaut : ⟨false⟩
— ⟨Labels⟩ := booléen pour afficher les noms des ensembles ; défaut : ⟨true⟩
— ⟨Ensembles⟩ := nom des ensembles ; défaut : ⟨$\mathcal{E}$/$\mathcal{F}$⟩
— ⟨PosLabels⟩ := position des labels, parmi ⟨haut/bas⟩. défaut : ⟨bas⟩

Le deuxième argument, optionnel et entre `<...>` propose des options, en langage `tikz` à passer à l'environnement.

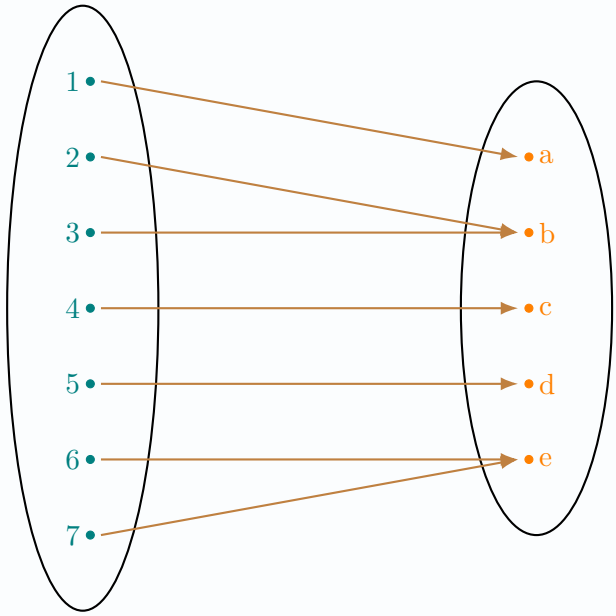
Le troisième argument, obligatoire et entre `{...}`, permet de préciser les *liaisons* sous la forme `x1/f(x1),x2/f(x2),...`.

9.3 Exemples

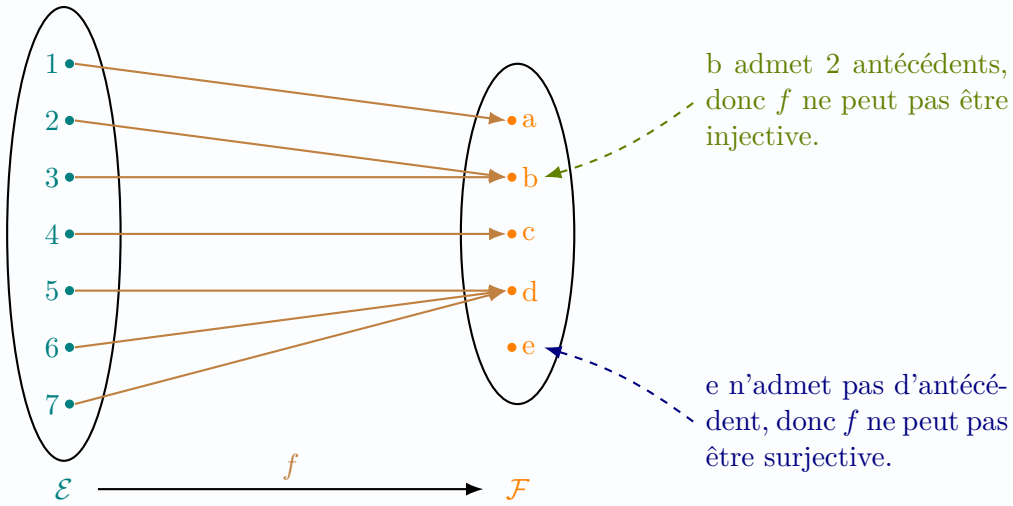
```
\DiagrammeSagittal[DistElem=1,DistEns=5,LargEns=1.75,Police={\Large\ttfamily},
    Epaisseur=1pt,NomAppli={$h$},E={a,b,c},F={A,C,H,P},
    PoliceLabels=\Large]{a/A,a/P,b/H,b/P,c/C}
```



```
\DiagrammeSagittal[%
    E={1,2,3,4,5,6,7},F={a,b,c,d,e},Labels=false,%
    DistElem=1,DistEns=6,LargEns=2,Offset=4pt,%
    CouleurE=teal,CouleurF=orange,CouleurAppli=brown,CouleurFleches=brown
    ]{1/a,2/b,3/b,4/c,5/d,6/e,7/e}
```




```
\begin{tikzpicture}
    \DiagrammeSagittal*[%
    E={1,2,3,4,5,6,7},F={a,b,c,d,e},DistEns=6,%
    CouleurE=teal,CouleurF=orange,CouleurAppli=brown,CouleurFleches=brown
    ]{1/a,2/b,3/b,4/c,5/d,6/d,7/d}
    \draw[lime!50!black,<-,thick,dashed,>=Latex] ($(Fb)+(12pt,0)$) to[bend
    right=10]++ (2,1) node[right] {\parbox{4cm}{b admet 2 antécédents, donc $f$ ne
    peut pas être injective.}} ;
    \draw[blue!50!black,<-,thick,dashed,>=Latex] ($(Fe)+(12pt,0)$) to[bend
    left=10]++ (2,-1) node[right] {\parbox{4cm}{e n'admet pas d'antécédent, donc
    $f$ ne peut pas être surjective.}} ;
\end{tikzpicture}
```



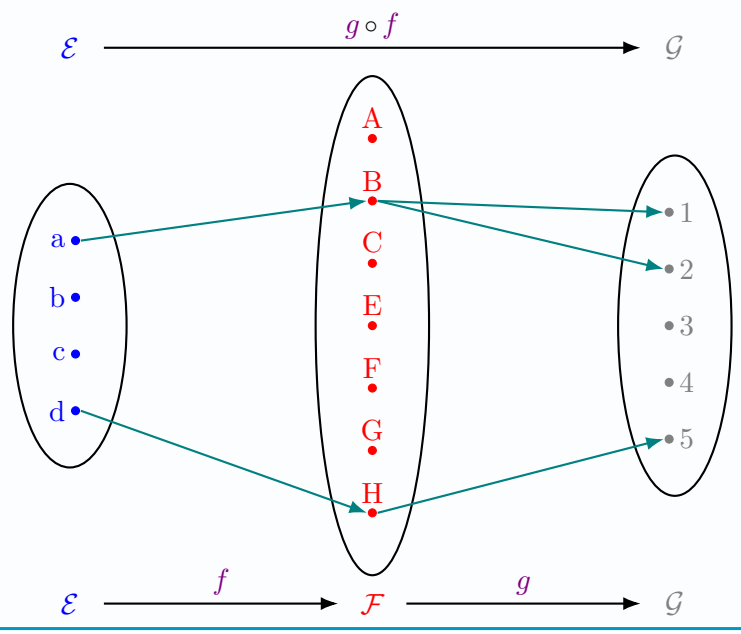
10 Diagramme sagittal d'une composée d'applications

10.1 Commande et fonctionnement global

La commande dédiée à la création d'un diagramme sagittal pour une application est `\DiagrammeSagittalCompo`.
Le diagramme créé est réalisé avec un environnement `tikzpicture`.

```
%commande autonome
\DiagrammeSagittalCompo[clés]<options tikz>{liaisons1}{liaisons2}

%commande à insérer dans un environnement tikzpicture
\begin{tikzpicture}
    \DiagrammeSagittalCompo*[clés]{liaisons1}{liaisons2}
\end{tikzpicture}
```

```
\DiagrammeSagittalCompo%
    [E={a,b,c,d},F={A,B,C,E,F,G,H},G={1,2,3,4,5}]%
    {a/B,d/H}%
    {B/1,B/2,H/5}
```



La majorité des paramètres sont personnalisables, mais le *thème* général est globalement *fixé*, dans le sens où ce sont les éléments *cosmétiques* qui pourront être modifiés.

La commande de création de `ProfSio` est volontairement pour des applications basiques, dans l'optique de travailler avec exemples en adéquation avec le programme de BTS SIO.

10.2 Arguments et clés

```
\DiagrammeSagittalCompo[clés]<options tikz>{liaisons1}{liaisons2}

\begin{tikzpicture}
    \DiagrammeSagittalCompo*[clés]{liaisons1}{liaisons2}
\end{tikzpicture}
```

Le code se charge, grâce aux ⟨clés⟩, de positionner et d'aligner les éléments des ensembles et les flèches.
De ce fait, les *écarts* entre les éléments d'un ensemble sont fixées globalement, tout comme le style général des flèches.

La version *étoilé* permet de ne pas créer l'environnement `tikzpicture`, pour d'éventuels rajouts ultérieurs :

- les éléments de l'ensemble de départ sont des nœuds nommés `(E...)` ;
- les éléments de l'ensemble du milieu sont des nœuds nommés `(F...)` ;
- les éléments de l'ensemble d'arrivée sont des nœuds nommés `(G...)`.

Les ⟨clés⟩ disponibles sont :

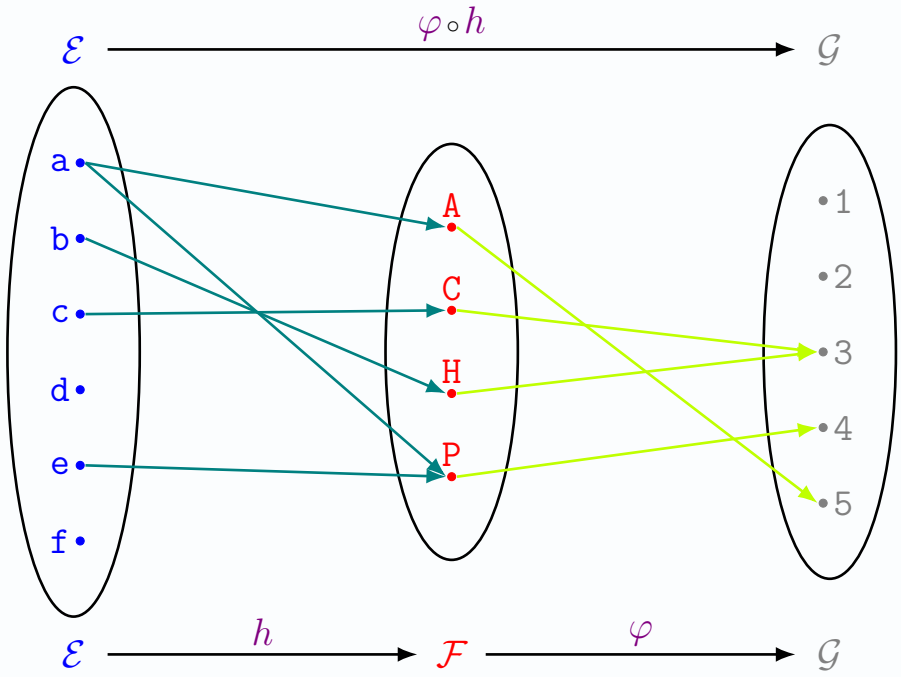
- ⟨DistElem⟩ := distance verticale entre les éléments ; défaut : ⟨0.75⟩
- ⟨DistEns⟩ := distance entre les « patates » ; défaut : ⟨4⟩
- ⟨LargEns⟩ := largeur des « patates » ; défaut : ⟨1.5⟩
- ⟨NomApplis⟩ := nom des applications ; défaut : ⟨f/g⟩
- ⟨CouleurE⟩ := couleur de l'ensemble de départ ; défaut : ⟨blue⟩
- ⟨CouleurApplis⟩ := couleurs des applications, ⟨Couleur⟩ ou ⟨Couleur_f/Couleur_g⟩ ; défaut : ⟨violet⟩
- ⟨CouleurF⟩ := couleur de l'ensemble du milieu ; défaut : ⟨red⟩
- ⟨CouleurG⟩ := couleur de l'ensemble d'arrivée ; défaut : ⟨gray⟩
- ⟨CouleursFleches⟩ := couleurs des flèches, ⟨Couleur⟩ ou ⟨Couleur_f/Couleur_g⟩ ; défaut : ⟨teal⟩
- ⟨TypeFleche⟩ := type de la flèche ; défaut : ⟨Latex⟩
- ⟨Offset⟩ := décalage entre les flèches et les éléments (points) ; défaut : ⟨2pt⟩
- ⟨Epaisseur⟩ := épaisseur des tracés ; défaut : ⟨0.8pt⟩
- ⟨Police⟩ := police pour les éléments ; défaut : ⟨vide⟩
- ⟨NoirBlanc⟩ := booléen pour forcer l'affichage en N&B ; défaut : ⟨false⟩
- ⟨Labels⟩ := booléen pour afficher les noms des ensembles ; défaut : ⟨true⟩
- ⟨Ensembles⟩ := nom des ensembles ; défaut : ⟨$\mathcal{E}$/$\mathcal{F}$/$\mathcal{G}$⟩
- ⟨PosLabels⟩ := position des labels, parmi ⟨haut/bas⟩. défaut : ⟨bas⟩

Le deuxième argument, optionnel et entre `<...>` propose des options, en langage `tikz` à passer à l'environnement.

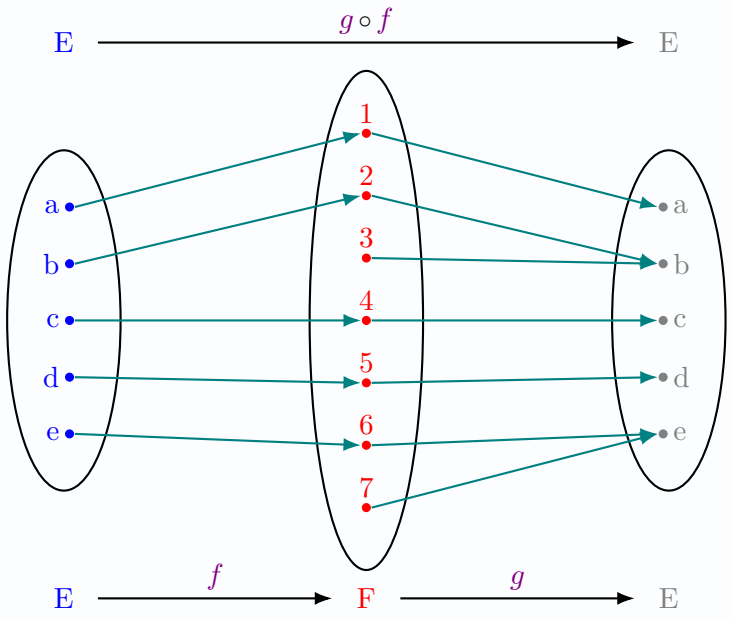
Les arguments `3` et `4`, obligatoires et entre `{...}`, permettent de préciser les *liaisons* sous la forme `x1/f(x1),x2/f(x2),...` et `y1/g(y1),y2/g(y2),...` .

10.3 Exemples

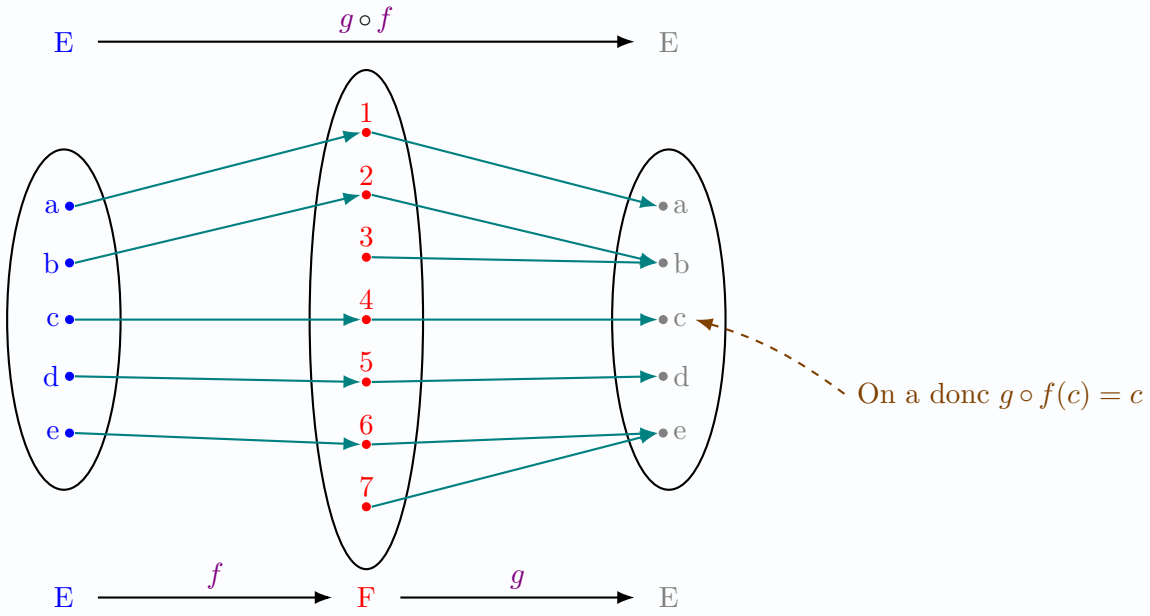
```
\DiagrammeSagittalCompo%
    [DistElem=1,DistEns=5,LargEns=1.75,Police={\Large\ttfamily},%
    Epaisseur=1pt,NomApplis={$h$/$\varphi$},CouleursFleches={teal/lime},%
    E={a,b,c,d,e,f},F={A,C,H,P},G={1,2,3,4,5},PoliceLabels=\Large]%
    {a/A,a/P,b/H,e/P,c/C}%
    {A/5,C/3,H/3,P/4}%
```




```
\DiagrammeSagittalCompo%
    [E={a,b,c,d,e},F={1,2,3,4,5,6,7},G={a,b,c,d,e},%
    Ensembles={E/F/E}]%
    {a/1,b/2,c/4,d/5,e/6}%
    {1/a,2/b,3/b,4/c,5/d,6/e,7/e}%
```



```
\begin{tikzpicture}
    \DiagrammeSagittalCompo*%
        [E={a,b,c,d,e},F={1,2,3,4,5,6,7},G={a,b,c,d,e},%
        Ensembles={E/F/E}]%
        {a/1,b/2,c/4,d/5,e/6}%
        {1/a,2/b,3/b,4/c,5/d,6/e,7/e}%
    \draw[orange!50!black,<-,thick,dashed,>=Latex] ($(Gc)+(12pt,0)$) to[bend
    left=10]++ (2,-1) node[right] {\parbox{4cm}{On a donc
    $g\,{\small\circ\,}f(c)=c$}} ;
\end{tikzpicture}
```



11 Table de vérité

11.1 Commande et fonctionnement global

La commande dédiée à la création d'une table de vérité (à deux variables minimum) est `\TableVerite`.
La commande est accessible **uniquement** en cas d'une compilation en LuaLaTeX !
Le tableau est créé grâce au package `nicematrix`.

Une **double compilation** peut être nécessaire pour le placement correct des filets !
Les fonctions LUA utilisées sont issues du `luatruthtable`, elles sont légèrement modifiées pour *coller* à une présentation plus classique.

```
\TableVerite[clés]<opts nicematrix>{vars}{colonnes_vars}{calculs}{colonnes_calculs}
```

```
\TableVerite{P}{$P$}%
    {lognot*P,P*logand*P,P*logor*P,P*iff*P,P*imp*P}%
    {$\lnot P$,$P \land P$,$P \lor P$,$P \Leftrightarrow P$,$P \Rightarrow P$}
```

P	$\neg P$	$P \land P$	$P \lor P$	$P \Leftrightarrow P$	$P \Rightarrow P$
V	F	V	V	V	V
F	V	F	F	V	V

```
\TableVerite{P,Q}{$P$,$Q$}{lognot*P,P*logand*Q}{$\lnot P$,$P \land Q$}
```

P	Q	$\neg P$	$P \land Q$
V	V	F	V
V	F	F	F
F	V	V	F
F	F	V	F

11.2 Arguments et clés pour la commande

```
\TableVerite[clés]<opts nicematrix>{vars}{colonnes_vars}{calculs}{colonnes_calculs}
```

En ce qui concerne la création du tableau, les ⟨clés⟩ sont :

- ⟨VF⟩ := pour préciser Vrai/Faux ; défaut : ⟨V/F⟩
- ⟨Swap⟩ := booléen pour commencer par FFF au lieu de VVV ; défaut : ⟨false⟩
- ⟨LargeursColonnes⟩ := largeur des colonnes, ⟨auto⟩ ou ⟨largeurG⟩ ou ⟨LargeurVar/LargeurCalc⟩ ; défaut : ⟨auto⟩
- ⟨CouleurEnonce⟩ := couleur de fond de la première ligne ; défaut : ⟨vide⟩
- ⟨CodeAvant⟩ := code `CodeBefore` (et avant l'éventuel coloriage de la première ligne) pour `nicetabular` ; défaut : ⟨vide⟩
- ⟨CodeApres⟩ := code `CodeAfter` pour `nicetabular`. défaut : ⟨vide⟩

Le deuxième argument, optionnel et entre `<...>` propose des options, en langage `nicematrix` à passer à la commande.

Le troisième argument, obligatoire et entre `{...}`, permet de spécifier les calculs à effectuer, en langage `luatruthtable`, notamment :

- `lognot*` pour le CONTRAIRE ;
- `*logand*` pour le ET ;
- `*logor*` pour le OU ;
- `*iff*` pour le ÉQUIVALENT ;
- `*imp*` pour le IMPLIQUE ;
- le reste est disponible dans la documentation (http://mirrors.ctan.org/macros/luatex/latex/luatruthtable/luatruthtable.pdf).

Le dernier argument, obligatoire et entre `{...}`, permet de spécifier les labels des calculs, en langage LaTeX cette fois-ci.

11.3 Compléments pour le package luatruthtable

Le tableau suivant présente les connecteurs logiques issues du package `luatruthtable` :

Opérateur	Syntaxe	Expression	Description
`lognot*`	`lognot*P`	$\neg P$	Négation de P
`*logand*`	`P*logand*Q`	$P \wedge Q$	Conjonction (et) de P et Q
`*logor*`	`P*logor*Q`	$P \vee Q$	Disjonction (ou) de P et Q
`*imp*`	`P*imp*Q`	$P \Rightarrow Q$	Implication de P vers Q
`*iff*`	`P*iff*Q`	$P \Leftrightarrow Q$	Équivalence de P et Q
`*lognand*`	`P*lognand*Q`	$\neg(P \wedge Q)$	NAND de P et Q
`*logxor*`	`P*logxor*Q`	$(P \vee Q) \wedge \neg(P \wedge Q)$	XOR de P et Q
`*lognor*`	`P*lognor*Q`	$\neg(P \vee Q)$	NOR de P et Q
`*logxnor*`	`P*logxnor*Q`	$(P \wedge Q) \vee (\neg P \wedge \neg Q)$	XNOR de P et Q

11.4 Exemples

```
\TableVerite{P,Q}{$P$,$Q$}{lognot*P,P*logand*Q}{$\lnot P$,$P \land Q$}
```

P	Q	$\neg P$	$P \wedge Q$
V	V	F	V
V	F	F	F
F	V	V	F
F	F	V	F

```
\TableVerite[LargeursColonnes=2cm]{P,Q}{$P$,$Q$}{lognot*P,P*logand*Q}{$\lnot P$,$P
    \land Q$}
```

P	Q	$\neg P$	$P \wedge Q$
V	V	F	V
V	F	F	F
F	V	V	F
F	F	V	F

```
\TableVerite[LargeursColonnes=1cm/2cm]{P,Q}{$P$,$Q$}{lognot*P,P*logand*Q}{$\lnot
    P$,$P \land Q$}
```

P	Q	$\neg P$	$P \land Q$
V	V	F	V
V	F	F	F
F	V	V	F
F	F	V	F

```
\TableVerite[CouleurEnonce=lightgray!25]{P,Q}{$P$,$Q$}{lognot*P,P*logand*Q}{$\lnot
    P$,$P \land Q$}
```

P	Q	$\neg P$	$P \land Q$
V	V	F	V
V	F	F	F
F	V	V	F
F	F	V	F

```
\TableVerite%
    [CodeAvant={\columncolor{red!15}{1}\columncolor{teal!15}{4}}]%
    {P,Q}{$P$,$Q$}{lognot*P,P*logand*Q}{$\lnot P$,$P \land Q$}
```

P	Q	$\neg P$	$P \land Q$
V	V	F	V
V	F	F	F
F	V	V	F
F	F	V	F

```
\TableVerite%
    [CodeApres={\UnderBrace[yshift=4pt]{1-4}{5-4}{et}}]%
    {P,Q}{$P$,$Q$}{lognot*P,P*logand*Q}{$\lnot P$,$P \land Q$}
\hspace{5mm}
\TableVerite%
    [Swap,CodeApres={\UnderBrace[yshift=4pt]{1-4}{5-4}{et}}]%
    {P,Q}{$P$,$Q$}{lognot*P,P*logand*Q}{$\lnot P$,$P \land Q$}
\hspace{5mm}
\TableVerite%
    [VF={Vrai/Faux},CodeApres={\UnderBrace[yshift=4pt]{1-4}{5-4}{et}}]%
    {P,Q}{$P$,$Q$}{lognot*P,P*logand*Q}{$\lnot P$,$P \land Q$}
\vspace*{0.75cm}
```

P	Q	$\neg P$	$P \land Q$
V	V	F	V
V	F	F	F
F	V	V	F
F	F	V	F

et

P	Q	$\neg P$	$P \land Q$
F	F	V	F
F	V	V	F
V	F	F	F
V	V	F	V

et

P	Q	$\neg P$	$P \land Q$
Vrai	Vrai	Faux	Vrai
Vrai	Faux	Faux	Faux
Faux	Vrai	Vrai	Faux
Faux	Faux	Vrai	Faux

et

```
\TableVerite%
    [CodeAvant={\columncolor{red!15}{5}\columncolor{red!15}{8}}]%
    {P,Q,R}%
    {$P$,$Q$,$R$}%
    {%
        Q*logand*R,P*logor*(Q*logand*R),P*logor*Q,%
        Q*logor*R,(P*logor*Q)*logand*(P*logor*R)
    }%
    {$Q \land R$,$P \lor (Q \land R)$,$P \lor Q$,$Q \lor R$,$(P \lor Q) \land (P
    \lor R)$}
```

P	Q	R	$Q \land R$	$P \lor (Q \land R)$	$P \lor Q$	$Q \lor R$	$(P \lor Q) \land (P \lor R)$
V	V	V	V	V	V	V	V
V	V	F	F	V	V	V	V
V	F	V	F	V	V	V	V
V	F	F	F	V	V	F	V
F	V	V	V	V	V	V	V
F	V	F	F	F	V	V	F
F	F	V	F	F	F	V	F
F	F	F	F	F	F	F	F

```
%Loi de De Morgan

\TableVerite%
    [CouleurEnonce=lightgray!15,LargeursColonnes=0.75cm/2cm,%
    CodeAvant={\columncolor{teal!10}{6}\columncolor{teal!10}{7}}]%
    {P,Q}{$P$,$Q$}%
    {lognot*P,lognot*Q,P*logand*Q,
        lognot*(P*logand*Q),(lognot*P)*logor*(lognot*Q)}%
    {$\lnot P$,$\lnot Q$,$P\land Q$,$\lnot(P\land Q)$,$(\lnot P)\lor(\lnot Q)$}

\TableVerite%
    [CouleurEnonce=lightgray!15,LargeursColonnes=0.75cm/2cm,VF={1/0},%
    CodeAvant={\columncolor{orange!10}{6}\columncolor{orange!10}{7}}]%
    {P,Q}{$P$,$Q$}%
    {lognot*P,lognot*Q,P*logand*Q,
        lognot*(P*logand*Q),(lognot*P)*logor*(lognot*Q)}%
    {$\lnot P$,$\lnot Q$,$P\land Q$,$\lnot(P\land Q)$,$(\lnot P)\lor(\lnot Q)$}
```

P	Q	$\neg P$	$\neg Q$	$P \land Q$	$\neg(P \land Q)$	$(\neg P) \lor (\neg Q)$
V	V	F	F	V	F	F
V	F	F	V	F	V	V
F	V	V	F	F	V	V
F	F	V	V	F	V	V

P	Q	$\neg P$	$\neg Q$	$P \land Q$	$\neg(P \land Q)$	$(\neg P) \lor (\neg Q)$
1	1	0	0	1	0	0
1	0	0	1	0	1	1
0	1	1	0	0	1	1
0	0	1	1	0	1	1