

Pensée et écriture non linéaires pour la génération de documents et programmes : le processeur KM2

Laurent Gouzènes

Institut Fredriks Bull
25 Janvier 2025

L'art du rangement(Ursus Wehrli)







De l'importance de la mise en page

ACTE 1 - SCÈNE V - DON DIÈGUE, DON RODRIGUE
DON DIÈGUE Rodrigue, as-tu du coeur ?
DON RODRIGUE Tout autre que mon père L'éprouverait sur l'heure.
DON DIÈGUE Agréable colère ! Digne ressentiment à ma douleur bien doux ! Je reconnais mon sang à ce noble courroux ; Ma jeunesse revit en cette ardeur si prompte. Viens, mon fils, viens, mon sang, viens réparer ma honte ; Viens me venger.
DON RODRIGUE De quoi ?
DON DIÈGUE D'un affront si cruel, Qu'à l'honneur de tous deux il porte un coup mortel : D'un soufflet. L'insolent en eût perdu la vie ; Mais mon âge a trompé ma généreuse envie ; Et ce fer que mon bras ne peut plus soutenir, Je le remets au tien pour venger et punir. Va contre un arrogant éprouver ton courage : Ce n'est que dans le sang qu'on lave un tel outrage ; Meurs, ou tue. Au surplus, pour ne te point flatter, Je te donne à combattre un homme à redouter ; Je l'ai vu, tout couvert de sang et de poussière, Porter partout l'effroi dans une armée entière. J'ai vu par sa valeur cent escadrons rompus ; Et pour t'en dire encor quelque chose de plus, Plus que brave soldat, plus que grand capitaine, C'est ...
DON RODRIGUE De grâce, achevez.
DON DIÈGUE Le père de Chimène.
DON RODRIGUE Le ...
DON DIÈGUE Ne réplique point, je connais ton amour, Mais qui peut vivre infâme est indigne du jour ;

(3)

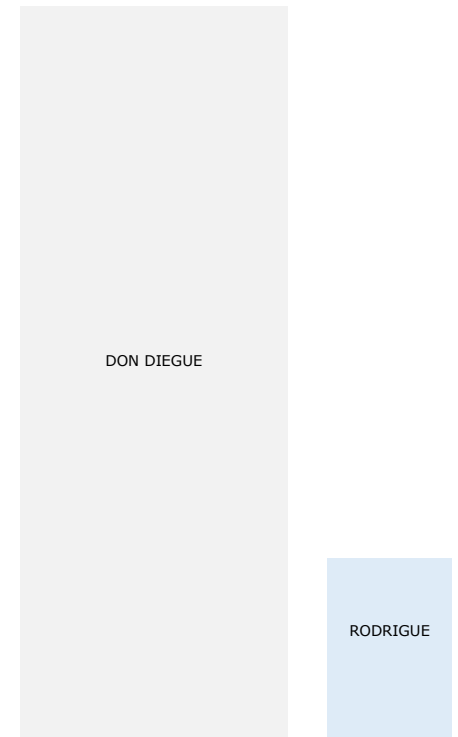
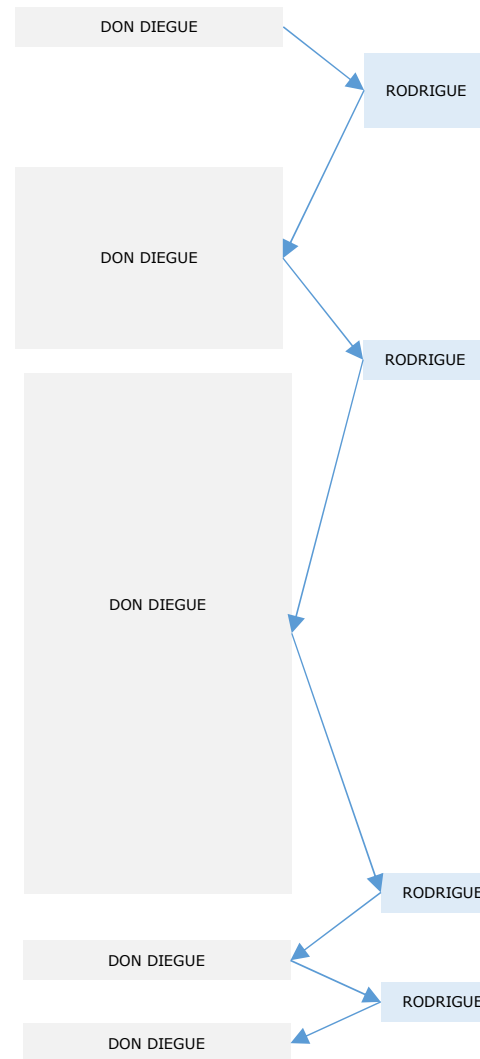
Document = contenu + contenant + ...

Numero  		
	DON DIÈGUE	DON RODRIGUE
1	Rodrigue, as-tu du coeur ?	
2		Tout autre que mon père L'éprouverait sur l'heure.
3	Agréable colère ! Digne ressentiment à ma douleur bien doux ! Je reconnais mon sang à ce noble courroux ; Ma jeunesse revit en cette ardeur si prompte. Viens, mon fils, viens, mon sang, viens réparer ma honte ; Viens me venger.	
4		De quoi ?
5	D'un affront si cruel, Qu'à l'honneur de tous deux il porte un coup mortel ; D'un soufflet. L'insolent en eût perdu la vie ; Mais mon age a trompé ma généreuse envie ; Et ce fer que mon bras ne peut plus soutenir, Je le remets au tien pour venger et punir. Va contre un arrogant éprouver ton courage : Ce n'est que dans le sang qu'on lave un tel outrage ; Meurs, ou tue. Au surplus, pour ne te point flatter, Je te donne à combattre un homme à redouter ; Je l'ai vu, tout couvert de sang et de poussière, Porter partout l'effroi dans une armée entière. J'ai vu par sa valeur cent escadrons rompus ; Et pour t'en dire encor quelque chose de plus, Plus que brave soldat, plus que grand capitaine, C'est ...	
6		De grâce, achevez.
7	Le père de Chimène.	
8		Le ...
9	Ne réplique point, je connais ton amour, Mais qui peut vivre infâme est indigne du jour ;	

Acte 1 – Scène 5

	DON DIÈGUE	DON RODRIGUE
		✖ Sous-sujet 1
DON DIÈGUE	Rodrigue, as-tu du coeur ?	
DON RODRIGUE		Tout autre que mon père L'éprouverait sur l'heure.
DON DIÈGUE	Agréable colère ! Digne ressentiment à ma douleur bien doux ! Je reconnais mon sang à ce noble courroux ; Ma jeunesse revit en cette ardeur si prompte. Viens, mon fils, viens, mon sang, viens réparer ma honte ; Viens me venger.	
DON RODRIGUE		De quoi ?
DON DIÈGUE	D'un affront si cruel, Qu'à l'honneur de tous deux il porte un coup mortel ; D'un soufflet. L'insolent en eût perdu la vie ; Mais mon âge a trompé ma généreuse envie ; Et ce fer que mon bras ne peut plus soutenir, Je le remets au tien pour venger et punir. Va contre un arrogant éprouver ton courage : Ce n'est que dans le sang qu'on lave un tel outrage ; Meurs, ou tue. Au surplus, pour ne te point flatter, Je te donne à combattre un homme à redouter ; Je l'ai vu, tout couvert de sang et de poussière, Porter partout l'effroi dans une armée entière. J'ai vu par sa valeur cent escadrons rompus ; Et pour t'en dire encor quelque chose de plus, Plus que brave soldat, plus que grand capitaine, C'est ...	
DON RODRIGUE		De grâce, achevez.
DON DIÈGUE	Le père de Chimène.	
DON RODRIGUE		Le ...
DON DIÈGUE	Ne réplique point, je connais ton amour, Mais qui peut vivre infâme est indigne du jour ;	

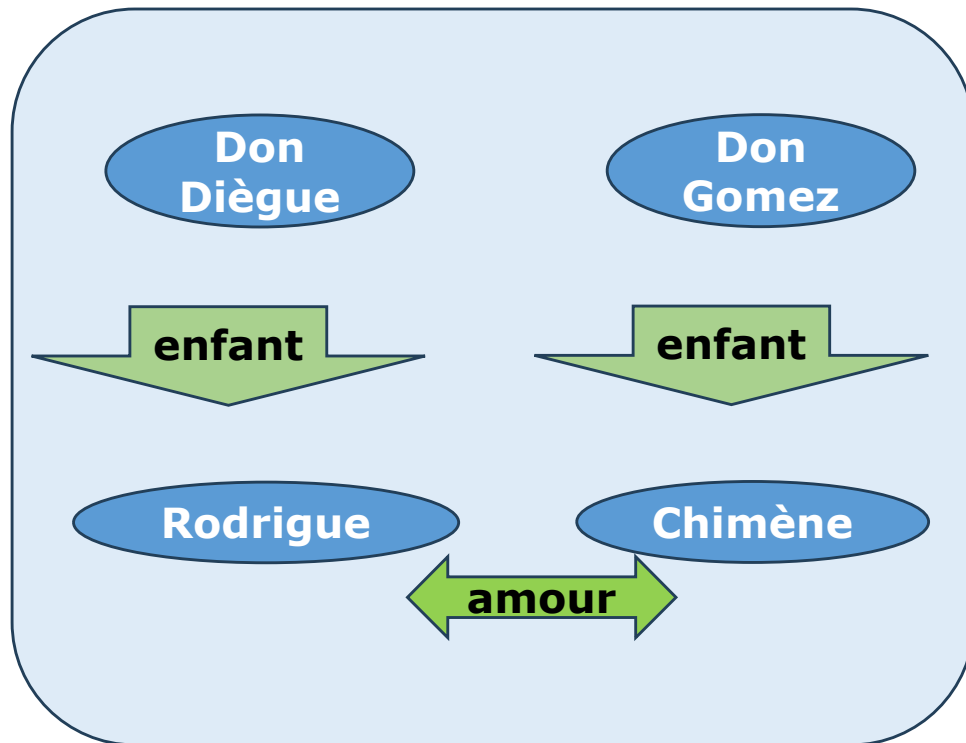
(4)



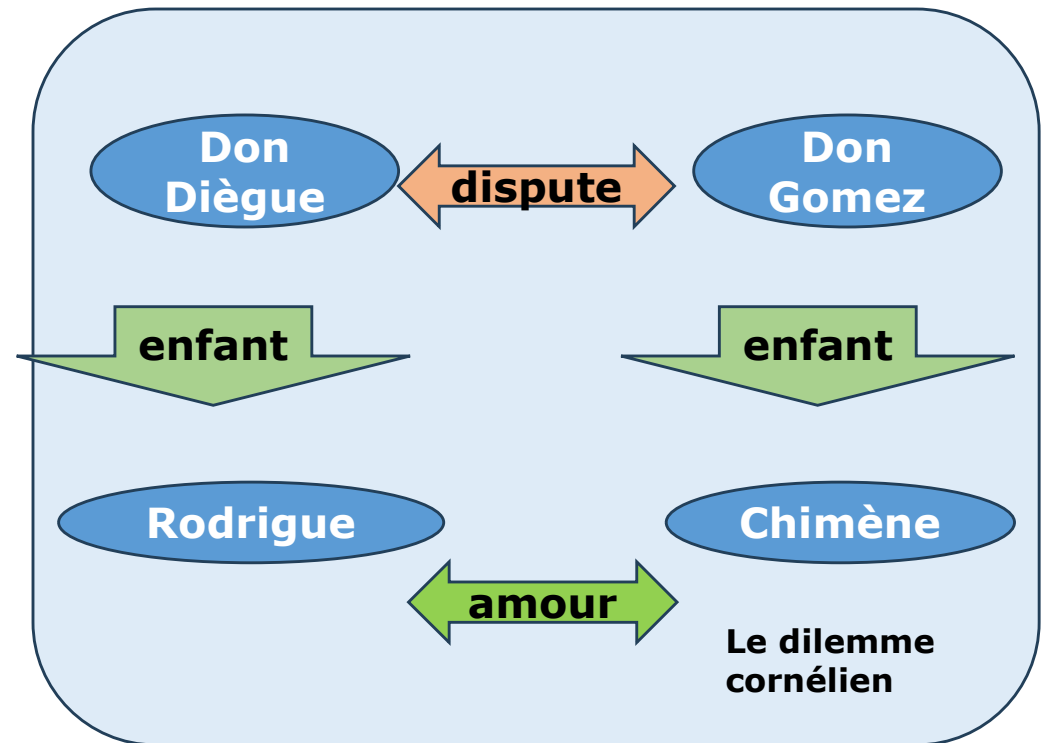
(5)

(5u)

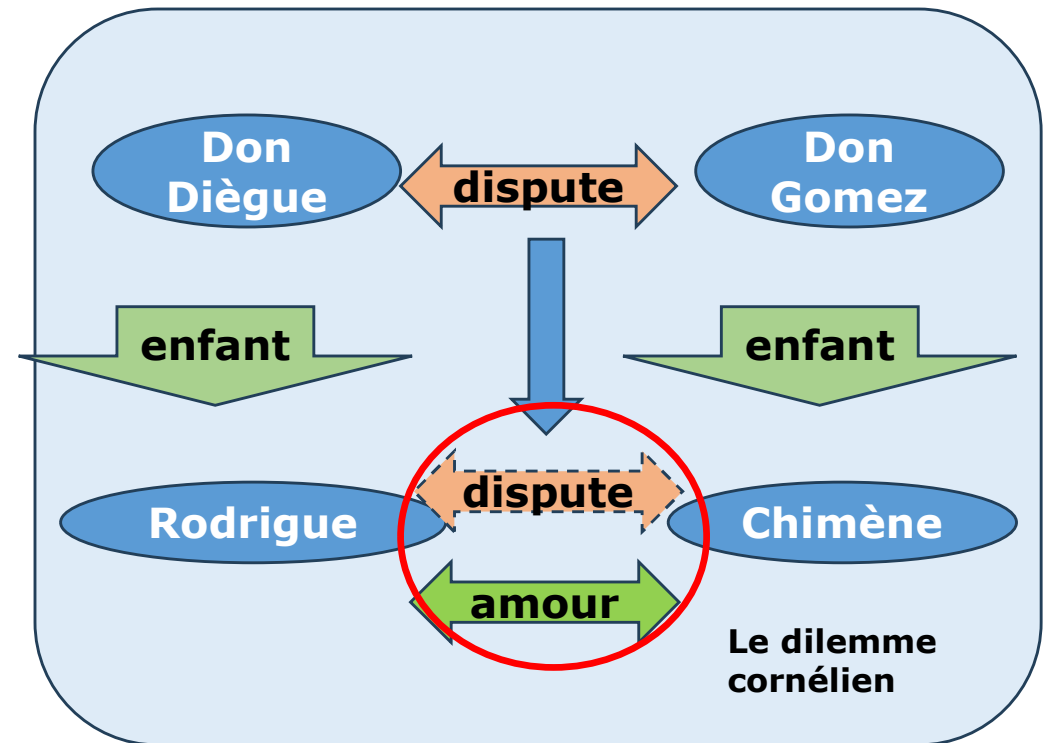
L'art du conteur



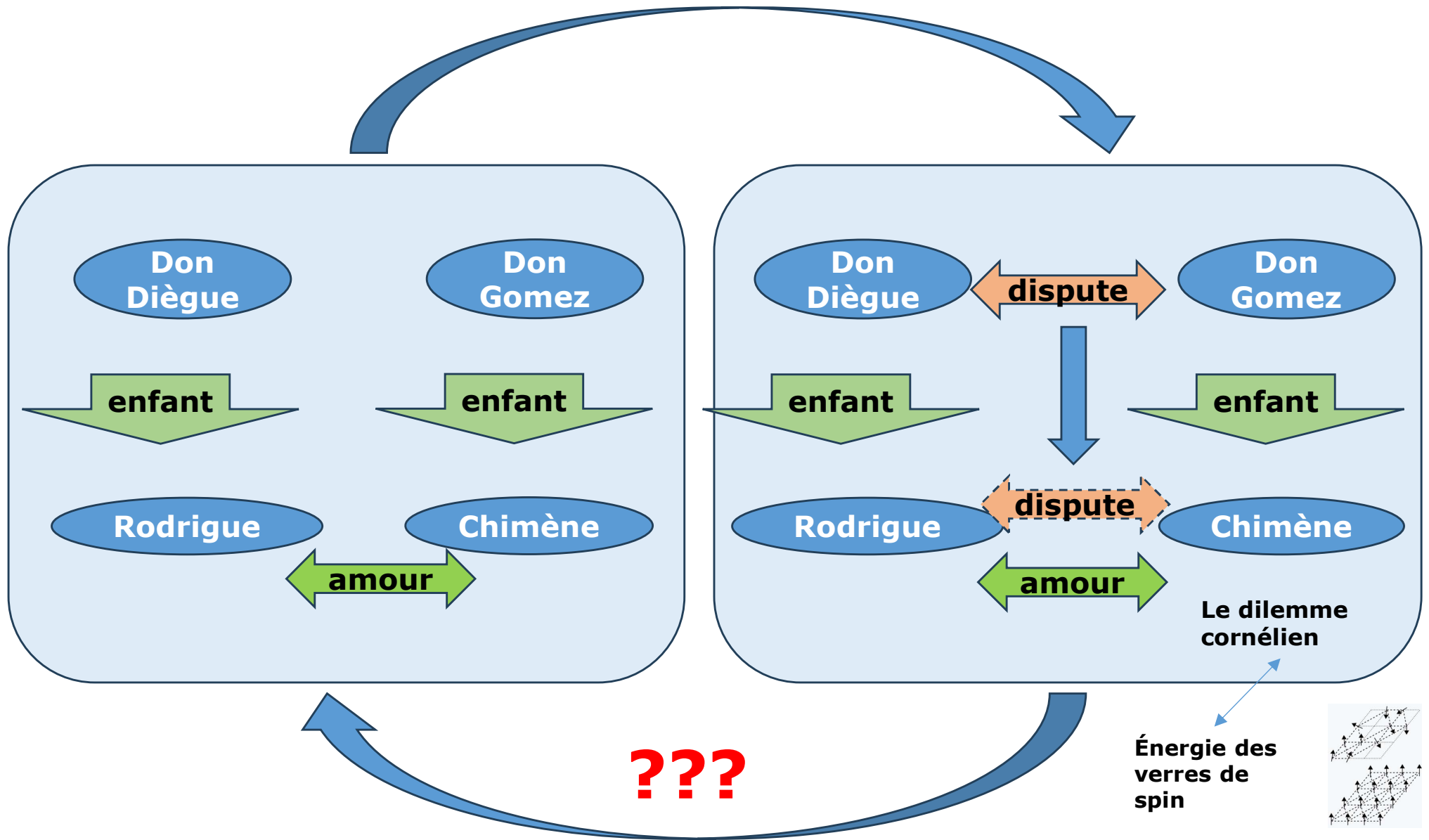
L'art du conteur



L'art du conteur

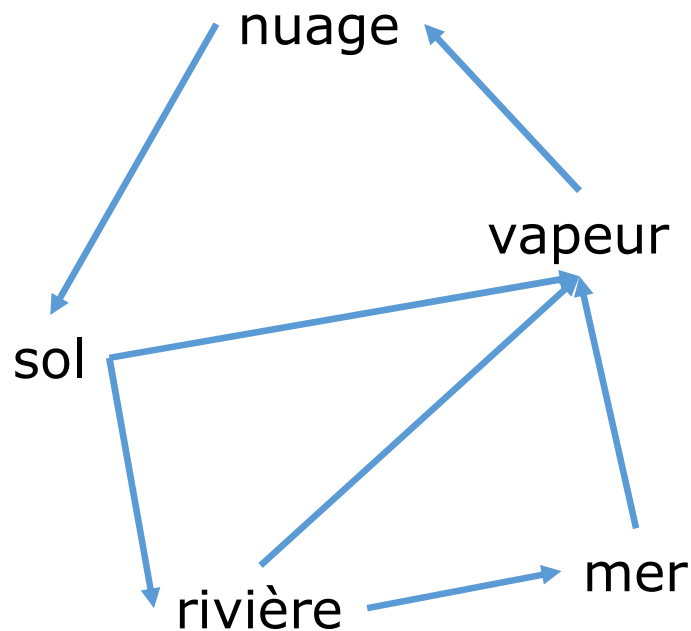


L'art du conteur



Cycle de l'eau : linéaire vs non-linéaire

Phénomène physique



Description littéraire ?

- **Point de départ = ?**
- **Contenu de chaque paragraphe ?**
- **Ordre des paragraphes ?**
- **Liens entre paragraphes**

Exemple :

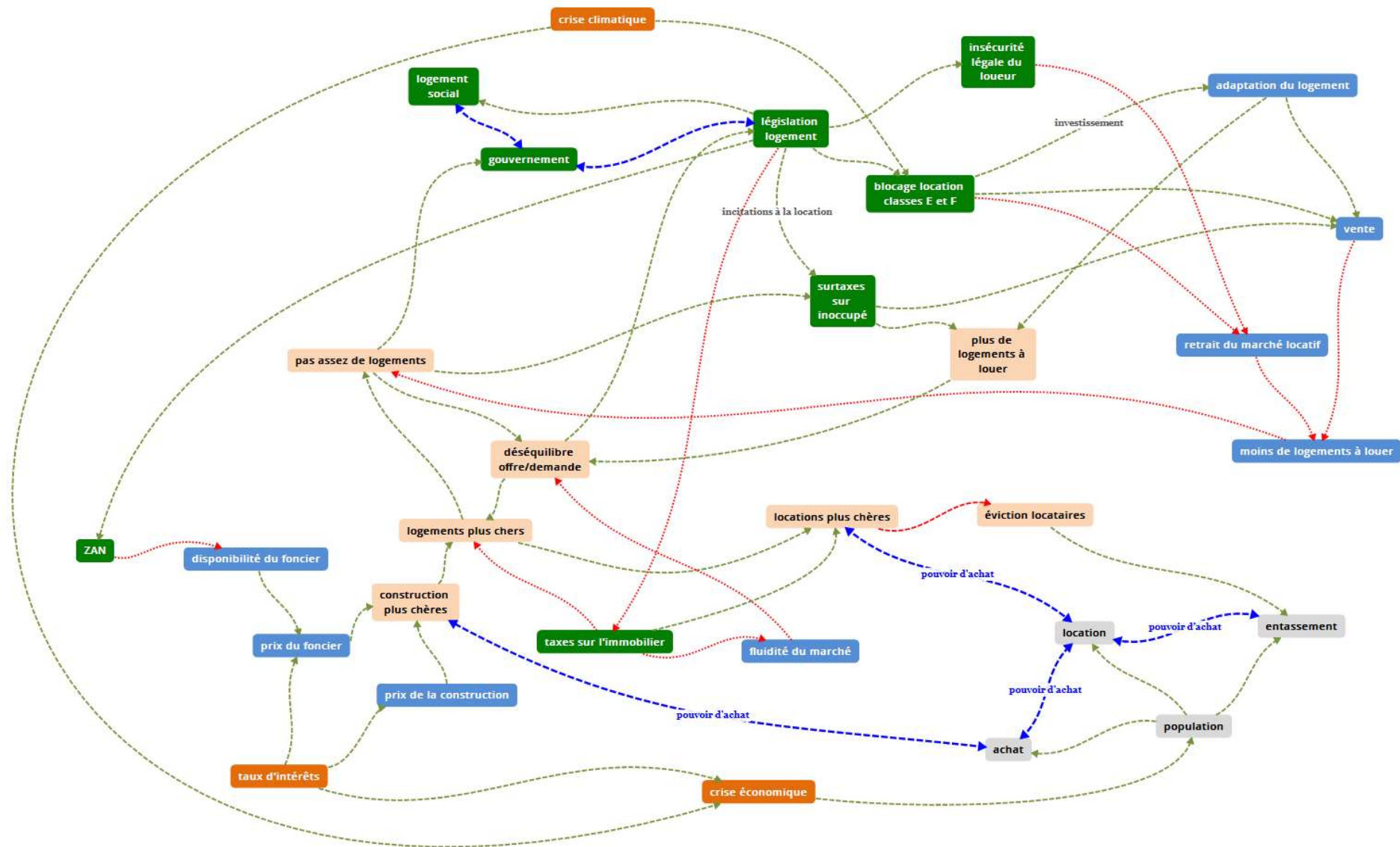
- Description depuis l'entrée :
 - La vapeur d'eau est produite par le sol, les rivières et les mers.
- Description depuis la sortie :
 - La rivière s'évapore et coule vers la mer

Au moins 10 textes possibles

Le texte linéaire fige des priorités, une interprétation, des biais cognitifs, etc

→ Difficultés pour exprimer la compréhension de systèmes complexes

Systeme complexe : la crise du logement



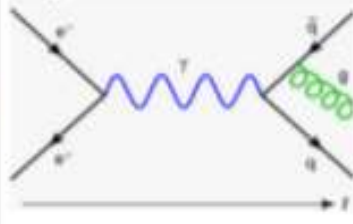



Expression linéaires et non linéaires de la connaissance

The following table illustrates these different points of views and uses.

Graphismes

Organisation

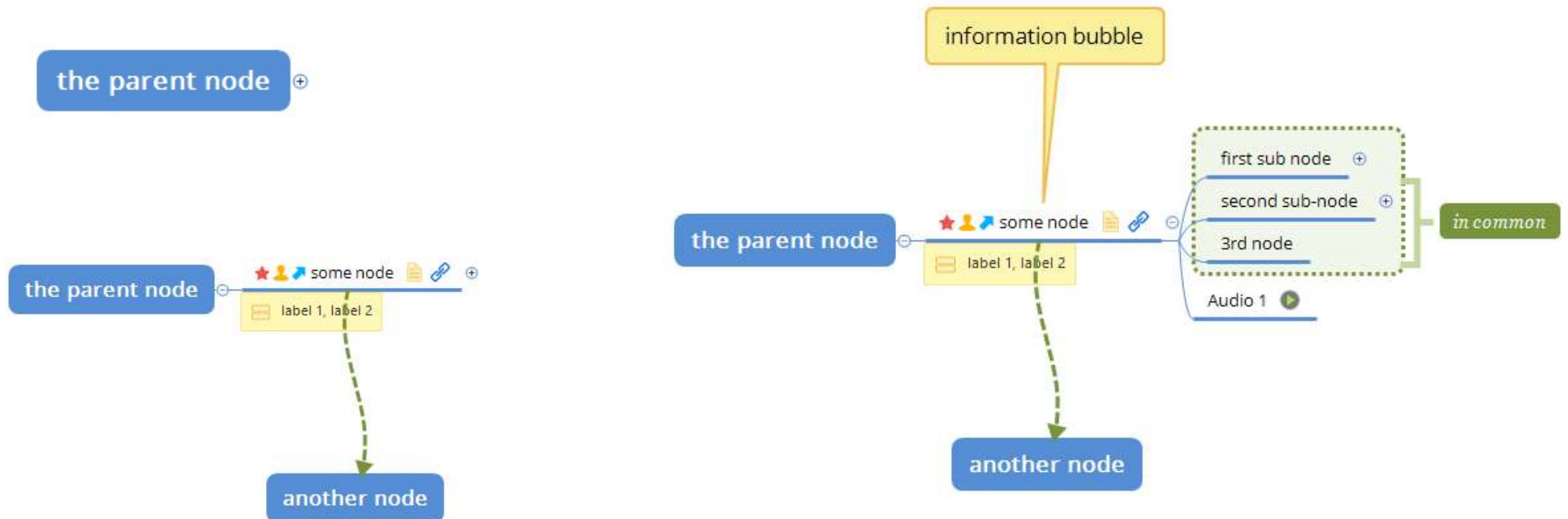
From various writing systems	Nsibidi 	ISOTYPES 	Typographs <i>Typography</i> Frutiger, TechCrunch TYPOGRAPHY Helvetica Bold, Grules <i>Typography</i> ITC Officina Bold, Bloglines Typography Interstate, 37signals Typography Merriam Pro Bold, Tailrank	Linguist This is a sentence in english, written with the latin set of characters.
to various meanings	Mathematics $\int_a^b f(x)$	Computer program <pre>def fact (n) : + DESCRIPTION + computes the factorial function in if n < 1 : returns 1 else : returns n * fact (n -1)</pre>	Physics 	Music 

+ systèmes complexes

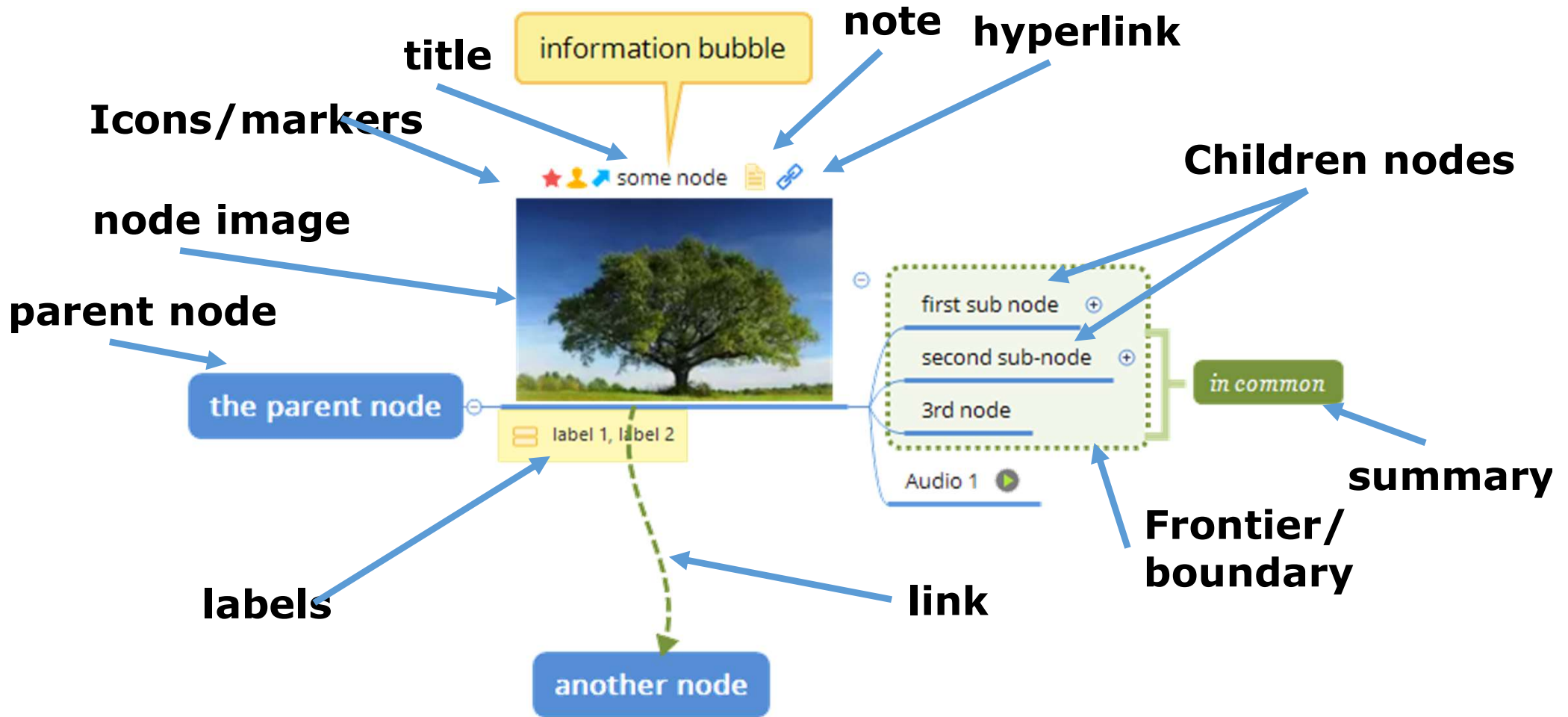
Diagrammes SGH

Structure of a SGH node

SGH Script Graphic Hyperlink

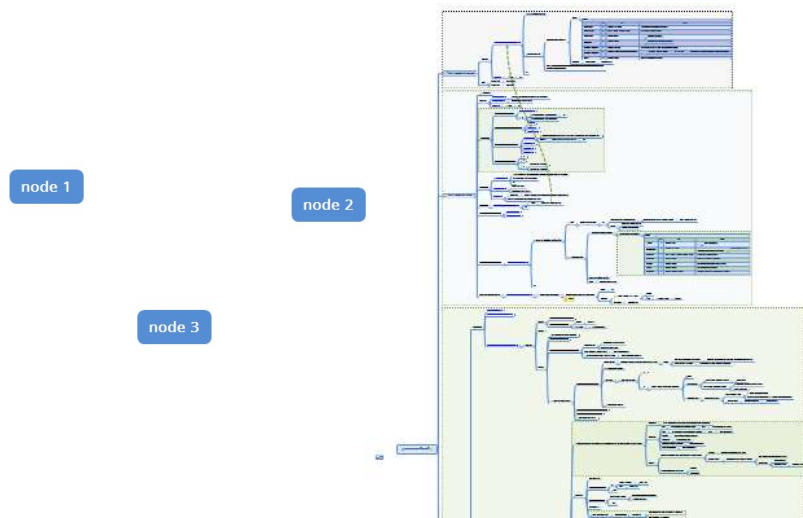


Vocabulary : a SGH node



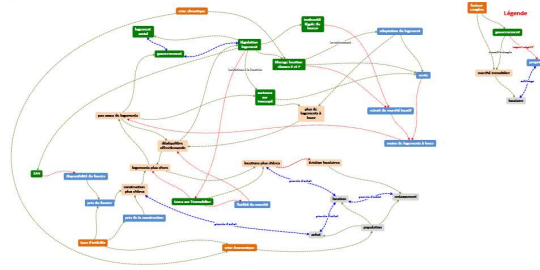
Assembly of SGH nodes → SGH schemes

4 ways of assembly



Juxtaposition

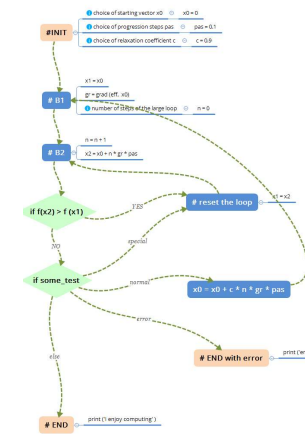
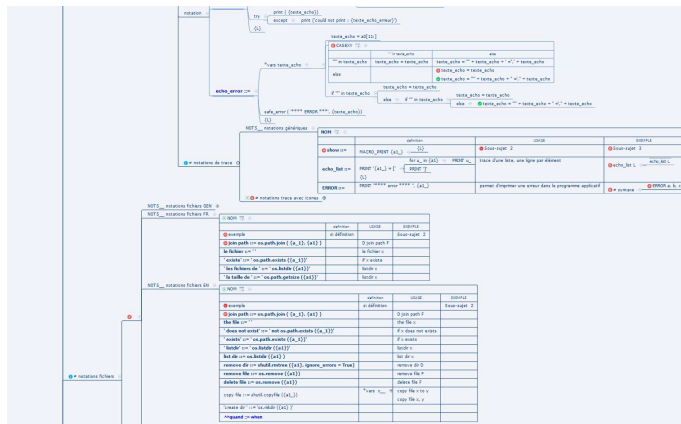
Tree



Graph

definition	USAGE	EXEMPLE
si definition		
example		Sous-sujet 2
join path := os.path.join ({a_1}, {a1})	D join path F	
the file := "	the file x	
' does not exist' := ' not os.path.exists ({a_1})'	if x does not exist	
' exists' := ' os.path.exists ({a_1})'	if x exists	
' listdir' := ' os.listdir ({a1})'	listdir x	
list dir := os.listdir ({a1})	list dir x	
remove dir := shutil.rmtree ({a1}, ignore_errors = True)	remove dir D	
remove file := os.remove ({a1})	remove file F	
delete file := os.remove ({a1})	delete file F	
copy file := shutil.copyfile ({a_1})	*vars x_...	copy file x to y
' create dir ' := ' os.mkdir ({a1})'	copy file x, y	
^* quand := when		

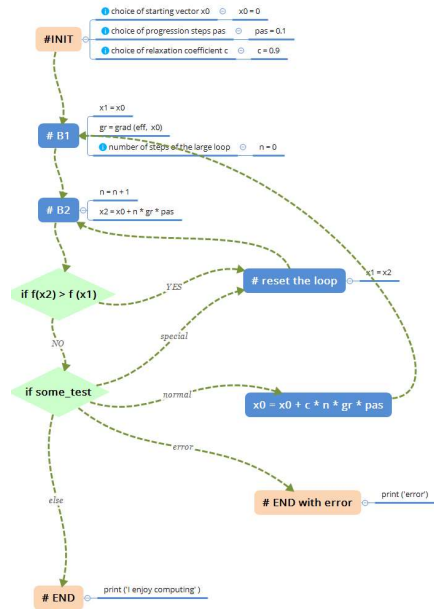
Table



And all combinations of these assemblies

Quelques exemples

processus



programmation

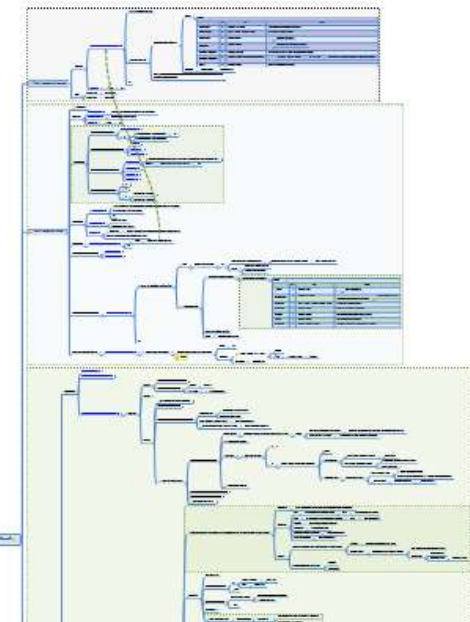
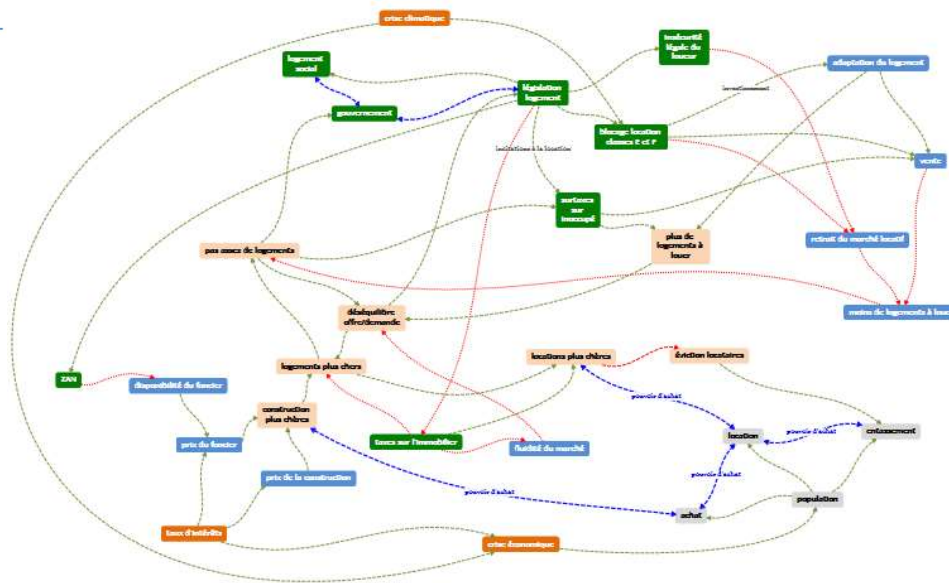


Schéma causal



Tableaux : une primitive indispensable

- **Caractéristiques:**

- **Structuration forte de l'information**
- **Intégration « sans couture » dans les forêts**
- **Les tableaux sont récursifs (tableau dans tableau, arbre dans tableau, tableau dans arbre, etc)**

trees should be read.

*module** command takes several arguments lang the language that will be used to interpret the nodes.

argument name	type of argument	possible values	
lang	string	py, python, html, cpp, js, javascript, ada, prolog, etc...	This argument is mandatory ! see list of supported languages and how to extend languages.
tab	integer <= 8	0 .. 8	for indented languages (programming languages) describes the standard equivalent of one tab.
offset	integer	0..8	All lines are systematically indented by a constant tab value.
no_start	boolean	ON/OFF, true/False	Standard start lines are for example in python : in html : etc
h_string	string	'a-----'	the h_string will be inserted systematically at the beginning of EACH line of the output.

examples

```
**module** lang = py
**module** lang = htm
**module** lang = html
**module** lang = cpp
```

Comment

```
this will start the processor to interpret sub-trees as python, html, c+/, etc
```

```
**module** lang = py, tab=4, h_string = 'a '
```

will produce a file with some python code which is all commented

Standard macros are available in all KM2 languages.

Basic icons for control

- ignore once
- ignore full sub_tree
- one line of documentation
- many lines of documentation
- standard macros

Standard macros are defined as :

- ignore once
- ignore full sub_tree
- one line of documentation
- many lines of documentation

Examples

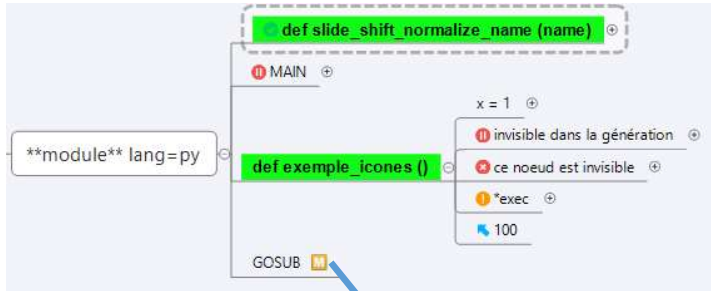
source code	generated code
<pre>% Example % this is ignored % all the subtree is ignored comment 1 line comment many lines foo foo :-</pre>	<pre>% EXAMPLE 2 % Example. foo. foo :- A, B, C, some comment D, E. foo. % END</pre>

Example of use

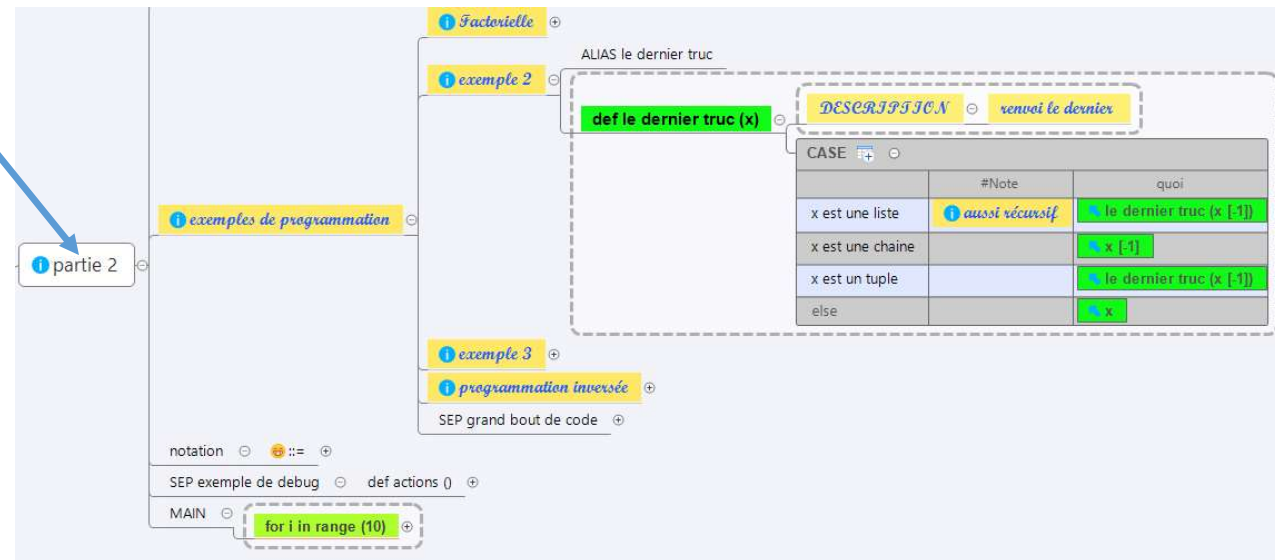
This is the standard way to write programs in KM2, in any language.

in Prolog

Usage des hyperliens : liens entre plans d'écriture



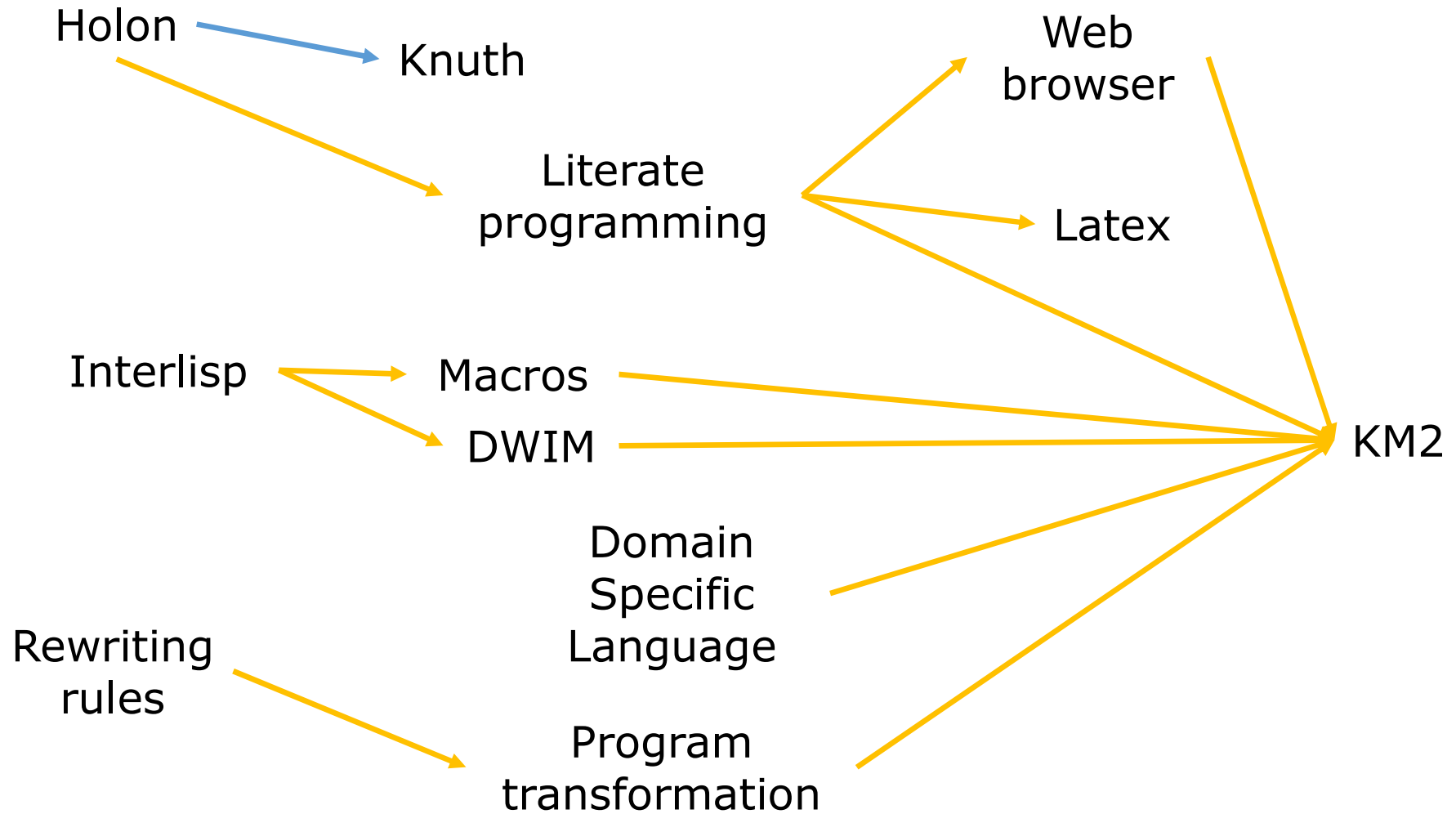
Autre fichier, autre page, etc



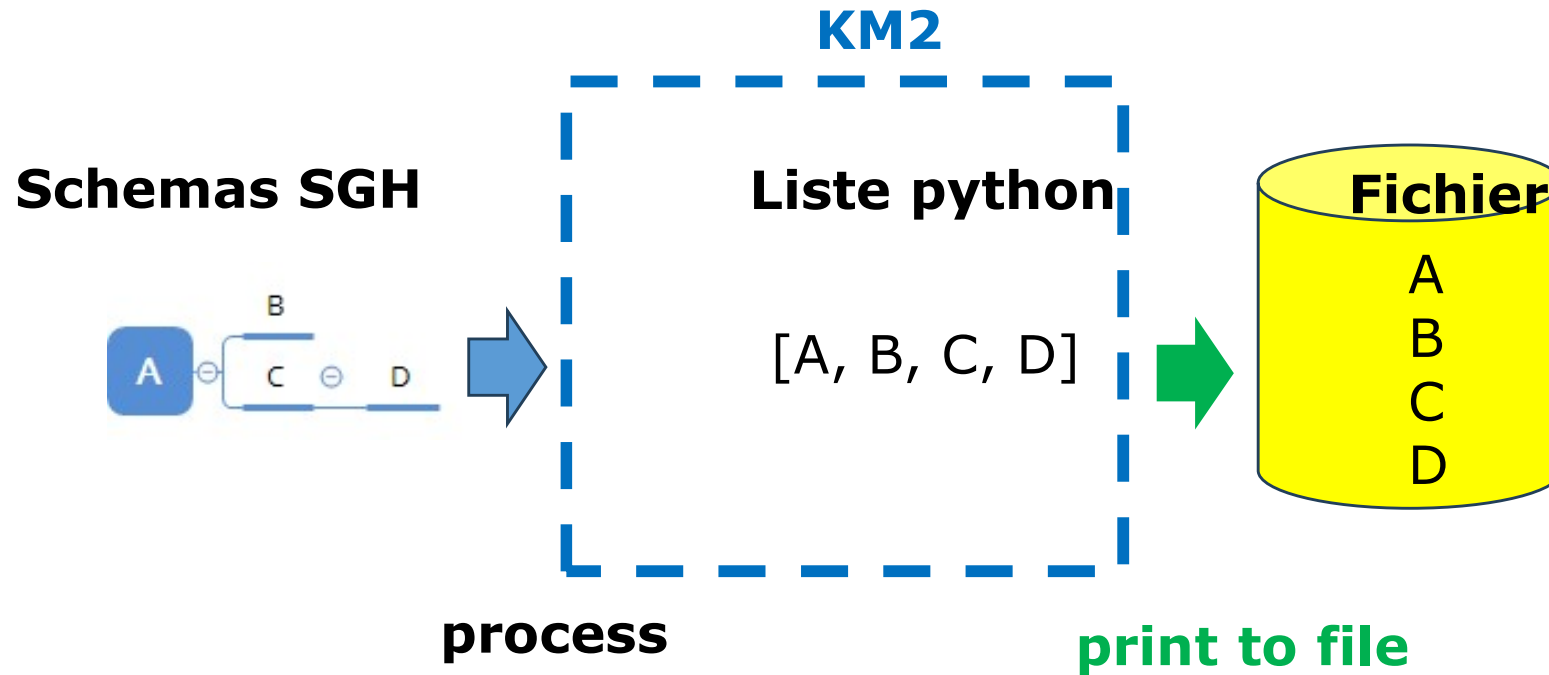
Permet la « factorisation » de textes, programmes, macros, etc

Le processeur KM2

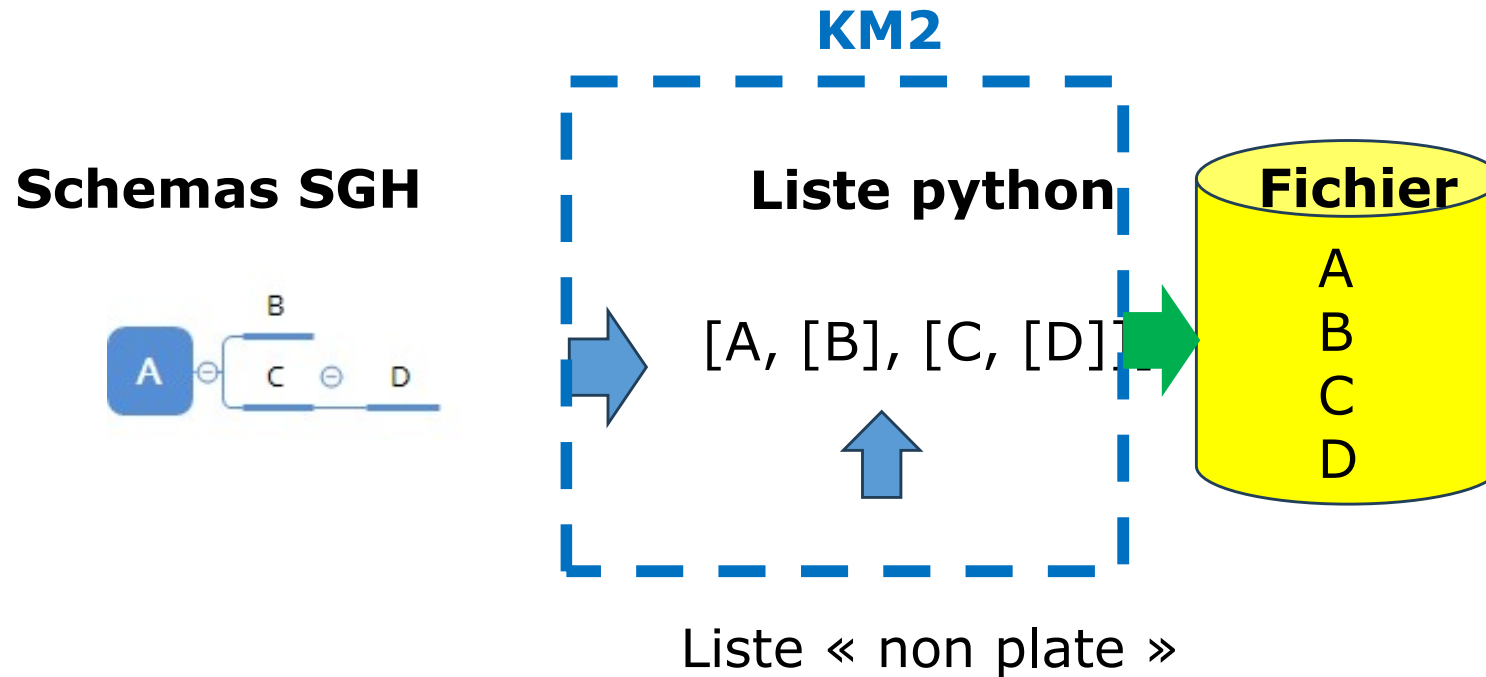
Héritages conceptuels



Etapes de transformation (base)

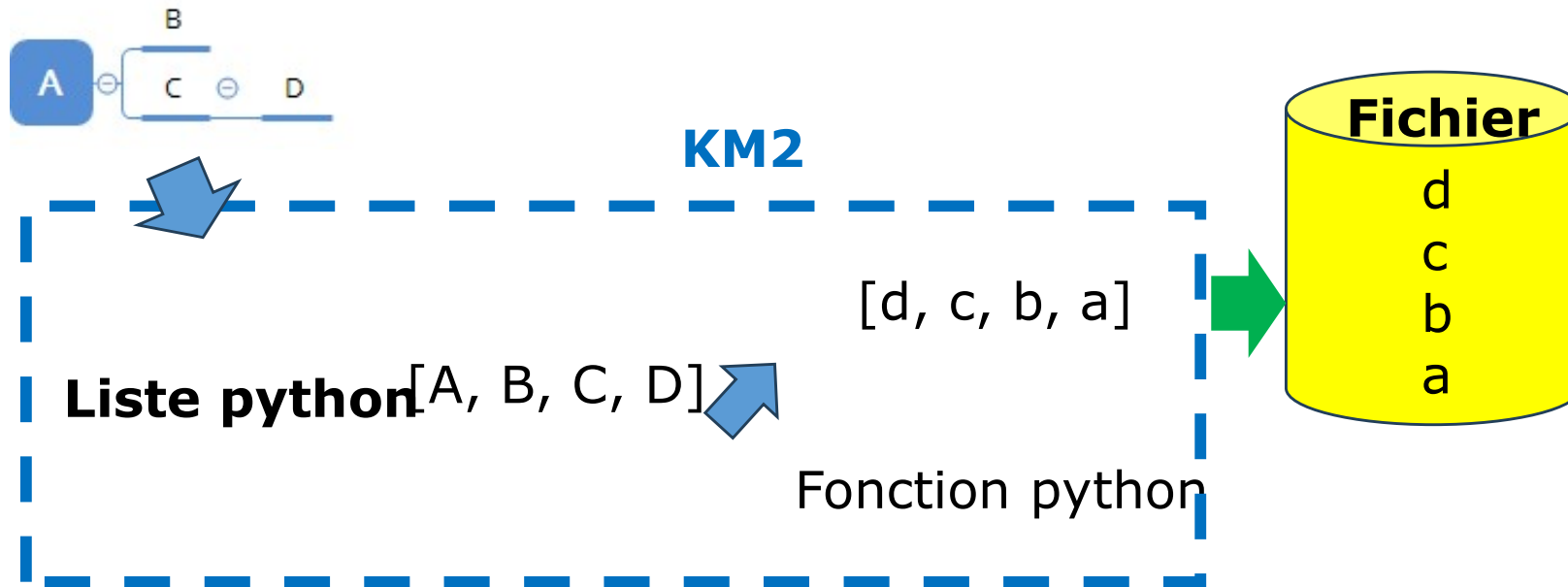


Etapes de transformation (liste non plate)

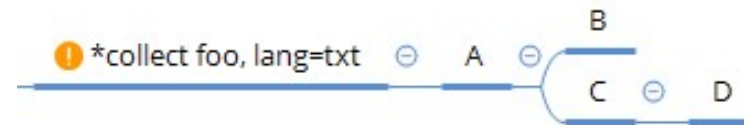


Etapes de transformation

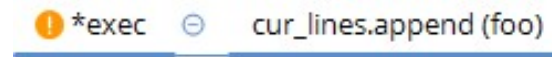
Schemas SGH



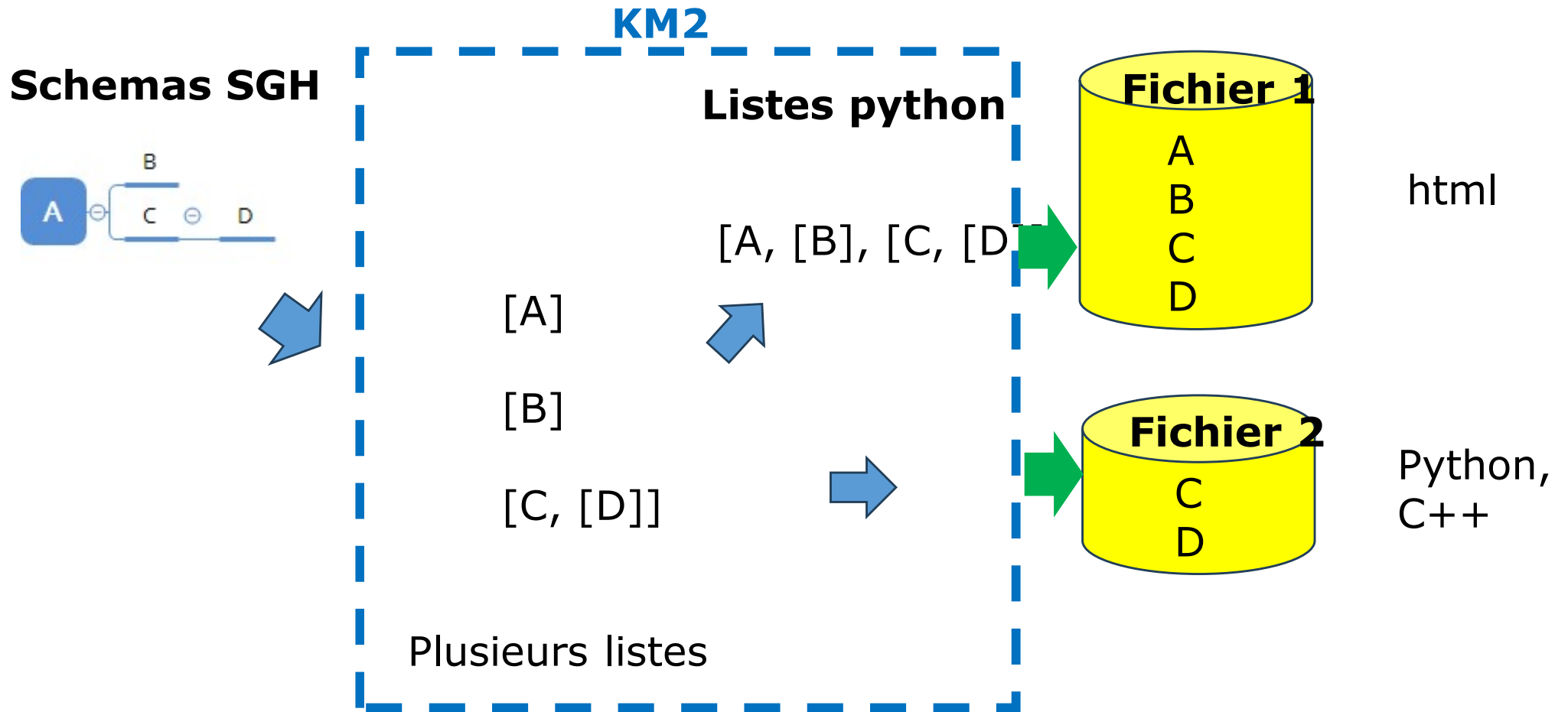
Creation de la liste foo



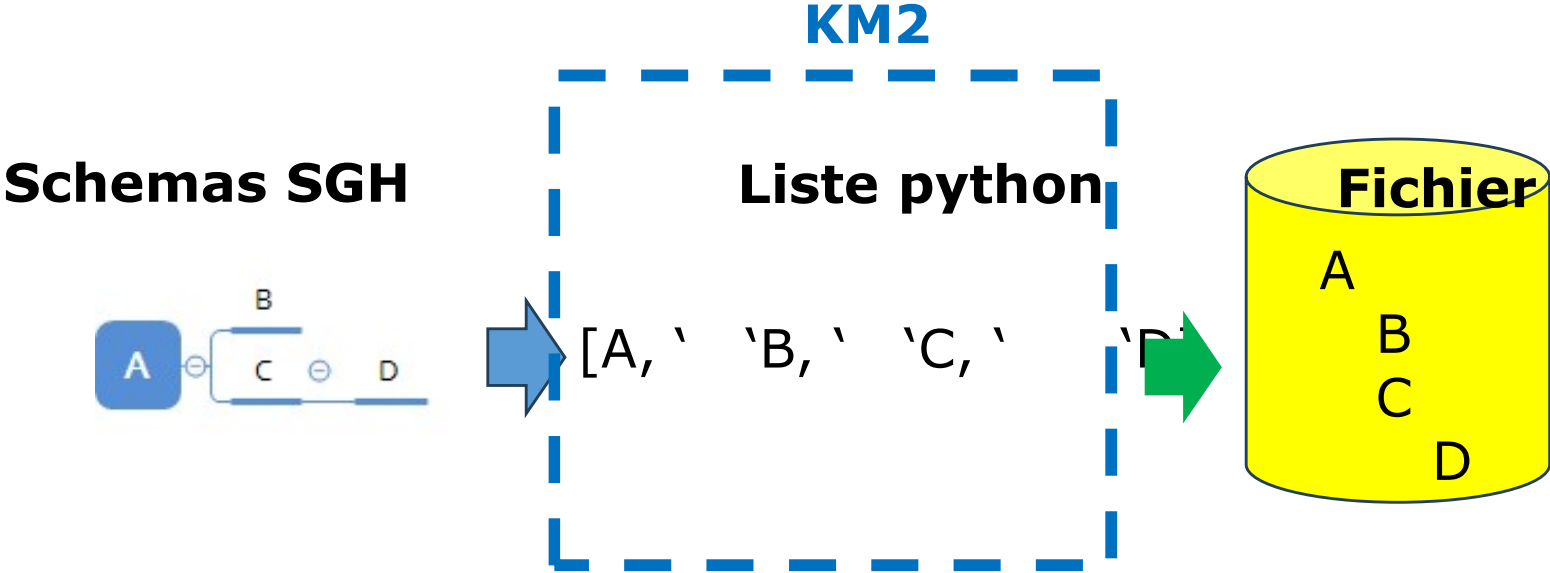
Insertion de la liste foo



Etapes de transformation : → plusieurs fichiers)



Etales de transformation (indentation)



SOURCE IN KM2

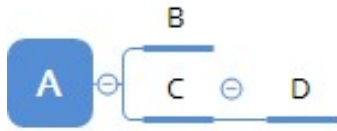


Generated by KM2 :

```
define foo (x)
  instruction 1
  instruction 2
    instruction 3
    instruction 4
  instruction 5
  instruction 6
```

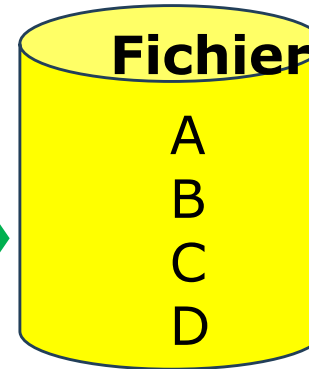
Etapes de transformation (base)

Schemas SGH



Liste python

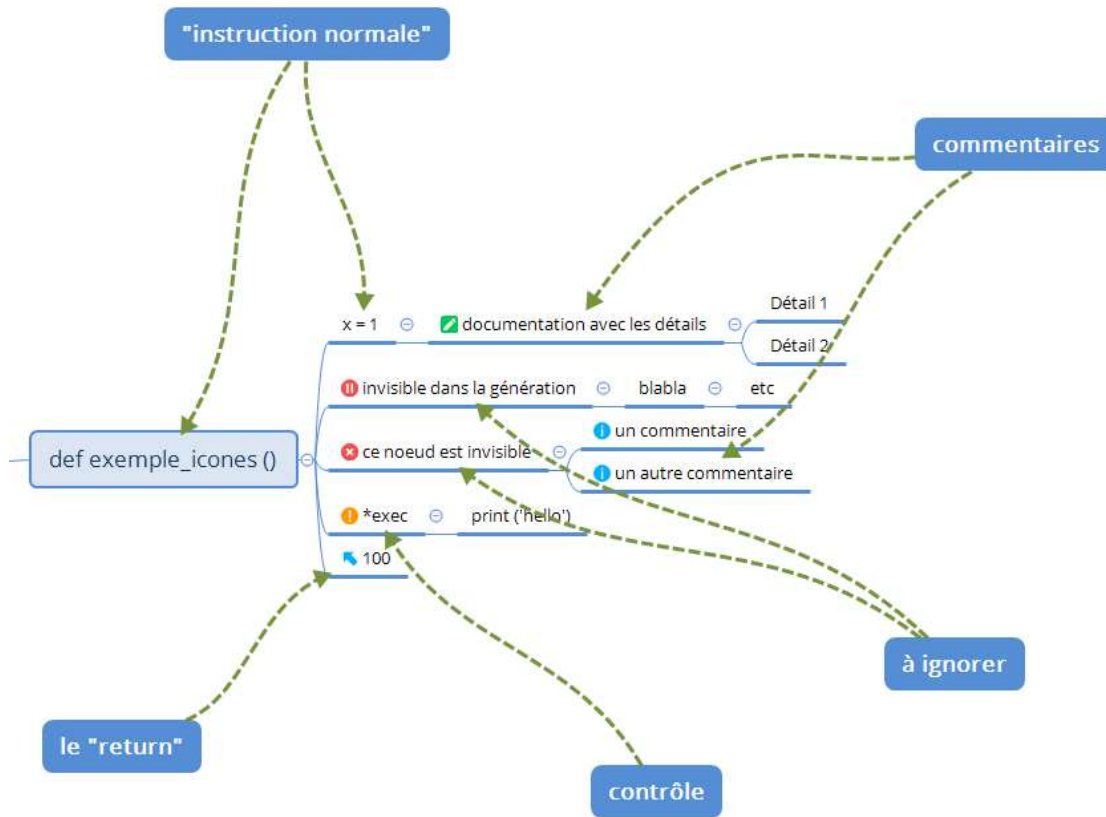
[A, B, C, D]



+ possibilités d'activer des fonctions python en direct ou via des macros → infini de possibilités

Macros et pilotage du processeur

Icones de contrôle (base)



```
def exemple_icons () :  
    x = 1  
    # documentation avec les détails  
    #     Détail 1  
    #     Détail 2  
    # un commentaire  
    # un autre commentaire  
    return 100
```

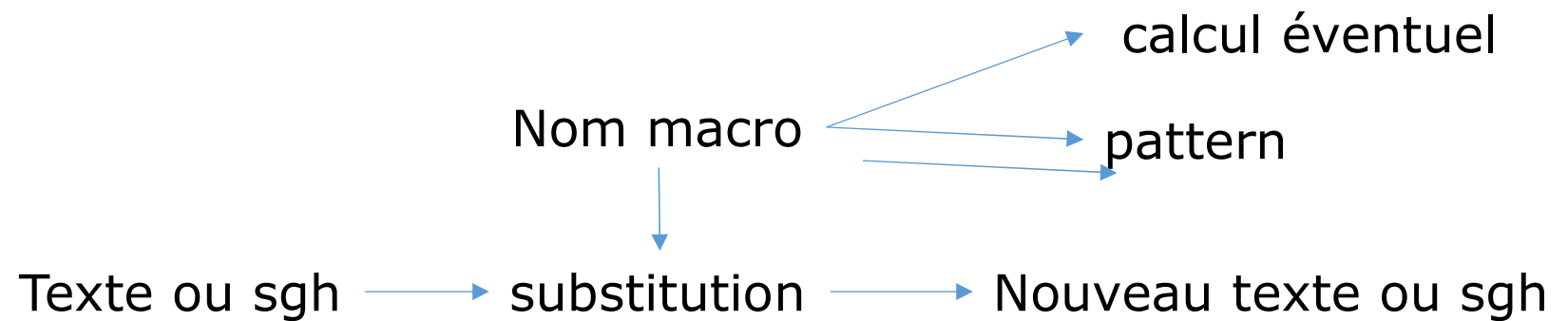
Exécution lors du traitement

Macros

Permettent de gérer la répétition de termes,
d'avoir une syntaxe alternative, etc

3 types :

- Textuelles
- Groupe de mots
- icones



Macros textuelles pour Javascript

```
@@ ::= document.getElementById({a1}).innerHTML
'@H' ::= document.getElementById({a1}).innerHTML
'@!' ::= document.getElementById({a1})
'@Id' ::= document.getElementById({a1})
'@V' ::= document.getElementById({a1}).value
notations basics for the DOM - '@C' ::= {a1}.childNodes
'@h' ::= {a1}.innerHTML
'@v' ::= {a1}.value
'@.' ::= 'document.'
'@.SI' ::= document.{a1}.selectedIndex
'@SI' ::= {a1}.selectedIndex
elements of type ::= document.getElementsByClassName({a1})
elements of style ::= document.querySelectorAll({a1})
```

Règles de réécriture, éventuellement modulables par programme

Macros icones



notation ⊖ 🛠 ::= ⊖ {a0} ⊖ {L}

print ("==> {a0}")

print ("<== {a0}")

SEP exemple de debug ⊖ def actions () ⊖

- 🛠 init ⊖ x = 0
- 🛠 calcul ⊖ for i in range (10) ⊖ x = x + i * i
- 🛠 impression ⊖ print ('x =', x)
- 🛠 fini pour actions ! ⊖ True

```
# >>> ===== exemple de debug =
def actions () :
    print ("==> init")
#     init
    x = 0
    print ("<== init")
    print ("==> calcul")
#     calcul
    for i in range (10) :
        x = x + i * i
    print ("<== calcul")
    print ("==> impression")
#     impression
    print ('x =', x)
    print ("<== impression")
    print ("==> fini pour actions !")
#     fini pour actions !
    return True
    print ("<== fini pour actions !")
# <<< ===== exemple de debug =
```



notation ⊖ 🛠 ::= ⊖ {a0} ⊖ {L}

print ("==> {a0}")

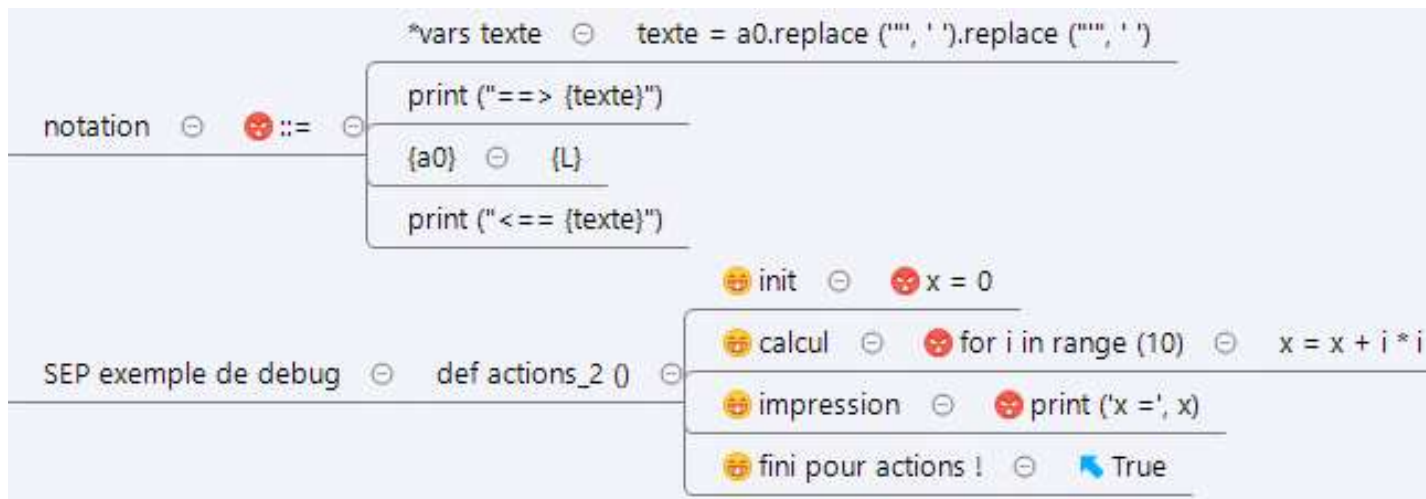
print ("<== {a0}")

SEP exemple de debug ⊖ def actions () ⊖

- 🛠 init ⊖ x = 0
- 🛠 calcul ⊖ for i in range (10) ⊖ x = x + i * i
- 🛠 impression ⊖ print ('x =', x)
- 🛠 fini pour actions ! ⊖ True

```
# >>> ===== exemple de debug =
def actions () :
#     init
    x = 0
#     calcul
    for i in range (10) :
        x = x + i * i
#     impression
    print ('x =', x)
#     fini pour actions !
    return True
# <<< ===== exemple de debug =
```

MACROS ICONES : exemple debug par instruction

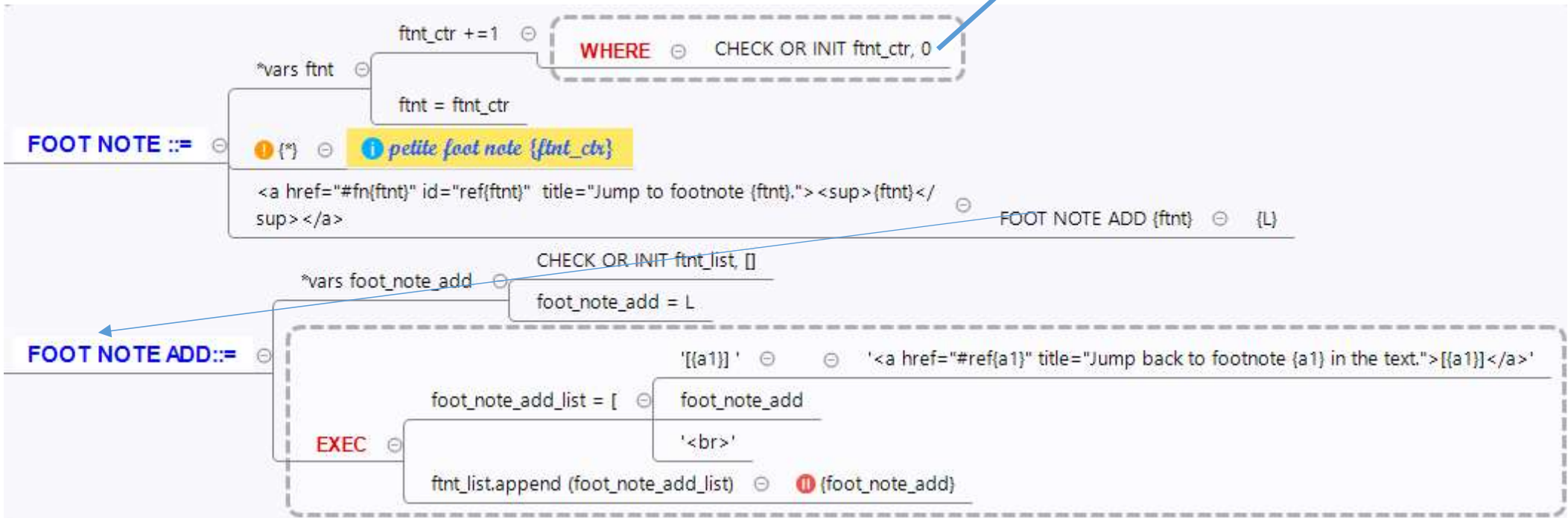
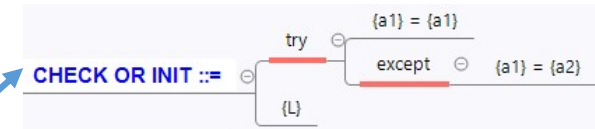


```
# >>> ===== exemple de debug ===
def actions_2 () :
#   init
  print ("==> x = 0")
  x = 0
  print ("<== x = 0")
#   calcul
  print ("==> for i in range (10)")
  for i in range (10) :
    x = x + i * i
  print ("<== for i in range (10)")
#   impression
  print ("==> print ( x = , x)")
  print ('x =', x)
  print ("<== print ( x = , x)")
#   fini pour actions !
  return True
# <<< ===== exemple de debug ===
```

Macros mots pour HTML

javascript

Exemple : gestion de footnotes en html



Html → fin de document
 Latex → bas de page

Contrôle du processeur

Name
*setf // *seta variable
*collect variable, lang = lang, indent = 4
*if, *else, *case, *switch condition
*loop // *for condition
*warn message
*flush variable, file_name
*insert file_name or link, from_, to_
*standard name
*import file link or file_name
*indent value
*inline ON/OFF
*controls show
*base64

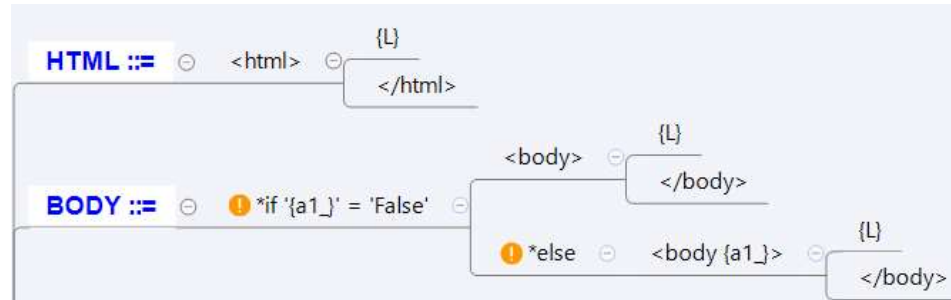
Name	Type of command	Usage
*setf // *seta variable	designation of tree	defines a variable
*collect variable, lang = lang, indent = 4	Control	builds the list of lines from the processed nodes
*if, *else, *case, *switch condition	Control	direct control of the processor
*loop // *for condition	Control	direct control of the processor
*warn message	Control for errors	reports an error
*flush variable, file_name	use external file	outputs the list into the file
*insert file_name or link, from_, to_	use external file	inserts lines from from_ to to_
*standard name	use external file	fetches form the STANDARDS file the "name", and processes on with the ssub_node
*import file link or file_name	use external file	fetch the correct tree from the file or the node
*indent value	parameter	sets the indent depth, as measured in tabs
*inline ON/OFF	parameter	allows functions to be embedded into each other, or must be separated
*controls show	parameter	shows all the controls . Used for debug.
*base64 ON/OFF	parameter	for images, creates a link to the file or inserts the data as base64 coding

Exemples

Génération de documents

Génération de html

Principe des macros :



SOURCE IN KM2



Generated by KM2 :

```
<html>
<head>
<style>
</style>
<!-- script : False -->
<script>
function foo (x) {
    return x;
}
</script>
</head>
<body>
hello world
</body>
</html>
```

1 tag html → 1 macro

→ 100 macros de base

+ 150 macros de niveau agrégé

Intégration du javascript et css

→ Compacité + lisibilité

Exemple html : macro de plus haut niveau

Insérer un bouton compteur dans une page

Définition et mise en page

```
LARGE ⊖ CENTER ⊖ BLUE ⊖ DEF INCR ctr_A ⊖ DEF INCR ctr_B ⊖ DEF INCR ctr_C ⊖ DEF INCR ctr_D
```

Affichage initial

```
ctr_A + 0 ctr_B + 0 ctr_C + 0 ctr_D + 0
```

Après quelques clics

```
ctr_A + 5 ctr_B + 2 ctr_C + 0 ctr_D + 10
```

```
notation DEF INCR ::=  
  SCRIPT ⊖ global value of '{a1}' = 0  
  ONLY ONCE def incr ctr function ⊖ SCRIPT ⊖ def incr_named_ctr (ctr_name) ⊖  
    let v = 'CTR_' + ctr_name ⊖ echo v ⊖ echo global value of ctr_name  
    global value of ctr_name += 1 ⊖ @H v = global value of ctr_name ⊖ global value of ctr_name  
  BOUTON {a1} +, incr_named_ctr ('{a1}') ⊖ ID CLASS CTR_{a1}, simple_counter ⊖ 0  
  {L}
```

Génération de CSS : principe

Le CSS est décrit dans des tables, de façon organisée.
Des mots clés (Selector, EXTEND) permettent des macros de plus haut niveau

SOURCE IN KM2

Selector is built from the css definition : selector {properties:values}.

Selector	Colors	Position	Sizes	Other	Note
Name	color of text color of backgrounds font description	float text align position	paddings, etc font size	etc	some documentation
#navbar	background-color: \$bg_color_1 color:\$text_color		font-size: 16px;	overflow: hidden;	
#navbar a	color:\$text_color text-decoration: none; background-color: \$blanc	float: left; text-align: center;	padding: 8px 16px; font-size: 16px;	display: block;	
#navbar a:hover	background-color: \$bg_color_2 color:\$text_color_inv				

Génération de CSS: exemple

<code>.dropdown</code>		<code>float: left;</code>	<code>font-size: 16px;</code>	<code>overflow: hidden;</code>
<code>.dropdown .dropbtn</code>	<code>background-color: \$bg_color_1</code> <code>border: none;</code> <code>color: \$text_color</code> <code>font: inherit;</code>		<code>padding: 8px;</code> <code>margin: 0;</code> <code>font-size: 16px;</code>	<code>outline: none;</code>
<code>.navbar a:hover, .dropdown:hover .dropbtn</code>	<code>background-color: \$bg_color_2</code>			
<code>.dropdown-content</code>	<code>background-color: \$bg_color_1</code> <code>box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);</code>	<code>position: absolute;</code> <code>left: 0;</code> <code>z-index: 1;</code>	<code>width: 100%;</code>	<code>display: none;</code>
<code>.dropdown-content .MenuEntete</code>	<code>background-color: \$bg_color_1</code> <code>color: \$text_color_inv</code>		<code>padding: 8px;</code>	
<code>.dropdown:hover .dropdown-content</code>				<code>display: block;</code>
<code>.foo</code>	EXTEND .dropdown <code>text-decoration: none;</code>			

```
}.foo .dropdown {
  font-size: 16px;
  float: left;
  overflow: hidden;
}
}.foo .dropdown .dropbtn {
  background-color: gold;
  border: none;
  color: black;
  font: inherit;
  padding: 8px;
  margin: 0;
  font-size: 16px;
  outline: none;
}
}.navbar a:hover, .dropdown:hover .dropbtn {
  background-color: var(--bg_color_2);
}
}.dropdown-content {
  background-color: gold;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  width: 100%;
  position: absolute;
  left: 0;
  z-index: 1;
  display: none;
}
}.dropdown-content .MenuEntete {
  background-color: gold;
  color: var(--text_color_inv);
  padding: 8px;
}
}.dropdown:hover .dropdown-content {
  display: block;
}
}
}.foo {
  /* inherits from .dropdown*/
  text-decoration: none;
}
```

Macro pour l'héritage

Exemple : styles pour documentation KM2

The image displays a CSS documentation tool interface. On the left, a tree view shows a hierarchy of styles: `navbar_km2`, `dropdown_km2`, `column_km2`, `row_km2`, and `sticky_km2`. A central box labeled "TEST CSS" is connected to the tree view. On the right, a table lists the styles with their corresponding CSS rules and properties.

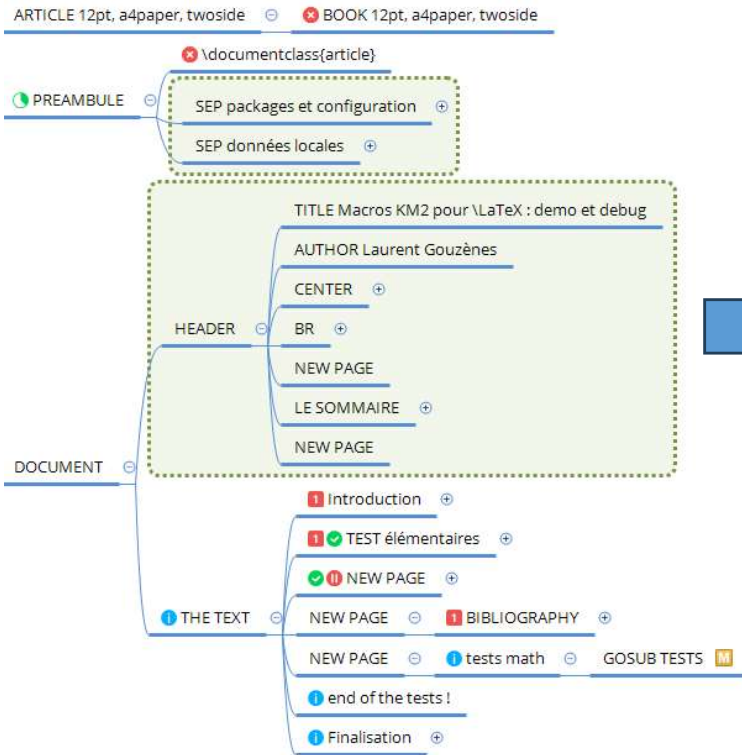
Selector	Code	Position	Size	Other	How
<code>h1</code>	<code>color: #000;</code> <code>color: #000;</code> <code>font: #000;</code> <code>border: #000;</code>	float	padding: 0;	etc	some documentation
<code>#navbar</code>	<code>background-color: #b3cde3;</code> <code>font: #000;</code> <code>color: #000;</code>	float: left;	font-size: 10px;	padding: 0 10px;	overflow: hidden;
<code>#navbar a</code>	<code>color: #000;</code> <code>font: #000;</code> <code>background-color: #b3cde3;</code>	float: left;	padding: 0 10px;	display: block;	
<code>#navbar a:hover</code>	<code>color: #000;</code> <code>background-color: #b3cde3;</code>				
<code>#navbar a:active</code>	<code>color: #000;</code> <code>background-color: #b3cde3;</code>				
<code>ul#mega-menu *</code>	<code>background-color: #b3cde3;</code>		font-size: 10px;		mega menu
<code>.navbar</code>	<code>font-family: Arial, Helvetica, sans-serif;</code>	float: left;	font-size: 10px;	text-decoration: none;	
<code>.navbar a</code>	<code>color: #000;</code>	float: left;	font-size: 10px;	padding: 0 10px;	
<code>.dropdown</code>	<code>background-color: #b3cde3;</code>	float: left;	font-size: 10px;	padding: 0 10px;	overflow: hidden;
<code>.dropdown .dropdown</code>	<code>border: none;</code> <code>color: #000;</code> <code>font: inherit;</code>			padding: 0 10px;	outline: none;
<code>.navbar a:hover, .dropdown:over .dropdown</code>	<code>background-color: #b3cde3;</code>				
<code>.dropdown-content</code>	<code>background-color: #b3cde3;</code> <code>border: none;</code> <code>border: none;</code> <code>margin: 0;</code> <code>font: inherit;</code>	position: absolute;	width: 100%;	display: none;	
<code>.dropdown-content .MenuTitle</code>	<code>background-color: #b3cde3;</code> <code>color: #000;</code>			padding: 0 10px;	
<code>.dropdown:over .dropdown-content</code>					display: block;
<code>foo</code>	<code>EXTEND .dropdown</code> <code>text-decoration: none;</code>				
<code>.column</code>	<code>background-color: #b3cde3;</code>	float: left;	width: 30%;	padding: 0 10px;	seul de la colonne entière
<code>.column a</code>	<code>color: #000;</code> <code>text-decoration: none;</code>	float: none;		padding: 0 10px;	display: block;
<code>.column a:over</code>	<code>background-color: #b3cde3;</code>				
<code>.row:after</code>					display: table;
<code>.sticky</code>		position: fixed;	width: 100%;	top: 0;	padding: 0 10px;
<code>.cs_short</code>	<code>align: center;</code> <code>SHORT text</code> <code>transform: lowercase;</code> <code>overflow: hidden;</code> <code>background-color: #b3cde3;</code>				
<code>div.menu:over</code>		position: sticky;	top: 10px;	width: 100%;	

Génération de latex/pdf

Schémas sgh → Fichier .tex → Fichier .pdf



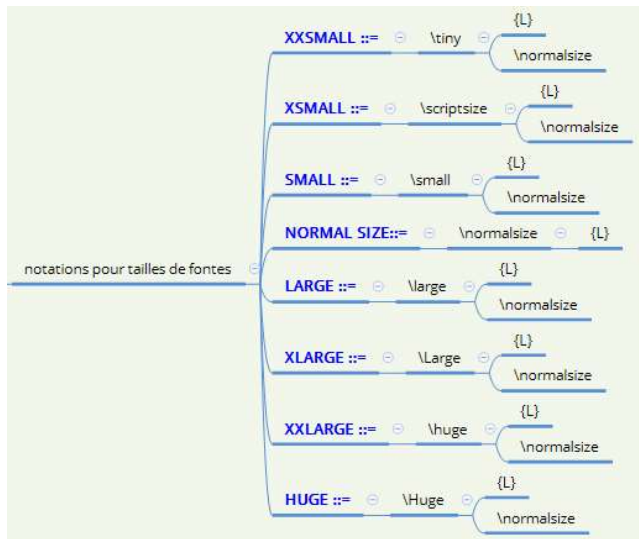
Obscur !



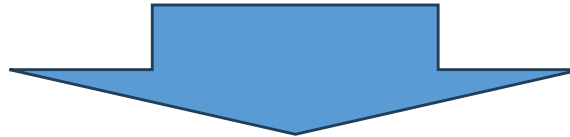
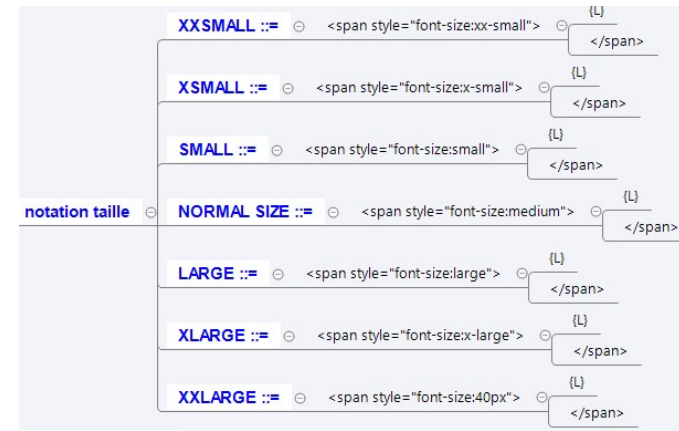
```
\usepackage{geometry}
\geometry{a4paper,margin=2.5cm}
% href et url s
\usepackage{hyperref}
\hypersetup{
  colorlinks=true,
  linkcolor=blue,
  filecolor=magenta,
  urlcolor=cyan,
  pdftitle={KM2 Example}}
\urlstyle{same}
% usage des figures
\usepackage[rightcaption]{sidecap}
\usepackage{wrapfig}
% pour générer des textes aléatoires
\usepackage{blindtext}
\usepackage{verbatim}
\usepackage{lipsum}
% numérotation des indentations en i.j.k - 4 niveaux max
\renewcommand{\labelenumii}{\arabic{enumi}.\arabic{enumii}}
\renewcommand{\labelenumiii}{\arabic{enumi}.\arabic{enumii}.\arabic{enumiii}}
\renewcommand{\labelenumiv}{\arabic{enumi}.\arabic{enumii}.\arabic{enumiii}.\arabic{enumiv}}
% gestion des erreurs
% cas 2
\vbaddness=10000
\hbadness=10000
\hfuzz=10pt
\vfuzz=10pt
% # <<< ===== packages et configuration =====
% # >>> ===== données locales =====
\graphicspath{{"C:/Users/lgouzenes.PACTENOVATION/Dropbox (Compte personnel)/KM3 NEW STUFF/MACROS KM2/XMIND/IMAGES/"}}
% # <<< ===== données locales =====
% # <<< ===== PREAMBULE =====
% # >>> ===== DOCUMENT =====
\begin{document}
\title{Macros KM2 pour \LaTeX : demo et debug}
\makeatletter
```



Génération de latex/pdf



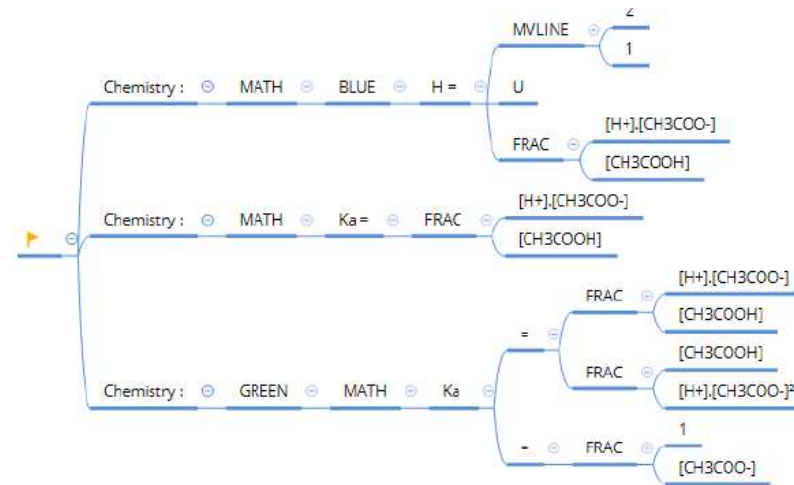
Mêmes macros qu'en html



Génération indifférenciée vers html ou latex

~100+ macros de base et de plus haut niveau

Example Latex



latex

html

- $H = \frac{2}{1} U \frac{[H+].[CH_3COO^-]}{[CH_3COOH]}$

- $Ka = \frac{[H+].[CH_3COO^-]}{[CH_3COOH]}$

- $Ka = \frac{[H+].[CH_3COO^-]}{[CH_3COOH]} \frac{[CH_3COOH]}{[H+].[CH_3COO^-]^2} = \frac{1}{[CH_3COO^-]}$

- $H = \frac{2}{1} U \frac{[H+].[CH_3COO^-]}{[CH_3COOH]}$

- $Ka = \frac{[H+].[CH_3COO^-]}{[CH_3COOH]}$

- $Ka = \frac{[H+].[CH_3COO^-]}{[CH_3COOH]} \frac{[CH_3COOH]}{[H+].[CH_3COO^-]^2} = \frac{1}{[CH_3COO^-]}$

Exemples Latex / génération de pdf

- maths equations : $Y = \frac{1}{(\frac{2}{x^{3/2}})} + \int_0^{\infty} e^{-st} h(t) dt$

- maths equations : $x = \frac{-b \pm \sqrt{(b^2 - 4ac)}}{2a} \sqrt{(b^2 - 4ac)}$

$$; U_{235} + H_4^{2-} = Pu_{239}$$

Generated by KM2 :

- simple vector

- vecteur : $\begin{pmatrix} A+200*x \\ B \\ C \end{pmatrix}$

- 1 colonne $\begin{matrix} A+200*x \\ B \\ C \end{matrix}$

- Simple matrices

- Matrice avec line: $\begin{pmatrix} A+200*x & B & C \\ A & B & C \end{pmatrix}$

- Matrice avec simple table : $\begin{pmatrix} A & B & C \\ A+200*x & 2 & 3 \\ X & Y & Z \end{pmatrix}$

- Determinant avec line : $D = \begin{vmatrix} A & B+200*x & C \\ A & B & C \\ A & 2B & 3C \end{vmatrix}$

- Determinant 2 avec simple table: $D = \begin{vmatrix} A & B & C \\ A+200*x & 2 & 3 \\ X & Y & Z \end{vmatrix}$

- Some matrix equations

- équation : $Xa = \begin{pmatrix} A+200*x \\ B \\ C \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = Z$

2.8.2 tableau de codes divers

du texte	du code
some text	<pre>def foo (n) : if n < 1 : return 1 return n * fact (n-1)</pre>
some more text	<pre>def bar (n) : if n < 1 : return 1 return n * fact (n-1)</pre>

test de table dans la table 4 :

cell1 dummy text a b c d e f g h i dummy text dummy text	cell2	cell3			
cell1 dummy text dummy text dummy text	cell5	cell5			
cell7	cell8	value1	value2	value 3	
		value4	value5	value6	
		value7	value8	hello	

Génération de fichiers Excel complexes

The VBA editor shows the following code:

```

    add_item ('Rent', 1000)
    add_item ('Gas', 100)
    add_item ('Food', 300)
    add_item ('Gym', 50)
    start
    fabrication d'un tableau
    formule 1 explicite
    for item, cost in (expenses)
    calcul des sommes du tableau
    BR
    BOLD
    add_item ('Total', 0)
    entourage du tableau par un quadrillage
    for i in range (3)
    ROUGE
    BR
    add_item ('hello-' + str(i))
    add_item ('item', 'cost', style = bold1)
    TABLE
    EXTRACT TABLE item, cost
    by_item
    FORMAT
    nouveau tableau, juste les infos
    BR
    test des names
    BR
    add_item ('Exchange_rate', 0.96)
    OLD
    BR
    add_item ('exemple:', '=Exchange_rate')
    BR
    BR
    add_item ('exemple:', '=Exchange_rate')
    workbook.define_name('Sales', '=S$1:$G$10')
    workbook.define_name('Exchange_rate', '=0.96')
    BR
    add_item ('Exchange Rate', 0.96)
    add_item ('=Exchange_rate + 3')
    set_width (0, 30)
    test des fonctions VBA
    pour tests VBA
    BR
    add_item ('20/02/1959')
    add_f ("=age@LEFT")
  
```

The spreadsheet content includes the following tables:

item	cost
Rent	1000
Gas	100
Food	300
Gym	50
Total	0

item	cost
pomme	1
poire	2

Exchange Rate	0,96	3,96					
Données	0	1	2	3	4	5	6
idem	0	1	2	3	4	5	6
cumul double	0	2	6	12	20	30	42
equation	0	1010	2040	3100	4200	5350	6560

TOTO	5
la constante vaut	5
EXEMPLE	1000
la somme vaut	2005

TUTU	TITI	TATA
TUTU	0	0
TITI	225	196
TATA	0	3
TUTU	0	6
TUTU	0	9
TUTU	0	12
TUTU	0	15
TUTU	0	18
FORMULES :	1305	12

Code VBA

```

' le but est de tester différentes possibilités de générer rapidement du code VBA à :
' =====> doublons
Function doublons (plage As Range, valeur As Integer) As Integer
    DESCRIPTION
    ' Compte le nombre d'apparitions de valeur dans plage
    ' =====Variables locales
    ' cellule pour parcourir le range :
    Dim cellule As Range
    ' décompte du nombre de doublons :
    Dim frequence As Integer
    ' =====
    frequence = 0
    For Each cellule in plage
        If (cellule.value = valeur) Then
            ' ici on doit indenter
            frequence = frequence + 1
        Else
            frequence = frequence + 0
        End If
    Next cellule
    doublons = frequence
End Function
' <===== doublons
' =====> age
Function age (la_date As Date) As Integer
    DESCRIPTION
    ' calcule le nombre d'années entières jusqu'à maintenant
    ' OUTPUT
    ' age en années
    age = Int((Date - la_date) / 365.25)
End Function
' <===== age
' =====> ttc
Function ttc (montant_ht As Single, taux_tva As Single) As Double
    DESCRIPTION
    ' calcule le prix en ttc à partir du prix en HT et du taux de la tva
    ' OUTPUT
    ' ttc en double
    ttc = montant_ht * (1 + taux_tva)
End Function
' <===== ttc
  
```

S'appuie sur la bibliothèque python xlsxwriter

Code VBA

Excel



Génération de documents .txt

Objectif : compatibilité max avec html et latex

The screenshot displays a SGH editor interface with several macros and their rendered output:

- TITRE**: exemple de texte simple
- CENTER**: Laurent Gouzènes
- SHOW LINE WIDTH**: LINE WIDTH 80
- ABSTRACT**: Le but de ce document est de montrer les possibilités de génération de fichiers en .txt à partir de graphes SGH avec l'outil KM2. Malgré les limitations de mode txt, on arrive à un document présentable !
- SAUT**: SAUT
- BR**: SOMMAIRE_
- SILENTLY DO**: Initialisation de la base de données de références
- Table**: un exemple de table
- Table**: une table
- BIB INTERN TABLE**: A table with columns: Ref, Author, Comment, Page, Source, Title.
- Mini-tests**: par défaut va toujours à la ligne, CAP, ceci est une phrase, très longue, en capitales, ligne 2, ligne 3, pour petit test.
- On peut numéroter des paragraphes**: aussi hiérarchiquement, et très hiérarchiquement, voilà !, ou aussi, Voici un peu de texte : plusieurs paragraphes à la suite. Et encore une autre phrase, c'est enfin fini !, voilà, et très hiérarchiquement, avec des détails supplémentaires, un exemple d'hyperlien, et une autre hiérarchie, et très hiérarchiquement, FOOT NOTE, 1* foot note, et très hiérarchiquement, FOOT NOTE, 2* foot note, et très hiérarchiquement, voilà !.

Mêmes icones et macros



Génération de documents .txt

Objectif : compatibilité max avec html et latex

```
EXEMPLE DE TEXTE SIMPLE
Laurent Gouzènes

Line width = 80      Mise à jour : 20 Jan 2025 @ 15:21:14

                ABSTRACT
Le but de ce document est de montrer les
possibilités de génération de fichiers en .txt à
partir de graphes SGH avec l'outil KM2. Malgré les
limitations de mode txt, on arrive à un document
présentable !

=====
SOMMAIRE
1 Mini-tests
2 On peut numéroter des paragraphes
2.1 aussi hiérarchiquement
2.2.1 et très hiérarchiquement
2.2 ou aussi
2.3 et une autre hiérarchie
2.4.2 et très hiérarchiquement
2.4.3 et très hiérarchiquement
2.4.4 et très hiérarchiquement
3 Exemples d'utilisation
4 Conclusion
Foot notes
Bibliographie citée
Bibliographie complète
=====

1 Mini-tests
=====
Line width = 80

par défaut va toujours à la ligne
Ceci Est Une Phrase Très Longue En Capitales
ligne 2
ligne 3 : txt ceci est une phrase très longue en capitales ceci est une phrase
très longue en capitales ceci est une phrase très longue en capitales SUIVI PAR
ceci est une phrase très longue en capitales ceci est une phrase très longue en
capitales ceci est une phrase très longue en capitales

2 On peut numéroter des paragraphes
=====

2.1 aussi hiérarchiquement
-----

2.2.1 et très hiérarchiquement
voilà !

2.2 ou aussi
-----
Voici un peu de texte : plusieurs paragraphes à la suite. Et encore une autre
phrase, c'est enfin fini !

2.3 et une autre hiérarchie
-----
voilà
```

→ Entête avec date auto

→ Abstract

→ Sommaire automatique

→ Numérotation
automatique des §

→ hiérarchie automatique
des §



Génération de document .txt

```
2.4.2 et très hiérarchiquement
voilà ![cf foot note 1]

avec des détails supplémentaires
un exemple d'hyperlien (http://google.com)

2.4.3 et très hiérarchiquement
voilà ![cf foot note 2]

2.4.4 et très hiérarchiquement
voilà !

3 Exemples d'utilisation
=====
Line width = 80
on peut faire des bullet points :
    * A
    * B
    * C Le but de ce document est de montrer les possibilités de génération de
        fichiers en .txt à partir de graphes SGH avec l'outil KM2. Malgré les
        limitations de mode txt, on arrive à un document présentable !
Line width = 80
Mais ces bullet points sont sans indentation intérieure.[BOB-24-2]
Line width = 80

4 Conclusion
=====
etc
un exemple d'hyperlien (http://google.com)

FOOT NOTES
=====
[1] in [2.4.2 et très hiérarchiquement] :
1° foot note
[2] in [2.4.3 et très hiérarchiquement] :
2° foot note :
    * avec des détails supplémentaires un exemple d'hyperlien
    * avec des détails supplémentaires un exemple d'hyperlien
    * avec des détails supplémentaires un exemple d'hyperlien

BIBLIOGRAPHIE CITEE
[BOB-24-2] : BOB l'éponge , KM3 : spongiuous writing , (2021) ( Web )
  An introduction to sponge schemes

BIBLIOGRAPHIE COMPLETE :
[BOB-24-2] : BOB l'éponge , KM3 : spongiuous writing , (2021) ( Web )
  An introduction to sponge schemes
[LG-22-1] : L GOUZENES , KM2 : non linear writing and programing , (2021) ( Web
)
  An introduction to SGH schemes

=====
SOMMAIRE
1 Mini-tests
2 On peut numéroter des paragraphes
```

→ Notes de bas de page

→ hyperliens

→ Notes de bas de page

→ Bibliographie
citée/ existante

→ 2° sommaire



Génération de programmes

S'affranchir des détails de la syntaxe

Example for Python :

SOURCE IN KM2



Generated by KM2 :

```
def fact (n) :  
# DESCRIPTION  
# returns the factorial of any number  
if n < 2 :  
    return 1  
else :  
    return n * fact (n-1)
```

Example for javascript :

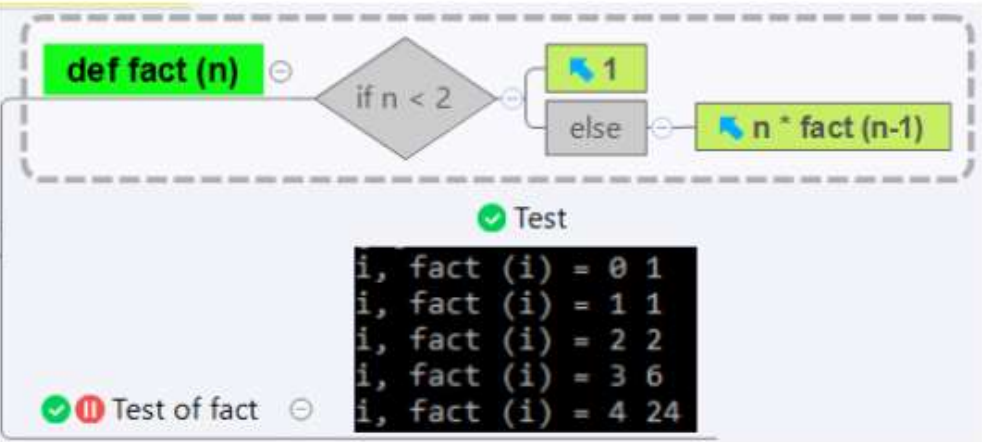
SOURCE IN KM2



Generated by KM2 :

```
function fact (n) {  
// DESCRIPTION  
// returns the factorial of any number  
if (n < 2) {  
    return 1;  
}  
else {  
    return n * fact (n-1);  
}  
}
```

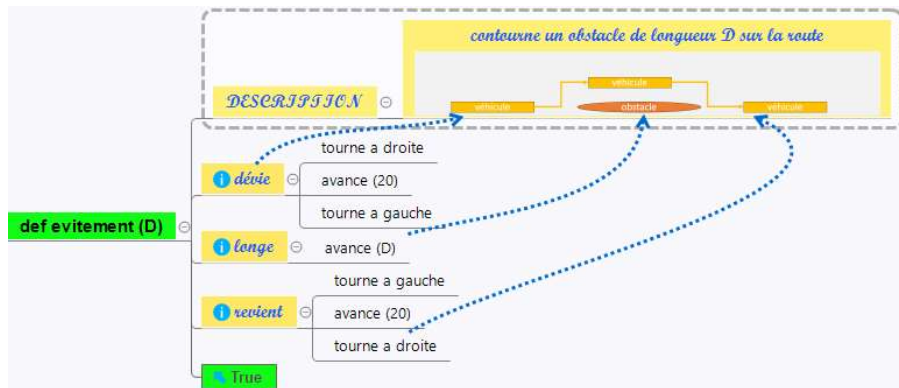
Programmation non linéaire // matricielle

Code with SGH notations	Code as a matrix :
 <p>Programms are defined and edited as trees, using also a rich set of graphic attributes and hyperlinks The development process is easily tractable.</p>	<pre>def fact (n) : # Computes "fact (n)" as "n!" - as classically defined if n < 2 : return 1 else : return n * fact (n-1)</pre>
	<p>The classical editing of computer programmes and documentation is imposed by the matrix structure of programs in the editor. In this case, there is no way to embed a screen copy into a program.</p>

Programmation intrinsèquement structurée / hiérarchisée

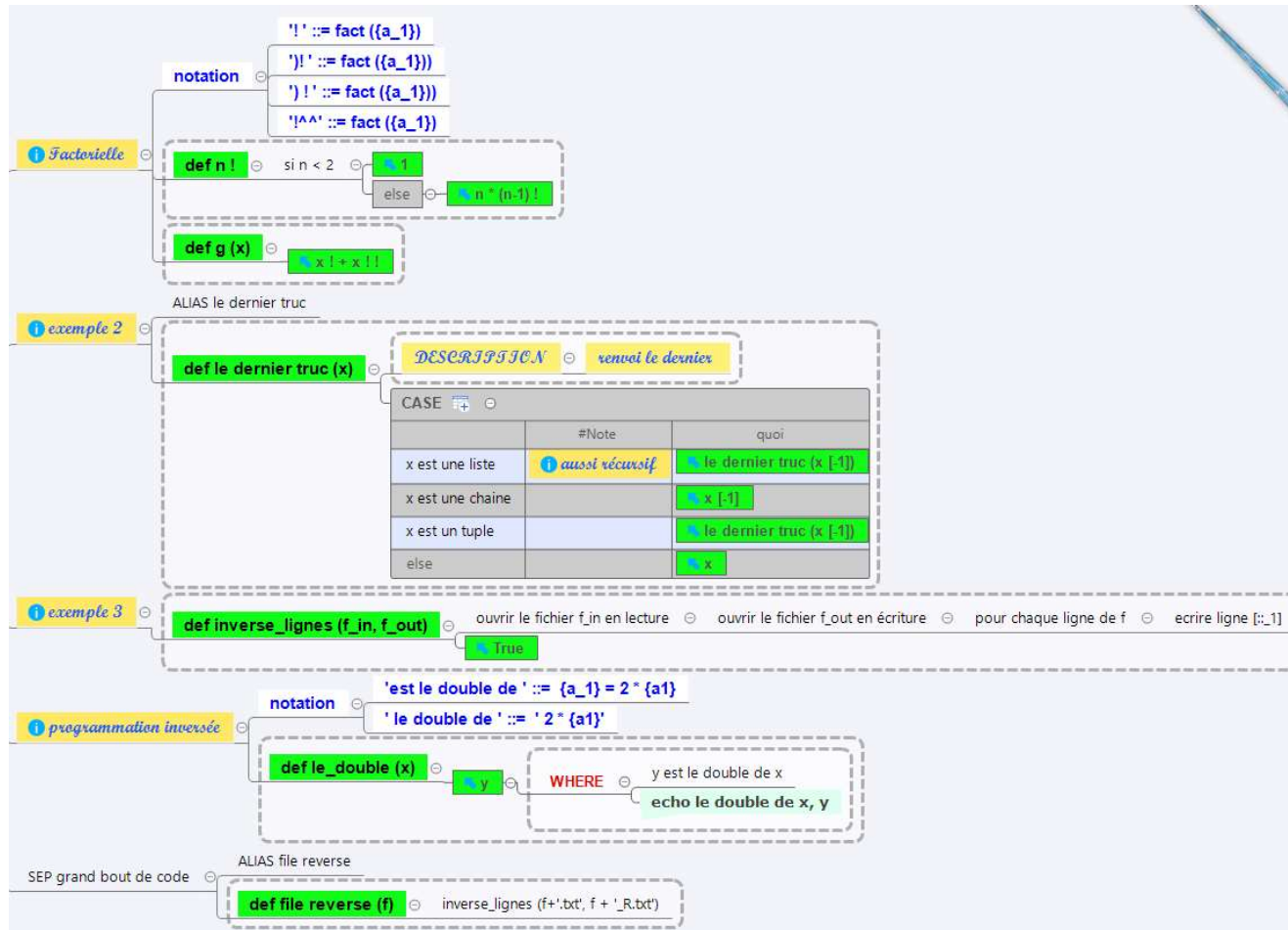
Génération de python

- Les macros KM2 sont étendues par la programmation en python
 - → il était indispensable de générer du python à la volée dans les macros



```
def evitement (D) :  
    ...  
    contourne un obstacl  
    ...  
#   dévie  
#   tourne_a_droite ()  
#   avance (20)  
#   tourne_a_gauche ()  
#   longe  
#   avance (D)  
#   revient  
#   tourne_a_gauche ()  
#   avance (20)  
#   tourne_a_droite ()  
#   return True
```


Une nouvelle génération de macros



```
#Factorielle
def fact (n) :
    if n < 2 :
        return 1
    else :
        return n *fact ( (n-1) )
def g (x) :
    return fact (x)+ fact (fact (x))
#exemple 2
def le_dernier_truc (x) :
    ...
    --> le_dernier_truc (x) :
    renvoi le dernier
    ...
    if isinstance (x, list) :
#        aussi récursif
        return le_dernier_truc (x [-1])
    elif isinstance (x, str) :
        return x [-1]
    elif isinstance (x, tuple) :
        return le_dernier_truc (x [-1])
    else :
        return x
#exemple 3
def inverse_lignes (f_in, f_out) :
    with open (f_in, 'r', encoding = 'utf8') as f_in :
        with open (f_out, 'w', encoding = 'utf8') as f_out :
            for ligne in f_in :
                f_out.write (ligne [::-1])
    return True
#programmation inversée
def le_double (x) :
    y = 2 * x
    print ( 'le double de x, y =',le double de x, y)
    return y
# >>> ===== grand bout de code =====
def file_reverse (f) :
    inverse_lignes (f+'.txt', f+'_R.txt')
# <<< ===== grand bout de code =====
#MAIN
if __name__ == "__main__" :
    for i in range (10) :
        print ( 'i, fact (i) =',i, i !)
```

Génération de javascript

```
'type of' ::= typeof ({a_1})
```

notations types FR

- 'est une liste' ::= Array.isArray({a_1})
- " n'est pas une liste" ::= ! Array.isArray({a_1})'
- est une chaîne ::= isString ({a_1})
- n'est pas une chaîne ::= ! isString ({a_1})
- est un float ::= isFloat ({a_1})
- n'est pas un float ::= ! isFloat ({a_1})
- est un dict ::= isDict ({a_1})
- n'est pas un dict ::= ! isDict ({a_1})
- est un nombre ::= isNumber ({a_1})
- n'est pas un nombre ::= ! isNumber ({a_1})

notations de type

- is a string ::= isString ({a_1})
- is not a string ::= ! isString ({a_1})
- is a float ::= isFloat ({a_1})
- is not a float ::= ! isFloat ({a_1})
- is a dict ::= isDict ({a_1})
- is not a dict ::= ! isDict ({a_1})
- is a number ::= isNumber ({a_1})
- is not a number ::= ! isNumber ({a_1})
- is an array ::= isArray ({a_1})
- is not an array ::= ! isArray ({a_1})
- is a list ::= isArray ({a_1})
- is not a list ::= ! isArray ({a_1})

notations type EN

- is a string ::= isString ({a_1})
- is not a string ::= ! isString ({a_1})
- is a float ::= isFloat ({a_1})
- is not a float ::= ! isFloat ({a_1})
- is a dict ::= isDict ({a_1})
- is not a dict ::= ! isDict ({a_1})
- is a number ::= isNumber ({a_1})
- is not a number ::= ! isNumber ({a_1})
- is an array ::= isArray ({a_1})
- is not an array ::= ! isArray ({a_1})
- is a list ::= isArray ({a_1})
- is not a list ::= ! isArray ({a_1})

```
".append (" ::= ".push ("
```

```
".append (" ::= ".push ("
```

pass ::= (L)

notation pour contrôle de flot

```
pass ::= (L)
```

```
except ::= catch (L)
```

```
is in string ::= ({a_1}.indexOf({a_1}) > -1)
```

```
is not in string ::= ({a_1}.indexOf ({a_1}) = -1)
```

```
starts with ::= {a_1} [0] == {a1}
```

```
ends with ::= {a_1}.slice (-1) == {a1}
```

```
est dans la chaîne ::= ({a1}.indexOf({a_1}) > -1)
```

```
n'est dans la chaîne ::= ({a1}.indexOf ({a_1}) = -1)
```

```
commence par ::= {a_1} [0] == {a1}
```

```
finit par ::= {a_1}.slice (-1) == {a1}
```

```
strip all ::= {a1}.map(e => e.trim())
```

notation

- notations string EN
- notations string FR

notation pour globales

```
global value of ::= window [{a1}]
```

```
valeur globale de ::= window [{a1}]
```

notations pour les listes

```
[:] ::= {a_1}.slice()
```

```
[1:] ::= {a_1}.slice(1)
```

```
[2:] ::= {a_1}.slice(2)
```

```
[:-1] ::= {a_1}.slice(0, -1)
```

```
[:-2] ::= {a_1}.slice(0,-2)
```

```
[1:-1] ::= {a_1}.slice(1,-1)
```

```
[-1] ::= {a_1}.at(-1)
```

```
[-2] ::= {a_1}.at(-2)
```

```
'last of' ::= {a1}.at(-1)
```

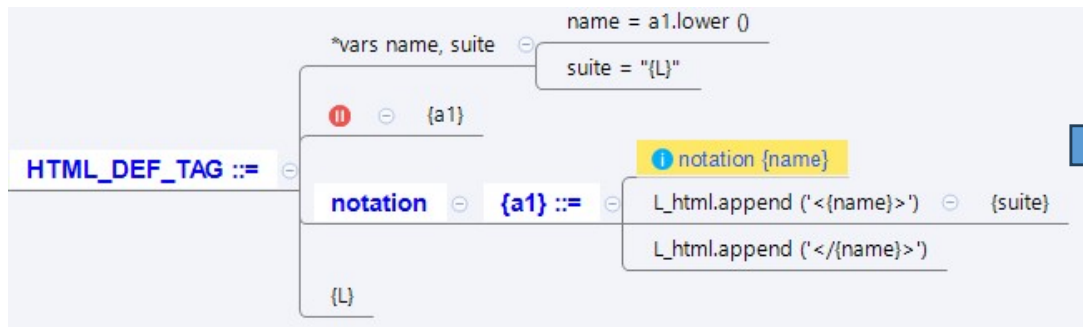
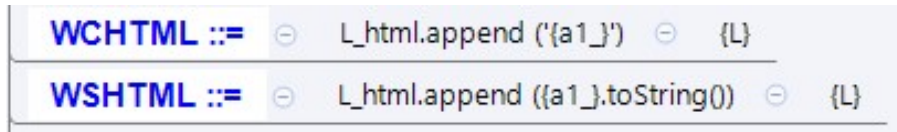
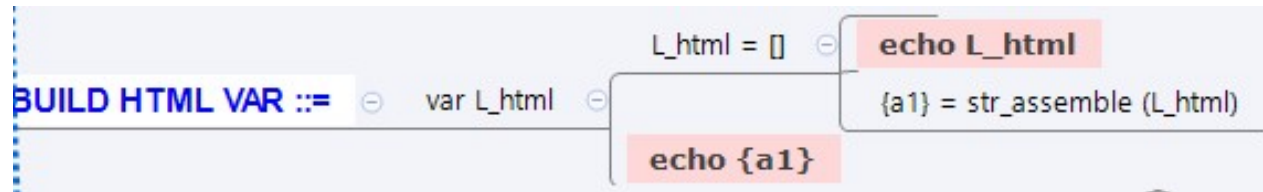
```
'first of' ::= {a1}[0]
```

Génération de javascript pour HTML

notation pour DOM HTML	notations basics for the DOM	<code>@@ ::= document.getElementById({a1}).innerHTML</code> <code>'@H' ::= document.getElementById({a1}).innerHTML</code> <code>'@h' ::= {a1}.innerHTML</code> <code>'@!' ::= document.getElementById({a1})</code> <code>'@Id' ::= document.getElementById({a1})</code> <code>'@V' ::= document.getElementById({a1}).value</code> <code>'@v' ::= {a1}.value</code> <code>'@C' ::= {a1}.childNodes</code> <code>'@.' ::= 'document.'</code> <code>'@.SI' ::= document.{a1}.selectedIndex</code> <code>'@SI' ::= {a1}.selectedIndex</code>
	notations more for DOM	<code>'elements of class' ::= document.getElementsByClassName({a1})</code> <code>'elements of type' ::= document.getElementsByClassName({a1})</code> <code>'elements with tag name' ::= document.getElementsByTagName({a1})</code> <code>'elements of style' ::= document.querySelectorAll({a1})</code> <code>'all links' ::= Array.from(document.links)</code> <code>toggler ::= ⊕</code> <code>for cle in collection ::= ⊕</code> <code>ASSOC ::= ⊕</code> <code>'collection to list' ::= Array.from({a1})</code>













notation pour clipboard	copy to clipboard ::= navigator.clipboard.writeText({a1}) {L}
	paste from clipboard ::= navigator.clipboard.readText()

Génération de javascript complexe pour HTML



- HTML_DEF_TAG HTML
- HTML_DEF_TAG_2 HEAD
- HTML_DEF_TAG BODY
- HTML_DEF_TAG STYLE
- HTML_DEF_TAG_2 NAV
- HTML_DEF_TAG TABLE
- HTML_DEF_TAG_2 TR, etc
- HTML_DEF_TAG_2 TD, etc
- HTML_DEF_TAG_2 TH, etc
- HTML_DEF_TAG_2 SPAN, etc
- HTML_DEF_TAG CENTER

Exemple construction d'un échiquier

ID RB ⊗ 
 ID DB ⊗ 
 ID FB ⊗ 
 ID CB ⊗ 
 ID TB ⊗ 
 ID pB ⊗ 
 ID RN ⊗ 
 ID DN ⊗ 
 ID FN ⊗ 
 ID CN ⊗ 
 ID TN ⊗ 
 ID pN ⊗ 

Blanches

noires

ID provisoire ⊗ les pièces ⊗ COTECOTE

SCRIPT ⊗ cache les pièces originales ⊗ hide (@' provisoire)

positionne les pièces

Blanches

Noires

les pions

for i in col_names ⊗ @H ('P' + i + '2') = @H 'pB'

for i in col_names ⊗ @H ('P' + i + 7) = @H 'pB'

@H 'Pb1' = @H 'CB'
 @H 'Pc1' = @H 'FB'
 @H 'Pd1' = @H 'DB'
 @H 'Pe1' = @H 'RB'
 @H 'Pf1' = @H 'FB'
 @H 'Pg1' = @H 'CB'
 @H 'Ph1' = @H 'TB'
 @H 'Pa1' = @H 'TB'
 @H 'Pa8' = @H 'TN'
 @H 'Pb8' = @H 'CN'
 @H 'Pc8' = @H 'FN'
 @H 'Pd8' = @H 'DN'
 @H 'Pe8' = @H 'RN'
 @H 'Pf8' = @H 'FN'
 @H 'Pg8' = @H 'CN'
 @H 'Ph8' = @H 'TN'



BUILD HTML VAR board ⊗ BR ⊗ avec générateur des Positions ⊗ TABLE

for ligne_num in rang_names ⊗ TR ⊗ WHTML '
' + ligne_num + '
'
 for col_num in col_names ⊗ BOARD CHANGER ⊗ ID 'P'+col_num+ligne_num ⊗ WHTML ''
 TD ⊗ WHTML ' '
 for col_num in col_names ⊗ TD ⊗ WHTML ' ' + col_num + ' '

@H 'BOARD' = board

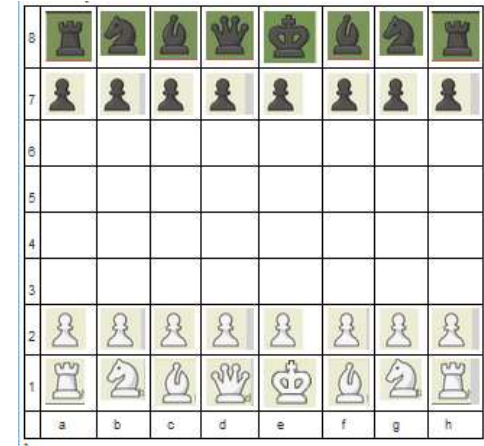
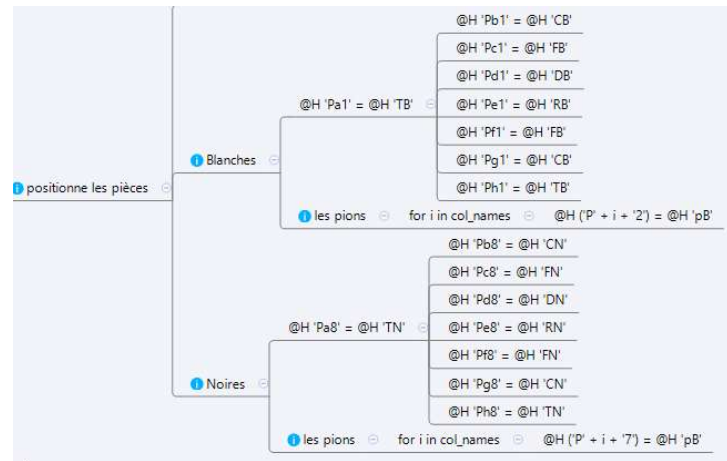
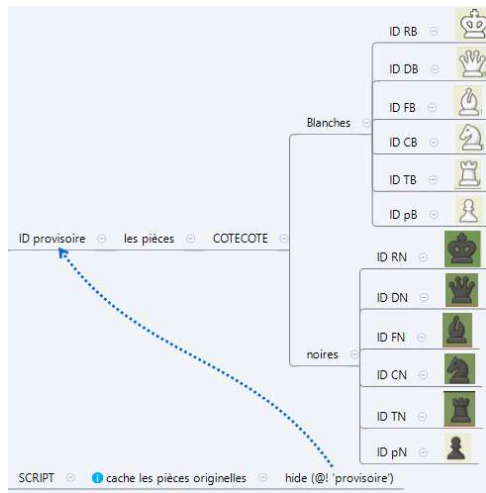


```

wl = ['Fa1', 'Db6', 'pf7', 'Re1'] ⊗ bl = ['Fa8', 'Db2', 'pf2', 'Re8']
board_html_place ('new_board', wl, bl)
    
```

exemple

Exemple construction d'un échiquier



```

BUILD HTML VAR board
BR
    avec générateur des Positions
TABLE
    for ligne_num in rang_names
        TR
            TD
                WHTML '&nbsp;'
            TR
                for col_num in col_names
                    TD
                        WHTML '&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;' + col_num + '&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;'
                TD
                    WHTML '<br>' + ligne_num + '<br>'
                for col_num in col_names
                    BOARD CHANGER ID 'P'+col_num+ligne_num WHTML ''
    @H 'BOARD' = board
    
```

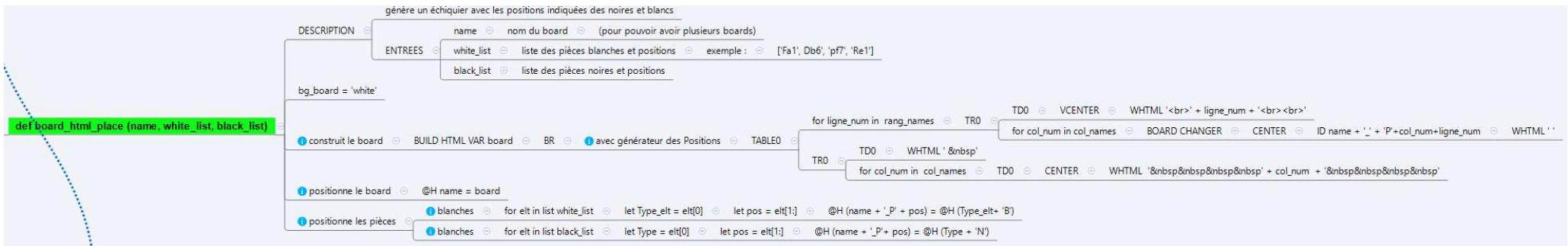


```

wl = ['Fa1', 'Db6', 'pf7', 'Re1']
bl = ['Fa8', 'Db2', 'pf2', 'Re8']
board_html_place ('new_board', wl, bl)
    
```

exemple

Générateur d'échiquier



```

function board_html_place (name, white_list, black_list) {
  /*
  génère un échiquier avec les positions indiquées des noires et blancs
  ENTREES
    name      nom du board      (pour pouvoir avoir plusieurs boards)
    white_list liste des pièces blanches et positions  exemple : ['Fa1', 'Db6', 'pF7', 'Re1']
    black_list liste des pièces noires et positions

  */
  bg_board = 'white';
  // construit le board
  var L_html;
  L_html = [];
  // notation br
  L_html.push ('<br>');
  // avec générateur des Positions
  L_html.push ('<table style="border:0px">');
  // ... for ligne_num in rang_names
  for (let ligne_num of rang_names) {
    L_html.push ('<tr style="border:0px">');
    L_html.push ('<td style="border:0px">');
    L_html.push ('<div style="display:table-cell; vertical-align:middle">');
    L_html.push ('<br>' + ligne_num + '<br><br>');
    L_html.push ('</div>');
    L_html.push ('</td>');
    // ... for col_num in col_names
    for (let col_num of col_names) {
      notation TD
      L_html.push ('<td>');
      L_html.push ('<div style="background-color:' + bg_board + '>');
      // notation center
      L_html.push ('<center>');
      L_html.push ('<span id="' + name + '_' + 'P'+col_num+ligne_num + '>');
      L_html.push (' ');
      L_html.push ('</span>');
      L_html.push ('</center>');
      L_html.push ('</div>');
      board_change_bg ();
      L_html.push ('</td>');
    }
    L_html.push ('</tr>');
  }
  L_html.push ('<tr style="border:0px">');
  L_html.push ('<td style="border:0px">');
  L_html.push ('<nbsp>');
  L_html.push ('</td>');
}
  
```

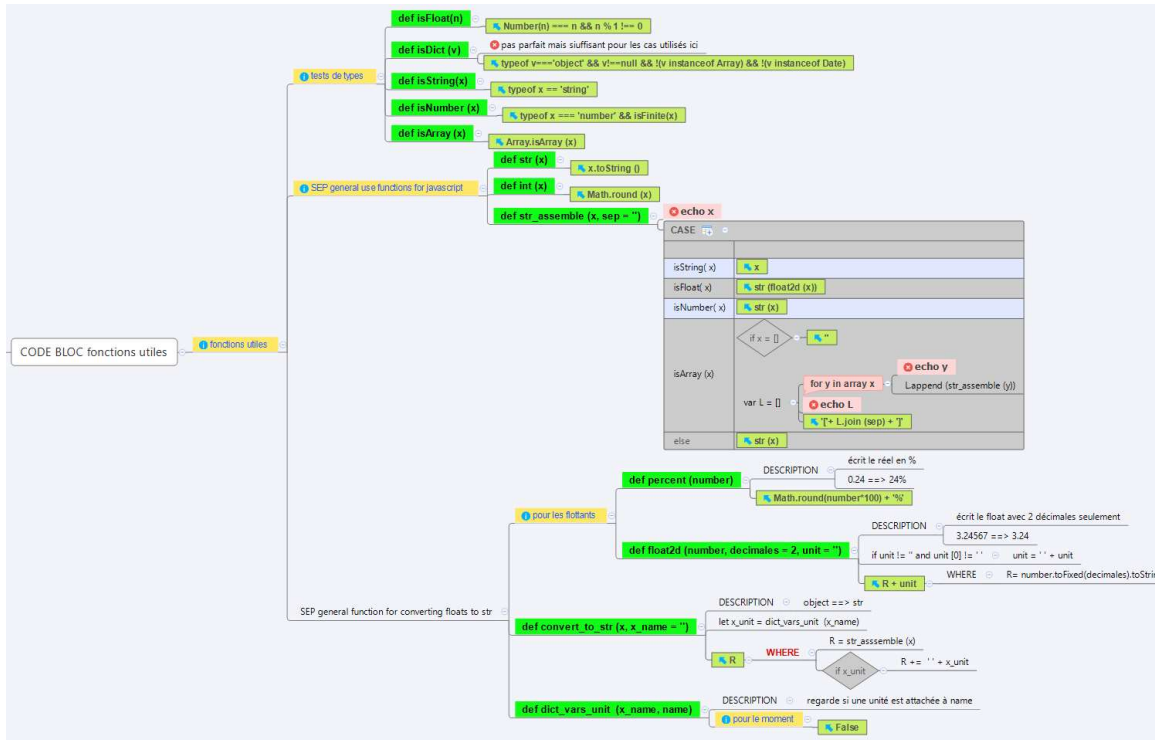


```

// ... for col_num in col_names
for (let col_num of col_names) {
  L_html.push ('<td style="border:0px">');
  notation center
  L_html.push ('<center>');
  L_html.push ('<span style="display:table-cell; vertical-align:middle">');
  L_html.push ('</span>');
  L_html.push ('</td>');
}
L_html.push ('</tr>');
L_html.push ('</td>');
// echo L_html
console.log ('L_html = ', L_html);
board = str_assemble (L_html);
// echo board
console.log ('board = ', board);
// positionne le board
document.getElementById (name).innerHTML = board;
// positionne les pièces
// blanches
// ... for elt in list white_list
for (let elt_index in white_list) {
  elt = white_list [ elt_index ];
  let Type_elt = elt[0];
  let pos = elt.slice (1);
  document.getElementById ((name + '_P' + pos)).innerHTML = document.getElementById (Type_elt + 'B').innerHTML;
}
// noires
// ... for elt in list black_list
for (let elt_index in black_list) {
  elt = black_list [ elt_index ];
  let Type_elt = elt[0];
  let pos = elt.slice (1);
  document.getElementById ((name + '_P' + pos)).innerHTML = document.getElementById (Type_elt + 'N').innerHTML;
}
-]
-]
w1 = ['Fa1', 'Db6', 'pF7', 'Re1'];
b1 = ['Fa8', 'Db2', 'pF2', 'Re8'];
board_html_place ('new_board', w1, b1);
</script>
  
```

Génération de javascript pour les fichiers html

Librairie de fonctions



USAGE

SCRIPT INSERT CODE BLOC fonctions utiles

Génération de programmes Ada

Langage typé, compilé, syntaxe hyper rigoureuse, etc pour 'faciliter la maintenance', performant

Avantages	Inconvénients
Tout décrire dans le détail	Verbosité lourd à écrire Répétitions Quelques tournures lourdes

1) écriture difficile et verbeuse, qui rebute beaucoup de programmeurs
2) Pas toujours assez près du système

→ retour C, C++

KM2

Lisibilité
Simplicité
Efficacité

Report 2022

Ada : déclaration de types et variables

Types and variables are declared in tables as follows :

*setf TYPES			
Nom	is	Subtype	Doc
name of type	possible intervals	name of type if subtype of another type	description of the variable
Numero_Bateau	range 1..25		There are only 25 boats
Capacite_Bateau	range 0..10		The size of each boat is bounded
Taille_Groupe	range 1..Capacite_Bateau/Last	Capacite_Bateau	A group of visitors must be lower than the size of a boat
Etat_Manager	(Attente_Bateau, Attente_Groupe, Demarrage_Bateau, Fermeture)		The different parts of the work of the manager of the boats

*setf VARIABLES			
Nom	Type	Doc	init
name of variable	type of variable	description of the variable	some value
Numero	Numero_Bateau	each boat has a number	
Bateau	Numero_Bateau		
Mon_Numero	Numero_Bateau		
Id	Numero_Bateau		
Notre_Bateau	Numero_Bateau		
Personnes	Taille_Groupe		
Occupation	Capacite_Bateau	size of the boat	
Deja_A_Bord	Capacite_Bateau		
Occupation	Taille_Groupe		
Taille	Taille_Groupe	size of a group	
Mess	String	pour message d'erreur	
Etat	Etat_Manager		
Fermeture_Demandee	Boolean		
Les_Bateaux	array(Numero_Bateau) of Un_Bateau	all boats	
Gen	Taille_AleatoireGenerator	for the simulation	
Fin_Du_Jeu	exception	end of simulation	

Report 2022

Génération de programmes Ada

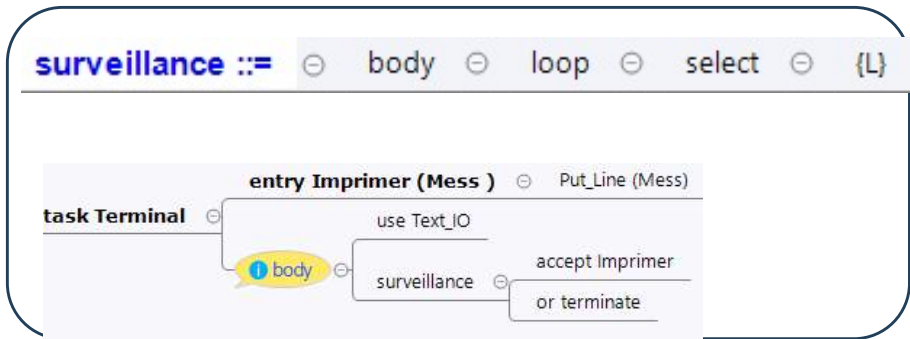
Code in KM2



Ada Code (generated automatically) :

```
-- >> task Terminal ----- --
task Terminal is
  declarations --
    entry Imprimer (Mess : String ) ;
end Terminal;
  body --

  task body Terminal is
    --
    body --
    use Text_IO;
    begin
      loop
        select
          accept Imprimer (Mess : String ) do
            Put_Line (Mess);
          end Imprimer;
        or terminate;
      end select;
    end loop;
  end Terminal;
```

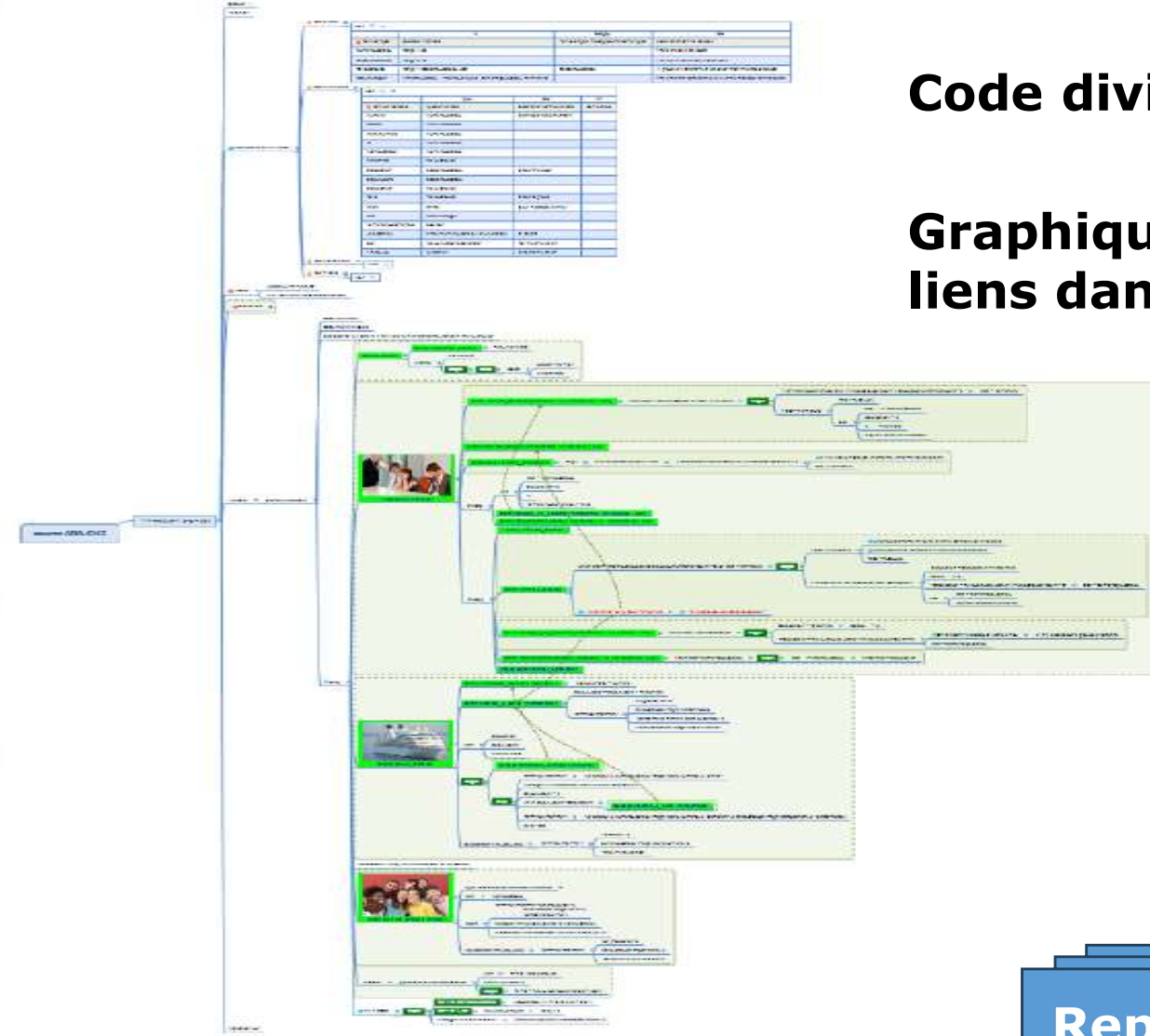


Patterns de programmation

Description unique et préalable des éléments utilisés
→ **complétion automatique, réordonnancement**
+ lisibilité du code

Langage ada : comparaison

Ada source : 240 lines in KM2 : 95 lines



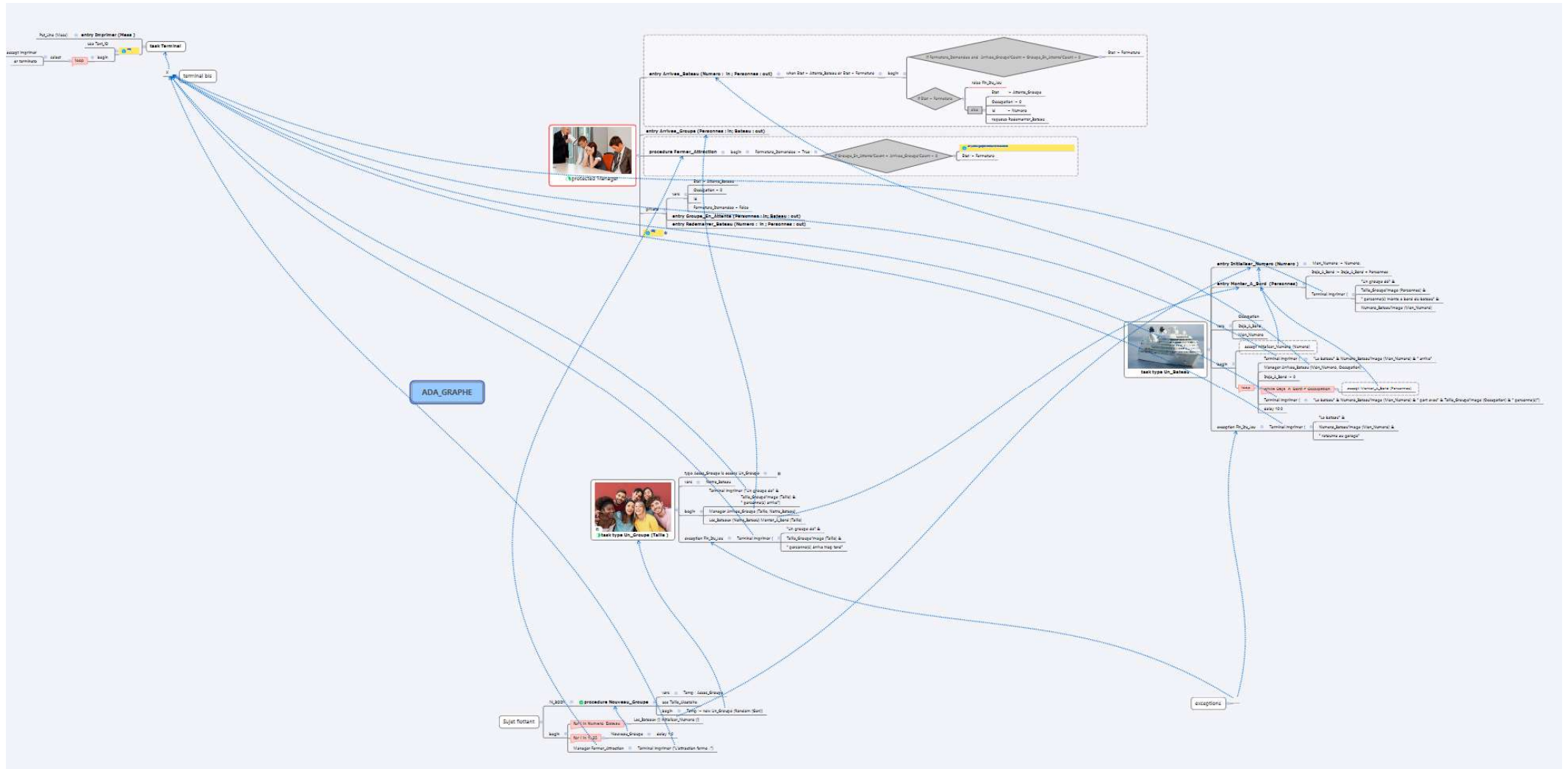
R

Code divisé par 2.5

Graphiques et liens dans le code

Report 2022

Programmes ada : une autre vision



Exemple dédié : Solver Cplex

calcul des KPI

#

KPI			
	doc	calcul	before
final_overcapacity	sur-occupation totale à la fin	$\Sigma (\text{mdl.max}(0, \text{occupancy_vars}[d, \text{NB_PERIODS}] - \text{capacity}[d]) \forall d)$	
nb_long_transfers	nb de transferts longue distance	$\Sigma (\text{use_link_vars}[d1, d2, t] \forall d1 \forall d2 \text{ if } \text{is_long}[d1][d2] \forall t)$	
nb_short_transfers	nb de transferts à petite distance	$\Sigma (\text{use_link_vars}[d1, d2, t] \forall d1 \forall d2 \text{ if not } \text{is_long}[d1][d2] \forall t)$	

CONTRAINTE			
	domaine	definition	en_clair
Short transfers bounds	$\forall t$	$\text{link_vars}[d1, d2, t] \leq 1 \forall d1 \forall d2 \text{ if not } \text{is_long}[d1][d2]$	
Initial state	$\forall d$	$\text{occupancy_vars}[d, 0] == \text{initial}[d]$	
structural constraint between user_link and link	$\forall d \forall d1 \forall t$	$\text{use_link_vars}[d, d1, t] == (\text{link_vars}[d, d1, t] \leq 1)$	cf definitions
number of transfers from a department less than current number of cases	$\forall d \forall t$	$\Sigma (\text{link_vars}[d, d1, t] \forall d1) \leq \text{occupancy_vars}[d, t]$	
maximum number of LONG transfers	$\forall t$	$\Sigma (\text{use_link_vars}[d1, d2, t] \forall d1 \forall d2 \text{ if } \text{is_long}[d1][d2]) \leq \text{MAX_NB_LONG_TRANSFERS_PER_PERIOD}$	
maximum number of SHORT transfers	$\forall d2$	$\Sigma (\text{use_link_vars}[d1, d2, t] \forall d1 \text{ if not } \text{is_long}[d1][d2] \forall t) \leq \text{MAX_NB_SHORT_TRANSFERS_PER_DEPARTMENT}$	à tout moment, on ne peut pas transférer à courte distance plus qu'un certain nombre de personnes
conservation constraints including new cases to come	$\forall d \forall t$	$\text{occupancy_vars}[d, t+1] == \text{new_rea}[d][t] + \text{occupancy_vars}[d, t] + \Sigma (\text{link_vars}[d1, d, t] \forall d1) - \Sigma (\text{link_vars}[d, d1, t] \forall d1)$	l'occupation à t+1 du dpt d est l'occupation précédente plus les nouvelles entrées locales plus les transferts (positifs ou négatifs)

Exemple dédié : solver cplex, code généré

```
# CONTRAINTE
# Short transfers bounds
#  $\forall t$ 
#  $\text{link\_vars}[d1, d2, t] \leq 1 \forall d1 \forall d2 \text{ if not is\_long}[d1][d2]$ 
mdl.add_constraints ( link_vars[d1, d2, t] <= 1 for d1 in deps for d2 in deps if not is_long[d1][d2] for t in transfer_periods )
# CONTRAINTE
# Initial state
#  $\forall d$ 
#  $\text{occupancy\_vars}[d, 0] == \text{initial}[d]$ 
mdl.add_constraints ( occupancy_vars[d, 0] == initial[d] for d in deps )
# CONTRAINTE
# structural constraint between user_link and link
#  $\forall d \forall d1 \forall t$ 
#  $\text{use\_link\_vars}[d, d1, t] == (\text{link\_vars}[d, d1, t] <= 1)$ 
# EXPLICATION
# cf definitions
mdl.add_constraints ( use_link_vars[d, d1, t] == (link_vars[d, d1, t] <= 1) for d in deps for d1 in deps for t in transfer_periods )
# CONTRAINTE
# number of transfers from a department less than current number of cases
#  $\forall d \forall t$ 
#  $\sum (\text{link\_vars}[d, d1, t] \forall d1) \leq \text{occupancy\_vars}[d, t]$ 
mdl.add_constraints ( mdl.sum (link_vars[d, d1, t] for d1 in deps ) <= occupancy_vars[d, t] for d in deps for t in transfer_periods )
# CONTRAINTE
# maximum number of LONG transfers
#  $\forall t$ 
#  $\sum (\text{use\_link\_vars}[d1, d2, t] \forall d1 \forall d2 \text{ if is\_long}[d1][d2]) \leq \text{MAX\_NB\_LONG\_TRANSFERS\_PER\_PERIOD}$ 
mdl.add_constraints ( mdl.sum (use_link_vars[d1, d2, t] for d1 in deps for d2 in deps if is_long[d1][d2]) <= MAX_NB_LONG_TRANSFERS_PER_PERIOD for t in transfer_periods )
# CONTRAINTE
# maximum number of SHORT transfers
#  $\forall d2$ 
#  $\sum (\text{use\_link\_vars}[d1, d2, t] \forall d1 \text{ if not is\_long}[d1][d2]) \leq \text{MAX\_NB\_SHORT\_TRANSFERS\_PER\_DEPARTMENT}$ 
# EXPLICATION
# à tout moment, on ne peut pas transférer à courte distance plus qu'un certain nombre de personnes
mdl.add_constraints ( mdl.sum (use_link_vars[d1, d2, t] for d1 in deps if not is_long[d1][d2] for t in transfer_periods ) <= MAX_NB_SHORT_TRANSFERS_PER_DEPARTMENT for d2 in deps )
# CONTRAINTE
# conservation constraints including new cases to come
#  $\forall d \forall t$ 
#  $\text{occupancy\_vars}[d, t+1] == \text{new\_rea}[d][t] + \text{occupancy\_vars}[d, t] + \sum (\text{link\_vars}[d1, d, t] \forall d1) - \sum (\text{link\_vars}[d, d1, t] \forall d1)$ 
# EXPLICATION
# l'occupation à t+1 du dpt d est l'occupation précédente plus les nouvelles entrées locales plus les transferts (positifs ou négatifs)
mdl.add_constraints ( occupancy_vars[d, t+1] == new_rea[d][t] + occupancy_vars[d, t] + mdl.sum (link_vars[d1, d, t] for d1 in deps ) - mdl.sum (link_vars[d, d1, t] for d1 in deps ) for d in deps for t in transfer_periods )
# ===== KPI
# définition du KPI final_overcapacity
# sur-occupation totale à la fin
final_overcapacity = mdl.sum (mdl.max(0, occupancy_vars[d, NB_PERIODS] - capacity[d]) for d in deps )
mdl.add_kpi ( final_overcapacity )
# définition du KPI nb_long_transfers
# nb de transferts longue distance
nb_long_transfers = mdl.sum (use_link_vars[d1, d2, t] for d1 in deps for d2 in deps if is_long[d1][d2] for t in transfer_periods )
mdl.add_kpi ( nb_long_transfers )
# définition du KPI nb_short_transfers
# nb de transferts à petite distance
nb_short_transfers = mdl.sum (use_link_vars[d1, d2, t] for d1 in deps for d2 in deps if not is_long[d1][d2] for t in transfer_periods )
mdl.add_kpi ( nb_short_transfers )
```

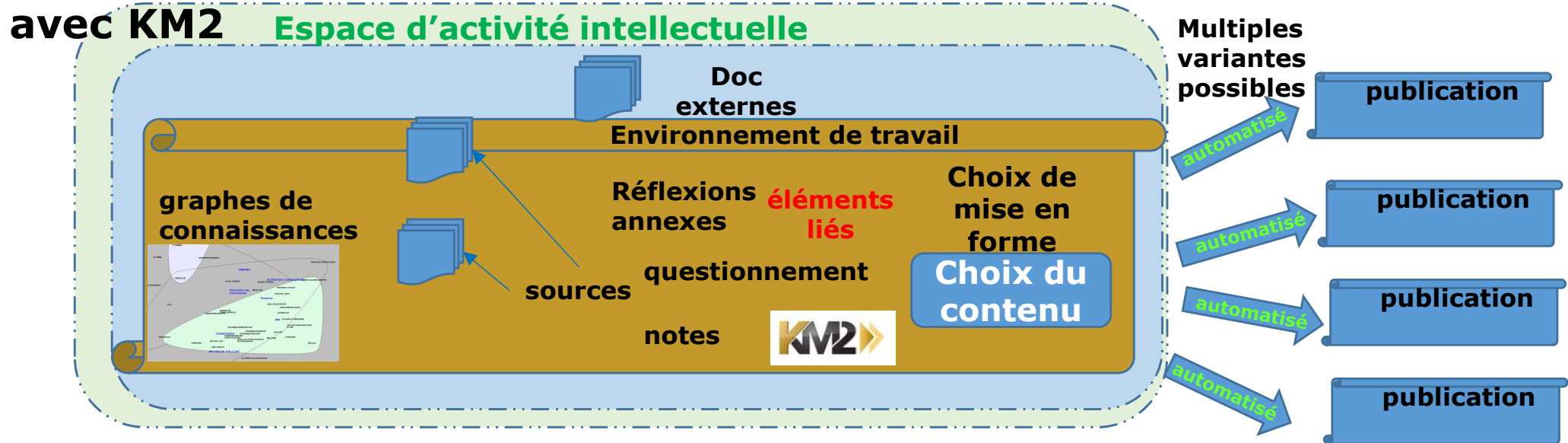
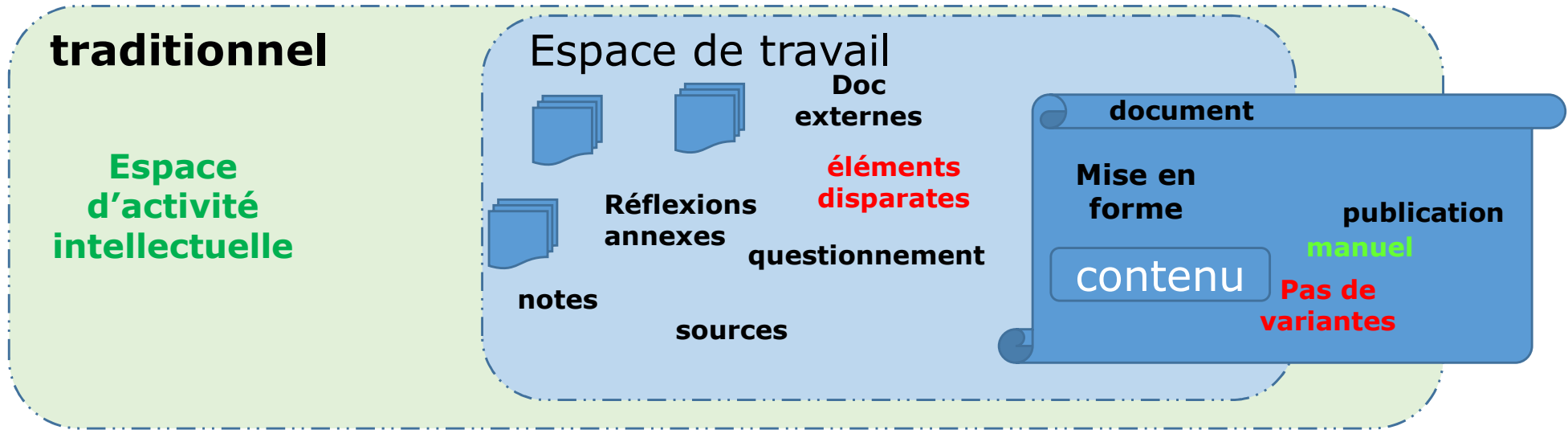
Doc complet

Nouveaux concepts pour la programmation

- Édition avec les schémas SGH → efficacité, densité
 - Graphiques
 - Tableaux
- Utilisation de macros : → confort, patterns
 - Textuelles
 - Groupe de mots ou icônes
- Programmation proche du pseudocode
- Nouveaux outils de debug et trace
- Programmation non linéaire
 - WHERE, =, objets structurés
 - Tableaux, hyperliens
- Source + documentation = 1 fichier → N fichiers html/css/js/python/ ...
 - Facilités pour inclure du code dans le html

Conclusions

KM2 : Un atelier de gestion de connaissance et de production de documents



CONCLUSION 1 : les schémas SGH

- Les schémas SGH sont un **moyen universel d'écriture** de documents complexes.
 - **Généralisation intégrée des documents textuels et du mind mapping**
- Les diagrammes SGH fournissent un formalisme pour la représentation de documents et programmes
 - beaucoup plus puissant et plus riche.
 - La rédaction de documents (texte) avec les SGH permet une nouvelle dimension de dynamique et de souplesse
- Les schémas SGH s'appuient sur un processeur qui permet d'en exploiter toute la puissance de représentation :
 - Extractions partielle et multiples vers plusieurs types de documents
 - Le contenu du document EST un contenu, et non pas un contenant
 - Possibilité d'exploiter la même information sous plusieurs angles

CONCLUSION 2 : le processeur de schémas (KM2)

- Permet de transformer un document SGH vers d'autres formats
 - Faisabilité acquise!
- L'écriture des macros pour une cible spécifique est relativement simple pour obtenir les séquences de base :
 - ~1 semaine pour ada, prolog en V0
 - (incluant quelques difficultés conceptuelles à résoudre)
- L'écriture des macros utiles, et leur bonne forme nécessite un temps d'expérience
 - Possibilités de capitaliser ces expériences dans les macros (rédaction/programmation)
 - Outils pour la vérification de macros et du code intégrés (debug + gestion d'erreurs)
- Le processeur KM2 ouvre de nouvelles portes sur la notion de codage :
 - Programmes non linéaires, utilisation de tables, cohérence code/ documentation, graphiques dans le code
 - Utilisation de patterns, d'expressions plus proche du langage naturel, etc
 - Gain en volume et temps
- **Le processeur est bootstrappé** : > 50% du code est en schémas SGH
 - > 50 000 lignes générées par KM2 en python,
 - > 10 000 lignes en JS,
 - des milliers de pages en html : la documentation KM2, rapports d'analyse, etc.
 - Amélioration continue

CONCLUSION 3 : génération de documents

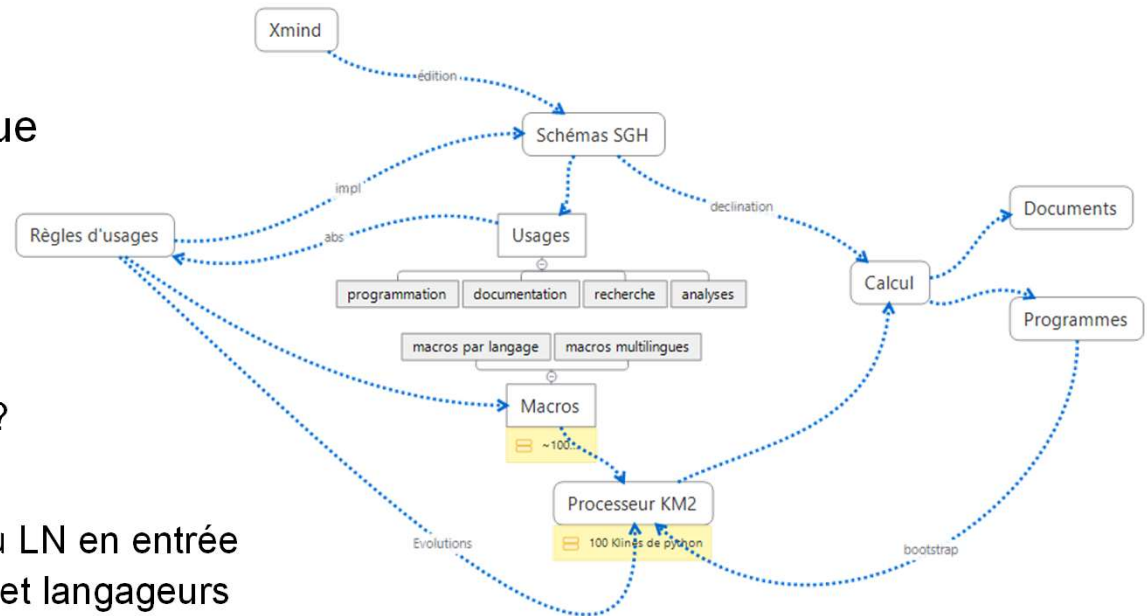
- Documentation agnostique vs la cible (pdf, html, txt, doc, ...)
 - Le HTML, avec le browser permet des opérations en lecture et usage inédites
- Gains de temps :
 - Rédaction,
 - Mise en page facilité : sommaires, dispositions spécifiques
- L'impossible / le compliqué devient possible
 - Trouver le bon jeu de macros → nouvelles possibilités
 - Génération de documents excel

Conclusion 4 : génération de programmes

- Description plus universelle de programmes
 - Utilisation systématique d'arbres
 - Aussi possible : tableaux → types, variables, fonctions, ... valeur tests
- Amélioration de :
 - Lisibilité : mots, graphiques
 - Taille du code
 - Cohérence code/documentation
 - Cibles multiples
 - Debugging (commentaires exécutables ...)
 - Etc
- Le processeur KM2 ouvre de nouvelles portes sur la notion de codage
 - Programmes non linéaires, utilisation de tables, cohérence code/ documentation, graphiques dans le code
 - Utilisation de patterns, d'expressions plus proche du langage naturel, etc
 - Gain en volume et temps
 - Utilisation de Pseudo Langage Naturel
 - → **éducation : revisiter la pédagogie ?**

Conclusion : suites

- Commencer la diffusion :
 - Compléter la documentation (déjà abondante)
 - Tutoriaux (partiels existants)
 - Communauté d'utilisateurs (par cible ?)
 - Usages : éducation, recherche, programmation professionnelle
 - Écriture de macros spécifiques et génériques
 - Retours d'expérience
- Disposer d'un éditeur de schémas dynamique
 - En particulier :
 - macros d'édition
 - Utilisation de patterns
 - Correction et syntaxe en temps réel
 - Génération de nœuds complémentaires
 - Version web ou basée sur un browser local ?
- Elargir les concepts :
 - Rajout de concepts : améliorer l'utilisation du LN en entrée
 - Utilisation d'outils de raisonnement : prolog et langageurs
 - etc = ?



Pour visiter la documentation

[La présentation du 25/01/2025 en pdf](#)

- <https://www.km2-conseil.fr//IFB-20250121/EXPOSE%20LG%20IFB%2021%20JANVIER%202025.pdf>
- [Version simplifiée du 25/01/2025 en htm](#)
 - <https://www.km2-conseil.fr//IFB-20250121/EXPOSE%20LG%20IFB%2021%20JANVIER%202025.htm>
- [Documentation de KM2 :](#)
 - https://www.km2-conseil.fr//KM2_DOC_WEB//KM2%20MACHINE%20AIDE%20DOCUMENTATION%20HTML.htm