Travailler sur les réseaux

Printemps 2014

Pierre Gradit
SARL Mezzonomy
34 Place de Catalogne
31700 Blagnac

♦

1. Travailler sur les mots

En informatique, le mot est un objet essentiel. Un mot est une suite de caractères. Ce paragraphe est un exemple d'un long mot de caractères latin-1 (iso-8859-1), classe composée des lettres de l'alphabet latin usuel auquel sont adjointes les espaces, les ponctuations, etc.

Pour pouvoir travailler sur cette classe d'objet, une simplification mathématique usuelle consiste à ne considérer que les énoncés sans espaces et sans ponctuation, juste les lettres de l'alphabet usuel :

azertyuiop

Enoncé 1 : Un mot de lettres

Il existe de multiples façons de mathématiser cette classe d'objet, les systèmes à base de réécriture de la fin des années 90 proposaient un outil élégant pour le spécifier :

op (__) : Mot x Mot → Mot (associative)
op _ : Lettre → Terme

Enoncé 2 : Spécification simplifiée d'un mot [MAUDE ELAN]

La grande force de ces théories et de considérer l'attribut « *(associative)* » comme un raccourci pour l'équation d'associativité :

op (__) : Mot x Mot → Mot
eq (a(bc)) = ((ab)c)
op _ : Lettre → Mot

Enoncé 3 : Spécification complète d'un mot

Considérons les équation de réduction ci-dessous :

eq (a(bc)) → (abc)
eq ((ab)c) → (abc)

Enoncé 4 : Calcul syntaxique de la simplification

Ces équations de réécritures orientées définissent donc un calcul : deux expressions complètement parenthésées représentent le même mot si elles se réduisent à la même expression syntaxiques simplifiée. Le calcul correspondant n'a pas besoin d'être déterministe, il lui suffit d'être confluent et ainsi aucun choix ne change le résultat final.

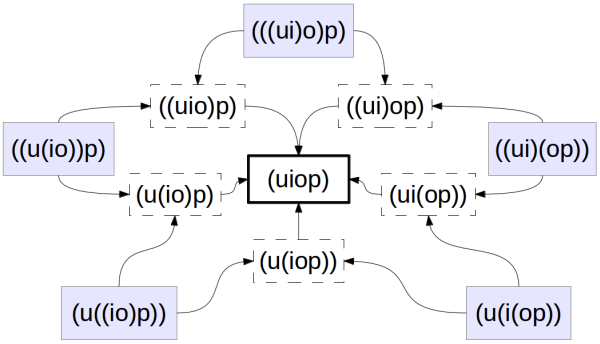


Figure 1 : Un mot comme classe d'équivalence d'un calcul non déterministe

La simplification de la syntaxe en *(uiop)* permet de cacher de la complexité ! Pour donner la mesure de la complexité cachée dans la syntaxe simplifiée, un exercice pour s'en rendre compte consiste à énumérer le nombre d'élément de la classe pour un mot de longueur $n > 1$.

Dans le cas des mots, une écriture syntaxique acceptable de la forme simplifiée, de l'objet représenté, existe. Dans le cas général, ce n'est pas le cas : l'objet représenté modulo la classe d'équivalence ne peut pas se réduire à une forme simplifiée imprimable simplement.

Par exemple, si on oublie la légende, quel est l'objet représenté par la figure n°1 ? Si vous pensez que c'est un « *graphe* » c'est exact, mais c'est aussi une « *relation* ». Si vous pensez que je « *joue sur les mots* », c'est exact. C'est mon métier, c'est ainsi.

2. Travailler sur les cycles

Continuons donc à jouer sur les mots, car c'est bien la classe la plus solide à notre disposition. Le terme de jeu s'entend ici pleinement car sans trop nous distraire dans les détails, nous allons ajouter une équation sur les représentation des mots – la forme syntaxique où une seule paire de parenthèse délimite le mot.

Avec cette syntaxe simplifiée des mots, nous créons des sortes de mots où les lettres sont indifféremment au début ou à la fin du mot, avec l'équation de l'énoncé ci-dessous :

$$eq\ (x\_) = (\_x)$$

Enoncé 7 : Sorte de mots cycliques

Nous pouvons reproduire la même construction, dites de « *construction de classes d'équivalence* » pour nos sortes de chaînes. Mais la simplification syntaxique n'est pas au rendez-vous.

(azerty)	(uiop)
(zertya)	(iopu)
(ertyaz)	(opui)
(rtyaze)	(puio)
(tyazer)	
(yazert)	

Enoncé 8 : Exemples de sorte de mots cycliques, avec les classes en colonnes

L'idée à la base de ma simplification syntaxique est de considérer que ces sortes de chaînes peuvent

être représenté par le graphe de la fonction « *successeur* », les deux classes identifiées dans l'énoncé se retrouvant sous la forme de deux composantes connexes du graphe ci-dessous :

Graphe 1 : Exemples de sorte de mots

La simplification de la syntaxe semble à portée de main, si la syntaxe des graphes était aussi robuste qu'on le prétend. J'étais à ce point précis lors de mon audition en 2001 pour rentrer au CNRS. Celui qui devait m'évaluer n'avait pas l'air très convaincu et il me posa une question difficile : « *et si j'ai plusieurs a ?* ».

(patatras)
(atatrasp)
(tatraspa)
(atraspat)
(traspata)
(raspatat)
(aspatatr)
(spatatra)

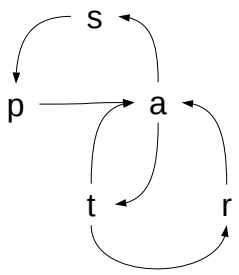


Figure 2 : Difficulté avec le graphe « *successeur* »

La difficulté est sérieuse et j'échouerais à formuler une réponse plus claire que celle-ci : pouvoir dupliquer les lettres suppose que la bijection s'applique non sur elles mais sur leur « *support* ».

C'est d'ailleurs assez précisément la réalité informatique du dessin à droite de la figure n°2, où les lettres sont écrites dans des cadres invisibles, qui servent aussi de support aussi aux connecteurs représentant la relation erronée de succession, qui est tout sauf une bijection. Les flèches pointent sur un objet que l'on ne voit pas, le cadre où la lettre est écrite.

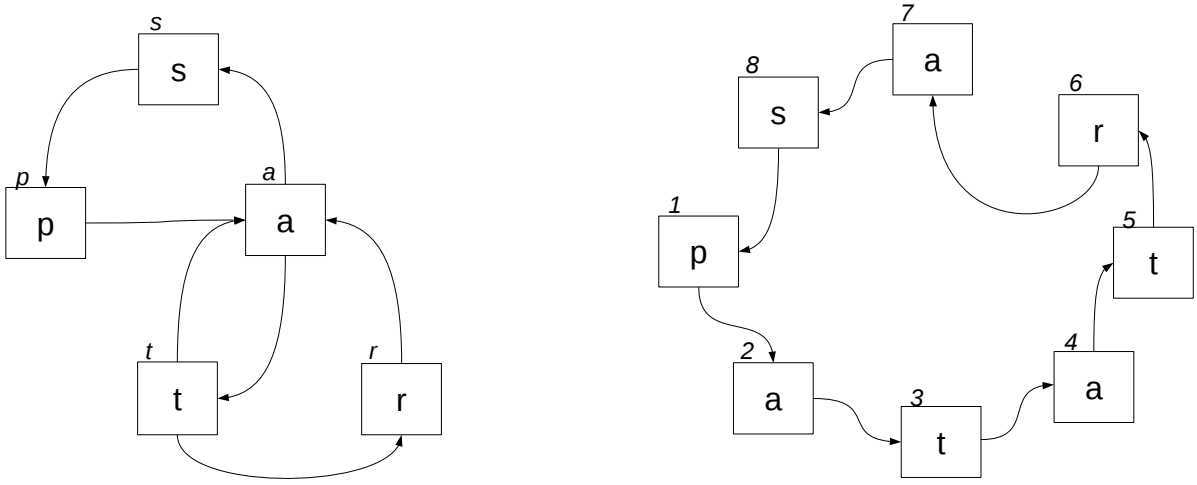


Figure 3 : Identification des supports

La partie gauche de la figure n°3 reprends la figure n°2 où les supports portent les mêmes identités que leur contenus, là où la partie droite donne des numéros aux supports permettant à différents supports de porter la même lettre.

_	(1+_)	Lettre
001	002	p
002	003	a
003	004	t
004	005	a
005	006	t
006	007	r
007	008	a
008	001	s

Enoncé 17 : (patatras) avec successeur sur les « *supports* »

Mais la simplification syntaxique est loin d'être au rendez-vous. Au contraire, tout ceci à l'apparence d'une inutile complexité, un vrai sac de nœuds pour un faux problème : l'évaluation fut un échec. Ce n'est que plusieurs années plus tard dans un contexte industriel, que je trouvais une solution satisfaisante aux problèmes soulevés ici, dont j'entame la présentation.

♦

Dans les systèmes permettant de dessiner des « *graphes* » issus du commerce, il existe un autre support que les rectangles, ce sont les segments. Ils ont en outre un sens typographique, le segment définit la « ligne de base » sur laquelle il est possible de déposer les caractères d'imprimerie, ce qui permet une bonne association de la lettre, voire du mot, écrit sur la ligne.

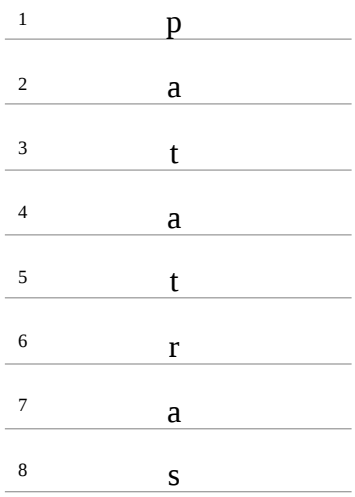


Figure 4 : Identification alternative des supports

La dernière question qui demeure pour réaliser notre simplification syntaxique consiste à dénoter sans ambiguïté la bijection successeur.

Pour faire une différence nette entre la fonction successeur et les trait supportant les lettres, nous prenons comme convention que cette forme sera toujours composée de forme courbe, lorsque les supports seront toujours droits.

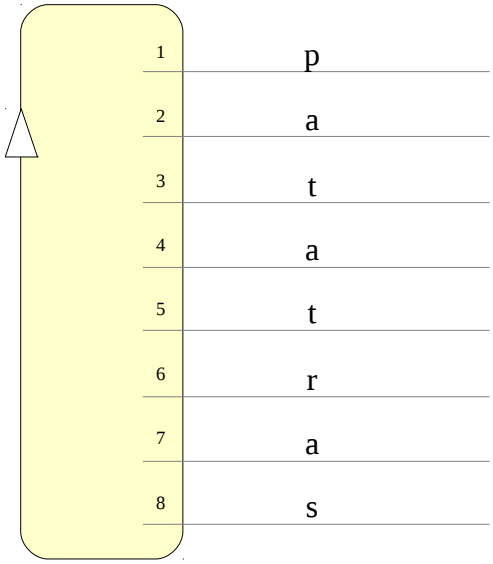


Figure 5 : Construction de la fonction successeur

Pour avoir notre premier énoncé complet dans notre syntaxe de graphe, il suffit de considérer que les « *supports* » sont les intersection entre les segments et la courbe. En outre, par convention, nous indiquons à part le « *sens de rotation* » des courbes fermées, une seule fois.

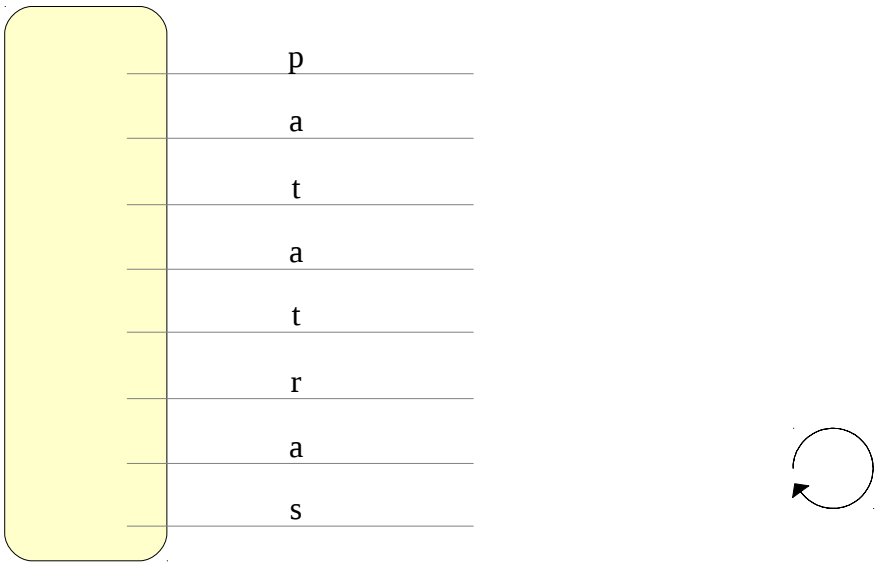


Diagramme 1 : (patatras)

Il s'agit bien d'une syntaxe, et nous pouvons produire d'autres graphe représentant le même objet, comme par exemple sous forme normale, construite à partir de pavages plan à base de quart de cercles.

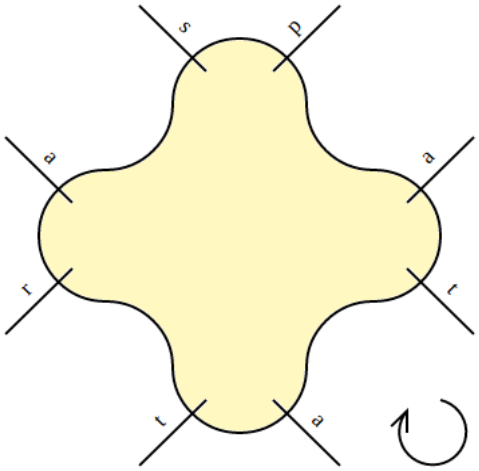


Diagramme 2 : forme pavée d'(atraspat)

Les diagramme de ce type se dessinent aujourd'hui de trois façons :

- **à la main (Diagramme n°2)** : une pointe sur un plan, à la manière d'un stylo. Mais ces diagrammes manuels , sur un papier, sur un tableau, s'ils sont aussi valides syntaxiquement que ceux construit avec les outils informatiques, sont difficiles à imprimes correctement.
- **avec les pavages (Diagramme n°2)** : un peu complexe à mettre en œuvre, ils sont idéals pour les applications théoriques et ce document n'aurait sans doute pas vu le jour sans cette technique rapide, précise et fluide, mais ils ne sont en aucun cas une nécessité de la syntaxe.
- **avec Office (Diagramme n°1)** : très économique et d'un bon rendu imprimé, l'utilisation des possibilités graphiques standard d'une suite bureautique permet de dessiner avec les outils de tous les jours des objets mathématiques consistants.

3. Travailler sur les BDD

Nous allons repartir sur une structure en apparence plus simple que les mots, puisque nous supprimons l'associativité de l'opération de collage :

op (__) : BDD x BDD → BDD
op _ : Lettre → BDD

Enoncé 2 : Syntaxe binaire de lettres

Ces structures sont appelés « *diagramme de décision binaire* » (ang. Binary Decision Diagram), et comme le terme « *diagramme* » le suggère, ces objets ont une forme graphique reconnue :

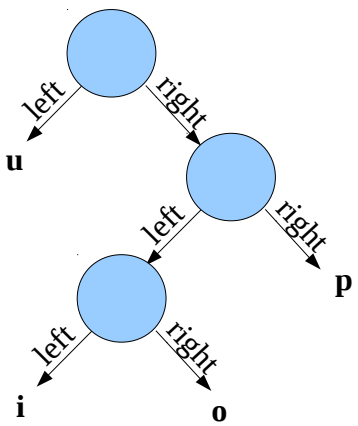


Figure 2 : (u((io)p)) sous forme de BDD

Cet forme graphique correspond à la décomposition de l'opérateur binaire (__) en deux opérateurs unaires left(_) et right(_), les formes graphiques pouvant toujours être mise sous la forme de valuation d'algèbres unaires.

op left(_) : BDD → BDD
op right(_) : BDD → BDD
op _ : Lettre → BDD

Enoncé 2 : graphe binaire de lettres

La figure n°2 est remarquablement proche de cette forme pavée ci dessous :

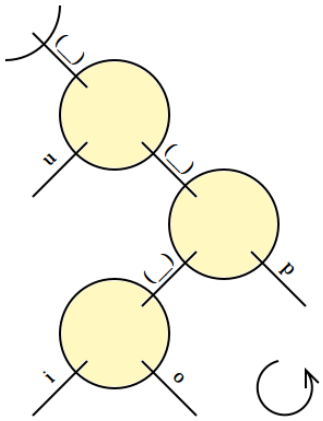
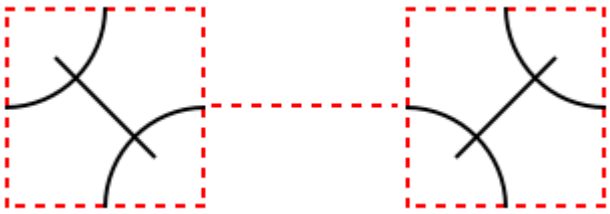


Diagramme 3 : forme normale d'(u((io)p))

Sans entrer encore dans tous les détails mathématiques, nous pouvons appliquer à ce diagramme une transformation graphique sur les « *arêtes* », qui correspond à faire tourner d'un quart de tour le pavé la contenant :



Règle 1 : Perspective

Appliquons à présent la règle « *perspective* » à l'ensemble des arêtes « (__) » du diagramme 3 :

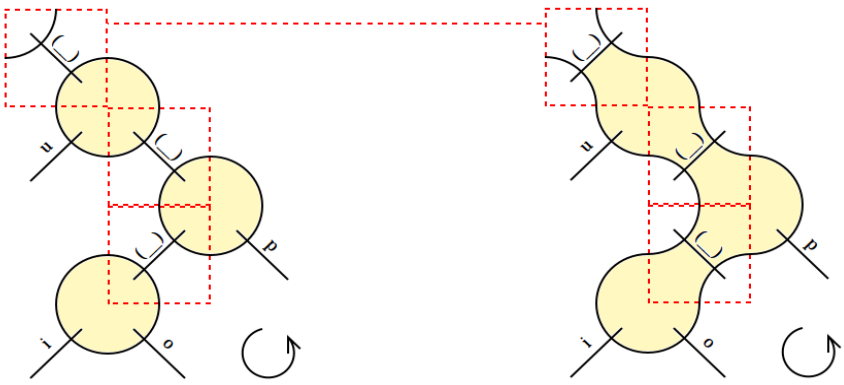


Diagramme 4 : transformation duale d'(u((io)p))

Nous avons à gauche la forme graphique standard des diagrammes de décision binaire, et à droite nous avons une image de la forme syntaxique. En effet, supposons que nous sommes une tête de lecture avec une pile qui parcours le fil dans le sens de parcours depuis le début, en haut à gauche : nous ouvrons la première parenthèse, puis la lettre u, puis nous ouvrons la deuxième parenthèse, puis la troisième, nous lisons la lettre i, puis la lettre o puis nous fermons la troisième parenthèse, lisons la lettre p, fermons la deuxième parenthèse, et enfin fermons la première. Soit très exactement le script de lecture que la même machine ferait de « *(u((io)p))* ».

Dans notre syntaxe, la forme syntaxique et la forme graphique du même BDD sont reliés par une opération élémentaire, l'application de la règle perspective à tous les segments. Avant de détailler cette opération dans un contexte mathématique, nous allons nous attarder sur la question de l'associativité.

Considérons le mot (uiop) – et non le cycle correspondant. Appliquons le même raisonnement à l'envers, nous pouvons dessiner le diagramme correspondant à sa lecture puis appliquer la règle de perspective à la seule arête.

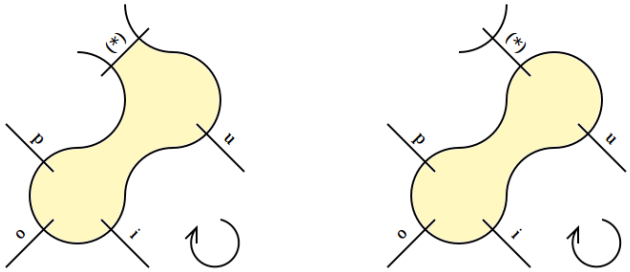


Diagramme 4 : (uiop) de la syntaxe au graphe

Considérons à présent le cycle (uiop), nous pouvons appliquer la même méthode de construction que pour (patatras), et considérons les deux diagrammes côte à côte.

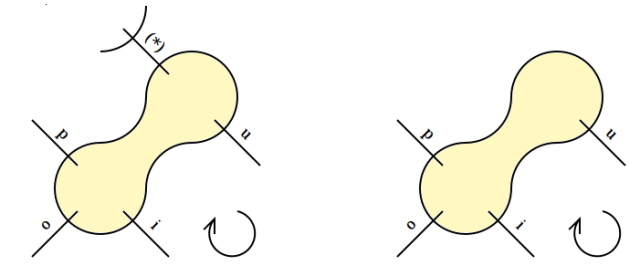


Diagramme 5 : (uiop) comme mot et comme cycle

Dans la formulation modale, la seule différence entre le mot et le cycle est une question de modalité, de point de vue : un mot est un cycle que l'on regarde à partir d'un point donné.

Cette capacité de pouvoir définir des représentations textuelles consistantes en chaque point d'un diagramme de formulation modale est une capacité fondamentale pour traiter de la question du travail sur les réseaux.

4. Travailler sur les diagrammes

Notre syntaxe graphique traite aussi bien les mots, les cycles, les arbres. Notre propos est à présent de définir l'objet mathématique que représentent les diagrammes de formulation modale.

L'écriture mathématique de cet objet provient du monde des fractions, des rationnels.

op (1+_) : Fraction → Fraction
op (1/_) : Fraction → Fraction

Enoncé 14 : opérateurs unaires sur les fractions

Si nous considérons les deux opérateurs unaires sur les fractions de l'énoncé n°13, nous avons deux bijections, *(1+_)* et *(1/_)*, dont l'une *(1/_)* est une convolution – i.e. nous avons *(1/(1/_))*=_. En posant que *(1)* est une fraction et en utilisant les opérations habituelles sur les entiers, le fait que n'importe quel fraction positive puisse être écrite de cette façon relève de l'algorithme d'Euclide – interprété dans le contexte des « *fractions continues* » [Wikipédia].

$$\frac{15\,625}{6\,842} = 2+\frac{1}{3+\frac{1\,019}{1\,941}} = 2+\cfrac{1}{3+\cfrac{1}{1+\cfrac{922}{1\,019}}} = \cdots = 2+\cfrac{1}{3+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{9+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{48}}}}}}}$$

Enoncé 15 : exemple de fraction continue calculable

Un diagramme de formulation modale est une valuation de cette algèbre unaire sur un ensemble calculable de nœuds. Un diagramme de formulation modale s'apparente à une forme de BDD :

op (1+_) : Noeud → Noeud (permutation)
op (1/_) : Noeud → Noeud (convolution)

Enoncé 16 : formulation modale

Comment lier cette expression algébrique à nos diagrammes ?

- Les nœuds sont les intersections entre les segments et les courbes.
- Les segments sont les orbites de convolution : l'ensemble des segments est noté $/^{Noeuds}$.
- Les courbes sont les orbites de la permutation : l'ensemble des courbes est noté $+^{Noeuds}$.

_	(1+_)	(1/_)	Valeur
001	002	001	p
002	003	002	a
003	004	003	t
004	005	004	a
005	006	005	t
006	007	006	r
007	008	007	a
008	001	008	s

Enoncé 17 : (patatras) en formulation modale

Comme toujours, cette syntaxe pas plus que la précédente ne représente vraiment l'objet concret, qui relève de la classe d'équivalence modulo le renommage des nœuds de cette instance. De la même façon, il existe une infinité de graphies possibles pour un diagramme donné, il n'empêche : que ce soit sous forme de table ou de graphies, le diagramme représenté est unique et non ambigu.

Dans notre convention d'écriture les nœuds sont explicitement de dimension nulle, comme intersection de deux objets de dimension 1 : les segments et les courbes. Cette convention est bien plus précise que la convention habituelle des graphes qui n'encapsule pas correctement la notion de nœud, cause de notre *(patatras)* de la figure n°1.

_	(1+_)	(1/_)	Valeur
001	002	001	(__)
002	009	002	u
003	007	009	(__)
004	005	007	(__)
005	006	005	i
006	004	006	o
007	008	004	(__)
008	003	008	p
009	001	003	(__)

Enoncé 17 : Diagramme n°3 sous forme de table

C'est volontairement que nous avons mis les nœuds dans l'ordre de lecture, pour montrer que l'opération qui permet de passer du diagramme n°3 ou diagramme n°4 est la première opération qui vient à l'esprit en formulation modale, la dualité, qui repose sur une composition de bijection, bijective par construction :

$$(1+\_)_{dual} = (1/\_) \circ (1+\_)$$
$$(1/\_)_{dual} = (1/\_)$$

Enoncé 18 : dualité (à gauche)

Dans l'exemple de l'énoncé n°17, nous avons bien $(1+\_)_{dual}$ qui est le successeur modulo le cardinal de l'ensemble (9) retrouvant l'idée d'un seul fil parcourant le tout.

_	(1+_)	(1/_)	(1/_)o(1+_)
001	002	001	002
002	009	002	003
003	007	009	004
004	005	007	005
005	006	005	006
006	004	006	007
007	008	004	008
008	003	008	009
009	001	003	001

Enoncé 19 : Calcul du graphe dual

Cette transformation peut se décomposer en la succession des transpositions, et la règle de « perspective » correspond à l'application d'une seule transposition.

Il n'est pas plus important de faire tourner « *perspective* » dans un sens ou dans l'autre, que le dual soit à gauche ou à droite. Le renommage a utiliser pour prouver l'unicité des diagrammes est la transposition elle-même. Cette propriété explique que nous mettions les informations « *sur* » les arêtes, puisqu'elle n'ont pas d'orientation fiable dès lors que l'on joue avec les perspectives.

5. Travailler sur XML

Si la première partie s'intéressait aux lettres pour présenter une première piste vers notre nouveau territoire, cette partie va partir d'un autre type d'énoncé : les énoncés du « *langage à balises étendues* » connu par son l' acronyme anglophone, XML, pour eXtended Markup Langage.

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Enoncé 12 : langage à balises étendues (XML)

Pour travailler sur cet objet, considérons la suite des événements de lecture liée à cet énoncé, ce qui revient à lister les appels à une API de type SAX :

```
début de « note »
        entrée dans « to »
                texte : « Tove »
        sortie de « to »
        entrée dans « from »
                texte : « Jani »
        sortie de « from »
        entrée dans « heading »
                texte : « Reminder »
        sortie de « heading »
        entrée dans « body »
                texte :  « Don't forget me this weekend! »
        sortie de « body »
fin de « note »
```

Enoncé 13 : événements de lecture (SAX)

Nous pouvons appliquer cotre recette commune à toutes les syntaxes qu'il faut intégrer dans la formulation modale, qui consiste a positionner les événements de lecture sur leur fil en liant les paires correspondant aux mêmes parenthèses.

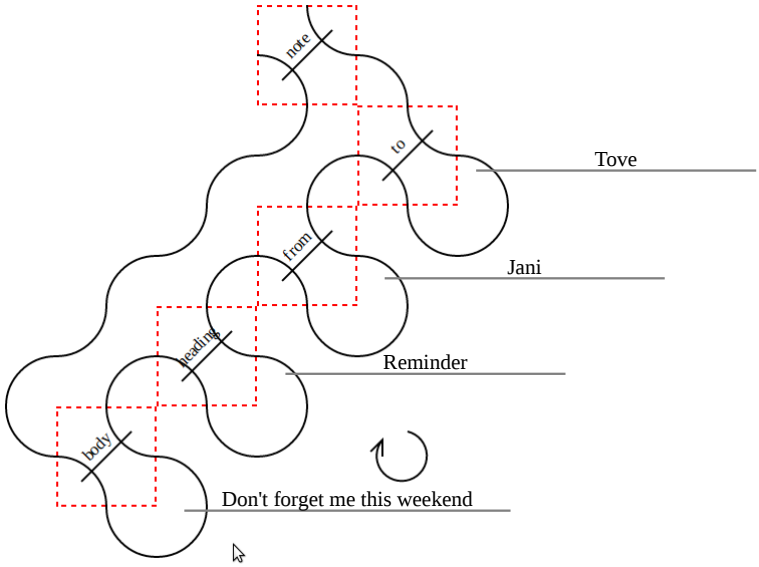


Diagramme 5 : Forme syntaxique de la « note » (SAX)

Il ne reste plus qu'à prendre la forme duale, où plus précisément appliquer la « perspective » sur tous les segments pour obtenir la forme arborescente classique, correspondant au modèle DOM.

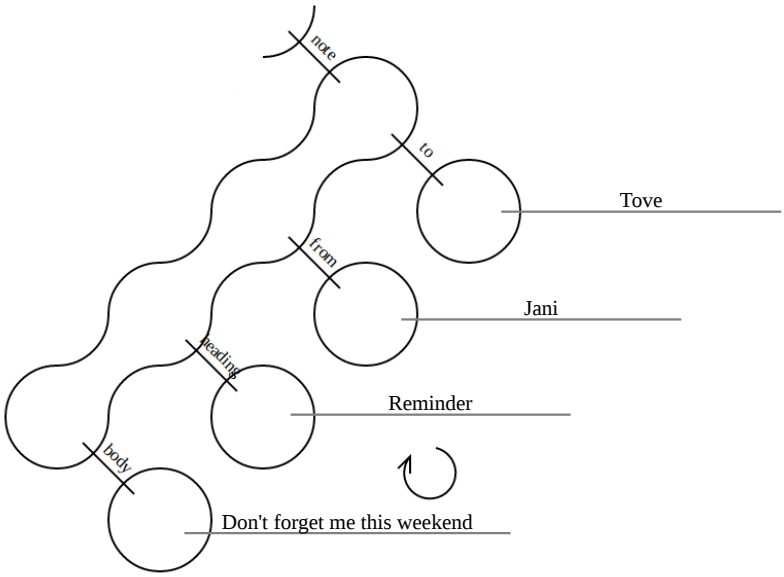


Diagramme 4 : Forme graphique de la « note » (DOM)

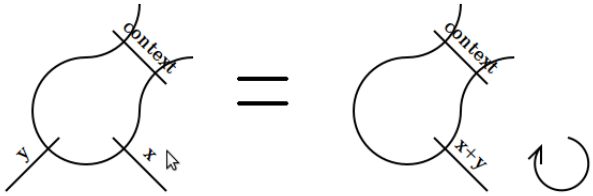
Cette dernière forme peut être mise sous forme de table, avec la même approche que pour le BDD – d'ailleurs une façon mathématique de voir XML est d'utiliser un BDD – en mettant les valeurs dans l'ordre de la lecture.

_	(1+_)	(1/_)	Valeur
001	002	001	document(note)
002	005	004	balise(to)
003	004	003	texte(Tove)
004	003	002	balise(to)
005	008	007	balise(from)
006	007	006	texte(Jani)
007	006	005	balise(from)
008	011	010	balise(heading)
009	010	009	texte(Reminder)
010	009	008	balise(heading)
011	001	013	balise(body)
012	013	012	texte(Don't forget me this weekend)
013	012	011	balise(body)

Enoncé 17 : Forme tabulaire de la « *note* » (DOM)

La formulation modale conserve comme particularité de mettre les étiquettes de valeur sur les orbites de la convolution (les « *segments* ») et donc de dupliquer les étiquettes des nœuds, cette redondance n'est pas forcément un handicap en soi.

Rajouter un dictionnaire d'attributs sur les « *segments* » ne pose aucune difficultés techniques, en revanche les nœuds texte nécessitent un minimum d'attention, car nous sommes passés de nœuds texte ne contenant que des lettres à des nœuds textes contenant des mots. Pour traiter correctement cette question nous sommes obligés de rajouter une équation de diagramme sur les nœuds texte :



Équation 1 : Concaténation libre de nœuds texte

Cette équation permet de créer des nœuds, et en particulier des nœuds textes vide. C'est ici que l'hypothèse de la calculabilité de la structure est importante. Si nous voulons avoir des chaînes de nœuds arbitrairement longues autour des éléments de texte, c'est que nous les calculons pour une intention précise, et nous verrons qu'elle existe.

6. Travailler sur les graphes

Nous nous sommes concentrés pour le moment sur des illustrations (mots, cycles, BDD, XML) relevant de l'imaginaire informatique. Nous allons maintenant voir ce notre syntaxe peut amener à la théorie des graphes, sans appesantir car ce n'est pas notre spécialité, mais juste pour explorer. Nous commençons cette exploration par un problème classique formalisé par Leonard Euler dans la seconde moitié du XVIIIème siècle : les Ponts de Konigsberg, du nom d'une ville de Prusse Orientale, devenue russe depuis la dernière guerre sous le nom de Kaliningrad, et enclavé après la chute du mur de Berlin par l'indépendance de la Lituanie. Cette ville est située à un endroit ou deux bras de la rivière Pregolya se rejoignent après avoir délimité plusieurs îles entre eux, et le problème concernent les deux dernières.

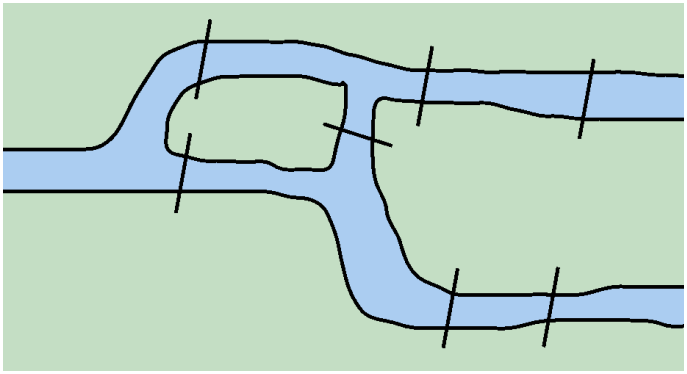


Figure 3 : Les ponts de Konigsberg

Le problème énoncé par Euler, était de savoir s'il pouvait faire une promenade (partir et revenir du même point) en passant une fois et une seule par chaque pont. La réponse est non, il est nécessaire que chaque composante de terre ferme (en vert) est un nombre pair de pont, sinon vous êtes nécessairement bloqué. Que cela forme une condition suffisante est un peu plus difficile. Une démonstration complète est donnée au titre « *graphe eulérien* » de Wikipédia.

Cette résolution relève du schéma classique de résolution, qui procède d'abord par une idéalisation, puis un calcul et enfin une interprétation.

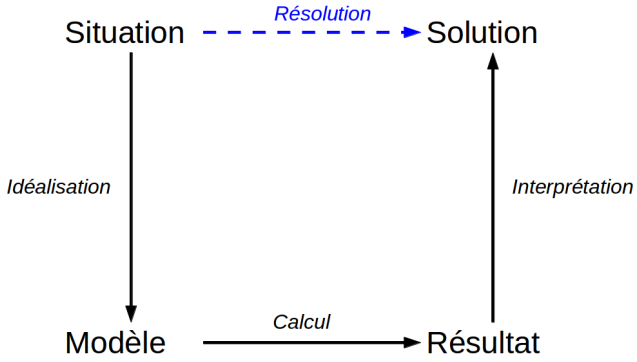
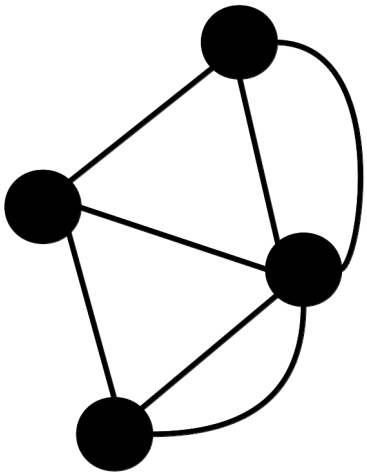


Figure 4 : Mécanisme de résolution

Par rapport aux approches classiques de ce problème, nous allons bien sûr porter une attention toute particulière à la question de l'idéalisation. Cette idéalisation est classiquement réalisée par identification des « composantes de terre » à des nœuds et les ponts à des arêtes d'un graphe non orienté.



Graphe 4 : Modèle classique des Ponts de Koenigsberg

Le problème d'Euler s'exprime par la possibilité d'écrire une bijection sur les arêtes n'ayant qu'un seul cycle (tiens, tiens...). La démonstration de la suffisance s'exprime d'ailleurs par composition de tels cycles. Mais cette démonstration est tout de même singulière : la nature mathématique des arêtes n'y est pas explicitée. Et ce n'est pas un cas particulier : il y a vraiment une anguille sous la roche.

Il faut dire que la définition formelle des graphes non-orientés est tout sauf triviale, la plus sûre est de considérer un graphe non orienté comme une valuation de l'algèbre avec équations donnée ci-dessous.

op source : Arête → Sommet (fonction)
op cible : Arête → Sommet (fonction)
op paire : Arête → Arête (convolution)
eq source(paire(x))=cible(x)

Enoncé 16 : signature des graphes non-orientés

Avec cette définition, il est possible de créer un vrai modèle du graphe n°4, mais ceci ne fait jamais partie de la présentation standard de la théorie des graphes, avouez tout de même que ceci n'est pas une preuve manifeste de solidité.

_	source(_)	cible(_)	paire(_)
001	001	002	002
002	002	001	001
003	002	003	004
004	003	002	003
005	002	003	006
006	003	002	005
007	004	003	008
008	003	004	007
009	004	003	010
010	003	004	009
011	001	004	012
012	004	001	011
013	001	003	014
014	003	001	013

Enoncé 16 : Modéle des Ponts de Koenigsberg

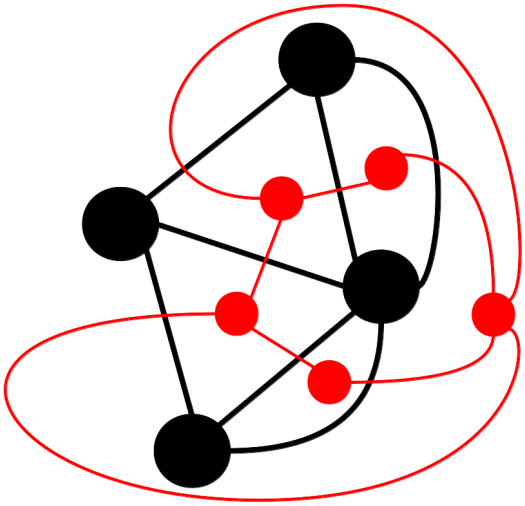
Remarquez, cela se comprend : c'est pas très sexy. Et puis surtout cela impose de penser que les arêtes sont des classes d'équivalence de la convolution *paire(_)*. Ce qui signifie en particulier que le parcours eulérien ne couvre qu'une moitié des arêtes syntaxiques... Dérangeant. Mieux vaut ne pas en parler, et la stratégie des articles de Wikipédia est généralement retenue par les théoriciens des graphes : ne pas faire des graphes un objet avec une « *vraie* » syntaxe. Nous avons évoqué ce que cette réalité « *sous le tapis* » nous a coûté en terme de carrière dans la recherche.

Néanmoins, la théorie des graphes peut faire appels aux matrices d'adjacence, mais ceci concerne techniquement les « *relations* » et non les « *graphes* » qui peuvent avoir plusieurs arêtes avec même source et même destination, comme le graphe historique d'Euler. Qui mentionne que les matrices d'adjacence ne concernent pas les graphes authentiques ?

♦

Une autre notion pour laquelle le flou artistique domine est la notion de dualité, qui ne se comprend qu'avec des graphes non-dirigés plongés dans une surface connexe – ce qui est toujours le cas puisque l'aspect syntaxique est souvent oblitéré.

Les nœuds du graphe dual sont les composantes connexes du plan et chaque arête du graphe initial relie dans le dual les composantes qui la bordent. Cela fait un paquet de définitions topologiques a adjoindre, souvent non élucidées. Néanmoins, et c'est la force et la faiblesse de la théorie des graphes, le cortex pallie à ses insuffisances par son extraordinaire capacité a interpréter les formes.



Graphe 4 : Dual classique (en rouge) des Ponts de Koenigsberg

La formulation modale fait un seule hypothèse topologique pour résoudre cette question, elle identifie les intersections entre cercle et segments comme les nœuds. A partir de là, résoudre le dual est un calcul.

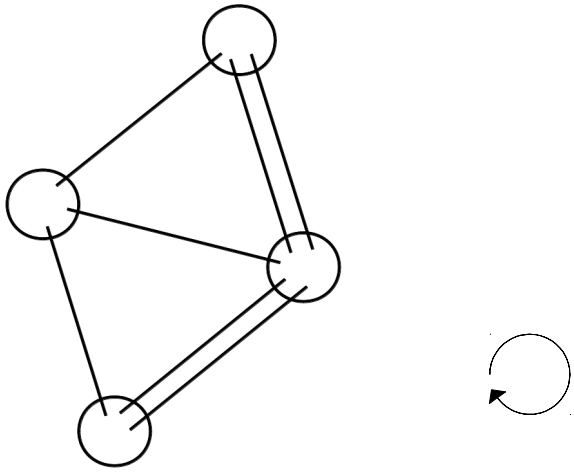


Diagramme 4 : Idéalisation classique des Ponts de Koenigsberg

Avant de nous intéresser à la dualité, nous allons montrer que nous étendons bien les graphes non dirigés en explicitant pour tout diagramme un graphe support unique (modulo isomorphisme, a*s usual*).

Sommets = $+^{Noeuds}$ (= l'ensemble des orbites courbes)
Arêtes = Noeuds
source(_) = orbite(+,_) (= son orbite courbe)
cible(_) = orbite(+, 1/_) (= l'orbite courbe de son inverse)

Enoncé 16 : Graphe non-orienté support d'un diagramme

Revenons à la dualité, nous avons pris le parti dans ce document de résoudre cette question de façon uniquement graphique, pour cela nous devons mettre le diagramme n°4 sous forme pavée et appliquer la règle perspective à toute les arêtes.

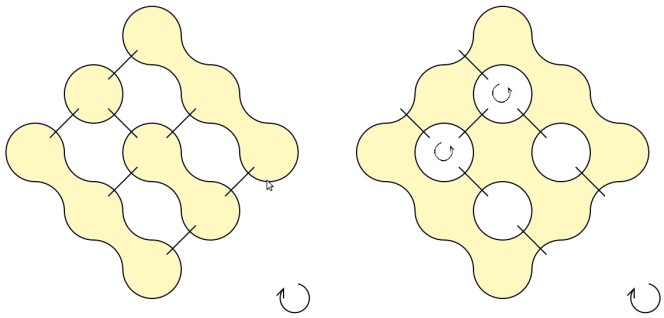


Diagramme 4 : Dualité de l'idéalisation classique

Techniquement, la « *planarité* » intervient ici. L'indication de « spin » vaut pour les courbes où le blanc est à l'extérieur et la couleur à l'intérieur. Dans le cas contraire, l'indication se comprend comme étant dans l'autre sens, comme indiqué sur le diagramme.

♦

Nous n'en avons pas fini avec le problème des ponts. Car la formulation modale reprend jusqu'à l'idéalisation du problème. La figure n°3 est presque un diagramme de formulation modale, il suffit de mettre des « *nœuds documents* » en « *amont* » et en « *aval* » pour « *borner le fleuve* » et nous obtenons un diagramme dessiné à la main syntaxiquement correct.

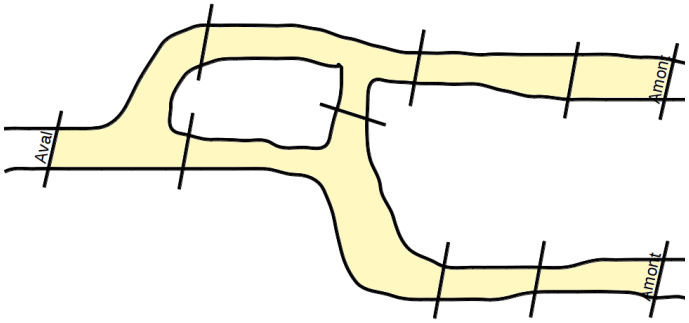


Diagramme 4 : Idéalisation alternative des Ponts de Koenigsberg

Considérons à présent la dualité de ce diagramme.

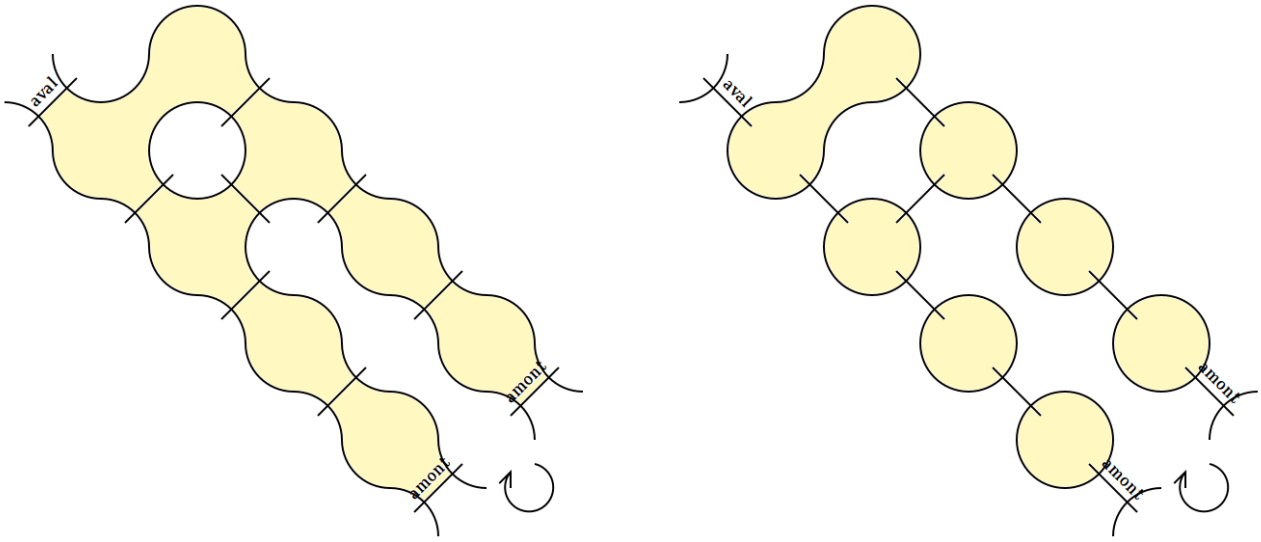


Diagramme 4 : Dualité alternative des Ponts de Koenigsberg

Arrêtons nous un instant sur la figure de droite. Si nous l'interprétons les nœuds documents comme de « *vrais* » nœud document XML, nous obtenons une illustration de notre motivation initiale énoncée en 2005 : disposer d'un modèle de données capable de fournir différents points de vue sous forme XML, chacun pouvant être interprété comme une « *perspective* » différente de la même structure. Ce « *cahier des charges* » est à la source de la première mouture de la formulation modale en 2008.

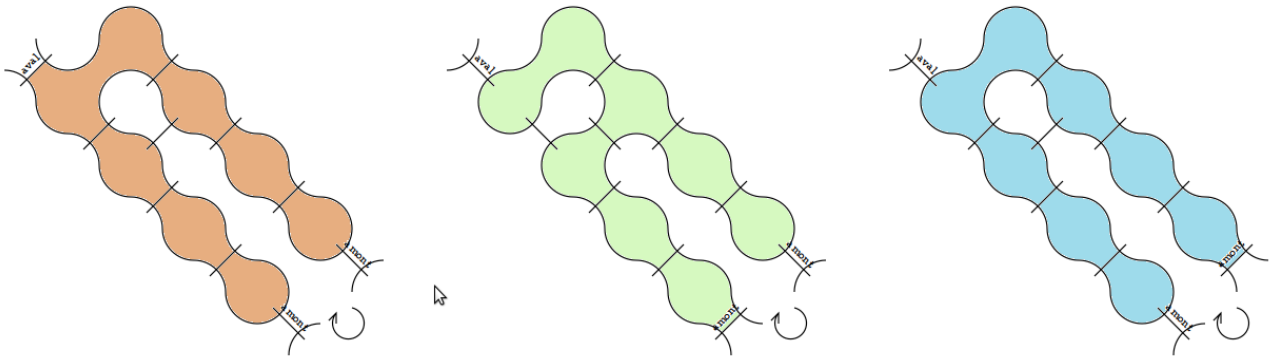


Diagramme 4 : Perspectives XML couvrantes

Ceci fut le premier résultat de notre programme de R&D financé par nos actionnaires et une subvention OSEO en 2010-2011 : un diagramme de formulation modale multi-document prend tout son sens avec les « *perspectives* », et donc avec les « *balises* » sur les « *arêtes* ».

Dans ce contexte de coopération, chaque utilisateur (un « *aval* » et deux « *amonts* ») perçoit un document XML différent peut le modifier et tous les autres utilisateurs sont immédiatement impactés, sans condition, ni autorisation. Nous avons appelé ce type de coopération de « *coproduction libre* », c'est en réalité assez inconfortable dans la vie réelle, chacun éprouvant la nécessité de travailler sur un espace qu'il maîtrise, soit directement, soit par délégation. Cette capacité est fourni dans les systèmes réels par la « *gestion de configuration* ».

7. Travailler sur les transformations

Pour pouvoir fournir un support à la coopération, nous devons pouvoir fournir une « *gestion de configuration* » . La gestion de la configuration repose sur le fait que les transformations des mots peuvent être des mots.

La technique par pavage permet d'envisager des transformations qui soient des diagrammes en faisant intervenir une autre couleur de trait.

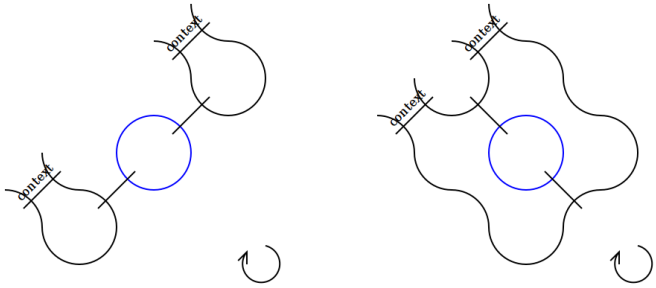


Diagramme 4 : Transformation

Détaillons les modèles de ces deux diagrammes, uniquement leur composante de couleur noire.

_	(1+_)	(1/_)	Valeur
001	002	001	document(context)
002	001	002	texte()
003	004	003	texte()
004	003	004	document(context)

_	(1+_)	(1/_)	Valeur
001	002	001	document(context)
002	004	002	texte()
003	001	003	texte()
004	003	004	document(context)

Enoncé 16 : Modèle des deux diagrammes

La seule différence entre les deux concerne la permutation, pour passer de l'une à l'autre, il faut les composer par la transposition des nœuds *002* et *003*. La courbe bleue fournit une illustration tout à fait satisfaisante de cette composition, les nœuds impliqués étant les « *inverses* » des « *nœuds* » intersection de la courbe bleue et des segments noirs.

Comme nous l'avons suggéré dans la partie concernant XML, les nœuds *texte()* vide peuvent être crée à la demande dans les textes, permettant d'offrir des points de construction nécessaire à l'application de transformations de ce type. Nous étendons cette équation à l'ensemble des nœuds pour disposer de points d'interaction partout où c'est nécessaire, et pouvant être enlevés à l'issu de la transformation.

[...]

8. Travailler sur la persistance

Les perspectives couvrantes permettent d'envisager un support de persistance capable de gérer le problème de ramasse-miettes.

[...]