MATHÉMATIQUES ET SCIENCES HUMAINES

P. ROSENSTIEHL

Labyrinthologie mathématique (I)

Mathématiques et sciences humaines, tome 33 (1971), p. 5-32

<http://www.numdam.org/item?id=MSH_1971__33__5_0>

Math. Sci. hum. (9e année, n° 33, 1971, p. 5-32)

LABYRINTHOLOGIE MATHÉMATIQUE (I)

par

P. ROSENSTIEHL [1]

« Un labyrinthe est une chose faite à dessein pour confondre les hommes »
Jorge Luis Borges [2].

INTRODUCTION

La labyrinthologie mathématique [3] se propose de résoudre les labyrinthes dans un langage formel. Cet article s'explique sur ce que l'on peut entendre par résolution.

Le voyageur égaré, nous dit la légende, n'a pas la carte du labyrinthe; il avance à tâtons et pourtant il réussit à tout explorer sans jamais reprendre deux fois la même voie dans le même sens. On note l'économie souhaitée de l'exploration, et aussi les moyens limités de l'explorateur; nous garderons toujours en tête ces deux impératifs. Les règles qu'il peut appliquer pour parvenir à ses fins avec si peu de moyens sont cependant multiples; une étude systématique des règles possibles donne une modernité inattendue à ce morceau illustre des récréations mathématiques : elle engendre des écritures de « listes » utiles à nombreux algorithmes de graphe, aux « procédures de recherche » en général, et aux « structures de données » des informaticiens. Il en résulte entre autres l'identification des isthmes et des points d'articulation du labyrinthe ; la construction d'un ordre total des lignes d'une part et des points du labyrinthe d'autre part, sont compatibles dans une certaine mesure avec l'adjacence de ces objets.

Le voyageur égaré est trop légendaire; réduisons-le au statut de simple signal, tel le repère qui se déplace dans un organigramme logique au fur et à mesure d'un calcul; et affectons l'intelligence, que le voyageur consacrait à choisir sa route, aux carrefours du labyrinthe eux-mêmes, dénommés pour la circonstance « automates finis ». On imagine la voiture folle visitant Paris, obéissant aux gestes péremptoires de sergents de ville placés aux carrefours et ayant tous appris la même leçon. Résoudre le labyrinthe, ce sera définir cet automate standard de carrefour, indépendamment de la dimension du labyrinthe, de sorte que la voiture touriste emprunte toute rue une fois dans chaque sens. Est-ce possible? La réponse est oui.

La généralisation est aisée : les automates ne connaissent qu'eux-mêmes et leurs voisins dans le réseau, ils calculent tous en parallèle. On leur demande d'afficher sur l'ensemble du labyrinthe par leurs états en fin de calcul, le plus court chemin d'un point à un autre, un arbre maximal, l'existence d'un circuit, un cycle eulérien ou un cycle hamiltonien s'il en est, etc. Chacun de ces problèmes de labyrinthe

---

1. Centre de Mathématique Sociale, EPHE, VIe Section.

2. « L'immortel », *Labyrinthes*, Paris, Gallimard, 1953.

3. Les méthodes de la labyrinthologie relèvent bien entendu de nombreux domaines autres que la mathématique, qui ne sont pas considérés dans cet article : la poétique, la littérature potentielle, la génétique, la psychanalyse, la psychologie, etc. Citons dans ce dernier domaine et à titre d'exemple, l'étude sur la perception de l'espace de la circulation parisienne chez les chauffeurs de taxi, par J. Pailhous [10]. Quant à l'histoire des labyrinthes célèbres de toutes les civilisations, on pourra y accéder à partir des notes bibliographiques de Rouse Ball [16]. Selon P. Lévy, le labyrinthe est un fait originairement égyptien qui repose sur des conceptions eschatologiques et initiatiques; il aurait été conçu à partir des premiers sanctuaires rupestres et de conceptions acquises par l'obstétrique.

est dit alors « soluble par automates finis ». On entrevoit les débouchés de telles constructions : la communication dans un réseau biologique, sociologique ou technologique, ou encore l'analyse et la conception de systèmes — organisme vivant, société humaine ou animale, ou machine — où toute activité se développe régionalement sans aucune organisation centralisatrice, et où l'ensemble pourtant se présente comme un tout cohérent.

La résolution d'un labyrinthe prend donc ici un double aspect, qui déjà donnait à l'historiette récréative son caractère fascinant.

D'une part, il s'agit de trouver une route dans un dédale de routes — on dira techniquement une flèche dans une catégorie labyrinthe — et comme la sagesse conseille pour construire sa route pas à pas de savoir aussi revenir sur ses pas (une démarche ordinaire parmi d'autres), tout parcours sera inversible — on aura techniquement une flèche de groupoïde, et le plus souvent même un mot d'un langage de Dyck.

D'autre part, il s'agit de définir la taille de mémoire utile, les tables de transformation de l'information, bref les automates capables de construire cette flèche ou ce mot, de les calculer, et ceci aussi bien pour un automate unique fonctionnant sur le mode centralisé habituel des algorithmes classiques que pour des automates dispersés dans le labyrinthe même que l'on veut résoudre.

Même « résolus », les labyrinthes ne cesseront de nous étonner par les problèmes fondamentaux qu'ils suggèrent à la théorie des algorithmes : en anglais un seul mot "maze" pour dire labyrinthe et ébahissement [1].

Plan de la partie I : Algorithmes pour résoudre un labyrinthe

1. Sortir du labyrinthe sans nouer le fil.
   - 1.1. Écriture : des lignes et pas de points.
   - 1.2. L'effacement des lettres inverses par repli.
   - 1.3. Les mots à minimum et maximum de replis.
2. Sortir du labyrinthe avec ou sans nœuds : les mots de Tarry.
   - 2.1. Arbre d'entrée et arbre de sortie d'un mot net.
   - 2.2. Les mots de Tarry d'un labyrinthe.
3. Quelques résultats conséquents à la résolution d'un labyrinthe.
   - 3.1. L'arbre d'un graphe, base de cycles.
   - 3.2. Les éternels Ponts de Königsberg.
   - 3.3. Quelques énoncés de type hamiltoniens.

PARTIE I : ALGORITHMES POUR RÉSOUDRE UN LABYRINTHE

Dans la présente partie, on étudie les diverses règles d'exploration d'un labyrinthe. En partie II, on définira les automates capables de choisir une exploration.

1. SORTIR DU LABYRINTHE SANS NOUER LE FIL

Dans le présent chapitre on étudie celles des explorations qui, comme le fil d'Ariane, se replient sur elles-mêmes, si bien que le fil déroulé le long de toute ligne nouvellement explorée, se trouve rembobiné complètement en fin d'exploration : le fil n'est jamais noué.

---

1. L'apologue de l'homme perdu dans la forêt figure en bonne place chez G. Th. Guilbaud, dans son chapitre « Pilotes, stratèges et joueurs » de *La Cybernétique* [5] ; l'auteur y lance un appel : « Il faudrait étudier de près la notion même de labyrinthe ».

Au chapitre 2, on se libérera de cette contrainte de repli — on n'hésitera pas à faire des nœuds autour des blocs de la ville — mais d'une certaine manière; au chapitre 3, on présentera divers problèmes de graphes résolus par les règles d'exploration étudiées.

1.1. Écriture : des lignes et pas de point

Un labyrinthe est un ensemble de points et de lignes entre ces points. Entre deux points figurent zéro, une, ou plusieurs lignes. On ne dit rien sur la situation des points, ni sur un espace où ils seraient plongés : les points sont donnés dans l'abstrait. De même pour les lignes; on ne sait pas ce que signifient longueurs, entrelacs; une ligne est un objet qui a deux bouts — confondus avec deux points, distincts ou non — donc un objet à deux sens; indiquons l'un et il y aura l'autre : tout cela est représentable sur la feuille de papier comme on l'a fait dans un cas particulier sur la figure 1, où l'un des sens d'une ligne donnée est « le sens de la flèche », l'autre « le sens contraire de la flèche ». Bien entendu, pas de sens interdit ! On prendra l'un comme l'autre, sauf spécifications spéciales, le choix des flèches étant pour l'instant arbitraire. Explorer ce sera parcourir à la suite des lignes qui se touchent. D'où la disparition des points dans les écritures qui vont suivre. Les parcours de lignes dans l'un et l'autre sens vont être notés par des « lettres », dites inverses. Une exploration sera un mot fait de ces lettres.

1.1.1. Définition des labyrinthes

Un labyrinthe L est un ensemble $\mathscr{A}$ fini non vide et de cardinal pair (appelé ensemble des lettres de L ou alphabet), ***muni,***

— ***d'une involution i sans point fixe*** (dite de changement de sens) notée par l'opérateur prime :

$$l \in \mathscr{A} \qquad \text{alors} \qquad l' \in \mathscr{A}, \qquad l' \neq l \qquad \text{et} \qquad (l')' = l.$$

— ***d'une relation d'équivalence*** (appelée « a même droite que » et dont les classes sont « les points » du labyrinthe, les lettres d'une même classe ayant comme droite un même point) notée par l'application d'appartenance des lettres dans l'ensemble X des classes :

$$\begin{aligned} &\mathrm{d} : \mathscr{A} \to \mathrm{X} \\ &\mathrm{d}\,(l) = \mathrm{d}\,(\mathrm{k}) \end{aligned}$$

pour « la droite de l est égale à la droite de k », ou « l a même droite que k ».

Remarque. Une lettre l aboutit à d (l); d'où part-elle ? Évidemment du point où aboutit sa lettre inverse, c'est-à-dire d (l'). Il sera donc commode de parler de la gauche d'une lettre en posant :

$$\begin{aligned} &\mathrm{g} : \mathscr{A} \to \mathrm{X} \\ \text{avec} \qquad &\mathrm{g}\,(l) = \mathrm{d}\,(l') \qquad\qquad (\text{pour } l \in \mathscr{A}). \end{aligned}$$

Il revient au même de se donner un labyrinthe par son triplet ($\mathscr{A}$, i, d) ou son triplet ($\mathscr{A}$, i, g).

On a noté qu'on appelle *points du labyrinthe* [1] les objets qui sont droite ou gauche de lettres; on appellera *lignes du labyrinthe* les paires $\{l\ l'\}$ de lettres inverses. X désigne l'ensemble des points, U l'ensemble des lignes du labyrinthe.

Un labyrinthe ($\mathscr{A}$, i, d) ***est dit orienté, par la donnée d'une partie*** $\mathscr{A}^+$ ***de*** $\mathscr{A}$, ***telle que :***

$$l \in \mathscr{A}^+ \Leftrightarrow l' \notin \mathscr{A}^+.$$

1. Au lieu de labyrinthe, points et lignes, nous disons le plus souvent ailleurs graphe, sommets et arcs, selon le vocabulaire de Berge [2] ; la terminologie retenue ici — et d'ailleurs qu'importe — ne vise pas tant à produire un effet d'archaïsme qu'à mettre en valeur notre sujet de cheminement de type labyrinthologique en fonction de données locales.

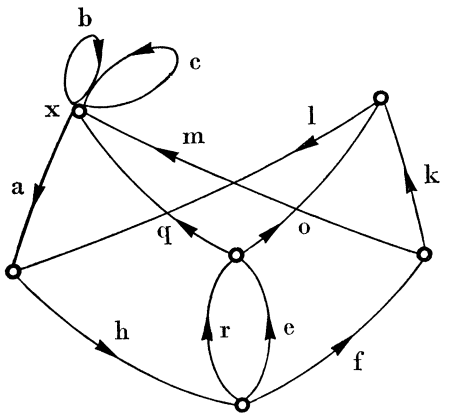


Fig. 1. Labyrinthe orienté connexe L

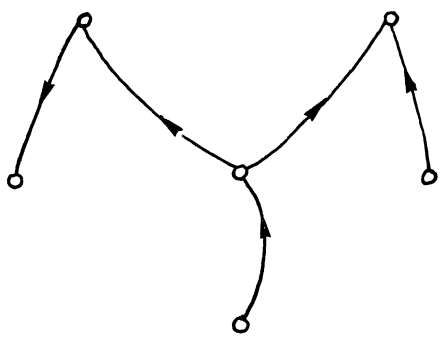


Fig. 2. Labyrinthe, arbre du labyrinthe de la figure 1

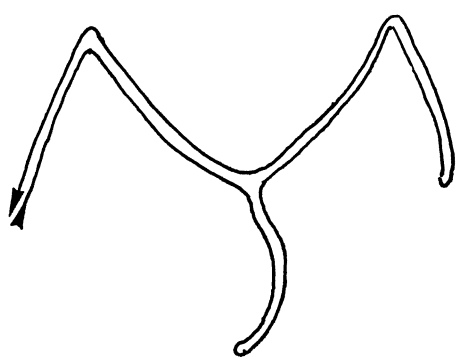


Fig. 3. Mot net d'arbre

Notre figure 1 représente un labyrinthe orienté :

« parcourir une ligne dans le sens de la flèche » représente un élément de $\mathscr{A}^+$,

« parcourir une ligne dans le sens contraire de la flèche », représente un élément de $\mathscr{A}^-$, le complémentaire de $\mathscr{A}^+$.

1.1.2. Mots et mots nets de labyrinthe

Avec les lettres de labyrinthe, on écrit des mots de labyrinthe. Commençons par un exercice. Sur la figure 1, je lis les mots :

ahre′
ahre′ro
rolh
f′h′a′.

Sans regarder la figure, j'ajoute d'autres mots aux mots ci-dessus :

ah, hr, re′, e′
er′h′a′
re′re′re′r
rolhre′r.

Bien entendu, il faudra, pour opérer en toutes circonstances, se munir de mots vides : un mot vide par sommet $x \in X$; on le notera Λ_x. Λ_x correspond à « ne rien faire alors qu'on se trouve en x » :

$$a\ \Lambda_y,\ \Lambda_x a,\ \Lambda_x\ \Lambda_x\ \Lambda_x a,\ \Lambda_x$$

sont des mots.

Nous étendons les applications gauche et droite comme on le devine :

$$d, g : \mathscr{A} \cup (\Lambda_x)_{x\in X} \to X \qquad \text{avec} \qquad d(\Lambda_x) = g(\Lambda_x) = x.$$

Lexique $\mathscr{L}$ des mots d'un labyrinthe. On appelle mot d'un labyrinthe $(\mathscr{A}, i, d)$ tout objet appartenant à l'une des trois classes suivantes :

— Λ_x avec $x \in X$, appelés mots vides du labyrinthe,

— l avec $l \in \mathscr{A}$, appelés mots-lettres du labyrinthe,

— $\sigma = l_1 \dots l_r\, l_{r+1} \dots l_p$ avec $l_r \in \mathscr{A} \cup (\Lambda_x)_{x\in X}$

et avec $d(l_r) = g(l_{r+1})$, pour $r = 1, 2, \dots, p-1$,

appelés mots de longueur p du labyrinthe.

La présence de l_r dans σ sera souvent écrite $l_r \in \sigma$, et si $l_r \in \mathscr{A}$ on dira que σ a au moins *une occurrence* de la lettre l_r. Le nombre d'occurrences de lettres dans un mot, augmenté du nombre de mots vides qui y figurent, est égal à la longueur de ce mot.

l_r et l_{r+1} sont dites contiguës dans σ.

L'ensemble des mots [1] du labyrinthe est noté $\mathscr{L}$ et appelé *lexique du labyrinthe.* Nous étendons les applications *gauche et droite* à tous les mots de labyrinthe comme on devine :

---

1. Notre écriture en « mots de lettres » des chaînes ou chemins dans un graphe n'est pas usuelle ; on pourra noter à ce sujet l'embarras des graphistes auteurs de manuels tels König, Berge, Tutte, Harari, Ore et Roy ; les écritures y sont diverses et hétérogènes le plus souvent. Nous montrons dans [15] l'opposition mot et vecteur dans l'étude des graphes.

$d, g : \mathscr{L} \to X$ avec $g(\sigma) = g(l_1)$ et $d(\sigma) = d(l_p)$.

On dira qu'un mot σ est *cyclique* si : $g(\sigma) = d(\sigma)$.

Concaténer : Concaténer $\sigma_1, \sigma_2 \in \mathscr{L}$, deux mots tels que : $d(\sigma_1) = g(\sigma_2)$, c'est écrire l'élément σ de $\mathscr{L}$ obtenu en mettant bout à bout σ_1 et σ_2 :

$$\sigma = \sigma_1\sigma_2 \qquad (d(\sigma_1) = g(\sigma_2)).$$

Les mots vides sont les éléments neutres de la concaténation : pour tout mot $\sigma \in \mathscr{L}$, on pose :

$$\sigma = \Lambda_{g(\sigma)}\,\sigma = \sigma\,\Lambda_{d(\sigma)}.$$

Il faut comprendre qu'il s'agit de trois écritures du même mot, trois écritures de longueurs différentes, mais avec le même nombre d'occurrences. On dira encore qu'il s'agit de trois mots égaux.

Résumons les précédentes définitions.

Théorème 0 : *Le lexique $\mathscr{L}$ d'un labyrinthe* L *est une catégorie dont les flèches sont les mots de* L, *les objets les mots vides associés aux points de* L.

Facteurs : Soit un mot σ s'écrivant : $\sigma = \sigma_1, \sigma_2, \sigma_3$, (grâce aux mots vides on peut mettre tout mot σ sous cette forme !) $\sigma_1, \sigma_2, \sigma_3$ sont appelés facteurs de σ, σ_1 facteur gauche, σ_3 facteur droit.

Labyrinthe connexe : Un labyrinthe est dit connexe si pour tout $l, k \in \mathscr{A}$, il existe $\sigma \in \mathscr{L}$ admettant l comme facteur gauche et k comme facteur droit.

Mots nets : Un mot σ d'un labyrinthe $(\mathscr{A}, i, d)$ est dit net, si dans σ figure une occurrence exactement de chaque lettre de $\mathscr{A}$.

On reconnaît dans le mot net une exploration qui emprunte chaque ligne du labyrinthe une fois dans chaque sens. D'où une première formulation du problème que l'on veut résoudre : construire un mot net d'un labyrinthe L donné.

Propriétés des mots nets

Théorème I : *Tout mot net est cyclique.*

Soit un labyrinthe $(\mathscr{A}, i, d)$ à $2m$ lettres comportant un mot net :

$$\sigma = l_1 \ldots l_r\, l_{r+1} \ldots l_{2m}.$$

Étant donné que pour tout point $x \in X$, il existe autant de lettres de $\mathscr{A}$ de gauche x que de lettres de droite x, toute injection j dans $\mathscr{A}$ définie pour $2m - 1$ lettres de $\mathscr{A}$ seulement, et satisfaisant $d(l) = g(j(l))$ pour tout l, se complémente à $\mathscr{A}$ nécessairement et de façon unique avec la même condition. (Voir en particulier le cas $j = i$.)

Revenons au mot net σ, et posons :

$$j(l_r) = l_{r+1}, \qquad \text{pour} \qquad r = 1, 2, \ldots, 2m - 1.$$

Il s'ensuit par complémentation que nécessairement :

$$j(l_{2m}) = l_1,$$

et avec :

$$d(l_{2m}) = g(l_1),$$

c'est-à-dire :

$$d(\sigma) = g(\sigma).$$

Théorème II : ***Tout labyrinthe admettant un mot net est connexe.***

Soit un labyrinthe d'alphabet $\mathscr{A}$ et de lexique $\mathscr{L}$. Soit σ un mot net de $\mathscr{L}$. Comme σ est cyclique, on a : $\sigma\sigma \in \mathscr{L}$. Pour tout : l, k, $\in \mathscr{A}$, il existe un facteur de $\sigma\sigma$ admettant l comme facteur gauche, k comme facteur droit. C.Q.F.D.

La réciproque, à savoir tout labyrinthe connexe admet un mot net, sera démontrée en I-1.3.2. (théorème IV).

1.2. L'EFFACEMENT DES LETTRES INVERSES PAR REPLIS

1.2.1. Exercice de repli sur un arbre

Considérons la figure 2. On y reconnaît un arbre; une définition en sera donnée par la suite, mais restons à présent au niveau intuitif de la figure. Partons de la lettre a par exemple. On écrit un mot net :

aa′q′e′eok′ko′q (voir figure 3)

ou bien :

aa′q′ok′ko′e′eq.

Regardons dans ces mots la position des lettres. Une lettre et sa lettre inverse peuvent être contiguës; c'est le cas, pour le premier mot, de a et a′.

Si elles ne sont pas contiguës, telles q et q′ dans le premier mot, regardons qui les séparent; o mais aussi o′, e mais aussi e′, k mais aussi k′ :

Si, dans le mot net d'un arbre, entre l et l′ apparaît la lettre e, alors e′ y apparaît aussi.

Dans le cas de deux lettres inverses contiguës, telles e′ et e, on dit qu'il y a *repli* du mot en e′ ou en d (e′). Sur un arbre les replis de mots nets n'ont lieu qu'aux points extrêmes de l'arbre, c'est-à-dire aux points qui ne sont gauche que d'une seule lettre :

Les replis dans les mots nets d'un arbre n'apparaissent que par nécessité.

Pour un point x autre que le point de départ, la première lettre (resp. dernière lettre) apparue dans un mot σ et de droite x (resp. de gauche x) est appelée lettre d'entrée (resp. lettre de sortie). On constate une troisième propriété :

Lettre d'entrée et lettre de sortie d'un même point sont inverses dans les mots nets d'arbre.

Les trois propriétés ci-dessus posées intuitivement pour les mots d'un arbre construits sur une figure, nous mettent la puce à l'oreille; tout est à peu près dit si l'on a vu qu'un labyrinthe quelconque (connexe) est un arbre dont on aurait soudé certains points. N'allons cependant pas trop vite, car si l'arbre d'un labyrinthe quelconque apparaîtra dans nos calculs, ce ne sera pas comme une donnée de départ, mais au contraire comme un résultat.

1.2.2. Mots réduits et mots neutres

Soient un labyrinthe L et son lexique $\mathscr{L}$.

Donnons-nous dans $\mathscr{L}$ une transformation telle que dans un mot, deux lettres inverses contiguës puissent être effacées (dans cette nouvelle algèbre ces lettres se neutralisent; sur les deux mots nets ci-dessus, en particulier, on dira : le fil déroulé en e′ est rembobiné en e).

Équivalence de deux mots

— Deux mots égaux ρ et τ sont dits équivalents : $(\rho = \tau) \Rightarrow (\rho \simeq \tau)$.

On a donc toujours $\rho \simeq \rho$, pour tout $\rho \in \mathscr{L}$.

— Deux mots de la forme $\alpha l l' \beta$ et $\alpha\ \beta$, où $l \in \mathscr{A}$ et $\alpha, \beta \in \mathscr{L}$, sont dits *voisins*.

En particulier si $g(l) = x$, c'est-à-dire si $\Lambda_x\, l = l$, alors ll' et Λ_x sont voisins.

— On définit maintenant sur $\mathscr{L}$ la relation d'équivalence suivante : deux mots ρ et τ sont dits *équivalents* s'ils sont égaux, ou s'il existe une suite de $(p+1)$ mots, p entier quelconque, σ_i $(i = 0, 1, \ldots, p)$, avec $\sigma_0 = \sigma$ et $\sigma_p = \tau$ et telle que σ_i soit voisin de σ_{i-1} pour $i = 1, 2, \ldots, p$. On écrit alors :

$$\rho \simeq \tau.$$

On va vérifier que la relation $\simeq$ est compatible avec la concaténation dans $\mathscr{L}$.

Théorème III : *Deux mots équivalents d'un labyrinthe* L *ont mêmes éléments neutres.*

Ceci s'écrit aussi en recourant aux applications droite et gauche :

$$(\rho \simeq \tau) \Rightarrow g(\rho) = g(\tau) \qquad \text{et} \qquad d(\rho) = d(\tau).$$

La proposition est vraie pour des mots égaux et pour deux mots équivalents voisins : $\alpha l l' \beta$ et $\alpha\ \beta$.

Or si $\rho \simeq \tau$, et si ρ et τ ne sont ni égaux ni voisins, il existe par définition une chaîne finie d'éléments équivalents voisins joignant ρ à τ le long de laquelle la gauche et la droite restent invariantes.

On a donc, si Λ_x et Λ_y sont respectivement l'élément neutre à gauche et à droite de ρ :

$$(\rho \simeq \tau) \Rightarrow (\Lambda_x\, \rho \simeq \Lambda_x\, \tau \qquad \text{et} \qquad \rho\, \Lambda_y \simeq \tau\, \Lambda_y). \qquad \text{C.Q.F.D.}$$

Il s'en suit que pour :

$$\rho, \sigma_1, \sigma_2, \sigma_1\rho, \rho\sigma_2, \tau \in \mathscr{L},$$

on a :

$$(\rho \simeq \tau) \Rightarrow (\sigma_1\, \rho \simeq \sigma_1\, \tau \qquad \text{et} \qquad \rho\, \sigma_2 \simeq \tau\, \sigma_2).$$

Mots inverses : A tout élément σ de $\mathscr{L}$ on associe un élément σ' appelé son « mot inverse » selon la règle suivante :

$$\begin{aligned} (\Lambda_x)' &= \Lambda_x & x &\in X, \\ (l)' &= l' & l &\in \mathscr{L}, \\ (\sigma_1\, \sigma_2)' &= \sigma_2'\, \sigma_1' & \sigma_1, \sigma_2 &\in \mathscr{L}. \end{aligned}$$

Le terme « inverse » prend maintenant sa signification algébrique; soit σ un élément de $\mathscr{L}$, Λ_x et Λ_y respectivement son élément neutre à gauche et à droite :

$$\Lambda_x\, \sigma = \sigma = \sigma\, \Lambda_y;$$

on vérifie aisément que :

$$\sigma\, \sigma' \simeq \Lambda_x \qquad \text{et} \qquad \sigma'\sigma \simeq \Lambda_y.$$

Ceci signifie que σ' *est l'élément inverse à droite et à gauche de* σ *dans la catégorie* $\mathscr{L}/\simeq$, ce qui justifie la terminologie employée précédemment.

La nouvelle catégorie définie comme quotient $\mathscr{L}/\simeq$ de la catégorie $\mathscr{L}$ par sa relation d'équivalence $\simeq$, est un groupoïde de Brandt [7].

Mots neutres

Considérons la classe[1] des mots σ de $\mathscr{L}$ tels que :

$$\sigma \simeq \Lambda_x \qquad (x \in X).$$

σ est dit mot neutre en x, ou simplement mot neutre de $\mathscr{L}$.

On écrit parfois simplement :

$$\sigma \simeq \Lambda \qquad \text{(on omet l'indice par commodité).}$$

L'élément neutre Λ_x est un mot neutre en x particulier.

Corollaire : Tout mot neutre est cyclique

Soit : $\sigma \simeq \Lambda_x$.

En vertu du théorème III énoncé ci-dessus :

$$g(\sigma) = g(\Lambda_x) = x,$$

et :

$$d(\sigma) = d(\Lambda_x) = x. \qquad \text{C.Q.F.D.}$$

Mots réduits

Par définition, un mot de $\mathscr{L}$ est dit *irréductible*, s'il n'existe aucun mot, à un nombre d'occurrences moindre, qui lui soit équivalent. Un mot est donc irréductible s'il ne contient aucun facteur ll' où $l \in \mathscr{L}$.

Si σ est irréductible, tout facteur de σ et de σ' est irréductible.

Si σ est réductible, il est équivalent à un mot irréductible unique $\pi(\sigma)$; la démonstration de cette propriété est laissée à titre d'exercice.

On va construire $\pi(\sigma)$, mot irréductible équivalent à σ, en recourant à l'algorithme suivant appelé *réducteur à pile par la gauche.*

On note σ_i le facteur gauche de σ à i occurrences de lettres, et $\pi(\sigma_i)$ le mot réduit de σ_i.

L'algorithme construit de proche en proche $\pi(\sigma_1)$, $\pi(\sigma_2)$, ..., $\pi(\sigma)$.

Montrons que des deux mots $\pi(\sigma_i)$ et $\pi(\sigma_{i+1})$ l'un est égal à la concaténation de l'autre et d'une lettre.

En effet, on a deux cas :

— $\pi(\sigma_i)\, l_{i+1}$ est irréductible et donc : $\pi(\sigma_{i+1}) = \pi(\sigma_i)\, l_{i+1}$.

— $\pi(\sigma_i)\, l_{i+1}$ est réductible parce que : $\pi(\sigma_i) = \rho\, l'_{i+1}$,
alors $\pi(\sigma_{i+1}) = \rho$ et donc : $\pi(\sigma_i) = \pi(\sigma_{i+1})\, l'_{i+1}$.

On dit que le mot $\pi(\sigma_i)$ est une « pile » en ce sens que sa construction successive pour $i = 1, \ldots$ ne fait intervenir que sa dernière lettre (« dessus de pile »). Dans l'algorithme ci-dessous, la pile est symbolisée par π.

Algorithme « réducteur à pile par la gauche »

I.0. — Lire $\sigma = l_1\, l_2 \ldots l_k$ où $l_i \in \mathscr{A}$. Poser $\pi \leftarrow l_1\, l_2$, $i \leftarrow 2$.

I.1. — Si $\pi = \Delta\, l\, l'$ avec $\Delta \in \mathscr{L}$, alors $\pi \leftarrow \Delta$, sinon aller ne I.2.

---

1. On reconnaît une écriture apparentée à celle des parenthèsages mathématiques. Dans la terminologie des grammaires formelles, nos mots neutres appartiennent à un langage « context-free » de Chomsky, et plus précisément à un langage de Dyck. On consultera à ce sujet Gross et Lentin [4], chap. XV.

I.2. — Si $i < p$ alors $i \leftarrow i+1$, $\pi \leftarrow \pi\, l_i$, et aller en I.1.

Si $i = p$ stop, écrire $\pi = \pi\,(\sigma)$.

(Le symbole x ← y est l'instruction signifiant que la variable x prend pour nouvelle valeur la valeur actuelle de y.)

Par construction, il apparaît que $\pi\,(\sigma)$ est, d'une part irréductible, d'autre part équivalent à σ.

Exercice. Le mot d'arbre considéré plus haut :

$$\sigma = aa'q'e'eok'ko'q$$

est neutre. En effet, les états successifs du mot σ et de la pile π sont respectivement :

σ	π
aa'	Λ
$aa'q'$	q'
$aa'q'e'$	$q'e'$
$aa'q'e'e$	q'
$aa'q'e'eo$	$q'o$
$aa'q'e'eok'$	$q'ok'$
$aa'q'e'eok'k$	$q'o$
$aa'q'e'eok'ko'$	q'
$aa'q'e'eok'ko'q$	Λ

Nous avons étudié les mots *nets* et les mots *neutres*. Explorer un labyrinthe « sans nouer le fil », c'est construire de ce labyrinthe un *mot net neutre*.

1.3. Les mots a minimum et maximum de replis

Nous allons, dans ce paragraphe, construire des mots nets neutres d'un labyrinthe connexe.

1.3.1. Les algorithmes MINIREPLI et MAXIREPLI

Algorithme MINIREPLI

Soient $\mathscr{L}$ le lexique d'un labyrinthe connexe ($\mathscr{A}$, i, g) et $l_1 \in \mathscr{A}$; on note $\pi\,(\sigma)$ le mot réduit de $\sigma \in \mathscr{L}$; l désigne un élément de $\mathscr{A}$.

Dans l'algorithme ci-dessous, si une instruction autre que STOP peut être effectuée, on passe à I.1. ; si une instruction ne peut pas être effectuée, on passe à la suivante.

I.0. — Poser $\sigma \leftarrow l_1$.

I.1. — Si $\sigma\, l \in \mathscr{L}$, avec $l, l' \notin \sigma$,

poser $\sigma \leftarrow \sigma\, l$.

I.2. — Si $\pi\,(\sigma) = \alpha\, l$ avec $\alpha \in \mathscr{L}$,

poser $\sigma \leftarrow \sigma\, l'$.

I.3. — Stop ; écrire $\rho = \sigma$.

Le mot ρ engendré par l'algorithme est appelé mot MINIREPLI de $\mathscr{L}$.

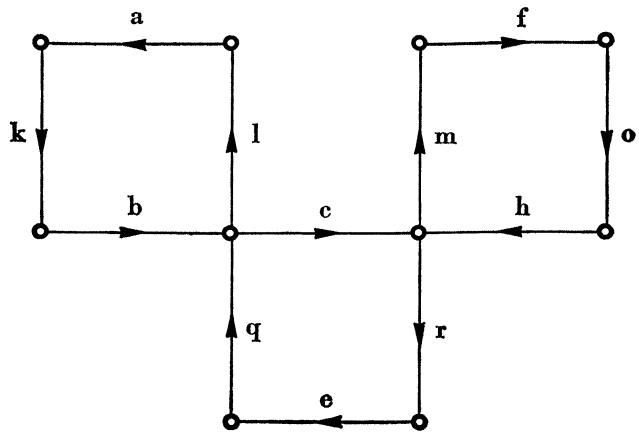


Fig. 4. Un autre exemple de labyrinthe.

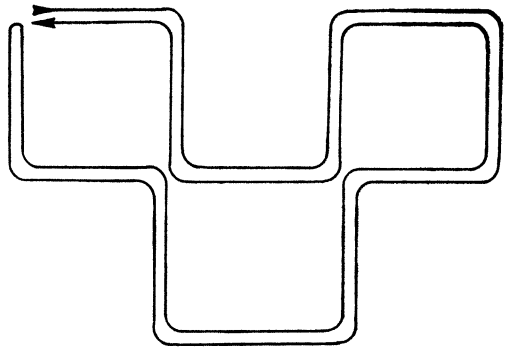


Fig. 5-a. Mot minirepli

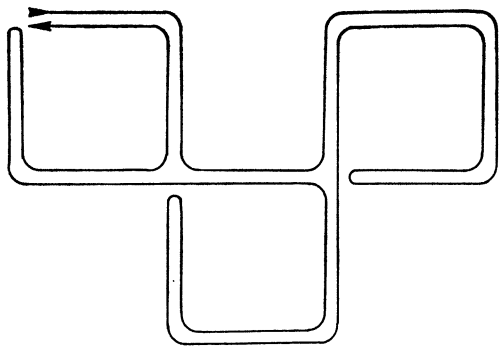


Fig. 5-b. Mot maxirepli

Exemple. Pour le labyrinthe de la figure 4, on a appliqué l'algorithme MINIREPLI, comme il est indiqué en figure 5-a. On a obtenu :

$$\rho = a'l'cmfohreqb'k'kbq'e'r'h'o'f'm'c'la.$$

On constate que ρ est net neutre, et que chaque fois qu'il comporte un repli, c'est par nécessité, tout au moins par nécessité locale. On donne en figure 6 un autre mot MINIREPLI, ρ_1, ayant deux replis alors que ρ n'en a qu'un.

$$\sigma_1 = a'l'q'e'r'c'b'k'kbcmfohh'o'f'm'reqla.$$

Algorithme MAXIREPLI de Trémaux.

Soient $\mathscr{L}$ le lexique d'un labyrinthe connexe ($\mathscr{A}$, i, g) et $l_1 \in \mathscr{A}$; on note $\pi(\sigma)$ le mot réduit de $\sigma \in \mathscr{L}$; l désigne un élément de $\mathscr{A}$.

Dans l'algorithme MAXIREPLI, on observe la même logique d'enchaînement des instructions que pour l'algorithme MINIREPLI à cela près que les instructions J.I et J.I *bis* sont lues dans un ordre arbitraire.

I.0. — Poser $\sigma \leftarrow l_1$.

J.1. — Si $\sigma\, l \in \mathscr{L}$, avec pour tout $k \in \sigma$: $lk \notin \mathscr{L}$,

poser $\sigma \leftarrow \sigma\, l$.

J.1. *bis* — Si $\sigma\, l \in \mathscr{L}$, avec $l, l' \notin \sigma$, avec pour un $k \in \sigma$: $lk \in \mathscr{L}$,

poser $\sigma \leftarrow \sigma\, ll'$.

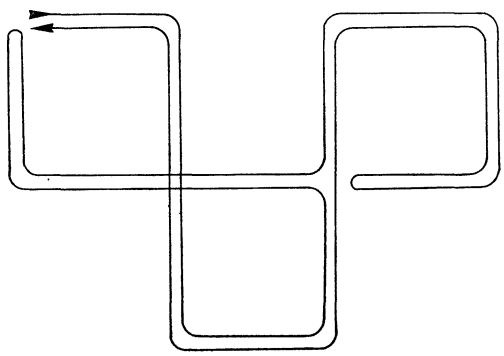


Fig. 6. Un autre mot minirepli

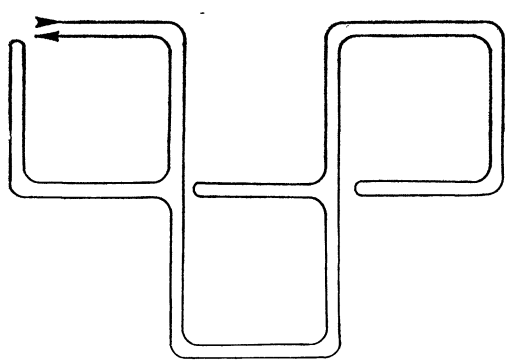


Fig. 7. Un autre mot maxirepli

I.2. — Si $\pi(\sigma) = \alpha\, l$, avec $\alpha \in \mathscr{L}$,

poser $\sigma \leftarrow \sigma\, l'$.

I.3. — Stop; écrire $\tau = \sigma$.

Le mot τ engendré par l'algorithme est appelé mot MAXIREPLI de L ou encore mot de Trémaux, l'algorithme[1] ci-dessus étant dû à Trémaux [21].

Exemple. Pour le labyrinthe de la figure 4, on a appliqué l'algorithme MAXIREPLI, comme il est indiqué en figure 5-b. On a obtenu :

$$\tau = a'\, l'\, cmfohh'\, o'\, f'\, m'\, reqq'\, e'\, r'\, c'\, b'\, k'\, kbla.$$

On constate que τ est net neutre, et que chaque fois qu'il était possible d'effectuer un repli dans le mot, cela a été fait.

On donne en figure 7, un autre mot MAXIREPLI τ_1, ayant trois replis comme τ et comme tous les mots MAXIREPLI du même labyrinthe. τ, a été obtenu en lisant l'instruction J.I *bis* avant J.I.

$$\tau_1 = a'l'q'e'r'c'cmfohh'o'f'm'reqb'k'kbla.$$

1.3.2. Preuves[2] sur les algorithmes MINIREPLI et MAXIREPLI

On remarque l'identité des instructions dénommées pareillement dans les deux algorithmes ci-dessus. Soient ρ et τ deux mots MINIREPLI et MAXIREPLI respectivement. Démontrons qu'ils sont des mots nets neutres de leur labyrinthe.

1) ρ et τ sont des mots de L.

On part d'un mot à une lettre qui appartient à $\mathscr{L}$.

L'instruction I.1. (ou les instructions J.1. et J.1. bis) n'engendre que des mots de $\mathscr{L}$. Quant au mot σ obtenu après application de I.2., il s'écrit : $\sigma = \alpha\, l\, \beta\, l'$ avec $\alpha\, l\, \beta \in \mathscr{L}$ et $\pi(\alpha\, l\, \beta) = \pi(\alpha\, l)$; donc β est un mot neutre de L, il est donc cyclique (*Cf.* 1.2.2.) : $g(\beta) = d(\beta)$.

Or $g(\beta) = d(l)$ et $d(l) = g(l')$, donc $d(\beta) = g(l')$; ce qui signifie que $\alpha\, l\, \beta\, l'$ est un mot de $\mathscr{L}$.

2) *ρ et τ sont des mots neutres*

Quand l'algorithme s'arrête, I.2. ne peut plus être appliqué; on a donc :

$$\pi(\sigma) = \Lambda.$$

3) *ρ et τ sont des mots nets*

Montrons que si un point x apparaît comme droite d'une lettre du mot ρ (resp. τ), toutes les lettres de gauche x sont représentées dans le mot; et en vertu de 2), toutes les lettres de droite x (inverses des précédentes) seront représentées dans le mot. Enfin, par définition de la connexité il s'en suivra que toutes les lettres de $\mathscr{A}$ seront représentées dans le mot.

---

1. Trahtenbrot [20], comme la plupart des auteurs, présente comme « algorithme de la recherche du chemin dans un labyrinthe » l'algorithme de Trémaux ; notons sa conclusion audacieuse : « *Il est donc peu probable que l'on puisse espérer trouver un algorithme plus simple que celui que nous avons proposé* ». On pourra se demander à ce sujet : à quels égards, l'algorithme Minirepli est-il plus simple que l'algorithme Maxirepli ?

2. Trémaux n'a apparemment pas publié sa preuve. Les preuves de Lucas [8] et de Ahrens [1] sont incomplètes. König [6] semble le premier à en donner une rédaction suffisante, toutefois dans un langage peu approprié qui nécessite quatre pages de son livre et qui ne révèle guère du contenu mathématique de ce problème.

Soit donc un point x apparaissant comme droite d'une lettre de ρ (resp. τ), et soit l_x la première lettre du mot telle que d $(l_x) = x$; x apparaît pour la dernière fois comme g (l'_x); il ne peut pas exister de lettre de gauche x non écrite, sinon en écrivant la lettre l'_x on aurait enfreint l'algorithme (on aurait appliqué I.2. alors qu'une instruction précédente était applicable). C.Q.F.D.

Résumons les preuves ci-dessus :

Théorème IV : ***Tout labyrinthe connexe admet des mots nets neutres***

Si L est un arbre, l'instruction J.1 bis de l'algorithme de Trémaux n'est jamais applicable et I.1. = J.1.; les deux algorithmes sont donc identiques. Ils nous permettent d'***associer à un arbre un mot net neutre*** σ, unique si l'on s'est donné l_1 et un ordre sur les lettres de même gauche.

Il n'est pas interdit lors de la construction d'un mot net neutre de passer d'un algorithme à l'autre, c'est-à-dire de choisir à volonté entre (I.1.) et (J.1.) + (J.1 bis), chaque fois que l'on est renvoyé à la première instruction. La démonstration ci-dessus le prouve. On a donné en figure 8 *a* un tel mot mixte.

Par contre, en figure 8 *b* on a indiqué un mot net neutre qui n'est ni MAXIREPLI ni MINIREPLI, ni mixte; il comporte un risque par l'écriture de *l* et par son repli en m; on n'était pas assuré par exemple au moment du repli en m, sans avoir une vue d'ensemble, que la lettre f pourrait être écrite. La faute de parcours mise en évidence est la suivante : la lettre qui a conduit pour la première fois en d (m), à savoir m, a été reprise en sens inverse, à savoir m′, avant que l'on ait écrit toute autre lettre de même gauche que m′.

Cette remarque nous introduit la règle de Tarry.

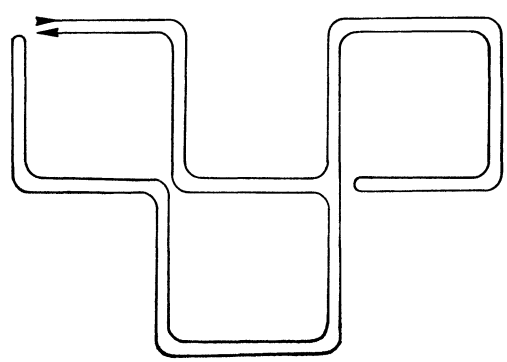


Fig. 8-a. Mot net neutre mixte: tantôt minirepli, tantôt maxirepli

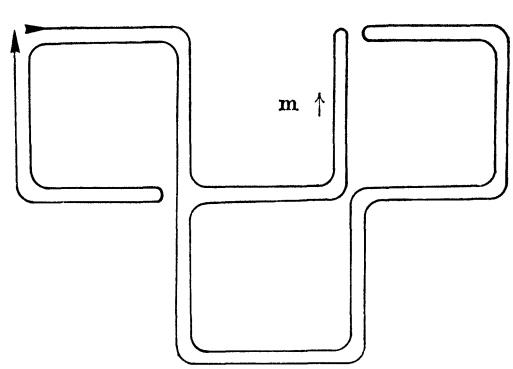


Fig. 8-b. Mot net neutre ni minirepli, ni maxirepli, ni mixte

2. SORTIR DU LABYRINTHE AVEC OU SANS NŒUDS : LES MOTS DE TARRY

Nous allons définir dans le présent chapitre la famille des mots de Tarry d'un labyrinthe dont les mots MINIREPLI et MAXIREPLI ne sont que des cas particuliers, ayant des propriétés spécifiques mises en valeur au chapitre 3.

2.1. Arbre d'entrée et arbre de sortie d'un mot net

Soit un labyrinthe L connexe, d'ensemble de points X, d'alphabet $\mathscr{A}$, et de lexique $\mathscr{L}$.

Soit $x_0 \in X$.

On appelle arbre de L *planté en* x_0, *toute partie* V *de* $\mathscr{A}$ *telle que:*

1) *l'application « droite », d, restreinte à* V *est injective, et son image est* $X - \{ x_0 \}$.

2) *pour toute lettre* $l \in V$, *il existe un mot* σ *de* $\mathscr{L}$ *dont les lettres sont de* V, *et tel que:* $g\ \sigma = x_0$ *et* l *est facteur droit de* σ *(condition de connexité).*

Si $\mathscr{A}$ *est lui même un arbre,* L *est appelé (labyrinthe) arbre.*

Nous aurions pu donner la même définition par l'application « gauche ». Nous dirons : la partie V′ de $\mathscr{A}$ constituée des lettres inverses aux lettres de l'arbre V est aussi appelée arbre de L planté en x_0 (lettres convergentes et non plus divergentes). V et V′ sont dits arbres *inverses* de L.

Soit μ un mot net du labyrinthe connexe L :

$$\mu = l_1\, l_2 \ldots l_{2m}.$$

Posons : $$x_0 = g(\mu)$$

et : $$l_i \in V \Leftrightarrow ((dl_j = dl_i) \Rightarrow j \geqslant i)\ \&\ (dl_i \neq x_0).$$

l_i est appelée lettre d'entrée au point d (l_i) pour le mot μ.

Posons maintenant :

$$l_i \in W \Leftrightarrow ((gl_j = gl_i) \Rightarrow j \leqslant i)\ \&\ gl_i \neq x_0.$$

l_i est appelée lettre de sortie au point g (l_i).

Théorème V : *Les lettres d'entrée (resp. de sortie) d'un mot net* μ *constituent un arbre V (resp.* W*) planté en* g (μ) *appelé arbre d'entrée (resp. de sortie) de* μ.

Démontrons cette propriété en deux temps.

1) Par définition d'une lettre d'entrée deux lettres de V ne sauraient avoir même droite. Et comme tout point autre que x_0 a sa lettre d'entrée, l'image de l'application d restreinte à V est bien $X - \{ x_0 \}$.

2) Soit $l_{i_1} \in V$. Si g $(l_{i_1}) \neq x_0$ alors il existe $l_{i_2} \in V$ avec $l_{i_2}\, l_{i_1} \in \mathscr{L}$.

On raisonnera de même avec l_{i_2}, comme avec l_{i_1}, et ainsi de suite; μ étant fini on atteindra une lettre l_{i_r} de gauche x_0. On aura : $l_{ir} \ldots l_{i_2}\, l_{i_1} \in \mathscr{L}$. C.Q.F.D.

La démonstration est identique pour l'arbre de sortie à celle de l'arbre d'entrée.

Nous avons défini les arbres *de* L plantés en x_0; donnons la définition plus large des arbres *dans* L plantés en x_0, en omettant simplement dans la condition (1) ci-dessus la contrainte « et son image est

$X - \{ x_0 \}$ », et en maintenant la condition (2) intégralement. Il apparaît alors que pour un mot σ quelconque d'un labyrinthe connexe L, les lettres d'entrée (resp. de sortie) de σ constituent un arbre dans L planté en g (σ), appelé arbre d'entrée (resp. de sortie) de σ. La démonstration est calquée sur ce qui précède.

2.2. Les mots de Tarry d'un labyrinthe

2.2.1. Définition

Soit L un labyrinthe connexe. *On appelle mot de Tarry de L, tout mot net* θ *de L, tel que l'arbre d'entrée V de* θ, *et l'arbre de sortie* W *de* θ, *soient inverses :*

$$W = V'.$$

En d'autres termes un mot de Tarry quitte un point pour la dernière fois par l'inverse de la lettre qui l'a mené pour la première fois en ce point. C'est ceci que l'on appelle la règle de Tarry [18], [19].

Il est aisé de vérifier que les mots MINIREPLI et MAXIREPLI sont des mots de Tarry.

2.2.2. Algorithme de Tarry

Soient $\mathscr{L}$ le lexique d'un labyrinthe connexe $L = (\mathscr{A}, i, g)$ et $l_1 \in \mathscr{A}$; on note σ un facteur gauche du mot de Tarry θ cherché, et V (σ) l'arbre d'entrée de σ. Dans l'algorithme ci-dessous, même logique que pour les deux précédents (1.3.1.).

K.0. — Poser $\sigma \leftarrow l_1$.

K.1. — Si $\sigma\, l \in \mathscr{L}$, avec $l \notin \sigma$ et $l' \notin V(\sigma)$

poser $\sigma \leftarrow \sigma\, l$.

K.2. — Si $\sigma\, l \in \mathscr{L}$, avec $l \notin \sigma$,

poser $\sigma \leftarrow \sigma\, l$.

K.3. — Stop ; écrire : $\theta = \sigma$.

Exemple. Pour le labyrinthe de la figure 9, on a appliqué l'algorithme de Tarry, comme il est indiqué en figure 10-a ; on a obtenu :

$$\theta = abcef'c'b'fe'a'h'kk'h.$$

θ est net mais non neutre.

En figure 10-b on a exhibé un mot net neutre φ qui n'est pas de Tarry, à cause de son repli en e contraire à l'instruction K.1. :

$$\varphi = abcc'b'ee'ff'a'h'kk'h.$$

La figure 11-a représente à la fois V (θ) et W (θ), les deux arbres inverses d'entrée et de sortie de θ. La figure 11-b représente l'arbre de sortie de φ ; mais on constate pour φ :

$$W(\varphi) \neq (V(\varphi))',$$

φ n'est pas un mot de Tarry.

Sur la figure 12-a on a tracé le cas particulier d'un mot de Tarry σ du type minirepli, et sur 12-b le cas particulier d'un mot de Tarry τ du type maxirepli ; les arbres d'entrée et de sortie de σ et τ sont représentés par la seule figure 11-a.

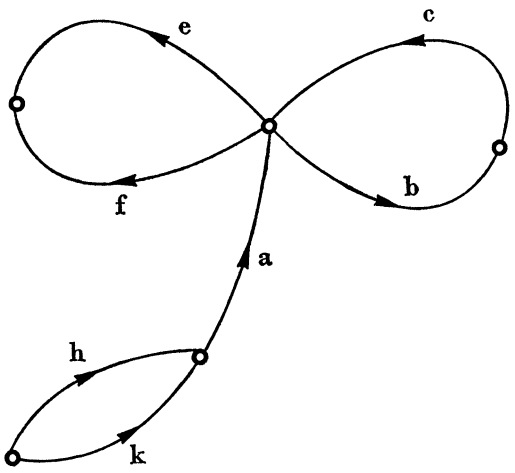


Fig. 9. Un autre exemple de labyrinthe

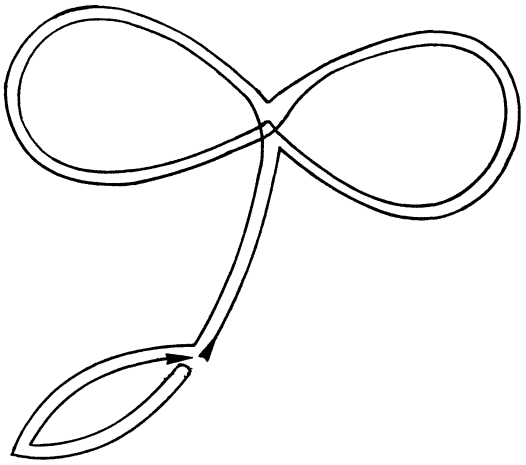


Fig. 10-a. Un mot de Tarry θ

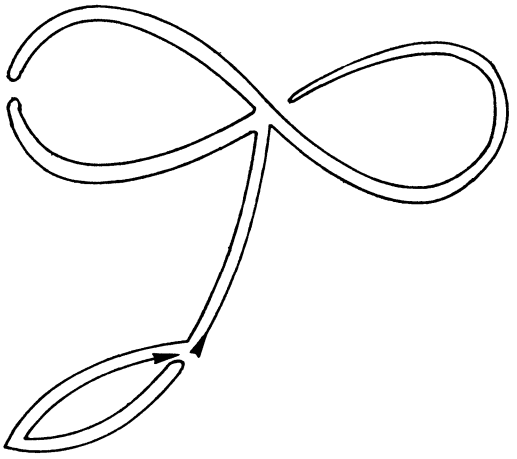


Fig. 10-b. Un mot net neutre φ *qui n'est pas mot de Tarry*

2.2.3. Preuve sur l'algorithme de Tarry

Prouvons que θ, mot engendré par l'algorithme dit de Tarry, est un mot de Tarry du labyrinthe connexe L; on prouvera successivement que θ est mot de L, que $W(\theta) = (V(\theta))'$, et que θ est net dans L.

1) θ est par construction un élément de $\mathscr{L}$.

2) Dans θ, l'inverse de toute lettre d'entrée est lettre de sortie. En effet, faisons d'abord deux remarques :

R.1. θ *est un mot cyclique et toute lettre de* $\mathscr{A}$ *ayant même droite ou même gauche que* θ *figure dans* θ. En effet :

— les lettres α ayant pour gauche $d(\theta)$, et
les lettres β ayant pour droite $d(\theta)$ sont en nombre égal dans $\mathscr{A}$;

— les α et les β figurent alternativement dans θ;

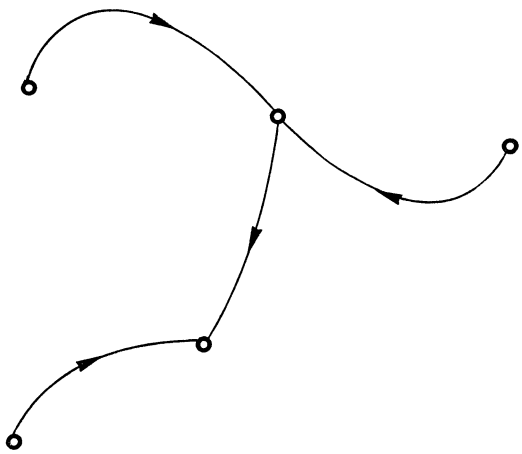


Fig. 11-a. Arbre $W(\theta)$, $(V(\theta))'$, $W(\varphi)$, $W(\rho)$, $(V(\rho))'$, $W(\tau)$, $V(\tau))'$

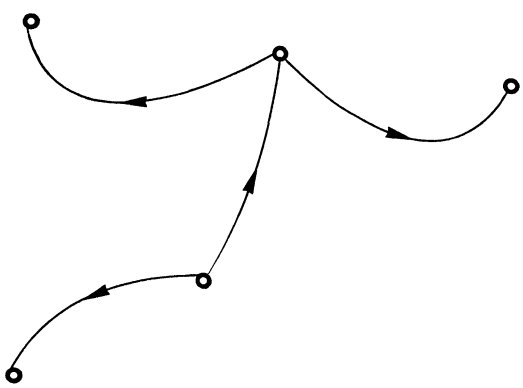


Fig. 11-b. Arbre de rentrée de $abcfe'c'ef'a'h'kk'h$

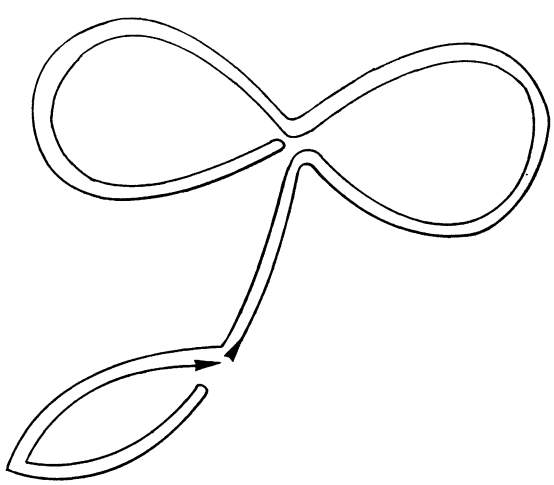


Fig. 12-a. Mot de Tarry ρ *de type minirepli*

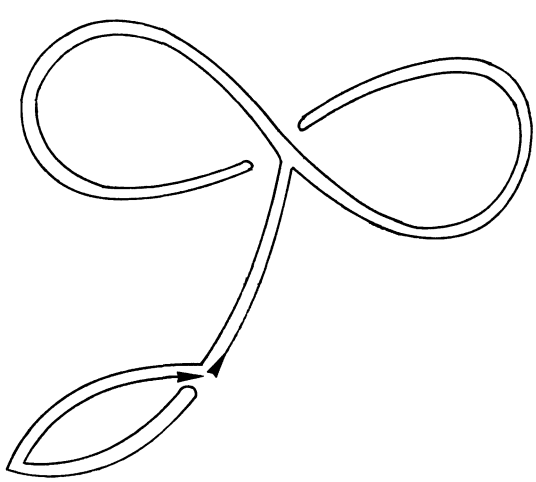


Fig. 12-b. Mot de Tarry τ *de type maxirepli*

— les α et les β figurent toutes dans θ car lorsque K.3. a été appliqué, ni K.1., ni K.2. ne pouvaient l'être et donc toutes les lettres α y figuraient alors.

Il s'en suit que dans θ une lettre α a précédé toutes les lettres de droite $d(\theta)$, c'est dire que $d(\theta) = g(\theta)$.

R.2. Si l'inverse α' d'une lettre d'entrée α figure dans θ, α' est lettre de sortie de θ; de plus toutes les lettres de même droite que α et leurs inverses figurent alors dans θ. Ceci découle de l'instruction K.2. qui n'est appliquée que si K.1. n'est pas applicable.

Considérons maintenant une lettre d'entrée de θ, soit $\alpha \in V(\theta)$. $V(\theta)$ étant un arbre planté en $g(\theta)$, à α est associé un mot σ_α de gauche $g(\theta)$, admettant α comme facteur droit, et dont les occurrences appartiennent à $V(\theta)$. En vertu de *R.1.* l'inverse de la première lettre de σ_α figure dans θ, et donc en vertu de *R.2.*, il en est de même de l'inverse de la seconde lettre de σ_α, puis de la troisième, etc., α' enfin figure dans θ, donc en vertu de *R.2.* : $\alpha' \in W(\theta)$.

3) Reste à montrer que θ est un mot net de L. Soit $l \in \mathscr{A}$ avec $l \notin \theta$. L étant connexe par hypothèse, il existe un mot de même gauche que θ, admettant l comme facteur droit; dans ce mot figure nécessairement un facteur à deux occurrences, rs, tels que $r \in \theta$ et $s \notin \theta$. $d(r)$ admet une lettre d'entrée α puisque $r \in \theta$. On a $\alpha' \in \theta$ en vertu de 2); et enfin $s \in \theta$ en vertu de *R.2.*, ce qui est contradictoire. C.Q.F.D.

Remarque finale. L'algorithme de Tarry est « local ».

Reprenons chaque instruction de l'algorithme de Tarry pour voir quelle information est mise en jeu.

Pour appliquer K.0., il suffit de savoir que l'écriture de θ commence.

Pour appliquer K.1., il suffit de savoir, pour le point $d(\sigma) \neq g(\sigma)$, quelle est l'inverse de sa lettre d'entrée, et pour toute lettre de gauche $d(\sigma)$ si elle figure ou ne figure pas dans σ.

Pour appliquer K.2., il suffit de la dernière information citée.

Pour appliquer K.3., c'est-à-dire arrêter l'exploration, il suffit de constater que ni K.1. ni K.2. ne sont applicables. En bref, pour écrire une nouvelle lettre de σ, ou arrêter d'écrire ($\sigma = \theta$), l'algorithme n'a recours qu'à des informations relatives aux lettres de gauche $d(\sigma)$. C'est pour cette raison que l'algorithme de Tarry est qualifié d'« algorithme local »; il en est de même des algorithmes « réducteur à pile », « minirepli » et « maxirepli ». Le caractère local des règles satisfait le voyageur égaré : le labyrinthe est résolu.[1]

Résumons :

Théorème VI : *Un labyrinthe connexe est résolu par tous ses mots nets dont l'arbre d'entrée et l'arbre de sortie sont inverses.*

Une définition des algorithmes locaux sera donnée en seconde partie en termes d'automates finis. Sans attendre les applications qui en résulteront, nous donnons maintenant quelques résultats se présentant comme des conséquences immédiates de la résolution d'un labyrinthe.

3. QUELQUES RÉSULTATS CONSÉQUENTS A LA RÉSOLUTION D'UN LABYRINTHE

Nous allons, dans le présent chapitre, présenter la résolution immédiate de quelques problèmes au moyen des mots engendrés aux chapitres 1 et 2 par des algorithmes locaux; la déduction elle-même, comme on le remarquera, ne mettra en jeu qu'une « information locale ».

1. Dans [23] nous proposons une formalisation des « algorithmes locaux », appelés encore « algorithmes myopes ».

3.1. L'arbre d'un graphe, base de cycles

Soit un labyrinthe connexe L à m lignes et n points.

Soit τ un mot maxirepli de L. Effaçons dans τ les lettres écrites au titre de l'instruction J.1. bis, c'est-à-dire les $m - n + 1$ facteurs de la forme ll'. Le mot restant, noté $\bar{\tau}$

1) est neutre,

2) a $2(n - 1)$ occurrences, c'est-à-dire couvre $n - 1$ lignes du labyrinthe,

3) admet chacun des n points du labyrinthe comme droite ou gauche d'une de ses lettres.

Un mot ayant les trois propriétés ci-dessus est appelé mot-arbre de L. $\bar{\tau}$ est un mot-arbre de L.

On remarque que l'ensemble des lettres figurant dans $\bar{\tau}$ avant leur inverse constituent l'arbre d'entrée de τ, celles figurant dans $\bar{\tau}$ après leur inverse constituent l'arbre de sortie de τ.

Exemple. Des mots τ et τ_1 cités en (1.3.1.), mots maxirepli du labyrinthe de la figure 4, on déduit :

$$\bar{\tau} = a'l'cmfoo'f'm'ree'r'c'b'bla$$
$$\bar{\tau}_1 = a'l'q'e'r'mfoo'f'm'reqb'bla.$$

Le mot réduit joignant g (b′) à d (r) dans τ est obtenu en prenant dans $\bar{\tau}\,\bar{\tau}$ le facteur ayant b′ comme facteur gauche et r comme facteur droit :

$$b'blaa'l'cmfoo'f'm'r,$$

et en le réduisant : cr.

Résumons : d'un mot maxirepli de L on sait extraire un mot-arbre de L, d'où :

— un mot réduit de tout point à tout point,

— une base de cycles de L,
en concaténant chacune des $m - n + 1$ lettres l produite par l'instruction J.1. bis, avec le mot réduit joignant d (l) à g (l) dans τ,

— le test d'isthme pour une ligne, ou la recherche de tous les isthmes.

Une autre application des mots maxirepli est la décomposition en blocs d'un graphe et l'identification de ses points d'articulation. Nous renvoyons pour cela à III-1 de [23].

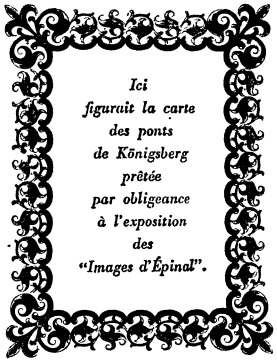
3.2. Les éternels Ponts de Königsberg

Dans son célèbre mémoire, Euler ([22] et [8]) pose le problème : « *A Königsberg, en Poméranie, il y a une île appelée Kneiphof; le fleuve qui l'entoure se divise en deux bras, sur lesquels sont jetés les sept ponts, a, b, c, d, e, f, g. Cela posé, peut-on arranger son parcours de telle sorte que l'on passe sur chaque pont, et que l'on ne puisse y passer qu'une seule fois ? Cela semble possible, disent les uns; impossible, disent les autres; cependant, personne n'a la certitude de son sentiment. Je me suis donc proposé le problème suivant, qui est très général:*

Quelle que soit la forme d'un fleuve, sa distribution en bras, par des îles en nombre quelconque, et quel que soit le nombre des ponts jetés sur le fleuve, trouver si l'on peut franchir celui-ci en passant une fois, et une seule, sur chacun des ponts. »

Ce problème[1] est présenté dans de nombreux ouvrages comme problème historique de topologie, et à juste titre, mais on pourrait croire parfois qu'il est relatif à la planarité de la ville; il n'en est rien. Euler pose le problème et démontre la nécessité de la parité de la valence de chaque sommet.

1. Les amateurs de codage ou de plans d'expérience ne manqueront pas de reconnaître dans les « Mots circulaires et équilibrés » une application des mots eulériens. On pourra, à ce sujet, consulter dans *Math. Sci. hum.* : M. Barbut, n° 17, 1966 ; H. Durup, n° 18, 1967 et, R. A. Chiappa et E. T. Oklander, n° 21, 1968.



Lucas [8] donne une solution constructive qui comporte des constats d'oubli : les quartiers inexplorés dans un premier temps sont repris, d'où le recours à des insertions sans méthode précise (« ces boucles peuvent venir se souder »). En fait, une promenade eulérienne se lit dans un mot minirepli par quelques effacements effectués « localement ».

Rouse Ball [16] avance « The theory of the description of mazes is included in Euler's theorem »; ceci est de notre point de vue une illusion. Certes, en dédoublant chaque ligne d'un labyrinthe, on construit un labyrinthe eulérien, dont tout mot eulérien (prenant les deux copies d'une même ligne en sens inverse) est mot net du labyrinthe de départ. Mais la construction d'un mot eulérien nécessite à notre sens, la construction d'un mot net !

Berge [2] pour sa part, cite un algorithme eulérien très simple que Lucas attribue à Fleury, mais qui recourt à un test d'isthme pour chaque ligne parcourue. Mais de notre point de vue, ce sous-algorithme (voir 3.1.) n'est autre qu'un algorithme de mot net.

Nous montrons ici que l'algorithme minirepli appliqué une seule fois résout le problème eulérien.

On appelle mot eulérien d'un labyrinthe, tout mot $\nu \in \mathscr{L}$, *cyclique, et tel que*

$$l \in \nu \Leftrightarrow l' \notin \nu \text{ pour tout } l \in \mathscr{A}.$$

Un labyrinthe admettant un mot eulérien est dit eulérien.

Soit $x \in X$; on appelle valence de x l'entier suivant :

$$v(x) = \text{card } g^{-1}(x) = \text{card } d^{-1}(x).$$

Théorème VII (Euler) : *Un labyrinthe connexe est eulérien si et seulement si, pour tout* $x \in X$, $v(x)$ *est pair.*

La condition est nécessaire. En effet, soit ν un mot eulérien :

— Les lettres de gauche x ne figurant pas dans ν sont inverses des lettres de droite x figurant dans ν, par définition de ν. Donc le nombre de lettres de droite ou gauche x figurant dans ν n'est autre que $\text{card } g^{-1}(x)$, la valence de x;

— Par ailleurs, le nombre des lettres de droite ou gauche x figurant dans ν est pair, car en considérant leur ordre d'apparition dans ν, on sait les coupler :

pour $x \neq g(\nu)$, on couple une lettre de rang impair dans l'ordre d'apparition avec celle qui la suit dans le mot,

pour $x = g(\nu)$, on couple une lettre de rang pair avec celle qui la suit dans le mot, la dernière étant supposée suivie de la première du mot.

Réciproquement, soit L un labyrinthe connexe dont tout point x est de valence $v(x)$ pair. Nous allons montrer que L est eulérien en construisant un mot eulérien de L. Pour cela, *considérons le mot* ρ *obtenu par l'algorithme minirepli, et effaçons les lettres écrites au titre de l'instruction* I.1., *ce qui donne un mot* ν.

On note :

$$\rho = p_1 s_1 p_2 s_2 \dots p_r s_r$$

où les p_i sont les facteurs constitués de toutes les lettres écrites au titre de l'instruction I.1. et les s_j les facteurs constitués de toutes les lettres écrites au titre de l'instruction I.2. On a :

$$\nu = s_1 \dots s_r.$$

Montrons que ν *est un mot eulérien.*

a) Il faut d'abord montrer que v est un mot de $\mathscr{L}$, ou encore que les p_i sont cycliques. Pour cela, appelons *valence restante en un point x relativement à un mot* σ, le nombre de lignes $\{ l, l' \}$ telles que :

$$x = d(l) = g(l') \qquad \text{et} \qquad l \notin \sigma, l' \notin \sigma.$$

Relativement à p_1, la valence restante des sommets autres que $g(p_1)$ apparaissant comme gauche d'une lettre de p_1 est paire ; celle de $g(p_1)$ l'est si et seulement si, $d(p_1) = g(p_1)$. Or, lorsqu'on va écrire s_1, l'instruction I.1. n'est plus applicable au sommet $d(p_1)$, donc la valence restante de $d(p_1)$ est nulle ; ce qui implique : $g(p_1) = d(p_1)$.

Par récurrence, on a pour un mot p_i (i quelconque) : la valence restante des sommets, autres que $g(p_i)$, apparaissant comme gauche de lettres dans p_i, est paire ; la valence restante de $d(p_i)$ est nulle, ce qui implique $g(p_i) = d(p_i)$.

b) Montrons maintenant que ν est cyclique. ρ étant mot net de L est cyclique ; donc $g(\rho) = d(\rho)$. Or, $d(\rho) = d(\nu)$ (la dernière lettre de σ est nécessairement écrite au titre de l'instruction I.2.) ; et p_1 étant cyclique, $g(\rho) = g(p_1) = d(p_1) = g(\nu)$. Donc : $g(\nu) = d(\nu)$.

c) Enfin σ est un mot net ; l'instruction I.2. fait apparaître en effet une lettre si et seulement si, elle ne fait pas apparaître son inverse :

$$l \in \nu \Leftrightarrow l' \notin \nu \qquad \text{pour} \qquad l \in \mathscr{A}. \qquad \text{C.Q.F.D.}$$

Formulons ce résultat un peu différemment :

Théorème VIII : *La suite des lettres figurant dans un mot minirepli d'un labyrinthe L après leur inverse, constitue un mot eulérien du labyrinthe si et seulement si, L est eulérien.*

Remarque sur l'orientation. Si un labyrinthe ($\mathscr{A}$, i, d) est orienté par la donnée de $\mathscr{A}^+$ ($\mathscr{A} = \mathscr{A}^+ + \mathscr{A}^-$), et que en tout point x le nombre de lettres de $\mathscr{A}^+$ de droite x est égal au nombre de lettres de $\mathscr{A}^+$ de gauche x, on construit un mot eulérien (dit orienté) dont toutes les lettres appartiennent à $\mathscr{A}^+$, en donnant simplement priorité, dans la construction du mot minirepli ρ, aux lettres de $\mathscr{A}^-$ sur celles de $\mathscr{A}^+$.

Exemples. Le labyrinthe de la figure 4 est eulérien. En figure 13-a on lit un mot minirepli et en 13-b le mot eulérien associé.

Par contre, pour un mot maxirepli, tel celui de la figure 14-a, l'effacement des lettres qui figurent dans τ avant leur inverse ne produit pas un mot du labyrinthe, comme on le constate en figure 14-b.

3.3. Quelques énoncés de type hamiltonien

Dans ce qui suit, k désigne un entier positif ou nul.

L'application $\varphi : \mathscr{A} \to U$ associe à toute lettre la ligne qui lui correspond naturellement. Dans le labyrinthe L,

— *2 lettres* l_1 *et* l_2 *sont dites k-adjacentes* s'il existe un mot de L à au plus $k + 2$ occurrences de lettres dont la première est l_1 et la dernière est l_2.

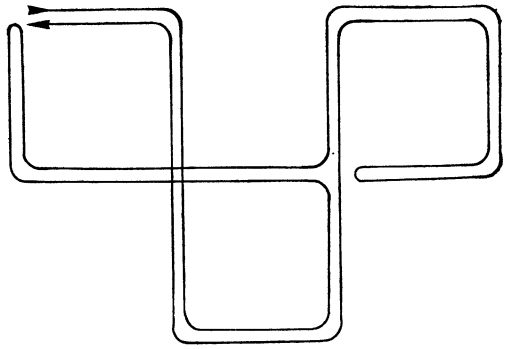


Fig. 13-a. Mot minirepli d'un labyrinthe eulérien

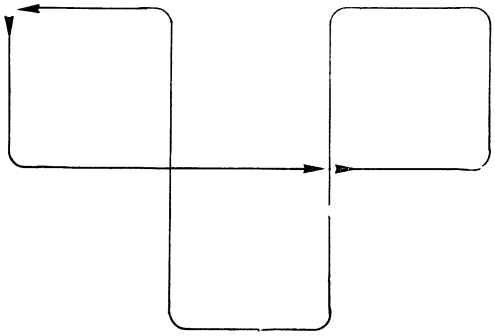


Fig. 13-b. Mot eulérien

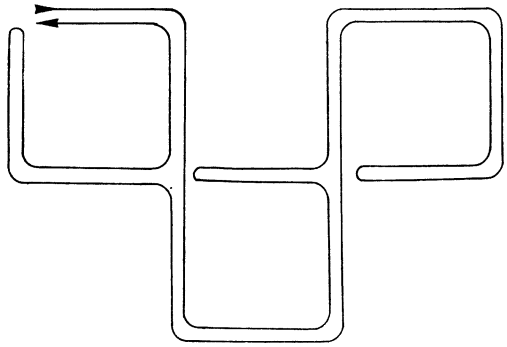


Fig. 14-a. Mot maxirepli τ d'un labyrinthe eulérien

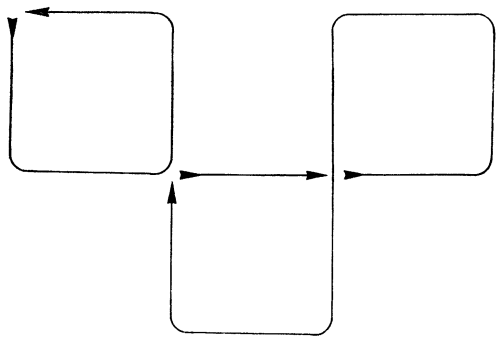


Fig. 14-b. τ après effacement des lettres qui y figurent avant leur inverse

— *2 lignes A_1 et A_2 sont dites k-adjacentes*, s'il existe 2 lettres *k-adjacentes* l_1 et l_2, telles que :

$$\varphi(l_1) = A_1 \qquad \text{et} \qquad \varphi(l_2) = A_2.$$

— *2 points x_1 et x_2 sont dits k-adjacents*, s'il existe 2 lettres *k*-adjacentes l_1 et l_2, telles que :

$$x_1 = g(l_1) \qquad \text{et} \qquad x_2 = g(l_2).$$

Théorème IX : *Pour tout labyrinthe connexe G, il existe des permutations circulaires des lettres, lignes et sommets respectivement, telles que l'image de chaque élément soit respectivement 0-adjacente, 1-adjacente et 2-adjacente à cet élément.*

Permutation des lettres. On a dans un mot net, une permutation circulaire des lettres de $\mathscr{A}$ telle que toute lettre et son image soient 0-adjacentes.

Permutation des lignes. Si on a remarqué que dans une opération de réduction d'un mot la parité de rang d'une lettre restante est invariante, on en déduit que *2 lettres inverses quelconques d'un mot net neutre sont de parités de rang distinctes.*

Si dans un mot net neutre de l on ne garde que les lettres de rang impair, et si à chaque lettre l on associe sa ligne $\varphi(l)$, on obtient ainsi une permutation circulaire des lignes (chaque ligne est représentée une fois et une seule, la première est image de la dernière), telle que toute ligne et son image soient 1-adjacentes.

Permutation des sommets. Considérons un mot net neutre $\bar{\sigma}$, mot-arbre de L (3.1.). D'après la proposition démontrée ci-dessus, il existe une permutation des lignes de l'arbre telle que toute ligne et son image soient 1-adjacentes.

Considérons maintenant le point $x_0 = g(\bar{\sigma}) = d(\bar{\sigma})$, puis à toute lettre de rang impair de σ associons un point de L de la manière suivante :

$$l \to d(l) \text{ si } l' \text{ est après } l \text{ dans } \sigma,$$
$$l \to g(l) \text{ si } l' \text{ est avant } l \text{ dans } \sigma.$$

Montrons qu'on obtient ainsi une permutation circulaire des points, ou encore que les n points de L sont numérotés de 0 à $n-1$ par l'application définie ci-dessus, comme les lettres de rang impair de $\bar{\sigma}$ sont numérotées dans leur ordre d'écriture.

Nous numéroterons le point x_0: 0 (on le mettra en premier élément de la permutation circulaire, l'image du dernier).

Montrons que les points autres que x_0 sont tous numérotés; comme il y a $n-1$ points autres que x_0 et $n-1$ paires de lettres inverses dans $\bar{\sigma}$, montrons simplement qu'aucun sommet n'est numéroté deux fois. Ceci résulte du fait que pour tout point x, autre que x_0, il existe une lettre unique de rang impair, l_x, telle que $d(l_x) = x$. On ne peut avoir :

— ni $x = d(l_i) = d(l_j)$, $\quad l_i \neq l_j$;

— ni $x = d(l_i) = g(l_j)$, car l'_j serait avant l_j et alors $x = d(l'_j)$; or, si une lettre figure en rang impair, son inverse n'y figure pas;

— ni $x = g(l_i) = g(l_j)$, car alors $x = d(l'_i) = d(l'_j)$.

De plus, dans cette numérotation, 2 points de numéros consécutifs sont 2-adjacents par construction de l'application définie ci-dessus.

Nous avons donc trouvé une permutation circulaire des points telle que chaque point et son image soient 2-adjacents. C.Q.F.D.

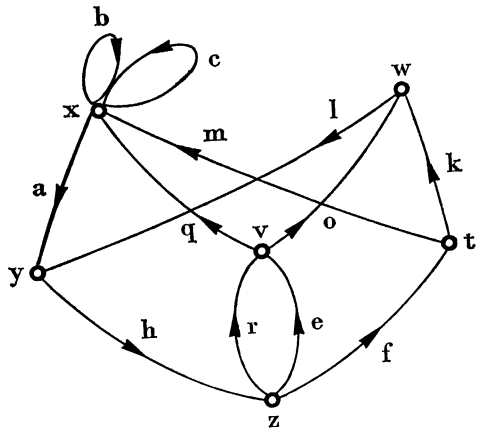


Fig. 15-a. Labyrinthe L

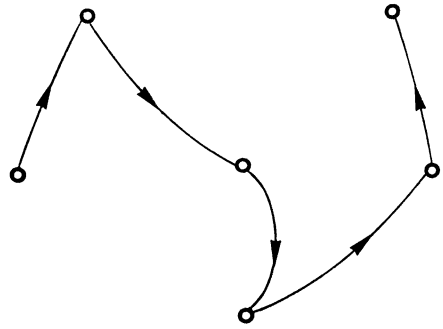


Fig. 15-b. Arbre d'entrée V (τ) du mot maximal τ de L

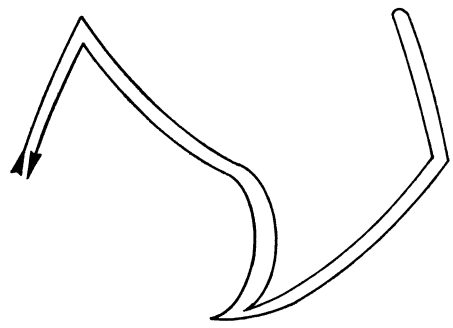


Fig. 15-c. Mot-arbre τ

Exemples

a) Sur le labyrinthe L de la figure 15, calculons un mot maxirepli et retenons les lettres de rang impair :

$$\tau = p_0 = a'bb'q'e'rr'h'hfkll'o'ok'mm'f'eqc'ca$$
$$p_1 = a'\ b'\ e'\ r'\ h\ k\ l'\ o\ m\ f'\ q\ c$$

p_0 est une permutation circulaire des lettres de L respectant la 0-adjacence, p_1 est une permutation circulaire des lignes de L respectant la 1-adjacence. Calculons maintenant le mot-arbre $\bar{\tau}$ de l'arbre d'entrée de τ donné en figure 15-c :

$$\bar{\tau} = a'q'e'fkk'f'eqa.$$

(Disons en passant que $\bar{\tau}$ est un mot miroir, parce que l'arbre d'entrée de τ est linéaire ; ce cas très particulier est dû à l'abondance des connexions entre les points de L, sur laquelle on pourra réfléchir.)

Retenons de $\bar{\tau}$ les lettres de rang impair et associons à chacune selon la règle un sommet, sans oublier d'ajouter le sommet de départ :

$$\begin{array}{rccccccc} & a' & & e' & & k & & f' & & q \\ p_2 = & y & & x & & z & & w & & t & & v. \end{array}$$

p_2 est une permutation circulaire des points de L respectant la 2-adjacence.

b) En figure 16 on a représenté un labyrinthe bien connu [12] pour lequel la permutation p_2 calculée respecte non seulement la 2-adjacence des points mais la 0-adjacence des points : le labyrinthe dans ce cas est dit ***hamiltonien.***

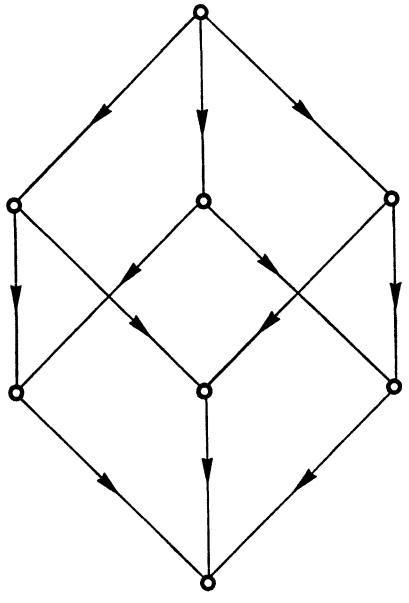


Fig. 16-a. Générateur du simplexe S_3

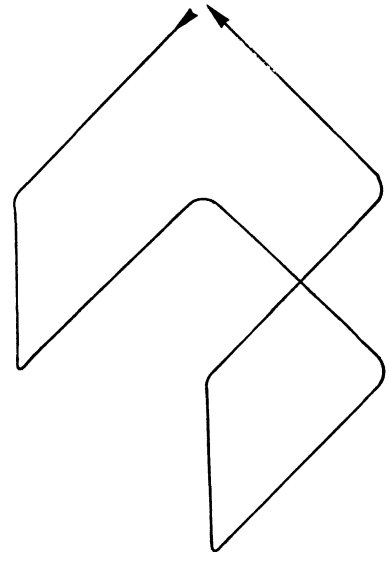


Fig. 16-b. Mot hamiltonien de S_3

Remarque finale. Les calculs effectués sont locaux.

On note bien que les diverses transformations effectuées ci-dessus sur les mots nets neutres sont de nature locale. Dans le cas de la numérotation des points, l'ordre circulaire de ceux-ci signifie à cet égard que

chaque point du labyrinthe étend son aire de communication — c'est-à-dire de calcul — à la 2-adjacence : il connaît de la sorte son prédécesseur et son successeur dans l'ordre circulaire. Quelle intelligence lui est-elle nécessaire pour en prendre connaissance ? Quel est ce point « automate fini » capable de faire la roue avec tous les autres ? On pourra y réfléchir avec [3] [9] et [13] ; nous y reviendrons.

(à suivre).

La réalisation graphique est dûe à C. Havas.

BIBLIOGRAPHIE

[1] Ahrens, W., *Mathematische Unterhaltungen und Spiele*, I, II, Leipzig, 1910-1918.

[2] Berge, C., *Graphes et hypergraphes*, chap. II : "Recouvrement des arêtes par des chaînes", Dunod, 1970, pp. 218-221.

[3] Berstel, J., "Résolution par un réseau d'automates, du problème des arborescences dans un graphe", *C. R. Acad. Sci., Paris*, 264, 1967, pp. 388-390.

[4] Gross, M., Lentin, A., *Notions sur les grammaires formelles*, Paris, Gauthier-Villars, 1967.

[5] Guilbaud, G. Th., *La Cybernétique*, Presses Universitaires de France, Coll. Que Sais-je ?, 1957. [Traduction anglaise : *What is Cybernetics*, Londres, Meinemann, 1959.]

[6] König, D., *Theorie der endlichen und unendlichen Graphen*, Londres, Chelsea, 1935.

[7] Lévy-Bruhl, J., *Introduction aux structures algébriques*, Paris, Dunod, 1968.

[8] Lucas, E., *Récréations mathématiques*, Paris, Albert Blanchard, 2e éd., nouv. tir., 1960.

[9] Moore, F., "The shortest path through a maze", *Proceedings of International Symposium in the Theory of Switching*, part II, Cambridge, Mass., Harvard University Press, 1959, pp. 285-292.

[10] Pailhous, J., "Représentation de l'espace urbain et cheminements", *Le travail humain*, 32 (3-4), 1969, pp. 239-270.

[11] Pham, F., "Singularité des processus de diffusion multiples", *Annales de l'Institut Henri Poincaré*, 6 (2), 1967.

[12] Rosenstiehl, P., "Quelques exercices à traiter sur simplexes : circuit hamiltonien sur le simplexe S_n", *Math. Sci. hum.*, n° 10, printemps 1965.

[13] Rosenstiehl, P., "Existence d'automates finis capables de s'accorder bien qu'arbitrairement connectés et nombreux", *ICC Bulletin*, vol. 5, International Computation Center, 1966, pp. 245-261.

[14] Rosenstiehl, P., "Graph problems solved by finite automata networks", *Calgary International Conference on Combinatorial Structures and their Applications*, non published paper, 1969.

[15] Rosenstiehl, P., *Graphes, leurs vecteurs et leurs mots*, cours rédigé en collaboration avec F. Moniez et J. C. Bermond, Doc. 5440319, École Pratique des Hautes Études, 1971.

[16] Rouse Ball, W. W., (revised by Coxeter H.S.M.), *Mathematical Recreations and Essays*, MacMillan, 1963.

[17] Sainte-Lagüe, A., *Les réseaux (ou graphes)*, Memorial des Sciences Mathématiques, Fasc. 18, Paris, Gauthier-Villars, 1926.

[18] TARRY, G., "Parcours d'un labyrinthe rentrant", *Assoc. franç. pour l'avanc. des sciences*, 1886, pp. 49-53.

[19] TARRY, G., "Le problème des labyrinthes", *Nouvelles annales de mathématiques*, XIV, 1895.

[20] TRAHTENBROT, B. A., *Algorithmes et machines à calculer*, Paris, Dunod, 1963.

[21] TRÉMAUX, cité par E. Lucas [8].

[22] EULER, *Solutio problematis ad geometriam situs pertinentis*, Mémoire de l'Académie des Sciences, Berlin, 1739.

[23] ROSENSTIEHL, P., FIKSEL, J. R., et HOLLIGER, A., "Intelligent Graphs" (Networks of Finite Automata Capable of Solving Graph problems), in R. C. Read (ed.), *Graph theory and computing*, Academic Press, New York (à paraître en 1972).