Chapitre 4

Graphes automatiques

Ce chapitre traite des graphes automatiques. Il s'agit d'une classe de graphes infinis constructifs qui généralise strictement celle des graphes HR-équationnels. Gardant à l'esprit l'idée de faire émerger des propriétés topologiques calculables; nous avons essayé d'apréhender la frontière idéale d'un graphe automatique de façon constructive. Il s'est avéré que le problème consistant à compter le nombre de bouts d'un tel graphe est indécidable. Pour démontrer cela, nous introduirons les graphes de configuration de machine de Turing, nous verrons que ces graphes sont automatiques. Nous nous servirons aussi du concept de machines auto-stabilisantes; il s'agit de machines contrôlant elles-même la validité de leurs calculs. La démonstration de l'indécidabilité du problème des bouts repose alors sur une analyse des propriétés géométriques globales du graphe des configurations d'une certaine machine auto-stabilisante.

Dans une seconde partie, nous établirons un lien entre les graphes automatiques et les $D0L$-systèmes de graphes : la suite de graphes finis produite en itérant un $D0L$-système est exactement la suite des sphères d'un graphe automatique; la réciproque et vraie pour une large classe de graphes automatiques. Cela nous conduira à démontrer que la connexité simultanée de tous les graphes d'une $D0L$-suite est une question indécidable. Ce résultat contribut à la compréhension de la limite de décidabilité de cette question, notamment du fait que les $D0L$-systèmes émulent les grammaires VR de graphes, i.e. les grammaires de remplacement de sommets (voir [ER97, CER93, CE95]), pour lesquelles la question précédente est décidable.

4.1 Graphes automatiques

Le concept de *graphe automatique* [BG00, CS, KN95, Mor99, Pel97, Sén92] vise à définir des graphes infinis de façon constructive. A cet égard, il généralise celui de graphe HR-équationnel. Le concept de graphe automatique que nous considérons ici provient de celui de *structures relationnelles automatiques introduites dans [KN95]* qui généralise la notion de groupe automatique. Une structure relationnelle automatique est une structure relationnelle dont le domaine est donné sous la forme d'un langage rationnel et dont les relations sont décrites par des automates. De fait, de multiples variations peuvent être envisagées, comme par exemple, les structures automatiques basées sur des langages d'arbres (voir [BG00]). La

définition des relations laisse une grande liberté; on peut considérer de façon très générale des relations rationnelles, des transducteurs équilibrés (cf. [Sén92]), ou encore des automates lettres à lettres comme dans le cas des groupes automatiques.

Dans notre cas, nous considèrerons une classe particulière de structures relationnelles automatiques, en fait de graphes automatiques, que nous appellerons graphes automatiques régulièrement accessible : Les sommets d'un tel graphe sont les classes d'équivalence des mots d'un langage rationnel stable par préfixe, relativement à une relation d'équivalence elle-même reconnue par un automate à deux entrées. Il y a deux types d'arcs; premièrement, les arcs *radiaux* : si un mot est obtenu à partir d'un autre en ajoutant simplement une lettre à la fin, alors les sommets représentés par ces deux mots sont liés par un arc dit radial, étiqueté par la lettre en question. Les arcs du second type, que l'on appellera arcs *transversaux*, sont décrits par des automates à deux entrées : si un couple de mots est reconnu par l'un des automates, alors les sommets représentés par ces mots sont liés par un arc; plusieurs automates sont utilisés pour définir les arcs transversaux afin de distinguer plusieurs étiquettes, chacun d'eux étant associé à une étiquette.

Suivant [ECH+92], par *automate à deux entrées* sur un alphabet A, nous entendons un automate à nombre fini d'états sur l'alphabet $(A \cup \{\$\}) \times (A \cup \{\$\})$, où \$ est un symbole de fin de chaîne non inclus dans A, dont le langage consiste en un ensemble de couples de mots (u, v) de $A^*.\$^* \times A^*.\$^*$ de même longueur avec $u \in A^*$ ou bien $v \in A^*$. Par extension, nous dirons que qu'un automate à deux entrées reconnaît un couple de mots $(u, v) \in A^* \times A^*$ s'il reconnaît en réalité le couple $(u.\$^{\max\{0,|v|-|u|\}}, v.\$^{\max\{0,|u|-|v|\}})$.

Nous proposons la définition formelle suivante :

Définition 14 (Graphes automatiques régulièrement accessible)
Une ***structure automatique*** *est un triplet $(W, M_0, \{M_\ell\}_{\ell \in L})$ où :*

- *W est un automate (à nombre fini d'états) sur un alphabet fini A dont le langage $\mathcal{L}(W) \subset A^*$ est stable par préfixe;*
- *M_0 est un automate à deux entrées dont le langage $\mathcal{L}(M_0) \subset A^* \times A^*$ est le graphe d'une relation d'équivalence sur A^*;*
- *L est un ensemble fini et pour tout $\ell \in L$, M_ℓ est un automate à deux entrées sur A.*

Une structure automatique définit un ***graphe automatique régulièrement accessible*** *de la façon suivante :*

- *l'ensemble des sommets est $\mathcal{L}(W)/\mathcal{L}(M_0)$. Comme $\mathcal{L}(W)$ est stable par préfixe, il contient le mot vide; dans la suite, on notera ε sa classe d'équivalence relativement à $\mathcal{L}(M_0)$. Les sommets ne sont pas étiquetés;*
- *les arcs sont des couples de mots de $\mathcal{L}(W)$; ils sont divisés en deux types. Les premiers sont les arcs* ***radiaux****, ce sont les couples de la forme $(w, w.a) \in \mathcal{L}(W) \times \mathcal{L}(W)$ avec $a \in A$; $(w, w.a)$ est étiqueté par a, son origine est le sommet représenté par w et son extrémité, le sommet représenté par $w.a$. Les arcs du second type sont les arcs* ***transversaux****, ce sont les couples (w_1, w_2) reconnus par l'un des M_ℓ; un tel arc est étiqueté par ℓ et connecte le sommet représenté par w_1 à celui représenté par w_2.*

Pour alleger le texte, les graphes automatiques régulièrement seront simplement appelés "graphes automatiques". Notons que les termes d'arc "radial" et d'arc "transversal" reçoivent une justification géométrique lorsque par exemple les automates à deux entrées définissant la relation d'équivalence pour les sommets et les arcs transversaux ne reconnaîssent que des couples de mots de mêmes longueurs.

La définition d'un graphe infini par une structure automatique est constructive. En premier lieu, on peut construire un langage rationnel qui est un système de représentants uniques pour les sommets: considérons un ordre linéaire sur A, l'alphabet de W (on conserve les notations de la définition 14); on considére ensuite l'ordre *militaire* (ou ordre "*shortlex*" dans la littérature anglo-saxonne) sur $\mathcal{L}(W)$, i.e. un mot w_1 est plus petit qu'un mot w_2 s'il comporte strictement moins de lettres que w_2, ou bien s'il en comporte autant mais il est alors plus petit que w_2 pour l'ordre lexicographique; c'est un *bon* ordre, et chaque classe d'équivalence de $\mathcal{L}(W)$ relativement à $\mathcal{L}(M_0)$ comporte un unique plus petit élément, c'est son représentant minimal pour l'ordre militaire. On montre que le langage des représentants minimaux des classes d'équivalence de $\mathcal{L}(W)$ est un langage rationnel (cf. [ECH$^+$92]). En second lieu, les arcs transversaux peuvent être reconnus de façon effective, ce qui n'est pas tout-à-fait clair au vu de la définition 14. On montre pour cela que chaque M_ℓ peut être transformé de telle sorte que le langage qu'il définit soit clôs relativement à $\mathcal{L}(M_0)$, i.e. si $w_1, w_2, w_1', w_2' \in \mathcal{L}(W)$ sont tels que pour $i = 1, 2$, w_i et w_i' soient $\mathcal{L}(M_0)$-équivalents, i.e. $(w_i, w_i') \in \mathcal{L}(M_0)$, alors $(w_1, w_2) \in \mathcal{L}(M_\ell)$ si et seulement si $(w_1', w_2') \in \mathcal{L}(M_\ell)$ (cf. [ECH$^+$92]).

Les graphes de Cayley de groupes automatiques fournissent naturellement beaucoup d'exemples de graphes automatiques (cf. [Eps88, ECH$^+$92]. Notons parmi eux les groupes libres (voir figure 4.1), qui donnent des exemples d'arbres ou encore les groupes abéliens libres qui donnent des exemples de grilles multi-dimensionnelles; les groupes hyperboliques au sens de Gromov sont fortement automatiques, et donnent également naissance à une famille très esthétique de graphes infinis automatiques. Mais le concept de graphe automatique

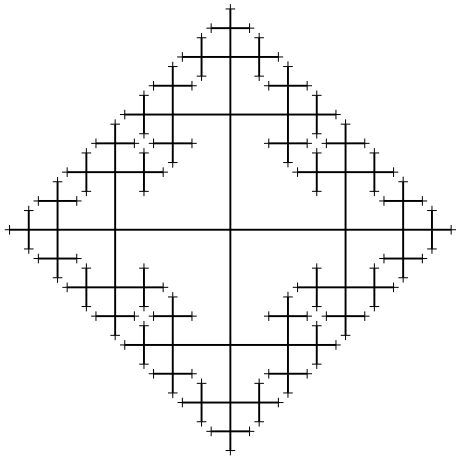

FIG. 4.1: *Graphe de Cayley du groupe libre à deux générateurs*

est bien plus général que celui de graphe de Cayley de groupe automatique; il n'impose pas les conditions de symétrie très fortes dues à l'hypothèse de structure de groupe, les graphes

automatiques ne sont pas, comme les graphes de Cayley, *sommet-transitif*, i.e. le groupe des automorphismes n'agit pas nécessairement transitivement sur l'ensemble des sommets (cf. [CS]). A cet égard, il est facile de voir que les arbres réguliers sont des graphes automatiques; c'est également le cas des graphes context-free au sens de [MS85] (cf. [Sén92]).

4.2 Graphe des configurations d'une machine de Turing

Machines de Turing

On considère des machines de Turing à une bande munies de plusieurs têtes de lecture/écriture; une particularité des machines que nous allons définir est qu'elles sont capables de détecter le fait qu'une tête soit en bout de bande ou non. On vérifie facilement que notre modèle est équivalent au concept classique de machine de Turing qui est présenté e.g. dans [HU79, Rog67]. Formellement, une *machine de Turing* à n têtes de lecture/écriture sera pour nous un 6-uplet $T = (Q, \Gamma, \delta, q_{\mathsf{init}}, F, \Box)$ où Q est l'ensemble fini des états de T; Γ est l'alphabet de bande; δ est une fonction partielle de $Q \times \Gamma^n \times \{0,1\}^n$ dans $Q \times \{1, ..., n\} \times (\Gamma \setminus \{\Box\}) \times \{\mathsf{gauche}, \mathsf{stop}, \mathsf{droite}\}$; $q_{\mathsf{init}} \in Q$ est l'état initial de T; $F \subset Q$ est l'ensemble des états finaux de T; enfin $\Box \in Q$ est le symbole de blanc. T est équippée d'une bande de mémoire infinie à droite. Au démarrage, la mémoire ne contient que des blancs, T est dans l'état q_{init} et toutes ses têtes sont positionnées sur la première cellule de la bande. Les calculs de T sont définis par δ : considérons une transition $\delta(q, a_1, ..., a_n, b_1, ..., b_n) = (q', i', a', D')$; cela veut dire que si T se trouve dans l'état q et que pour $i = 1, ..., n$, la cellule sur laquelle pointe la i-ième tête contient le symbole a_i et est la première cellule de la bande si et seulement si $b_i = 1$, alors T passe dans l'état q' et sa i-ième tête se déplace à gauche (respectivement se déplace à droite ou garde sa position) si $D' = \mathsf{gauche}$ (respectivement si $D' = \mathsf{droite}$ ou $D' = \mathsf{stop}$), après avoir écrit le symbole a' sur la bande. Notons que T ne peut pas écrire le symbole de blanc sur la bande.

Une *description instantanée* de T est un mot sur l'alphabet $(Q \times \{1, ..., n\}) \cup (\Gamma \setminus \{\Box\})$ de la forme $w_0 q^{i_1} w_1 q^{i_2} ... w_{n-1} q^{i_n} w_n$ où w_j est un mot sur l'alphabet $\Gamma \setminus \{\Box\}$; pour simplifier les notations, (q, i_j) est noté q^{i_j} et $\{i_1, ..., i_n\} = \{1, ..., n\}$. Un tel objet a pour but de décrire T à un instant donné : T est dans l'état q, la bande de mémoire contient le mot $w_0 w_1 ... w_n \Box^\omega$ et pour $j = 1...n$, la i_j-ième tête est positionnée sur la cellule contenant le premier symbole de w_j. Soient $i' > i$, si la i-ième et la i'-ième tête de T sont positionnées sur la même cellule, la description instantanée de T sera de la forme $wq^i q^{i'} w'$ et non de la forme $wq^{i'} q^i w'$; ainsi, la description instantanée de T à un instant donné est unique. Soient Id_1 et Id_2 deux descriptions instantanées de T; on écrit $\mathsf{Id}_1 \underset{T}{\rightarrow} \mathsf{Id}_2$ (respectivement $\mathsf{Id}_1 \overset{*}{\underset{T}{\rightarrow}} \mathsf{Id}_2$, $\mathsf{Id}_1 \overset{+}{\underset{T}{\rightarrow}} \mathsf{Id}_2$) si Id_2 est le résultat d'un pas de calcul (respectivement d'un calcul fini, d'un calcul fini non vide) effectué par T à partir de Id_1.

Graphe des configurations

L'un des outils essentiels pour la preuve du théorème 9 énoncé page 113 est une légère variation du concept de *graphe des configurations* d'une machine de Turing (cf. [Pap94]). Soit $T = (Q, \Gamma, \delta, q_{\mathsf{init}}, F, \square)$ une machine de Turing munie de n têtes. Soit $A = (Q \times \{1, ..., n\}) \cup (\Gamma \setminus \{\square\})$. On considère le langage $\mathcal{L} \subset A^*$ des descriptions instantanées de T; notons que $\mathcal{L}$ regroupe *toutes* les descriptions instantanées de T, pas seulement celles qui sont issues de la configuration initiale. On considère alors le langage $\mathcal{L}.\square^*$ des mots de la forme $\mathsf{Id}.\square.\square....\square$ avec $\mathsf{Id} \in \mathcal{L}$; notons qu'un mot de $\mathcal{L}.\square^*$ code sans ambiguité une description instantanée de T. Soit maintenant le langage $\overline{\mathcal{L}.\square^*}$ de tous les préfixes des mots de $\mathcal{L}.\square^*$. On considère alors le graphe G_T dont les sommets dont les mots de $\overline{\mathcal{L}.\square^*}$, sans étiquette, et dont les arcs sont définis comme suit; soient w_1 et w_2 deux mots de $\overline{\mathcal{L}.\square^*}$,

- s'il existe un symbole $a \in A$ tel que $w_2 = w_1.a$, alors il y a un arc étiqueté par a de w_1 à w_2;
- si w_1 et w_2 représentent respectivement deux descriptions instantanées Id_1 et Id_2 telles que $\mathsf{Id}_1 \underset{T}{\rightarrow} \mathsf{Id}_2$, alors il y a un arc de w_1 à w_2; cet arc n'est pas étiqueté. Notons que w_1 et w_2 ne représentent pas nécessairement des descriptions instantanées complètes de T, ils n'en représentent *a priori* que des préfixes.

Lemme 4.2.1 *G_T est un graphe automatique dont on peut calculer une structure automatique de façon effective.*

Idée de preuve. D'abord, $\overline{\mathcal{L}.\square^*}$ est rationnel. W est un automate qui le reconnaît et M_0 définit la relation d'équivalence triviale, i.e. diagonale. L'automate transversal doit reconnaître les couples de forme $(v_1, v_2) \in (\overline{\mathcal{L}.\square^*})^2$ qui codent les calculs élémentaires de T; pour ce faire, il doit d'abord deviner la transition qui s'applique à v_1, puis vérifier que l'on obtient bien v_2 en l'appliquant à v_1; ce qui est assûrément possible avec une mémoire finie.
□

Soit v un sommet de G_T; la distance qui le sépare de ε, i.e. la longueur minimale d'un chemin le conduisant à ε, est exactement la longueur v en tant qu'élément de $\overline{\mathcal{L}.\square^*}$; on la note $|v|$.

4.3 Machines de Turing auto-stabilisantes

Soit $T = (Q, \Gamma, \delta, q_{\mathsf{init}}, F, \square)$ une machine de Turing à une tête sans état final telle que pour tout $q \in Q$, $a \in A$ et $b \in \{0, 1\}$, $\delta(q, a, b)$ soit définie. Nous allons décrire une machine $\widetilde{T}$ munie de deux têtes qui simule le calcul de T. Le but de cette construction est de satisfaire la propriété énoncée dans le lemme 4.3.2; $\widetilde{T}$ devra être capable de contrôler elle-même son propre calcul, en ceci qu'elle doit se rendre compte elle-même si ce qu'elle est en train de calculer fait partie de son calcul normal, i.e. celui provenant de sa description instantanée initiale, ou non. Notons bien qu'étant donnée une machine de Turing quelconque, on n'est généralement pas en mesure d'anticiper son comportement sur n'importe quelle description instantanée. La simulation se fait en calculant les descriptions instantanées successives de T;

ces dernières sont stockées sur la bande de $\widetilde{T}$ au fur et à mesure, sans répétition. A chaque tour, $\widetilde{T}$ vérifie que la liste des descriptions instantanées stockées sur sa bande est bien celle escomptée; si cela n'est pas le cas, $\widetilde{T}$ en conclut qu'elle est en train de faire un mauvais calcul, elle passe alors dans un état spécial : l'état `BUG`.

Nous décrivons ici de façon informelle le détail du fonctionnement de $\widetilde{T}$. Le lecteur trouvera une construction plus rigoureuse en annexe de ce chapitre.

Etape 1. $\widetilde{T}$ commence par écrire la description instantanée initiale de T sur la bande. Plus précisément, la première tête écrit la chaîne $\#q_{\mathsf{init}}$ au début de la bande. Le symbole $\#$ sert de séparateur entre les descriptions instantanées de T stockée sur la bande. Notons que le mot q_{init} constitue la description instantanée initiale de T.

Etape 2. $\widetilde{T}$ vérifie que la bande contient les descriptions instantanées successives qui apparaissent lors du calcul de T à partir de q_{init}; elle vérifie aussi qu'aucune d'entre elles n'est répétée deux fois. Rigoureusement, elle contrôle que le contenu de la bande est de la forme

$$\#\mathsf{Id}_0\#\mathsf{Id}_1\#...\#\mathsf{Id}_k \tag{4.1}$$

où $\mathsf{Id}_0 = q_{\mathsf{init}}$; pour tout $i \in \{0, ..., k-1\}$, $\mathsf{Id}_i \underset{T}{\rightarrow} \mathsf{Id}_{i+1}$ et pour tout $j \neq i$, $\mathsf{Id}_j \neq \mathsf{Id}_i$. $\widetilde{T}$ vérifie que $\mathsf{Id}_i \underset{T}{\rightarrow} \mathsf{Id}_{i+1}$ en se servant de la première tête de lecture pour parcourir Id_i et de la seconde pour contrôler la forme de Id_{i+1}. Cette opération est possible : il suffit de vérifier la correspondance des symboles de Id_i et de Id_{i+1} trois par trois. Si ces conditions ne sont pas remplies, cela veut dire que $\widetilde{T}$ n'est pas sur le bon calcul, elle passe donc dans l'état `BUG`; Une fois dans cet état, $\widetilde{T}$ boucle indéfiniment sans modifier le contenu de la bande, ni les positions des têtes.

Etape 3. $\widetilde{T}$ calcule la description instantanée suivante dans le calcul de T qui est stocké sur la bande. Si elle n'apparaît pas déjà dans la liste, elle est écrite à la suite. Pour ce faire, la première tête se positionne au début de Id_k (on conserve les notations de (4.1) dans la description de l'étape 2) et la seconde se positionne au tout début de la bande. Soit Id_{k+1} la description instantanée de T obtenue après un pas de calcul à partir de Id_k. La première tête parcourt alors Id_k tandis que la seconde contrôle que Id_0 est bien distincte de Id_{k+1}; après cela la première tête retourne au début de Id_k et la seconde se positionne au début de Id_1; par la même méthode, $\widetilde{T}$ vérifie alors que Id_1 est distincte de Id_{k+1}; et ainsi de suite; cette opération est faite pour tous les Id_i. Finalement, si elles sont toutes différentes de Id_{k+1}, cette dernière est écrite à la suite de Id_k.

$\widetilde{T}$ retourne alors à l'étape 2.

Notons que $\widetilde{T}$ ne simule pas T au sens strict du terme. En effet, une description instantanée de T ne peut apparaître plusieurs fois sur la bande; ainsi, si la suite des descriptions instantanées décrivant le calcul de T est ultimement périodique, $\widetilde{T}$ inscrit les premières descriptions instantanées sur sa bande jusqu'à la première répétition, après quoi, le contenu de la bande de $\widetilde{T}$ n'est plus modifié. Notons que dans ce cas, la suite des descriptions instantanées

décrivant le calcul de $\widetilde{T}$ est elle-même ultimement périodique.

Lemme 4.3.1 (Comportement normal de $\widetilde{T}$)
Soit $\widetilde{\mathsf{Id}}$ une description instantanée décrivant $\widetilde{T}$ au début de la deuxième étape, avec un contenu de bande de la forme :

$$\#\mathsf{Id}_0\#\mathsf{Id}_1\#...\#\mathsf{Id}_k$$

où les Id_i sont les descriptions instantanées de T avec $\mathsf{Id}_0 = q_{\mathsf{init}}$, pour tout $i = 0, ..., k-1$, $\mathsf{Id}_i \underset{T}{\rightarrow} \mathsf{Id}_{i+1}$, et pour tout $i \neq j$, $\mathsf{Id}_i \neq \mathsf{Id}_j$.

Soit Id_{k+1} la description instantanée de T suivant directement Id_k. Si Id_{k+1} apparaît déjà dans $\widetilde{\mathsf{Id}}$, i.e. s'il existe $i \in \{0, ..., k\}$ tel que $\mathsf{Id}_{k+1} = \mathsf{Id}_i$, alors il existe une description instantanée $\widetilde{\mathsf{Id}}'$ avec le même contenu de bande que celui de $\widetilde{ID}$ telle que $\widetilde{\mathsf{Id}} \overset{+}{\underset{\widetilde{T}}{\rightarrow}} \widetilde{\mathsf{Id}}'$. Sinon, $\widetilde{\mathsf{Id}} \overset{+}{\underset{\widetilde{T}}{\rightarrow}} \widetilde{\mathsf{Id}}'\#\mathsf{Id}_{k+1}$.

Ce lemme est l'un des but de la construction de $\widetilde{T}$; la preuve se fait par récurrence à partir de la description complète de $\widetilde{T}$ qui est donnée en appendice de ce chapitre; elle est laissée au lecteur.

Le lemme suivant donne la principale propriété de $\widetilde{T}$: à partir de n'importe quelle description instantanée, soit $\widetilde{T}$ rejoint son calcul normal, i.e. celui provenant de sa description instantanée initiale au bout d'un temps fini, soit elle passe dans l'état `BUG` et y boucle indéfiniment.

Lemme 4.3.2 (Propriété principale des machines auto-stabilisantes)
Soit $\widetilde{\mathsf{Id}}$ une description instantanée de $\widetilde{T}$. Alors

- *soit il existe une description instantanée $\widetilde{\mathsf{Id}}'$ de $\widetilde{T}$ telle que $\widetilde{\mathsf{Id}} \overset{*}{\underset{\widetilde{T}}{\rightarrow}} \widetilde{\mathsf{Id}}'$ et $\widetilde{q}^1_{\mathsf{init}}\widetilde{q}^2_{\mathsf{init}} \overset{*}{\underset{\widetilde{T}}{\rightarrow}} \widetilde{\mathsf{Id}}'$ où $\widetilde{q}_{\mathsf{init}}$ est l'état initial de $\widetilde{T}$,*
- *ou bien $\widetilde{\mathsf{Id}}$ conduit $\widetilde{T}$ dans l'état `BUG` après un calcul fini.*

Ce lemme est une conséquence directe du lemme 4.5.3; la preuve est omise. Notons que les descriptions instantanées qui sont matières au premier cas ne sont pas seulement celles provenant du calcul normal de $\widetilde{T}$; par exemple, juste avant le début de l'étape de 2, on peut modifier la position des têtes de telle sorte que l'on obtient une description instantanée de $\widetilde{T}$ qui n'apparaît pas dans son calcul normal; de toute façon, au commencement de l'étape 2, les deux têtes sont positionnées en bout de bande (voir appendice), et donc le comportement de $\widetilde{T}$ n'en est pas affecté.

4.4 Indécidabilité du problème des bouts

Nous reprenons, dans le cadre des graphes, le concept de bout donné dans la définition 4. Le lemme suivant est facile, la démonstration est omise.

Lemme 4.4.1 *Soit G un graphe infini connexe et v_0, l'un de ses sommets; alors G a un et un seul bout si et seulement si le complémentaire de chaque boule centrée en v_0 ne contient qu'une seule composante connexe infinie. En particulier, si le complémentaire de chaque boule centrée en v_0 est connexe, alors G n'a qu'un seul bout.*

Notons que lorsque l'on s'occupe de la topologie des graphes, on ne se préoccupe pas de l'orientation des arcs; ainsi, nous parlons de connexité et non de forte connexité.

Preuve du théorème 9

La démonstration du théorème 9 est une réduction au problème de l'arrêt des machines de Turing, comme à l'ordinaire pour démontrer l'indécidabilité d'un problème donné. On se donne une machine de Turing à une tête que l'on va modifier légèrement. On considère ensuite la machine auto-stabilisante qui lui est associée, puis le graphe des configurations de cette dernière dont on modifiera quelque peu la géométrie. Nous démontrerons ensuite la proposition 4.4.1 qui affirme que ce graphe possède un bout et un seul si et seulement si la machine initiale ne s'arrête pas; ce qui implique trivialement le théorème 9.

Soit donc $T = (Q, \Gamma, \delta, q_{\text{init}}, F, \square)$ une machine de Turing à une tête. On commence par modifier T comme suit. Tout d'abord, les états de T appartenant à F ne sont plus considérés comme des états finaux, i.e. on pose $F = \emptyset$, sans pour autant éliminer les états en question; au lieu de cela, on ajoute de nouvelles transitions de telle sorte qu'une fois que T atteint un tel état, elle y reste indéfiniment sans modifier le contenu de la bande et sans déplacer sa tête de lecture/écriture. D'autre part, on ne veut pas que T puisse rester bloquée dans une configuration pour cause de transition manquante, i.e. on veut que δ soit définie sur tout triplet de $Q \times \Gamma \times \{0, 1\}$; pour atteindre cette propriété, on ajoute à T un nouvel état dit état "puits" dans lequel T passe toutes les fois qu'il manque une transition; une fois dans cet état, T boucle indéfiniment sans modifier ni le contenu de la bande, ni la position de sa tête de lecture/écriture. La troisième étape de la modification de T consiste à faire en sorte que tant qu'elle n'a pas atteint d'état originalement considéré comme final, elle occupe de plus en plus d'espace sur la bande; pour ce faire, il suffit par exemple d'imposer à T, entre chaque pas de calcul, d'aller écrire un nouveau symbole en fin de bande. Les détails de cette transformation sont omis. On obtient finalement une machine de Turing, que l'on note encore T, *qui ne s'arrête jamais et dont le calcul occupe un espace borné sur la bande si et seulement si la machine initiale s'arrête après un calcul fini.*

Considérons maintenant $\widetilde{T}$ la machine auto-stabilisante associée à T (voir section 4.3); puis $G_{\widetilde{T}}$, le graphe automatique des configurations de $\widetilde{T}$ défini dans la section 4.2. On modifie légèrement la géométrie de $G_{\widetilde{T}}$: d'abord, on ajoute une branche infinie constituée d'arcs radiaux étiquetés par un nouveau symbole noté $\texttt{BUG}$; cette branche démarre à la racine ε (voir figure 4.2). Ce nouveau symbole d'étiquette ne doit pas être confondu avec les symboles $\texttt{BUG}^1$ et $\texttt{BUG}^2$ qui font référence aux couples $(\texttt{BUG}, 1)$ et $(\texttt{BUG}, 2)$ utilisé pour coder les descriptions instantanées de $\widetilde{T}$. On ajoute ensuite à la structure automatique de $G_{\widetilde{T}}$ un nouvel automate $M_{\texttt{BUG}}$ définissant une famille d'arcs transversaux connectant tous les sommets de $G_{\widetilde{T}}$ qui définissent des descriptions instantanées dont l'état est l'état $\texttt{BUG}$. Plus précisément, $M_{\texttt{BUG}}$ reconnait les couples de mots de la forme (v_1, v_2) avec $v_1 = w.q^i.w'.q^j.w''.\square\square...\square$ de longueur $\ell > 0$ où $q = \texttt{BUG}$ et $v_2 = \underbrace{\texttt{BUG.BUG...BUG}}_{l}$. On obtient un graphe automatique que l'on note encore $G_{\widetilde{T}}$.

Lemme 4.4.2
Soient v et v' deux sommets de $G_{\widetilde{T}}$. Supposons qu'il existe deux descriptions instantanées

$\widetilde{\mathsf{Id}}$ *et* $\widetilde{\mathsf{Id}}'$ *dont* v *et* v' *représentent respectivement des préfixes telles que* $\widetilde{\mathsf{Id}} \xrightarrow[\widetilde{T}]{*} \widetilde{\mathsf{Id}}'$. *Il existe alors un chemin dans* $G_{\widetilde{T}}$ *qui connecte* v *et* v' *et qui reste à une distance supérieure ou égale à* $\min\{|v|, |v'|\}$ *de* ε.

Preuve. Soit $\widetilde{\mathsf{Id}} = \widetilde{\mathsf{Id}_0} \to \widetilde{\mathsf{Id}_1} \to ... \to \widetilde{\mathsf{Id}_k} = \widetilde{\mathsf{Id}}'$ la suite de descriptions instantanées codant le calcul de $\widetilde{T}$ de $\widetilde{\mathsf{Id}}$ à $\widetilde{\mathsf{Id}}'$. Soit $\ell_{\max} = \max_{0 \le i \le k}\{|\widetilde{\mathsf{Id}_i}|\} = |\widetilde{\mathsf{Id}}'|$; notons que $\ell_{\max} \ge \min\{|v|, |v'|\}$. Pour tout $i \in \{0, k\}$, soit $v_i = \widetilde{\mathsf{Id}_i}\square^{\ell_{\max} - |\widetilde{\mathsf{Id}_i}|}$. Alors, par définition de $G_{\widetilde{T}}$, pour chaque $i = 0, ..., k-1$, il existe un arc transversal connectant v_i à v_{i+1}et le chemin $v_0 v_1 ... v_k$ reste à distance supérieure ou égale à $\min\{|v|, |v'|\}$. Par ailleurs, v (respectivement v') est connecté à v_0 (respectivement v_k)par un chemin d'arcs radiaux.
□

Proposition 4.4.1
$G_{\widetilde{T}}$ possède un et un seul bout si et seulement si la machine initiale T ne s'arrête jamais; dans le cas contraire, $G_{\widetilde{T}}$ possède exactement deux bouts.

Preuve. **Supposons que T n'atteint jamais d'état final.** Notre but est de prouver que $G_{\widetilde{T}}$ ne possède qu'un seul bout. Soit $k > 0$ et soit v un sommet de $G_{\widetilde{T}}$ situé à distance supérieure à k de ε. On montre que v est connecté à la branche infinie $\mathtt{BUG}^\omega$ par un chemin restant à distance supérieure à k de ε. Soit $\widetilde{\mathsf{Id}}$ une description instantanée de $\widetilde{T}$ dont v représente un préfixe. Conformément au lemme 4.3.2, il y a deux cas à envisager :

1. soit il existe une description instantanée $\widetilde{\mathsf{Id}}'$ telle que $\widetilde{\mathsf{Id}} \xrightarrow[\widetilde{T}]{*} \widetilde{\mathsf{Id}}'$ et $\widetilde{q}^1_{\mathsf{init}}\widetilde{q}^2_{\mathsf{init}} \xrightarrow[\widetilde{T}]{*} \widetilde{\mathsf{Id}}'$ où $\widetilde{q}_{\mathsf{init}}$ est l'état initial de $\widetilde{T}$,
2. ou bien $\widetilde{\mathsf{Id}}$ conduit $\widetilde{T}$ dans l'état BUG après un calcul fini.

Cas 1: Rappelons que T a été modifiée de telle sorte qu'elle occupe de plus en plus d'espace au fur et à mesure de son calcul. Cela implique que la suite des descriptions instantanées qui code ce calcul ne peut pas être ultimement périodique. Ainsi, la suite de descriptions instantanées de T qui est stockée par $\widetilde{T}$ s'accroît strictement, on peut donc choisir $\widetilde{\mathsf{Id}}'$ de longueur supérieure à k. En se penchant sur l'étape 2 de l'algorithme des machines auto-stabilisantes, on montre sans difficulté que l'on peut choisir $\widetilde{\mathsf{Id}}'$ de telle sorte que les deux têtes se trouvent toutes les deux à droite de la k-ième cellule.
Soit v' le sommet de $G_{\widetilde{T}}$ qui est associé à $\widetilde{\mathsf{Id}}'$. Par le lemme 4.4.2, il existe un chemin de v à v' restant à distance supérieure à k de ε.
Soient w_1, w_2, w_3 les mots sur l'alphabet de bande de $\widetilde{T}$, q l'état de $\widetilde{T}$ et $i, i' \in \{1, 2\}$, $i \neq i'$ tels que $\widetilde{\mathsf{Id}}' = w_1 q^i w_2 q^{i'} w_3$. Comme les deux têtes de lecture/écriture se trouvent à droite de la k-ième cellule, $|w_1| > k$. Soit v'' le sommet associé à w_1. Triviallement, v' est connecté à v'' en parcourant en sens inverse un chemin d'arcs radiaux qui reste à distance supérieure à k de ε. Soit $\widetilde{\mathsf{Id}}'' = w_1 \mathtt{BUG}^1 \mathtt{BUG}^2$. En parcourant successivement deux arcs radiaux respectivement étiquetés par $\mathtt{BUG}^1$ et $\mathtt{BUG}^2$ en direction opposée à ε, on connecte v'' au sommet représentant $\widetilde{\mathsf{Id}}''$ qui, lui-même, est connecté à la branche infinie $\mathtt{BUG}^{|\widetilde{\mathsf{Id}}''|}$ par un arc transversal reconnu

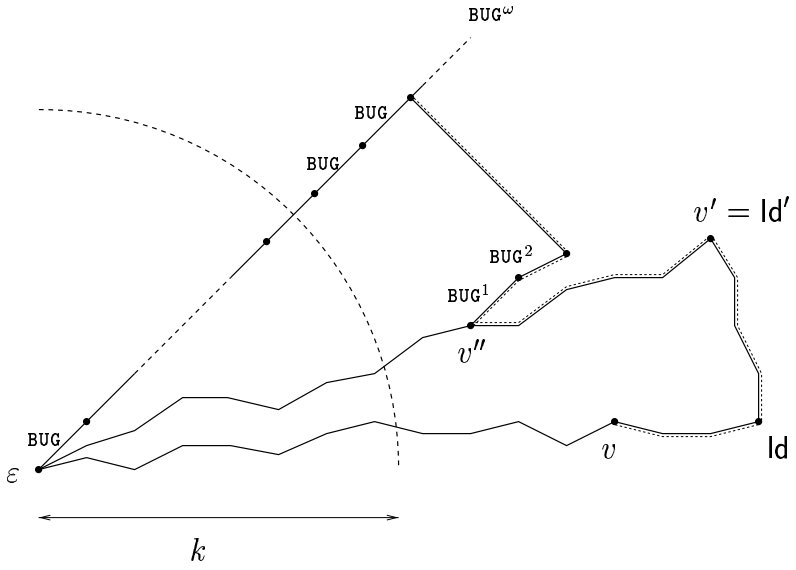

FIG. 4.2: *Cas* 1

par $M_{\texttt{BUG}}$. On a finalement connecté v à la branche infinie $\texttt{BUG}^\omega$, tout en restant à distance supérieure à k de ε (voir figure 4.2).
Case 2*:* Soit $\widetilde{\mathsf{Id}}_{\texttt{BUG}}$ la première description instantanée de $\widetilde{T}$ dont l'état est l'état BUG qui est accessible à partir de $\widetilde{\mathsf{Id}}$. Par le lemme 4.4.2, v est alors connecté au sommet associé à $\widetilde{\mathsf{Id}}_{\texttt{BUG}}$ par un chemin restant à distance supérieure à k de ε. Et par construction, il existe un arc transversal entre $\widetilde{\mathsf{Id}}_{\texttt{BUG}}$ et $\texttt{BUG}^{|\widetilde{\mathsf{Id}}_{\texttt{BUG}}|}$.

Le complémentaire de la boule centrée en ε de rayon k est donc connecté; ce qui prouve avec le lemme 4.4.2 que $G_{\widetilde{T}}$ ne possède qu'un seul bout.

Supposons que T atteint un état final par un calcul fini. Soient $k > 0$ et v un sommet de $G_{\widetilde{T}}$ situé à distance supérieure à k de ε. Soit $\widetilde{\mathsf{Id}}$ une description instantanée dont v représente un préfixe.

Comme précédemment, considérons les deux cas prévus par le lemme 4.3.2. Si $\widetilde{\mathsf{Id}}$ conduit $\widetilde{T}$ à l'état BUG, alors v est connecté à $\texttt{BUG}^\omega$ par un chemin restant à distance supérieure à k de ε.

Supposons donc que $\widetilde{T}$ ne passera jamais dans l'état BUG en calculant à partir de $\widetilde{\mathsf{Id}}$. Par hypothèse, T atteint un état final et par construction, elle y reste indéfiniment, laissant le contenu de la bande et les positions des têtes inchangées: la suite des descriptions instantanées de T entre dans une boucle. Soit Id la première description instantanée qui est répétée dans le calcul de T.

Un fois que $\widetilde{T}$ a écrit Id sur la bande, elle n'écrit plus aucune autre description instantanée de T (voir lemme 4.3.1). L'espace occupé par $\widetilde{T}$ cesse alors de grandir, il est borné; cela implique que la suite des descriptions instantanées de $\widetilde{T}$ est ultimement périodique. Soit $\widetilde{\mathsf{Id}}''$ la première description instantanée de $\widetilde{T}$ qui est répétée.

On montre que v est connecté à la branche infinie $\widetilde{\mathsf{Id}}''\Box^\omega$: par le lemme 4.3.2, il existe une description instantanée de $\widetilde{T}$ $\widetilde{\mathsf{Id}}'$ telle que $\widetilde{\mathsf{Id}} \xrightarrow[\widetilde{T}]{*} \widetilde{\mathsf{Id}}'$ and $\widetilde{q}^1_{\mathsf{init}}\widetilde{q}^2_{\mathsf{init}} \xrightarrow[\widetilde{T}]{*} \widetilde{\mathsf{Id}}'$. Ainsi, $\widetilde{\mathsf{Id}}' \xrightarrow[\widetilde{T}]{*} \widetilde{\mathsf{Id}}''$ et donc $\widetilde{\mathsf{Id}} \xrightarrow[\widetilde{T}]{*} \widetilde{\mathsf{Id}}''$. Par le lemme 4.4.2, v est connecté à $\widetilde{\mathsf{Id}}''$ par un chemin restant à distance supérieure à k de ε.

Nous avons donc prouvé que tout sommet est connecté à la branche infinie $\mathtt{BUG}^\omega$ ou bien à $\widetilde{\mathsf{Id}}''\Box^\omega$ par un chemin restant éloigné de ε. Il reste à montrer que ces deux branches représentent deux bouts différents.

Soit $k > |\widetilde{\mathsf{Id}}''|$. Supposons qu'il existe un sommet v de la forme $\widetilde{\mathsf{Id}}''\Box^\ell$ avec $\ell > 0$ qui est connecté à la branche $\mathtt{BUG}^\omega$ par un chemin non-orienté α restant à distance supérieure à k de ε.

Soit $\mathcal{I}$ l'ensemble des descriptions instantanées de $\widetilde{T}$ qui conduisent à $\widetilde{\mathsf{Id}}''$. Comme $\widetilde{T}$ ne peut pas écrire le symbole de blanc, cet ensemble est de cardinal fini. Soit v' le premier sommet de α qui représente un préfixe d'une description instantanée $\widetilde{\mathsf{Id}}_{v'}$ qui n'appartient pas à $\mathcal{I}$. Notons que $v' \neq v$. Soit v'' le sommet de α qui précéde directement v'. Par construction de v', toute description instantanée dont v'' est un préfixe est dans $\mathcal{I}$. Mais $\mathcal{I}$ est fini; et on montre facilement que si un sommet donné est préfixe de plusieurs descriptions instantanées, il en est alors préfixe d'une infinité. Cela implique que v'' est de la forme $\widetilde{\mathsf{Id}}_{v''}\Box^{\ell'}$ où $\widetilde{\mathsf{Id}}_{v''}$ est une description instantanée complète et $\ell' > 0$. De plus, par définition de $\mathcal{I}$, $\widetilde{\mathsf{Id}}_{v''} \to \widetilde{\mathsf{Id}}''$ et donc, $|\widetilde{\mathsf{Id}}_{v''}| \leq \widetilde{\mathsf{Id}}'' < k$. Supposons maintenant que v' et v'' soient connectés par un arc radial; comme $|v'| \geq k$, $\widetilde{\mathsf{Id}}_{v''}\Box$ est alors préfixe de v'; par conséquent $\widetilde{\mathsf{Id}}_{v'} = \widetilde{\mathsf{Id}}_{v''}$ ce qui contredit le fait que $\widetilde{\mathsf{Id}}_{v'}$ ne conduit pas à $\widetilde{\mathsf{Id}}''$. Ainsi, si v' et v'' sont connectés, c'est par un arc transversal, ce qui veut dire que $\widetilde{\mathsf{Id}}_{v'} \xrightarrow[\widetilde{T}]{} \widetilde{\mathsf{Id}}_{v''}$ ou bien $\widetilde{\mathsf{Id}}_{v''} \xrightarrow[\widetilde{T}]{} \widetilde{\mathsf{Id}}_{v'}$; comme $\widetilde{\mathsf{Id}}_{v''} \in \mathcal{I}$, les deux possibilités conduisent au fait que $\widetilde{\mathsf{Id}}_{v'} \xrightarrow[\widetilde{T}]{} \widetilde{\mathsf{Id}}''$, ce qui est une contradiction.
□

On obtient comme conséquence immédiate le résultat suivant :

Théorème 9
Le problème de savoir si un graphe automatique possède un et un seul bout est indécidable.

Question : *On sait que le nombre de bouts d'un groupe hyperbolique au sens de Gromov est calculable [Ger99]; qu'en est-il pour les graphes automatiques hyperboliques au sens de Gromov ?*

4.5 $D0L$-systèmes de graphes et graphes automatiques

4.5.1 $D0L$-systèmes de graphes

Les transformations de graphes dont nous nous occupons ici, i.e. les *$D0L$-systèmes de graphes*, généralisent aux graphes le concept de L-système [RS80]; notons que ce dernier a

déjà été étendu aux structures multi-dimensionnelles sous le nom de systèmes de génération de cartes, i.e. $M0L$-systèmes (cf. [LR78, DDL82, Lin86]); les $D0L$-systèmes que nous introduisons ici généralisent le concept de *ions* (cf. [Pey81, Pey86, Nar93]).

Il s'agit de substitutions simultanées, déterministes et indépendantes du contexte des sommets et des arcs. D'abord, chaque sommet est remplacé par un graphe fini qui est fonction de son étiquette; puis chaque arc donne naissance à plusieurs arcs connectant les sommets issus de son origine avec les sommets issus de son extrémité; ces arcs sont définis relativement à l'étiquette de l'arc original. Les substitutions de tous les sommets puis de tous les arcs se font *simultanément.*

Définition 15 ($D0L$-Systèmes de graphes)
*Un $D0L$-**système de graphe** δ est défini par un 3-uplet (L, Δ_V, δ_E) où :*

- *$L = L_V \cup L_E$ est un ensemble fini de symboles. L_V est l'ensemble des **étiquettes de sommets** et L_E est l'ensemble des **étiquettes d'arcs**;*
- *Δ_V est une application associant à chaque étiquette de sommet un graphe fini de type 0 dont les sommets (respectivement les arcs) sont étiquetés sur L_V (respectivement sur L_E). Ces graphes sont équipés d'une structure supplémentaire : un ordre linéaire sur l'ensemble de leurs sommets;*
- *Δ_E est une application associant à chaque étiquette d'arc un graphe fini biparti dont les deux ensembles de sommets sont linéairement ordonnés; le couple formé par les deux parties de sommets est lui-même ordonné. Les arcs sont étiquetés sur L_E; les sommets ne sont pas étiquetés.*

*Nous dirons qu'un graphe de type 0 est un **δ-graphe** si ses sommets sont étiquetés sur L_V et ses arcs sur L_E. L'application de δ sur un δ-graphe G consiste à effectuer les transformations parallèles suivantes :*

- *chaque sommet $v \in V$ est remplacé par $\Delta_V(lab(v))$;*
- *chaque arc $e \in E$ est remplacé par $\Delta_E(lab(e))$. Les sommets de la première partie de ce dernier sont recollés sur les sommets du graphe issu de l'origine de e suivant les ordres linéaires; s'il y a trop de sommets a recoller, les sommets excédentaires sont supprimés de $\Delta_E(lab(e))$, ainsi que les arcs auxquels ils appartiennent. Les sommets de la seconde partie de $\Delta_E(lab(e))$ sont recollés sur les sommets du graphe issu de l'extrémité de e suivant une procédure similaire.*

*Une fois que les substitutions de tous les sommets et de tous les arcs sont faites, les différents ordres linéaires sur les sous-ensembles de sommets deviennent inutiles et sont oubliés. Le graphe obtenu est noté $\delta(G)$. Une suite de graphe $(\Gamma_n)_{n\geq 0}$ est appelée une $D0L$-**suite** s'il existe un $D0L$-système de graphe δ tel que pour tout $n \geq 0$, $\Gamma_{n+1} = \delta(\Gamma_n)$.*

La figure 4.3 décrit les règles d'un $D0L$-système de graphe qui permet d'engendrer des arbres binaires. Ici, $L_V = \{i, e\}$ et $L_E = \{i\}$; notons que $L = L_V$. Les règles de substitution de sommet sont décrites dans la partie gauche de la figure; le système de substitution d'arc

est décrit dans la partie droite, il n'y a qu'une seule règle. Un sommet étiqueté par i (comme **i**nterne) n'est pas modifié : il est remplacé par un graphe sans arc dont le sommet unique est étiqueté par i. Un sommet étiqueté par e (comme **e**xterne) est substitué par un graphe à trois sommets dont le premier est étiqueté par i et les deux autres sont étiquetés par e. Pour ce qui concerne l'unique règle de substitution d'arc, elle spécifie qu'un arc étiqueté par i reste inchangé; après substitution, il connecte le premier sommet du graphe issu de son origine au premier sommet du graphe issu de son extrémité. Les premières itérations de ce système sont représentées dans la figure 4.4.

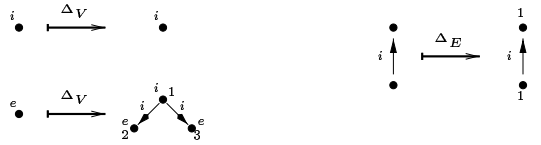

FIG. 4.3: *Exemple de D0L-système de graphes*

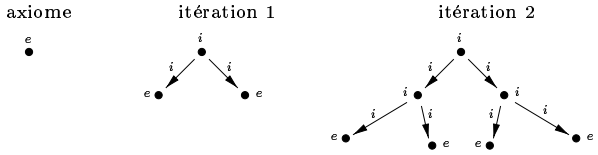

FIG. 4.4: *Premières itérations du D0L-système décrit dans la figure 4.3*

Les figures 4.5 et 4.6 décrivent un autre exemple de $D0L$-système qui engendre des grilles carrées.

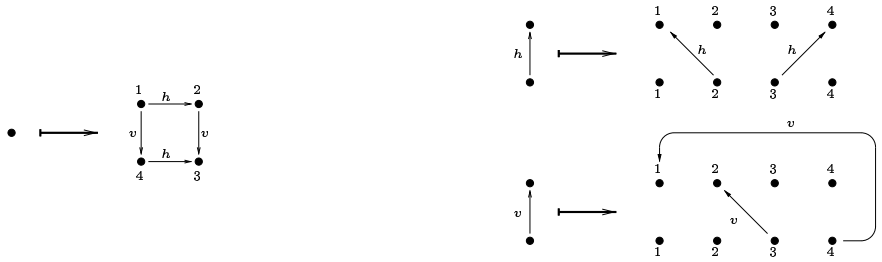

FIG. 4.5: *Un D0L-système de graphe qui engendre une suite de grilles carrées. Il n'y a qu'un seul type d'étiquette pour les sommets, c'est pourquoi il n'est pas indiqué ici.*

4.5.2 Connexité dans les $D0L$-suites de graphes et bouts de graphes automatiques

Nous allons voir dans cette partie comment les notions de $D0L$-systèmes de graphes et de graphes automatiques sont liées; une structure automatique apparaît de façon naturelle comme un outil pour décrire une $D0L$-suite de graphes finis dans sa globalité.

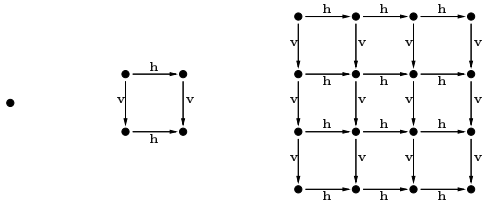

FIG. 4.6: *Une D0L-suite de grille engendrée par le D0L-système décrit dans la figure 4.5*

Nous nous intéresserons ici à une classe spéciale de graphes automatiques, ceux qui satisfont les conditions suivantes :

- la relation d'équivalence rationnelle qui participe à la définition des sommets est triviale, i.e. diagonale;
- les arcs transversaux ne connectent que des couples de sommets associés à des mots de même longueur;
- tous les états des automates définissant les arcs transversaux sont des états finaux.

Ces conditions impliquent de bonnes propriétés géométriques. Remarquons d'abord que pour les graphes possédant ces propriétés, les termes d'arc radial et d'arc transversal prennent tout leur sens; la sphère de rayon n centrée en ε est formée par les sommets associés aux mots de longueur n ainsi que des arcs transversaux les connectant. La troisième condition nous assure que dans un tel graphe, tout arc transversal laisse dans les sphères inférieures des traces qui sont matérialisées par des arcs transversaux qui sont en quelque sorte ses aïeuls.

Le résultat suivant est fondamental dans cette partie, il montre comment les $D0L$-suites de graphes sont décrites de façon globale par des structures automatiques.

Proposition 4.5.1
Quitte à modifier les étiquettes, la suite des sphères d'un graphe automatique satisfaisant les conditions précédentes est une D0L-suite de graphes. Réciproquement, toute D0L-suite de graphe dont le graphe initial n'a qu'un seul sommet est la suite des sphères d'un graphe automatique.

Preuve. Soit G un graphe automatique satisfaisant les condition précédentes, défini par une structure automatique $(W, M_0, \{M_\ell\}_{\ell \in L})$. En se servant des méthodes usuelles de déterminisation, on peut considérer sans perte de généralité que les automates qui définissent la structure automatique de G sont tous déterministes. Nous allons construire un $D0L$-système de graphe δ_G tel que pour tout $n \geq 0$, la $(n+1)$-ième sphère de G est le résultat de l'application de δ_G à la n-ième.

L'ensemble des étiquettes de sommets associé à δ_G est Q_W, l'ensemble des états de W; l'ensemble des symboles d'étiquettes d'arcs est $Q_W \times (\bigcup_\ell M_\ell) \times Q_W$. Commençons par modifier les étiquettes de G : la lecture d'un sommet v donné conduit W dans un état q_v; on définit la nouvelle étiquette de v comme étant q_v. De façon similaire, on définit de nouvelles étiquettes pour les arcs que l'on choisit dans $Q_W \times (\bigcup_\ell M_\ell) \times Q_W$: soit e un arc transversal

de G initialement étiqueté par ℓ; cet arc connecte un couple de sommets (v_1, v_2); soient q_1 et q_2 les états dans lesquels v_1 et v_2 conduisent respectivement W (par abus de notation, on confond ici un sommet avec le mot qui le code, ce qui ne pose pas de problème étant donné que M_0 définit une relation d'équivalence triviale); soit q_ℓ l'état dans lequel (v_1, v_2) conduit M_ℓ. On définit alors la nouvelle étiquette de e comme étant le triplet (q_1, q_ℓ, q_2).

Les règles de substitution de δ_G sont définies en fonction des transitions de W et des M_ℓ. Regardons d'abord les règles de substitution de sommet. On commence par choisir un ordre linéaire sur A, l'alphabet de W; soit q un élément de Q_W; c'est un symbole d'étiquette de sommet pour δ_G; comme W est déterministe, l'ordre sur A induit un ordre linéaire sur l'ensemble des transitions de W d'origine q. $\Delta_V(q)$ est alors défini comme étant le graphe muni d'autant de sommets qu'il y a de transitions dans W d'origine q; l'ordre linéaire sur ces sommets est induit par l'ordre sur les dites transitions, qui a été défini précédemment; leurs étiquettes, qui sont prises parmi les états de W, sont les cibles des transitions en question. $\Delta_V(q)$ n'est pourvu d'aucun arc.

La construction des règles de substitution d'arc suit des idées similaires. Considérons un triplet (q_1, q_ℓ, q_2) où $q_i \in Q_W$ $(i = 1, 2)$ et $q_\ell \in Q_{M_\ell}$; c'est un symbole d'étiquette d'arc pour δ_G. $\Delta_E(q_1, q_\ell, q_2)$ est un graphe biparti dont la première partie de sommet (respectivement la seconde) est formée d'autant de sommets qu'il y a de transitions d'origine q_1 (respectivement q_2) dans W. Comme précédemment, les ordres linéaires définis sur chacune de ces parties sont induits par les ordres définis sur les ensembles de transitions de W correspondants. Comme il est convenu dans la définition 15, les sommets de $\Delta_E(q_1, q_\ell, q_2)$ ne sont pas étiquetés; ses arcs sont définis comme suit : soit (v_1, v_2) un couple de sommets de $\Delta_E(q_1, q_\ell, q_2)$, v_1 appartenant au premier sous-ensemble de sommet et v_2, au second; soit $a_1 \in A$ (respectivement $a_2 \in A$) le dernier symbole du mot représentant v_1 (respectivement v_2). Si dans M_ℓ, il existe une transition d'origine q_ℓ qui est étiquetée par le couple (a_1, a_2), on ajoute à $\Delta_E(q_1, q_\ell, q_2)$ un arc connectant v_1 à v_2, étiqueté par le triplet (q'_1, q'_ℓ, q'_2) où $q'_1 \in Q_W$ (respectivement $q'_2 \in Q_W$) est l'état dans lequel a_1 (respectivement a_2) conduit W à partir de q_1 (respectivement à partir de q_2); et $q'_\ell \in Q_{M_\ell}$ est l'état dans lequel (a_1, a_2) conduit M_ℓ à partir de q_ℓ. On considère alors la $D0L$-suite de graphe $(G_n)_n$ engendrée par δ_G à partir du graphe G_0 formé d'un seul sommet étiqueté par q_0, l'état initial de W, et pour tout ℓ, une boucle étiquetée par $(q_0, q_{\ell 0}, q_0)$ où $q_{\ell 0}$ est l'état initial de M_ℓ. Nous laissons au lecteur le soin de vérifier que, aux étiquettes près, G_n est exactement la sphère de G centrée en ε de rayon n.

Passons à la réciproque. Soit δ un $D0L$-système de graphe associé à une $D0L$-suite de graphes finis $(G_n)_n$ dont le premier n'est constitué que d'un seul sommet. Nous allons construire un graphe automatique qui définit cette $D0L$-suite de graphe au sens suivant. Considérons G_0; l'application de δ à G_0 donne pour résultat G_1. Considérons le graphe réalisé par l'union disjointe de G_0 et de G_1 en connectant le sommet de G_0 à tous les sommets de G_1; on ajoute à ce graphe un copie de $G_2 = \delta(G_1)$, puis on connecte chaque sommet de G_1 aux sommets de G_2 auxquels il a donné naissance via δ. En continuant ce procédé, on construit par limite inductive un graphe infini G qui est automatique. Nous allons construire une structure automatique $(W, M_0, \{M_\ell\}_{\ell \in L})$ qui le définit.

L'ensemble des états de W est défini comme étant l'ensemble des symboles d'étiquettes de sommets de δ. Ses transitions sont construites en fonction des règles de substitution de

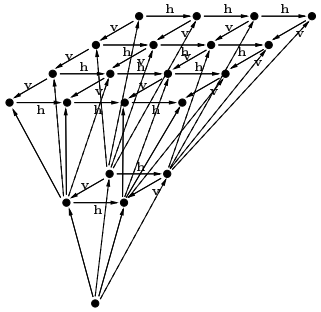

FIG. 4.7: *Le graphe automatique codant la $D0L$-suite de grilles décrites dans la figure 4.6*

sommet de δ; on suit l'idée réciproque de celle décrite précédemment : on définit l'alphabet de W comme étant l'ensemble des entiers inférieurs au nombre maximum de sommets produits par la subdivision d'un sommet par δ. Soit ℓ_v un symbole d'étiquette de sommet de δ, c'est aussi un état de W; on crée, dans W, autant de transitions qu'il y a de sommets dans le graphe associé à ℓ_v par δ, i.e. $\Delta_V(\ell_v)$; une telle transition est étiquetée par l'entier qui est le rang du sommet de $\Delta_V(\ell_v)$ auquel elle est associée; sa cible est le symbole d'étiquette du sommet en question. Tous les états de W sont des états finaux. M_0 quant-à-lui, est trivial, i.e. il définit la relation d'équivalence diagonale sur $\mathcal{L}(W)$.

Passons à la construction des M_ℓ. Premièrement, nous devons modifier δ de telle sorte que tous les arcs de chacun des $\Delta_E(\ell)$ soient orientés du premier sous-ensemble de sommet vers le second. On réalise cette transformation en ajoutant à l'ensemble des symboles d'étiquettes d'arcs L_E de δ, une copie disjointe de lui-même $L_E^{-1} = \{\ell^{-1} \mid \ell \in L_E\}$ dont nous nous servirons pour coder les arcs dont l'orientation doit être inversée. Dans chaque $\Delta_E(\ell)$, chaque arc qui est orienté du second sous-ensemble de sommets vers le premier est inversé ainsi que son étiquette, i.e. ℓ_e est remplacée par ℓ_e^{-1}. Δ_E associe à chaque $\ell^{-1} \in L_E^{-1}$ un graphe biparti qui est une copie de $\Delta_E(\ell)$ dans laquelle l'orientation du couple de sous-ensembles de sommets est inversée et chaque arc orienté du second sous-ensemble de sommet vers le premier est inversé, ainsi que son étiquette. On obtient finalement un nouveau $D0L$-système que l'on note δ' qui est tel que tous les arcs de chacun des $\Delta'_E(\ell)$ sont orientés du premier sous-ensemble de sommets vers le second. Notons que pour tout δ-graphe H, on obtient $\delta(H)$ à partir de $\delta'(H)$ en inversant l'orientation de chaque arc étiqueté par un élément de L_E^{-1} ainsi que son étiquette.

Nous allons maintenant construire la famille des automates à arcs transversaux $\{M'_\ell \mid \ell \in L_E \cup L_E^{-1}\}$ associée à δ'. Pour ce faire, considérons le graphe M construit de la façon suivante : l'ensemble des sommets de M est définit comme étant l'ensemble des symboles d'étiquettes d'arcs de δ', i.e. $L_E \cup L_E^{-1}$. Considérons l'un de ces symboles d'étiquettes d'arcs ℓ_e; on ajoute à M un arc étiqueté par un couple d'entier (i, j) connectant ℓ_e à ℓ'_e toute les fois qu'il existe dans $\Delta_E(\ell_e)$ un arc étiqueté par ℓ'_e connectant le i-ième sommet du premier sous-ensemble de sommet de $\Delta_E(\ell_e)$ au j-ième sommet du second sous-ensemble; M code ainsi les règles de substitution d'arc de δ'. On considère alors $\widetilde{M}$ le graphe obtenu comme union disjointe de M et de W, dans lequel toute étiquette d'arc de la forme i est remplacé par (i, i); pour chaque arc e de chaque $\Delta_V(\ell_v)$, on ajoute un arc dans $\widetilde{M}$ connectant $\ell_v \in V_W$ à $\ell_e \in V_M$,

où ℓ_e est l'étiquette de e; cet arc est étiqueté par (i,j), où i (respectivement j) est le rang de l'origine de e (respectivement sa cible) relativement à l'ordre linéaire sur l'ensemble des sommets de $\Delta_V(\ell_v)$. Finalement, pour chaque $\ell \in L_E \cup L_E^{-1}$, on définit M'_ℓ comme étant une copie de $\widetilde{M}$ avec comme état initial, l'état initial de W et $\ell \in V_M$ comme unique état final. Pour chaque $\ell \in L_E$, on construit maintenant M_ℓ, l'automate à arcs transversaux associé à δ, en se servant de M'_ℓ et de $M'_{\ell^{-1}}$: c'est l'union disjointe des deux; l'étiquette de chaque transition de $M'_{\ell^{-1}}$ qui est de la forme (i,j) est remplacée par (j,i). Nous laissons au lecteur le soin de vérifier que la structure automatique que nous avons construite définit bien G.
□

Soit G un graphe automatique défini par une structure automatique $(W, M_0, \{M_\ell\}_{\ell \in L})$ satisfaisant les conditions de la proposition 4.5.1. On montre facilement que si toutes les sphères de G sont connexes, alors G possède un bout et un seul; pour obtenir la réciproque, i.e. la proposition 4.5.2, il faut couper les branches finies de G, i.e. les chemins d'arcs radiaux dans G qui ne sont préfixe d'aucun chemin infini. Il se peut en effet qu'un chemin d'arcs radiaux ne puisse se prolonger en un chemin infini; un tel phénomène correspond dans W à l'existence d'états à partir desquels aucun circuit n'est accessible. On règle ce problème en supprimant simplement ces états; ce qui peut se faire de façon effective sans trop de difficultés; on laisse au lecteur le soin de le vérifier. On obtient alors un graphe que l'on note G' qui est encore automatique. On a le résultat suivant :

Proposition 4.5.2 *On conserve les notations précédentes. G possède un bout et un seul si et seulement si toute les sphères de G' sont connexes.*

Preuve. Montrons d'abord que G' possède autant de bouts que G. Pour voir cela, on remarque d'abord que G' possède les mêmes branches infinies que G; de fait, on peut considérer G' comme un sous-graphe de G. Ce qui pourrait se passer est que deux branches infinies qui sont équivalentes dans G ne le sont pas dans G'. Considérons donc deux branches infinies $\alpha_1 = v_0^1 v_1^1 ... v_k^1 ...$ et $\alpha_2 = v_0^2 v_1^2 ... v_k^2 ...$ $(v_0^1 = v_0^2 = \varepsilon)$. Nous allons vérifier que si α_1 et α_2 sont équivalentes dans G, i.e. si elles sont connectées dans le complémentaire dans G de n'importe quelle boule centrée en ε, alors pour tout k, il existe un chemin de S_k, la sphère de G centrée en ε de rayon k, qui connecte x_k^1 et x_k^2 et qui n'emprunte que des sommets éléments de branches infinies. Supposons donc α_1 et α_2 équivalentes. Pour tout $k' > k$, on choisit un chemin $p'_{k'}$ qui connecte $v_{k'}^1$ à $v_{k'}^1$ et qui reste à distance supérieure à k' de ε; on associe alors à $p'_{k'}$ un chemin $p_{k'}$ de S_k qui connecte v_k^1 à v_k^1 défini comme suit : on considère le chemin de S_k constitué de tout les ancêtres des sommets de $p'_{k'}$; on élimine de ce chemin toutes les boucles, on obtient alors $p_{k'}$. Notons que $p_{k'}$ est bien un chemin puisque tous les états de tous les M_ℓ sont des états finaux. Maintenant, comme S_k est finie, le nombre de chemins sans boucle connectant v_k^1 à v_k^1 est aussi fini. Par conséquent, l'un d'eux est égal à $p_{k'}$ pour une infinité de k'; cela implique que tous les sommets constituant ce chemin sont chacun ancètre d'une infinité de sommets, et donc préfixe d'une branche infinie. Ce chemin, qui *a priori* est un chemin de G est aussi un chemin de G'; nous avons prouvé que G et G' ont le même nombre de bouts.

Il reste à voir que G' possède un et un seul bout si et seulement si toutes ses sphères sont connexes. On montre juste que si G possède un seul bout, alors toutes ses sphères sont connexes; la réciproque est facile et laissée au lecteur. Supposons donc qu'une sphère S'_k de

G' n'est pas connexe; soient z_1 et z_2 deux sommets de S'_k qui ne sont pas connectés dans S'_k. On choisit deux branches infinies α_1 et α_2 dont z_1 et z_2 sont respectivement des préfixes. Alors ces deux branches repésentent deux bouts différents; en effet, un chemin quelconque les connectant dans le complémentaire d'une boule centrée en ε de rayon supérieur à k induirait un chemin les connectant dans S'_k.
□

Il est facile de montrer que le graphe automatique construit à partir d'une machine de Turing dans la preuve de la proposition 4.4.1 satisfait les deux premières conditions de la proposition 4.5.1. Pour qu'il satisfasse la troisième, on considère tous les états de tous les automates de sa structure automatique comme des états finaux. Si cela ajoute des arcs transversaux, cela ne modifie pas fondamentalement sa géométrie et en particulier, on vérifie facilement que le nombre de bouts est inchangé. Ceci étant, après élimination des branches finies, on peut appliquer la proposition 4.5.2 pour obtenir le résultat suivant :

Théorème 10
La question consistant à savoir si tous les graphes obtenus en itérant un D0L-système de graphe sont connexes est indécidable.

4.5.3 Emulation des grammaires de remplacement de sommets

Remplacement de sommets. Soit $\Gamma = (V, E, L, vert, lab)$ un graphe et $v \in V$. On suppose que L est l'union disjointe de deux ensembles L_V et L_E qui sont respectivement l'ensemble des symboles d'étiquettes de sommets et l'ensemble des symboles d'étiquettes d'arcs. Suivant [ER97, Chap. 1.3], par *graphe avec plongement*, on entend un couple (D, C) où D est un graphe et $C \subset L_V \times L_E \times L_E \times V_D \times \{in, out\}$; C est appelée la *relation de connection* de (D, C). Le *remplacement* de v par D relativement à C est alors défini comme suit : v est supprimé de Γ et remplacé par une copie de D; Pour chaque *instruction de connection* $(\sigma, \alpha, \beta, x, out) \in C$, tout arc de Γ étiqueté par α connectant v à un sommet $w \neq v$ étiqueté par σ donne naissance à un arc étiqueté par β allant de x à w; les arcs entrant sont traités de façon similaires par les instruction de connection comportant un "*in*" à la place de "*out*".

Grammaires VR déterministes. On appelle *grammaire edNCE* [ER97, CER93, CE95] tout tuple (L_V, L'_V, L_E, P, S) où L_V est un ensemble de symboles d'étiquettes de sommets; $L'_V \subset L_V$ est l'ensemble des *symboles terminaux*; P est l'ensemble des *productions* et $S \in L_V \setminus L'_V$ est le *symbole non-terminal initial*, *l'axiome*. Les productions ont pour forme $X \to (D, C)$ où $X \in L_V \setminus L'_V$ est un symbole non-terminal et (D, C) est un graphe avec plongement qui est étiqueté sur $L_V \cup L_E$. Soient Γ et Γ' deux graphes étiquetés sur $L_V \cup L_E$, $p : X \to (D, C)$ une production et $v \in V_\Gamma$ un sommet étiqueté par X; on écrit $\Gamma \Rightarrow_{v,p} \Gamma'$ pour dire que Γ' est le graphe obtenu à partir de Γ en remplaçant v par D relativement à C. Comme il est d'usage en théorie des langages formels, on considère $\mathcal{L}(G)$, le langage de tous les graphes produits par G à partir de son symbole non-terminal initial S. Nous dirons qu'une grammaire edNCE est une grammmaire de remplacement de sommets (en abrégé grammaire VR) si elle est *confluente*, i.e.si $H \Rightarrow_{v_1 p_1} H_1 \Rightarrow_{v_2 p_2} H_{12}$ et $H \Rightarrow_{v_2 p_2} H_2 \Rightarrow_{v_1 p_1} H_{21}$ sont deux dérivations avec $v_1, v_2 \in V_H$ et $v_1 \neq v_2$, alors $H_{12} = H_{21}$ (cf. [ER97, Chap. 1, Prop. 1.3.6]). Par ailleurs, nous ne considérerons que des grammaires VR *déterministes*, i.e. des grammaires comprenant au plus une production par symbole non-terminal.

VR-Suites de graphes

Soit $G = (L_V, L'_V, L_E, P, S)$ une grammaire VR déterministe. On considère la suite $(\Gamma_n)_n$ de graphes définie par récurrence selon le schéma suivant :

- Γ_0 est le graphe de remplacement de la production de G associé à son symbole initial S.
- Pour tout $n \geq 0$, Γ_{n+1} est obtenu à partir de Γ_n en remplaçant selon G chaque sommet étiqueté par un non-terminal pour lequel une production est définie. Comme G est confluente, l'ordre dans lequel les dérivations sont faites n'importe pas.

Une telle suite de graphes est appelée une *VR-suite* de graphes.

Lemme 4.5.1
Soit G une grammaire VR déterministe; soit $(\Gamma_n)_n$ la VR-suite de graphes associée. Alors tous les graphes de $\mathcal{L}(G)$ sont connexes si et seulement si Γ_n est connexe pour tout n.

Idée de preuve. Le sens direct est trivial; pour voir la réciproque, notons que tout graphe de $\mathcal{L}(G)$ peut être obtenu à partir d'un Γ_n pour n assez grand en contractant quelques sous-graphes en sommets simples et éventuellement en ajoutant des arcs et/ou en modifiant des étiquettes d'arcs. Si Γ_n est connexe, il le reste après une telle transformation.
□

Une des principales propriétés de décidabilité des VR-langages de graphes est que leurs théories logiques monadiques du second ordre, avec quantifications sur les ensembles de sommets seulement, sont décidables (cf. [ER97, Chap. 1.4.2]). Comme la connexité est exprimable dans ce langage logique, il s'en suit le résultat suivant :

Proposition 4.5.3 *Il existe un algorithme pour décider si tous les graphes d'une VR-suite sont connexes ou non.*

Emulation

Proposition 4.5.4 *Quitte à modifier les étiquettes, toute VR-suite de graphes est une D0L-suite.*

Idée de preuve. Soit $(\Gamma_n)_n$ une VR-suite de graphe associée à une grammmaire VR déterministe $G = (L_V, L'_V, L_E, P, S)$. Nous allons construire un $D0L$-système de graphe δ qui engendre $(\Gamma_n)_n$.

Premièrement, l'ensemble des symboles d'étiquettes de sommets de δ est celui de G, i.e. L_V. Δ associe à tout symbole symbole terminal ℓ un graphe formé d'un seul sommet étiqueté par ℓ sans arc; et à tout symbole non-terminal, elle associe le graphe de remplacement défini par la production de G qui lui est associée, avec un ordre linéaire fixé de façon arbitraire sur l'ensemble des sommets.

L'ensemble des symboles d'étiquettes d'arcs de δ est défini comme étant $L_V \times L_E \times L_V$; pour chaque arc e de chaque Γ_n, on définit un nouveau symbole d'étiquette (ℓ_1, ℓ_e, ℓ_2) où ℓ_1 est l'étiquette de la source de e, ℓ_2 est l'étiquette de sa cible et ℓ_e est l'étiquette originale

de e; on modifie de la même manière les étiquettes de tous les arcs des graphes définis par Δ_V. Passons aux règles de substitution d'arc; soit (ℓ_1, ℓ_e, ℓ_2) une étiquette d'arc pour δ; on considère d'abord le graphe formé de deux sommets v_1 et v_2 respectivement étiquetés par ℓ_1 et ℓ_2 et d'un arc étiqueté par ℓ_e connectant v_1 à v_2. On considère alors le graphe obtenu en remplaçant v_1 et v_2 comme le dicte G. On supprime tous les arcs connectant des sommets issus du même sommet v_1 ou v_2. On obtient alors un graphe biparti; chacune des deux parties de sommet hérite d'un ordre linéaire donné par $\Delta_V(\ell_1)$ et $\Delta_V(\ell_2)$. Les étiquettes d'arcs sont modifiées comme précédemment et les étiquettes de sommets sont supprimées. Le graphe que l'on obtient est par définition $\Delta_E(\ell_1, \ell_e, \ell_2)$.

On omet ici la vérification du fait que δ engendre bien $(\Gamma_n)_n$, chose plus ennuyeuse que difficile.
□

4.5.4 Annexe au chapitre 4

Nous décrivons ici le détail du fonctionnement de $\widetilde{T}$. $\widetilde{T}$ a deux têtes de lecture/écriture que l'on note t_1 et t_2; l'alphabet de bande est $\Gamma_{\widetilde{T}} = \Gamma_T \cup \{\#, \$\} \cup Q_T \cup \{\square\}$ où Q_T est l'ensemble des états de T, et Γ_T est son alphabet moins le symbole de blanc; rappelons que $\widetilde{T}$ ne peut pas écrire le symbole de blanc. Pour une simplicité toute relative de la description qui suit, nous nous servons d'un langage simple. On définit d'abord quelques instructions élémentaires :

- `read`(t_i) désigne le symbole lu par t_i. La position de t_i ne change pas.
- `write`(t_i, u) où $u \in (\Gamma_{\widetilde{T}} \setminus \{\square\})^*$ signifie que t_i écrit u sur la bande et se déplace sur la cellule suivant directement le dernier symbole écrit.
- `move-left`(t_i) signifie que t_i se déplace d'une cellule sur la gauche. Si t_i est positionnée sur la première cellule, alors elle ne bouge pas.
- `move-right`(t_i) signifie que t_i se déplace d'une cellule sur la droite.
- BEGINNING$_i$ est un booléen qui est vrai si et seulement si t_i est positionnée sur la première cellule de la bande.

Nous utiliserons aussi quelques structures élémentaires de contrôle :

- `if` ... ... `else` ...
- `switch` ...
  - `case` ...
    ...
  - `case` ...
    ...
  - `default`
    ...
- `goto` ...

Les commentaires sont signalés par //.

L'algorithme suivi par $\tilde{T}$ est divisé en trois parties. Sa description utilise des variables dont les domaines sont de cardinaux finis; certaines d'entre elles par exemple varient dans Q_T. Nous nous servirons également de macros; les macros doivent simplement être remplacées toutes les fois qu'elles apparaissent par le code correspondant. On pourra vérifier que la description réelle de T, i.e. en termes d'états et de transitions, peut être obtenue à partir du code suivant en traduisant chaque ligne une à une. Chaqu'une d'entre elles donne naissance à plusieurs états relativement aux valeurs possibles des variables. Nous ne donnerons pas de méthode rigoureuse pour une telle construction qui est certes pénible mais à coup sûr possible.

Notons à cet égard qu'un compilateur pour la construction de machine de Turing est en cours de développement. Le but de ce travail consiste à mettre en œuvre un assistant de preuve relatif au machine de Turing. Ce projet pourrait par exemple conduire à réaliser des démonstrations rigoureuses pour les lemmes 4.5.2 et 4.5.3.

Algorithme principal:

```
Etape 1:
// T commence par ecrire la description instantanee initiale de T
        Initialisation();
Etape 2:
// T verifie que le contenu de la bande est de la forme #Id_0#Id_1#...#Id_k□□...
// ou Id_0 = q_0 et Id_i → Id_{i+1}. Il verifie aussi qu'il n'y a pas de repetition
// de description instantanee.
        Verify-History();
Etape 3:
// Ensuite, elle calcule la description instantanee qui suit directement la derniere qui est ecrite sur la bande.
// S'il elle n'apparait pas deja, alors elle l'ajoute en fin de bande.
        Next-ID();

    goto Etape 2;
```

Etape 1 - Initialisation:

T commence par écrire la description instantanée initiale de T.

```
Initialisation():
    write(t_1, #q_init);
    Back-to-the-Beginning(t_1);
    end;
```

Etape 2 - Vérification de l'historique :

Cette procédure vérifie que la bande contient la séquence des descriptions instantanées de T qui sont produites par un calcul fini de T à partir de q_{init}. Elle vérifie aussi qu'aucune description instantanée n'apparaît plus d'une fois sur la bande. Notons qu'après l'exécution de cette procédure, si elle ne conduit pas $\tilde{T}$ dans l'état `BUG`, t_1 se trouve au début de la dernière description instantanée et t_2 se trouve en début de bande.

```
Verify-History():
```

```
        Back-to-the-Beginning(t1);
        if (Deal-With(t1, #q_init, VERIFY-MODE) = FALSE)   // On verifie que le debut de la bande est
            BUG;                                           // de la forme #q_init#... or #q_init□...
        if (read(t1) ≠ □, #)
            BUG;
        Back-to-the-Beginning(t1);
        Back-to-the-Beginning(t2);
        move-right(t2); move-right(t2);
(i)     if (read(t2) = □)
            Back-to-the-Beginning(t2);
            end;
        Look-Forward();
        if (Simulation(VERIFY-MODE) = FALSE)
            BUG;
        goto (i);
```

Cette procédure vérifie que la description instantanée lu par t_1, qui est notée Id_1, n'apparaît pas à droite de la position de t_2.

```
Look-Forward():
        write(t2, $);                                      // t2 ecrit une marque pour garder sa position en memoire.
(i)     move-right(t1);
        switch
            • case (read(t1) = read(t2) and read(t1) ∈ Q_T ∪ Γ_T)
                    move-right(t2);
                    goto (i);
            • case (read(t1) ∈ {#, □, $} and read(t2) ∈ {#, □})   // t1 et t2 ont lu la meme
                    BUG;                                   // description instantanee
        move-left(t1); GotoPreviousID(t1);                 // t1 retourne au debut de Id1.
        GotoNextID(t2);                                    // t2 se deplace sur la description instantanee suivante.
        if (read(t2) = #)
            move-right(t2);
            goto (i);
(ii)    move-left(t2);                                     // t2 a atteint la fin de la bande.
        if (t2 ≠ $)                                        // Elle retourne alors a la marque, qu'elle remplace par #.
            goto (ii);
        write(t2, #);
        move-left(t2);
        end;
```

Etape 3 - **Description instantanée suivante :**

Le but de cette procédure est de calculer la description instantanée suivante et de l'ajouter en fin de bande si elle n'est pas déjà stockée. On suppose que la bande contient la séquence des descriptions instantanées de T séparées par des symboles séparateurs #. On suppose aussi que t_1 lit un séparateur # suivi par une description instantanée Id_1. Soit Id_2 le résultat d'un calcul de T sur Id_1; la procédure cherche Id_2 dans la liste des descriptions instantanées déjà stockées à droite de t_2; si elle ne la trouve pas, elle l'écrit alors en fin de bande.

```
Next-ID():
(i)     if (read(t2) = □)
            Simulation(WRITE-MODE);
            end;
        if (Simulation(VERIFY-MODE) = TRUE)
```

```
        end;
    move-left(t1);
    GotoPreviousID(t1);
    GotoNextID(t2);
    goto (i);
```

Macro principale: simulation du calcul de T.

Dans cette macro, T simule le calcul de T. Le paramètre state peut prendre comme valeur VERIFY-MODE ou bien WRITE-MODE.

On suppose que t_1 est positionnée sur un séparateur # qui est suivi d'une description instantanée de T notée Id_1. Si ça n'est pas le cas, la macro retourne FALSE.

Dans le VERIFY-MODE, on suppose que le symbole lu par t_2, qui doit être un séparateur, est suivi d'une description instantanée de T que l'on note Id_2. T vérifie que Id_2 est bien le résultat d'un calcul élémentaire de T sur Id_1. La macro retourne TRUE ou $Faux$ suivant le cas. Si le résultat est TRUE, t_1 (respectivement t_2) est positionnée sur la cellule suivant Id_1 (respectivement Id_2) qui doit contenir un séparateur ou bien un blanc.

Dans le mode WRITE-MODE, t_2 écrit sur la bande $\#\mathsf{Id}_2$.

```
Simulation(state):
    if (read(t1) ≠ #)
        return FALSE;
    move-right(t1);
    if (Deal-With(t2, #, state) = FALSE);
        return FALSE;

    c1 ← read(t1); move-right(t1);
    if (c1 ∉ Q_T ∪ Γ_T)
        return FALSE;
    c2 ← read(t1); move-right(t1);
    if (c2 ∉ Q_T ∪ Γ_T ∪ {#, □})
        return FALSE;

(i) c3 ← read(t1); move-right(t1);
    if (c3 ∉ Q_T ∪ Γ_T ∪ {#, □})
        return FALSE;
    switch
        • case c1 ∈ Q_T and c2 ∉ Q_T:
                if (c2 = #)
                        c2 ← □;
                (c, q, D) ← δ_T(c1, c2, 1);
                if (D = left, stay)
                        if (Deal-With(t2, qc, state) = FALSE)
                                return FALSE;
                else
                        if (Deal-With(t2, cq, state) = FALSE)
                                return FALSE;
                switch
                        • case c2 = □:
                                move-left(t1); move-left(t1);
                                if (read(t2) ≠ #, □)
                                        return FALSE;
                                return TRUE;
                        • case c2 ∈ Γ_T:
                                if (c3 ∈ {#, □})
```

```
                                move-left(t_1);
                                if (read(t_2) ≠ #, □)
                                    return FALSE;
                                return TRUE;
                            else
                                if (Deal-With(t_2, c_3, state) = FALSE)
                                    return FALSE;
                                goto (ii);
        • case c_1 ∈ Γ_T and c_2 ∈ Q_T and c_3 ∉ Q_T:
            if (c_3 = #)
                c_3 ← □;
            (c, q, D) ← δ_T(c_2, c_3, 0);
            switch
                • case D = left:
                    if (Deal-With(t_2, qc_1c, state) = FALSE)
                        return FALSE;
                • case D = stay:
                    if (Deal-With(t_2, c_1qc, state) = FALSE)
                        return FALSE;
                • case D = right:
                    if (Deal-With(t_2, c_1cq, state) = FALSE)
                        return FALSE;
            switch
                • case c_3 = □:
                    move-left(t_1);
                    if (read(t_2) ≠ #, □)
                        return FALSE;
                    return TRUE;
                • case c_3 ∈ Γ_T:
                    goto (ii);
        • case c_1, c_2 ∈ Γ_T:
            if (Deal-With(t_2, c_1, state) = FALSE)
                return FALSE;
            c_1 ← c_2; c_2 ← c_3;
            goto (i);
        • default:
            return FALSE;

(ii)    if (read(t_1) ∉ #, □)
            if (read(t_1) ∉ Γ_T)
                return FALSE;
            if (Deal-With(t_2, read(t_1)), state) = FALSE)
                return FALSE;
            move-right(t_1);
            goto (ii);
        if (read(t_2) ≠ #, □)
            return FALSE;
        return TRUE
```

Macros:

La macro suivante déplace t_i au début de la bande.

```
Back-to-the-Beginning(t_i):
(i)     if (not BEGINNING_i)
            move-left(t_i);
            goto (i);
        end;
```

Cette macro déplace t_i sur le premier séparateur ou blanc qui arrive à droite de sa position. Si elle est en train de lire un tel symbole, alors elle ne bouge pas.

```
GotoNextID(t_i):
(i)     if (read(t_i) ≠ #, □)
            move-right(t_i);
            goto (i);
        end;
```

Cette macro déplace t_i sur le premier séparateur ou blanc qui arrive à gauche de sa position. Si elle est en train de lire un tel symbole, alors elle ne bouge pas. Si elle n'en trouve pas avant d'arriver en bout de bande, elle conduit $\underset{\sim}{T}$ dans l'état BUG.

```
GotoPreviousID(t_i):
(i)     if (BEGINNING_i)
            BUG;
        if (read(t_i) ≠ #)
            move-left(t_i);
            goto (i);
        end;
```

Dans cette macro, t_i vérifie que le mot u est bien écrit sur la bande à partir de sa position ou bien écrit u à partir de sa position suivant la valeur de state qui peut être VERIFY-MODE ou bien WRITE-MODE. Après une execution normale, i.e. retournant TRUE, t_i est positionnée juste après u.

```
Deal-With(t_i, u, state):                    // Let u = u_0...u_n ∈ Γ*_T̃.
        switch
        • case (state = VERIFY-MODE)
            if (read(t_i) ≠ u_0)
                  return FALSE;
            move-right(t_i);
            ...
            if (read(t_i) ≠ u_n)
                  return FALSE;
            move-right(t_i);
        • case (state = WRITE-MODE)
            write(t_i, u);
        return TRUE;
```

Les démonstrations des lemmes suivants sont plus techniques que difficiles; elle peuvent être faites par un examen rigoureux de $\underset{\sim}{T}$.

Lemme 4.5.2 (Comportement de Simulation)
La procédure Simulation(VERIFY-MODE) *retourne* TRUE *si et seulement si, au début de son execution, t_1 (respectivement t_2) est positionnée sur un séparateur # suivi d'un description instantanée* Id_1 *(respectivement* Id_2*) qui se termine par un séparateur ou bien un*

blanc, et telle que $\mathsf{Id}_1 \underset{T}{\rightarrow} \mathsf{Id}_2$. *Après une execution retournant* TRUE, t_1 *(respectivement* t_2*) est positionnée juste à droite de* Id_1 *(respectivement* Id_2*).*

La procédure Simulation(WRITE-MODE) *retourne* TRUE *si et seulement si, au début de son execution,* t_1 *est positionnée sur un séparateur suivi d'une description instantanée* Id_1 *qui se termine par un séparateur ou bien un blanc. Dans ce cas,* t_2 *écrit* $\#\mathsf{Id}_2$ *où* Id_2 *est la description instantanée produite par* T *après un calcul élémentaire sur* Id_1.

Lemme 4.5.3 (Comportement de Verify-History)
Soit $\tilde{\mathsf{Id}}$ *une description instantanée de* $\tilde{T}$. *Alors*

- *après un calcul fini à partir de* $\tilde{\mathsf{Id}}$, $\tilde{T}$ *entre dans la procédure* Verify-History. *Soit* $\tilde{\mathsf{Id}}'$ *la description instantanée de* T *à cet instant.*
- *Si* $\tilde{T}$ *parvient à exécuter* Verify-History *sans encombre sur* $\tilde{\mathsf{Id}}'$, *i.e. sans aller dans l'état* BUG, *alors le contenu de la bande décrit par* $\tilde{\mathsf{Id}}'$ *est de la forme* $\#\mathsf{Id}_0\#\mathsf{Id}_1\#...\#\mathsf{Id}_k$ *où* $\mathsf{Id}_0 = q_{\mathsf{init}}$ *est la description instantanée initiale de* T; *pour tout* $i \in \{0, k-1\}$: $\mathsf{Id}_i \underset{T}{\rightarrow} \mathsf{Id}_{i+1}$ *et pour tout* $i \neq j$: $\mathsf{Id}_i \neq \mathsf{Id}_j$.