Mémoire de magistère

Thomas Mainguy

le 12 Octobre 2010

Table des matières

Première partie

Introduction au domaine de recherche

Introduction au domaine de recherche: Méthodes statistiques en linguistique computationnelle

Thomas Mainguy

7 octobre 2010

Résumé

Dans beaucoup de domaines, on peut être amené à étudier une chaîne de symboles $x = \dots x_0 x_1 \dots x_n \dots$, générée par une source aléatoire. Ces domaines peuvent aller de la prédiction météo, financière, à la compression de données. Dans tous les cas, on cherche à comprendre les propriétés de la source à partir des seules informations contenues dans la chaîne. On va ici étudier des modèles particuliers permettant de modéliser ces chaînes, utilisant des grammaires formelles telles que les grammaires hors contexte et les grammaires minimalistes, modèles particulièrement intéressants en linguistique par exemple. Le but sera de construire une méthode permettant de traiter par exemple une suite de phrases, et d'apprendre la grammaire générant ces phrases.

Table des matières

1 Modélisation statistique de chaînes à espace fini

1.1 Les langues humaines comme sources aléatoires de chaînes de symboles

Les phrases prononcées, écrites, ... par les locuteurs d'une langue peuvent être vues comme une simple succession de symboles (des mots). On peut donc assimiler la grammaire d'une langue en une source de telles suites, i.e. un processus aléatoire générant des suites, ici finies, de symboles. La modélisation des langues comme objet probabiliste, des grammaires comme générateurs aléatoires de phrases, permet d'appliquer les outils classiques de la statistique à l'étude des langues humaines. La compréhension de la grammaire d'une langue peut donc être vue comme la compréhension du processus aléatoire générant les phrases de cette langue, de la loi des phrases produites.

On modélisera donc une langue (assimilée à sa grammaire), comme une source de loi $\mathbb{P}$, générant des chaînes $x = x_0x_1 \dots x_n \dots$ de symboles dans un alphabet $\mathcal{X}$. On cherchera donc, pour comprendre une langue, à comprendre la loi $\mathbb{P}$.

1.2 Enjeux

Considérons donc une telle source de loi $\mathbb{P}$, générant une chaîne $x = x_0x_1 \dots x_n \dots$ de symboles dans un alphabet $\mathcal{X}$, que l'on observe.

Deux grands points de vue peuvent être considérés : l'estimation et la compression.

Le but de l'estimation est de comprendre le mieux possible le fonctionnement de la source, pour pouvoir prédire par exemple la suite de la chaîne. Il s'agit d'estimer au mieux la loi de la source, $\mathbb{P}$, par $\hat{\mathbb{P}}$, fonction de l'observation x. On cherche à minimiser un critère mesurant la distance de $\hat{\mathbb{P}}$ à $\mathbb{P}$, distance pouvant varier en fonction du problème considéré : distance L_p, variation totale, etc.

La compression, quant à elle, cherche à optimiser une perte cumulée : on cherche à obtenir une représentation concise de la chaîne, ce qui est différent de chercher à décrire la simple fréquence des phrases. Dans le cas des codes Elias, on va quand même chercher à estimer $\mathbb{P}$ par $\hat{\mathbb{P}}$, mais le critère à minimiser sera différent : $\mathbb{E}\sum_i \log(\hat{\mathbb{P}}(X_i|X_0 \dots X_{i-1}))$, X étant une chaîne générée par la source. Ce critère correspond à la longueur du code codant X.

Dans les deux cas, le problème du modèle et de la paramétrisation est crucial. On doit en effet faire un compromis entre la précision du modèle (qui évidemment augmente avec le nombre de paramètres) et l'estimation de ces paramètres (qui est meilleure si on en a peu). Pour fixer les idées, imaginons que l'on observe une fonction polynomiale $p = \sum^d a_i X^i$ en les entiers, bruitée : $f(i) = p(i) + \varepsilon_i$, ε_i étant un terme aléatoire. On veut estimer p par un polynome $\hat{p} = \sum^{d'} \hat{a}_i X^i$. Le nombre de paramètres est le degré limite que l'on s'autorise $(+1)$, $d'+1$ (on a $d'+1$ éléments à estimer, les a_i). Si l'on ne connaît pas à l'avance le véritable d, le choix de d' est crucial : trop faible, et notre estimation sera trop grossière ($d = 0$ estimera juste la moyenne de la fonction), et trop fort, le bruit ne sera pas lissé ($d = n - 1$, si on a observé n points, interpolera tous les points mesurés, comme s'il n'y avait pas de bruit).

Notons que dans le cas de l'étude des langues humaines via un corpus (i.e. une collection de phrases), on a en réalité deux sources imbriquées : la source générant la suite de phrase, où les symboles sont donc des phrases entières, et la source générant chaque phrase, vue comme suite de mots.

1.3 Modèles i.i.d., Markoviens

Le modèle le plus simple pour modéliser ces chaînes est le modèle i.i.d. : tous les symboles rencontrés sont indépendants et identiquement distribués selon une loi p :

$$\mathbb{P}(x_0 \dots x_n) = \prod p(x_i)$$

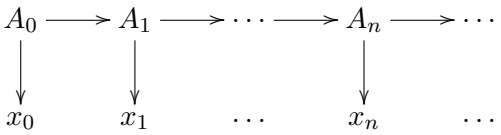
Ici, le nombre de paramètres est très réduit : il est égal au cardinal de l'alphabet moins 1.

Il est possible de raffiner un peu ce modèle en permettant la source d'avoir un peu de mémoire, et de conditionner la loi d'un symbole par rapport à son contexte. Le modèle d'une chaîne de Markov conditionne par rapport au symbole précédent :

$$\mathbb{P}(x_0 \dots x_n) = \prod \mathbb{P}(x_i | x_{i-1})$$

Ici, on a autant de lois que de symboles dans l'alphabet, donc $|\mathcal{X}|(|\mathcal{X}| - 1)$ paramètres.

Il est possible de raffiner encore ces modèles, par exemple avec des modèles de n-grams, où le conditionnement se fait sur les n derniers symboles, ou encore avec des modèles de Markov cachés, où l'on considère que la véritable chaîne de Markov n'est observée uniquement qu'au travers des observations x_i :



Ici, (A_i) est une chaîne de Markov, mais on n'observe que (x_i).

La probabilité est alors :

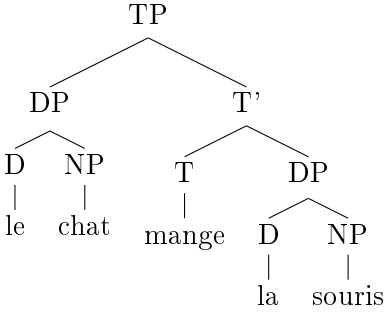
$$\mathbb{P}(x_0 \dots x_n) = \prod \mathbb{P}(A_{i+1} | A_i) p(x_i | A_i)$$

1.4 Un peu de linguistique...

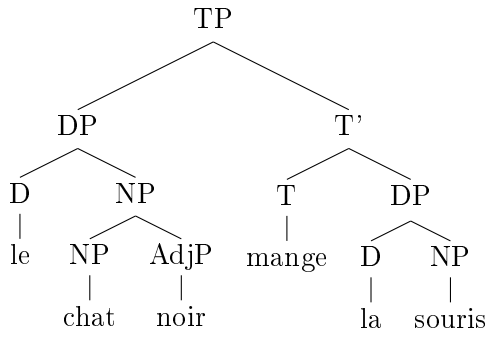
Une particularité des modèles markoviens réside dans le fait qu'ils ont une structure linéaire : chaque symbole dépend de ses prédécesseurs directs, dans l'ordre. Dans le cas de la prédiction, cette hypothèse paraît tout à fait satisfaisante : le passé proche influence sans doute plus le futur que le passé lointain. En revanche, pour d'autres problèmes, on peut avoir envie d'utiliser une structure plus souple. Pour prendre un exemple, en linguistique, on peut avoir des mots s'insérant dans la phrase, qui n'influent sans doute pas sur la suite. Considérons par exemple : 'le chat mange la souris' et 'le chat noir mange la souris'. Dans un modèle markovien, le verbe 'mange' est dans un contexte complètement différent dans les phrases 1 et 2, puisqu'il suit dans un cas 'chat' et dans l'autre 'noir'. Pourtant on a bien envie de dire qu'il n'y a pas de réelle différence, puisque c'est dans les deux cas un verbe dont le sujet est un chat.

Les linguistes utilisent, pour décrire les phrases, des arbres binaires. Ainsi les arbres (un peu simplifiés) modélisant ces deux phrases sont :

(1) a. 'Le chat mange la souris.'



b. ‘Le chat noir mange la souris.’



On voit déjà ici, que le verbe ‘mange’ est dans les deux cas dans des positions similaires : il faudrait monter et descendre de 3 niveaux dans l’arbre depuis ‘mange’ pour voir la première différence.

Quelles propriétés des langues humaines ont conduit à ces représentations ?

- le principe de *constituance* : les mots d’une phrase se regroupent entre eux pour former des *constituants*, de certaines catégories. Des constituants de même catégorie sont syntaxiquement interchangeables (i.e. remplacer l’un par l’autre ne change pas la grammaticalité de la phrase, même si la sémantique peut poser problème : ‘le chat mange la souris’ et ‘le chien de ma grand-mère mange la souris’ indique que ‘le chat’ et ‘le chien de ma grand mère’ sont deux constituants de même type). C’est ce principe qui permet de représenter une phrase par une arbre. Par exemple, dans une phrase du type ‘Le chat noir mange la souris.’, ‘chat’, ‘mange la souris’, ‘le chat noir’, sont des constituants (entre autres), mais pas ‘noir mange la’.
- l’hypothèse de *binarité* : on suppose que les constituants se regroupent par deux. Dans la phrase précédente, ‘le chat noir’ se compose de deux constituants, ‘le’ et ‘chat noir’. Ceci permet d’avoir une hypothèse de structure sur les arbres que l’on construit. Ceci reste une hypothèse, mais jusqu’ici, les linguistes sont toujours arrivés à former des grammaires la vérifiant.
- le *mouvement* : il y a de fortes raisons de penser que, outre se regrouper par deux, les constituants peuvent se déplacer dans la phrase, si certaines conditions sont remplies. On peut considérer par exemple la paire ‘Le chat a mangé la souris hier’ et ‘[Quelle souris] le chat a mangée _ hier’. Les grammaires construites devront permettre ce type de mouvement de constituants, et seulement ceux-ci. On ne veut pas permettre, par exemple, ‘Mangé quelle le chat a souris hier’ (‘mangé quelle’ n’est pas un constituant, mais ‘quelle souris’ l’est) !
- le principe de *tête* : les dépendances au sein d’un arbre donné semblent restreintes. Si l’on considère les exemples suivants (une $*$ indique que la phrase est agrammaticale) :

(2) a. ‘Je crois que [Marie vient]/*[vient Marie].’
b. *‘Je crois si [Marie vient]/[vient Marie].’
c. *‘Je crois quand [Marie vient]/[vient Marie].’
d. ‘Je demande si [Marie vient]/*[vient Marie].’
e. *‘Je demande que [Marie vient]/[vient Marie].’
f. ‘Je demande quand [Marie vient]/[vient Marie].’

Il ressort de ces phrases que le verbe principal ne pose de contraintes que sur la conjonction de la relative, et pas sur reste. L’explication retenue repose sur le principe de *tête*, disant que chaque constituant a une unique tête, et que l’extérieur de ce constituant n’a accès qu’à celle-ci. Ici, la tête de la relative serait donc la conjonction, ‘que’, ‘si’, ‘quand’,...

1.5 Grammaires hors contexte

Un type de grammaire utilisé en linguistique est constitué des grammaires hors contexte (CFG, Context Free Grammars). Ces grammaires génèrent les arbres de haut en bas, partant d'un symbole racine et réécrivant les symboles des feuilles : Par exemple, si le symbole racine est S (Sentence) et les règles de réécriture s'indiquent par des $\longrightarrow$:

(3)
a. S$\longrightarrow$DP VP
b. DP$\longrightarrow$D NP
c. D$\longrightarrow$'le'
d. D$\longrightarrow$'la'
e. NP$\longrightarrow$NP AdjP
f. NP$\longrightarrow$'chat'
g. NP$\longrightarrow$'souris'
h. AdjP$\longrightarrow$'noir'
i. VP$\longrightarrow$V DP
j. V$\longrightarrow$'mange'

Cette grammaire génère (entre autre) les deux arbres en (1) (elle génère aussi des phrases agrammaticales comme 'Le souris noir noir noir mange la chat', mais ceci est aisément corrigé).

Il est aisé de placer des probabilités sur une telle grammaire, en utilisant un simple processus de branchement : chaque règle de réécriture a une certaine probabilité conditionellement au symbole réécrit. La probabilité de l'arbre de (1-a) sera

$$\mathbb{P}((3\text{-a}))\mathbb{P}((3\text{-b}))\mathbb{P}((3\text{-c}))\mathbb{P}((3\text{-f}))\mathbb{P}((3\text{-i}))\mathbb{P}((3\text{-j}))\mathbb{P}((3\text{-b}))\mathbb{P}((3\text{-c}))\mathbb{P}((3\text{-g}))$$

1.6 Grammaires minimalistes

Un autre type de grammaire, plus complexe, est l'ensemble les grammaires minimalistes (MG). Elles ont été proposées par E. Stabler [Sta97] dans le cadre du programme minimaliste de Chomsky [Cho95].

Ces grammaires génèrent les arbres de bas en haut, en agglomérant les constituants ensemble. Une grammaire n'est alors plus décrite par une série de règles, mais par une liste de mots, accompagnés de catégories, déterminant les constituants avec lesquels ils peuvent se regrouper. Une explication du fonctionnement de ces grammaires serait trop longue pour figurer ici, mais peut être trouvé dans l'article original de E. Stabler [Sta97] ou dans mon article de M2 joint.

Les grammaires minimalistes ont plusieurs avantages :

- Tout d'abord, elles présentent un changement de paramétrage. Alors que les CFG réécrivent sans se soucier du contexte, les grammaires minimalistes utilisent la notion de tête : les catégories d'un seul constituant (la tête) sont conservées plus haut dans l'arbre lors de la fusion de deux constituants. La nature de la tête est donc gardée tout au long de l'arbre, jusqu'à ce qu'on change de tête.
- De plus, elles permettent de modéliser le mouvement, contrairement aux CFG. Le mouvement est ici restreint, comme on le souhaite, à des montées de constituants (seuls des constituants peuvent se déplacer, et uniquement en montant dans la structure de l'arbre).

2 Projet

Le but est de développer un algorithme utilisant les structures arborescentes présentées ci-dessus (CFG, MG sans et avec mouvement) pour la modélisation statistique de chaînes aléatoires. La plupart des modèles jusque-là considérés utilisaient des modèles markoviens, car les chaînes

étaient supposées ergodiques stationnaires (i.e. dont la loi ne change pas en fonction de la position du mot considéré). Dans le cas de textes linguistiques, cette hypothèse paraît peu pertinente, puisque la position d'un mot semble influer grandement sur la nature de celui-ci. Puisque la principale motivation du projet reste la modélisation des langues humaines, on prendra dans tout ce qui suit des phrases du français en exemple, même si tout ceci devrait pouvoir s'appliquer à n'importe quel type de chaîne (dont on suppose la structure similaire à celle des grammaires considérée).

Plusieurs problèmes apparaissent dans ce projet.

2.1 Analyse syntaxique

Le premier problème est celui de l'analyse syntaxique : retrouver la structure d'une phrase. Dans les phrases sur lesquelles on veut travailler, on n'a accès qu'à l'ordre des mots ; pour les modèles markoviens, la structure sous-jacente se réduisait à cet ordre, mais on veut ici travailler sur des structures arborescentes plus complexes, qui ne sont pas directement accessibles. On a donc besoin, connaissant la grammaire et une nouvelle phrase, d'un algorithme capable de reconstruire la structure syntaxique de celle-ci. Un algorithme effectuant ce travail est appelé un analyseur syntaxique, ou parseur.

Il est aussi intéressant de considérer des parseurs reconstruisant la structure 'online', i.e. au fur et à mesure que la phrase est lue, dans l'ordre. Il existe deux grands types de parseurs ayant cette propriété : les parseurs top-down et les parseurs bottom-up.

2.1.1 Parseurs bottom-up

Ces analyseurs, les plus simples à implémenter, reconstruisent l'arbre de bas en haut. Il parcourent la phrase de gauche à droite, et agglomèrent les constituants consécutifs compatibles jusqu'à obtenir un arbre complet. Si l'on essaye de parseur la phrase 'Le chat noir mange la souris' (1-b), on effectuerait les étapes suivantes (les constituants construits sont entre crochets) :

1. Le parseur lit le mot 'le'. Ce mot forme à lui seul un constituant de type D, qui est donc mémorisé par le parseur.
$$[D]$$
2. Le parseur lit alors le mot 'chat'. Ce mot forme un constituant de type NP. Les deux constituants D et NP peuvent être groupés ensemble, mais pas obligatoirement (on peut avoir un adjectif). Le parseur forme donc deux hypothèses, l'une où il regroupe D et NP en DP, l'autre où il les laisse 'en vrac'.
$$([DP]; [D, NP])$$
3. Le parseur lit 'noir'. C'est un AdjP. Dans le première hypothèse, il ne peut pas se combiner avec le DP. Il est donc rajouté à la liste de constituants formés. Dans la seconde hypothèse, il peut se combiner avec le NP pour former un autre NP, qui peut se combiner avec le D pour former un DP (mais pas obligatoirement, on pourrait avoir un 'chat noir famélique').
$$([DP, AdjP]; [DP]; [D, NP])$$
4. Le parseur lit 'mange'. ce constituant de type T pourrait se combiner avec un DP, mais les mots sont dans le mauvais sens (il faut T puis DP). Donc le parseur rajoute ce constituant à ses hypothèses.
$$([DP, AdjP, T]; [DP, T]; [D, NP, T])$$

5. Le parseur lit 'la'. Type D, ne peut se combiner.

$$([DP, AdjP, T, D]; [DP, T, D]; [D, NP, T, D])$$

6. Le parseur lit 'souris'. Type NP, peut se combiner avec D pour former DP, qui se combine avec T pout former T', qui se combine avec DP *dans la seconde hypothèse* pour former TP.

$$([DP, AdjP, T']; [TP]; [D, NP, T'])$$

7. Fin de la phrase, on a une hypothèse terminée (la seconde, [TP]). L'analyse est finie, et on a la bonne structure.

On remarque dans cet exemple la difficultée posée par l'ambiguïté des langues : le fait que les adjectifs sont optionnels, par exemple, force le parseur à considérer plusieurs hypothèses. Enfin, jusqu'à la dernière étape, on a à chaque fois un paquet de constituants refusant de communiquer entre eux, et donc inutilisables.

2.1.2 Parseurs top-down

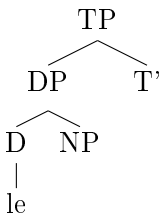
Ces analyseurs construisent les arbres de haut en bas. Bien plus complexes que les bottom-up, ils ont cependant des propriétés extrêmement intéressantes.

L'idée de ces parseurs est de commencer avec un arbre complet, mais 'vide' : juste la racine de l'arbre. Au fur et à mesure que l'on lit les mots de la phrase à analyser, on essaye de les 'greffer' à l'arbre, en respectant leur ordre. Petit exemple avec toujours la même phrase, 'Le chat noir mange la souris' (1-b) :

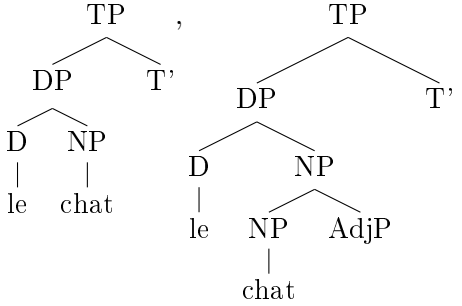
1. Le parseur commence avec autant d'hypothèses qu'il a de catégories racines. Dans la grammaire en (1), on en a une seule : TP.

TP

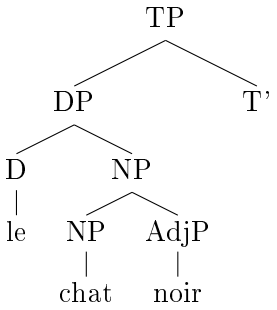
2. Le premier mot, 'le', est lu. Étant en tête de phrase, on a une seule possibilité : déterminant du sujet de TP. On construit donc l'arbre jusqu'au D sujet pour l'hypothèse TP :



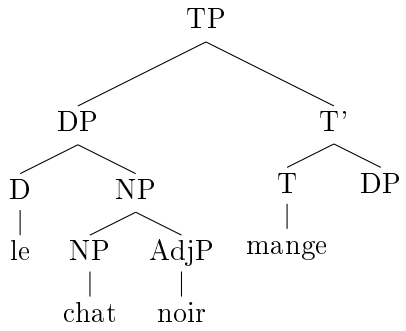
3. Le second mot, 'chat', est lu. Étant second mot de la phrase, il est sans doute le nom commun correspondant à 'le'. Cependant, il est peut-être modifié par un ou plusieurs adjectifs. On va donc créer deux hypothèses :



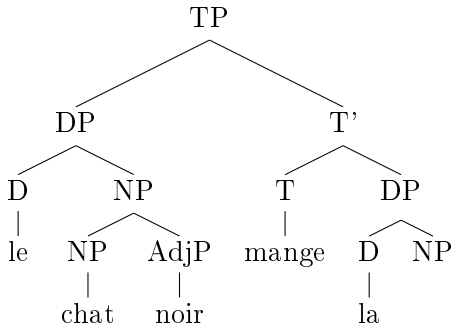
4. Troisième mot, ‘noir’. Il peut se placer à sa place réservée dans la seconde hypothèse, mais pas dans la première. Il serait peut-être possible de trouver une autre place, mais ici, nous allons simplement supprimer cette hypothèse (qui est la mauvaise).



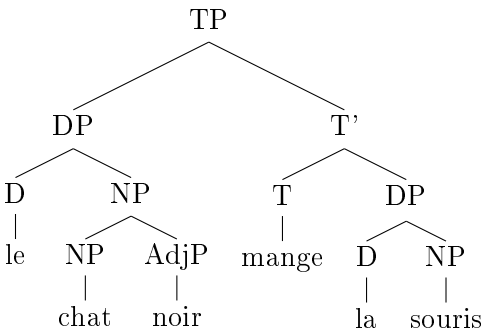
5. ‘mange’ a sa place en-dessous de T’ (moyennant l’ajout d’un T et DP) :



6. on lit ‘la’, qui se place sous DP :



7. Enfin, le dernier mot, ‘souris’ se place sous le NP :

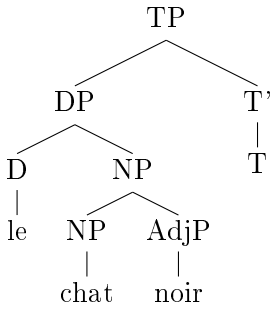


8. La phrase est finie, on a un arbre complet. L’analyse est réussie.

Il est intéressant de remarquer qu'ici, contrairement au cas bottom-up, les hypothèses sont systématiquement constituées d'un unique arbre (pas forcément complet). Par exemple, à l'étape suivant la lecture de 'mange', le parseur bottom-up avait, pour sa meilleure hypothèse, un groupe nominal ('le chat noir') et un verbe ('mange'), sans savoir que c'était le chat qui mangeait. Alors que le parseur top-down, lui, a une unique hypothèse (contre trois), disant déjà que l'on a un chat noir mangeant quelque chose. On a ici l'illustration d'un autre avantage : le parseur prédit même à ce point que l'on s'attend à entendre un groupe nominal, et même un déterminant s'il continue de développer les nœuds de son arbre, ce qu'il fait en pratique.

Il est cependant bon de remarquer que, par soucis de simplicité, j'ai occulté beaucoup de choses dans cette description, dans lesquelles résident tout les problèmes de l'implémentation de ces parseurs.

Tout d'abord, comment 'accrocher' les nouveaux mots. Ici, la grammaire est très simple, et on voit assez bien la chose. En revanche, pour des grammaires plus complexes, la tâche est plus ardue. De plus, il faut réussir à programmer la machine de manière à être sûr que toutes les possibilités soient explorées... La solution est de réellement 'descendre' dans l'arbre en réécrivant les nœuds menant à la position dans laquelle on s'attend à avoir le prochain mot lu, *avant même de l'avoir vu*. Par exemple, à la fin de l'étape 5, l'hypothèse n'est pas celle présentée ci-dessus, mais toutes les hypothèses où l'on a réécrit les nœuds sur le chemin gauche dessous T', jusqu'à ce que l'on attende un mot. En particulier, on aura

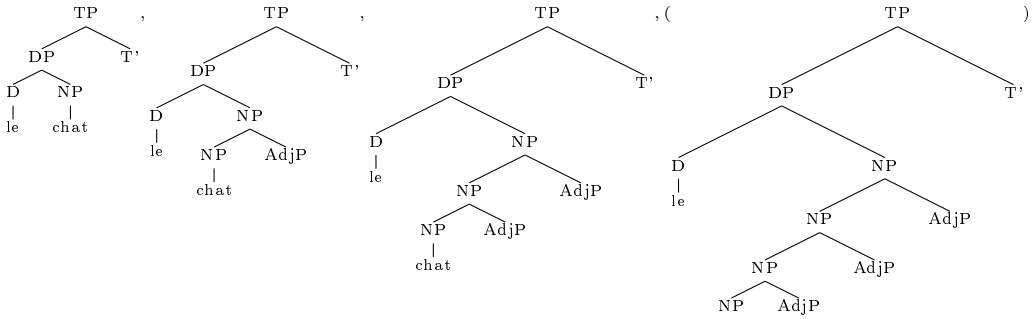


Ceci augmente le pouvoir prédictif de l'analyseur, mais complexifie énormément l'algorithme, puisque qu'un nombre très important d'hypothèses peuvent être considérées, particulièrement au début, où les débuts de toutes les phrases possibles doivent être considérées (plus d'une par catégorie de premier mot possible : conjonction, déterminant, adverbe, nom propre, verbe, etc.).

Ceci nous amène au plus gros problème, grossièrement écarté lors de l'exemple, en 3. On a en effet considéré deux hypothèses à ce point, avec 0 ou 1 adjectif. Il aurait en fait fallu considérer un nombre infini d'hypothèses, une pour chaque nombre d'adjectifs suivant le nom ('chat', 'chat noir', 'chat noir famélique', 'chat noir famélique affamé', etc.). Les adjectifs étant fixés plus bas que le lien entre NP et D (ce sont des adjoints à NP, pour citer le terme linguistique), il faut laisser de la place pour autant d'adjectifs que l'on peut rencontrer, c'est-à-dire... une infinité. Ce type d'analyseur rencontre obligatoirement ce problème pour n'importe quelle langue récursive à gauche (i.e. qui peut avoir des branches gauches aussi longues que l'on veut). Hélas, un grand nombre de langues humaines sont dans ce cas.

La solution de ce problème réside, par exemple, dans l'introduction d'une hiérarchie dans les hypothèses, par exemple au moyen de probabilités. Si l'on est capable de donner une probabilité à ces arbres incomplets, on peut stipuler que l'on ne s'occupe que des hypothèses les plus probables, et laisser les autres pour après, si les hypothèses plus probables se révèlent fausses. Dans notre exemple, il est très peu probable d'avoir, mettons, plus de deux adjectifs à droite, et donc on va laisser de côté ces hypothèses, sous la forme par exemple d'un arbre dont les nœuds ne sont pas encore réécrits après un certain niveau.

Les hypothèses seront donc :



la dernière hypothèse gardant en mémoire que 'chat' n'est pas encore analysé, mais qu'elle est peu probable et qu'il n'est pas nécessaire d'y revenir pour l'instant. Si les hypothèses précédentes échouent, alors, oui.

Bien évidemment, ceci suppose de pouvoir placer des probabilités sur des arbres incomplets. Mais c'est un autre problème.

2.1.3 Solutions du problème

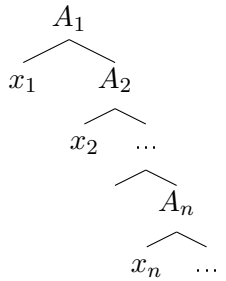
Le problème du parsing des grammaires a été bien étudié par les linguistes. Dans le cas des grammaires hors contexte, un parseur top-down est assez simple à implémenter, du fait de la structure de ces grammaires. On a par exemple les travaux de B. Roark [Roa97], décrivant un parseur top-down pour les grammaires hors contexte, utilisant des probabilités pour les arbres incomplets.

Dans le cas des grammaires minimalistes, il est très facile de construire un parseur bottom-up (les phrases sont construites de cette manière), mais bien plus délicat de construire un parseur top-down. Cette problématique a été le sujet de mon stage de M2, qui a conduit à la réalisation d'un parseur top-down, probabiliste, pour grammaires minimalistes, et à une réécriture de ces grammaires permettant une lecture top-down et la construction de probabilités pour les arbres incomplets. Voir l'article joint pour plus de détails.

2.2 La construction de la grammaire

Le second problème est la construction de la grammaire. La structure des phrases dans les modèles markoviens étant déjà donnée, seul restait le problème d'estimation des paramètres. Ici, il faut construire la structure même des phrases. La suite de ce papier va donner quelques pistes pour résoudre ce problème, qui constitue mon projet de thèse.

On peut ici faire l'analogie avec les modèles de Markov cachés, qui peuvent aussi être vus comme construisant un arbre, même si sa structure est connue à l'avance, hormis les étiquettes des nœuds internes :



La difficulté dans ce modèle est d'estimer correctement les variables cachées. Dans notre cas, les variables cachées, au lieu de se réécrire en un élément du lexique et une autre variable cachée,

se réécrivent comme un couple d'éléments du lexique ou de l'ensemble des variables cachées.

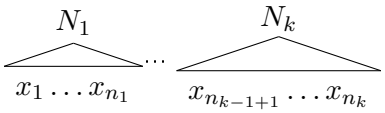
Définition 2.1. *On appellera ici* variable cachée *une loi sur* $(\mathcal{X} \cup \mathcal{A})^2$ *de la forme* $p_1 \otimes p_2$*, où* $\mathcal{X}$ *est le lexique de la grammaire et* $\mathcal{A}$ *l'ensemble des variables cachées.*

Par commodité, on notera une telle loi $(\sum \delta_u) \otimes (\sum \delta_{u'})$, où δ_u représente la masse de Dirac en l'élément u. Cette notation n'est pas normalisée, mais a l'avantage de garder en mémoire le nombre d'occurences observées, important pour l'estimation.

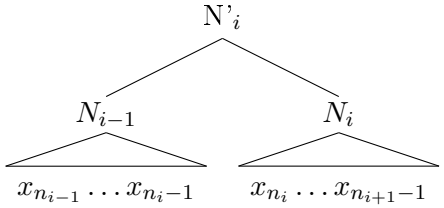
Les grammaires qui nous intéressent ne peuvent pas parser n'importe quelle phrase, contrairement aux modèles markoviens. Il va donc falloir travailler sur des grammaires partielles, qui devront donner finalement des grammaires complètes.

On va donc commencer par considérer une grammaire très simple, un simple modèle i.i.d.. Une phrase $x = x_1 \dots x_n$ aura donc pour probabilité $\mathbb{P}(x) = \left(\frac{1}{|\mathcal{X}|}\right)^n$. Cette grammaire va servir de base pour la construction de notre grammaire.

Une nouvelle phrase va ainsi être considérée comme générée par la grammaire i.i.d., pour être ensuite réécrite en agglomérant des couples de mots contigüs ensemble. On travaillera donc sur des suites d'arbres binaires :

(4)


où $x_1 \dots x_{n_k}$ est la séquence de mots de la phrase. On vise à obtenir finalement des arbres binaires ($k = 1$), en regroupant deux arbres binaires N_{i-1} et N_i en



La probabilité d'une telle phrase est obtenue comme la probabilité d'avoir une source i.i.d. générant la suite (N_i) (sur un alphabet étendu pour comprendre les étiquettes N_i), multiplié par la probabilité de chaque arbre, donné par un simple processus de branchement :

(5)
$$\mathbb{P}\left(\begin{array}{c} N \\ \diagup\ \diagdown \\ N_1 \quad N_2 \end{array}\right) = \mathbb{P}(N_1|N, g)\mathbb{P}(N_2|N, d)$$

On conditionne pour l'instant sur tous les parents et, dans le cas des arbres binaires, si on considère le fils gauche ou droit. On pourra modifier ceci, pour prendre en compte la notion de tête par exemple, en ne gardant le conditionnement que pour un des fils (la tête, donc).

Le nœud du problème est de trouver la bonne façon de créer ces nœuds internes, c'est-à-dire de construire les variables cachées étiquettant les probabilités de transition dans l'arbre syntaxique.

On peut par exemple estimer ces variables cachées de façon récursive : étant donné une nouvelle phrase, que l'on aura analysée selon la grammaire courante sous la forme $N_1 \dots N_k$, on va considérer toutes les nouvelles variables cachées obtenues en groupant deux N_i, N_{i+1}. On choisit ensuite la plus vraisemblable en utilisant un critère de compression ou de MV, que l'on rajoutera à la grammaire. Les nouvelles variables cachées pourront être obtenues soit en en créant une nouvelle, soit en modifiant une déjà existante.

Pour résumer, la construction pourrait se faire de la sorte :

1. À la première étape, on commence avec la grammaire $\mathcal{G}_0$ i.i.d., et l'on analyse la première phrase du corpus : $x^1 = x_1^1 \dots x_{n_1}^1$ de probabilité selon $\mathcal{G}_0$ valant $(\frac{1}{|\mathcal{X}|})^n$. On considère alors toutes les nouvelles variables cachées

$$A = (\delta_{x_i^1}) \otimes (\delta_{x_{i+1}^1}).$$

   On garde celle qui maximise par exemple le critère de vraisemblance

$$\mathbb{P}_{\mathcal{G}_1^i}(x) = (\frac{1}{|\mathcal{X}| + k_i})^{n-2k_i} (\frac{k_i}{|\mathcal{X}| + k_i})^{k_i}$$

   où k_i est le nombre de couples (x_i^1, x_{i+1}^1) rencontrés dans x^1. On a ici noté $\mathcal{G}_1^i$ la grammaire générant de façon i.i.d. la nouvelle variable cachée et les éléments du lexique, la probabilité de la variable cachée étant $\frac{k_i}{|\mathcal{X}|+k_i}$ (On l'a vue k_i fois) et les éléments du lexique étant choisis uniformément. On pourrait aussi estimer les probabilités des éléments du lexique en comptant leur nombre d'occurence.

   Remarquons qu'il n'y a pas nécessairement une unique variable maximisant cette quantité, dans ce cas, elles sont toutes équivalentes, on peut soit toutes les garder, ou en choisir une aléatoirement. La grammaire obtenue est notée $\mathcal{G}_1$.

2. À chaque étape, on analyse la nouvelle phrase $x^k = x_1^k \dots x_{n_k}^k$ au moyen de la grammaire $\mathcal{G}_k$, ce qui donne $x^k = A_1^k \dots A_{n'_k}^k$, où les A_i^k sont des variables cachées (dominant un arbre binaire), ou des éléments du lexique. À ce moment, on met à jour nos estimées (on a vu de nouvelles instances des A_i^k, de nouveaux mots, etc...). On considère alors toutes les nouvelles variables cachées : pour tous les couples contigüs $A_i^k A_{i+1}^k$, on a une nouvelle variable

$$A = (\delta_{A_i^k}) \otimes (\delta_{A_{i+1}^k}),$$

   et une modification des variables déjà existantes

$$(\sum \delta_u) \otimes (\sum \delta_{u'}) \longrightarrow (\sum \delta_u + \delta_{A_i^k}) \otimes (\sum \delta_{u'} + \delta_{A_{i+1}^k}).$$

   Comme à la première étape, on maximise un critère pour choisir la meilleure variable, ce qui donne la nouvelle grammaire $\mathcal{G}_{k+1}$.

3 Conclusion

Le problème de la construction des grammaires semble être la prochaine étape dans l'utilisation des grammaires linguistiques pour l'étude des chaînes aléatoires. On va donc commencer par étudier le problème dans le cas des grammaires hors contextes, pour tenter ensuite de passer aux grammaires minimalistes, sans, puis avec, mouvement.

Notre approche cherche à construire des grammaires qui approcheront le mieux possible la distribution observée. Dans le cas de la linguisitique, les variables de ces grammaires seront peut-être très différentes de celles des grammaires construites par les linguistes, qui ne tentent pas par exemple de faire entrer la sémantique dans la syntaxe. Pour un syntacticien, 'la souris mange un chat' est tout à fait correcte, même si c'est une phrase que l'on ne risque pas d'entendre très souvent. Il sera cependant intéressant de comparer les grammaires obtenues et les grammaires linguistiques. Si elles sont très semblables, ce sera un signe que peut-être la syntaxe contrôle tout de même en grande partie la sémantique.

Références

[Cho95] Noam Chomsky. The minimalist program. *MIT Press, Cambridge, Massachusetts*, 1995.

[Roa97] Brian Roark. Probabilistic top-down parsing and language modeling. *Logical aspects of computational linguistics*, 1997.

[Sta97] Edward Stabler. Derivational minimalism. *Logical aspects of computational linguistics*, 1997.

Deuxième partie

Mémoire de M2

A probabilistic top-down parser for minimalist grammars

T. Mainguy

this paper was written during an internship
supervised by E. Stabler (UCLA) and O. Catoni (ENS)

September 21, 2010

Abstract

This paper describes a probabilistic top-down parser for minimalist grammars. Top-down parsers have the great advantage of having a certain predictive power during the parsing, which takes place in a left-to-right reading of the sentence. Such parsers have already been well-implemented and studied in the case of Context-Free Grammars (see for example [Roa97]), which are already top-down, but these are difficult to adapt to Minimalist Grammars, which generate sentences bottom-up. I propose here a way of rewriting Minimalist Grammars as Linear Context-Free Rewriting Systems, allowing us to easily create a top-down parser. This rewriting allows also to put a probabilistic field on these grammars, which can be used to accelerate the parser. I propose also a method of refining the probabilistic field by using algorithms used in data compression.

Contents

Throughout this paper, I will refer as a *subtree* of a tree $\mathcal{T}$, the set of nodes in $\mathcal{T}$ dominated by a particular node, which will be the root of the subtree. On the other hand, a cut is the set of the leaves of a finite prefix tree of $\mathcal{T}$.

1 Introduction

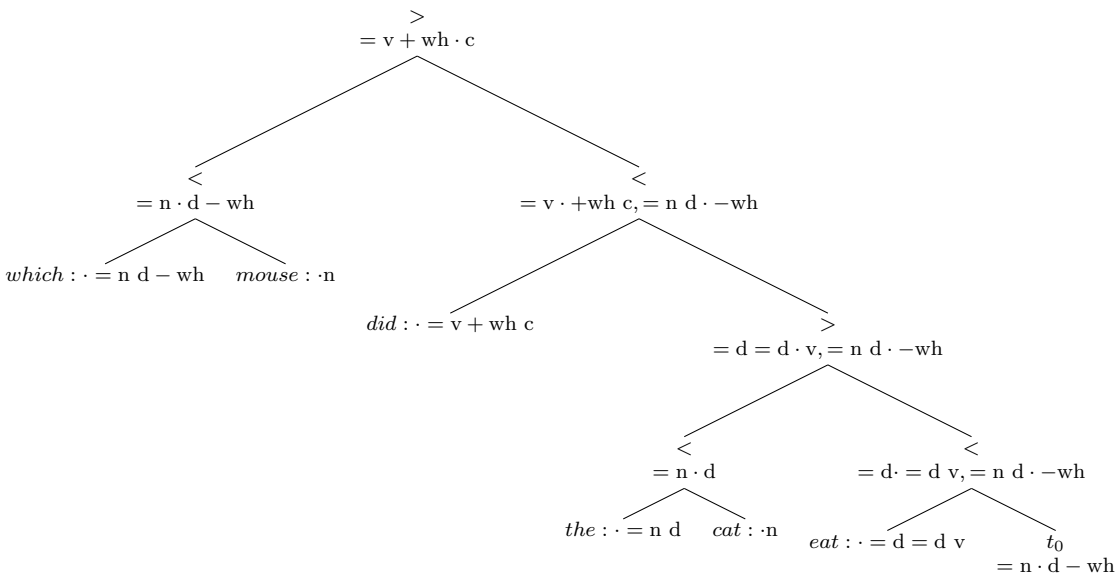
The idea of this parser is to see a minimalist grammar (MG) as a linear context-free rewriting system (LCFRS) on its derivation trees. This transformation allows us to work on a grammar without movement, generating sentences from top to bottom (on contrary of MG, which generates sentences bottom-up), and to put a probabilistic field on it.

1.1 Minimalist grammars

Minimalist grammars are designed to generate (subparts of) human natural languages. They are framed in Chomsky's minimalist program [Cho95], and were first described by E. Stabler in [Sta97]. For the sake of clarity, I will in this paper use slightly different convention to represent the trees generated by a minimalist grammar.

Minimalist grammars distinguishe themselves from more classical context-free grammars by the fact that they allow *movement*, commonly required by syntacticians to generate such sentences as (for example) 'Which mouse did the cat eat', where 'which mouse' is base-generated at the end of the sentence (in the object position), and moves at the front. The tree corresponding to this sentence, as generated by the toy Minimalist Grammar we will consider here as example, is the following:

(1)



where t_0 denotes the *trace* of the subtree 'which mouse', which has moved in front of the sentence. A trace is kept for psychological reasons, as these traces can be shown to be still present for the computation of the meaning of sentences. They also allows to keep what is called the *deep structure*, corresponding to the tree where no movement happened, and all constituents are in their base position, where lexical selection takes place.

A minimalist grammar takes several *lexical elements*, and builds a tree with them. The toy grammar we consider will have the following lexical items:

(2)
- •*mouse* :: n
- •*cat* :: n
- •*the* ::= n d
- •*which* ::= n d − wh
- •*ate* ::= d = d c
- •*eat* ::= d = d v
- •*did* ::= v + wh c
- •*did* ::= v c

This grammar generates (roughly) all affirmative/interrogative past sentences about a cat and a mouse eating each other.

As can be seen, many symbols are used next to the actual *phonetic contents* of the words (the, eat, cat, etc...). These are *syntactic features*, and the sequence of these in a lexical item represents its *syntactic category*, and is all that is needed to compute the tree. Two lexical items with the same lexical category can be freely interchanged without losing grammaticality.

The *syntactic features* may be of four types:

- *categories*, represented by a string of letters, among which is the *distinguished feature* c, used to recognise the grammatical outputs. For example, n. The set of categories will be noted Cat.
- *selectors*, represented by the string of letters of a category, preceded by a =. For example, =n. The set of selectors will be noted Sel.
- *licensees*, represented by a string of letters preceded by a -. For example, -wh. The set of licensees will be noted Licensee.
- *licensors*, represented by a string of letters corresponding to a licencee, preceded by a +. For example, +wh. The set of licensors will be noted Licensor.

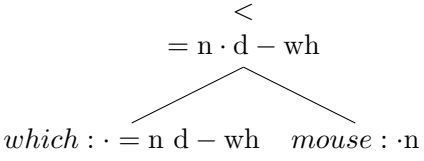
Syntactic features must follow a certain order, to ensure good formation of trees : Syn = (Select(Select∪ Licensor)*)CatLicensee*

The trees are computed by using two functions on the lexical items, to form *constituents* (i.e., trees):

- *merge*, when a selector selects a corresponding category,
- *move*, when a licensee moves to a corresponding licensor.

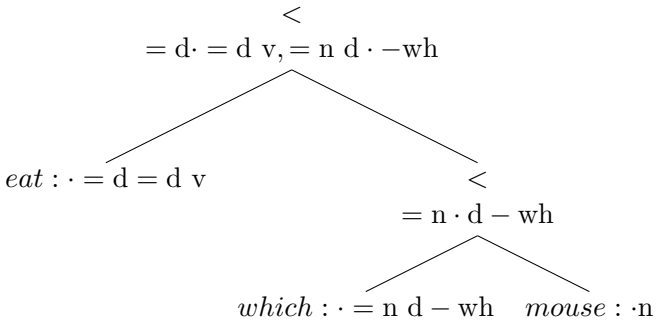
Let's see how this works on our little tree:

- Take *which* : · = n d − wh and *mouse* : ·n. We added a · in front of the syntactic categories to keep track of the derivation. Here, the two features just right of the dot (the *current features* are =n and n. It's a selector and its corresponding category, so we can *merge* them to a bigger constituent:

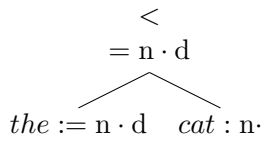


Both of the syntactic categories are copied to the root of the new constituent, with their dots moved one step right, since the current features were used. The category with the selector always comes first. If, as it is the case for the syntactic category of mouse, the dot ends up at the far right, the category may be left out, since it won't have any role in the further derivations. The $<$ indicates the *head* of the constituent, i.e. the constituent where the selector came from.

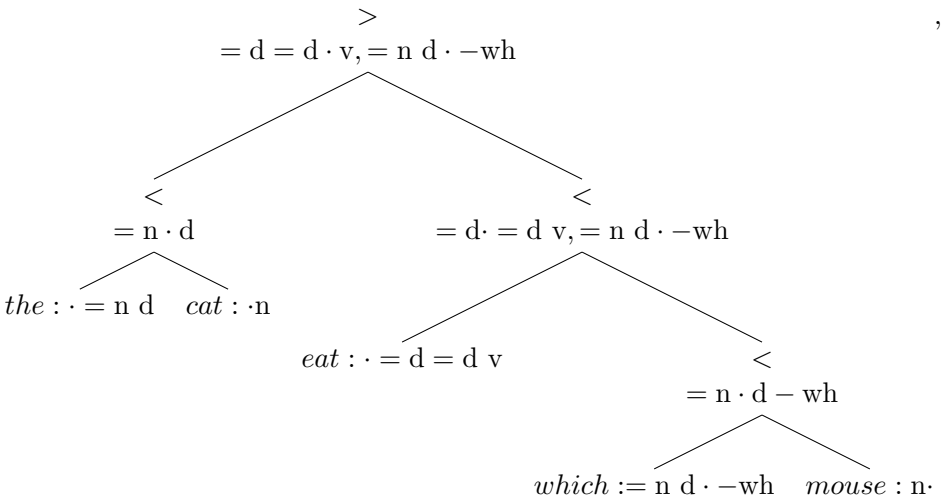
- Then merging the new constituent, whose syntactic category is $= \mathrm{n} \cdot \mathrm{d} - \mathrm{wh}$, with $eat : \cdot = \mathrm{d} = \mathrm{dv}$ (note the selector =d, corresponding to the current d) gives:



Note also that here, the second syntactic category still has something right of the dot, so stays.

- Together with 

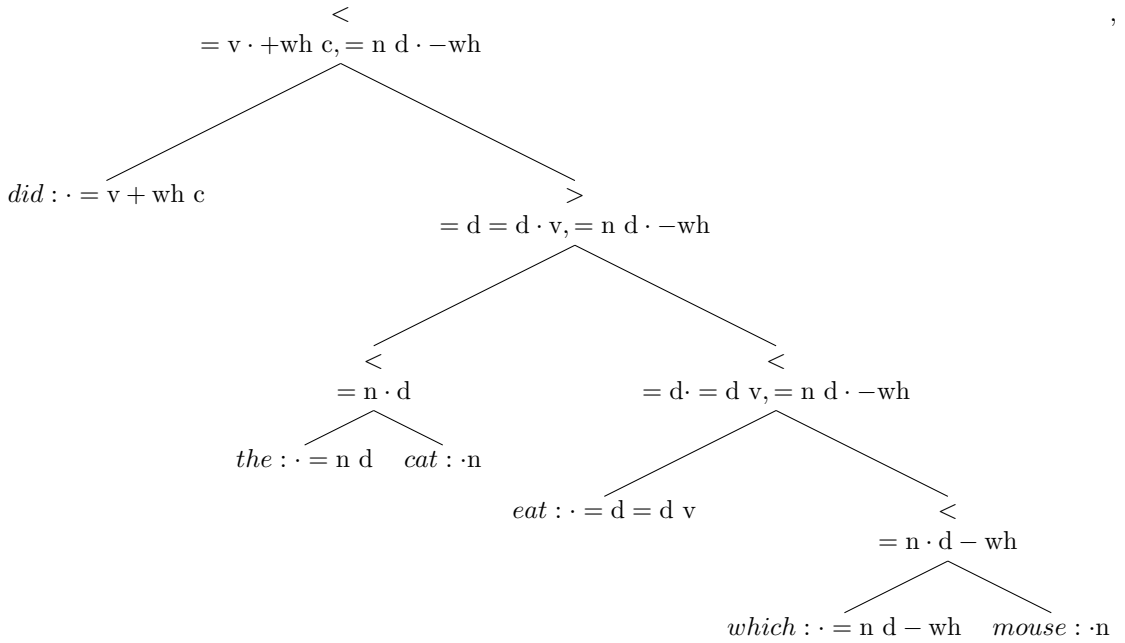
, it merges into:



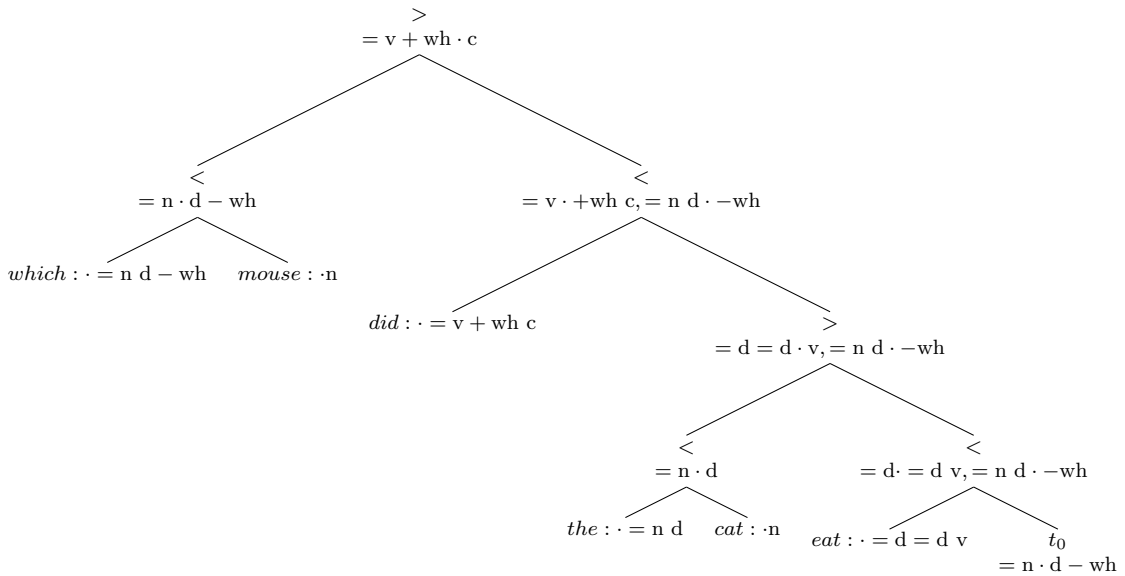
Note that for merging, only the first category in the list of syntactic categories is considered. Here, in $= \mathrm{d} \cdot = \mathrm{d}\ \mathrm{v}, = \mathrm{n}\ \mathrm{d} \cdot - \mathrm{wh}$, only $= \mathrm{d} \cdot = \mathrm{d}\ \mathrm{v}$ is considered (in fact, only $\cdot = \mathrm{d}$).

Note also that if the constituent with the selector is *complex* (i.e. is not formed of a single lexical item), as it is here the case, the merging happens in the other way: head right and selected constituent left. This has to do with the fact that english is a SVO language.

- It can then merge with *did* : · = v + wh c, giving:



- Finally, we can apply *move*. this function applies to a *single constituent*, whose first syntactic category has a *licensor* right of the dot. Here, +wh. It will then scan the other categories to find a corresponding licensee right of a dot (here, ·-wh), and move the corresponding constituent to the top of the tree, giving the final sentence:



The dots are moved as usual, the > indicates the constituent where the licensor was, and in this case, since the only feature right of a dot is the distinguished feature c, we know that the derivation yielded a grammatical output.

The constituent moved corresponds to the biggest constituent whose head is the lexical element containing the considered lincensee (this corresponds to the syntactic notion of maximal projection).

The phonetical content of this sentence is the concatenation of the phonetic contents of its leaves, in left-to-right reading: 'which mouse did the cat eat'.

The notion of *head* is an important notion in linguistics, since, by the principle of *locality of selection*, we want to restrict the amount of information that an item has access to. As such, the only information about a constituent accessible from outside (for a merge operation, for example), is the right-of-the-dot features of its head, that is, the features of the first syntactic category (minus the left-of-the-dot ones, kept only for the sake of historic bookkeeping).

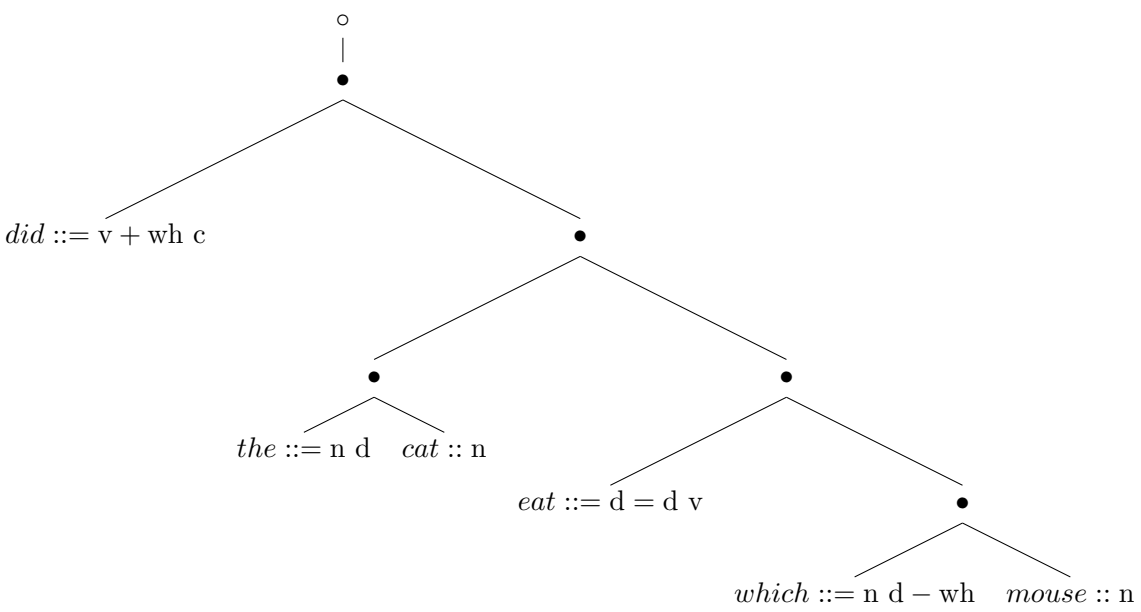
1.2 Derivation trees

Another way of representing the constituents generated by a grammar is by using its *derivation tree*:

Definition 1.1. *The* derivation tree *of a constituent is a binary tree showing the history of its building by the functions merge and move. Its leaves are lexical items, and its nodes are labelled by either* • *(merge, it's then a binary node) or* ∘ *(move, it's then a unary node).*

For example, the derivation tree of the previous example (1), is:

(3)



Let's note that to each subtree of a derivation tree corresponds a unique constituent, appearing in the construction of the main one. We can then label each node of a derivation tree by the syntactic category of the corresponding constituent.

2 Probabilistic minimalist grammars

2.1 Derivation trees of MG as LCFRS-derived trees

The basis of this method is to see minimalist derivation trees as trees generated by linear context-free rewriting systems (LCFRS). Putting a probability field on these is indeed very easy. A similar approach was also used in the minimalist parser of H. Harkema in his thesis [Har01].

We thus take a general *minimalist grammar* $\mathcal{G} = (\sigma, \text{Feat}, \text{Lex}, \mathcal{F})$.

The closure of Lex under $\mathcal{F}$ gives the outputs of the grammars. Here, we will not consider these outputs, but the derivation trees describing the process giving these outputs.

One important difference between context-free grammars, for which probabilistic versions are well-studied, and minimalist grammars is that CFG generate trees from top to bottom, by means of rules rewriting each non-terminal node by a number of other nodes, while a MG generates trees from bottom to the top, by merging and moving elements. While in the CFG case, we begin with a single symbol and then choose rules to rewrite it (thus enabling us to assign probabilities to the process by assigning probabilities to the rewriting rules), in MG, we begin with a bunch of lexical items, not necessarily compatible with each other, and merge them together (and occasionally moving them too). Here we will present a way of seeing the generating process of MG as a LCFRS, which, as CFG, generates from top to bottom with a set of rules. The differents non-terminal symbols will be defined by closure of a certain set of axioms (starting symbols) under a set of inference rules, giving this way a top-down way of generating derivation trees of MG.

2.1.1 Categories and partial outputs

In order to do this, we will first define a particular type of objects, called categories, which will be the non-terminal symbols of our LCFRS:

Definition 2.1. *A category is either a lexical item, or a sequence of the form* $[\gamma_0 \cdot \delta_0, \dots, \gamma_k \cdot \delta_k]$*, where* $\gamma_0, \dots, \gamma_k, \delta_0, \dots, \delta_k$ *are elements of Syn, or a special symbol start. A* simple *category is a category with its first dot at the leftmost place (and* $k = 0$*). Otherwise, it is a* complex *category. start is neither simple or complex.*

Categories corresponds exactly to the list of syntactic categories defined in 1.1, although our definition allows here categories which cannot be generated by a Minimalist Grammar. We will of course only be interested in those who are.

We then define a *partial output* as a string $\Delta_1 \dots \Delta_n$ of categories. These represent a particular stage in the construction of a minimalist derivation tree by the corresponding LCFRS, the different categories being the categories of the partial derivation tree which is build.

2.1.2 Axiom

There is a single axiom, the category *start*.

2.1.3 Inference rules

These rules correspond to the rewriting rules of the Linear Context-Free Rewriting System $\mathcal{S}$ corresponding to our Minimalist Grammar $\mathcal{G}$. For each possible application of one of the functions *merge*

or *move* of grammar $\mathcal{G}$ giving a particular category Δ, there is a corresponding inference rule (which gives quite a lot of rules...). Then, given a particular category Δ, the rules will tell how this particular type of tree (remember that categories describe a particular type of trees generated by the grammar) can be un-merged or un-moved into one (in case of un-move) or two (in case of un-merge) different types of trees. To this must be added the rules expanding the *start* category, and the lexicalisation rules. The first allows us to begin with a unique symbol, instead of all categories ending with the distinguished feature. The second ones allow us to leave the lexical part of the parsing up to the last moment. Thus here we are:

1. Start rules: for every lexical item $\gamma :: \delta$ c,
   Start:
   $\overline{start \longrightarrow [\delta \cdot \mathrm{c}]}$
2. Re-writing rules for complex categories:
   (a) Un-merge rules: the left-hand category is of the form $[\delta = \mathrm{x} \cdot \beta, S]$
      i. Cases where the selector was a simple tree ($\delta = \epsilon$):
         A. For any lexical item of feature string γ x,
            Unmerge-1:
            $\overline{[= \mathrm{x} \cdot \beta, S] \longrightarrow [\cdot = \mathrm{x}\ \beta][\gamma \cdot \mathrm{x}, S]}$
         B. For any element $(\gamma\ \mathrm{x} \cdot \varphi) \in S$, with $S' = S - (\gamma\ \mathrm{x} \cdot \varphi)$,
            Unmerge-3, simple:
            $\overline{[= \mathrm{x} \cdot \beta, \gamma\ \mathrm{x} \cdot \varphi, S'] \longrightarrow [\cdot = \mathrm{x}\ \beta][\gamma \cdot \mathrm{x}\ \varphi, S']}$
            It should be noted that necessarily, $\varphi \neq \emptyset$.
      ii. Cases where the selector was a complex tree:
         A. For any decomposition $S = U \sqcup V$, and any lexical item of feature string γ x,
            Unmerge-2:
            $\overline{[\delta = \mathrm{x} \cdot \beta, S] \longrightarrow [\delta \cdot = \mathrm{x}\ \beta, U][\gamma \cdot \mathrm{x}, V]}$
         B. For any element $(\gamma\ \mathrm{x} \cdot \varphi) \in S$, and any decomposition $S = U \sqcup V \sqcup (\gamma\ \mathrm{x} \cdot \varphi)$,
            Unmerge-3, complex:
            $\overline{[\delta = \mathrm{x} \cdot \beta, \gamma\ \mathrm{x} \cdot \varphi, S'] \longrightarrow [\delta \cdot = \mathrm{x}\ \beta, U][\gamma \cdot \mathrm{x}\ \varphi, V]}$
            As in 2(a)iB, φ has to be non empty.
   (b) Un-move rules: the left-hand category is of the form $[\delta + \mathrm{f} \cdot \beta, S]$
      i. For any $(\gamma - \mathrm{f} \cdot \varphi) \in S$ (necessarily unique by the Shortest Movement Constraint), with $S' = S - (\gamma - \mathrm{f} \cdot \varphi)$,
         Unmove-2:
         $\overline{[\delta' + \mathrm{f} \cdot \beta, \gamma - \mathrm{f} \cdot \varphi, S'] \longrightarrow [\delta' \cdot +\mathrm{f}\ \beta, \gamma \cdot -\mathrm{f}\ \varphi, S']}$
      ii. If there is no $(\gamma - \mathrm{f} \cdot \varphi) \in S$, then for any lexical item of feature string $\gamma - \mathrm{f}$,
         Unmove-1:
         $\overline{[\delta' + \mathrm{f} \cdot \beta, S] \longrightarrow [\delta' \cdot +\mathrm{f}\ \beta, \gamma \cdot -\mathrm{f}, S]}$
3. Re-writing rules for simple categories: for any lexical item $\lambda :: \beta$,
   Lexicalize:
   $\overline{[\cdot\beta] \longrightarrow \lambda :: \beta}$.

The set of *relevant partial outputs* can thus be defined as the closure of the axiom *start* under the inference rules. This set describes exactly all possible partial outputs given by the LCFRS $\mathcal{S}$, i.e. all possible strings of categories obtained by a cut through a tree generated by the LCFRS $\mathcal{S}$. Such a string correspond to a selection of outputs (not necessarily complete) generated by the minimalist grammar $\mathcal{G}$, such that they can be put together by application of *merge* and *move*, in the same order (two categories will get merged only if they are adjacent in the string) to obtain a complete output. A relevant output is a relevant partial output consisting of only lexical items. It corresponds to grammatical sentences.

The *relevant categories* are exactly the categories that appear in a relevant partial output. They correspond to the possible sets of similar partial trees generated by the grammar $\mathcal{G}$. They are in finite number, since, by the Shortest Movement Constraint, no two identical licensees can appear in the feature strings of a relevant category (omitting the first string). Thus two identical feature strings (diverging only by the position of the dot) can't appear together, and therefore the total length of all the feature strings of a relevant category is bounded by the sum of the length of all the feature strings of the lexical items, which is finite.

2.1.4 Derivation trees

With this formalism, we have now a quite straightformard way of defining minimalist derivation trees, in a way that enables us to put very simply probabilities on them: they are just the trees obtained by maximal application of rewriting rules to the axiom *start*. The probability is simply given by a probability field on the rules.

2.2 Probabilities on MG derivation trees

To define a probability field on the derivation trees of a MG, we now just have to put conditional probabilities on the rules discussed before, given the initial relevant category. The probability of a given tree will then be the product of the probabilities of the rules that generate it, as for regular probabilistic linear context-free rewriting systems. There can be however quite a lot of such rules and relevant categories, even if the MG is quite simple, but they can all be computed beforehand with the only knowledge of the grammar, thanks to the definition by closure of these categories. Indeed, we will see a simple method permitting to compute both the relevant categories and the inference rules that are needed.

It should be noted that the functions (Merge-1,2,3 and Move-1,2) having potentially given birth to a given relevant category are quite few (at most two), *only if we use the dot notation*, which keeps track of a minimal part of the history of the derivation. This is why the relevant categories should include all features of the lexical item potentially heading the tree (and not just the ones on the right of the dot).

To settle things a bit, we will here illustrate this method with a little example.

2.3 Example : $a^n b^n$

We will here consider the MG with the following lexical items (ϵ being the empty string):

(4)
$\bullet\epsilon :: \mathrm{c}$
$\bullet\epsilon :: = \mathrm{a} + \mathrm{m}\ \mathrm{c}$
$\bullet a :: = \mathrm{b}\ \mathrm{a} - \mathrm{m}$
$\bullet b :: \mathrm{b}$
$\bullet b :: = \mathrm{a} + \mathrm{m}\ \mathrm{b}$

This grammar generates exactly the strings of the form $a^n b^n$, $n \in \mathbb{N}$. Since this is a context-free language, we wouldn't have needed to use licensors and licencees, but for the sake of getting a language simple enough with enough rules (especially movement ones), we will work on this one.

We now want to get the relevant categories of this language, and the corresponding 'context-free rules'. A quite straightforward way to obtain them is to start from the axiom *start* and follow the inference rules to close the set of relevant categories. From *start* we apply the schemes to get all applicable rules, apply them, get some new relevant categories, apply the schemes to get new rules, apply them, etc... Since they are in finite number, this algorithm will eventually terminate, giving us all the relevant categories and needed rules (we won't get them all, since the schemes could apply to non-relevant categories, but we don't want those in any case).

So here we go:

- starting rules: we search for all lexical items whose features ends with c. There are two here, giving two different relevant categories: $\epsilon :: \mathrm{c}$ and $\epsilon :: = \mathrm{a} + \mathrm{m}\ \mathrm{c}$. We have thus two rules:

  Start: $start \longrightarrow [\cdot \mathrm{c}]$
  Start: $start \longrightarrow [= \mathrm{a} + \mathrm{m} \cdot \mathrm{c}]$

  We have now two new relevant categories: $[\cdot \mathrm{c}]$ and $[= \mathrm{a} + \mathrm{m} \cdot \mathrm{c}]$. We will now write the rules with these on the left side of the arrow.

- $[\cdot \mathrm{c}]$ correspond to case 3. There is but one lexical item with features c, which is $\epsilon :: \mathrm{c}$, so we have a single rule:

  Lexicalize: $[\cdot \mathrm{c}] \longrightarrow \epsilon :: \mathrm{c}$

  No new relevant category is created, so we can move to the next one:

- $[= \mathrm{a} + \mathrm{m} \cdot \mathrm{c}]$ corresponds to the case 2b, so we can have two possibilities. Since there is no 'S', only the case 2(b)ii can apply. We must then look for lexical items whose last feature is $-\mathrm{m}$. There is but one (and thus only one corresponding relevant category), $a :: = \mathrm{b}\ \mathrm{a} - \mathrm{m}$. So we have one possible rule:

  Unmove-1: $[= \mathrm{a} + \mathrm{m} \cdot \mathrm{c}] \longrightarrow [= \mathrm{a} \cdot + \mathrm{m}\ \mathrm{c}, = \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}]$

  We have now a new relevant category, $[= \mathrm{a} \cdot + \mathrm{m}\ \mathrm{c}, = \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}]$.

- $[= \mathrm{a} \cdot + \mathrm{m}\ \mathrm{c}, = \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}]$ corresponds to case 2(a)i. For case 2(a)iA, we have to look for a lexical item whose last feature is a. Since there is no such item, we fall back to 2(a)iB. Here we have to look in 'S' for feature strings of type $\gamma\ \mathrm{a} \cdot \varphi$. There is only one, namely $= \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}$, so we have one rule:

Unmerge-3, simple: $[= \mathrm{a} \cdot +\mathrm{m}\ \mathrm{c}, = \mathrm{b}\ \mathrm{a} \cdot -\mathrm{m}] \longrightarrow [\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{c}][= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$

We got here two more relevant categories, $[\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{c}]$ and $[= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$.

- $[\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{c}]$ corresponds to case 3, and there is but one lexical item with the corresponding features, so we have one additional rule:

  Lexicalize: $[\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{c}] \longrightarrow \epsilon :: = \mathrm{a} + \mathrm{m}\ \mathrm{c}$

- $[= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$ corresponds to case 2(a)i. We first try case 2(a)iA. We look for lexical items with last feature b. There are two such items, namely $b :: \mathrm{b}$ and $b :: = \mathrm{a} + \mathrm{m}\ \mathrm{b}$. We then have two rules:

  Unmerge-1: $[= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}] \longrightarrow [\cdot = \mathrm{b}\ \mathrm{a} - \mathrm{m}][\cdot \mathrm{b}]$

  Unmerge-1: $[= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}] \longrightarrow [\cdot = \mathrm{b}\ \mathrm{a} - \mathrm{m}][= \mathrm{a} + \mathrm{m} \cdot \mathrm{b}]$

  Since 'S' is here empty, case 2(a)iB can't apply, and we move on to the three newly discovered relevant categories, $[\cdot = \mathrm{b}\ \mathrm{a} - \mathrm{m}]$, $[\cdot \mathrm{b}]$ and $[= \mathrm{a} + \mathrm{m} \cdot \mathrm{b}]$.

- $[\cdot = \mathrm{b}\ \mathrm{a} - \mathrm{m}]$ is ready to be lexicalized, there is still only one corresponding lexical item, so we get the rule:

  Lexicalize: $[\cdot = \mathrm{b}\ \mathrm{a} - \mathrm{m}] \longrightarrow a :: = \mathrm{b}\ \mathrm{a} - \mathrm{m}$

- $[\cdot \mathrm{b}]$ is in the same case, we thus have:

  Lexicalize: $[\cdot \mathrm{b}] \longrightarrow b :: \mathrm{b}$

- $[= \mathrm{a} + \mathrm{m} \cdot \mathrm{b}]$ corresponds to the case 2(b)ii, with only one corresponding lexical item, thus the rule:

  Unmove-1: $[= \mathrm{a} + \mathrm{m} \cdot \mathrm{b}] \longrightarrow [= \mathrm{a} \cdot +\mathrm{m}\ \mathrm{b}, = \mathrm{b}\ \mathrm{a} \cdot -\mathrm{m}]$

- $[= \mathrm{a} \cdot +\mathrm{m}\ \mathrm{b}, = \mathrm{b}\ \mathrm{a} \cdot -\mathrm{m}]$ corresponds to case 2(a)iB, and we have one rule:

  Unmerge-3, simple: $[= \mathrm{a} \cdot +\mathrm{m}\ \mathrm{b}, = \mathrm{b}\ \mathrm{a} \cdot -\mathrm{m}] \longrightarrow [\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{b}][= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$

  Since $[= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$ has already been treated, we can move to the last untreated relevant category:

- $[\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{b}]$ is ready to be lexicalized:

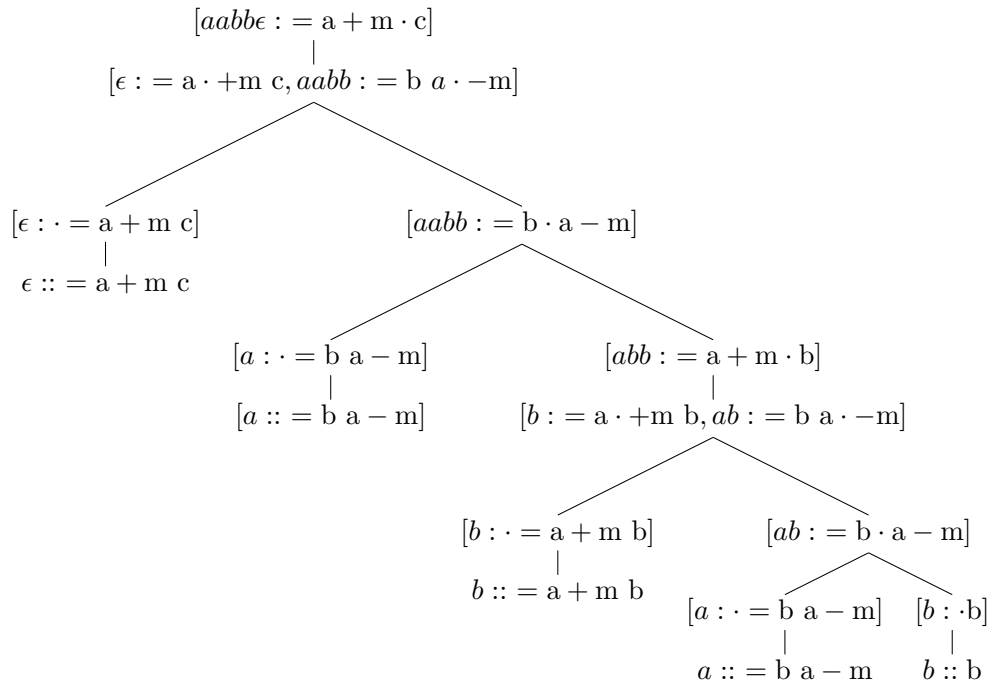
  Lexicalize: $[\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{b}] \longrightarrow b :: = \mathrm{a} + \mathrm{m}\ \mathrm{b}$

We are now ready to give probabilities to these rules, conditioned by the left-hand side. The assignment here is quite easy : apart from the two cases where there are two possible rules (axiom choice and category and $[= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$), the conditioned probability will be 1 (there is no choice). For the two other cases, we can assign any probability λ to one rule, and give the other a probability $1 - \lambda$. We can now give the following table:

$start$	$\longrightarrow [\cdot \mathrm{c}]$	Start	$\mathbb{P}(.) = \lambda$
$start$	$\longrightarrow [= \mathrm{a} + \mathrm{m} \cdot \mathrm{c}]$	Start	$\mathbb{P}(.) = 1 - \lambda$
$[\cdot \mathrm{c}]$	$\longrightarrow \epsilon :: \mathrm{c}$	Lexicalize	$\mathbb{P}(.) = 1$
$[= \mathrm{a} + \mathrm{m} \cdot \mathrm{c}]$	$\longrightarrow [= \mathrm{a} \cdot + \mathrm{m}\ \mathrm{c}, \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}]$	Unmove-1	$\mathbb{P}(.) = 1$
$[= \mathrm{a} \cdot + \mathrm{m}\ \mathrm{c}, \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}]$	$\longrightarrow [\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{c}][= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$	Unmerge-3, simple	$\mathbb{P}(.) = 1$
$[\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{c}]$	$\longrightarrow \epsilon :: = \mathrm{a} + \mathrm{m}\ \mathrm{c}$	Lexicalize	$\mathbb{P}(.) = 1$
$[= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$	$\longrightarrow [\cdot = \mathrm{b}\ \mathrm{a} - \mathrm{m}][\cdot \mathrm{b}]$	Unmerge-1, simple	$\mathbb{P}(.) = \mu$
$[= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$	$\longrightarrow [\cdot = \mathrm{b}\ \mathrm{a} - \mathrm{m}][= \mathrm{a} + \mathrm{m} \cdot \mathrm{b}]$	Unmerge-1, simple	$\mathbb{P}(.) = 1 - \mu$
$[\cdot = \mathrm{b}\ \mathrm{a} - \mathrm{m}]$	$\longrightarrow a :: = \mathrm{b}\ \mathrm{a} - \mathrm{m}$	Lexicalize	$\mathbb{P}(.) = 1$
$[\cdot \mathrm{b}]$	$\longrightarrow b :: \mathrm{b}$	Lexicalize	$\mathbb{P}(.) = 1$
$[= \mathrm{a} + \mathrm{m} \cdot \mathrm{b}]$	$\longrightarrow [= \mathrm{a} \cdot + \mathrm{m}\ \mathrm{b}, = \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}]$	Unmove-1	$\mathbb{P}(.) = 1$
$[= \mathrm{a} \cdot + \mathrm{m}\ \mathrm{b}, = \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}]$	$\longrightarrow [\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{b}][= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$	Unmerge-3, simple	$\mathbb{P}(.) = 1$
$[\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{b}]$	$\longrightarrow b :: = \mathrm{a} + \mathrm{m}\ \mathrm{b}$	Lexicalize	$\mathbb{P}(.) = 1$

We will now end by giving the probability of a particular derivation tree:

(5)



All the rules here have probability 1, except the top one, the choice of the start rule $start \longrightarrow [aabb\epsilon : = \mathrm{a} + \mathrm{m} \cdot \mathrm{c}]$, which has probability $1 - \lambda$, the one from $[aabb : = \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$, which has probability $1 - \mu$, and the one from $[ab : = \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$, which has probability μ. So the complete tree has probability $\mu(1 - \mu)\lambda$, and, for example, the subtree headed by $[b : = \mathrm{a} \cdot + \mathrm{m}\ \mathrm{b}, ab : = \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}]$ has probability μ.

2.4 The Cats and Mouses example

Let's now get back to our toy grammar (2) and see how it rewrites:

(6)
- •*mouse* :: n
- •*cat* :: n
- •*the* ::= n d
- •*which* ::= n d − wh
- •*ate* ::= d = d c
- •*eat* ::= d = d v
- •*did* ::= v + wh c
- •*did* ::= v c

The rules are the following:

1	*start*	$\longrightarrow$ [= d = d · c]	Start
2	*start*	$\longrightarrow$ [= v + wh · c]	Start
3	*start*	$\longrightarrow$ [= v · c]	Start
4	[= d = d · c]	$\longrightarrow$ [= n · d][= d· = d c]	Unmerge-2
5	[= d· = d c]	$\longrightarrow$ [· = d = d c][= n · d]	Unmerge-1
6	[· = d = d c]	$\longrightarrow$ *ate* :: = d = d c	Lexicalize
7	[= n · d]	$\longrightarrow$ [· = n d][·n]	Unmerge-1
8	[· = n d]	$\longrightarrow$ *the* :: = n d	Lexicalize
9	[·n]	$\longrightarrow$ *mouse* :: n	Lexicalize
10	[·n]	$\longrightarrow$ *cat* :: n	Lexicalize
11	[= v + wh · c]	$\longrightarrow$ [= v · +wh c, = n d · −wh]	Unmove-1
12	[= v · +wh c, = n d · −wh]	$\longrightarrow$ [· = v + wh c][= d = d · v, = n d · −wh]	Unmerge-1
13	[· = v + wh c]	$\longrightarrow$ *did* :: = v + wh c	Lexicalize
14	[= d = d · v, = n d · −wh]	$\longrightarrow$ [= n · d][= d· = d v, = n d · −wh]	Unmerge-2
15	[= d = d · v, = n d · −wh]	$\longrightarrow$ [= n · d − wh][= d· = d v]	Unmerge-3, complex
16	[= d· = d v, = n d · −wh]	$\longrightarrow$ [· = d = d v][= n · d − wh]	Unmerge-3, simple
17	[· = d = d v]	$\longrightarrow$ *eat* :: = d = d v	Lexicalize
18	[= n · d − wh]	$\longrightarrow$ [· = n d − wh][·n]	Unmerge-1
19	[· = n d − wh]	$\longrightarrow$ *which* :: = n d − wh	Lexicalize
20	[= d· = d v]	$\longrightarrow$ [· = d = d v][= n · d]	Unmerge-1
21	[= v · c]	$\longrightarrow$ [· = v c][= d = d · v]	Unmerge-1
22	[· = v c]	$\longrightarrow$ *did* :: = v c	Lexicalize
23	[= d = d · v]	$\longrightarrow$ [= n · d][= d· = d v]	Unmerge-2

3 The probabilistic top-down parser

The parser will work on an ordered list of hypothesis, which he will expand in turn during the parse of the sentence. Before beginning presenting the algorithm, some definitions are needed:

3.1 Definitions

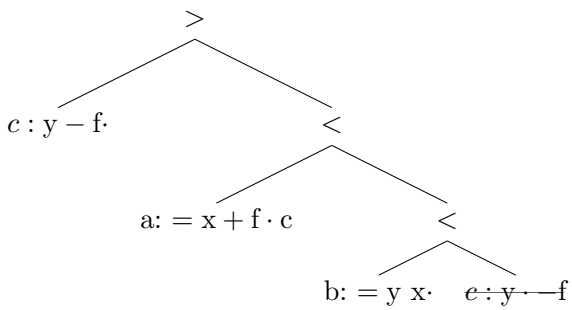
One difficulty in working with derivation trees instead of regular derived trees, is that the order of the words cannot be easily deduced (short of redoing the actual derivation). In order to keep track of the position of a category in the *derived tree* (so the parser may know in which order to expand the tree), we introduce *position indices*, which denotes positions in the derived tree from its root by a chain of digits (0 if going down left, 1 if going down right). From this perspective we can also define a *successor* operator on them, corresponding to a left-to-right sweep of the tree.

Consider the grammar given by the following lexical items:

(7) 1.$a :: = \mathrm{x} + \mathrm{f}\ \mathrm{c}$
2.$b :: = \mathrm{y}\ \mathrm{x}$
3.$c :: \mathrm{y} - \mathrm{f}$

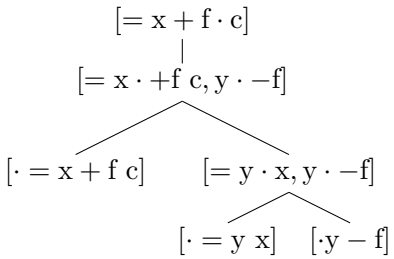
This grammar will generate the derived tree:

(8)



Corresponding to this tree is the derivation tree:

(9)



The parser should try to expand the nodes leading to the first leaf *in the derived tree* (8), but is actually building the *derivation tree* (9). As such, it should begin by expanding right-most nodes, then switch back to left-most ones when c is parsed to parse a, etc... Position indices showing in which position which category is can be computed online and incorporated to the derivation tree, for example $0/$ for all categories corresponding to c, since its final position is just one branch down and left from the root. The parser will just have to expand the unexpanded nodes with lowest (i.e.

leftmost) position indices. In order to do this, it can keep track of a *pointer* telling up to which point nodes have been expanded, and expand the corresponding one. Then upgrading the pointer with the adequate notion of successor keeps the parser working. To formalize this:

Definition 3.1. *A* position index *is a element* $\pi \in \{0,1\}^* \cup \{-1\}$.
Its successor $s(\pi)$ *is defined to be:*

$$s(\pi) = \begin{cases} \alpha 1 & \text{if } \pi = \alpha 0 \beta, \beta \in 1^* \\ -1 & \text{if } \pi \in 1^* \\ \textit{undefined} & \textit{otherwise} \end{cases}$$

Two positions indices π, π' correspond *if* $\pi' = \pi\beta, \beta \in 0^*$. *In this case, we say also that* π points to π'.

The notion of *correspondence* enables the parser to have some liberty in the pointer indicating the index to be expanded. Indeed, the parser will not try to expand the node with the index exactly equal to the pointer, but just corresponding to it, that is, equal to the pointer with as many 0s as possible following, or, in the derived tree, down the leftmost path from the node indicated by the pointer, which is what we would want: the first unexpanded node down the pointer.

Definition 3.2. *A* situated category *is a pair* $\alpha^n/[F^n]$, *where* α^n *is a sequence of* n *position indices and* F^n *is a sequence of* n *dotted feature strings (so that* $[F^n]$ *is a category). For readability, we will write* $\langle \alpha_1, \ldots, \alpha_n \rangle / [F_1, \ldots, F_n]$ *as* $[\alpha_1/F_1, \ldots, \alpha_n/F_n]$.

Definition 3.3. *A* hypothesis *is a 5-uple* (T, π, p, s, Δ) *where:*

- T *is a finite set of situated categories (the nodes of the partial derivation tree),*
- π *is a position index, the* pointer*, pointing to the next node to expand,*
- $p \in [0,1]$ *is the probability of the hypothesis,*
- s *is a dotted input string, and*
- Δ *is the sequence of rules used to obtain this hypothesis from the axiom start.*

The dotted input string s is the string of word of the phrase being parsed, with a dot indicating up to which point it has already been parsed (in fact, up to which point the words have been scanned). For example, if $s =$ "The cat has $\cdot$ eaten the mouse", this means that this hypothesis has already scanned (i.e., recognised a node for) the words "The", "cat" and "has", but not yet "eaten", "the" and "mouse".

3.2 Position indices and nodes

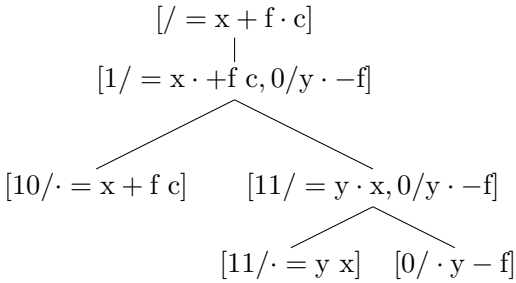
The parser will expand the hypothesis trees in a quite particular way, corresponding to a left-to-right reading of the output sentence. Since movement is possible in MG, the parser will have to keep track of the 'position' of the different elements, to only expand the leafs corresponding to the currently parsed word. This is the role of the position indices.

A position index different from -1 will represent a particular subtree in the final *derived* tree, where the traces of the moved nodes are deleted (moving up its sister to the position of its mother). The position index $\alpha_0 \dots \alpha_k$ corresponds to the subtree dominated by the node obtained by going down in the tree from its root, left if $\alpha_0 = 0$, right otherwise, then again left if $\alpha_1 = 0$, right otherwise, etc...

Back to our toy grammar (7):

1. $a :: = \mathrm{x} + \mathrm{f}\ \mathrm{c}$
2. $b :: = \mathrm{y}\ \mathrm{x}$
3. $c :: \mathrm{y} - \mathrm{f}$

The derivation tree with indexed relevant features corresponding to (8):



The indexed relevant category at the root of the derivation tree has a empty position string since it represent the derived tree itself, and in $[11/ = \mathrm{y} \cdot \mathrm{x}, 0/\mathrm{y} \cdot -\mathrm{f}]$ for example, we have $11/ = \mathrm{y} \cdot \mathrm{x}$ because this relevant category represent the tree under the node obtained if you go right (1), then right again (11) from the root node of the derived tree (without the moving categories, since they will move so won't end up at the same place). We have similarly $0/\mathrm{y} \cdot -\mathrm{f}$ because the subtree described by $\mathrm{y} \cdot -\mathrm{f}$ ends up as the left (0) daughter of the root of the derived tree.

The assignment of these position strings is given by the inference rules, which will be discussed later.

3.3 Axiom

The axiom of the parser are exactly the same as the axiom for the LCFRS corresponding to our MG discussed in 2, plus an empty position string (it represents the whole derived tree...). Its probability will be of course 1, and the pointer will be set as ϵ. So, if the phrase to be parsed is ω, we have a parser axiom $(\epsilon/start, \epsilon, 1, \cdot\omega, \langle\ \rangle)$.

3.4 Inference rules

We here have exactly the same inference rules as before, exept that these will assign position strings too. So here they are:

1. Start rules: for every lexical item $\gamma :: \delta\ \mathrm{c}$,
   Start: $\epsilon/start \longrightarrow [\epsilon/\delta \cdot \mathrm{c}]$

2. Un-merge rules: the left-hand category is of the form $[\alpha/\delta = \mathrm{x} \cdot \theta, S]$
   (a) Cases where the selector was a simple tree ($\delta = \epsilon$):
      i. For any lexical item of feature string γ x,
      Unmerge-1:

$$[\alpha/ = \mathrm{x} \cdot \theta, S] \longrightarrow \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ [\alpha 0/\cdot = \mathrm{x}\ \theta] \quad [\alpha 1/\gamma \cdot \mathrm{x}, S] \end{array}$$

      t is here s if $\gamma = \emptyset$ (and thus $S = \emptyset$ too), and c otherwise.
      ii. For any element $(\gamma\ \mathrm{x} \cdot \varphi) \in S$, with $S' = S - (\gamma\ \mathrm{x} \cdot \varphi)$,
      Unmerge-3, simple:

$$[\alpha/ = \mathrm{x} \cdot \theta, \beta/\gamma\ \mathrm{x} \cdot \varphi, S'] \longrightarrow \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ [\alpha/\cdot = \mathrm{x}\ \theta] \quad [\beta/\gamma \cdot \mathrm{x}\ \varphi, S'] \end{array}$$

      t is here s if $\gamma = \emptyset$ (and thus $S' = \emptyset$ too), and c otherwise. It should be noted that necessarily, $\varphi \neq \emptyset$.
   (b) Cases where the selector was a complex tree:
      i. For any decomposition $S = U \sqcup V$, and any lexical item of feature string γ x,
      Unmerge-2:

$$[\alpha/\delta = \mathrm{x} \cdot \theta, S] \longrightarrow \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ [\alpha 1/\delta\cdot = \mathrm{x}\ \theta, U] \quad [\alpha 0/\gamma \cdot \mathrm{x}, V] \end{array}$$

      t is, as always, s if $\gamma = \emptyset$ (and thus V has to be empty too), and c otherwise.
      ii. For any element $(\gamma\ \mathrm{x} \cdot \varphi) \in S$, and any decomposition $S = U \sqcup V \sqcup (\gamma\ \mathrm{x} \cdot \varphi)$,
      Unmerge-3, complex:

$$[\alpha/\delta = \mathrm{x} \cdot \theta, \beta/\gamma\ \mathrm{x} \cdot \varphi, S'] \longrightarrow \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ [\alpha/\delta\cdot = \mathrm{x}\ \theta, U] \quad [\beta/\gamma \cdot \mathrm{x}\ \varphi, V] \end{array}$$

      t is still s if $\gamma = \emptyset$ (and thus V has to be empty too), and c otherwise. As in 2(a)iB, φ has to be non empty.
3. Un-move rules: the left-hand category is of the form $[\delta + \mathrm{f} \cdot \theta, S]$
   (a) For any $(\gamma - \mathrm{f} \cdot \varphi) \in S$ (necessarily unique by the Shortest Movement Constraint), with $S' = S - (\gamma - \mathrm{f} \cdot \varphi)$,
   Unmove-2:

$$[\alpha/\delta' + \mathrm{f} \cdot \theta, \beta/\gamma - \mathrm{f} \cdot \varphi, S'] \longrightarrow \begin{array}{c} \circ \\ | \\ [\alpha/\delta' \cdot +\mathrm{f}\ \theta, \beta/\gamma \cdot -\mathrm{f}\ \varphi, S'] \end{array}$$

   (b) If there is no $(\gamma - \mathrm{f} \cdot \varphi) \in S$, then for any lexical item of feature string $\gamma - \mathrm{f}$,
   Unmove-1:

$$[\alpha/\delta' + \mathrm{f} \cdot \theta, S] \longrightarrow \begin{array}{c} \circ \\ | \\ [\alpha 1/\delta' \cdot +\mathrm{f}\ \theta, \alpha 0/\gamma \cdot -\mathrm{f}, S] \end{array}$$

There is no lexicalise rule, since it will in fact be replaced by a '*scan* rule', checking if the feature string of the word currently parsed corresponds to the current feature string.

3.5 Top-down parser

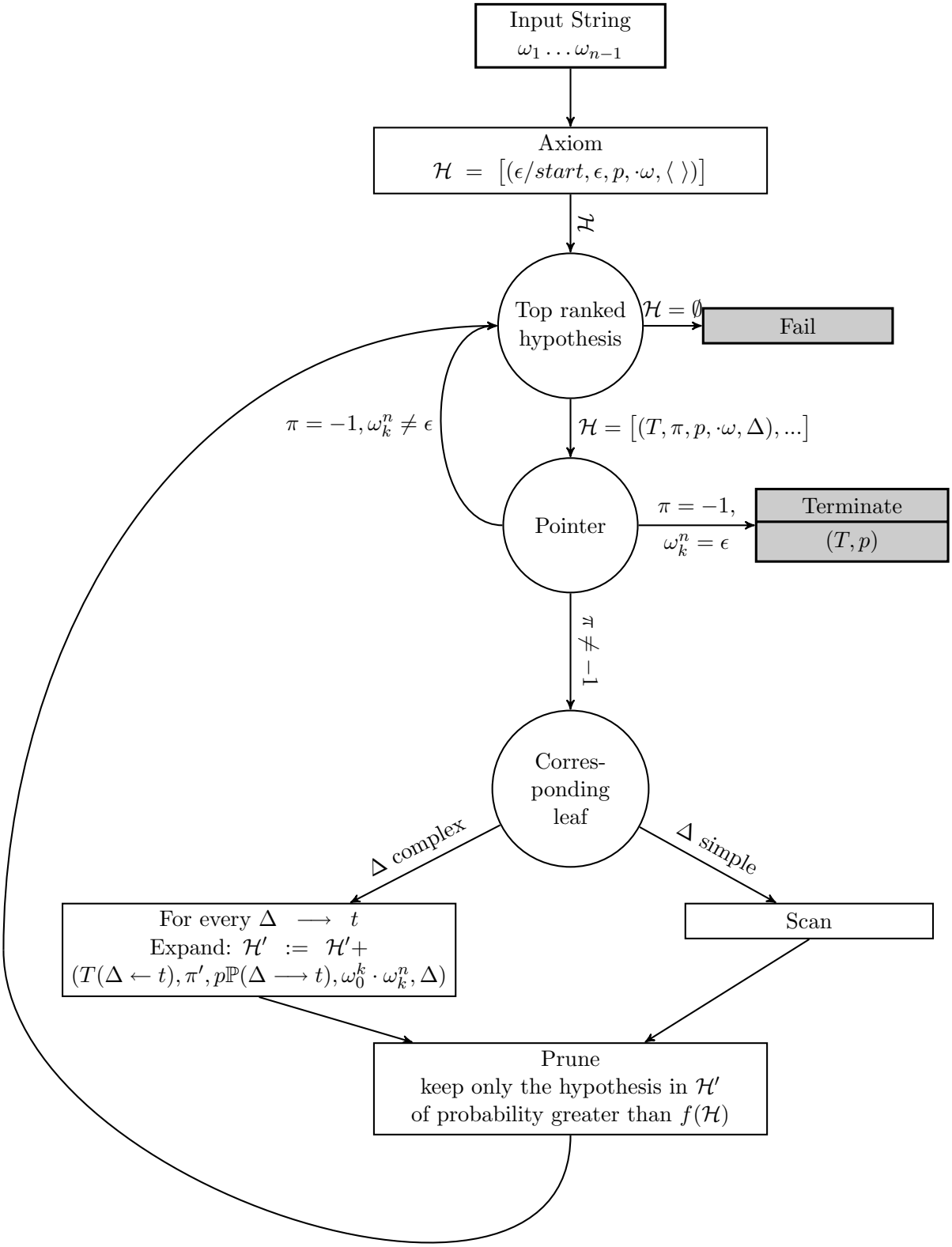
The parser takes an *input string* $\omega = \omega_0 \dots \omega_{n-1}$, a minimalist grammar $\mathcal{G}$ rewrited into a LCFRS $\mathcal{S}$ and a *beam function* f, setting a threshold to the probability of the selected hypothesis. It will work on a *priority queue* of hypothesis $\mathcal{H}$. The function f can be very general, here we will consider that its argument is the priority queue $\mathcal{H}$. The parser works as following:

1. **Beginning:** The parser start with the queue of hypothesis consisting of the axiom $(\epsilon/start, \epsilon, 1, \cdot\omega, \langle\,\rangle)$ of the grammar.
2. **Expanding:** At each step, the parser will:
   - take the top-ranked hypothesis $(T, \pi, p, \omega_0^k \cdot \omega_k^n, \Delta)$ (i.e. the hypothesis with greatest p) in the priority queue,
   - check the corresponding position string pointer. If $\pi = -1$, and the parsing dot in $\omega_0^k \cdot \omega_k^n$ is at the far right (i.e. $k = n$), then the parser terminates and returns the sequence of rules Δ. If the phrase is not completely parsed ($\pi = -1$ but $k < n$), the hypothesis is deleted and the parser moves to the next one. If $\pi \neq -1$, the parser moves to the next step, and tries to:
   - find the leaf of T, C, in which is the position string α corresponding to the pointer π. If $\alpha \neq \pi$, π is set to α.
   - expand C. For this, we have two possibilities:
     (a) **Expand:** If C is a complex situated category, the parser will delete the current hypothesis $(T, p, \pi, \omega_0^k \cdot \omega_k^n, \Delta)$ and add to the priority queue, for all possible inference rules $C \longrightarrow t$, a new hypothesis $(T', \pi', p\mathbb{P}(C \longrightarrow t), \omega_0^k \cdot \omega_k^n, \Delta@C \longrightarrow t)$, such that T' is T where the node C has been replaced by t, and π' is either $\pi 0$ if the rule did change the value of the position string corresponding to π (i.e. if the rule was Unmerge-1, Unmerge-2 and Unmove-1, and the first element of C had α for position string), and π in the other cases. @ is the concatenation operator.
     (b) **Scan:** If C is a simple indexed category, say $C = [\alpha/\cdot\delta]$, then the parser will delete the current hypothesis $(T, \pi, p, \omega_0^k \cdot \omega_k^n, \Delta)$, and try to lexicalize C. It will do two things:
        i. **Scan, ϵ:** If there is a rule $[\cdot\delta] \longrightarrow \epsilon :: \delta$, then a new hypothesis $(T', s(\pi), p\mathbb{P}([\cdot\delta] \longrightarrow \epsilon :: \delta), \omega_0^k \cdot \omega_k^n, \Delta@[\cdot\delta] \longrightarrow \epsilon :: \delta)$ is added to the priority queue, where T' is T where the leaf C was replaced by $\epsilon :: \delta$.
        ii. **Scan, $\not\epsilon$:** If $\omega_k :: \delta$ is in the grammar, then a new hypothesis $(T', s(\pi), p\mathbb{P}([\cdot\delta] \longrightarrow \omega_k :: \delta), \omega_0^k\omega_k \cdot \omega_{k+1}^n, \Delta@[\cdot\delta] \longrightarrow \omega_k :: \delta)$ is added to the priority queue, where T' is T where the leaf C was replaced by $\omega_k :: \delta$.

        If these two steps fail, then no hypothesis is added to the priority queue.

     The new hypothesis are inserted in the priority queue at their 'right place', i.e. after all hypothesis of higher probability.
   - **Prune:** The parser deletes all hypothesis of the priority queue whose probability is lower than $f(\mathcal{H})$.

   If $\mathcal{H}$ is empty, then the parse failed and the sentence is judged ungrammatical.


Input String
$\omega_1 \ldots \omega_{n-1}$
Axiom
$\mathcal{H} = [(\epsilon/start, \epsilon, p, \cdot\omega, \langle\ \rangle)]$
$\mathcal{H}$
Top ranked hypothesis
$\mathcal{H} = \emptyset$
Fail
$\pi = -1, \omega_k^n \neq \epsilon$
$\mathcal{H} = [(T, \pi, p, \cdot\omega, \Delta), ...]$
Pointer
$\pi = -1,$
$\omega_k^n = \epsilon$
Terminate
(T, p)
$\pi \neq -1$
Corres-ponding leaf
Δ complex
Δ simple
For every $\Delta \longrightarrow t$
Expand: $\mathcal{H}' := \mathcal{H}' +$
$(T(\Delta \leftarrow t), \pi', p\mathbb{P}(\Delta \longrightarrow t), \omega_0^k \cdot \omega_k^n, \Delta)$
Scan
Prune
keep only the hypothesis in $\mathcal{H}'$
of probability greater than $f(\mathcal{H})$

3.6 Example

Here we will present a small example of the parsing of a particular sentence of the grammar we presented in (4), consisting of the the lexical items:

1. $\epsilon :: \mathrm{c}$
2. $\epsilon :: = \mathrm{a} + \mathrm{m}\ \mathrm{c}$
3. $a :: = \mathrm{b}\ \mathrm{a} - \mathrm{m}$
4. $b :: \mathrm{b}$
5. $b :: = \mathrm{a} + \mathrm{m}\ \mathrm{b}$

The corresponding LCFRS was consisting of these rules:

S1	*start*	$\longrightarrow$	$[\cdot \mathrm{c}]$	Start	$\mathbb{P}(.) = .7$
S2	*start*	$\longrightarrow$	$[= \mathrm{a} + \mathrm{m} \cdot \mathrm{c}]$	Start	$\mathbb{P}(.) = .3$
L1	$[\cdot \mathrm{c}]$	$\longrightarrow$	$\epsilon :: \mathrm{c}$	Lexicalize	$\mathbb{P}(.) = 1$
Mv1	$[= \mathrm{a} + \mathrm{m} \cdot \mathrm{c}]$	$\longrightarrow$	$\circ$ $\vert$ $[= \mathrm{a} \cdot + \mathrm{m}\ \mathrm{c}, \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}]$	Unmove-1	$\mathbb{P}(.) = 1$
Mg1	$[= \mathrm{a} \cdot + \mathrm{m}\ \mathrm{c}, \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}]$	$\longrightarrow$	$\bullet$ $[\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{c}] \quad [= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$	Unmerge-3, simple	$\mathbb{P}(.) = 1$
L2	$[\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{c}]$	$\longrightarrow$	$\epsilon :: = \mathrm{a} + \mathrm{m}\ \mathrm{c}$	Lexicalize	$\mathbb{P}(.) = 1$
Mg2	$[= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$	$\longrightarrow$	$\bullet$ $[\cdot = \mathrm{b}\ \mathrm{a} - \mathrm{m}] \quad [\cdot \mathrm{b}]$	Unmerge-1, simple	$\mathbb{P}(.) = .4$
Mg3	$[= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$	$\longrightarrow$	$\bullet$ $[\cdot = \mathrm{b}\ \mathrm{a} - \mathrm{m}] \quad [= \mathrm{a} + \mathrm{m} \cdot \mathrm{b}]$	Unmerge-1, simple	$\mathbb{P}(.) = .6$
L3	$[\cdot = \mathrm{b}\ \mathrm{a} - \mathrm{m}]$	$\longrightarrow$	$a :: = \mathrm{b}\ \mathrm{a} - \mathrm{m}$	Lexicalize	$\mathbb{P}(.) = 1$
L4	$[\cdot \mathrm{b}]$	$\longrightarrow$	$b :: \mathrm{b}$	Lexicalize	$\mathbb{P}(.) = 1$
Mv2	$[= \mathrm{a} + \mathrm{m} \cdot \mathrm{b}]$	$\longrightarrow$	$\circ$ $\vert$ $[= \mathrm{a} \cdot + \mathrm{m}\ \mathrm{b}, = \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}]$	Unmove-1	$\mathbb{P}(.) = 1$
Mg4	$[= \mathrm{a} \cdot + \mathrm{m}\ \mathrm{b}, = \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}]$	$\longrightarrow$	$\bullet$ $[\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{b}] \quad [= \mathrm{b} \cdot \mathrm{a} - \mathrm{m}]$	Unmerge-3, simple	$\mathbb{P}(.) = 1$
L5	$[\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{b}]$	$\longrightarrow$	$b :: = \mathrm{a} + \mathrm{m}\ \mathrm{b}$	Lexicalize	$\mathbb{P}(.) = 1$

Here we took $\lambda = .7$ and $\mu = .4$.

We will now try to parse the string *aabb*, which is generated by the grammar (the ϵ is of course omitted).

- The parser begins with his stack consisting of the *axiom* of the grammar:

$$\mathcal{H} = ((\epsilon/start, \epsilon, 1, \cdot aabb, \langle\ \rangle))$$

- The parsers takes the top-ranked hypothesis, $(\epsilon/start, \epsilon, 1, \cdot aabb, \langle\ \rangle)$, its pointer (null), the corresponding leaf, *start*, and tries to expand it. There are two possibilities, which are added to the hypothesis queue, ordered by decreasing possibilities:

$$\mathcal{H} = (([\epsilon/ \cdot \mathrm{c}], \epsilon, .7, \cdot aabb, \langle S1\rangle), ([/ = \mathrm{a} + \mathrm{m} \cdot \mathrm{c}], \epsilon, .3, \cdot aabb, \langle S2\rangle))$$

- The parser now takes the top-ranked hypothesis, $([\epsilon/ \cdot \mathrm{c}], \epsilon, .7, \cdot aabb, \langle S1\rangle)$, its pointer (null), the corresponding leaf, $[/ \cdot \mathrm{c}]$, and try to scan it since it is a simple category. There is only one rule whose left-size is $[/ \cdot \mathrm{c}]$, L1, with probability 1. The corresponding word is empty, so the scan succeeds, the pointer is increased to -1 and the new hypothesis is added to the queue in place of the old one:

$$\mathcal{H} = (([\epsilon/\epsilon :: \mathrm{c}], -1, .7, \cdot aabb, \langle S1, L1\rangle), ([/ = \mathrm{a} + \mathrm{m} \cdot \mathrm{c}], \epsilon, .3, \cdot aabb, \langle S2\rangle))$$

- The parser takes once more the top-ranked analysis, $([\epsilon/\epsilon :: \mathrm{c}], -1, .7, \cdot aabb, \langle S1, L1\rangle)$. Its pointer is -1, so the parser checks if the parse is indeed over. No, since the string left of the dot is non-empty. The current hypothesis is now deleted, and the new queue is fed to the parser:

$$\mathcal{H} = (([/ = \mathrm{a} + \mathrm{m} \cdot \mathrm{c}], \epsilon, .3, \cdot aabb, \langle S2\rangle))$$

- The top-ranked analysis is now the only one in the queue, $(\cdot aabb, [/ = \mathrm{a} + \mathrm{m} \cdot \mathrm{c}], .3,\ )$. Its pointer is null, the corresponding leaf is $[/ = \mathrm{a} + \mathrm{m} \cdot \mathrm{c}]$, which the parser will try to expand. There is only one rule, Mv1:$[= \mathrm{a} + \mathrm{m} \cdot \mathrm{c}] \longrightarrow \begin{array}{c} \circ \\ | \\ [= \mathrm{a} \cdot + \mathrm{m}\ \mathrm{c}, \mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}] \end{array}$, so the hypothesis is replaced by the new one:

$$\mathcal{H} = \left(\begin{array}{c} \circ \\ | \\ [1/ = \mathrm{a} \cdot + \mathrm{m}\ \mathrm{c}, 0/\mathrm{b}\ \mathrm{a} \cdot - \mathrm{m}] \end{array}, 0, .3, \cdot aabb, \langle S2, Mv1\rangle) \right)$$

(note that the pointer was modified since the position vector of the expanded element was modified by the rule)

- The parser goes on, giving the new queue:

$$\mathcal{H} = \left(\begin{array}{c} \circ \\ | \\ \bullet \\ \diagup\ \diagdown \\ [1/\cdot = \mathrm{a} + \mathrm{m}\ \mathrm{c}] \quad [0/ = \mathrm{b} \cdot \mathrm{a} - \mathrm{m}] \end{array}, 0, .3, \cdot aabb, \langle S2, Mv1, Mg1\rangle) \right)$$

- And so forth...

4 Proofs of soundness and completeness

4.1 Soundess of the pointer

We will here demonstrate that the parser, at each step, will indeed find the correct leaf to expand with the current pointer.

First we modify a little, for convenience of the proof, our definition of a position string:

Definition 4.1. *A* position index *(and a* pointer*) is a dotted, almost null binary sequence* $\alpha \cdot \beta, \alpha\beta \in \{0,1\}^k\bar{0}$ *for some* k*.*

A position index and a pointer correspond *if their undotted sequences are the same.*

The set of all undotted position indexes is naturally ordered by the lexicographic order.

Note: This definition is consistent with the one I proposed in the precedent section, the dot being the place where the 'new' position indexes are to be truncated to obtain the 'old' ones.

Proposition 4.1. *There is a one-to-one application from the set of all position indexes* $\mathcal{I}$ *to the set* $\mathcal{ST}$ *of all subtrees of the infinite complete binary tree* $\mathcal{T}$*. The head of the subtrees corresponding to the positions indexes of the type* $\alpha_1 \dots \alpha_n \cdot \bar{0}$ *are exactly the nodes of depth* n *of the tree. We thus have a notion of domination on the position indexes, corresponding to the notion of domination in the tree (by convention, a node dominates itself). A position index* $\alpha \cdot \bar{0}$ *dominate another position index* $\alpha' \cdot \bar{0}$ *if* α *is a prefix of* α'*.*

Proof. Let $\phi : \begin{array}{ccc} \mathcal{I} & \longrightarrow & \mathcal{ST} \\ \alpha_1 \dots \alpha_n \cdot \bar{0} & \longrightarrow & T \end{array}$ where T is the subtree headed by the node obtained by, starting from the root of $\mathcal{T}$, for each α_i, going left if $\alpha_i = 0$ and right otherwise. This application has clearly all the above properties. □

Lemma 4.1. *For every cut* C *of position indexes,*

i. if $\beta \cdot \bar{0}$ *is in the cut* C*, then there is no* $\beta\gamma \cdot \bar{0}$ *in* C*, for every* $\gamma \in \{0,1\}^+$*.*

ii. If $\beta 0\gamma \cdot \bar{0}$ *is in the cut* C*, then there is a unique* $k \in \mathbb{N}$ *such that* $\beta 10^k \cdot \bar{0}$ *is in the cut.*

iii. The lexicographic order can be extended to the dotted *elements of* C*.*

Proof. Let n be the depth of the cut.

i. Suppose that we have $\beta \cdot \bar{0}$ and $\beta\gamma \cdot \bar{0}$ in the cut, for some $\gamma \in \{0,1\}^+$. Then $\beta\gamma 0^n \cdot \bar{0}$ is dominated by both $\beta \cdot \bar{0}$ and $\beta\gamma \cdot \bar{0}$, which is a contradiction.

ii. Since C is a cut, $\beta 10^n \cdot \bar{0}$ must be dominated by a unique element of C, say $\delta \cdot \bar{0}$. δ is then a prefix of $\beta 10^n$. But δ cannot be a prefix of $\beta 0\gamma$, since otherwise $\delta \cdot \bar{0}$ would dominate $\beta 0\gamma \cdot \bar{0}$, which is already dominated by itself. So δ must be of the form $\beta 10^k, k < n$. The unicity follows from i.

iii. This follows directly from i.

□

We can now prove that the parser will never have a pointer problem:

Theorem 4.1. *At each point of the parse, the position indexes of (all) the hypothesis form a cut, the already scanned position indexes form a prefix set of all the position indexes (for the lexicographic order), and the pointer correspond to the smallest unscanned position index (which exists since the set is finite), or is possibly -1 if there is none.*

Proof. We'll prove this result by recurrence on the number of steps done by the parser.

- At the beginning, the set of the position indexes is reduced to $\{\cdot\bar{0}\}$, and the pointer is $\cdot\bar{0}$, so the result is trivially true.
- Suppose that at step n, the position indexes of (all) the hypothesis form a cut, the already scanned position indexes form a prefix set of all the position indexes (for the lexicographic order), and the pointer correspond to the smallest scanned position index (which exists since the set is finite), or is possibly -1 if there is none. Let $\alpha \cdot \bar{0}$ be the position index corresponding to the pointer (unique by hypothesis), and $[\beta \cdot \bar{0}/\Delta, \ldots, \alpha \cdot \bar{0}/\Delta', \ldots]$ the leaf to be expanded. By hypothesis, $\alpha \cdot \bar{0} \leq \beta \cdot \bar{0}$. Let $(\delta_1, \ldots, \delta_k//\alpha \cdot \bar{0}, \ldots, \beta \cdot \bar{0}, \ldots)$ the positions indexes, lexicographically ordered, the scanned ones left of //. By hypothesis, this is a cut. The state of the parser at step $n+1$ can be obtained by four different cases:
  1. if the position index corresponding to the pointer, $\alpha \cdot \bar{0} (< \beta \cdot \bar{0})$, is not modified by the rules, two cases:
     (a) if the main position index of the leaf being expanded, $\beta \cdot \bar{0}$, is not modified (i.e. during Un-merge-3 or Un-move-2), no position indexes are modified, nor the pointer, so the result still holds.
     (b) if the main position index of the leaf being expanded, $\beta \cdot \bar{0}$, is modified (i.e. during Un-merge-1,-2 or Un-move-1), the new position indexes are (lexicographically ordered, the already scanned ones left of the //)
     $(\delta_1, \ldots, \delta_k//\alpha \cdot \bar{0}, \ldots, \beta 0 \cdot \bar{0}, \beta 1 \cdot \bar{0}, \ldots)$.
     This is trivially still a cut, the pointer is still $\alpha \cdot \bar{0}$, corresponding to the smallest unscanned position indexes $\alpha \cdot \bar{0}$.
  2. if the position index corresponding to the pointer, $\alpha \cdot \bar{0} (= \beta \cdot \bar{0})$, is modified by the rules, the new position indexes are
     $(\delta_1, \ldots, \delta_k//\alpha 0 \cdot \bar{0}, \alpha 1 \cdot \bar{0}, \ldots)$.
     This is still trivially a cut, and the new pointer is $\alpha 0 \cdot \bar{0}$, corresponding to the smallest unscanned position index $\alpha 0 \cdot \bar{0}$.
  3. if the parser *scans* the leaf (and then $\alpha \cdot \bar{0} = \beta \cdot \bar{0}$):
     If $\alpha \in 1^*$, the pointer is set to -1, there cannot be a greater item in the cut than $\alpha \in 1^*$ (since it would then have to have α as a prefix, which is impossible by lemma 4.1), so the result is true.
     Let's then write $\alpha = \gamma 01^m$. The pointer is set to $\gamma 1 \cdot \bar{0}$. The new position indexes are
     $(\delta_1, \ldots, \delta_k, \gamma 01^m \cdot \bar{0}//\zeta \cdot \bar{0}, \ldots)$.
     Since the only thing that changed here is the position of the //, this is still a cut. By lemma 4.1, there exists a unique $\xi = \gamma 10^r$ in the cut, since $\alpha = \gamma 01^m$ was in it. This being the smallest position index greater than $\alpha = \gamma 01^m$, we have $\zeta = \xi = \gamma 10^r$, which is corresponding to the new pointer. This ends the demonstration.

□

4.2 Completeness

Here we demonstrate the completeness of the parser, without the pruning step, i.e. that if the string to be parsed can indeed be generated by the grammar, then the parser will eventually parse it.

Lemma 4.2. *Let $(a_n)_{n\in\mathbb{N}} \in (\Sigma^*)^{\mathbb{N}}$ a infinite sequence of finite distinct strings over a finite alphabet Σ. Suppose that the following hold:*

$$\forall n \in \mathbb{N}, \text{ all prefixes of } a_n \text{ are in } \{a_k, k \leq n\}. \tag{1}$$

Then there exists an infinite sequence $(x_k)_{k\in\mathbb{N}} \in \Sigma^{\mathbb{N}}$ such that $(x_0 \dots x_k)_{k\in\mathbb{N}}$ is a subsequence of $(a_n)_{n\in\mathbb{N}}$.

Proof. Let's build this sequence recursively:

- Among all elements of Σ, there is an element which is prefix of infinitely many elements of $(a_n)_{n\in\mathbb{N}}$. Let's call it x_0. By hypothesis, $x_0 \in (a_n)_{n\in\mathbb{N}}$.
- Suppose we have $x_0, \dots, x_N$ such that $(x_0 \dots x_k)_{k\in[\![0,N]\!]}$ is a finite subsequence of $(a_n)_{n\in\mathbb{N}}$. Without loss of generality, we can restrict $(a_n)_{n\in\mathbb{N}}$ to its subsequence composed of the elements $(x_0 \dots x_k)$, $k \in [\![0, N]\!]$ and all elements of $(a_n)_{n\in\mathbb{N}}$ with prefix $x_0 \dots x_N$. This new sequence is still infinite, and has the prefix property 1.

  Among all elements of Σ, there is an element x such that $x_0 \dots x_N x$ is prefix of infinitely many elements of $(a_n)_{n\in\mathbb{N}}$. Let $x = x_{N+1}$. By hypothesis, $x_{N+1} \in (a_n)_{n\in\mathbb{N}}$.
- $(x_k)_{k\in\mathbb{N}}$ has the property we seek.

□

Theorem 4.2. *For all $p \in (0, 1]$, if there is no looping chain of rules of probability 1 in the grammar, there are finitely many partial derivation trees (PDT) of probability $\geq p$.*

Proof. A partial derivation tree is exactly defined by the string of rules deriving it. So a PDT will here be seen, when convenient, as a string of rules.

Suppose there were infinitely many PDT of probability $\geq p$. Then we have a sequence of strings of rules as defined by lemma 4.2. The lemma then gives us a sequence $A_0, \dots, A_n, \dots$ of rules such that $\forall n\ A_0 \dots A_n$ defines a correct partial derivation tree (since $A_0 \dots A_n$ is in the initial sequence, composed of correct PDTs).

To this sequence of rules corresponds an infinite sequence of growing PDT, all of which have probability $\geq p$. Since the sequence is growing and infinite, there is an infinite path in the limit of the trees, given by the sequence of rules $(A_{\varphi(n)})_{n\in\mathbb{N}}$.

Or, differently put, there is a sequence of rules $(A_{\varphi(n)})_{n\in\mathbb{N}}$, such that for all $n \geq 1$, the left side of A_n is in the right side of A_{n-1}.

Then there is a finite sequence $A_{\varphi(k)}, \dots, A_{\varphi(k')}$ such that

$$(A_{\varphi(n)})_{n\in\mathbb{N}} = A_{\varphi(0)}, A_{\varphi(1)}, \dots, A_{\varphi(k-1)}, (A_{\varphi(k)}, \dots, A_{\varphi(k')})^{\mathbb{N}}. \tag{2}$$

Indeed, let $\mathcal{A}$ be the finite state automaton with:

- states $A_{\varphi(n)}, n \in \mathbb{N}$ and END,
- starting state $A_{\varphi(0)}$, ending state END,

- transitions rules $A_{\varphi(n)} \longrightarrow A_{\varphi(n+1)}$ and $A_{\varphi(n)} \longrightarrow END$ for all n.

This automaton generates exactly all prefixes of $(A_{\varphi(n)})_{n\in\mathbb{N}}$. By the pumping lemma, there is a string $A_{\varphi(0)}A_{\varphi(1)}\dots A_{\varphi(N)}$ and two integers k, k' such that $A_{\varphi(0)}A_{\varphi(1)}\dots A_{\varphi(k-1)}(A_{\varphi(k)}\dots A_{\varphi(k')})^*A_{\varphi(k'+1)}\dots A_{\varphi(N)}$ is in the language recognised by $\mathcal{A}$, that is, prefixes of $(A_{\varphi(n)})_{n\in\mathbb{N}}$. This is exactly 2.

Let $p' = \prod_k^{k'} \mathbb{P}(A_{\varphi(i)})$. Then for all $n \in \mathbb{N}$, p'^n is an upper bound of the probability of some PDT of probability $\geq p$ (take any PDT where $(A_{\varphi(k)}, \dots, A_{\varphi(k')})^n$ is in the sequence of its rules). Thus $p'^n \geq p > 0\ \forall n$, which is impossible unless $p' = 1$. This contradicts the hypothesis that no looping chain of rules in the grammar has probability 1. □

Corollary 4.1. *If there is no looping chain of rules of probability* 1 *in the grammar, the parser is complete.*

Proof. If there is no looping chain of rules of probability 1, then Theorem 4.2 holds.

Let $A_0, \dots, A_n$ be the sequence of rules giving the parse of the parsed string. We'll show that for all $k \in [\![0, n]\!]$, the parse will have after finitely many steps $A_0 \dots A_k$ as its top-ranked hypothesis.

Proof. Let $p_k = \prod_{i=0}^{k} \mathbb{P}(A_i)$. Let's prove the result by recursion on k:

- for $k = 0$, A_0 is an axiom so is in the priority queue from the beginning of the parse. By theorem 4.2, there are finitely many PDTs of probability greater than p_0, say N. Since the parser will have each of them at most once as its top-ranked hypothesis, A_0 will be the top-ranked hypothesis before step $N + 1$.
- suppose that after M steps, $A_0 \dots A_k$ is the top-ranked hypothesis of the parser. Then at the $(M + 1)^{th}$ step, the parser will expand $A_0 \dots A_k$, and put (among others) $A_0 \dots A_k A_{k+1}$ in the priority queue. Since, by theorem 4.2, there are finitely many PDTs of probability greater than p_{k+1}, say N, and the parser will have each of them at most once as its top-ranked hypothesis, $A_0 \dots A_k A_{k+1}$ will be the top-ranked hypothesis before step $M + 1 + N + 1$.

  This completes the recursion.

□

The result follows from the case $k = n$. □

5 Conditioning the rules with the CTW algorithm

In this section we present a way to improve the performances of the parser, using the Context Tree Weighting (CTW) algorithm, whose description and properties may be found in [WST95] and an implementation in [FP]. The algorithm is originally intended to be used in data compression, but its construction of context trees allows us to use it in our parser.

The CTW algorithm uses a double mixture of context trees, and its force resides in the fact that conditional probabilities may be computed recursively, allowing for a great decrease in compuation time. Its idea is to mix together the Krichevski-Trofimov estimators for all possible context trees of depth less than a certain M, allowing for a near-optimal coding (and as such, estimate of conditional probabilities, in the sense of the Kullback-Leibler divergence).

5.1 Quick overview of the CTW

The setting is the following : consider a stationary ergodic source of unknown law $\mathbb{P}$. We want to estimate $\mathbb{P}$ by $\hat{\mathbb{P}}$, which will be a mixture of context tree laws.

5.1.1 Sources with a context tree

Definition 5.1. *A complete prefix dictionnary $\mathcal{D}$ on $\mathcal{X}$ (of cardinal K) is a cut of $\mathcal{X}^*$ (seen as a tree), that is, a finite subpart of $\mathcal{X}^*$ such that for all $x_{-\infty:-1}$, there is a unique m such that $x_{-m:-1} \in \mathcal{D}$. Let's call f its context function, defined by $f(x_{-\infty:-1}) = x_{-m:-1}$. Its depth, noted $l(\mathcal{D})$, is the maximum length of its elements (or, more simply, the depth of the cut).*

Suppose that we have reasons to think that $\mathbb{P}$ has a context tree $\mathcal{D}$ (or at least, can be succesfully approximated by such a law), that is, $\mathbb{P}$ is stationary and for all $x_{-\infty:n}$, $\mathbb{P}(X_n = x_n | X_{-\infty:n-1} = x_{-\infty:n-1}) = \mathbb{P}(X_n = x_n | f(X_{-\infty:n-1}) = f(x_{-\infty:n-1}))$.

We then want to estimate the conditional probabilities for all contexts in $\mathcal{D}$. Suppose that, for a context s, θ^s is the law on $\mathcal{X}$, conditionaly on the context s. Then, for a source with context $\mathcal{D}$, of parameter $(\theta^s)_{s\in\mathcal{D}}$,

$$\begin{aligned}\mathbb{P}_{\mathcal{D},\theta}(X_{1:n} = x_{1:n} | X_{-\infty:0} = x_{-\infty:0}) &= \prod_{i=1}^{n} \mathbb{P}_{\mathcal{D},\theta}(X_{1:n} = x_{1:n} | f(X_{-\infty:0}) = f(x_{-\infty:0})) \\ &= \prod_{s\in\mathcal{D}} \mathbb{P}_{\theta^s}(S_s(x_{1:n}; x_{-\infty:0}))\end{aligned}$$

where $\mathbb{P}_{\theta^s}$ is the law of a string of i.i.d. variables of law θ^s, and $S_s(x_{1:n}; x_{-\infty:0})$ is the substring of symbols in $x_{1:n}$ with context s.

The idea is to mix all those probabilities for all θ^s : for a prior distribution $\nu_{\mathcal{D}}(d\theta) = \prod_{s\in\mathcal{D}} \nu(d\theta^s)$, where ν is a measure on the set of possible θ^s, the simplex $\Theta = \{(\theta_1, \ldots, \theta_K) \in [0,1]^K | \sum \theta_i = 1\}$, we get:

$$\begin{aligned}KT_{\mathcal{D},\nu}(x_{1:n} | x_{-\infty:0}) &= \int_{\Theta^{\mathcal{D}}} \mathbb{P}_{\mathcal{D},\theta}(x_{1:n} | x_{-\infty:0}) \\ &= \prod_{s\in\mathcal{D}} \int_{\Theta} \mathbb{P}_{\theta^s}(S_s(x_{1:n}; x_{-\infty:0})) \nu(d\theta^s)\end{aligned}$$

A good choice for ν is a Dirichlet $\mathbb{D}(1/2, \ldots, 1/2)$ distribution: the Dirichlet Law with parameter $\alpha = (\alpha_1, \ldots, \alpha_K)$ is the law on Θ with density

$$f(\theta_1, \ldots, \theta_K) = \frac{\Gamma(\alpha_1 + \ldots + \alpha_K)}{\Gamma(\alpha_1) \ldots \Gamma(\alpha_K)} \prod_{i=1}^{K} \theta_i^{\alpha_i}$$

with respect to the Lebesgue density.

This distribution has nice properties, for example, an oracle inequality for the code-length. This choice gives the *Krichevski-Trofimov* estimator $KT_{\mathcal{D}} = KT_{\mathcal{D},\mathbb{D}(1/2,\ldots,1/2)}$ for sources with a context tree.

Lemma 5.1. *Let*

- $c_s^y(x_{1:n}|x_{-\infty:0})$ *denote the number of* y *in context* s,
- $C_s(x_{1:n}|x_{-\infty:0}) = \sum_y c_s^y(x_{1:n}|x_{-\infty:0})$, *the number of occurence of context* s.

Then:

$$KT_{\mathcal{D}}(x_{1:n}|x_{-\infty:0}) = \prod_{s\in\mathcal{D}} \frac{\Gamma(K/2)}{\Gamma(1/2)^K} \frac{\prod_{y\in\mathcal{X}} \Gamma(c_s^y(x_{1:n}|x_{-\infty:0}) + 1/2)}{\Gamma(C_s(x_{1:n}|x_{-\infty:0}) + 1/2)}$$

These quantities may be recursively computed by the following lemma (for a binary alphabet):

Lemma 5.2. *Let* $P_e(a,b)$ *denote the K-T estimator (giving the probability of having a* 0*) for a particular context* s, a *(resp* b*) representing the number of 0 (resp 1) seen in context* s. *Then* $P_e(0,0) = 1$, *and for* $a \geq 0, b \geq 0$,

$$P_e(a+1,b) = \frac{a+1/2}{a+b+1} P_e(a,b) \qquad \text{and} \qquad P_e(a,b+1) = \frac{b+1/2}{a+b+1} P_e(a,b).$$

The proof will not be presented here, but is quite easy, and may be found in [WST95]. The case of an alphabet of size K is identical, but much longer to write...

5.1.2 The Context Tree Weighting Method

When the context tree of our source is unknown, one solution is to mix over all possible context trees (of a certain maximum depth). The Context Tree Weighting methods consists in this idea: if π is a probability on context trees, we take

$$CTW(x_{1:n}) = \sum_{\mathcal{D}} \pi(\mathcal{D}) KT_{\mathcal{D}}(x_{1:n})$$

Typically, we will take for π a branching law, in which all nodes of depth less than a certain M has probability $\alpha \leq 1/K$ of having K daughters.

An important result is that this method is *universal*:

Theorem 5.1. *If* $(X_n)_{n\in\mathbb{N}}$ *is ergodic, stationary of law* $\mathbb{P}$, *then,* $\mathbb{P}$*-a.s.,*

$$\lim_{n\to\infty} -\frac{1}{n} \log CTW_\alpha(X_{1:n}) = H(\mathbb{P})$$

where $H(\mathbb{P})$ *is the entropy of* $\mathbb{P}$.

The great advantage of this method is that it may be recursively computed.

We will now present quickly how this can be done.

Let us denote, for each context s and $x \in \mathcal{X}$, x_s as the number of x seen in context s, and $P_e((x_s)_{x\in\mathcal{X}}, y)$ the corresponding K-T estimator (that is, the probability of having $y \in \mathcal{X}$ in context s).

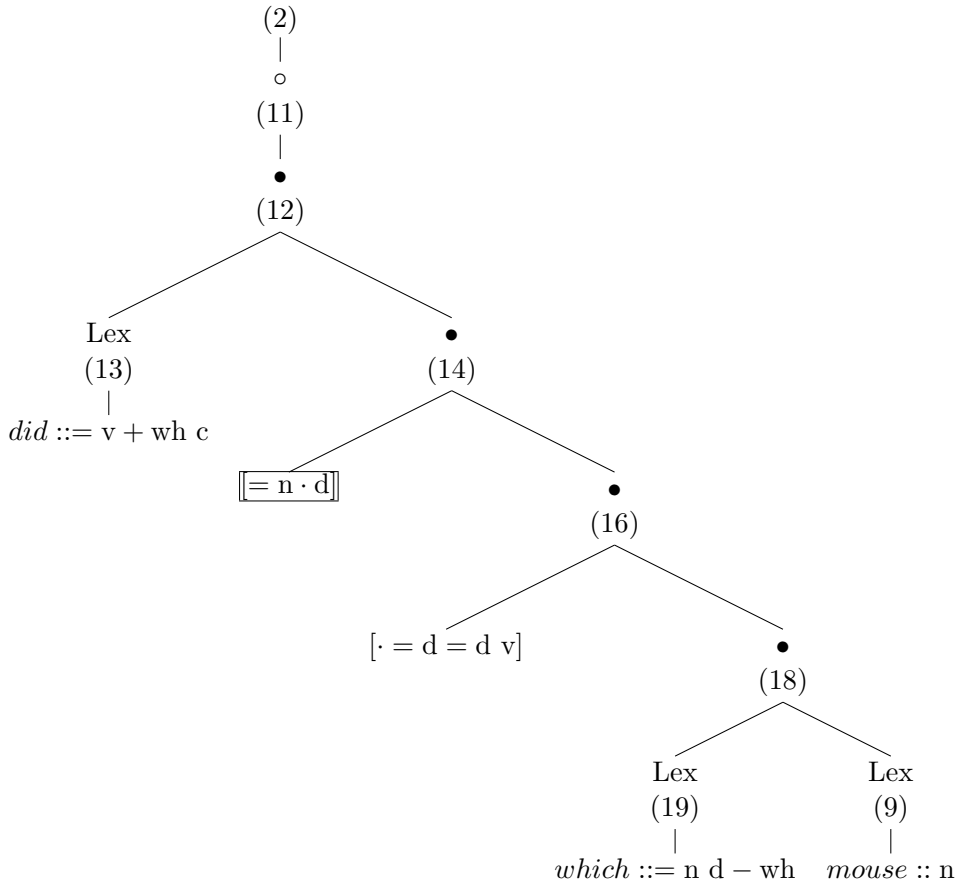
Definition 5.2. *To each node* s *of the context tree* $\mathcal{T}$ *of depth* M, *we assign a weighted probability* $\mathbb{P}_w^s$ *which is defined as*

$$\mathbb{P}_w^s(y) = \begin{cases} \frac{(K-1)P_e((x_s),y) + \prod_{\varphi\in\mathcal{X}} \mathbb{P}_w^{\varphi s}(y)}{K} & \text{for } 0 \leq l(s) < M \\ P_e((x_s), y) & \text{otherwise} \end{cases}$$

This construction has the expected property, that is, $\mathbb{P}_w^s(.) = CTW(.|s)$, for the $\alpha = 1/K$ mixture.

5.2 Application for the parser

This algorithm may be applied to our parser, to condition the rewriting rules. We can, with this method, condition on whatever we want, provided it appears as a linear string of symbols. An obvious (and perhaps a bit naïve) choice would be to condition on the string of rules already used in the parse. Another would be to condition on the string of rules *descending directly to the expanded node from the root*. For example, let's see, with our cats and mouses grammar (2) the sentence 'Which mouse did the cat eat' (3), when the parser will try to expand the node giving 'the cat' (at point 'Which mouse did · the cat eat'): it will be for example in state (index of the rules indicated in (.))



and will have already constructed the following string of rules: 2-11-12-14-16-18-19-9-13. It will condition on the string of rules 2-11-12-14, meaning that:

- it's the subject of a VP with moving object (14),
- it's a past sentence (12),
- it's an interrogative sentence (11 and 2)

Indeed, only the first seems relevant, but using all string of rules will condition first by the fact that it is a past sentence, that a mouse is involved, etc... and will have to go up 6 rules to know that it is a subject that is currently expanded... which seems the important information. This conditioning allows to condition roughly on the successive heads c-commanding the expanded node (plus movement information, which is difficult to get rid of). Of course we could want a different

method for lexicalisation rules, where thematic and semantic information would be better... but such a method is difficult to implement, having to insure that conditioning still gives a proper distribution.

Three points seem important to precise:

- First, although the CTW algorithm works on any finite alphabet, it is much more efficient on binary ones. Ron Begleiter and Ran El-Yaniv discussed a method in [BEY06] to make the algorithm binary even in the case of an non-binary alphabet, by putting chains of CTWs (i.e. sorting the rules in a binary tree, and having a CTW algorithm for each branchement).
- Second, the distribution, as can be seen in the previous examples, is very sparse... a given context can be followed by two or three different rules, even one, while the alphabet is huge, with 23 rules in the cats and mouses grammar (which is very simple). Of course, more complete grammar will induce a lot more variability in the possible rules for a given node, but the number of rules will grow too. However, the restriction of rules expanding a given node can be implemented directly in the structure of the context tree of the CTW, provided we know in advance the grammar -which is of course the case.
- Finally, it is possible that, although the *grammar* allows for a choice of rewriting rules for a given category, there is in fact no such choice (or with vanishing probability). Then it is possible to use a slightly different base estimator instead of the Krichevski-Trofimov one: the zero-redundancy estimator $P_e^{ZR}(a,b)$ defined as:

$$P_e^{ZR}(a,b) = \begin{cases} \frac{1}{2}P_e(a,b) & \text{for } a>0, b>0 \\ \frac{1}{2}P_e(a,0)+\frac{1}{4} & \text{for } a>0, b=0 \\ \frac{1}{2}P_e(0,b)+\frac{1}{4} & \text{for } a=0, b>0 \\ 1 & \text{for } a=b=0 \end{cases}$$

This estimator better recognises sources generating only 0s or only 1s.

Conclusion

The method described here permits to see Minimalist Grammars as the more 'classical' and above all simpler Linear Context-Free Rewriting Systems (which don't have movement, and generate sentences *top-down*), by taking a different point of view - considering derivation trees instead of derived trees. This enabled us to easily put a probability field on these grammars, and to parse them in a *top-down, incremental way*, giving a progressive parse as the words of the sentence are discovered. The probability field allowed us to implement a *beam-search* in the parser, pruning the different hypothesis to select only the more likely ones. This should accelerate the parser, while making it fail in identifying 'garden-path'-type sentences. The use of more refined probabilistic tools as the CTW algorithm permits to have a better estimation of the real probability field, by conditionning the expanding rules by its context - here, the nature of the c-commanding heads, as required by the current linguistic theories.

References

[BEY06] Ron Begleiter and Ran El-Yaniv. Superior guarantees for sequential prediction and lossless compression via alphabet decomposition. *J. Mach. Learn. Res.*, 7:379–411, 2006.

[Cho95] Noam Chomsky. The minimalist program. *MIT Press, Cambridge, Massachusetts*, 1995.

[FP] Erik Franken and Marcel Peeters. Overview of the context tree weighting version 0.1 implementation. *"http://www.ele.tue.nl/ctw/download/ctw-v01_manual.pdf"*.

[Har01] Hendrik Harkema. Parsing minimalist grammars. 2001.

[Roa97] Brian Roark. Probabilistic top-down parsing and language modeling. *Logical aspects of computational linguistics*, 1997.

[Sta97] Edward Stabler. Derivational minimalism. *Logical aspects of computational linguistics*, 1997.

[WST95] Frans M.J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context tree weighting method : basic properties. *IEEE-IT*, 41(3), 1995.

Troisième partie

Exposé de maîtrise

Mémoire de maîtrise
Longs chemins auto-évitants

Thomas MAINGUY et Robin STEPHENSON
Sujet proposé par Wendelin WERNER

Résumé

Nous étudions certaines propriétés des longs chemins auto-évitants aléatoires dans $\mathbb{Z}^d$ choisis 'uniformément'.

Dans un premier temps, on étudie le nombre de chemins auto-évitants à N pas, et on améliore la simple approximation géométrique du nombre de chemins à N pas. On met ensuite en évidence une loi de probabilité et un phénomène de renouvellement pour une certaine catégorie importante de tels chemins.

Enfin, après avoir démontré un théorème de Kesten sur l'apparition de certains motifs dans les longs chemins, et l'avoir appliqué pour améliorer l'estimation de la première section, nous établissons une loi de probabilité sur les chemins auto-évitants de longueur infinie dans un demi-espace de deux manières différentes.

Table des matières

Introduction et considérations générales

Dans un graphe infini, on s'intéresse à une classe particulière de marches : celles qui ne se recoupent jamais. On peut vouloir obtenir des informations sur le nombre de tels chemins, et sur leur structure. Une catégorie importante de graphes dans ce domaine est celle des réseaux de $\mathbb{R}^d$, et nous nous restreindrons dans cette étude au réseau hypercubique $\mathbb{Z}^d$.

Les résultats que nous établirons seront de deux types : des résultats de nature combinatoire et d'autres de nature probabiliste. Les premiers nous renseigneront sur le cardinal de certains ensembles de chemins auto-évitants. Outre établir la croissance quasi-géométrique du nombre de chemins auto-évitants à N pas, et quelques autres résultats dignes d'intérêt, ils permettrons de construire une loi sur les chemins auto-évitants dans le demi-espace droit, et de faire apparaître un phénomène de renouvellement, qui a lieu lorsque le chemin atteint pour la dernière fois une abscisse.

0.1 Premières définitions

Définition 0.1. *Un* chemin *est une suite finie $\omega = (\omega(0), \dots, \omega(N))$ de plus proches voisins dans $\mathbb{Z}^d$. On dit qu'il est* auto-évitant *si la suite est de plus injective. On note $\mathcal{C}$ l'ensemble des* chemins auto-évitants*, considérés à translations près. Par commodité, on prendra $\omega(0) = 0$.*

Pour $\omega = (\omega(0), \dots, \omega(N))$, on note $|\omega| = N$ sa longueur, ou encore son nombre de pas, et on définit son étendue $A = \max_{i,j} \omega_1(i) - \omega_1(j)$, où on note y_i la i-ième coordonnée de y dans la base canonique de $\mathbb{Z}^d$.

On note c_N le nombre de chemins auto-évitants de longueur N.

On notera aussi $\omega[i, j]$ le chemin $(\omega(i), \dots, \omega(j))$, pour $i \leqslant j$.

Plusieurs classes particulières de ces chemins nous intéresseront à l'avenir :

Définition 0.2. *On note $\mathcal{H}$ l'ensemble des chemins auto-évitants qui restent dans le demi-espace $\{x_1 > 0\}$ à partir de leur premier pas, et h_N le cardinal des chemins de $\mathcal{H}$ à N pas.*

On dit qu'un chemin auto-évitant à N pas ω est un pont *s'il vérifie*

$$\forall i \in [\![1, N]\!] \quad 0 < \omega_1(i) \leqslant \omega_1(N).$$

On notera $\mathcal{B}$ l'ensemble des ponts et b_N le cardinal de l'ensemble des ponts à N pas.

Définition 0.3. *Si ω et ω' sont deux chemins de longueurs respectives N et N', on peut définir leur* concaténation *$\omega \circ \omega'$ comme le chemin ρ de longueur $N + N'$ vérifiant*

$$\rho(i) = \begin{cases} \omega(i) & \text{pour } i \in [\![0, N]\!] \\ \omega(N) + \omega'(i - N) & \text{pour } i \in [\![N + 1, N + N']\!]. \end{cases}$$

On remarque que, si la concaténation de deux chemins auto-évitants n'en donne pas nécéssairement un troisième, la concaténation de deux ponts en donne toujours un troisième.

Définition 0.4. *On dit qu'un pont ω est irréductible s'il ne peut pas s'écrire comme concaténation de deux ponts de longueur strictement positive. On notera $\mathcal{I}$ l'ensemble des ponts irréductibles et λ_N le nombre de ponts irréductibles à N pas.*

Par convention, il n'existe pas de ponts irréductibles de longueur nulle.

Définition 0.5. *Soit ω un pont à N pas. On dit que $j \in [\![\omega_1(0), \omega_1(N)]\!]$ est un* point de rupture *de ω s'il existe $r \in [\![1, N]\!]$ tel que $\forall i \in [\![0, r]\!]$ $\omega_1(i) \leqslant j$ et $\forall i \in [\![r + 1, N]\!]$ $\omega_1(i) > j$.*

On remarque qu'un pont est irréductible si, et seulement si, il n'a pas de point de rupture autre que l'abscisse de son dernier point.

Proposition 0.1. *Un pont s'écrit de manière unique comme concaténation de ponts irréductibles.*

Démonstration. Soit ω un pont non irréductible, soit i le plus grand entier tel que $\omega_1(i)$ soit le plus petit point de rupture de ω. Alors $\eta^{[1]} = (\omega(0), \dots, \omega(i))$ est un pont irréductible, le reste est un pont, de longueur strictement plus petite. C'est le premier terme de la décomposition.

L'unicité de $\eta^{[1]}$ provient de l'observation suivante : Étant donné un pont, il existe un unique pont irréductible le commençant, tel que le reste soit encore un pont (éventuellement un point) : l'abscisse du dernier point du pont irréductible doit être le premier point de rupture, et le pont ne doit plus y retourner, ce qui donne la maximalité de i dans la construction précédente.

On conclut par récurrence. □

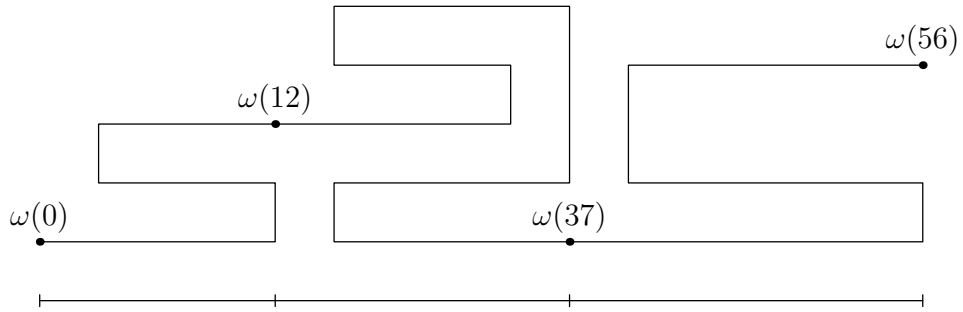


FIG. 1 – Un pont et sa décomposition en ponts irréductibles. Les points indiquent les limites des ponts irréductibles, les traits les points de rupture.

Définition 0.6. *Si $\mathcal{A}$ est un ensemble de chemins, on note pour $N \in \mathbb{N}$, $\mathcal{A}_N$ l'ensemble des chemins dans $\mathcal{A}$ à N pas.*

0.2 La constante de connectivité

On cherche à comprendre l'accroissement du nombre de chemins auto-évitants avec la longueur. Comme on a, pour tout N, l'encadrement $d^N \leqslant c_N \leqslant 2d(2d-1)^{N-1}$ (car, pour obtenir un chemin auto-évitant, on peut se limiter à prendre les coordonnées croissantes, mais les retours immédiats sont interdits), on peut penser que le comportement de c_N est grosso modo géométrique. On a en fait le résultat suivant :

Théorème 0.1 (constante de connectivité)**.** *Il existe un réel strictement positif μ, appelé* constante de connectivité*, tel que*

$$c_N^{1/N} \underset{N\to\infty}{\longrightarrow} \mu.$$

De plus, pour tout N, $c_N \geqslant \mu^N$.

C'est une conséquence directe du lemme suivant

Lemme 0.1 (sous-additivité)**.** *Soit $(u_n)_{n\geqslant 1}$ une suite réelle sous-additive, c'est à dire telle que*

$$\forall n, m \geqslant 1,\ u_{n+m} \leqslant u_n + u_m.$$

Alors la suite $(u_n/n)_{n\geqslant 1}$ converge et

$$\frac{u_n}{n} \underset{n\to\infty}{\longrightarrow} \inf_{k\geqslant 1} \frac{u_k}{k}.$$

Démonstration. On a les inégalités évidentes :

$$\inf_{n\geqslant 1}\frac{u_n}{n}\leqslant \liminf_{n\to\infty}\frac{u_n}{n}\leqslant \limsup_{n\to\infty}\frac{u_n}{n}.$$

Il suffit donc de montrer que $\limsup_{n\to\infty}(u_n/n)\leqslant \inf_{n\geqslant 1}(u_n/n)$. Soient alors n et p deux entiers. Posons

$$M=\max_{r\in[\![0,p-1]\!]}u_r.$$

Soient q et r respectivement le quotient et le reste de la division euclidienne de n par p, on a par une récurrence évidente que $u_n\leqslant qu_p+u_r$. Comme $q\leqslant n/p$, on a

$$\frac{u_n}{n}\leqslant q\frac{u_p}{n}+\frac{u_r}{n}\leqslant\frac{u_p}{p}+\frac{M}{n}.$$

En prenant la limite supérieure, on obtient :

$$\limsup_{n\to\infty}\frac{u_n}{n}\leqslant\frac{u_p}{p}.$$

Ceci étant vrai pour tout $p\geqslant 1$, on a alors l'inégalité voulue. □

Remarquant de plus que, pour tous N et M, tout chemin de longueur $N+M$ s'écrit comme la concaténation d'un chemin de longeur N et d'un chemin de longueur M, si bien que $c_{N+M}\leqslant c_Nc_M$. En passant au logarithme, on obtient l'existence de la constante de connectivité μ, qui vérifie d'ailleurs $d\leqslant\mu\leqslant 2d-1$ et $c_N\geqslant\mu^N$, puisque le lemme nous affirme que la limite est en réalité l'infimum.

Remarques : La concaténation de deux ponts étant toujours un pont, la suite $(b_N^{1/N})_{N\in\mathbb{N}}$ converge vers une 'constante de connectivité des ponts' μ_B, vérifiant par 'sur-multiplicativité' $b_N\leqslant\mu_B^N$. Cette constante est trivialement inférieure à μ puisqu'un pont est un chemin auto-évitant. On verra qu'il y a en fait égalité.

Ceci ne nous donne cependant pas énormément d'informations sur le comportement asymptotique de c_N. Nous montrerons plus tard que $c_{N+2}/c_N\longrightarrow\mu^2$.

Il est également conjecturé que

$$c_N\sim C\mu^N N^{\gamma-1}.$$

Ceci a été démontré en dimension supérieure à 5, avec $\gamma=1$.

0.3 Fonctions génératrices

Un certain nombre de raisonnements dans la suite s'appuient sur l'utilisation des séries génératrices. Introduisons-en donc certaines particulièrement importantes :

Définition 0.7. *On définit la* fonction génératrice χ *des chemins auto-évitants par*

$$\chi(z)=\sum_{N\in\mathbb{N}}c_Nz^N=\sum_{\omega}z^{|\omega|}.$$

On définit de même H_z, B_z et Λ_z les fonctions génératrices des chemins dans un demi-espace, ponts, et ponts irréductibles respectivement.

On remarquera que, les coefficients des séries étant toujours positifs, quand z tend vers le rayon de convergence de la série par valeurs inférieures, la valeur de la série en z converge vers sa valeur en son rayon de convergence, éventuellement infini (convergence monotone).

La section précédente, avec $\lim_{N\to\infty}c_N^{1/N}=\mu$ nous donne le rayon de convergence de χ : c'est $z_c=\mu^{-1}$.

De plus, comme $c_N\geqslant\mu^N$, on a, pour $z\in[0,z_c)$, $\chi(z)\geqslant 1/(1-\mu z)$. En particulier, $\chi(z_c)=+\infty$.

1 Une borne et une de ses conséquences

La sous-additivité nous dit seulement que $c_N = \mu^N \exp(o(N))$. Dans cette section, nous établissons une majoration de $c_N \mu^{-N}$ en $\exp(O(\sqrt{N}))$, ce qui est la meilleure majoration connue en dimension 2. Nous en tirons ensuite des conséquences très importantes sur les ponts.

1.1 Une borne pour $c_N \mu^{-N}$

Théorème 1.1. *Pour tout $B > \pi\sqrt{2/3}$, il existe $N_0 \geqslant 1$ tel que*

$$\forall N \geqslant N_0,\ c_N \leqslant \mu^{N+1} \mathrm{e}^{B\sqrt{N}}.$$

Pour démontrer ce théorème, on a d'abord besoin de quelques résultats préliminaires.

Définition 1.1. *Soit N un entier strictement positif. Une* partition *de N est la donnée d'une suite finie d'entiers $N_1 > N_2 > \cdots > N_k > 0$ telle que $N_1 + \cdots + N_k = N$. On note P(N) le cardinal de l'ensemble des partitions de N.*

Remarque : Une telle partition de N est équivalente à la donnée d'une suite stricement croissante $(a_i)_{i \in [\![1,k]\!]}$ telle que $a_k = N$ et $a_1 > a_2 - a_1 > \cdots > a_k - a_{k-1}$ par la bijection $(N_j)_j \mapsto (\sum_{i=1}^{j} N_i)_j$.

Nous admettrons le résultat suivant de combinatoire.

Théorème 1.2. *On a le comportement asymptotique suivant :*

$$\log P(N) \sim \pi \sqrt{\frac{N}{3}} \quad \textit{quand } N \to \infty.$$

Ce résultat est ici admis, une démonstration peut être trouvée dans [HR17].

Proposition 1.1. *Pour tout $N \geqslant 1$, on a $h_N \leqslant P(N) b_N$.*

Démonstration. Soient $N \geqslant 1$ et $\omega \in \mathcal{H}_N$. Nous allons transformer ω en un pont par la méthode suivante, aussi décrite dans la figure 2 : soient

$$L = \max\{\omega_1(j) : i \in [\![1, N]\!]\} \quad \text{et} \quad i_0 = \max\{i \in [\![1, N]\!] : \omega_1(i) = L\}.$$

Considérons ensuite le chemin $\omega^{(1)}$ qui commence comme ω mais tel que, à partir de i_0, $\omega^{(1)}$ est la réflection de ω par rapport à l'hyperplan $x_1 = L$. Autrement dit :

$$\omega^{(1)}(i) = \begin{cases} \omega(i) & \text{si } i \leqslant i_0, \\ (2L - \omega_1(i), \omega_2(i), \ldots, \omega_3(i)) & \text{si } i \geqslant i_0. \end{cases}$$

Le choix de L fait que $\omega^{(1)}$ est encore auto-évitant et est aussi clairement un élément de $\mathcal{H}_N$. Si on réitère ce procédé, on obtient un entier i_1 et un chemin $\omega^{(2)}$ qui, à partir de i_1, est la réflection de $\omega^{(1)}$ par rapport à l'hyperplan $x_1 = \omega^{(1)}(i_1)$. En continuant, on obtient une suite finie strictement croissante d'entiers $(i_k)_{k \in [\![0,p]\!]}$ et des chemins $(\omega^{(k)})_{k \in [\![0,p]\!]}$. Remarquons alors que $i_0 \neq 0$, $i_p = N$ (sinon on pourrait réitérer le procédé), que $\omega^{(p)}$ est un pont (précisément car $i_p = N$) et que $\omega_1(i_1) > |\omega_1(i_2) - \omega_1(i_1)| > \cdots > |\omega_1(i_p) - \omega_1(i_{p-1})| > 0$ grâce au choix de L à chaque étape.

La donnée du pont final $\omega^{(p)}$ et des $|\omega_1(i_j) - \omega_1(i_{j-1})|$, qui forment une partition de l'étendue de $\omega^{(p)}$ (qui est inférieure à N), caractérisent injectivement ω. Si on note $b_{N,A}$ le nombre de ponts à N pas d'étendue A, alors

$$h_N \leqslant \sum_{A=1}^{N} P(A) b_{N,A} \leqslant P(N) \sum_{A=1}^{N} b_{N,A} \leqslant P(N) b_N. \qquad \square$$

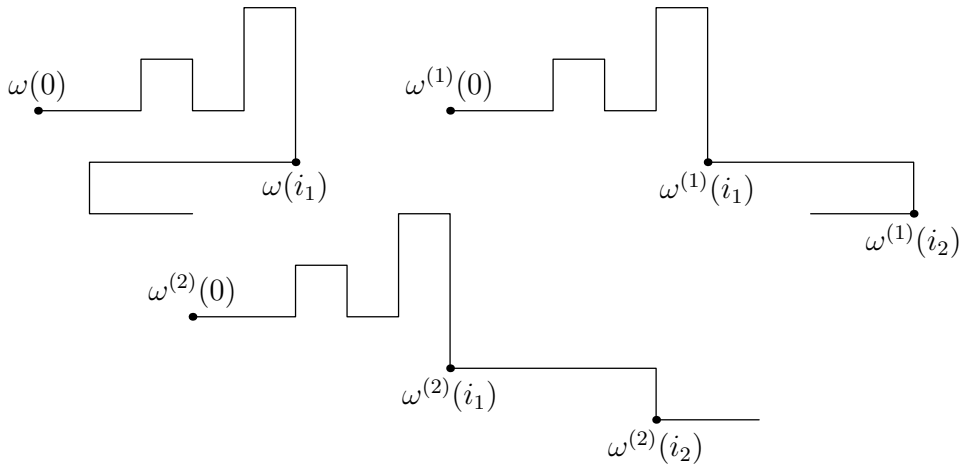


FIG. 2 – Illustration de la preuve de la proposition 1.1

Lemme 1.1. *Pour tout $n \geqslant 1$, on a $c_n \leqslant \sum_{m=0}^{n} h_{n-m} h_{m+1}$.*

Démonstration. soit ω un chemin auto-évitant de longeur $n \geqslant 1$. Considérons $M = \min_i \omega_1(i)$ et m le plus grand i tel que $\omega_1(i) = M$. Les deux chemins $(\omega(m), \ldots, \omega(n))$ et $(\omega(m) - e_1, \omega(m), \omega(m-1), \ldots, \omega(0))$ sont clairement dans $\mathcal{H}_{n-m}$ et $\mathcal{H}_{m+1}$ respectivement, et déterminent complètement ω. □

Preuve du théorème. Soit $B > \pi\sqrt{2/3}$, soit $\varepsilon > 0$ tel que $\varepsilon < B - \pi\sqrt{2/3}$. Prenons un réel K tel que

$$\forall n \geqslant 1,\ P(n) \leqslant K \exp\left[(B-\varepsilon)\sqrt{n/2}\right],$$

qui existe bien par le théorème 1.2.

Ceci, avec le lemme et la proposition précédente, le fait que $\forall i, j,\ b_i b_j \leqslant b_{i+j}$ et l'inégalité $\forall x, y \geqslant 0, \sqrt{x} + \sqrt{y} \leqslant \sqrt{2x+2y}$, donne, pour $n \geqslant 1$:

$$\begin{aligned}
c_n &\leqslant \sum_{m=0}^{n} h_{n-m} h_{m+1} \\
&\leqslant \sum_{m=0}^{n} b_{m+1} b_{n-m} P(m+1) P(n-m) \\
&\leqslant K^2\, b_{n+1} \sum_{m=0}^{n} \exp\left((B-\varepsilon)\left[\sqrt{\frac{m+1}{2}} + \sqrt{\frac{n-m}{2}}\right]\right) \\
&\leqslant b_{n+1}(n+1)K^2\, \exp[(B-\varepsilon)\sqrt{n+1}].
\end{aligned}$$

Enfin, comme $b_{n+1} \leqslant \mu^{n+1}$ et en utilisant la marge donnée par le ε, on trouve donc un N_0 tel que

$$\forall n \geqslant N_0, \qquad c_n \leqslant \mu^{n+1} \exp B\sqrt{n}.$$

□

1.2 Une conséquence sur les ponts

Proposition 1.2. *On a :*

$$b_N^{1/N} \to \mu.$$

Ceci veut dire que la 'constante de connectivité des ponts' est en fait égale à μ.

Ce résultat peut être amélioré. En effet, on sait maintenant que le rayon de convergence de la fonction caractéristique des ponts $B_z = \sum_{N=0}^{\infty} b_N z^N$ est $z_c = \mu^{-1}$. En fait, cette série diverge au point critique, c'est-à-dire qu'on a

$$B_{z_c} = \sum_{N=0}^{\infty} b_N \mu^{-N} = +\infty.$$

Démonstration. Le fait que $b_N^{1/N} \to \mu$ découle du théorème précédent, car, pour N assez grand, on a

$$\mu^{N-1} \mathrm{e}^{-B\sqrt{N}} \leqslant b_N \leqslant \mu^N.$$

(Voir la dernière ligne des inégalités de la preuve précédente, et $c_N \geqslant \mu^N$)

Pour montrer la divergence de la série, rappelons-nous la démonstration de la proposition 1.1 grâce à laquelle on peut voir qu'un chemin dans un demi-espace est caractérisé par la donnée d'une suite finie de ponts, dont l'étendue est strictement décroissante. Plus précisément, si on note $b_{m,A}$ le nombre de ponts de longueur m et d'étendue A, on a, pour $N \geqslant 1$:

$$h_N \leqslant \sum_k \left(\prod_{i=1}^{k} b_{m_i, A_i} \right),$$

où la somme se fait sur tous les entiers $k \geqslant 1$, les suites $(A_i)_{i \in [\![1,k]\!]}$ strictement positives et strictement décroissantes et les suites $(m_i)_{i \in [\![1,k]\!]}$ strictement positives dont la somme vaut N. Alors, pour $z > 0$, on a :

$$\sum_{N=0}^{\infty} h_N z^N \leqslant \prod_{A=1}^{\infty} \left(1 + \sum_{m=1}^{\infty} b_{m,A} z^m \right).$$

Cela se voit en comparant les termes de même degré de chaque côté. En combinant ceci et l'inégalité classique $1 + x \leqslant \mathrm{e}^x$ pour $x \geqslant 0$, on obtient :

$$\sum_{N=0}^{\infty} h_N z^N \leqslant \exp\left(\sum_{A=1}^{\infty} \sum_{m=1}^{\infty} b_{m,A} z^m \right) = \exp(B_z - 1).$$

Combiner ceci avec le lemme 1.1 permet d'obtenir, par produit de Cauchy :

$$\sum_{N=0}^{\infty} c_N z^N \leqslant z^{-1} \left(\sum_{N=0}^{\infty} h_N z^N \right)^2 \leqslant z^{-1} e^{2(B_z - 1)}.$$

On sait que le terme de gauche diverge quand $z = z_c$, donc celui de droite aussi, ce qui conclut la preuve. □

2 Ponts et renouvellement

Dans cette section, on va essayer de construire un loi de probabilité sur les ponts irréductibles, et ce faisant, faire apparaître un phénomène de renouvellement dans la construction des ponts, vus comme concaténation de ponts irréductibles.

Pour ce faire, on a besoin de définir une fonction particulière, la *masse*.

2.1 La masse

Définition 2.1. *Pour $x \in \mathbb{Z}^d$ et N entier, on note $c_N(x)$ le nombre de chemins auto-évitants à N pas commençant en 0 et se terminant en x, et $G_z(x)$ la série génératrice de la suite ainsi définie.*

Proposition 2.1 (décroissance exponentielle de G_z)**.** *Pour tout $z \in (0, z_c)$, il existe des constantes positives D_z et θ_z telles que, pour tout x dans $\mathbb{Z}^d$*

$$G_z(x) \leqslant D_z \mathrm{e}^{-|\log \theta_z| \ \|x\|_\infty}.$$

Démonstration. On a

$$G_z(x) \leqslant \sum_{N \geqslant \|x\|_\infty} c_N z^N.$$

Pour tout $\varepsilon > 0$, il existe K_ε tel que, pour tout $N \in \mathbb{N}$, $c_N \leqslant K_\varepsilon(\mu+\varepsilon)^N$. En sommant, on a

$$G_z(x) \leqslant \sum_{N \geqslant \|x\|_\infty} K_\varepsilon((\mu+\varepsilon)z)^N$$

En choissant ε tel que $(\mu+\varepsilon)z = \theta_z < 1$, on obtient

$$G_z(x) \leqslant K_\varepsilon \frac{\theta_z^{\|x\|_\infty}}{1-\theta_z} \leqslant D_z \mathrm{e}^{-|\log \theta_z| \ \|x\|_\infty},$$

où on a posé $D_z = K_\varepsilon/(1-\theta_z)$.

□

On est ainsi amené à poser la définition suivante :

Définition 2.2. *Pour $z > 0$, on note*

$$m(z) = \liminf_{n \to \infty} \frac{-\log G_z(n, 0, \ldots, 0)}{n}$$

la 'masse', que l'on peut voir comme le taux de décroissance exponentielle de G_z.

Remarquons que, comme $G_z(n, 0, \ldots, 0) \geqslant z^n$ (le chemin direct de $(0, \ldots, 0)$ à $(n, 0, \ldots, 0)$ est un pont, donc $b_n(n, 0, \ldots, 0) \geqslant 1$), on a $m(z) \leqslant -\log z$.

2.2 Théorie du renouvellement sur les ponts

Ici, nous allons construire une loi de probabilité sur les ponts irréductibles, en faisant apparaître une équation de renouvellement sur les ponts, renouvellement qui a lieu lors d'un 'changement de pont irréductible'. Cette loi nous permettra, dans le dernier chapitre, de construire aléatoirement des ponts infinis.

Pour commencer l'étude, on va tout d'abord se placer au point critique, c'est-à-dire en z_c. On va voir qu'alors, on peut définir une loi de probabilité sur les ponts irréductibles ω en leur donnant un poids de $z_c^{|\omega|}$.

Ensuite, on se placera en-dessous du point critique pour étudier plus en détails les points de chute des ponts irréductibles. Le 'poids' des ponts est alors un peu plus complexe, puisqu'un facteur faisant intervenir la masse et l'étendue apparaît.

2.2.1 Au point critique

Un pont auto-évitant est en fait une succession de ponts irréductibles. Les extrémités de ces ponts irréductibles peuvent donc être vues comme des 'points de renouvellement' du pont, en ce sens qu'il ne repasse jamais derrière ce point. La suite consiste à essayer de faire apparaître une équation de renouvellement et à en tirer des conséquences.

On a tout d'abord, l'identité suivante :

$$b_N = \sum_{s=1}^{N} \lambda_s b_{N-s} + \delta_{N,0} \, ,$$

puisque un pont se décompose de manière unique en un pont irréductible, suivi d'un pont. On en déduit que, pour $z \in (0, z_c)$,

$$\begin{aligned} B_z &= \sum_{N \geqslant 0} b_N z^N \\ &= \sum_{N \geqslant 0} \left(\sum_{s=1}^{N} \lambda_s b_{N-s} + \delta_{N,0} \right) z^N \\ &= 1 + \sum_{N \geqslant 0} \sum_{s=1}^{N} \lambda_s b_{N-s} z^N \\ &= 1 + \left(\sum_{k \geqslant 0} b_k z^k \right) \left(\sum_{s=1}^{N} \lambda_s z^s \right) \end{aligned}$$

et donc

$$B_z = \frac{1}{1 - \Lambda_z} \cdot$$

Comme alors B_z est fini, on en déduit que $\Lambda_z < 1$.
D'après la proposition 1.2, $\lim\limits_{z \to z_c} B_z = +\infty$, et donc $\Lambda_{z_c} = 1$, c'est à dire, comme $z_c = 1/\mu$,

$$\sum_{k \geqslant 0} \frac{\lambda_k}{\mu^k} = 1.$$

On a ainsi une loi de probabilité sur les ponts irréductibles, avec la probabilité $\mathbb{P}(\omega) = \mu^{-|\omega|}$. Cette loi sera étudiée plus en détail dans la dernière section.

Intuitivement, on donne un 'poids' exponentiel en sa longueur à chaque pont irréductible. Ceci permet de 'pénaliser' les ponts irréductibles de grande longueur.

Si l'on pose ainsi $p_k = \lambda_k \mu^{-k}$ et $a_N = b_N \mu^{-N}$, on obtient l'équation :

$$a_N = \sum_{k \geqslant 0} p_k a_{N-k}.$$

Ceci illustre le lien entre la loi sur les ponts irréductibles et la construction de ponts, donnant une loi qui sera étudiée en dernière section. Cette équation s'interprète comme une équation de renouvellement discret au sens suivant :

Si on se donne une famille de variables aléatoires i.i.d. $(X_i)_{i \in \mathbb{N}}$ à valeurs dans $\mathbb{N}$, de loi $\mathbb{P}(X_1 = k) = p_k$ (typiquement des longueurs de ponts irréductibles), alors

$$a_N = \mathbb{P}(\exists k \in \mathbb{N} : X_1 + \cdots + X_k = N).$$

Le nombre a_N s'interprète ainsi comme la probabilité qu'un renouvellement ait lieu au temps N, c'est-à-dire que l'on 'change' de pont irréductible au N-ième pas avec une probabilité a_N.

On utilise ensuite le

Théorème 2.1 (du renouvellement). *Soient $(f_i)_{i\geqslant 1}$ et $(g_i)_{i\geqslant 0}$ deux suites positives, telles que l'on ait $0 < g = \sum_{n\geqslant 0} g_n < \infty$ et $f_1 > 0$. On pose $f = \sum_{n\geqslant 1} f_n$, et on définit la suite $(v_n)_{n\in\mathbb{N}}$ comme suit :*

$$\begin{cases} v_0 = g_0 \\ \forall n > 0, v_n = \sum_{i=1}^{n} f_i v_{n-i} + g_n. \end{cases}$$

Alors :

(i) si $f < 1$, $\lim_{n\to\infty} v_n = 0$, et $\sum v_n = g/(1-f)$;
(ii) si $f = 1$, $\lim_{n\to\infty} v_n = g/(\sum k f_k)$, et donc $\sum v_n$ diverge ;
(iii) si $f > 1$, $\limsup_{n\to\infty} v_n^{1/n} > 1$.

La preuve est détaillée en annexe B.
Ici, on a $f_n = p_n$, et $f = 1$, $g_n = \delta_{n,0}$ et $v_n = b_n\mu^{-n} = a_n$, et donc

$$\frac{b_n}{\mu^n} \underset{n\to\infty}{\longrightarrow} \frac{1}{\sum k p_k}.$$

Ceci permet d'affiner un résultat vu précédemment, à savoir que la 'constante de connectivité des ponts' est μ :
– Si $\sum k p_k$ (i.e. le temps moyen entre deux renouvellements) est fini, alors $b_N \sim C\mu^N$.
– Dans le cas où ce temps est infini, $b_N = \mathrm{o}(\mu^N)$.

2.2.2 Points de chute des ponts irréductibles

On s'intéresse maintenant aux 'points de chute' des ponts irréductibles. Pour cela, on se place en-dessous du point crtique, en $z \in (0, z_c)$.

Définition 2.3. *On note, pour N et L entiers, $b_{N,L}$ le nombre de ponts à N pas commençant en 0 et se terminant dans l'hyperplan $\{x_1 = L\}$, et, pour $y \in \mathbb{Z}^{d-1}$, $b_{N,L}(y)$ le nombre de ponts à N pas commençant en 0 et se terminant en (L, y).*

Si $\mathcal{A}$ est un ensemble de chemins, on note, pour L entier et $y \in \mathbb{Z}^{d-1}$, $\mathcal{A}(L)$ et $\mathcal{A}(L, y)$ l'ensemble des chemins dans $\mathcal{A}$ se terminant dans l'hyperplan $\{x_1 = L\}$, respectivement au point (L, y), et $A_z(L)$ et $A_z(L, y)$ leur fonction caractéristique.

Pour commencer, on a les égalités suivantes :

$$b_{N,L} = \sum_{k=0}^{N}\sum_{j=0}^{L} \lambda_{k,j} b_{N-k,L-j} + \delta_{N,0}\delta_{L,0}$$

et

$$b_{N,L}(y) = \sum_{k=0}^{N}\sum_{j=0}^{L}\sum_{v\in\mathbb{Z}^{d-1}} \lambda_{k,j}(v) b_{N-k,L-j}(y-v) + \delta_{N,0}\delta_{L,0}\delta_{y,0}.$$

De ceci, on déduit l'égalité sur les fonctions caractéristiques :

$$B_z(L) = \sum_{j=0}^{L} \Lambda_z(j) B_z(L-j) + \delta_{L,0}.$$

Pour le reste de cette section, fixons donc $z \in (0, z_c)$. On définit ensuite :

$$\begin{cases} p_L = p_L(z) = \Lambda_z(L)\mathrm{e}^{Lm(z)} = \sum_{\omega\in\mathcal{I}(L)} z^{|\omega|}\mathrm{e}^{Lm(z)} \\ a_L = a_L(z) = B_z(L)\mathrm{e}^{Lm(z)} = \sum_{\omega\in\mathcal{B}(L)} z^{|\omega|}\mathrm{e}^{Lm(z)}, \end{cases}$$

et $P(s)$ et $A(s)$ leurs fonctions génératrices respectives.
On obtient toujours une équation du type renouvellement : $a_N = \sum_{k\geqslant 0} p_k a_{N-k}$.

On a de plus le lemme suivant, qui nous affirme que l'on a bien une loi de probabilité :

Lemme 2.1. $\sum_{k\geqslant 0} p_k = 1$.

Démonstration. On a toujours l'égalité $A(s) = 1/(1-P(s))$. Par définition de $m(z)$, on obtient que le rayon de convergence de $A(s)$ est 1. De plus, on peut montrer par un raisonnement très long et fastidieux que nous ne détaillerons pas ici (le lecteur pourra trouver tous les détails dans [MS93]), que la suite (a_L) est minorée par $\chi(z)^{-2}$, et donc $A(s)$ diverge en 1.

On a donc bien $P(1) = \sum_{k\geqslant 0} p_k = 1$. □

Intuitivement, on a ici affaire à la même loi qu'au paragraphe précédent, mais on n'est plus au point critique. On doit alors donner un poids en $z^{|\omega|}$ à un pont irréductible, pondéré par l'exponentielle de son étendue, fois la masse. On peut montrer que $m(z)$ tend vers 0 quand $z \to z_c$, et quand $z = z_c$ on retrouve ainsi la loi précédente. Cependant les raisonnements suivants ne s'appliquent pas au cas $z = z_c$.

L'avantage de s'être placé en-dessous de z_c réside dans le fait que l'on peut utiliser cette loi pour choisir, non plus la longueur des ponts irréductibles, mais leur étendue (et donc les points de rupture du pont construit par concaténation), voire leur point de chute. En effet, rien ne nous dit que $a_L(z)$ converge quand z tend vers z_c.

On obtient aussi, par le théorème 2.1, $\lim_{L\to\infty} a_L = 1/(\sum_{k\geqslant 0} kp_k) > 0$, donc $\sum_{k\geqslant 0} kp_k < \infty$.

D'un point de vue probabiliste, ceci signifie que l'espérance de l'étendue d'un pont irréductible choisi aléatoirement selon cette loi est finie.

En fait, on a beaucoup mieux : les moments de la suite $(p_k)_k$ sont tous finis, ceci découlant du théorème suivant :

Théorème 2.2 (séparation des masses)**.** *On définit la 'masse pour les ponts irréductibles'*

$$m_\Lambda(z) = \liminf_{L\to\infty} \frac{-\log \Lambda_z(L)}{L}.$$

Alors

$$m_\Lambda(z) > m(z) \text{ pour tout } z < z_c,$$

ce qui est équivalent à

$$\limsup_{k\to\infty} p_k^{1/k} < 1.$$

La preuve de ce théorème est renvoyée en annexe A.

L'équivalence des deux assertions s'obtient par un calcul rapide avec les définitions de p_k et m_Λ, qui donne $\limsup p_k^{1/k} = m/m_\Lambda$.

On remarque aussi que

$$\Lambda_z(L) = \sum_{N\geqslant 3L} \lambda_{N,L} z^N \leqslant \sum_{N\geqslant 3L} \mu^N z^N = \frac{(\mu z)^{3L}}{1-\mu z}$$

et donc

$$m_\Lambda(z) \geqslant -3\log(\mu z).$$

De plus, comme $\lambda_{3L,L} \geqslant 1$ (voir le chemin $E^L N O^{L-1} N E^{L-1}$),

$$m_\Lambda(z) \leqslant -3\log z.$$

Le théorème du renouvellement nous donnait que la probabilité d'avoir un renouvellement en L (a_L) converge. Le théorème suivant nous montre que cette probabilité converge exponentiellement vite.

Théorème 2.3 (convergence exponentielle de $(a_L)_{L\in\mathbb{N}}$). *Pour tout $z \in (0, z_c)$, il existe $\varepsilon(z) > 0$ tel que, si l'on note $C_z = 1/\sum_{k\geqslant 0} kp_k$,*

$$|a_L - C_z| \leqslant \mathrm{e}^{-L\varepsilon(z)}.$$

Démonstration. On considère la fonction de variable complexe $s \mapsto P(s)$. On a vu que $P(1) = \sum p_k = 1$. Les coefficients de P étants réels, positifs, on en déduit que pour tout $s \neq 1$ et $|s| < 1$, on a $|P(s)| < 1$. Le théorème de séparation des masses nous dit qu'il existe $R = (\limsup p_k^{1/k})^{-1} > 1$ tel que P soit analytique sur $D(0, R)$. On a donc $r \in (1, R)$ tel que 1 soit le seul zéro de $P - 1$ sur $D(0, r)$. On remarque aussi que $P'(1) = C_z^{-1} \neq 0$.

Le théorème des résidus appliqué à la fonction analytique $s \mapsto \frac{s^{-L-1}}{1-P(s)}$ donne ainsi :

$$\begin{array}{ccccccc} \mathrm{Res}\left(\frac{s^{-L-1}}{1-P(s)}, 0\right) & + & \mathrm{Res}\left(\frac{s^{-L-1}}{1-P(s)}, 1\right) & = & \frac{1}{2\mathrm{i}\pi}\oint_{|s|=r} \frac{s^{-L-1}}{1-P(s)}\mathrm{d}s \\ a_L & - & C_z & = & \mathrm{O}(r^{-L}). \end{array}$$

Ce qui donne le résultat avec $r = \mathrm{e}^{\varepsilon(z)}$. □

On a à present un théorème nous donnant la répartition asymptotique des points de chute sur l'hyperplan de chute d'un pont, quand l'étendue du pont tend vers l'infini.

Théorème 2.4 (Distribution gaussienne du point de chute). *Pour tout $z \in (0, z_c)$, il existe $\delta_z > 0$ tel que*

$$\left| B_z(L, y)\mathrm{e}^{Lm(z)}L^{(d-1)/2} - C_z\frac{1}{(\pi\delta_z)^{(d-1)/2}}\mathrm{e}^{-\frac{|y|^2}{\delta_z L}} \right| \underset{L\to\infty}{\longrightarrow} 0 \qquad \textit{uniformément en } y.$$

Intuitivement, cela signifie que, asymptotiquement, les points de renouvellement (i.e. les points de chute) de ponts infinis construits avec cette loi suivent, conditionnellement à la première coordonnée, une loi gaussienne.

Démonstration. On considère le problème d'un point de vue probabiliste : On se donne ainsi (X_n, Y_n) des variables aléatoires indépendantes identiquement distribuées à valeurs dans $\mathbb{N} \times \mathbb{Z}^{d-1}$, de loi donnée par $\mathbb{P}((X_n, Y_n) = (L, y)) = p_{L,y} = \Lambda_z(L, y)\mathrm{e}^{Lm(z)}$ (le point d'arrivée d'un pont irréductible tiré d'après la loi de probabilité précédente). On construit ainsi un chemin auto-évitant dans un demi-espace en concaténant les ponts irréductibles obtenus : $S_n = \sum_{i=0}^{n}(X_i, Y_i)$. On a alors le résultat suivant :

$$\mathbb{P}(\exists n \in \mathbb{N} | S_n = (L, y)) = B_z(L, y)\mathrm{e}^{Lm(z)} = a_{L,y}.$$

En effet, supposons L et y non nuls (sinon la formule donne $1 = 1...$). On a alors

$$\begin{aligned} B_z(L, y)\mathrm{e}^{Lm(z)} &= \sum_{j=1}^{L}\sum_{v\in\mathbb{Z}^{d-1}} \Lambda_z(j, v)B_z(L - j, y - v)\mathrm{e}^{Lm(z)} \\ &= \sum_{j=1}^{L}\sum_{v\in\mathbb{Z}^{d-1}} \mathbb{P}((X, Y) = (j, v))B_z(L - j, y - v)\mathrm{e}^{(L-j)m(z)} \\ &= \sum_{k=1}^{L}\sum_{\substack{n_1+\cdots+n_k=L \\ v_1+\cdots+v_k=y}}\prod_{i=1}^{k}\mathbb{P}((X, Y) = (n_i, v_i)) \\ &= \mathbb{P}(X_1 + \cdots + X_k = L \text{ et } Y_1 + \cdots + Y_k = y \text{ pour un } k \in \mathbb{N}). \end{aligned}$$

Le théorème à démontrer dit juste que cette probabilité se factorise en probabilités gaussiennes de variance proportionnelle à $L/\mathbb{P}$(l'hyperplan $(L,.)$ est touché). C'est intuitivement vrai, à condition d'avoir Y de variance finie.

Or Y a un moment exponentiel : Si $z \in (0, z_c)$, il existe $s > 0$ tel que $\mathbb{E}(\mathrm{e}^{s|Y|}) < \infty$.

En effet, m_Λ est une fonction finie ($-3\log(\mu z) \leqslant m_\Lambda \leqslant -3\log z$), concave, donc continue car décroissante, en $\log z$ (la concavité provient d'une application de l'inégalité de Hölder, qui dit que si $(a_n) \geqslant 0$, $\beta \mapsto -\log \sum a_n \mathrm{e}^{n\beta}$ est concave, ici $\beta = \log z$ et $a_n = \lambda_{n,L}$, et du fait que la limite inférieure conserve la concavité). En se souvenant de la séparation des masses $m_\Lambda > m$, on a donc $s > 0$ tel que $m_\Lambda(\mathrm{e}^s z) > m(z)$ pour $\mathrm{e}^s z < z_c$, et ainsi :

$$\begin{aligned}\mathbb{E}(\mathrm{e}^{s|Y|}) &= \sum_{L,y} \mathrm{e}^{s|y|} \Lambda_z(L,y) \mathrm{e}^{Lm(z)} \\ &\leq \sum_{L,N} \mathrm{e}^{sN} \lambda_{N,L}(L) z^N \mathrm{e}^{Lm(z)} \\ &= \sum_L \Lambda_{z\mathrm{e}^s}(L) \mathrm{e}^{Lm(z)} < \infty.\end{aligned}$$

Reste à utiliser le

Théorème 2.5 (Stam)**.** *Si :*

(i) $0 < \mathbb{E}(X_1) = \theta < \infty$ *;*

(ii) le $\frac{d-1}{2}$*-ième moment de* X_1 *est fini ;*

(iii) $\mathrm{cov}(Y_{1,i}, Y_{1,j}) = 0$ *dès que* $i \neq j$ *et* $v = \mathrm{var}(Y_{1,i}) < \infty$ *;*

(iv) la distribution de (X_n, Y_n) *n'a pas de périodicité ;*

et si l'on note

$$\varphi(L,y) = \frac{1}{\theta}\left(\frac{\theta}{2\pi v}\right)^{(d-1)/2} \mathrm{e}^{-\theta|y|^2/2vL},$$

$$U(L,y) = \mathbb{E}\Big(\sum \mathbb{1}_{(X_1+\cdots+X_n, Y_1+\cdots+Y_n)=(L,y)}\Big),$$

alors $|L^{(d-1)/2}U(L,y) - \varphi(L,y)| \underset{L\to\infty}{\longrightarrow} 0$ *uniformément en* y*.*

Ce théorème est ici admis, la preuve complète (dans un cadre beaucoup plus général) est détaillée dans [Sta69, Sta71].

Dans notre cas, les hypothèses sont vérifiées :

(i) $X_1 > 0$ p.s., $\theta = C_z^{-1} < \infty$;

(ii) X_1 a un moment exponentiel car $\limsup p_k^{1/k} < 1$;

(iii) $(Y_{1,i}, Y_{1,j})$ suit la même loi que $(Y_{1,i}, -Y_{1,j})$ par symétrie, on a de plus une variance des Y_i indépendante de i. La variance v est fini par le lemme précédent ;

(iv) Il suffit de remarquer que $\forall (l,y)\ \mathbb{P}((X_1,Y_1) = (L,y)) > 0$, ce qui est évident.

Comme $U(L,y) = B_z(L,y)\mathrm{e}^{Lm(z)}$, avec $\delta_z = \frac{2v}{\theta}$, on a bien

$$\left| L^{(d-1)/2} B_z(L,y)\mathrm{e}^{Lm(z)} - (C_z^{-1})^{-1}\left(\frac{1}{\pi\delta_z}\right)^{\frac{d-1}{2}} \mathrm{e}^{-\frac{|y|^2}{\delta_z L}} \right| \underset{L\to\infty}{\longrightarrow} 0 \qquad \text{uniformément en } y. \quad \square$$

3 Théorème des motifs et une application

3.1 Théorème des motifs de Kesten

Définition 3.1. *On appelle* motif *un chemin auto-évitant (que l'on ne verra que comme partie d'un chemin plus grand).*

On dit qu'un motif $M = (m(0), \dots, m(n))$ apparaît au j-ième pas *du chemin* ω *s'il existe* $v \in \mathbb{Z}^d$ *tel que* $\omega(j+k) = m(k) + v$ *pour tout* $k \in [\![0, n]\!]$.

On note $c_N[k, M]$ *le nombre de chemins auto-évitants à* N *pas où* M *apparaît au plus* k *fois.*

On dit que M *est un* motif propre *si pour tout entier* k *il existe un chemin auto-évitant dans lequel* M *apparaît au moins* k *fois.*

Un motif propre est en fait un motif que l'on peut retrouver souvent dans les chemins auto-évitants, et la propriété suivante en fournit une caractérisation sympathique.

Proposition 3.1. *Un* M *motif est propre si et seulement s'il existe un cube* Q *et un chemin auto-évitant* φ *contenu dans* Q *et dont les extrémités sont des coins de* Q*, tels qu'il y ait une occurrence de* M *dans* φ*.*

Le sens indirect est clair, le lecteur trouvera une démonstration du sens direct dans [HW85].

Définition 3.2. *Soit* $Q = \{x \ : \ \forall i \in [\![1, d]\!] \ a_i \leqslant x_i \leqslant a_i + b\}$ *un cube. On définit alors son 'adhérence'* $\overline{Q} = \{x \in \mathbb{Z}^d \ : \ \forall i \in [\![1, d]\!] \ a_i - 2 \leqslant x_i \leqslant a_i + b + 2\}$ *et sa 'frontière'* $\delta Q = \overline{Q} \setminus Q$*.*

Soient M *un motif à* n *pas et* Q *un cube tels que* M *est contenu dans* Q*, et ses extrémités sont des coins de* Q*. On dit que* (M, Q) apparaît au j-ième pas *de* ω *s'il existe* $v \in \mathbb{Z}^d$ *tel que* $\omega(j+k) = m(k) + v$ *pour tout* $k \in [\![0, n]\!]$*, et* $\omega(i) \notin Q + v$ *pour tout* $i < j$ *ou* $i > j + n$*.*

On note $c_N[k, (M, Q)]$ *le nombre de chemins auto-évitants à* N *pas où* (M, Q) *apparaît au plus* k *fois.*

Le prochain résultat, dû à Kesten, est très intéressant : il dit qu'un motif propre donné apparaît souvent dans presque tous les chemins.

Théorème 3.1 (Théorème des motifs de Kesten)**.**
(i) Soit Q *un cube et* M *un motif comme dans la définition précédente. Il existe alors un* $a > 0$ *tel que :*

$$\limsup_{N \to \infty} (c_N[aN, (M, Q)])^{1/N} < \mu.$$

(ii) Pour tout motif propre M*, il existe un* $a > 0$ *tel que :*

$$\limsup_{N \to \infty} (c_N[aN, M])^{1/N} < \mu.$$

Remarquons que la propriété (i) implique la propriété (ii). En effet, si M est un motif propre, il suffit de voir que $c_N[aN, M] \leqslant c_N[aN, (\varphi, Q)]$ où φ et Q sont donnés par la proposition précédente.

Définition 3.3. *Soient* ω *un chemin auto-évitant à* N *pas et* r *un entier. On définit, pour* $j \in [\![0, N]\!]$ *:*

$$Q(j) = \{x \in \mathbb{Z}^d \ : \ |x_i - \omega_i(j)| \leqslant r \quad \forall i \in [\![1, d]\!]\}.$$

On dit que E^* a lieu au j-ième pas *de* ω *si* $Q(j)$ *est entièrement recouvert par* ω*. Pour tout* $k \geqslant 1$*, on dit que* E_k a lieu au j-ième pas *de* ω *si au moins* k *points de* $\overline{Q}(j)$ *sont visités par* ω *et on dit que* $\tilde{E}_k$ a lieu au j-ième pas *de* ω *si* E_k *ou* E^* *a lieu.*

Dans la suite, on utilisera le symbole E pour E^*, E_k, ou $\tilde{E}_k$. Si m est un entier, on dira que $E(m)$ a lieu au j-ième pas de ω si E a lieu au m-ième pas du chemin $(\omega(j-m), \dots, \omega(j+m))$. On définit $c_N[k, E]$ et $c_N[k, E(m)]$ comme étant respectivement le nombre de chemins auto-évitants à N pas où E a lieu au plus k fois et le nombre de chemins auto-évitants à N pas où $E(m)$ a lieu au plus k fois.

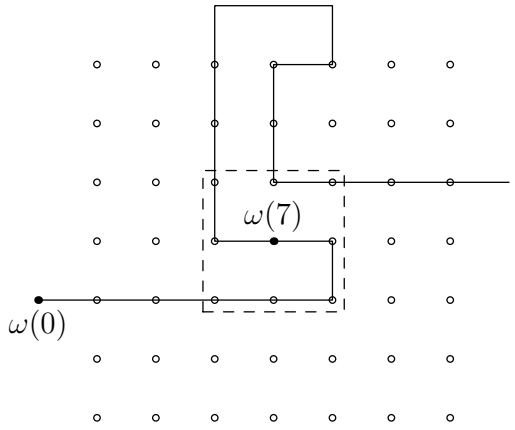


FIG. 3 – Illustration de la définition dans $\mathbb{Z}^2$ avec $r = 1$. En $j = 7$, E^* et E_{18} ont lieu.

Lemme 3.1. *Si*

$$\liminf_{N\to\infty} c_N[0,E]^{1/N} < \mu,$$

alors il existe un $a > 0$ et un entier m tels que

$$\limsup_{N\to\infty} c_N[aN, E(m)]^{1/N} < \mu.$$

Grossièrement, ce lemme nous dit que, si presque tous les chemins remplissent au moins un cube, alors ils en remplissent souvent. Et c'est effectivement le cas, comme on le verra au lemme suivant.

Démonstration. Puisque $c_N[0,E] = c_N[0,E(N)]$, il existe un $\varepsilon > 0$ et un entier m tels que

$$c_m[0,E(m)] < (\mu(1-\varepsilon))^m$$

et

$$c_m < (\mu(1+\varepsilon))^m.$$

On considère un chemin auto-évitant ω à N pas et on pose $M = \lfloor N/m \rfloor$. Si $E(m)$ a lieu au plus k fois dans ω alors $E(m)$ a lieu dans au plus k des M sous-chemins

$$(\omega((i-1)m), \omega((i-1)m+1), \ldots, \omega(im)) \qquad \text{avec } i \in [\![1, M]\!].$$

On en déduit la majoration

$$\begin{aligned} c_N[k,E(m)] &\leqslant \sum_{j=0}^{k} \binom{M}{j} c_m^j (c_m[0,E(m)])^{M-j} c_{N-Mm} \\ &\leqslant \mu^{Mm} c_{N-Mm} \sum_{j=0}^{k} \binom{M}{j} (1+\varepsilon)^{jm} (1-\varepsilon)^{Mm-jm}. \end{aligned}$$

Il suffit alors de montrer qu'il existe un $\rho > 0$ et un $t < 1$ tels que, pour tout M assez grand

$$c_m[\rho M, E(m)]^{1/M} < t\mu^m$$

(on prendra alors $0 < a < \rho/m$).

Or, si ρ est assez petit, on a

$$\sum_{j=0}^{\rho M} \binom{M}{j} (1+\varepsilon)^{jm} (1-\varepsilon)^{Mm-jm} \leqslant (\rho M+1) \binom{M}{\lfloor \rho M \rfloor} \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^{\rho M m} (1-\varepsilon)^{Mm}.$$

Quand M tend vers l'infini, la racine M-ième du membre de droite tend, par un calcul long mais facile utilisant la formule de Stirling, vers

$$\frac{1}{\rho^{\rho}(1-\rho)^{(1-\rho)}} \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^{\rho m} (1-\varepsilon)^{m}$$

qui est inférieur à 1 pour $0 < \rho < \rho_0$ avec ρ_0 assez petit. On conclut facilement. □

On a encore besoin d'un lemme, qui nous dit que presque tous les chemins 'remplissent' au moins un cube.

Lemme 3.2. $\liminf_{N\to\infty} c_N[0, E^*]^{1/N} < \mu$.

Démonstration. Supposons par l'absurde que $lim_{N\to\infty} c_N[0, E^*]^{1/N} = \mu$.
On va montrer qu'il existe alors un k tel que presque tous les chemins visitent k points de $\overline{Q}(j)$, pour beaucoup de j, et jamais plus.

On remarque que $c_N[0, \tilde{E}_k]$ est croissant en k. De plus, si E^* n'a pas lieu, alors $E_{(2r+5)^d}$ non plus, si bien que $c_N[0, E^*] \leqslant c_N[0, \tilde{E}_{(2r+5)^d}] \leqslant c_N$, et donc

$$\lim_{N\to\infty} (c_N[0, \tilde{E}_{(2r+5)^d}])^{1/N} = \mu.$$

Ensuite, on a $c_N[0, \tilde{E}_{r+3}] = 0$ pour $N \geqslant r+2$. On en déduit qu'il existe $k \in [\![r+3, (2r+5)^d - 1]\!]$ tel que

$$\lim_{N\to\infty} c_N[0, \tilde{E}_k]^{1/N} < \mu$$

et

$$\lim_{N\to\infty} c_N[0, \tilde{E}_{k+1}]^{1/N} = \mu.$$

D'après le lemme précédent, il existe donc $a > 0$ et un entier m tels que

$$\limsup_{N\to\infty} c_N[aN, \tilde{E}_k(m)]^{1/N} < \mu.$$

On définit l'ensemble $\mathcal{T}_N$ des chemins auto-évitants de longueur N tels que $\tilde{E}_{k+1}$ n'a jamais lieu, et $\tilde{E}_k(m)$ au moins aN fois. On a

$$|\mathcal{T}_N| \geqslant c_N[0, \tilde{E}_{k+1}] - c_N[aN, \tilde{E}_k(m)]$$

et donc

$$\lim_{N\to\infty} |\mathcal{T}_N|^{1/N} = \mu.$$

La suite de la démonstration repose sur le fait suivant :
Étant donné un chemin auto-évitant ω dans $\mathcal{T}_N$, on considère les $\overline{Q}(j)$ dont exactement k points sont visités par ω. On peut alors remplacer, dans certains de ces cubes, ω par un chemin auto-évitant qui les remplit complètement. Ceci fournit un grand nombre de chemins auto-évitants visitant plus de k points dans assez de cubes pour contredire le résultat précédent. Les cubes choisis doivent en particulier être d'intersection vide.
Nous ne rédigerons pas ici les détails, que le lecteur intéressé trouvera dans [MS93]. □

Preuve du théorème de Kesten. Soient M et Q comme dans l'énoncé du théorème. Quitte à translater et allonger M, on peut supposer que $Q = \{x \in \mathbb{Z}^d \ : \ |x_i| \leqslant r \quad \forall i \in [\![1, d]\!]\}$. Le raisonnement est en tout point similaire à celui de la démonstration précédente : on suppose par l'absurde que pour tout $a > 0$, $\limsup_{N\to\infty}(c_N[aN, (M, Q)])^{1/N} = \mu$.

On dit que $\overline{E}^*$ a lieu au j-ième pas de ω si $\overline{Q}(j)$ est complètement recouvert par ω. Les deux lemmes précédents nous disent qu'il existe un $b > 0$ et un entier m tels que

$$\limsup_{N\to\infty}(c_N[bN, \overline{E}^*(m)])^{1/N} < \mu.$$

Soit $c > 0$, on définit $\mathcal{G}_N$ comme étant l'ensemble des chemins auto-évitants tels que (P, Q) a lieu au plus cN fois et $\overline{E}^*(m)$ au moins bN fois. Comme précédemment, on a que

$$\lim_{N\to\infty} |\mathcal{G}_N|^{1/N} = \mu.$$

Le même argument que pour le lemme précédent fournit encore une contradiction : quand un cube est complètement rempli, on peut remplacer la portion de ω qui est dedans par un chemin contenant le motif M. □

3.2 Théorème ratio-limite

Cette partie vise à montrer un résultat énoncé dans la première section : $c_{N+2}/c_N \longrightarrow \mu^2$, ainsi qu'un analogue pour les ponts.

Lemme 3.3. *Soit $(a_n)_{n\geqslant 1}$ une suite de réels strictement positifs et posons, pour $n \geqslant 1$, $\varphi_n = a_{n+2}/a_n$. Supposons que :*

(i) $(a_n^{1/n})_{n\geqslant 1}$ converge vers un réel strictement positif α ;

(ii) $\liminf_{n\to\infty} \varphi_n > 0$;

(iii) il existe un réel $D > 0$ tel que, pour n assez grand,

$$\varphi_n\varphi_{n+2} \geqslant \varphi_n^2 - \frac{D}{n}\cdot$$

Alors la suite $(\varphi_n)_{n\geqslant 1}$ converge et

$$\varphi_n \underset{n\to\infty}{\longrightarrow} \alpha^2.$$

Démonstration. Nous allons montrer que la suite $(\sigma_n)_{n\geqslant 1}$ définie par $\sigma_n = \varphi_n - \alpha^2$ n'admet pas de valeur d'adhérence non nulle dans $\overline{\mathbb{R}}$. Déjà, (ii) et (iii) permettent de voir qu'il existe $B > 0$ tel que, pour n assez grand,

$$\varphi_{n+2} \geqslant \varphi_n - \frac{B}{n}\cdot$$

Supposons que $(\sigma_{n(j)})_{j\geqslant 1}$ converge vers $\varepsilon \in [0, \infty]$ où $(n(j))_{j\geqslant 1}$ est une suite strictement croissante d'entiers. Posons, pour tout $j \geqslant 1$,

$$M(j) = \left\lfloor \frac{n(j)\sigma_{n(j)}}{2B} \right\rfloor.$$

Pour j assez grand et $0 \leqslant k \leqslant M(j)$, une récurrence évidente permet de voir que

$$\begin{aligned}
\varphi_{n(j)+2k} &\geqslant \varphi_{n(j)} - \frac{kB}{n(j)} \\
&\geqslant \alpha^2 + \sigma_{n(j)} - \frac{M(j)B}{n(j)} \\
&\geqslant \alpha^2 + \frac{\sigma_{n(j)}}{2}
\end{aligned}$$

On déduit de ceci que

$$\frac{a_{n(j)+2M(j)}}{a_{n(j)}} = \prod_{k=0}^{M(j)-1} \varphi_{n(j)+2k} \geqslant \left(\alpha^2 + \frac{\sigma_{n(j)}}{2}\right)^{M(j)}.$$

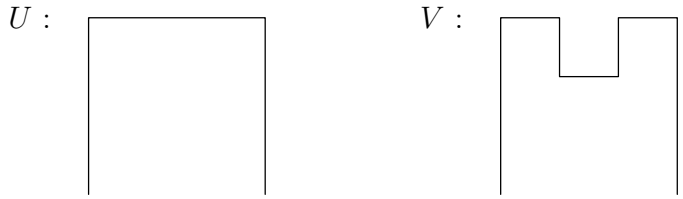
En prenant la racine $M(j)$-ième et en faisant tendre j vers l'infini, on trouve $\alpha^2 \geqslant \alpha^2 + \frac{\varepsilon}{2}$, ce qui implique $\varepsilon = 0$.

On procède de la même manière pour montrer que la suite $(\sigma_n)_{n\geqslant 1}$ n'a pas de valeur d'adhérence strictement négative. On en déduit donc qu'elle tend vers 0. □

Proposition 3.2. *Les suites $(c_n)_{n\geqslant 1}$ et $(b_n)_{n\geqslant 1}$ vérifient les hypothèses du lemme précédent.*

Démonstration. La condition (i) a déjà été prouvée dans les deux cas, avec $\alpha = \mu$. La condition (ii) n'est pas très difficile à voir : on sait que $b_{n+2} \geqslant b_n b_2$ donc $b_{n+2} \geqslant b_n$. Pour voir que $c_{n+2} \geqslant c_n$, considérons un chemin auto-évitant ω et M le maximum de la première coordonnée des points de ω. Si l'hyperplan d'équation $x_1 = M$ contient deux points consécutifs de ω, mettons $\omega(i)$ et $\omega(i+1)$, alors on remplace la liaison entre ces deux points par $(\omega(i), \omega(i) + e_1, \omega(i+1) + e_1, \omega(i+1))$. Si ce n'est pas le cas, alors le seul point dans cet hyperplan est le premier ou le dernier point de ω. Dans ce cas, on rajoute e_1 puis $2e_1$ à ce point. Cette construction donne une injection des chemins auto-évitants de longueur n dans ceux de longeur $n+2$.

Nous allons maintenant montrer la propriété (iii), ce qui va nécessiter l'utilisation du théorème des motifs. Posons, pour $n \in \mathbb{N}$, $a_n = b_n$ ou c_n, selon la suite pour laquelle nous voulons montrer (iii), et, similairement, soit $\mathcal{A}_n$ l'ensemble des ponts ou des chemins auto-évitants à n pas . Introduisons les motifs U et V, qui sont dans le plan engendré par e_1 et e_2 : ils commencent à l'origine, le motif U prend le chemin $N^3E^3S^3$ et V suit $N^3ESENES^3$.



Soit Q le cube défini par :

$$Q = \{x \in \mathbb{Z}^d, \forall i \in [\![0, d]\!], \quad 0 \leqslant x_i \leqslant 3\}.$$

U et V sont contenus dans Q et leurs extrémités sont des coins de Q. Pour i et j deux entiers positifs, posons $\mathcal{A}_n(i,j)$ l'ensemble des éléments de $\mathcal{A}_n$ pour lesquels (U,Q) et (V,Q) ont lieu exactement i et j fois respectivement, et $a_n(i,j)$ son cardinal. Définissons aussi

$$a_n(\geqslant i, \geqslant j) = \sum_{k\geqslant i, l\geqslant j} a_n(k,l).$$

Donnons-nous un $n \geqslant 1$, un $i \in \mathbb{N}$ et un $j \in \mathbb{N}$, et considérons l'ensemble des paires (ω, ω') avec $\omega \in \mathcal{A}_n(i,j)$, $\omega' \in \mathcal{A}_{n+2}(i-1, j+1)$ telles que ω' peut être obtenu à partir de ω en remplaçant une occurrence de (U,Q) par (V,Q). En comptant le nombre de ω' qui conviennent pour un ω et inversement, on voit que

$$\textit{nombre de paires} = i a_n(i,j) = (j+1) a_{n+2}(i-1, j+1).$$

Faisons maintenant un peu de calcul ; soit $n \geqslant 1$, on a

$$a_{n+2}(\geqslant 0, \geqslant 1) = \sum_{i\geqslant 1, j\geqslant 0} a_{n+2}(i-1, j+1) = \sum_{i\geqslant 1, j\geqslant 0} \frac{i a_n(i,j)}{j+1}$$

ainsi que

$$a_{n+4}(\geqslant 0, \geqslant 2) = \sum_{i\geqslant 2, j\geqslant 0} \frac{i(i-1)a_n(i,j)}{(j+1)(j+2)}.$$

L'inégalité de Cauchy-Schwarz nous dit que

$$\left(\sum_{i\geqslant 1, j\geqslant 0} \frac{i^2 a_n(i,j)}{j+1}\right)^2 \leqslant \left(\sum_{i\geqslant 1, j\geqslant 0} a_n(i,j)\right)\left(\sum_{i\geqslant 1, j\geqslant 0} \frac{i^2 a_n(i,j)}{(j+1)^2}\right)$$

et donc que

$$[a_{n+2}(\geqslant 0, \geqslant 1)]^2 \leqslant a_n \sum_{i\geqslant 1, j\geqslant 0} \frac{i a_n(i,j)}{(j+1)^2}.$$

Pour $n \geqslant 1$, posons

$$\Xi_n = \frac{a_{n+4}(\geqslant 0, \geqslant 2)}{a_n} - \left(\frac{a_{n+2}(\geqslant 0, \geqslant 1)}{a_n}\right)^2$$

ainsi que ($\varphi_n = \frac{a_{n+2}}{a_n}$)

$$\hat{\Xi}_n = \varphi_n \varphi_{n+2} - \varphi_n^2 - \Xi_n.$$

Pour obtenir ce que l'on veut, nous allons montrer que $\hat{\Xi}_n$ tend vers 0 exponentiellement vite, et ensuite minorer par quelque chose de la forme $-D/n$ ou D est strictement positif .

$$\begin{aligned}|\hat{\Xi}_n| &\leqslant \left|\frac{a_{n+4} - a_{n+4}(\geqslant 0, \geqslant 2)}{a_n}\right| + \left|\frac{a_{n+2}^2 - [a_{n+2}(\geqslant 0, \geqslant 1)]^2}{a_n^2}\right| \\ &\leqslant \frac{c_{n+4}[1,(V,Q)]}{a_n} + \frac{2a_{n+2}c_{n+2}[0,(V,Q)]}{a_n^2}\end{aligned}$$

Le théorème des motifs et le fait que $a_n^{1/n} \to \mu$ nous disent que $\hat{\Xi}_n$ décroît vers 0 exponentiellement vite. Regardons maintenant Ξ_n. Le théorème des motifs nous donne un $\lambda > 0$ tel que

$$\limsup_{n\to\infty} \left(1 - \frac{a_n(\geqslant 0, \geqslant \lambda n)}{a_n}\right)^{1/n} < 1.$$

De plus,

$$\begin{aligned}\Xi_n &\geqslant \left(\sum_{i\geqslant 0, j\geqslant 0} \frac{i(i-1)a_n(i,j)}{(j+1)(j+2)} - \sum_{i\geqslant 0, j\geqslant 0} \frac{i a_n(i,j)}{(j+1)^2}\right)\frac{1}{a_n} \\ &\geqslant \left(\sum_{i\geqslant 0, j\geqslant 0} \frac{-(i^2+ij+i)a_n(i,j)}{(j+1)^2(j+2)}\right)\frac{1}{a_n}.\end{aligned}$$

Si i ou j est strictement plus grand que n, le terme correspondant dans la somme est nul. On peut donc minorer $-(i^2+ij+i)$ par $-3n^2$. On découpe ensuite la somme entre $j < \lambda n$ et $\lambda n \leqslant j \leqslant n$ pour obtenir :

$$\Xi_n \geqslant \frac{-3n^2 a_n(\geqslant 0, \geqslant \lambda n)}{(\lambda n)^3 a_n} + (-3n^2)\left(1 - \frac{w_n(\geqslant 0, \geqslant \lambda n)}{a_n}\right).$$

Le terme de gauche est équivalent à $-\frac{3}{\lambda^3 n}$ alors que l'autre décroît exponentiellement vers 0. Le résultat est alors prouvé. □

Théorème 3.2. *On a les limites suivantes :*

(a) $\frac{c_{n+2}}{c_n} \underset{n\to\infty}{\longrightarrow} \mu^2$,

(b) $\frac{b_{n+1}}{b_n} \underset{n\to\infty}{\longrightarrow} \mu$.

Démonstration. Les résultats précédents nous donnent facilement (a) ainsi que le fait que $b_{n+2}/b_n \to \mu^2$. Montrons donc (b). Pour tout $j \in \mathbb{N}$, posons $L_j = \liminf_{n\to\infty} b_{n-j}/b_n$. Nous voulons montrer que $L_1 = \mu^{-1}$ et que la limite inférieure est en fait une limite.

Le résultat précédent nous dit que, pour tout j, $L_{j+2} = \mu^{-2} L_j$. On en déduit que $L_j = \mu^{-j}$ si j est pair, et $L_j = \mu^{1-j} L_1$ si j est impair.

Le fait que $b_n = \sum_{s=1}^{n} \lambda_s b_{n-s} + \delta_{n,0}$ (où λ_s est le nombre de ponts irréductibles à s pas) nous permet d'obtenir pour $j < n$

$$\frac{b_{n-j}}{b_n} = \sum_{s=1}^{n-j} \lambda_s \frac{b_{n-j-s}}{b_n}.$$

Le lemme de Fatou donne alors

$$L_j \geqslant \sum_{s=1}^{\infty} \lambda_s L_{j+s}.$$

Posons

$$\Sigma_{pair} = \sum_{s\geqslant 1, s\ pair} \lambda_s \mu^{-s} \ et\ \Sigma_{impair} = \sum_{s\geqslant 1, s\ impair} \lambda_s \mu^{-s}$$

On a $\Sigma_{pair} + \Sigma_{impair} = 1$ (voir section 3.2.1). L'inéquation précédente appliquée avec $j = 0$ donne

$$1 \geqslant L_1 \lambda \Sigma_{impair} + \Sigma_{pair}.$$

On en déduit $L_1 \leqslant \mu^{-1}$. En appliquant l'inéquation avec $j = 1$, on trouve

$$L_1 \geqslant \mu^{-1} \Sigma_{impair} + L_1 \Sigma_{pair}$$

et on en déduit $L_1 \geqslant \mu^{-1}$, on a donc montré que

$$\liminf_{n\to\infty} \frac{b_{n-1}}{b_n} = \mu^{-1}.$$

En réutilisant ceci, on a

$$\limsup_{n\to\infty} \frac{b_{n-1}}{b_n} = \limsup_{n\to\infty} \frac{b_{n-1}}{b_n + 1} \frac{b_{n+1}}{b_n} = \mu^{-2} L_1^{-1} = \mu^{-1},$$

ce qui conclut la preuve de (b). □

4 Lois de probabilités sur les ponts infinis

Dans cette section, nous allons étudier deux manières de définir une loi de probabilité sur l'ensemble des 'ponts infinis', c'est-à-dire les chemins auto-évitants infinis ω, tels que pour tout $i > 0$, $\omega_1(i) > 0$.

La première est instinctive, définie grâce aux événements cylindriques, et utilise les résultats de la section 4.2, la seconde s'appuie sur le travail de la section 3, en construisant un pont infini comme concaténation de ponts irréductibles.

Il s'avère que ce sont en réalité deux visions de la même loi.

4.1 Une loi naturelle

On va ici montrer que la proportion de ponts de longueur N commençant par un chemin auto-évitant ω donné (parmi les ponts de longueur N) converge quand $N \to \infty$.

Définition 4.1. *On définit $\mathcal{B}_n(\omega)$ l'ensemble des ponts de longueur n prolongeant un chemin auto-évitant dans le demi-espace droit ω donné, puis on définit :*

$$P_n^B(\omega) = \frac{|\mathcal{B}_n(\omega)|}{b_n} \quad et$$

$$P^B(\omega) = \lim_{n\to\infty} P_n^B(\omega)$$

(la limite existe effectivement, voir ci-dessous).

Théorème 4.1. *Soit ω un chemin auto-évitant dans le demi-espace droit, alors la limite $P^B(\omega)$ existe.*

Démonstration. On a d'abord besoin d'un peu de notations.
On pose d'abord $m = |\omega|$. Si β est un pont à n pas qui prolonge ω, on note $M(\beta, \omega)$ le plus petit entier $i \geqslant m$ tel que $\beta_1(i)$ soit un point de rupture de β.
Pour $k \geqslant m$, on note $\mathcal{E}_k(\omega)$ l'ensemble des ponts à k pas, tels que $M(\beta, \omega) = k$.

Si β prolonge ω, $M(\beta, \omega)$ est l'unique entier k tel que $(\beta(0), \dots, \beta(k)) \in \mathcal{E}_k(\omega)$ et $(\beta(k), \dots, \beta(n))$ soit un pont.

D'où l'égalité :

$$|\mathcal{B}_n(\omega)| = \sum_{k=m}^{n} |\mathcal{E}_k(\omega)| b_{n-k}.$$

On va à présent prouver que

$$P_n^B(\omega) \underset{n\to\infty}{\longrightarrow} \sum_{k\geqslant m} |\mathcal{E}_k(\omega)| \mu^{-k}.$$

Si l'on divise l'égalité précédente par b_n, prendre la limite inférieure nous donne (en se souvenant que $\lim \frac{b_{n+1}}{b_n} = \mu$) :

$$\liminf P_n^B(\omega) \geqslant \sum_{k\geqslant m} |\mathcal{E}_k(\omega)| \mu^{-k}$$

par le lemme de Fatou.

Il reste donc à majorer la limite supérieure :

Soient $k \geqslant m$ et $\beta \in \mathcal{E}_k(\omega)$. Si β est un pont irréductible, on pose $I = 0$. Sinon, on définit I comme le plus grand entier i tel que $\beta_1(i)$ soit un point de rupture de β.

Par définition de $\mathcal{E}_k(\omega)$, $I \leqslant m$. Alors $(\beta(0), \dots, \beta(I)) = (\omega(0), \dots, \omega(I))$ et $(\beta(I), \dots, \beta(k))$ est un pont irréductible. On obtient ainsi

$$|\mathcal{E}_k(\omega)| \leqslant \sum_{i=0}^{m} \lambda_{k-i} \quad \forall k \geqslant m.$$

À présent, soit $J \geqslant m$. Pour tout $n > J$, en se souvenant que $b_n = \sum_{s=0}^{n} \lambda_s b_{n-s}$,

$$\begin{aligned}
|\mathcal{B}_n(\omega)| - \sum_{k=m}^{J} |\mathcal{E}_k(\omega)| b_{n-k} &\leqslant \sum_{k=J+1}^{n} \left(\sum_{i=0}^{m} \lambda_{k-i} \right) b_{n-k} \\
&\leqslant \sum_{i=0}^{m} \left(\sum_{k=J+1-m+i}^{n} \lambda_{k-i} b_{n-k} \right) \\
&\leqslant \sum_{i=0}^{m} \left(b_{n-i} - \sum_{r=0}^{J-m} \lambda_r b_{n-i-r} \right).
\end{aligned}$$

On divise par b_n, et on obtient

$$\limsup P_n^B(\omega) - \sum_{k=m}^{J} |\mathcal{E}_k(\omega)| \mu^{-k} \leqslant \sum_{i=0}^{m} \left(\mu^{-i} - \sum_{r=0}^{J-m} \lambda_r \mu^{-i-r} \right).$$

Le terme de droite tend vers 0 quand $J \to \infty$, ce qui prouve le résultat. □

La 'loi' P^B est compatible au sens suivant : si $|\omega| \leqslant n$, alors

$$P^B(\omega) = \sum_{\substack{\rho > \omega \\ |\rho| = n}} P^B(\rho).$$

Le théorème d'extension de Kolmogorov nous dit alors que l'on peut définir une loi de probabilité P_∞^B sur les ponts infinis, donnée par ses valeurs sur les événements cylindriques $\{\zeta[0,m] = \omega\}$:

$$P_\infty^B(\{\zeta[0,m] = \omega\}) = P^B(\omega).$$

4.2 La loi 'suite de ponts irréductibles'

On va maintenant s'intéresser à une autre façon de construire des chemins infinis auto-évitants dans un demi-plan, à partir de concaténation de ponts irréductibles, sur lesquels on a déjà une loi de probabilité vue dans la section 3.2.1.

Définition 4.2. *On se donne une famille dénombrable de ponts irréductibles aléatoires indépendants $(\eta^{[i]})_{i>0}$, tirés selon la loi vue dans la section 3.2. On note $X_i = |\eta^{[i]}|$. On a ainsi $\mathbb{P}(X_i = k) = \lambda_k \mu^{-k}$. On obtient ainsi un chemin infini auto-évitant aléatoire $\rho = \eta^{[1]} \circ \eta^{[2]} \circ \ldots$, dont on notera la loi Q^B.*

Théorème 4.2. *Les deux lois de probabilité sur les chemins infinis auto-évitants dans un demi-plan sont identiques, i.e., pour tout $m \geqslant 1$, pour tout ω chemin auto-évitant dans le demi-espace droit à n pas,*

$$Q^B\{\rho[0,m] = \omega\} = P_\infty^B\{\zeta[0,m] = \omega\}.$$

Démonstration. Pour $k \geqslant 1$, on pose A_k l'événement '$\rho_1(k)$ est un point de rupture pour ρ', c'est-à-dire

$$\begin{aligned}
A_k &= \{\forall r \leqslant k \ \rho_1(r) \leqslant \rho_1(k) \text{ et } \forall r > k \ \rho_1(r) > \rho_1(k)\} \\
&= \{\exists i \geqslant 1 \ X_1 + \cdots + X_i = k\}.
\end{aligned}$$

Alors pour tout pont à k pas β, $Q^B(\{\rho[0,k] = \beta\} \cap A_k) = \mu^{-k}$.

En effet, soit β un pont à k pas. On peut écrire de manière unique $\beta = \varphi^{[1]} \circ \cdots \circ \varphi^{[j]}$, où les $\varphi^{[i]}$ sont des ponts irréductibles. Alors

$$\begin{aligned} Q^B(\{\rho[0,k] = \beta\} \cap A_k) &= Q^B(\{\eta^{[1]} = \varphi^{[1]}, \ldots, \eta^{[j]} = \varphi^{[j]}\}) \\ &= \mu^{-|\varphi^{[1]}|} \ldots \mu^{-|\varphi^{[j]}|} \\ &= \mu^{-(|\varphi^{[1]}|+\cdots+|\varphi^{[j]}|)} = \mu^{-|\beta|} = \mu^{-k}. \end{aligned}$$

À présent, fixons $m \geqslant 1$, ω un chemin auto-évitant dans le demi-espace droit à m pas. On pose $T = \min\{i \mid X_1 + \cdots + X_i \geqslant m\}$. On a ensemble les deux événements $\{\rho[0,k] \in \mathcal{E}_k(\omega)\}$ et A_k si et seulement si on a les deux événements $\{\rho[0,m] = \omega\}$ et $\{X_1 + \cdots + X_T = k\}$. D'où

$$\begin{aligned} Q^B(\{\rho[0,m] = \omega\}) &= \sum_{k \geqslant m} Q^B(\{\rho[0,m] = \omega\} \cap \{X_1 + \cdots + X_T = k\}) \\ &= \sum_{k \geqslant m} Q^B(\{\rho[0,k] \in \mathcal{E}_k(\omega)\} \cap A_k) \\ &= \sum_{k \geqslant m} |\mathcal{E}_k(\omega)| \mu^{-k} \\ &= P^B_\infty(\zeta[0,m] = \omega), \end{aligned}$$

ce que l'on voulait. □

4.3 Conclusion : vers une courbe continue...

Nous avons ainsi créé une courbe aléatoire auto-évitante discrète de longueur infinie. Les deux constructions précédentes montrent que cette mesure sur les chemins infinis combine en quelque sorte deux propriétés :

- Elle est d'une certaine façon une loi uniforme sur les chemins auto-évitants de longueur infinie.
- On peut la découvrir au fur et à mesure comme un processus $(\gamma_n)_{n \geqslant 0}$, et donc la voir comme un 'processus de croissance'.

On peut alors s'interroger sur l'existence d'un analogue continu. En se souvenant que les marches aléatoires simples convergent vers le mouvement brownien lorsque l'on fait tendre la maille δ du réseau vers 0, on peut se demander s'il n'y a pas de résultat similaire pour les chemins auto-évitants. Cette analogie suggère la conjecture suivante, lorsque γ est la marche infinie aléatoire dans le demi-espace que nous venons de construire :

Conjecture : *La loi de la courbe $\delta\gamma$ converge (en un sens à préciser) lorsque la maille du réseau δ tend vers 0 vers la loi d'une courbe continue auto-évitante.*

En dimension 2, on connait en fait un candidat naturel pour cette courbe auto-évitante aléatoire limite. Il est en effet conjecturé que, tout comme le mouvement brownien, cet objet limite est en un certain sens invariant par 'transformations conformes' ce qui permet de caractériser ce candidat naturel. Ceci fait intervenir des processus de croissance dits SLE (Evolution de Schramm-Loewner) obtenus par itérations de transformations conformes aléatoires, mais ceci dépasse le cadre de ce mémoire. Une propriété remarquable de cette loi limite conjecturée, est que les courbes obtenues sont presque sûrement de dimension fractale 4/3.

ANNEXES

A Séparation des masses

Le but de cette section est de montrer le résultat suivant :

Théorème A.1. *$m_\Lambda(z) > m(z)$ pour tout $z < z_c$.*

Pour cela, on va s'intéresser aux *retours* d'un pont.

A.1 Les retours

Définition A.1. *Si ω est un pont à N pas, un* retour *de ω est un sous-chemin $\omega[s,t]$ $(0 \leqslant s < t \leqslant N)$ tel que :*
(i) $\forall i \in [\![s+1, t-1]\!]\ \omega_1(t) \leqslant \omega_1(i) < \omega_1(s)$,
(ii) $\forall i \in [\![0, s-1]\!]\ \omega_1(i) \leqslant \omega_1(s)$,
(iii) $\forall i \in [\![t+1, N]\!]\ \omega_1(t) < \omega_1(i)$.
Son étendue *est $\omega_1(s) - \omega_1(t)$.*

Lemme A.1. *Soit ω un pont à N pas. Si $k \in [\![0, N]\!]$, il existe au plus un retour $\omega[s,t]$ contenant $\omega(k)$.*

Démonstration. Soit $\omega[s;t]$ un tel retour. On va montrer que s et t sont uniquement déterminés.
Les points (i) et (ii) impliquent que, si $M = \max_{j \in [\![0,k-1]\!]} \omega_1(j)$, $s = \max\{i \in [\![0, k-1]\!], \omega_1(i) = M\}$.
Les points (i) et (iii) impliquent que, si $m = \min_{j \in [\![k+1,N]\!]} \omega_1(j)$, $t = \max\{i \in [\![k+1, N]\!], \omega_1(i) = m\}$. □

Corollaire A.1. *Deux retours d'un même pont sont égaux ou disjoints.*

Définition A.2. *Soit ω un pont à N pas. On dit qu'un retour $\omega[s,t]$* couvre *l'entier j si $\omega_1(0) \leqslant j < \omega_1(N)$.*

Lemme A.2. *Pour tout τ $(\omega_1(0) \leqslant \tau < \omega_1(N))$, τ est soit un point de rupture, soit est couvert par un retour.*

Démonstration. Soit τ comme précédemment, qui n'est pas un point de rupture. On pose alors $T = \{j \in [\![0, N]\!] :\ \omega_1(j) = \tau\}$, $\tau_m = \min(T) < \tau_M = \max(T)$ (car $|T| \geqslant 3$). Soit $M = \max\{\omega_1(i) : i \in [\![\tau_m, \tau_M]\!]\} \geqslant \tau$.
Si $M > \tau$, on pose :

$$\begin{cases} s = \max\{i \in [\![\tau_m, \tau_M]\!] :\ \omega_1(i) = M\}, \\ m = \min\{\omega_1(i) :\ i \in [\![s, \tau_M]\!]\}, \\ t = \max\{i \in [\![s, \tau_M]\!] :\ \omega_1(i) = m\}. \end{cases}$$

Alors $\omega[s,t]$ est un retour couvrant τ.
Si $M = \tau$, on pose :

$$\begin{cases} m = \min\{\omega_1(i) :\ i \in [\![\tau_m, \tau_M]\!]\}, \\ t = \max\{i \in [\![\tau_m, \tau_M]\!] :\ \omega_1(i) = m\}, \\ M' = \max\{\omega_1(i) :\ i \in [\![\tau_m, t]\!]\}, \\ s = \max\{i \in [\![s, \tau_M]\!] :\ \omega_1(i) = M'\}. \end{cases}$$

Alors $\omega[s,t]$ est un retour couvrant τ. □

A.2 Preuve du théorème de séparation des masses

Dans tout ce qui suit, on fixe $z \in (0, z_c)$ (on oubliera ainsi les indices $_z$ dans les fonctions génératrices, par exemple), et on utilise la notation suivante : si S est un ensemble de chemins auto-évitants, on note $FG(S) = \sum_{\omega \in S} z^{|\omega|}$ sa fonction génératrice.

On va commencer par montrer que les points de rupture sont 'communs', au sens où un long intervalle sur l'axe des abscisses a peu de chances de ne pas comporter de points de rupture.

Définition A.3. *Soient $c \geqslant 0$, $T \geqslant 1$, $L \geqslant c+T$. On définit $B^*(L; c, T)$ la fonction génératrice des ponts d'étendue L, commençant en 0, qui n'ont pas de points de rupture dans $[\![c+1, c+T-1]\!]$.*

Lemme A.3. *Il existe une fonction décroissante $\varepsilon(T)$, qui tend vers 0 quand $T \to \infty$, vérifiant pour tous $L \geqslant 0$, $T \in [\![0, L]\!]$, $c \in [\![0, L+T]\!]$:*

$$B^*(L; c, T) \leqslant \mathrm{e}^{-mL}\varepsilon(T).$$

Démonstration. En considérant le dernier point de rupture avant c et le premier après $c+T$, on voit que

$$B^*(L; c, T) = \sum_{i=0}^{c} \sum_{j=c+T}^{L} B(i)\Lambda(j-i)B(L-j)$$

et donc

$$\begin{aligned} B^*(L; c, T)\mathrm{e}^{mL} &= \sum_{i=0}^{c} \sum_{j=c+T}^{L} a_i p_{j-i} a_{L-j} \\ &\leqslant \sum_{i=T}^{L} i p_i, \end{aligned}$$

où l'on a réutilisé les notations de la section 3.2 et l'on s'est souvenu que $a_n \leqslant 1$.

Si l'on pose $\varepsilon(T) = \sum_{i \geqslant T} i p_i$, on a ce que l'on voulait. □

À présent, on cherche à majorer la fonction génératrice des ponts irréductibles Λ.
Pour cela, on fixe un entier Q, et pour $L \gg Q$, on divise $[\![0; L]\!]$ en blocs de taille Q. On regarde ensuite les retours qui couvrent les extrémités des blocs. On va ainsi montrer que les ponts irréductibles ont soit une majorité de petits retours, ce qui permet d'avoir des ponts irréductibles au 'centre' de chaque bloc, et d'utiliser le lemme précédent pour majorer, soit une grande quantité de grands retours, que l'on majore via $B(k) \leqslant \mathrm{e}^{-mk}$.

Soient T et δ des entiers positifs, et on pose $Q = 2\delta + T$: δ sera l'étendue maximale d'un 'petit' retour, et T la longueur des 'sous-ponts irréductibles' dans les blocs. Pour L grand, on pose $k = \lfloor L/Q - 1 \rfloor$, et on divise $[\![0; L]\!]$ en $k+1$ sous-intervalles. On pose $A = \{Q; 2Q; \ldots; kQ\}$ les extrémités des blocs (autres que 0 et L). On prend un sous-ensemble $S = \{n_1; \ldots; n_\tau\}$ de A, non vide, avec $n_1 < \cdots < n_\tau$. On pose $n_0 = 0$ et $n_{\tau+1} = L$.

Rappelons que $\mathcal{I}(L)$ désigne l'ensemble des ponts irréductibles d'étendue L.

On définit aussi :

$\mathcal{I}_L(\leqslant \delta, S)$ l'ensemble des ponts irréductibles d'étendue L, tels qu'aucun points de S ne soit couvert par un retour d'étendue supérieure à δ,

$\mathcal{J}_L(S; \sigma_1, \ldots, \sigma_\tau)$ l'ensemble des ponts irréductibles d'étendue L tels que pour tout i entre 1 et τ, il y ait un retour d'étendue σ_i, couvrant n_i et aucun autre n_j, $j \neq i$.

On a à présent trois lemmes : le premier affirme que tout pont irréductible comporte soit beaucoup de petits retours, soit un nombre conséquent de grands retours. Les suivants majorent les fonctions génératrices de ces deux types de ponts.

Lemme A.4.

$$\mathcal{I}_L \subset \left(\bigcup_{S\subset A;|S|\geqslant k/2} \mathcal{I}_L(\leqslant \delta; S) \right) \bigcup \left(\bigcup_{S\subset A;|S|\geqslant 1} \bigcup_{\substack{\sigma_1>\delta;\ldots;\sigma_\tau>\delta \\ \sigma_1+\cdots+\sigma_\tau>k\delta/2}} \mathcal{J}_L(S;\sigma_1,\ldots,\sigma_\tau) \right).$$

Démonstration. On suppose $\omega \in \mathcal{I}_L \setminus \bigcup_{S\subset A;|S|\geqslant k/2} \mathcal{I}_L(\leqslant \delta; S)$.

Alors au moins $k/2$ points de A sont couverts par des retours d'étendue supérieure à δ. Certains peuvent cependant couvrir plus d'un point de A. On choisit donc τ minimal tel qu'il existe τ retours de ω, chacun d'étendue supérieure à δ, tels qu'au moins $k/2$ points de A soient couverts par au moins un de ces retours. Par minimalité de τ, pour chaque retour, il existe un point de A couvert par ce retour et aucun autre. Ceci donne les n_i (et σ_i) cherchés. □

Lemme A.5. $FG\left(\mathcal{I}_L(\leqslant \delta, S)\right) \leqslant \mathrm{e}^{-mL} \left(\chi\varepsilon(T)\right)^{\tau+1}$.

Démonstration. On pose

$$\begin{aligned} r_0 &= 0, \\ q_{\tau+1} &= |\omega|, \\ q_i &= \min\{j,\ \omega_1(j+1) = n_j + 1\}, \\ r_i &= \max\{j,\ \omega_1(j) = n_i\}. \end{aligned}$$

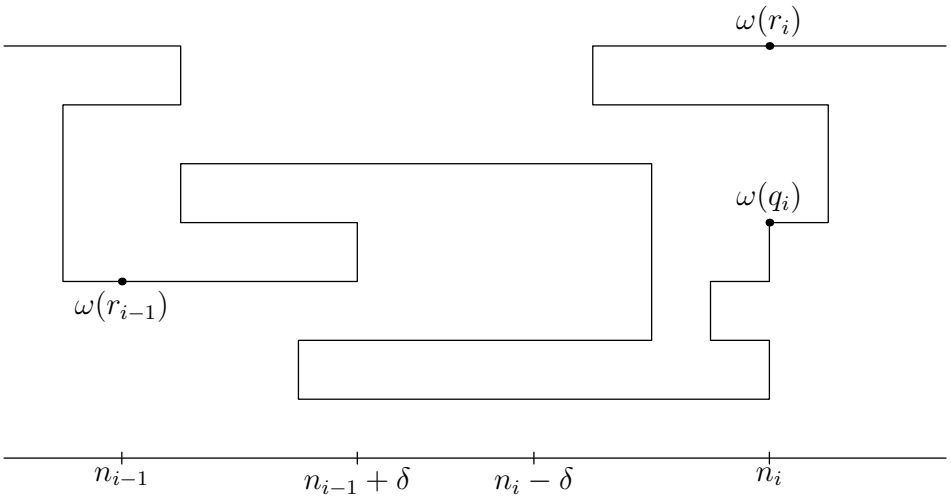


FIG. 4 – Illustration de la preuve du lemme A.5

On remarque que $q_i < r_i$ pour tout i (il n'y a pas de points de rupture). On a de plus $\omega_1(j) \geqslant n_i - \delta$ pour tout $j \geqslant q_i$ et $\omega_1(j) \leqslant n_i + \delta$ pour tout $j \leqslant r_i$.

En particulier, on obtient $r_{i-1} < q_i$ car $n_i - n_{i-1} > 2\delta$. De plus, $\omega[r_{i-1}, q_i]$ est un pont, et le reste de ω reste au-dehors de $\{x \in \mathbb{Z}^d,\ n_{i-1} - \delta < x_1 < n_i + \delta\}$.

En coupant ω aux points $q_1, r_1, \ldots, q_\tau, r_\tau$, on a $2\tau+1$ sous-chemins, et

$$\begin{aligned} FG\left(\mathcal{I}_L(\leqslant \delta, S)\right) &\leqslant \chi^\tau \prod_{i=1}^{\tau+1} B^*(n_i - n_{i-1}; \delta, n_i - n_{i-1} - 2\delta) \\ &\leqslant \chi^\tau \prod_{i=1}^{\tau+1} \mathrm{e}^{-m(n_i - n_{i-1})} \varepsilon(n_i - n_{i-1} - 2\delta) \\ &\leqslant \chi^{\tau+1} \prod_{i=1}^{\tau+1} \mathrm{e}^{-m(n_i - n_{i-1})} \varepsilon(T) \qquad (\text{car } \varepsilon \text{ décroît}) \\ &\leqslant \mathrm{e}^{-mL} \left(\chi \varepsilon(T)\right)^{\tau+1}. \end{aligned}$$

□

Lemme A.6. $FG\left(\mathcal{J}_L(S; \sigma_1, \ldots, \sigma_\tau)\right) \leqslant \mathrm{e}^{-mL} \chi^{2\tau} \mathrm{e}^{-m(\sigma_1 + \cdots + \sigma_\tau)}$.

Démonstration. Pour tout i entre 1 et τ, on a un retour $\omega[s_i, t_i]$ d'étendue σ_i tel que :

$$n_{i-1} < \omega_1(t_i) \leqslant n_i < \omega_1(s_i) \leqslant n_{i+1}.$$

Ceci implique que $t_i < s_{i+1}$ par la disjonction des retours.

On pose $l_0 = 0$ et :

$$\begin{aligned} f_i &= \min\{r > l_{i-1},\ \omega_1(r+1) = n_i + 1\}, \\ l_i &= \max\{r,\ \omega_1(r) = n_i\}. \end{aligned}$$

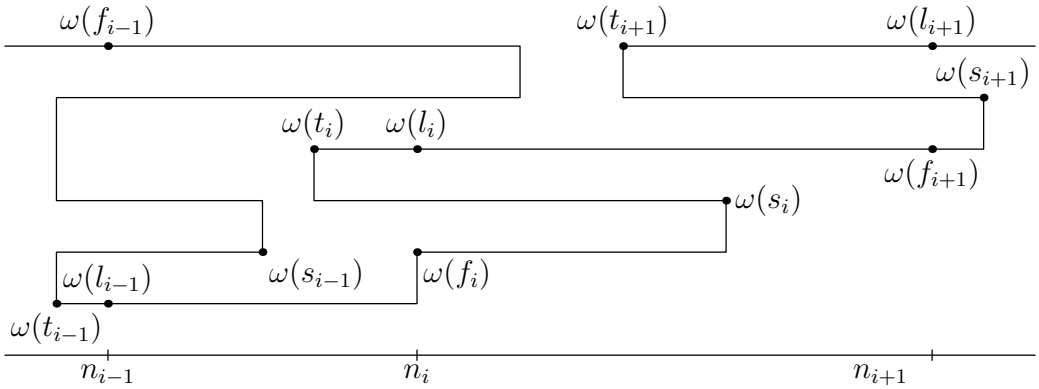


FIG. 5 – Illustration de la preuve du lemme A.6

Alors, on a $l_{i-1} < f_i < s_i < t_i \leqslant l_i$, et $\omega[l_{i-1}, f_i]$ est un pont d'étendue $n_i - n_{i-1}$. Si l'on découpe ω en chaque f_i, s_i, t_i et l_i, on obtient (on rappelle que $B(L)$ est la fonction génératrice des ponts détendue L, elle est de plus inférieure à e^{-mL}) :

$$\begin{aligned} FG\left(\mathcal{J}_L(S; \sigma_1, \ldots, \sigma_\tau)\right) &\leqslant \left(\prod_{i=1}^{\tau+1} B(n_i - n_{i-1})\right) \left(\prod_{i=1}^{\tau} \chi B(\sigma_i) \chi\right) \\ &\leqslant \mathrm{e}^{-mL} \chi^{2\tau} \mathrm{e}^{-m(\sigma_1 + \cdots + \sigma_\tau)}. \end{aligned}$$

□

On peut à présent démontrer le théorème selon lequel $m_\Lambda > m$.

Démonstration. On fixe T et δ tels que :

$$2\left(\chi\varepsilon(T)\right)^{\frac{1}{2}} < \frac{1}{2},$$
$$\left(1+\frac{\chi^2}{1-\mathrm{e}^{-m/2}}\right)\mathrm{e}^{-m\delta/4} < \frac{1}{2},$$

et on pose $Q = T + 2\delta$, et $k = \lfloor L/Q - 1 \rfloor$.

On obtient donc :

$$\Lambda_z(L) = FG(\mathcal{I}_L) \leqslant 2^k \mathrm{e}^{-mL}\left(\chi\varepsilon(T)\right)^{1+k/2} + \mathrm{e}^{-mL}\sum_{\tau=1}^{k}\binom{k}{\tau}\chi^{2\tau}\sum_{\substack{\sigma_1>\delta;\ldots;\sigma_\tau>\delta\\ \sigma_1+\cdots+\sigma_\tau>k\delta/2}}\mathrm{e}^{-m(\sigma_1+\cdots+\sigma_\tau)}.$$

Pour tout D et $r < m$,

$$\begin{aligned}\sum_{\substack{\sigma_1>\delta;\ldots;\sigma_\tau>\delta\\ \sigma_1+\cdots+\sigma_\tau>D}}\mathrm{e}^{-m(\sigma_1+\cdots+\sigma_\tau)} &\leqslant \sum_{\sigma_1>\delta;\ldots;\sigma_\tau>\delta}\mathrm{e}^{-m(\sigma_1+\cdots+\sigma_\tau)}\mathrm{e}^{r(\sigma_1+\cdots+\sigma_\tau-D)}\\ &\leqslant \left(\frac{\mathrm{e}^{(r-m)\delta}}{1-\mathrm{e}^{(r-m)}}\right)^{\tau}\mathrm{e}^{-rD}.\end{aligned}$$

Si l'on pose $r = m/2$ et $D = k\delta/2$, on a finalement :

$$\begin{aligned}\Lambda_z(L) &= FG(\mathcal{I}_L)\\ &\leqslant \mathrm{e}^{-mL}\left(\left[2\left(\chi\varepsilon(T)\right)^{1/2}\right]^k + \left[1+\frac{\chi^2\mathrm{e}^{-m\delta/2}}{1-\mathrm{e}^{-m/2}}\right]^k\mathrm{e}^{-m\delta k/4}\right)\\ &\leqslant \mathrm{e}^{-mL}\left(\left(\frac{1}{2}\right)^k + \left(\frac{1}{2}\right)^k\right)\\ &\leqslant 2\mathrm{e}^{-mL}2^{2-L/Q},\end{aligned}$$

et donc :

$$m_\Lambda = \liminf\frac{-\log\Lambda_z(L)}{L} \geqslant m + \frac{\log 2}{Q} > m. \qquad \square$$

B Démonstration du théorème du renouvellement

On montre ici le théorème du renouvellement :

Théorème B.1 (du renouvellement)**.** *Soient $(f_i)_{i\geqslant 1}$ et $(g_i)_{i\geqslant 0}$ deux suites positives, telles que l'on ait $0 < g = \sum_{n\geqslant 0} g_n < \infty$ et $f_1 > 0$. On pose $f = \sum_{n\geqslant 1} f_n$, et on définit la suite $(v_n)_{n\in\mathbb{N}}$ comme suit :*

$$\begin{cases} v_0 = g_0\\ \forall n > 0, v_n = \sum_{i=1}^{n} f_i v_{n-i} + g_n. \end{cases}$$

Alors :

(i) si $f < 1$, $\lim_{n\to\infty} v_n = 0$, et $\sum v_n = g/(1-f)$;
(ii) si $f = 1$, $\lim_{n\to\infty} v_n = g/(\sum k f_k)$, et donc $\sum v_n$ diverge ;
(iii) si $f > 1$, $\limsup_{n\to\infty} v_n^{1/n} > 1$.

Démonstration. On aura d'abord besoin du lemme suivant :

Lemme B.1. *Si $(\alpha_n)_{n\geq 0}$ et $(\beta_{n,k})_{m,k\geq 0}$ deux suites positives, telles que $\sum \alpha_n < \infty$ et qu'il existe B tel que pour tous m, k, $\beta_{m,k} < B$, alors*

$$\limsup_{k\to\infty} \sum_{n\geqslant 0} \alpha_n \beta_{n,k} \leqslant \sum_{n\geqslant 0} \alpha_n \limsup_{k\to\infty} \beta_{n,k}.$$

Ce résultat provient du lemme de Fatou, en remarquant que $B - \beta_{n,k} \geqslant 0$. □

Ce lemme établi, on pose :

$$F(s) = \sum_{n\geqslant 0} f_n s^n, \qquad G(s) = \sum_{n\geqslant 0} g_n s^n, \qquad V(s) = \sum_{n\geqslant 0} v_n s^n,$$

avec la convention évidente $f_0 = 0$. La définition de la suite $(v_n)_{n\geqslant 0}$ nous dit que (on peut effectivement permuter les sommes puisque les termes sont tous positifs)

$$V(s) = G(s) + F(s)V(s).$$

Voyons d'abord le point (iii).
Supposons donc $f > 1$ et $\limsup v_n^{1/n} \leqslant 1$. On a alors $V(s) < \infty$ dès que $|s| < 1$.
Or $V(s) \geqslant F(s)V(s)$ et donc $F(s) \leqslant 1$ pour $|s| < 1$. Par convergence monotone, on en déduit que $f = F(1) \leqslant 1$ ce qui est absurde.

Voyons maintenant les points restants.
Si $f \leqslant 1$, on a pour tout $n \geqslant 0$, $0 \leqslant v_n \leqslant \sum g_i$ par une récurrence immédiate, et donc $(v_n)_{n\geqslant 0}$ est une suite positive bornée. Ainsi, pour $|s| < 1$,

$$V(s) = \frac{G(s)}{1 - F(s)}.$$

En faisant tendre s vers 1, on obtient que, si $f < 1$, $V(1) = g/(1-f) < \infty$, ce qui est (i).

Reste à présent à voir le point (ii).
Supposons donc $f = 1$. En sommant les relations définissant v_n, on obtient

$$\sum_{n=0}^{N} v_n = \sum_{n=0}^{N} g_n + \sum_{n=1}^{N} \sum_{i=0}^{n} f_i v_{N-i}.$$

Si l'on pose à présent

$$r_n = 1 - \sum_{i=0}^{n} f_i = \sum_{i\geqslant n+1} f_i,$$

on obtient

$$\sum_{n=0}^{N} r_n v_{N-n} = \sum_{n=0}^{N} g_n.$$

On pose ensuite

$$r = \sum_{n\geqslant 0} r_n = \sum_{k\geqslant 0} k f_k.$$

Formellement, on obtient le résultat $\lim_{N\to\infty} v_N = g/r$ en faisant tendre N vers l'infini dans la formule précédente. La suite consiste à justifier le passage à la limite.

Posons $u = \limsup_{n\to\infty} v_n$, qui est fini puisque v_n est bornée. On prend $(v_{\varphi(n)})$ une sous-suite qui converge vers u. On va montrer que pour tout entier k, $\lim\limits_{n\to\infty} v_{\varphi(n)-k} = u$.

En effet, le lemme donne l'inégalité $\limsup_k \sum_n f_n v_{\varphi(n)-k} \leqslant \sum_n f_n \limsup_k v_{\varphi(n)-k} = \sum f_n u$ (appliqué à $\alpha_n = f_n$ et $\beta_{n,k} = v_{\varphi(n)-k}$). Si l'on pose $u_* = \liminf v_{\varphi(n)-1}$, cette inégalité, utilisée dans la formule

$$v_{\varphi(n)} - f_1 v_{\varphi(n)-1} = g_{\varphi(n)} + \sum_{i=2}^{\varphi(n)} f_i v_{\varphi(n)-i}$$

donne, en passant à la limsup,

$$u - f_1 u_* \leqslant 0 + \sum_{i \geqslant 2} f_i u.$$

Puisque $f_1 > 0$ et $f = 1$, on obtient $u \leqslant u_*$, et donc $u = u_*$. Le résultat est donc vrai pour $k = 1$, et donc pour tout k par une récurrence immédiate.

Si on remplace N par $\varphi(n)$ dans $\sum_{n=0}^{N} r_n v_{N-n} = \sum_{n=0}^{N} g_n$, le lemme de Fatou donne que

$$\sum_{n \geqslant 0} r_n u \leqslant g.$$

Si $r = \infty$, on a bien le résultat : $u = 0$.
Si $r < \infty$, on note $u' = \liminf v_n$, et on se donne $(v_{\psi(n)})$ qui tend vers u'. Si on remplace N par $\psi(n)$ dans la même équation que précédemment, et que l'on applique le lemme, en se souvenant que $u \geqslant \limsup_{n\to\infty} v_{\psi(n)-k}$, on obtient

$$r_0 u' + \sum_{n \geqslant 1} r_n u \geqslant g,$$

ce qui donne que $u' \geqslant u$ et donc $u' = u$.

Ainsi, v_n tend bien vers u et le théorème de convergence dominée appliqué à $\sum_{n=0}^{N} r_n v_{N-n} = \sum_{n=0}^{N} g_n$ donne bien $ru = g$.

□

Références

[HR17] G.H. Hardy and S. Ramanujan. Asymptotic formulae for the distribution of integers of various types. *Proc. Lond. Math. Soc. (2)*, 16 :112–132, 1917.

[HW85] J.M. Hammersley and S.G. Whittington. Self-avoiding walks in wedges. *J. Phys. A : Math. Gen. (2)*, 18 :101–111, 1985.

[MS93] N. Madras and G. Slades. *The self-avoiding walk*. Birkhaüser, 1993.

[Sta69] A. J. Stam. Renewal theory in r dimensions. *Compositio Mathematica*, 21 :383–399, 1969.

[Sta71] A. J. Stam. Renewal theory in r dimensions II. *Compositio Mathematica*, 23 :1–13, 1971.