Un Algorithme d'Analyse de Type Earley pour Grammaires à Concaténation d'Intervalles

Laura Kallmeyer[1] Wolfgang Maier[1] Yannick Parmentier[2]

(1) SFB 441 / Universität Tübingen, Nauklerstr. 35, D-72074 Tübingen
(2) LORIA / Nancy Université, Campus Scientifique, BP 239,
F-54506 Vandœuvre-Lès-Nancy Cedex

lk@sfs.uni-tuebingen.de wo.maier@uni-tuebingen.de parmenti@loria.fr

Résumé. Nous présentons ici différents algorithmes d'analyse pour grammaires à concaténation d'intervalles (*Range Concatenation Grammar, RCG*), dont un nouvel algorithme de type *Earley*, dans le paradigme de l'analyse déductive. Notre travail est motivé par l'intérêt porté récemment à ce type de grammaire, et comble un manque dans la littérature existante.

Abstract. We present several different parsing algorithms for Range Concatenation Grammar (RCG), *inter alia* an entirely novel *Earley*-style algorithm, using the deductive parsing framework. Our work is motivated by recent interest in range concatenation grammar in general and fills a gap in the existing literature.

Mots-clés : Analyse syntaxique déductive, grammaires à concaténation d'intervalles.

Keywords: Deductive parsing, range concatenation grammar.

1 Introduction

Les grammaires à concaténation d'intervalles (RCG) (Boullier, 2000) ont récemment reçu une attention particulière dans différents contextes de travail. Ceux-ci incluent la traduction automatique dirigée par la syntaxe (Søgaard, 2008), le développement de grammaires (Sagot, 2005) ou encore l'extraction de grammaires à partir de treebanks (Maier & Søgaard, 2008).

En outre, d'un point de vue formel, les RCGs sont intéressantes car elles génèrent exactement la classe des langages analysables en un temps polynômial (Bertsch & Nederhof, 2001). En particulier, elles sont plus puissantes que la classe des RCGs « simples », une sous-classe de RCG qui est équivalente aux systèmes de réécriture hors-contexte linéaires (*Linear Context-Free rewriting Systems, LCFRS*), aux grammaires d'arbres adjoints multi-composantes locales aux arbres (*Set-Local Multi-Component Tree-Adjoining Grammars, SL-MCTAG*) (Weir, 1988) et aux grammaires hors-contexte multiples (*Multiple Context-Free Grammars, MCFG*) (Seki *et al.*, 1991) (les preuves sont données par Boullier (1998)). Or, il a été démontré que cette dernière classe des RCGs simples est incapable de traiter certains phénomènes de la langue naturelle, comme par exemple la permutation d'arguments dans le « brouillage » (*scrambling*) des langues dites à ordre des mots libres (Becker *et al.*, 1992), ou encore les constructions non

semi-linéaires telles que l'empilement de cas en géorgien ancien (Michaelis & Kracht, 1996), et les nombres chinois (Radzinski, 1991), phénomènes que les RCGs peuvent traiter (Boullier, 1999).

Aussi, des algorithmes d'analyse pour RCG ont été présentés par (i) Boullier (2000), qui définit un algorithme d'analyse descendante directionnelle, et (ii) Barthélemy *et al.* (2001), qui ajoutent un oracle à l'algorithme de Boullier afin de limiter l'espace de recherche. La classe des RCGs simples a reçu une plus grande attention, comme en témoignent les travaux sur l'analyse à base d'automates de De La Clergerie (2002), ceux sur l'analyse déductive de LCFRS de Burden et Ljunglöf (2005), ou encore l'approche de Kanazawa (2008) pour l'analyse de MCFG au moyen de Datalog.

Dans cet article, nous ne prétendons pas motiver l'utilisation des RCGs (pour cela, se référer aux articles sus-cités), mais cherchons plutôt à combler un manque dans la littérature sur les algorithmes d'analyse pour RCG non-simples. Plus précisément, nous fournissons une formulation de l'analyse de la classe entière des RCGs dans le cadre de l'analyse par déduction. Nous présentons dans un même cadre des algorithmes descendants, des algorithmes de type *CYK*, et un algorithme de type *Earley*, facilitant ainsi une comparaison des différentes stratégies d'analyse. Pour profiter pleinement de cette comparaison, il est utile d'avoir des connaissances en analyse déductive, et notamment des règles de déduction du type de celles présentées par Shieber *et al.* (1995). L'article est structuré comme suit. En Section 2, nous introduisons les notions préliminaires nécessaires. En Section 3, nous introduisons les algorithmes descendants, en Section 4 les algorithmes de type *CYK* et en Section 5 un algorithme de type *Earley*. Nous terminons en donnant quelques chiffres sur l'efficacité relative des ces algorithmes.

2 Notions préliminaires

Les RCGs sont des grammaires dont les productions (appelées clauses) réécrivent des prédicats couvrant certaines parties de la phrase en d'autres prédicats. Par exemple, une clause de la forme $S(aXb) \rightarrow S(X)$ indique qu'un prédicat S est vrai pour une partie de la phrase si cette partie commence par un a et finit par un b, et si, de plus, S est également vrai pour la portion de phrase comprise entre a et b. Une clause $S(c) \rightarrow \epsilon$ indique que S est vrai pour tout terminal c, sans autre condition. La RCG ayant les deux clauses $S(aXb) \rightarrow S(X), S(c) \rightarrow \epsilon$ génère le langage $\{a^n c b^n \mid n \geq 0\}$.

Dans cet article, nous nous intéressons uniquement aux RCG positives car elles correspondent à la variante utilisée dans les travaux sus-mentionnés. Les RCG non-positives, appelées RCG négatives, permettent l'utilisation de prédicats négatifs de la forme $\overline{A(\alpha_1, \ldots, \alpha_n)}$. De tels prédicats permettent de reconnaître le langage complémentaire de celui généré au moyen des prédicats positifs correspondants (voir Boullier (2000) pour plus de détails).

Définition 1 (Grammaire à concaténation d'intervalle)**.** *Une* grammaire à concaténation d'intervalle *(RCG) est un 5-uplet $G = \langle N, T, V, P, S \rangle$. N est un ensemble fini de noms de prédicats, avec une fonction d'arité définie comme suit : dim: $N \rightarrow \mathbb{N} \setminus \{0\}$, T et V sont des ensembles finis de symboles terminaux et de variables respectivement. P est un ensemble fini de clauses de la forme $\psi_0 \rightarrow \psi_1 \ldots \psi_m$, où $m \geq 0$ et chacun des $\psi_i, 0 \leq i \leq m$, est un prédicat de la forme $A_i(\alpha_1, \ldots, \alpha_{dim(A_i)})$ avec $A_i \in N$ et $\alpha_j \in (T \cup V)^*$ pour $1 \leq j \leq dim(A_i)$. Comme raccourci de notation pour $A_i(\alpha_1, \ldots, \alpha_{dim(A_i)})$, nous utilisons $A_i(\vec{\alpha})$. $S \in N$ est le nom du prédicat de départ, tel que $dim(S) = 1$.*

Le concept d'intervalle est primordial pour les RCG. Nous définissons ce concept et celui de vecteur d'intervalle ci-dessous.

Définition 2 (Intervalle)**.** *Pour tout $w \in T^*$, où $w = w_1 \dots w_n$ avec $w_i \in T$ pour $1 \leq i \leq n$, nous définissons :*
- $Pos(w) := \{0, \dots, n\}$.
- *Une paire $\langle l, r \rangle \in Pos(w) \times Pos(w)$ avec $l \leq r$ est un* intervalle *dans w. Sa* production *$\langle l, r \rangle(w)$ est la sous-chaîne $w_{l+1} \dots w_r$.*
- *Pour deux intervalles $\rho_1 = \langle l_1, r_1 \rangle, \rho_2 = \langle l_2, r_2 \rangle$: si $r_1 = l_2$, alors $\rho_1 \cdot \rho_2 = \langle l_1, r_2 \rangle$; sinon $\rho_1 \cdot \rho_2$ n'est pas défini.*

Définition 3 (Vecteur d'intervalles)**.** *Pour un $w \in T^*$ donné, nous appelons un vecteur $\phi = (\langle x_1, y_1 \rangle, \dots, \langle x_k, y_k \rangle)$* vecteur d'intervalles *de dimension k dans w si $\langle x_i, y_i \rangle$ est un intervalle dans w pour $1 \leq i \leq k$. $\phi(i).l$ (resp. $\phi(i).r$) dénote alors le premier (resp. second) composant du i^e élément de ϕ, c'est-à-dire x_i (resp. y_i).*

Afin d'instancier une clause de la grammaire, nous avons besoin de déterminer les intervalles couverts par toutes les variables dans la clause, et de toutes les occurrences de symboles terminaux. Par souci de clarté, nous supposons que les variables d'une clause et les occurrences de symboles terminaux sont équipées d'indices distincts, commençant à 1, ordonnés de gauche à droite, et tels que, pour les variables, seule la première occurrence donne lieu à un nouvel indice. Nous introduisons alors une fonction $\Upsilon : P \to \mathbb{N}$ retournant l'indice maximal dans une clause. De plus, nous définissons $\Upsilon(c, x)$ pour une clause c et une variable ou une occurrence d'un terminal x donnés comme étant l'indice de x dans c.

Définition 4 (Instanciation de clause)**.** *Une* instanciation *d'une clause $c \in P$ avec $\Upsilon(c) = j$ par rapport à une chaîne w est donnée par le vecteur d'intervalles ϕ avec $dim(\phi) = j$. L'application de ϕ à un prédicat $A(\vec{\alpha})$ dans c associe toutes les occurrences de $x \in (T \cup V)$ avec $\Upsilon(c, x) = i$ dans $\vec{\alpha}$ à $\phi(i)$. Si le résultat est défini (c'est-à-dire, les images de variables adjacentes peuvent être concaténées), celui-ci est appelé un* prédicat instancié *et le résultat de l'application de ϕ à tous les prédicats dans c, s'il est défini, est appelé une* clause instanciée.

En plus des vecteurs d'intervalles, nous introduisons également des vecteurs de contraintes d'intervalle. Ceux-ci sont des vecteurs contenant des paires composées de variables d'extrémité d'intervalle, et un ensemble de contraintes sur ces variables.

Définition 5 (Vecteur de contraintes d'intervalles)**.** *Soit $V_r = \{r_1, r_2, \dots\}$ un ensemble de variables d'extrémité d'intervalle.*

Un vecteur de contraintes d'intervalles *de dimension k est une paire $\langle \vec{\rho}, C \rangle$ où*
- *$\rho \in (V_r^2)^k$; nous définissons $V_r(\rho)$ comme l'ensemble des variables d'extrémité d'intervalle apparaissant dans $\vec{\rho}$.*
- *C est un ensemble de contraintes c_r ayant l'une des formes suivantes :*
  $r_1 = r_2$, $k = r_1$, $r_1 + k = r_2$, $k \leq r_1$, $r_1 \leq k$, $r_1 \leq r_2$ *ou* $r_1 + k \leq r_2$ *pour* $r_1, r_2 \in V_r(\rho)$ *et* $k \in \mathbb{N}$.

Nous disons qu'un vecteur d'intervalles ϕ satisfait un vecteur de contraintes d'intervalles $\langle \rho, C \rangle$ ssi ϕ et ρ ont la même dimension k et qu'il existe une fonction $f : V_r \to \mathbb{N}$ associant $\rho(i).l$ à $\phi(i).l$ et $\rho(i).r$ à $\phi(i).r$ pour tout $1 \leq i \leq k$ telle que toutes les contraintes dans C soient satisfaites. De plus, nous disons qu'un vecteur de contraintes d'intervalles $\langle \rho, C \rangle$ est satisfiable ssi il existe un vecteur d'intervalles ϕ qui le satisfait.

Définition 6 (Vecteur de contraintes d'intervalles d'une clause)**.** *Pour toute clause c, nous définissons son* vecteur de contraintes d'intervalles $\langle \rho, C \rangle$ *par rapport à une chaîne w avec $|w| = n$ comme suit :*

- *ρ a pour dimension $\Upsilon(c)$, et toutes les variables d'extrémité d'intervalle dans ρ sont différentes deux à deux.*
- *Pour tout $\langle r_1, r_2 \rangle \in \rho$: $0 \leq r_1,\ r_1 \leq r_2,\ r_2 \leq n \in C$.*
  Pour toutes les occurrences x de terminaux dans c avec $i = \Upsilon(c, x)$: $\rho(i).l{+}1 = \rho(i).r \in C$.
  Pour toutes x, y qui sont des variables ou des occurrences de terminaux dans c telles que xy est une sous-chaîne d'un des arguments de c : $\rho(\Upsilon(c, x)).r = \rho(\Upsilon(c, y)).l \in C$.
  Ce sont toutes les contraintes dans C.

Intuitivement, un vecteur de contraintes d'intervalles d'une clause capture toute l'information sur les extrémités formant un intervalle, sur les intervalles contenant un unique terminal, et sur les occurrences de variables / terminaux adjacentes dans la clause.

Une dérivation RCG consiste en la réécriture de prédicats instanciés :

Définition 7 (Dérivation)**.** *Étant donné une RCG G et une chaîne d'entrée w, nous définissons la relation $\Rightarrow_{G,w}$ (appelée dérive) sur les chaînes des prédicats instanciés de la manière suivante. Soient Γ_1, Γ_2 des chaînes de prédicats instanciés. Si $A_0(\vec{\alpha_0}) \rightarrow A_1(\vec{\alpha_1}) \dots A_m(\vec{\alpha_m})$ est l'instanciation d'une clause $c \in P_G$, alors $\Gamma_1 A_0(\vec{\alpha_0}) \Gamma_2 \Rightarrow_{G,w} \Gamma_1 A_1(\vec{\alpha_1}) \dots A_m(\vec{\alpha_m}) \Gamma_2$.*

Intuitivement, si le membre de gauche d'une clause instanciée apparaît dans la chaîne de prédicats instanciés, il peut être remplacé par son membre de droite.

Définition 8 (Langage)**.** *Le* langage *d'une RCG G est l'ensemble des chaînes qui peuvent être réduites au mot vide : $L(G) = \{w \mid S(\langle 0, |w| \rangle) \stackrel{+}{\Rightarrow}_{G,w} \epsilon\}$.*

La capacité générative des RCGs dépasse celle des formalismes légèrement sensibles au contexte. Par exemple, on peut considérer la RCG $G = \langle \{S, eq\}, \{a\}, \{X, Y\}, P, S \rangle$ avec $P = \{S(XY) \rightarrow S(X)eq(X, Y), S(a) \rightarrow \epsilon,\ eq(aX, aY) \rightarrow eq(X, Y), eq(a, a) \rightarrow \epsilon\}$. Il est facile de constater que $L(G) = \{a^{2^n} \mid n \geq 0\}$. Ce langage n'est pas légèrement sensible au contexte car il n'a pas la propriété de croissance constante.

Par souci de clarté, nous supposons dans ce qui suit, sans perte de généralité, que les arguments vides (ϵ) apparaissent uniquement dans les clauses dont les membres de droite sont vides.[1]

3 Analyse descendante

Analyse descendante non-directionnelle L'idée de l'analyse descendante est d'instancier le prédicat de départ par rapport à la chaîne d'entrée toute entière et de vérifier récursivement s'il existe un moyen de réduire tous les prédicats des membres de droite à ϵ.

Les items ont la forme $[A, \phi, \mathit{flag}]$, où A est un prédicat, ϕ est un vecteur d'intervalles de dimension $dim(A)$ (contenant les intervalles avec lesquels sont instanciés les arguments de A) et *flag* $\in \{c, p\}$ indique si l'item a été complété ou prédit.

[1] Toute RCG peut être transformée en une RCG satisfaisant cette condition : pour cela, il suffit d'introduire un nouveau prédicat unaire Eps et une clause $Eps(\epsilon) \rightarrow \epsilon$, puis, pour chaque clause x dont le membre de droite n'est pas ϵ, de remplacer chaque argument ϵ qui apparaît dans c par une nouvelle variable X_ϵ et d'ajouter le prédicat $Eps(X_\epsilon)$ au membre de droite de c.

Comme axiome, nous prédisons S couvrant toute la chaîne d'entrée. Ainsi, la règle **initialisation** est la suivante :

$$\frac{}{[S, (\langle 0, n\rangle), p]}$$

L'opération **prédiction** prédit de nouveaux items à partir des items précédemment prédits :

$$\frac{[A_0, \phi, p]}{[A_1, \phi_1, p] \ldots [A_k, \phi_k, p]}$$

s'il existe une clause $A_0(\vec{x}_0) \rightarrow A_1(\vec{x}_1) \ldots A_k(\vec{x}_k)$ avec une instanciation ψ telle que $\psi(c) = A(\phi) \rightarrow A_1(\phi_1) \ldots A_k(\phi_k)$.

Puisque, à la différence de l'analyse descendante standard pour grammaires hors-contexte, nous commençons avec l'intégralité de la chaîne d'entrée à l'initialisation, nous avons besoin d'un moyen pour propager l'information sur les prédicats instanciés. Ceci se fait via un *flag* p/c, dont la valeur est donnée par les opérations de lecture et de complétion. L'opération de **lecture** met la valeur du flag à c pour un item décrivant un prédicat préalablement prédit :

$$\frac{[A, \phi, p]}{[A, \phi, c]}$$

s'il existe une clause $c = A(\vec{x}) \rightarrow \epsilon$ avec une instanciation ψ telle que $\psi(A(\vec{x})) = A(\phi)$. La règle de **complétion** met le flag à c pour un prédicat d'un membre de droite complété :

$$\frac{[A_0, \phi, p], [A_1, \phi_1, c] \ldots [A_k, \phi_k, c]}{[A_0, \phi, c]}$$

La reconnaissance réussit s'il existe un moyen de déclarer que le prédicat de départ est complété. Pour cela, nous utilisons un item **but** : $[S, (\langle 0, n\rangle), c]$.

Analyse descendante directionnelle L'algorithme ci-dessus peut être amélioré en évaluant les prédicats d'un membre de droite de manière ordonnée (de gauche à droite), et en interrompant l'évaluation dès qu'une instanciation de prédicat échoue. Cette variante correspond à l'algorithme présenté dans Boullier (2000).[2] Aussi, nous devons distinguer les *items passifs* des *items actifs*. Les items passifs ont les mêmes forme et signification que les items de l'algorithme non-directionnel présenté précédemment. Les items actifs permettent de déplacer un signet (appelé *dot*) le long d'un membre de droite d'une clause : $[A(\vec{x}) \rightarrow \Phi \bullet \Psi, \phi]$ où $A(\vec{x}) \rightarrow \Phi\Psi$ est une clause et ϕ est un vecteur d'intervalles de dimension $j = \Upsilon(A(\vec{x}) \rightarrow \Phi\Psi)$ qui fournit l'instanciation de la clause.

L'axiome est la prédiction du prédicat de départ couvrant toute la chaîne d'entrée. Ainsi la règle **initialisation** est la même que celle de l'algorithme non-directionnel. L'item **but** ne change pas non plus. Nous avons à présent deux opérations de prédiction. La première, **prédiction-règle**, prédit des items actifs avec leur dot au début de leur membre de droite, pour un item passif prédit donné :

$$\frac{[A, \psi, p]}{[A(\vec{x}) \rightarrow \bullet\Psi, \phi]} \text{ avec } \phi(A(\vec{x})) = A(\psi)$$

La seconde, **prédiction-préd**, prédit un item passif pour un prédicat suivant le dot dans un item actif :

$$\frac{[A(\vec{x}) \rightarrow \Phi \bullet B(\vec{y})\Psi, \phi]}{[B, \psi, p]} \text{ avec } \phi(B(\vec{y})) = B(\psi)$$

[2] Notons que cet algorithme n'est pas celui implanté dans le système SYNTAX développé par Boullier *et al.*

L'opération de **lecture** est la même que dans le cas non-directionnel. L'opération de **complétion** déplace le dot au-dessus d'un prédicat dans le membre de droite d'un item actif si l'item passif correspondant a été complété :

$$\frac{[B, \phi_B, c], [A(\vec{x}) \rightarrow \Phi \bullet B(\vec{y})\Psi, \phi]}{[A(\vec{x}) \rightarrow \Phi B(\vec{y}) \bullet \Psi, \phi]} \text{ avec } \phi(B(\vec{y})) = B(\phi_B)$$

Une fois que le dot a atteint la fin du membre de droite, nous pouvons effectuer une **conversion** de l'item actif en un item passif *complété* :

$$\frac{[A(\vec{x}) \rightarrow \Phi\bullet, \phi]}{[A, \psi, c]} \text{ avec } \phi(A(\vec{x})) = A(\psi)$$

Un problème évident de cet algorithme est que la règle **prédiction-règle** doit calculer toutes les instanciations possibles des clauses correspondant à un prédicat instancié donné. Prenons par exemple la RCG pour $\{a^{2^n} \mid n \geq 0\}$ de la Section 2. Si $w = aaaa$, à partir de $[S, (\langle 0, 4\rangle), p]$ **prédiction-règle** prédirait (entre autres) tous les items actifs $[S(X_1Y_2) \rightarrow \bullet S(X_1)eq(X_1, Y_2), (\langle 0, r\rangle, \langle r, 4\rangle)]$ avec $r \in \{0, 1, 2, 3, 4\}$. Le calcul de toutes les instanciations possibles est très coûteux, et va être évité dans l'algorithme de type *Earley* présenté en Section 5. En effet, dans ce dernier, nous utiliserons des vecteurs de contraintes d'intervalles (au lieu de vecteurs d'intervalles) pour ne prédire qu'un seul item actif :

$$[S(X_1Y_2) \rightarrow \bullet S(X_1)eq(X_1, Y_2), \langle\langle(r_1, r_2), (r_3, r_4)\rangle, \{0 = r_1, r_1 \leq r_2, r_2 = r_3, r_3 \leq r_4, 4 = r_4\}\rangle]$$

4 Analyse ascendante

Analyse CYK L'analyse CYK (Cocke, Younger, Kasami) est une technique d'analyse ascendante non-directionnelle. Les items ont la forme $[A, \phi]$ où A est un prédicat et ϕ un vecteur d'intervalles de dimension $dim(A)$.
Règle de **lecture**:

$$\frac{}{[A, \phi]}$$

s'il existe une clause $c = A(\vec{x}) \rightarrow \epsilon$ avec une instanciation ψ telle que $\psi(A(\vec{x})) = A(\phi)$.
Règle de **complétion**:

$$\frac{[A_1, \phi_1] \ldots [A_k, \phi_k]}{[A, \phi]}$$

où $A(\phi) \rightarrow A_1(\phi_1) \ldots A_k(\phi_k)$ est une clause instanciée.

L'item **but** est : $[S, (\langle 0, n\rangle)]$.

Analyse ascendante directionnelle Un désavantage évident de l'algorithme CYK est que, afin de réaliser une opération de *complétion*, tous les $A_1, \ldots, A_k$ du membre de droite doivent être vérifiés pour les items correspondants. Ce qui mène à un grand nombre d'indices qui doivent être vérifiés en même temps. Pour éviter cela, nous pouvons à nouveau déplacer un signet le long du membre de droite d'une clause. Comme dans le cas descendant directionnel, en plus des items ci-dessus (appelés items *passifs*), nous ajoutons des items *actifs*. Dans les items actifs, nous gardons une trace des positions déjà déterminées pour les extrémités gauche et droite des occurrences de variables et terminaux. Ceci est réalisé par l'enrichissement des items liés à la

Grammaire G : $S(XY) \rightarrow S(X)eq(X,Y)$, $S(a_1) \rightarrow \epsilon$, $eq(a_1X, a_2Y) \rightarrow eq(X,Y)$, $eq(a_1,a_2) \rightarrow \epsilon$
Items générés pour la chaîne $w = aa$ (les contraintes $0 \leq r_1, r_2 \leq n$ pour un intervalle $\langle r_1, r_2\rangle$ sont omises) :

	Item	Opération
1.	$[S, (\langle 0,1\rangle)]$	lecture $S(a_1) \rightarrow \epsilon$
2.	$[S, (\langle 1,2\rangle)]$	lecture $S(a_1) \rightarrow \epsilon$
3.	$[eq, (\langle 0,1\rangle, \langle 0,1\rangle)]$	lecture $eq(a_1,a_2) \rightarrow \epsilon$
4.	$[eq, (\langle 0,1\rangle, \langle 1,2\rangle)]$	lecture $eq(a_1,a_2) \rightarrow \epsilon$
5.	$[eq, (\langle 1,2\rangle, \langle 0,1\rangle)]$	lecture $eq(a_1,a_2) \rightarrow \epsilon$
6.	$[eq, (\langle 1,2\rangle, \langle 1,2\rangle)]$	lecture $eq(a_1,a_2) \rightarrow \epsilon$
7.	$[S(XY) \rightarrow \bullet S(X)eq(X,Y), \{X.l \leq X.r, X.r = Y.l, Y.l \leq Y.r\}]$	initialisation,
8.	$[eq(a_1X, a_2Y) \rightarrow \bullet eq(X,Y),$ $\{a_1.l + 1 = a_1.r, a_1.r = X.l, X.l \leq X.r, a_2.l + 1 = a_2.r, a_2.r = Y.l, Y.l \leq Y.r\}]$	initialisation
9.	$[S(XY) \rightarrow S(X) \bullet eq(X,Y), \langle \{\ldots, 0 = X.l, 1 = X.r\}]$	complétion 7. avec 1.
10.	$[S(XY) \rightarrow S(X) \bullet eq(X,Y), \{\ldots, 1 = X.l, 2 = X.r\}]$	complétion 7. avec 2.
11.	$[eq(a_1X, a_2Y) \rightarrow eq(X,Y)\bullet, \{\ldots, 1 = X.l, 2 = X.r, 1 = Y.l, 2 = Y.r\}]$	complétion 8. avec 6.
12.	$[S(XY) \rightarrow S(X)eq(X,Y)\bullet, \{\ldots, 0 = X.l, 1 = X.r, 1 = Y.l, 2 = Y.r\}]$	complétion 9. avec 4.
13.	$[eq, (\langle 0,2\rangle, \langle 0,2\rangle)]$	conversion 11.
14.	$[S, (\langle 0,1\rangle, \langle 1,2\rangle)]$	conversion 12.

FIG. 1 – Trace d'une analyse CYK directionnelle.

clause. Les items actifs ont la forme $[A(\vec{x}) \rightarrow \Phi \bullet \Psi, \langle \rho, C\rangle]$ avec $A(\vec{x}) \rightarrow \Phi\Psi$ une clause, $\Phi\Psi \neq \epsilon$, $\Upsilon(A(\vec{x} \rightarrow \Phi\Psi)) = j$ et $\langle \rho, C\rangle$ un vecteur de contraintes d'intervalles de dimension j. Nous imposons que $\langle \rho, C\rangle$ doit être satisfiable. Les items qui diffèrent uniquement par une bijection des variables d'intervalles sont considérés comme équivalents.

La règle de **lecture** et celle de **but** sont les mêmes que dans le cas non-directionnel. En outre, la règle **initialisation** introduit des clauses dont le dot est au début du membre de droite :

$$\frac{}{[A(\vec{x}) \rightarrow \bullet\Phi, \langle \rho, C\rangle]}$$

$A(\vec{x}) \rightarrow \Phi$ étant une clause avec pour vecteur de contraintes d'intervalles $\langle \rho, C\rangle, \Phi \neq \epsilon$.

La règle de **complétion** déplace le dot au-dessus d'un prédicat dans le membre de droite d'un item actif si l'item passif correspondant a été complété :

$$\frac{\begin{array}{c}[B, \phi_B], \\ [A(\vec{x}) \rightarrow \Phi \bullet B(x_1 ... y_1, ..., x_k ... y_k)\Psi, \langle \rho, C\rangle]\end{array}}{[A(\vec{x}) \rightarrow \Phi B(x_1 ... y_1, ..., x_k ... y_k) \bullet \Psi, \langle \rho, C'\rangle]}$$

où $C' = C \cup \{\phi_B(j).l = \rho(\Upsilon(x_j)).l, \phi_B(j).r = \rho(\Upsilon(y_j)).r \mid 1 \leq j \leq k\}$. Notons que les conditions sur les items nécessitent que le nouvel ensemble de contraintes pour ρ soit satisfiable.

La règle de **conversion** convertit un item actif dont le dot est à la fin du membre de droite en un item passif complété :

$$\frac{[A(\vec{x}) \rightarrow \Psi\bullet, \langle \rho, C\rangle]}{[A, \phi]}$$

s'il existe une instanciation ψ de $A(\vec{x}) \rightarrow \Psi$ qui satisfasse $\langle \rho, C\rangle$ tel que $\psi(A(\vec{x})) = A(\phi)$.

Un exemple de trace d'exécution est donné en Fig. 1. Par souci de clarté, au lieu de variables d'intervalles, nous utilisons $X.l$ (resp. $X.r$) pour l'extrémité gauche (resp. droite) de l'intervalle associé à X.

5 Algorithme de type Earley

Règles de déduction Nous ajoutons à présent une opération de prédiction à l'algorithme CYK avec items actifs, ce qui conduit à un algorithme de type Earley. Les items passifs sont enrichis avec un flag additionnel qui peut prendre les valeur p ou c selon que l'item est prédit ou complété. De plus, ils contiennent un vecteur de contraintes d'intervalles puisque, lorsqu'on prédit un prédicat, les extrémités gauche et droite de ses arguments peuvent ne pas être connues.

Les items passifs ont soit la forme $[A, \langle \rho, C\rangle, p]$ pour les items prédits, où $\langle \rho, C\rangle$ est un vecteur de contraintes d'intervalles de dimension $dim(A)$, soit la forme $[A, \phi, c]$ pour les items complétés où ϕ est un vecteur d'intervalles de dimension $dim(A)$. Les items actifs sont les mêmes que dans le cas CYK. L'axiome est la prédiction d'un prédicat S couvrant la chaîne d'entrée, c'est-à-dire, la règle **initialisation** est la suivante :

$$\frac{}{[S, \langle(\langle r_1, r_2\rangle), \{0 = r_1, n = r_2\}\rangle, p]}$$

Nous avons deux opérations de prédiction. La première, **prédiction-règle**, prédit des items actifs avec le dot au début de leur membre de droite, pour un item passif prédit donné :

$$\frac{[A, \langle \rho, C\rangle, p]}{[A(x_1 \dots y_1, \dots, x_k \dots y_k) \rightarrow \bullet\Psi, \langle \rho', C'\rangle]}$$

où $\langle \rho', C'\rangle$ est obtenu à partir du vecteur de contraintes d'intervalles de la clause $A(x_1 \dots y_1, \dots, x_k \dots y_k) \rightarrow \Psi$ en prenant toutes les contraintes de C, en associant tous les $\rho(i).l$ à $\rho'(\Upsilon(x_i)).l$ et tous les $\rho(i).r$ à $\rho'(\Upsilon(y_i)).r$, et en ajoutant les contraintes résultant au vecteur de contraintes de la clause. La seconde opération, **prédiction-pred**, prédit un item passif pour le prédicat suivant le dot dans un item actif :

$$\frac{[A(...) \rightarrow \Phi \bullet B(x_1 ... y_1, ..., x_k ... y_k)\Psi, \langle \rho, C\rangle]}{[B, \langle \rho', C'\rangle, p]}$$

où $\rho'(i).l = \rho(\Upsilon(x_i)).l$, $\rho'(i).r = \rho(\Upsilon(y_i)).r$ pour tout $1 \leq i \leq k$ et $C' = \{c \,|\, c \in C, c$ ne contient que des variables d'intervalles de $\rho'\}$. L'opération **lecture** peut être appliquée si un prédicat prédit peut être dérivé par une ϵ-clause :

$$\frac{[A, \langle \rho, C\rangle, p]}{[A, \phi, c]}$$

s'il existe une clause $A(\vec{x}) \rightarrow \epsilon$ avec une instanciation possible ψ qui satisfasse $\langle \rho, C\rangle$ telle que $\psi(A(\vec{x})) = A(\phi)$.

Finalement, les règles de **complétion**, de **conversion** sont celles de l'algorithme CYK avec items actifs à ceci près que nous ajoutons des flags c aux items passifs apparaissant dans ces règles. L'item **but** est le même que précédemment. La Fig. 2 illustre cet algorithme avec la RCG et la chaîne d'entrée définies dans l'exemple de la Fig. 1.

Correction et complétude Il est aisé de constater que l'algorithme de type Earley est à la fois correct et complet. Plus précisément, si un item complété est généré, alors le prédicat correspondant peut être dérivé : $[A, \psi, c] \Rightarrow A(\psi)$. De plus, si on peut dériver un constituant $A(\psi)$, alors on peut également générer l'item correspondant. Soit Γ une chaîne de prédicats instanciés. Alors $S(\langle 0, n\rangle) \stackrel{*}{\Rightarrow}_l A(\psi)\Gamma \stackrel{*}{\Rightarrow}_l \Gamma$ ssi $[A, \psi, c]$ où $\stackrel{*}{\Rightarrow}_l$ signifie "dérivation plus à gauche". En particulier, $[S, (\langle 0, n\rangle), c]$ ssi $S(\langle 0, n\rangle) \stackrel{*}{\Rightarrow} \epsilon$.

	Item	Opération
1	$[S, \langle(\langle r_1, r_2\rangle), \{0 = r_1, r_1 \leq r_2, 2 = r_2\}\rangle, p]$	initialisation
2	$[S(XY) \rightarrow \bullet S(X)eq(X,Y), \{X.l \leq X.r, X.r = Y.l, Y.l \leq Y.r, 0 = X.l, 2 = Y.r\}]$	prédiction-règle de 1
3	$[S, \langle(\langle r_1, r_2\rangle), \{0 = r_1, r_1 \leq r_2\}\rangle, p]$	prédiction-pred de 2
4	$[S, (\langle 0, 1\rangle), c]$	lecture de 3
5	$[S(XY) \rightarrow \bullet S(X)eq(X,Y), \{X.l \leq X.r, X.r = Y.l, Y.l \leq Y.r, 0 = X.l, \}]$	prédiction-règle de 3
6	$[S(XY) \rightarrow S(X) \bullet eq(X,Y), \{\ldots, 0 = X.l, 2 = Y.r, 1 = X.r\}]$	complét. de 2 avec 4
7	$[S(XY) \rightarrow S(X) \bullet eq(X,Y), \{X.l \leq X.r, X.r = Y.l, Y.l \leq Y.r, 0 = X.l, 1 = X.r\}]$	complét. de 5 avec 4
8	$[eq, \langle(\langle r_1, r_2\rangle, \langle r_3, r_4\rangle\rangle), \{r_1 \leq r_2, r_2 = r_3, r_3 \leq r_4, 0 = r_1, 2 = r_4, 1 = r_2\}\rangle]$	prédiction-pred de 6
9	$[eq(a_1X, a_2Y) \rightarrow \bullet eq(X,Y), \{a_1.l + 1 = a_1.r, a_1.r = X.l, X.l \leq X.r,$ $a_2.l + 1 = a_2.r, a_2.r = Y.l, Y.l \leq Y.r, X.r = a_2.l, 0 = a_1.l, 1 = X.r, 2 = Y.r\}]$	prédiction-règle de 8
	$\ldots$	
10	$[eq, (\langle 0, 1\rangle, \langle 1, 2\rangle), c]$	lecture 8
11	$[S(XY) \rightarrow S(X)eq(X,Y)\bullet, \{\ldots, 0 = X.l, 2 = Y.r, 1 = X.r, 1 = Y.l\}]$	complét. de 6 avec 10
12	$[S, (\langle 0, 2\rangle), c]$	conversion 11

FIG. 2 – Trace d'une analyse de type Earley pour la chaîne aa.

Obtenir une forêt d'analyse Jusqu'ici, nous avons décrit des reconnaisseurs et non des analyseurs. Cependant, chaque fois qu'une conversion est réalisée, une clause complètement instanciée a été trouvée. En collectant ces clauses, nous obtenons une représentation compacte de la forêt. En partant d'un prédicat S couvrant toute la chaîne d'entrée, nous pouvons extraire directement les analyses de cette représentation.

Complexité Il est clair que tous les algorithmes présentés ici sont polynômiaux par rapport à la taille de la chaîne d'entrée. Un grand facteur de complexité avec RCG, comme l'a montré Boullier (2000), est le nombre maximal de variables d'intervalles apparaissant dans un argument d'un prédicat. Dans notre approche à la Earley, nous essayons de retarder au maximum le calcul des valeurs que ces intervalles peuvent prendre, afin de réduire l'espace de recherche. Afin d'avoir une idée de l'efficacité de notre approche, nous donnons une évaluation du coût relatif des algorithmes directionnel descendant et de type Earley. Ces algorithmes ont été testés sur différents mots du langage $L = \{a^{2^n} | n \leq 0\}$. La table ci-dessous donne le nombre d'items générés. Nous constatons que la propagation de contraintes d'intervalles augmente la quantité d'information transportée dans un item, et ainsi diminue grandement le nombre d'items.[3]

Mots	**Earley**	**Descendant**	**Mots**	**Earley**	**Descendant**
a^2	15	21	a^{16}	100	539
a^8	55	164	a^{32}	185	1894

6 Conclusion

Nous avons présenté différents algorithmes d'analyse pour la classe entière des RCGs, au moyen du paradigme de l'analyse par déduction. Seul l'algorithme descendant directionnel avait été présenté jusqu'alors. La différence cruciale entre cet algorithme et notre algorithme de type Earley est que, alors que le premier calcule toutes les instanciations de clause lors des opérations de prédiction, le second évite cela en utilisant une technique de mise à jour dynamique d'un ensemble de contraintes sur les extrémités des intervalles. Les expériences menées montrent que l'algorithme de type Earley génère bien moins d'items, ce qui confirme que la propagation de contraintes d'intervalles est une méthode viable pour un calcul paresseux des intervalles.

[3]Bien entendu, la présence de contraintes rend la comparaison entre items plus complexe, nécessitant ainsi l'utilisation de représentations de bas niveau et de techniques efficaces de résolution de contraintes.

Références

BARTHÉLEMY F., BOULLIER P., DESCHAMP P. & DE LA CLERGERIE É. (2001). Guided parsing of Range Concatenation Languages. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, p. 42–49.

BECKER T., RAMBOW O. & NIV M. (1992). *The Derivationel Generative Power of Formal Systems or Scrambling is Beyond LCFRS*. Technical Report IRCS-92-38, Institute for Research in Cognitive Science, University of Pennsylvania.

BERTSCH E. & NEDERHOF M.-J. (2001). On the complexity of some extensions of RCG parsing. In *Proceedings of the Seventh International Workshop on Parsing Technologies*, p. 66–77, Beijing, China.

BOULLIER P. (1998). *Proposal for a Natural Language Processing Syntactic Backbone*. Rapport de Recherche RR-3342, Institut National de Recherche en Informatique et en Automatique, Le Chesnay, France.

BOULLIER P. (1999). Chinese numbers, mix, scrambling, and range concatenation grammars. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL'99)*, p. 53–60, Bergen, Norway.

BOULLIER P. (2000). Range concatenation grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, p. 53–64, Trento, Italy.

BURDEN H. & LJUNGLÖF P. (2005). Parsing linear context-free rewriting systems. In *Proceedings of the Ninth International Workshop on Parsing Technology*, p. 11–17, Vancouver, British Columbia.

KANAZAWA M. (2008). A prefix-correct earley recognizer for multiple context-free grammars. In *Proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+9)*, p. 49–56, Tübingen, Germany.

MAIER W. & SØGAARD A. (2008). Treebanks and mild context-sensitivity. In *Proceedings of the 13th Conference on Formal Grammar 2008*, Hamburg, Germany.

MICHAELIS J. & KRACHT M. (1996). Semilinearity as a Syntactic Invariant. In *Logical Aspects of Computational Linguistics*, Nancy.

RADZINSKI D. (1991). Chinese number-names, tree adjoining languages, and mild context-sensitivity. *Computational Linguistics*, **17**, 277–299.

SAGOT B. (2005). Linguistic facts as predicates over ranges of the sentence. In *Proceedings of LACL 05*, number 3492 in Lecture Notes in Computer Science, p. 271–286, Bordeaux, France.

SEKI H., MATSUMURA T., FUJII M. & KASAMI T. (1991). On multiple context-free grammars. *Theoretical Computer Science*, **88**, 191–229.

SHIEBER S. M., SCHABES Y. & PEREIRA F. C. N. (1995). Principles and implementation of deductive parsing. *Journal of Logic Programming*, **24**(1& 2), 3–36.

SØGAARD A. (2008). Range concatenation grammars for translation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, Manchester, England.

VILLEMONTE DE LA CLERGERIE E. (2002). Parsing mildly context-sensitive languages with thread automata. In *Proceedings of the 19th International Conference on Computational Linguistics*, p. 1–7, Taipei, Taiwan.

WEIR D. J. (1988). *Characterizing mildly context-sensitive grammar formalisms*. PhD thesis, University of Pennsylvania, Philadelphia, PA.