Analyse Syntaxique par Contraintes pour les Grammaires de Propriétés à Traits

Denys Duchier, Thi-Bich-Hanh Dao, Yannick Parmentier

Analyse Syntaxique par Contraintes pour les Grammaires de Propriétés à Traits

Denys Duchier **Thi-Bich-Hanh Dao** **Yannick Parmentier**

Laboratoire d'Informatique Fondamentale d'Orléans – Université d'Orléans
Bâtiment 3IA, Rue Léonard de Vinci – 45067 Orléans Cedex 2
prenom.nom@univ-orleans.fr

Résumé

Les Grammaires de Propriétés (GP) constituent un formalisme à base de contraintes capable de décrire la syntaxe, à la fois d'énoncés bien-formés et d'énoncés agrammaticaux. Duchier *et al.* (2010) proposent une sémantique formelle des GP en théorie des modèles et modélisent l'analyse syntaxique en GP sous la forme d'un Problème de Satisfaction de Contraintes (CSP). Dans cet article, nous présentons une extension de ces travaux afin de manipuler de nouveaux types de propriétés, qui permettent d'exprimer des relations entre constituants syntaxiques sous forme de contraintes sur structures de traits (*e.g.* la propriété d'*accord*). Nous décrivons ensuite une extension du modèle CSP pour traiter ces nouveaux types de propriétés lors de l'analyse syntaxique.

1 Introduction

Une des tâches spécifiques du Traitement Automatique des Langues est l'analyse syntaxique. Cette tâche a pour but de construire, à partir d'une description formelle de la syntaxe de la langue naturelle (*i.e.*, une grammaire), une structure exprimant les relations entre les divers constituants d'un énoncé. Cette structure prend généralement une forme arborescente, on parle alors d'arbre syntaxique.

De nombreux formalismes grammaticaux ont été proposés pour décrire la syntaxe de la langue naturelle, généralement en se basant sur des systèmes de réécriture. Un problème majeur de ces formalismes est leur manque de robustesse. En effet, ils ne permettent pas d'analyser des énoncés qui sont mal-formés, même lorsqu'il s'agit d'erreurs mineures. Les grammaires formelles de type *syntaxe fondée sur la théorie des modèles*, qui se concentrent sur une validation de modèles en terme de contraintes satisfaites, par contre, sont naturellement adaptées au traitement de *quasi-expressions* (*i.e.*, d'énoncés agrammaticaux produits par l'Homme).

Blache [1] a proposé le formalisme des Grammaires de Propriétés (GP), qui est un formalisme à base de contraintes, pour décrire à la fois des énoncés grammaticaux et agrammaticaux. Dans [2], Duchier *et al.* proposent une sémantique formelle des GP en théorie des modèles et modélisent l'analyse syntaxique en GP sous la forme d'un Problème de Satisfaction de Contraintes (CSP). Cependant, ces travaux ne permettent d'exprimer que des contraintes entre constituants syntaxiques relativement grossières (*e.g.*, exclusion mutuelle entre constituants de certains types, *etc.*). Ces contraintes ne permettent pas d'exprimer des restrictions plus fines, c'est-à-dire des contraintes sur des structures de traits, telles que l'accord entre constituants par exemple. Dans notre travail, nous étendons la sémantique des GP pour pouvoir exprimer ces contraintes sur des structures de traits dans un cadre formel en théorie des modèles. En plus de la définition formelle de nouveaux types de contraintes, notre travail aborde l'extension de la modélisation de l'analyse syntaxique sous forme de CSP de Duchier *et al.* afin de traiter ces nouvelles contraintes.

2 Grammaires de Propriétés (GP)

Les grammaires de propriétés [1] constituent un formalisme permettant de décrire la langue naturelle en terme de contraintes locales, appelées *propriétés*. Ces propriétés peuvent être violées indépendamment les unes les autres, ce qui rend possible la description d'énoncés agrammaticaux et également une notion de grammaticalité (ratio entre propriétés violées et

propriétés vérifiées). En première approximation, nous pouvons considérer des propriétés de la forme $A : \psi$, spécifiant que, pour chaque nœud de catégorie A, la contrainte ψ s'applique sur les nœuds fils de A (notés B, C ci-dessous). La contrainte ψ a l'une des formes suivantes :

Obligation	$A : \triangle B$	au moins un B
Unicité	$A : B!$	au plus un B
Linéarité	$A : B \prec C$	B précède C
Exigence	$A : B \Rightarrow C$	si $\exists$ B, alors $\exists$ C
Exclusion	$A : B \not\Rightarrow C$	pas B et C
Constituence	$A : S?$	tout enfant $\in S$
Accord	$A : B \rightsquigarrow C$	accord entre B et C

En pratique, la description des langues naturelles n'assigne pas une catégorie atomique à un constituant syntaxique, mais plutôt, une structure de traits (appelée aussi matrice attributs-valeurs), contenant des informations diverses telles que le genre, le temps, *etc.* Dans un arbre syntaxique, chaque nœud est donc étiqueté non pas par une unique catégorie, mais par un ensemble de *traits* (incluant un trait *catégorie*). Pour avoir une description fine sous forme de propriétés, nous devons donc étendre le formalisme pour imposer des contraintes entre traits. Commençons par décrire formellement ces structures de traits.

Soit $\mathcal{F}$ un ensemble fini de traits $\{f_1, \ldots, f_n\}$, où chaque f_i prend sa valeur dans un semi-treillis borné D_i. On note $\top_i$ pour le plus grand élément de D_i. On suppose que parmi les traits f_i, il existe un trait `cat` désignant la catégorie syntaxique. On considère des matrices attributs-valeurs (AVM) de type $\mathcal{M} = [f_1{:}D_1, \ldots, f_n{:}D_n]$. On note $M|_{f_i}$ pour la valeur du trait f_i dans M. Les AVMs forment aussi un semi-treillis avec l'ordre $M_1 \sqsubseteq M_2$ ssi $\forall i$, $M_1|_{f_i} \sqsubseteq M_2|_{f_i}$. On omettra f_i si sa valeur est $\top_i$. Une expression-AVM est une AVM dans laquelle des traits peuvent être associés à des variables partagées (contraintes de co-référence). Dans ce contexte, si S est une expression-AVM (*i.e.*, une AVM contenant des traits dont les valeurs sont liées par l'utilisation de variables partagées, notées dans ce qui suit $X, Y, \ldots$), alors S^v est une AVM obtenue en remplaçant, dans S, chaque occurrence de $f_i{:}X$ par $f_i{:}\top_i$. Cette notion de valeur d'AVM S^v nous sera utile dans les cas où l'on souhaite ignorer les contraintes de co-référence (voir Section 3). De plus, si S_0, S_1, S_2 sont des expressions-AVM, alors $E(S_0, S_1, S_2)$ désigne l'ensemble des équations de co-référence $(S_i|_f{:}X, S_j|_g{:}X)$, que nous notons $(i, f) \doteq (j, g)$, pour tout attribut f dans S_i et g dans S_j partageant une variable X, avec $0 \leq i \leq 2$, $0 \leq j \leq 2$, et $f, g \in \mathcal{F}$. Ce concept d'ensemble d'équations $E(S_0, S_1, S_2)$ nous sera utile dans les cas où l'on souhaite imposer des contraintes de co-référence entre traits appartenant à des triplets d'AVMs S_0, S_1, S_2 (voir Section 3).

À partir de cette définition des structures de traits, nous pouvons étendre les propriétés de GP définies ci-dessus respectivement comme suit : $S_0 : \triangle S_1$, $S_0 : S_1!$, $S_0 : S_1 \prec S_2$, $S_0 : S_1 \Rightarrow S_2$, $S_0 : S_1 \not\Rightarrow S_2$, $S_0 : s_1?$, $S_0 : S_1 \rightsquigarrow S_2$, où S_0, S_1, S_2 sont des expressions-AVM (et s_1 un ensemble d'expressions-AVM). Pour illustrer ces propriétés à traits, prenons l'exemple de la propriété d'accord. Celle-ci peut être utilisée pour modéliser, au sein d'un groupe verbal, l'accord en genre, nombre et personne entre le participe passé du verbe et un complément d'objet direct lorsque celui-ci est réalisé avant le verbe (*je l'ai aimée*) :

$$\texttt{GV} : \texttt{V}\begin{bmatrix}\text{mode} & \text{part-passé}\\ \text{genre} & \text{X}\\ \text{nombre} & \text{Y}\\ \text{pers} & \text{Z}\end{bmatrix} \rightsquigarrow \texttt{Pro}\begin{bmatrix}\text{cas} & \text{COD}\\ \text{genre} & \text{X}\\ \text{nombre} & \text{Y}\\ \text{pers} & \text{Z}\end{bmatrix}$$

Dans cet exemple, le trait `cat` (de valeur `GV`, `V` et `Pro`) est extrait des matrices afin de respecter les conventions de notation en Grammaires de Propriétés.

Soit $\mathcal{W}$ un ensemble de *mots*. Un lexique est un sous-ensemble de $\mathcal{W} \times \mathcal{M}$ (un lexique associe donc à chaque mot une ou plusieurs AVMs). Une *grammaire de propriétés* G est un couple (P_G, L_G), où P_G est un ensemble de propriétés et L_G un lexique.

3 Sémantique formelle des GP

Nous allons maintenant étendre la spécification formelle en théorie des modèles des GP proposées par Duchier *et al.* [3] afin de tenir compte des propriétés à traits introduites précédemment.

Classe de modèles. Comme dans [3], la sémantique des grammaires de propriétés est donnée par une interprétation sur la structure des arbres syntaxiques τ. Cette structure est définie comme suit. Nous notons $\mathbb{N}_0$ l'ensemble $\mathbb{N}\backslash\{0\}$. Un *domaine d'arbre* D est un sous-ensemble fini de $\mathbb{N}_0^*$ (*i.e.*, l'ensemble des mots construits sur le vocabulaire $\mathbb{N}_0$) clos pour les préfixes et les frères-gauches. En d'autres termes, $\forall \pi, \pi' \in \mathbb{N}_0^*, \forall i, j \in \mathbb{N}_0$:

$$\begin{array}{lcl} \pi\pi' \in D & \Rightarrow & \pi \in D \\ i < j \wedge \pi j \in D & \Rightarrow & \pi i \in D \end{array}$$

Un arbre syntaxique $\tau = (D_\tau, L_\tau, R_\tau)$ consiste en un domaine d'arbre D_τ, une fonction d'étiquetage $L_\tau : D_\tau \to \lfloor \mathcal{M} \rfloor$ associant à chaque nœud une AVM *minimale* (pour $\sqsubseteq$) et une fonction $R_\tau : D_\tau \to \mathcal{W}^*$ associant à chaque nœud sa réalisation surfacique. Pour des raisons pratiques, nous définissons aussi la fonction

d'arité $A_\tau : D_\tau \to \mathbb{N}$ comme suit, $\forall \pi \in D_\tau$:

$$A_\tau(\pi) = \max\{0\} \cup \{i \in \mathbb{N}_0 \mid \pi i \in D_\tau\}$$

Instances de propriété. Comme dans [3], nous définissons des instances de propriété p sur un arbre τ (notées $\mathcal{I}_\tau[\![p]\!]$), correspondant à des paires $X @ Y$, où X est une propriété telle que présentée en Section 2, et Y un n-uplet de nœuds (*i.e.*, de chemins) sur lesquelles s'applique la propriété (voir Fig. 1).

$$\begin{aligned}
\mathcal{I}_\tau[\![G]\!] &= \cup\{\mathcal{I}_\tau[\![p]\!] \mid \forall p \in P_G\} \\
\mathcal{I}_\tau[\![S_0 : S_1 \prec S_2]\!] &= \{(S_0 : S_1 \prec S_2)@\langle \pi, \pi i, \pi j\rangle \mid \forall \pi, \pi i, \pi j \in D_\tau, i{\neq}j\} \\
\mathcal{I}_\tau[\![S_0 : \triangle S_1]\!] &= \{(S_0 : \triangle S_1)@\langle \pi\rangle \mid \forall \pi \in D_\tau\} \\
\mathcal{I}_\tau[\![S_0 : S_1!]\!] &= \{(S_0 : S_1!)@\langle \pi, \pi i, \pi j\rangle \mid \forall \pi, \pi i, \pi j \in D_\tau, i{\neq}j\} \\
\mathcal{I}_\tau[\![S_0 : S_1 \Rightarrow S_2]\!] &= \{(S_0 : S_1 \Rightarrow S_2)@\langle \pi, \pi i\rangle \mid \forall \pi, \pi i \in D_\tau\} \\
\mathcal{I}_\tau[\![S_0 : S_1 \not\Rightarrow S_2]\!] &= \{(S_0 : S_1 \not\Rightarrow S_2)@\langle \pi, \pi i, \pi j\rangle \mid \forall \pi, \pi i, \pi j \in D_\tau, i{\neq}j\} \\
\mathcal{I}_\tau[\![S_0 : s_1?]\!] &= \{(S_0 : s_1?)@\langle \pi, \pi i\rangle \mid \forall \pi, \pi i \in D_\tau\} \\
\mathcal{I}_\tau[\![S_0 : S_1 \rightsquigarrow S_2]\!] &= \{(S_0 : S_1 \rightsquigarrow S_2)@\langle \pi, \pi i, \pi j\rangle \mid \forall \pi, \pi i, \pi j \in D_\tau, i{\neq}j\}
\end{aligned}$$

FIGURE 1 – Instances de propriété d'une grammaire G sur un arbre τ

Par exemple, sur la Fig. 1, la seconde ligne indique qu'une propriété de linéarité s'instancie sur un triplet de nœuds $\langle \pi, \pi i, \pi j\rangle$ composé d'un nœud père et de deux nœuds fils.

Pertinence. Nous avons ainsi une instance pour chaque propriété de P_G et chaque n-uplet de nœuds de τ. Nous devons en outre distinguer les instances des propriétés pertinentes de celles qui ne le sont pas, pour cela nous introduisons un prédicat P_τ (voir Fig. 2). Cette définition de la pertinence étend celle de [3] en y intégrant des comparaisons entre AVMs.

$$\begin{aligned}
&P_\tau((S_0 : S_1 \prec S_2)@\langle \pi, \pi i, \pi j\rangle) \equiv \\
&\qquad (L_\tau(\pi) \sqsubseteq S_0^v) \ \wedge\ (L_\tau(\pi i) \sqsubseteq S_1^v) \ \wedge\ (L_\tau(\pi j) \sqsubseteq S_2^v) \\
&P_\tau((S_0 : \triangle S_1)@\langle \pi\rangle) \equiv \\
&\qquad L_\tau(\pi) \sqsubseteq S_0^v \\
&P_\tau((S_0 : S_1!)@\langle \pi, \pi i, \pi j\rangle) \equiv \\
&\qquad (L_\tau(\pi) \sqsubseteq S_0^v) \ \wedge\ (L_\tau(\pi i) \sqsubseteq S_1^v) \ \wedge\ (L_\tau(\pi j) \sqsubseteq S_1^v) \\
&P_\tau((S_0 : S_1 \Rightarrow S_2)@\langle \pi, \pi i\rangle) \equiv \\
&\qquad (L_\tau(\pi) \sqsubseteq S_0^v) \ \wedge\ (L_\tau(\pi i) \sqsubseteq S_1^v) \\
&P_\tau((S_0 : S_1 \not\Rightarrow S_2)@\langle \pi, \pi i, \pi j\rangle) \equiv \\
&\qquad (L_\tau(\pi) \sqsubseteq S_0^v) \ \wedge\ ((L_\tau(\pi i) \sqsubseteq S_1^v) \ \vee\ (L_\tau(\pi j) \sqsubseteq S_2^v)) \\
&P_\tau((S_0 : s_1?)@\langle \pi, \pi i\rangle) \equiv \\
&\qquad L_\tau(\pi) \sqsubseteq S_0^v \\
&P_\tau((S_0 : S_1 \rightsquigarrow S_2)@\langle \pi, \pi i, \pi j\rangle) \equiv \\
&\qquad (L_\tau(\pi) \sqsubseteq S_0^v) \ \wedge\ (L_\tau(\pi i) \sqsubseteq S_1^v) \ \wedge\ (L_\tau(\pi j) \sqsubseteq S_2^v)
\end{aligned}$$

FIGURE 2 – Pertinence d'instances sur un arbre τ

Ainsi, sur la Fig. 2, nous remarquons que pour évaluer la pertinence d'une propriété, nous ne tenons pas compte des éventuelles co-références entre traits. Nous vérifions uniquement que les AVMs étiquetant les nœuds concernés ont des traits dont les valeurs sont compatibles avec la propriété (pour la relation $\sqsubseteq$).

Satisfaction. Lorsqu'une instance est pertinente, elle peut également être satisfaite. Pour définir cette notion de satisfaction, nous étendons le prédicat S_τ de [3] comme illustré en Fig. 3. Par exemple, la dernière ligne de la Fig. 3 définit la satisfaction pour la propriété d'accord (la seule manipulant des contraintes de co-référence entre traits). Cettte satisfaction dépend de la satisfaction d'équations de co-référence. Soient M_0, M_1, M_2 des AVMs, celles-ci satisfont les équations de co-référence de S_0, S_1, S_2 (noté $M_0, M_1, M_2 \models E(S_0, S_1, S_2)$), ssi $M_i|_f{=}M_j|_g$ pour tout $(i, f)\doteq(j, g)$ dans $E(S_0, S_1, S_2)$. Pour la propriété de linéarité (première ligne de la Fig. 3) par contre, la satisfaction n'est conditionnée que par l'ordre entre les nœuds fils du noeud π, sur lesquels s'instancie la propriété.

$$\begin{aligned}
&S_\tau((S_0 : S_1 \prec S_2)@\langle \pi, \pi i, \pi j\rangle) \equiv \\
&\qquad i < j \\
&S_\tau((S_0 : \triangle S_1)@\langle \pi\rangle) \equiv \\
&\qquad \vee\{(L_\tau(\pi i) \sqsubseteq S_1^v) \mid 1 \le i \le A_\tau(\pi)\} \\
&S_\tau((S_0 : S_1!)@\langle \pi, \pi i, \pi j\rangle) \equiv \\
&\qquad i = j \\
&S_\tau((S_0 : S_1 \Rightarrow S_2)@\langle \pi, \pi i\rangle) \equiv \\
&\qquad \vee\{(L_\tau(\pi j) \sqsubseteq S_2^v) \mid 1 \le j \le A_\tau(\pi)\} \\
&S_\tau((S_0 : S_1 \not\Rightarrow S_2)@\langle \pi, \pi i, \pi j\rangle) \equiv \\
&\qquad (L_\tau(\pi i) \not\sqsubseteq S_1^v) \ \vee\ (L_\tau(\pi j) \not\sqsubseteq S_2^v) \\
&S_\tau((S_0 : s_1?)@\langle \pi, \pi i\rangle) \equiv \\
&\qquad L_\tau(\pi i) \sqsubseteq x \quad \text{for some } x \text{ in } s_1 \\
&S_\tau((S_0 : S_1 \rightsquigarrow S_2)@\langle \pi, \pi i, \pi j\rangle) \equiv \\
&\qquad L_\tau(\pi), L_\tau(\pi i), L_\tau(\pi j) \models E(S_0, S_1, S_2)
\end{aligned}$$

FIGURE 3 – Satisfaction d'instances sur un arbre τ

Rappelons que l'intérêt de définir ces relations de pertinence et satisfaction d'instance, est, comme le précise [3], de pouvoir ensuite calculer un score reflétant le degré de grammaticalité d'un arbre syntaxique. Ainsi, un arbre syntaxique τ est un modèle de G s'il maximise le ratio $I^+_{G,\tau}/I^0_{G,\tau}$, où $I^0_{G,\tau}$ représente le nombre de contraintes pertinentes, et $I^+_{G,\tau}$ celui des contraintes pertinentes et satisfaites.

4 Analyse syntaxique des GP avec traits

À présent, nous montrons comment convertir cette sémantique de GP avec traits sous forme d'un analyseur par CSP, en étendant l'approche de Duchier *et al.* [2], qui utilise le concept d'arbres rectangulaires. Rappelons simplement ici, que ces arbres rectangulaires sont des arbres dont les nœuds sont placés de manière univoque dans une grille de taille fixée (suppression des symétries). Cette grille permet de plus de restreindre le nombre de nœuds composant un modèle bien que ce nombre soit *a priori* inconnu.

Pour un énoncé de m mots, nous savons que chaque modèle a m nœuds feuilles. Par contre, nous ne connaissons pas la hauteur de ces modèles, nous décidons de faire de cette hauteur n un paramètre du

CSP. Ainsi, chaque modèle est représenté par une matrice $\mathcal{W}$ telle que w_{ij} (avec $1 \leq i \leq n$, et $1 \leq j \leq m$) réfère au nœud situé à la position (i, j) sur la grille (la coordonnée (1,1) étant située dans le coin en bas à gauche). La Fig. 4 illustre ceci au moyen de l'arbre syntaxique de la phrase "Pierre mange la pomme".

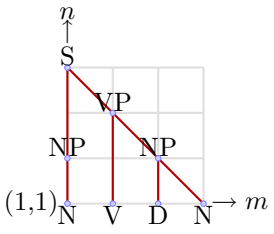


FIGURE 4 – Arbre syntaxique placé sur une grille

Notons que sur une telle grille, nous distinguons deux types de nœuds, ceux qui sont utilisés par un modèle (dits nœuds actifs), et ceux qui ne le sont pas (nœuds inactifs). Soit V l'ensemble des nœuds, V^+ est l'ensemble des nœuds actifs et V^- celui des nœuds inactifs. Un nœud est soit actif soit (strictement) inactif : $V = V^+ \uplus V^-$ ($\uplus$ étant l'union disjointe). La liste des contraintes permettant de produire de telles arbres rectangulaires est donnée dans [2].

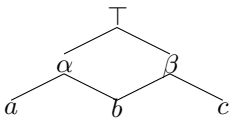
Représenter des contraintes sur semi-treillis. Comme cela a été présenté en Section 3, les contraintes sur structures de traits sont définies formellement sous forme de contraintes sur des AVMs dont les traits prennent leur valeur dans des semi-treillis. Voyons comment encoder ces contraintes sous forme de contraintes ensemblistes (notons que cet encodage relativement simple pourrait être étendu en utilisant des techniques telles que celles proposées par *Fall* [4]).

Dans les définitions de la *pertinence* et de la *pertinence et satisfaction*, nous devons représenter la relation $L_\tau(w) \sqsubseteq S^v$, où S est une expression-AVM et S^v est obtenue en remplaçant, dans S, chaque occurrence de $f{:}X$, où X est une variable, par $f{:}\top$. S^v est donc une AVM $[f_1{:}v_1, \ldots, f_n{:}v_n]$ sans variable, telle que chaque $v_i = S^v|_{f_i}$ ait sa valeur dans D_i.

Puisque $L_\tau(w)$ doit être instanciée par une AVM *minimale*, si $L_\tau(w) \sqsubseteq S^v$ alors chaque $L_\tau(w)|_{f_i}$ doit être associé avec une constante minimale $c_i \in D_i$ (*i.e.* un élément de D_i minimal pour l'ordre partiel $\sqsubseteq$), et telle que $c_i \sqsubseteq v_i$. Nous notons $\lfloor v_i \rfloor$ l'ensemble des valeurs minimales plus petites ($\sqsubseteq$) que v_i dans D_i. La relation $L_\tau(w) \sqsubseteq S^v$ peut alors être représentée par :

$$\bigwedge_{i=1}^{n} L_\tau(w)|_{f_i} \in \lfloor S^v|_{f_i} \rfloor \qquad (1)$$

Considérons par exemple une AVM $S^v = [f{:}\alpha]$, et le semi-treillis suivant.



Dans ce contexte, $L_\tau(w)$ satisfait la relation $L_\tau(w) \sqsubseteq S^v$ ssi $L_\tau(w)|_f$ appartient à $\lfloor \alpha \rfloor = \{a, b\}$.

La co-référence se représente comme suit. Soient S_0, S_1, S_2 des expressions-AVM. Rappelons que l'ensemble des équations de co-référence $E(S_0, S_1, S_2)$ est l'ensemble des équations $(i, f_u) \doteq (j, f_v)$ pour tout $f_u{:}X$ dans S_i et $f_v{:}X$ dans S_j. La relation $L_\tau(w_0), L_\tau(w_1), L_\tau(w_2) \models E(S_0, S_1, S_2)$ est représentée par :

$$\bigwedge_{(i,f_u)\doteq(j,f_v)\in E(S_0,S_1,S_2)} L_\tau(w_i)|_{f_u} = L_\tau(w_j)|_{f_v} \qquad (2)$$

Si nous retournons à notre problème de calcul de modèles d'arbres syntaxiques, nous devons assigner à chaque nœud actif w une étiquette $L_\tau(w)$, qui est une AVM minimale (pour $\sqsubseteq$). On pourrait modéliser $L_\tau(w)|_{f_i}$ au moyen *d'une variable de domaine fini* sur les valeurs minimales de D_i, mais alors que faire des nœuds inactifs ? Nous décidons plutôt de modéliser $L_\tau(w)|_{f_i}$ par une *variable ensembliste* de cardinalité au plus 1 :

$$L_\tau(w)|_{f_i} \subseteq \lfloor \top_i \rfloor \qquad |L_\tau(w)|_{f_i}| \leq 1$$

$$w \in V^+ \quad \Leftrightarrow \quad |L_\tau(w)|_{f_i}| = 1 \qquad \forall i \in [1, n]$$

et la relation de subsomption $L_\tau(w) \sqsubseteq S^v$ de (1) est alors représentée par :

$$\bigwedge_{i=1}^{n} L_\tau(w)|_{f_i} \subseteq \lfloor S^v|_{f_i} \rfloor \qquad (3)$$

Enfin, les nœuds feuilles de notre modèle doivent pouvoir être reliés aux mots du lexique L_G constitué de paires (mot, AVM) :

$$\bigwedge_{j=1}^{m} \exists M_j \ (\text{word}_j, M_j) \in L_G \wedge L_\tau(w_{1j}) \sqsubseteq M_j$$

où word_j réfère au j^e mot de la phrase à analyser.

À présent, nous pouvons exprimer les relations de *pertinence* et *pertinence et satisfaction* sous forme de contraintes sur des arbres rectangulaires comme suit (pour des raisons de place, nous ne présentons que la contrainte d'accord, les autres se traitant de manière similaire).

Accord. Comme mentionné en Fig. 1, la propriété $S_0 : S_1 \rightsquigarrow S_2$ s'instancie sur des triplets de nœuds, ses instances I sont donc de la forme :

$$(S_0 : S_1 \rightsquigarrow S_2)@\langle w_{i_0j_0}, w_{i_1j_1}, w_{i_2j_2}\rangle$$

$\forall w_{i_0j_0}, w_{i_1j_1}, w_{i_2j_2} \in V$, tels que $w_{i_1j_1} \neq w_{i_2j_2}$. Une telle instance est pertinente ssi $w_{i_0j_0}$ est actif et possède une étiquette $L_\tau(w_{i_0j_0}) \sqsubseteq S_0^v$, et $w_{i_1j_1}$ et $w_{i_2j_2}$ sont des nœuds fils (relation de parenté notée $\downarrow$ ci-dessous) dont les étiquettes respectent les contraintes $L_\tau(w_{i_1j_1}) \sqsubseteq S_1^v$ et $L_\tau(w_{i_2j_2}) \sqsubseteq S_2^v$:

$$P(I) \Leftrightarrow \left(\begin{array}{c} w_{i_0j_0} \in V^+ \wedge w_{i_1j_1} \in \downarrow w_{i_0j_0} \wedge \\ w_{i_2j_2} \in \downarrow w_{i_0j_0} \wedge L_\tau(w_{i_0j_0}) \sqsubseteq S_0^v \wedge \\ L_\tau(w_{i_1j_1}) \sqsubseteq S_1^v \wedge L_\tau(w_{i_2j_2}) \sqsubseteq S_2^v \end{array} \right)$$

Sa satisfaction dépend du fait que les co-références définies dans les étiquettes de $w_{i_0j_0}, w_{i_1j_1}, w_{i_2j_2}$ soient satisfaites (contrainte (2) introduite précédemment). Donc, nous avons :

$$S(I) \Leftrightarrow \left[\begin{array}{l} P(I) \wedge \\ L_\tau(w_{i_0j_0}), L_\tau(w_{i_1j_1}), L_\tau(w_{i_2j_2}) \models E(S_0, S_1, S_2) \end{array} \right]$$

5 Implantation

La modélisation de l'analyse syntaxique des grammaires de propriété sous forme de CSP présentée ici repose sur la modélisation de Duchier *et al.* [2]. Dans leurs travaux, les auteurs proposaient une implantation en C++ au moyen de la bibliothèque Gecode [5].

L'extension de cette implantation aux propriétés à base de structures de traits nécessitait une refonte importante, en partie pour des raisons techniques liées au langage C++. Nous avons donc choisi de re-développer l'analyseur syntaxique en langage Python, toujours en utilisant Gecode pour la résolution de contraintes. [1]

Comme dans les travaux antérieurs de Duchier *et al.*, l'objectif principal de cette implantation n'est pas de fournir un analyseur performant dans le sens où il pourrait se comparer aux analyseurs existant pour d'autres formalismes grammaticaux, mais un analyseur dépourvu de toute heuristique, permettant ainsi au linguiste d'observer directement les conséquences de choix de représentations de la syntaxe de la langue. En quelque sorte, l'analyseur permet au linguiste d'avoir une estimation de la complexité d'un modèle de la langue.

Cette implantation étant en cours de développement, elle n'a pas pu être confrontée à des grammaires de taille réelle, nous ne disposons donc pas, à l'heure actuelle, de *benchmarks*. L'analyseur intègre actuellement un support expérimental pour les propriétés à base de traits, et une évaluation de son "efficacité" est envisagée, en utilisant la grammaire de [6].

1. Via un *binding* Gecode/Python développé en interne, et disponible librement à l'adresse `https://launchpad.net/pygecode`.

6 Conclusion

Dans cet article, nous avons présenté (i) une sémantique en théorie des modèles des Grammaires de Propriété à Traits, et (ii) la définition de l'analyse syntaxique en GP par conversion de cette sémantique en un problème de satisfaction de contraintes. L'intérêt de ce travail est de fournir une définition formelle des structures de traits dans le formalisme GP, et une implantation de ces structures de traits dans un analyseur de GP en programmation par contraintes (cette implantation est en cours de réalisation au moyen de la librairie Gecode).

Pour optimiser l'analyse syntaxique en pratique, nous envisageons plusieurs pistes de travail, notamment (i) la définition de l'analyse syntaxique par composition d'analyses partielles (chacune implantée sous forme d'un CSP), (ii) la parallélisation de l'exploration de l'arbre de recherche.

Enfin, une autre piste de recherche intéressante concerne la définition de poids à associer aux différentes contraintes en fonction des catégories impliquées ou du type de propriété (ordre entre catégories *vs* catégorie manquante) afin d'obtenir un modèle et un score de grammaticalité plus proche de l'intuition humaine lorsque des énoncés agrammaticaux sont manipulés.

Références

[1] Philippe Blache. Constraints, Linguistic Theories and Natural Language Processing. In D. Christodoulakis, editor, *Natural Language Processing*, Lecture Notes in Artificial Intelligence Vol. 1835. Springer-Verlag, 2000.

[2] Denys Duchier, Thi-Bich-Hanh Dao, Yannick Parmentier, and Willy Lesaint. Une modélisation en CSP des grammaires de propriétés. In *Sixièmes Journées Francophones de Programmation par Contraintes (JFPC)*, pages 123–132, Caen, France, 2010.

[3] Denys Duchier, Jean-Philippe Prost, and Thi-Bich-Hanh Dao. A model-theoretic framework for grammaticality judgements. In *Conference on Formal Grammar (FG2009)*, Bordeaux, France, 2009.

[4] Andrew Fall. *Reasoning with taxonomies*. PhD thesis, School of Computing Science, Simon Fraser University, December 1996.

[5] Gecode Team. Gecode : Generic constraint development environment, 2012. Available from `http ://www.gecode.org`.

[6] Jean-Philippe Prost. *Modelling Syntactic Gradience with Loose Constraint-based Parsing*. Cotutelle Ph.D. Thesis, Macquarie University, Sydney, Australia, and Université de Provence, Aix-en-Provence, France, December 2008.