Une modélisation en CSP des grammaires de propriétés

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

Une modélisation en CSP des grammaires de propriétés

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

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 à la fois des énoncés bien-formés et des énoncés agrammaticaux, ce qui en fait un formalisme particulièrement intéressant pour traiter de la gradience de grammaticalité, comme l'a démontré Prost [12]. Duchier *et al* [7] ont défini une sémantique des grammaires de propriétés en théorie des modèles. Cet article poursuit ce travail en montrant comment, à partir de cette sémantique, définir l'analyse syntaxique en GP sous la forme d'un Problème de Satisfaction de Contraintes (CSP), traitable au moyen de la programmation par contraintes.

Abstract

Property grammars offer a constraint-based formalism capable of handling both well-formed and deviant utterances. It is thus well-suited for addressing issues of gradience of grammaticality, as demonstrated by Prost [12]. Duchier *et al* [7] contributed precise model-theoretic semantics for property grammars. The present article follows up on that work and explains how to turn such a formalization into a concrete constraint satisfaction problem (CSP), solvable using constraint programming.

1 Introduction

Dans ce travail, nous nous intéressons à une tâche spécifique du domaine du Traitement Automatique des Langues, à savoir 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 un système de réécriture (*e.g.* réécriture de chaînes pour les grammaires hors-contexte [4], ou encore réécriture d'arbres pour les grammaires d'arbres adjoints [10]). Un problème majeur de ces formalismes, dans un contexte de Traitement Automatique de la Langue, 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.

Comme l'ont démontré Pullum et Scholz [13], les grammaires formelles de type *syntaxe générative-énumérative*, dans la mesure où elles se concentrent sur la génération de modèles bien formés, sont intrinsèquement inadaptées au traitement d'énoncés agrammaticaux. Par contre, des grammaires formelles de type *syntaxe fondée sur la théorie des modèles*, qui se concentrent sur une validation de modèles en termes de contraintes satisfaites, sont naturellement adaptées au traitement de *quasi-expressions*.

Blache [2, 3] a proposé le formalisme des Grammaires de Propriétés (GP), comme un formalisme à base de contraintes, permettant d'analyser à la fois des énoncés grammaticaux et agrammaticaux. Prost [12] a développé une technique d'analyse syntaxique utilisant GP, et permettant de produire une structure syntaxique pour tout type d'énoncé, tout en y associant un jugement précis sur la grammaticalité de l'énoncé en question. Duchier *et al* [7] ont fourni une sémantique en théorie des modèles pour GP, ainsi qu'une définition logique formelle des travaux d'analyse syntaxique menés par Prost. Dans cet article, nous montrons comment une telle formalisation de la sémantique de GP, peut être convertie en un CSP, ouvrant la voie à l'implantation d'un analyseur syntaxique à

base de contraintes, qui calcule des arbres syntaxiques optimaux (en termes de grammaticalité d'énoncé), au moyen d'une recherche de type *branch-and-bound*.

L'article est structuré comme suit. En section 2, nous introduisons le formalisme des grammaires de propriétés. En section 3, nous définissons une sémantique des grammaires de propriétés basée sur la théorie des modèles. Cette sémantique va servir de cadre à la définition de la tâche d'analyse syntaxique en termes de problème de satisfaction de contraintes. Cette définition va reposer sur deux types de contraintes permettant de construire l'arbre syntaxique d'un énoncé. Le premier type de contraintes, les contraintes de structure d'arbre, sera présenté en section 4, le second type de contraintes, les contraintes de propriétés, en section 5. Cette définition à base de contraintes de l'analyse syntaxique pour les grammaires de propriétés mènera naturellement à une implantation en programmation par contraintes, qui sera introduite en section 6. Enfin, en section 7, nous comparons notre travail avec les approches existantes, et en section 8, nous concluons en présentant les travaux futurs.

2 Grammaires de propriétés

Les grammaires de propriétés [2, 3] constituent un formalisme permettant de décrire la langue naturelle en termes de contraintes locales, appelées *propriétés*. Ces propriétés peuvent être violées indépendemment les unes des autres, ce qui rend possible la description d'énoncés agrammaticaux (*i.e.*, qui ne vérifieraient pas l'ensemble des propriétés de la langue), et également de définir une notion de grammaticalité (ratio entre propriétés violées et propriétés vérifiées).

Les propriétés utilisées pour décrire la langue se basent sur des observations linguistiques : ordre (partiel) entre les mots, exclusion mutuelle entre certains mots dans un contexte proche, cooccurrence systématique de certains mots dans un contexte proche, non-répétition de certains mots dans un contexte proche, présence facultative de certains mots, *etc.*

Chaque propriété a la forme $A : \psi$, où A représente la catégorie (*i.e.*, l'étiquette) d'un nœud dans un arbre syntaxique, et ψ la contrainte qui s'applique sur les nœuds fils de celui-ci. L'ensemble des propriétés auxquelles nous nous intéressons est le suivant :

obligation $A : \triangle B$, pour tout nœud d'étiquette A, présence d'au moins un nœud fils étiqueté B,

unicité $A : B!$, pour tout nœud d'étiquette A, présence d'au plus un nœud fils étiqueté B,

linéarité $A : B \prec C$, pour tout nœud d'étiquette A, en cas de présence de nœuds fils étiquetés B et C, B précède C,

exigence [1] $A : B \Rightarrow C$, pour tout nœud d'étiquette A, la présence d'un nœud fils étiqueté B implique la présence d'un nœud fils étiqueté C,

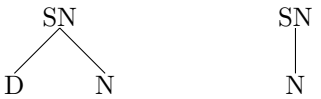
exclusion $A : B \not\Rightarrow C$, pour tout nœud d'étiquette A, des nœuds fils d'étiquettes B et C sont mutuellement exclusifs,

constituence $A : S$, pour tout nœud d'étiquette A, les étiquettes des nœuds fils appartiennent à l'ensemble S.

A partir de cet ensemble de propriétés, il est possible d'énoncer un certain nombre de règles grammaticales décrivant la syntaxe de la langue naturelle. Par exemple, une règle grammaticale stipulant qu'un groupe nominal contient exactement un nom et que ce nom peut être précédé d'un déterminant, pourrait être représentée par l'ensemble de propriétés ci-dessous :

$$_{(1)}\texttt{SN} : \{\texttt{D}, \texttt{N}\},\ _{(2)}\texttt{SN} : \triangle\texttt{N},\ _{(3)}\texttt{SN} : \texttt{N}!,\ _{(4)}\texttt{SN} : \texttt{D}!,$$
$$_{(5)}\texttt{SN} : \texttt{D} \prec \texttt{N},\ _{(6)}\texttt{D} : \{\},\ _{(7)}\texttt{N} : \{\}$$

où SN réfère à l'étiquette *Syntagme Nominal* (groupe nominal), D à *Déterminant*, et N à *Nom*. La propriété (1) ($\texttt{SN} : \{\texttt{D}, \texttt{N}\}$) indique que, dans un arbre syntaxique, les nœuds fils du nœud représentant le groupe nominal doivent être étiquetés D ou N. (2) ($\texttt{SN} : \triangle\texttt{N}$) indique qu'un nœud SN domine directement au moins un nœud N. (3) ($\texttt{SN} : \texttt{N}!$) indique qu'un nœud SN domine directement au plus un nœud N. (4) ($\texttt{SN} : \texttt{D}!$) indique qu'un nœud SN domine directement au plus un nœud D. (5) ($\texttt{SN} : \texttt{D} \prec \texttt{N}$) indique que, pour tout nœud d'étiquette SN, si ce nœud comporte deux nœuds fils d'étiquettes respectives D et N, alors le nœud d'étiquette D précède celui d'étiquette N. Enfin, (6) et (7) permettent de s'assurer que les nœuds d'étiquettes D et N sont des nœuds feuilles (seuls les mots du lexique peuvent être situés en dessous de ces nœuds). L'ensemble de ces propriétés a exactement deux modèles d'arbre syntaxique ayant la racine SN :



Les étiquettes introduites dans cet exemple sont appelées *catégories syntaxiques* et renseignent sur le type d'un constituant de la phrase. Une grammaire de propriétés définit donc des contraintes sur les relations entre les différents types de constituants apparaissant dans une phrase. Pour faire le lien avec le lexique, on définit une association entre mots et catégories syntaxiques. Par exemple, on peut spécifier que le mot *pomme* est un nom (catégorie N) via :

$$\mathsf{cat}(\mathit{pomme}) = \texttt{N}$$

[1] Également appelée *cooccurrence* dans la littérature.

Notons qu'il est possible qu'un même mot soit associé à plusieurs catégories (on parle alors d'ambiguïté lexicale). C'est le cas par exemple du mot *ferme* qui peut correspondre à un nom, un adjectif ou encore un verbe (forme conjuguée du verbe *fermer*).

3 Sémantique des grammaires de propriétés en théorie des modèles

En section précédente, nous avons présenté le formalisme des grammaires de propriétés. Ici, nous allons voir comment définir une sémantique à ce formalisme dans le cadre de la théorie des modèles. Cette sémantique a été introduite par Duchier *et al* [7]. Elle nous permettra de traduire la tâche de l'analyse syntaxique en GP sous la forme d'un problème de satisfaction de contraintes.

Sémantique forte Nous interprétons les grammaires de propriétés sous forme d'arbres syntaxiques. Un arbre syntaxique τ est un modèle *fort* d'une grammaire $\mathcal{G}$, si et seulement si, pour chaque nœud n de τ, et pour chaque propriété p de $\mathcal{G}$, si la propriété p en question est *pertinente* au nœud n, alors elle y est également *satisfaite*.

Cette interprétation sous forme d'arbre syntaxique nous conduit à considérer des *instances* de propriété. Une instance s'applique sur un nœud d'un arbre syntaxique candidat.

Pour illustrer cela, considérons la propriété de linéarité $\texttt{SN} : \texttt{D} \prec \texttt{N}$ vue à la section précédente. Lors de la recherche d'un modèle d'arbre syntaxique, cette propriété est *instanciée* à chaque nœud de l'arbre syntaxique candidat. Plus précisément, pour chaque nœud n de cet arbre, nous allons considérer toutes les paires possibles de nœuds fils (n_1, n_2). Nous allons, pour cette configuration (n, n_1, n_2), évaluer le fait que la propriété soit pertinente (*i.e.*, qu'elle doive s'appliquer) et également qu'elle soit satisfaite. Nous notons cette instance de la propriété de linéarité sous la forme suivante :

$$\texttt{SN} : \texttt{D} \prec \texttt{N} \ @ \ \langle n, n_1, n_2 \rangle$$

Pour que cette instance de propriété soit pertinente, il faut que n soit étiqueté SN, et n_1 (respectivement n_2) soit étiqueté D (resp. N). Pour être satisfaite, il faut en plus que le nœud n_1 précède n_2.

Considérons un autre type de propriété, à savoir la constituence, illustrée par $\texttt{SN} : \{\texttt{D}, \texttt{N}\}$. Cette propriété est, elle aussi, instanciée à chaque nœud de l'arbre syntaxique candidat. Ici, pour pouvoir vérifier si cette propriété est pertinente à un nœud donné n, il nous suffit de considérer un seul de ses nœuds fils, appelons le n_1. Ainsi, nous notons une instance de la propriété de constituence sous la forme suivante :

$$\texttt{SN} : \{\texttt{D}, \texttt{N}\} \ @ \ \langle n, n_1 \rangle$$

Une telle instance est pertinente lorsque n est étiqueté SN, et satisfaite lorsque n_1 est de plus étiqueté D ou N.

Nous avons donc différents types de propriétés selon le nombre de nœuds impliqués dans son instanciation. Nous avons vu que les propriétés de linéarité s'instancient sur un triplet de nœuds[2], celles de constituence sur un couple de nœuds. Un dernier type de propriété correspond à celles dont l'instanciation s'applique sur un nœud, ce qui est le cas de l'obligation. Pour ce type de propriété, une instance de propriété a la forme suivante :

$$\texttt{SN} : \triangle\texttt{N} \ @ \ \langle n \rangle$$

Pour tout nœud n de l'arbre syntaxique candidat, si ce nœud est étiqueté SN alors l'instance de propriété est pertinente, et s'il a en outre au moins un nœud fils étiqueté N, alors elle est satisfaite.

Dans un modèle fort, toutes les instances de propriété qui sont pertinentes doivent obligatoirement être satisfaites.

Sémantique relâchée Un arbre syntaxique τ est un modèle *relâché* d'une grammaire de propriétés $\mathcal{G}$, si son *score d'adéquation* est maximal. Par score d'adéquation, nous entendons le ratio entre instances de propriétés satisfaites parmi l'ensemble des instances de propriétés pertinentes.

Dans un modèle relâché, il est possible que certaines instances de propriétés, bien que pertinentes, ne soient pas satisfaites. Le fait de pouvoir violer certaines propriétés permet de calculer des modèles d'arbres syntaxiques correspondant à des énoncés agrammaticaux. Le score d'adéquation nous permet ainsi de chiffrer l'adéquation syntaxique du modèle.

Exemple Pour illustrer ces deux sémantiques de GP, considérons la grammaire jouet $\mathcal{G}$ suivante :

$$_{(1)}\texttt{P} : \{\texttt{SN}, \texttt{VP}\}, \ _{(2)}\texttt{SN} : \{\texttt{D}, \texttt{N}\}, \ _{(3)}\texttt{VP} : \{\texttt{V}, \texttt{SN}\},$$

$$_{(4)}\texttt{P} : \triangle\texttt{VP}, \ _{(5)}\texttt{P} : \texttt{VP}!, \ _{(6)}\texttt{P} : \triangle\texttt{SN}, \ _{(7)}\texttt{P} : \texttt{SN}!,$$

$$_{(8)}\texttt{P} : \texttt{SN} \prec \texttt{VP}, \ _{(9)}\texttt{VP} : \triangle\texttt{V}, \ _{(10)}\texttt{VP} : \texttt{V}!, \ _{(11)}\texttt{VP} : \texttt{SN}!,$$

$$_{(12)}\texttt{SN} : \texttt{D}!, \ _{(13)}\texttt{SN} : \triangle\texttt{N}, \ _{(14)}\texttt{SN} : \texttt{D} \Rightarrow \texttt{N}, \ _{(15)}\texttt{SN} : \texttt{D} \prec \texttt{N},$$

$$_{(16)}\texttt{cat}(\textit{mange}) = \texttt{V}, \ _{(17)}\texttt{cat}(\textit{la}) = \texttt{D},$$

$$_{(18)}\texttt{cat}(\textit{Pierre}) = \texttt{N}, \ _{(19)}\texttt{cat}(\textit{pomme}) = \texttt{N}$$

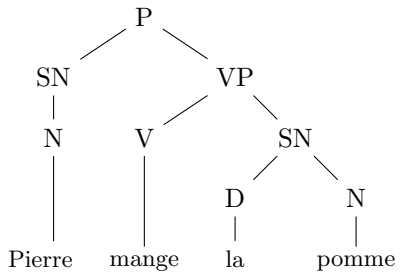
$\mathcal{G}$ permet de construire un arbre syntaxique pour la phrase "Pierre mange la pomme"[3]. Les propriétés (16)

[2] Ce qui est également le cas de l'unicité, de l'exigence, et de l'exclusion.

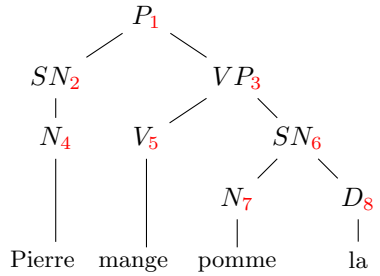
[3] Il est possible qu'une grammaire décrive plusieurs arbres syntaxiques pour un même énoncé, on parle alors d'ambiguïté syntaxique.

à (19) forcent la catégorie des nœuds feuilles de notre arbre syntaxique, elles ne peuvent être violées.

À partir de cette grammaire et de la phrase "Pierre mange la pomme", nous pouvons construire l'arbre syntaxique (*i.e.*, l'unique modèle ici en sémantique forte, qui est également le modèle de score d'adéquation maximal[4] en sémantique relâchée) suivant :



Si nous considérons dans un deuxième temps l'énoncé agrammatical "Pierre mange pomme la", il n'existe plus d'arbre syntaxique en sémantique forte. Cependant, modulo la violation de la propriété de linéarité fixant l'ordre entre les nœuds d'étiquettes `D` et `N`, il est possible d'obtenir un modèle d'arbre syntaxique de score d'adéquation maximal 14/15, représenté ci-dessous[5] :



Ce score d'adéquation correspond au ratio entre instances de propriétés pertinentes et instances pertinentes et satisfaites. Par exemple, la propriété de consituence (1)$\mathtt{P} : \{\mathtt{SN}, \mathtt{VP}\}$ est instanciée, pertinente et satisfaite à deux reprises (pour chacun des fils du nœud 1) :

$$(1)\mathtt{P} : \{\mathtt{SN}, \mathtt{VP}\} \ @ \ \langle 1, 2\rangle \qquad (1)\mathtt{P} : \{\mathtt{SN}, \mathtt{VP}\} \ @ \ \langle 1, 3\rangle$$

Les autres propriétés instanciées et pertinentes sont les suivantes (la seule propriété non satisfaite est précédée du symbole $\times$)[6] :

$$(2)\mathtt{SN} : \{\mathtt{D}, \mathtt{N}\} \ @ \ \langle 2, 4\rangle$$
$$(2)\mathtt{SN} : \{\mathtt{D}, \mathtt{N}\} \ @ \ \langle 6, 7\rangle \qquad (2)\mathtt{SN} : \{\mathtt{D}, \mathtt{N}\} \ @ \ \langle 6, 8\rangle$$
$$(3)\mathtt{VP} : \{\mathtt{V}, \mathtt{SN}\} \ @ \ \langle 3, 5\rangle \qquad (3)\mathtt{VP} : \{\mathtt{V}, \mathtt{SN}\} \ @ \ \langle 3, 6\rangle$$
$$(4)\mathtt{P} : \triangle\mathtt{VP} \ @ \ \langle 1\rangle \qquad (6)\mathtt{P} : \triangle\mathtt{SN} \ @ \ \langle 1\rangle$$
$$(8)\mathtt{P} : \mathtt{SN} \prec \mathtt{VP} \ @ \ \langle 1, 2, 3\rangle \qquad (9)\mathtt{VP} : \triangle\mathtt{V} \ @ \ \langle 3\rangle$$
$$(13)\mathtt{SN} : \triangle\mathtt{N} \ @ \ \langle 2\rangle \qquad (13)\mathtt{SN} : \triangle\mathtt{N} \ @ \ \langle 6\rangle$$
$$(14)\mathtt{SN} : \mathtt{D} \Rightarrow \mathtt{N} \ @ \ \langle 6, 8, 7\rangle \qquad \times \ (15)\mathtt{SN} : \mathtt{D} \prec \mathtt{N} \ @ \ \langle 6, 8, 7\rangle$$

Ce qui nous donne 14 instances de propriété pertinentes et satisfaites sur 15 instances pertinentes.

4 Contraintes de structure d'arbre

Étant données une grammaire de propriétés et une expression (qui peut éventuellement être une quasi-expresssion), l'objectif de notre approche est de trouver tous les quasi-modèles qui peuvent être candidats pour une analyse syntaxique et de retenir ceux qui maximisent un score défini. Un arbre d'analyse syntaxique est un arbre dont les feuilles correspondent aux mots de l'expression, et dont les nœuds sont étiquetés par des catégories de la grammaire. Pour définir notre CSP, nous devons spécifier les variables utilisées pour modéliser un arbre syntaxique, or nous ne connaissons pas *a priori* le nombre de nœuds que contient un tel arbre. Nous choisissons donc de borner ce nombre en considérant une grille, sur laquelle seront placés les nœuds d'un arbre syntaxique.

Soit m le nombre de mots de l'expression à analyser. Chaque arbre syntaxique a m feuilles (les grammaires de propriétés n'utilisent pas de nœuds "vides" ϵ). Nous notons n la profondeur maximale d'un arbre syntaxique (n est un paramètre du problème). Nous allons chercher les arbres syntaxiques qui peuvent être placés sur un sous-ensemble de nœuds d'une grille de taille $n \times m$. Dans cette section, nous présentons la structure d'arbre rectangulaire, les contraintes permettant de définir un arbre rectangulaire sur une grille $n \times m$ et une réalisation de ces contraintes.

4.1 Arbres rectangulaires

Nous référons à un nœud d'un arbre syntaxique par sa position sur une grille de taille $n \times m$, dont les lignes sont numérotées de 1 à n et les colonnes de 1 à m, la ligne du bas portant le numéro 1.

Un arbre est caractérisé par un ensemble de nœuds, une racine et pour chaque nœud, un nœud père. De façon duale, à chaque nœud nous pouvons associer un ensemble des nœuds fils. Chaque nœud de l'arbre se plaçant sur un point de la grille, l'arbre s'étale sur un sous-ensemble des points de la grille.

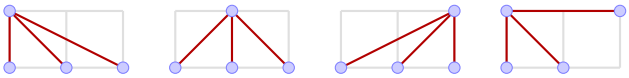
Cependant, il existe plusieurs façons d'étaler un même arbre sur une grille, si l'on ne tient pas compte

[4] Sur cet arbre, toutes les propriétés pertinentes sont satisfaites, et donc ce score d'adéquation vaut 1.

[5] Les numéros annotant les nœuds servent uniquement à référer au nœud dans l'explication du score.

[6] La sémantique donnée à l'unicité par Duchier *et al* [7] est qu'elle n'est pertinente que lorsqu'elle est violée, elle n'apparaît donc pas ici.

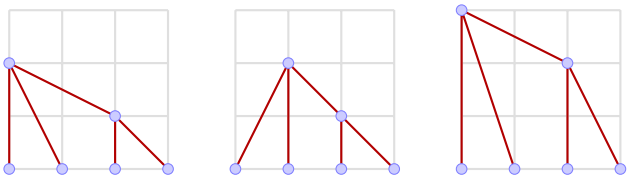
des symétries. Par exemple, sur une grille de taille 2×3, un arbre de 4 nœuds peut entre autres avoir les représentations équivalentes suivantes :



Afin d'arriver à une représentation unique d'un arbre sur une grille, nous restreignons les arbres à être *rectangulaires*. Ces arbres satisfont les conditions suivantes :

1. les feuilles se situent toutes au niveau le plus bas de la grille (la ligne numéro 1) ;
2. chaque nœud de l'arbre doit se situer dans la même colonne que la feuille la plus à gauche de son sous-arbre (ceci implique que le sous-arbre à partir d'un nœud s'étale sur le rectangle en bas à droite du nœud) ;
3. chaque nœud doit se situer à un niveau plus haut que le niveau des nœuds de son sous-arbre (ceci implique que la racine se situe au niveau le plus haut de l'arbre) ;
4. chaque nœud qui n'est pas une feuille doit avoir au moins un fils au niveau inférieur immédiat (ceci implique qu'il n'y a pas de niveau intermédiaire non occupé par l'arbre).

Par exemple, parmi les arbres suivants, seul le premier est un arbre rectangulaire, les deux autres ne le sont pas car la condition 2 n'est pas vérifiée pour le second et la condition 4 n'est pas vérifiée pour le dernier :



Ce problème de reconnaissance d'arbres sur une grille est à distinguer de celui de la reconnaissance d'arbres sur un graphe, pour lequel il existe plusieurs travaux (*e.g.* Beldiceanu et al. [1], Prosser et Unsworth [11]). En effet, ces derniers visent à calculer un arbre ou une forêt d'arbres couvrants dans un graphe, en cherchant le nœud racine. Dans notre cas, nous ne connaissons pas *a priori* les nœuds de notre arbre, nous savons uniquement qu'ils peuvent occuper certaines positions sur une grille de taille fixée (grille qui ne correspond pas à des arêtes d'un graphe).

4.2 Contraintes de structure d'arbre

Arbre Nous écrivons w_{ij}, avec $1 \leq i \leq n$ et $1 \leq j \leq m$, pour les nœuds de la grille. Soit V l'ensemble des nœuds de la grille. L'arbre s'étale sur un sous-ensemble de V. Un nœud est appelé *actif* s'il est utilisé par l'arbre et *inactif* sinon. Soient V^+ l'ensemble des nœuds actifs et V^- l'ensemble des autres nœuds. Une contrainte relie donc ces deux ensembles avec V :

$$V = V^+ \uplus V^-$$

où $\uplus$ représente "l'union disjointe". Utilisant la technique de modélisation dans [6], pour chaque nœud w, nous écrivons $\downarrow w$ pour l'ensemble des nœuds fils de w, $\downarrow^+ w$ pour ses descendants et $\downarrow^* w$ pour w et ses descendants. D'une façon duale, nous écrivons $\uparrow w$ pour l'ensemble des parents de w, $\uparrow^+ w$ pour ses ancêtres et $\uparrow^* w$ pour w et ses ancêtres. Les contraintes reliants ces ensembles sont :

$$\downarrow^+ w = \uplus\{\downarrow^* w' \mid w' \in \downarrow w\} \qquad \downarrow^* w = \{w\} \uplus \downarrow^+ w$$
$$\uparrow^+ w = \uplus\{\uparrow^* w' \mid w' \in \uparrow w\} \qquad \uparrow^* w = \{w\} \uplus \uparrow^+ w$$

L'utilisation de l'union disjointe ici est justifiée par le fait que dans un arbre, chaque nœud ne peut appartenir à deux sous-arbres de même niveau. La contrainte suivante renforce la dualité :

$$w \in \uparrow w' \;\Leftrightarrow\; w' \in \downarrow w$$

Le fait que dans un arbre chaque nœud a au plus un parent est représenté par la contrainte suivante :

$$|\uparrow w'| \leq 1$$

Un nœud inactif n'appartient pas à l'arbre, il n'a donc ni parents ni enfants :

$$w \in V^- \Rightarrow \downarrow w = \uparrow w = \emptyset$$

Nous écrivons R pour l'ensemble des nœuds qui sont racines de l'arbre. Un arbre doit avoir une seule racine :

$$|R| = 1$$

et une racine ne peut être enfant d'un autre nœud :

$$V^+ = R \uplus (\uplus\{\downarrow w \mid w \in V\})$$

Arbre rectangulaire Pour chaque nœud w, nous écrivons $\Downarrow w$ pour l'ensemble des colonnes occupées par le sous-arbre ancré au nœud w. Nous écrivons $c(w)$ pour la colonne du nœud w et $\ell(w)$ pour sa ligne :

$$c(w_{ij}) = j \qquad \ell(w_{ij}) = i$$

La première condition d'un arbre rectangulaire concerne les feuilles. Les feuilles de l'arbre correspondent aux mots de l'expression. Elles doivent se

situer sur la ligne numéro 1 de la grille. Comme le nombre de colonnes de la grille est égal au nombre de mots, chaque nœud de cette ligne doit être actif :

$$\{w_{1j} \mid 1 \leq j \leq m\} \subseteq V^+$$

La seconde condition indique que chaque nœud actif doit se situer dans la même colonne que la feuille la plus à gauche de son sous-arbre. Elle est imposée par la contrainte suivante :

$$w_{ij} \in V^+ \quad \Leftrightarrow \quad c(w_{ij}) = \min \Downarrow w_{ij}$$

Par la troisième condition, chaque nœud de l'arbre doit se situer à un niveau plus haut que le niveau des nœuds de son sous-arbre. Cette condition jointe à une conséquence de la seconde condition est représentée par les contraintes de domaine suivantes. Pour chaque nœud w_{ij} :

- les nœuds descendants sont dans le rectangle en bas à droit du nœud, la colonne du nœud incluse

$$\downarrow^+ w_{ij} \subseteq \{w_{lk} \mid 1 \leq l < i,\ j \leq k \leq m\}$$

- les nœuds ascendants sont dans le rectangle en haut à gauche du nœud, la colonne du nœud incluse

$$\uparrow^+ w_{ij} \subseteq \{w_{lk} \mid i < l \leq n,\ 1 \leq k \leq j\}$$

La dernière condition indique que chaque nœud actif qui n'est pas une feuille doit avoir au moins un fils à la ligne inférieure immédiate. Elle est imposée par la contrainte suivante pour chaque nœud w_{ij}, avec $1 < i \leq n$:

$$w_{ij} \in V^+ \quad \Leftrightarrow \quad i - 1 \in \{\ell(w) \mid w \in \downarrow w_{ij}\}$$

Projection Les feuilles de l'arbre doivent se situer toutes à la première ligne de la grille, chacune d'elles occupe donc une seule colonne :

$$\Downarrow w_{1j} = \{j\}$$

Pour les nœuds de niveau supérieur, la projection correspond à une union disjointe des projections des fils :

$$\Downarrow w_{ij} = \uplus\{\Downarrow w \mid w \in \downarrow w_{ij}\} \qquad 1 < j \leq m$$

La projection de chaque nœud ne doit pas contenir de trou (les arbres sont projectifs)[7] :

$$\mathsf{convex}(\Downarrow w) \qquad \forall w \in V$$

[7] Cette contrainte de convexité d'un ensemble d'entiers est par exemple implantée dans la librairie Gecode.

Catégories Dans un arbre syntaxique, chaque nœud actif doit être étiqueté par une catégorie syntaxique. L'ensemble des catégories possibles pour chaque nœud sera déterminé par les contraintes de propriétés présentées dans la section 5. En ce qui concerne les feuilles, chacune d'elles sera étiquetée par la catégorie du mot correspondant :

$$\mathsf{cat}(w_{1j}) = \mathsf{cat}(\mathrm{mots}_j)$$

Les nœuds non actifs sont les seuls étiquetés par la catégorie `none` :

$$\mathsf{cat}(w) = \texttt{none} \quad \Leftrightarrow \quad w \in V^-$$

4.3 Réalisation des contraintes

Pour déterminer un arbre, les éléments à calculer sont V^+, V^-, R, les fonctions $\downarrow, \downarrow^+, \downarrow^*$ et $\uparrow, \uparrow^+, \uparrow^*$, la fonction $\Downarrow$ et la fonction cat.

Chaque nœud (i, j) de la grille est identifié par le numéro $(i-1)m + j$. L'ensemble V des nœuds de la grille est donc l'ensemble des entiers de 1 à nm. Pour désigner les ensembles V^+, V^- et R, nous utilisons des variables ensemblistes dont les éléments sont dans V. Les contraintes reliant ces ensembles sont facilement traduites par des contraintes ensemblistes.

Concernant les fonctions $\downarrow, \downarrow^+, \downarrow^*$ et $\uparrow, \uparrow^+, \uparrow^*$, chacune d'elles est représentée par un tableau indexé sur les éléments de V, où le i-ème élément du tableau est une variable ensembliste dont les éléments sont dans V. Par exemple le fait que le 9ème élément du tableau $\downarrow$ est l'ensemble $\{1, 4, 6\}$ indique que les enfants du nœud numéro 9 sont les nœuds numéro 1, 4 et 6. Quant à la fonction $\Downarrow$, elle est représentée par un tableau également indexé sur les éléments de V. Chaque i-ème élément du tableau est une variable ensembliste dont les éléments sont dans l'intervalle $[1, m]$. Cette variable désigne l'ensemble des colonnes possibles pour le nœud numéro i.

Les contraintes concernant ces ensembles sont posées à l'aide des contraintes `element`, `convex`, `min`, d'union disjointe, de domaine et de cardinalité. Les contraintes faisant intervenir les connecteurs logiques implication et équivalence sont réalisées à l'aide des contraintes réifiées et des variables booléennes. Nous présentons en particulier la contrainte :

$$\downarrow^+ w = \uplus\{\downarrow^* w' \mid w' \in \downarrow w\}$$

Soit i le numéro du nœud w. Cette contrainte est réalisée par une contrainte de sélection introduite dans [6] $A = \uplus\langle B_1, \ldots, B_n\rangle[S]$ qui signifie $A = \uplus_{i \in S} B_i$. Ici la suite $\langle B_1, \ldots, B_n\rangle$ est le tableau réalisant la fonction $\downarrow^*$ et l'ensemble S est le i-ème élément du tableau réalisant la fonction $\downarrow$.

La dernière fonction cat est également représentée par un tableau indexé sur les éléments de V. Chaque i-ème élément du tableau est une variable entière, indiquant la catégorie étiquetant le nœud numéro i.

5 Contraintes pour les propriétés

Après avoir construit la structure d'arbre, il faut déterminer le degré de satisfaction des propriétés de l'arbre. Nous présentons dans cette section les contraintes réalisant les propriétés des grammaires de propriétés et une contrainte qui optimise le degré de satisfaction des propriétés.

5.1 Instance de propriété

Dans un premier temps, revenons sur la notion d'instance de propriété introduite en section 3.

Les propriétés correspondent à des contraintes, qui doivent être satisfaites par un modèle. Une contrainte s'applique sur certains paramètres (des variables du modèle). Dans notre cas, nous appliquons une contrainte à des nœuds d'une grille. Suivant le type de contrainte, cette application peut concerner un, deux ou trois nœuds. Une contrainte de propriété appliquée à un n-uplet de nœuds de la grille est ce que nous appelons une instance de propriété. Chaque instance de propriété dépend uniquement de la grammaire et de la grille. La détermination d'un modèle d'arbre syntaxique passe par l'évaluation de la pertinence et de la satisfaction de ces instances de propriétés.

5.2 Contraintes pour les propriétés

Pour chaque instance I, on définit deux variables booléennes $P(I)$ indiquant sa pertinence et $S(I)$ indiquant sa pertinence et sa satisfaction.

Linéarité La propriété $A : B \prec C$ nécessite les instances I de la forme :

$$A : B \prec C \ @ \ \langle w_{i_0j_0}, w_{i_1j_1}, w_{i_2j_2} \rangle$$

où $w_{i_1j_1}$ et $w_{i_2j_2}$ sont des fils de $w_{i_0j_0}$ et $w_{i_1j_1} \neq w_{i_2j_2}$. L'instance I est pertinente si le nœud $w_{i_0j_0}$ est actif, si les nœuds $w_{i_1j_1}$ et $w_{i_2j_2}$ sont ses enfants, et chaque nœud est étiqueté par la catégorie demandée :

$$P(I) \Leftrightarrow \left(\begin{array}{l} w_{i_0j_0} \in V^+ \wedge \mathsf{cat}(w_{i_0j_0}) = A \wedge \\ w_{i_1j_1} \in \downarrow w_{i_0j_0} \wedge w_{i_2j_2} \in \downarrow w_{i_0j_0} \wedge \\ \mathsf{cat}(w_{i_1j_1}) = B \wedge \mathsf{cat}(w_{i_2j_2}) = C \end{array} \right)$$

Sa satisfaction est définie par $j_1 < j_2$, la variable $S(I)$ est donc définie par

$$S(I) \quad \Leftrightarrow \quad P(I) \wedge j_1 < j_2$$

Obligation La propriété $A : \triangle B$ indique que chaque nœud d'étiquette A de l'arbre doit avoir au moins un nœud fils étiqueté B. Cette propriété nécessite donc les instances I de la forme :

$$A : \triangle B \ @ \ \langle w_{i_0j_0} \rangle$$

Une instance est pertinente si $w_{i_0j_0}$ est un nœud actif étiqueté par la catégorie A :

$$P(I) \quad \Leftrightarrow \quad w_{i_0j_0} \in V^+ \wedge cat(w_{i_0j_0}) = A$$

Elle est satisfaite si au moins un nœud fils est étiqueté par B :

$$S(I) \quad \Leftrightarrow \quad P(I) \wedge \bigvee_{w_{ij} \in \downarrow w_{i_0j_0}} cat(w_{ij}) = B$$

Unicité La propriété $A : B!$ indique que chaque nœud d'étiquette A de l'arbre doit avoir au plus un fils étiqueté B, l'existence est cependant optionnelle. Cette propriété nécessite les instances I de la forme :

$$A : B! \ @ \ \langle w_{i_0j_0}, w_{i_1j_1}, w_{i_2j_2} \rangle$$

où $w_{i_1j_1}$ et $w_{i_2j_2}$ sont des fils de $w_{i_0j_0}$ et $w_{i_1j_1} \neq w_{i_2j_2}$. L'instance I est pertinente si le nœud $w_{i_0j_0}$ est actif et étiqueté par A et les nœuds $w_{i_1j_1}$ et $w_{i_2j_2}$ sont ses enfants et sont étiquetés par B :

$$P(I) \Leftrightarrow \left(\begin{array}{l} w_{i_0j_0} \in V^+ \wedge \mathsf{cat}(w_{i_0j_0}) = A \wedge \\ w_{i_1j_1} \in \downarrow w_{i_0j_0} \wedge w_{i_2j_2} \in \downarrow w_{i_0j_0} \wedge \\ \mathsf{cat}(w_{i_1j_1}) = B \wedge \mathsf{cat}(w_{i_2j_2}) = B \end{array} \right)$$

Elle est satisfaite si $w_{i_1j_1}$ et $w_{i_2j_2}$ sont le même nœud :

$$S(I) \quad \Leftrightarrow \quad P(I) \wedge w_{i_1j_1} = w_{i_2j_2}$$

Exigence La propriété $A : B \Rightarrow C$ indique que pour tout nœud d'étiquette A de l'arbre, la présence d'un nœud fils étiqueté B implique la présence d'un nœud fils étiqueté C. Cette propriété nécessite donc les instances I de la forme :

$$A : B \Rightarrow C \ @ \ \langle w_{i_0j_0}, w_{i_1j_1} \rangle$$

Une instance est pertinente si le nœud $w_{i_0j_0}$ est actif et étiqueté par A et le nœud $w_{i_1j_1}$ étiqueté par B est un de ses fils :

$$P(I) \Leftrightarrow \left(\begin{array}{l} w_{i_0j_0} \in V^+ \wedge w_{i_1j_1} \in \downarrow w_{i_0j_0} \wedge \\ \mathsf{cat}(w_{i_0j_0}) = A \wedge \mathsf{cat}(w_{i_1j_1}) = B \end{array} \right)$$

Elle est satisfaite si un nœud fils de $w_{i_0j_0}$ est étiqueté par la catégorie C :

$$S(I) \quad \Leftrightarrow \quad P(I) \wedge \bigvee_{w_{ij} \in \downarrow w_{i_0j_0}} \mathsf{cat}(w_{ij}) = C$$

Exclusion La propriété $A : B \not\Leftarrow C$ indique qu'aucun nœud d'étiquette A de l'arbre ne peut avoir à la fois un nœud fils étiqueté B et un nœud fils étiqueté C. Cette propriété nécessite alors les instances I de la forme :

$$A : B \not\Leftarrow C \ @ \ \langle w_{i_0j_0}, w_{i_1j_1}, w_{i_2j_2} \rangle$$

où $w_{i_1j_1}$ et $w_{i_2j_2}$ sont des fils de $w_{i_0j_0}$ et $w_{i_1j_1} \neq w_{i_2j_2}$. L'instance I est pertinente si le nœud $w_{i_0j_0}$ est actif et étiqueté par A et les nœuds $w_{i_1j_1}$ et $w_{i_2j_2}$ sont ses enfants avec soit $w_{i_1j_1}$ est étiqueté par B soit $w_{i_2j_2}$ est étiqueté par C :

$$P(I) \Leftrightarrow \left(\begin{array}{l} w_{i_0j_0} \in V^+ \wedge \mathsf{cat}(w_{i_0j_0}) = A \wedge \\ w_{i_1j_1} \in \downarrow w_{i_0j_0} \wedge w_{i_2j_2} \in \downarrow w_{i_0j_0} \wedge \\ (\mathsf{cat}(w_{i_1j_1}) = B \vee \mathsf{cat}(w_{i_2j_2}) = C) \end{array} \right)$$

Sa pertinence et satisfaction est définie par :

$$S(I) \Leftrightarrow P(I) \wedge (\mathsf{cat}(w_{i_1j_1}) \neq B \vee \mathsf{cat}(w_{i_2j_2}) \neq C)$$

Constituance La propriété $A : S$ indique que pour tout nœud d'étiquette A de l'arbre, les étiquettes des nœuds fils appartiennent à l'ensemble S. Cette propriété nécessite les instances I de la forme :

$$A : S \ @ \ \langle w_{i_0j_0}, w_{i_1j_1} \rangle$$

Une instance est pertinente si le nœud $w_{i_0j_0}$ est actif et étiqueté par A et le nœud $w_{i_1j_1}$ est un de ses enfants :

$$P(I) \Leftrightarrow \left(\begin{array}{l} w_{i_0j_0} \in V^+ \wedge w_{i_1j_1} \in \downarrow w_{i_0j_0} \wedge \\ \mathsf{cat}(w_{i_0j_0}) = A \end{array} \right)$$

Elle est satisfaite si la catégorie étiquetant le nœud $w_{i_1j_1}$ est dans S :

$$S(I) \quad \Leftrightarrow \quad P(I) \wedge \mathsf{cat}(w_{i_1j_1}) \in S$$

Contrainte d'optimalité Une instance est comptée si elle est pertinente. Elle est comptée comme positive si elle est pertinente et satisfaite, et comme négative sinon. Soient $\mathcal{I}$ l'ensemble de tous les instances, $\mathcal{I}^0$ l'ensemble des instances pertinentes et $\mathcal{I}^+$ l'ensemble des instances positives. Le degré de satisfaction des propriétés pour l'arbre est donc déterminé par le ratio $|\mathcal{I}^+|/|\mathcal{I}^0|$. Dans le cadre de la sémantique forte, les modèles sont ceux dont ce ratio vaut 1, car toute instance pertinente est satisfaite. Dans le cadre de la sémantique relâchée, il est possible qu'il existe des instances pertinentes mais non satisfaites. Dans ce cadre, les quasi-modèles sont ceux qui maximisent ce ratio.

5.3 Réalisation des contraintes

Nous considérons toujours une grille de taille $n \times m$. Pour chaque propriété nous considérons toutes les instances possibles à partir de chaque point de la grille. Puisque les arbres sont rectangulaires, pour chaque nœud $w_{i_0j_0}$ les nœuds fils sont dans le rectangle $DR(w_{i_0j_0})$ en dessous à droite de $w_{i_0j_0}$ sur la grille. Soient i le numéro de ligne de $w_{i_0j_0}$ et j son numéro de colonne. Le rectangle $DR(w_{i_0j_0})$ est formé par les points (i', j') avec $1 \leq i' < i$ et $j \leq j' \leq m$. Pour les instances de la forme $A : \psi \ @ \ \langle w_{i_0j_0}, w_{i_1j_1} \rangle$, le couple $\langle w_{i_0j_0}, w_{i_1j_1} \rangle$ se forme donc par $w_{i_0j_0}$ et un nœud $w_{i_1j_1} \in DR(w_{i_0j_0})$. Et pour les instances de la forme $A : \psi \ @ \ \langle w_{i_0j_0}, w_{i_1j_1}, w_{i_2j_2} \rangle$, le triplet $\langle w_{i_0j_0}, w_{i_1j_1}, w_{i_2j_2} \rangle$ se forme par $w_{i_0j_0}$ et deux nœuds $w_{i_1j_1}, w_{i_2j_2} \in DR(w_{i_0j_0})$. Étant données une grille et une grammaire, nous pouvons donc calculer le nombre d'instances possibles. Soit $\mathcal{I}$ l'ensemble de ces instances. En considérant les propriétés dans un ordre fixé (par exemple l'ordre de leur apparition dans la grammaire), nous pouvons ordonner les instances de $\mathcal{I}$ et accéder à chacune par un indice. Remarquons également que le nombre d'instances ne dépend que de la grammaire et de la taille de la grille, il ne dépend pas de l'expression à analyser.

Les variables booléennes $P(I)$ et $S(I)$ sont donc organisées en deux tableaux P et S indexés sur les instances de $\mathcal{I}$. Les conditions de pertinence et de satisfaction sont pour la plupart des propriétés des conjonctions et/ou disjonctions des contraintes d'égalité ou d'appartenance. Les appartenances sont représentées par des variables booléennes, dont la valeur est définie par une contrainte réifiée. Les valeurs de $P(I)$ et $S(I)$ pour chaque instance I sont également reliées à ces conditions par des contraintes réifiées.

Dans le cadre de la sémantique relâchée, les modèles recherchés sont ceux qui maximisent le ratio $|\mathcal{I}^+|/|\mathcal{I}^0|$, où $\mathcal{I}^0$ est l'ensemble des instances pertinentes et $\mathcal{I}^+$ l'ensemble des instances pertinentes et satisfaites. Du fait que pour chaque instance I, les variables $P(I)$ et $S(I)$ sont booléennes, leur valeurs peuvent être considérées comme 0 ou 1. La cardinalité de ces ensembles peut être calculée par :

$$|\mathcal{I}^0| = \sum_{I \in \mathcal{I}} P(I) \qquad |\mathcal{I}^+| = \sum_{I \in \mathcal{I}} S(I)$$

Dans le cadre de la sémantique forte, du fait que chaque instance pertinente doit aussi être satisfaite, il suffit d'ajouter, pour chaque instance I, une contrainte d'implication $P(I) \Rightarrow S(I)$.

6 Implantation d'un prototype

L'approche à base de CSP décrite dans cet article a été implantée en utilisant la bibliothèque de programmation par contraintes Gecode [9].

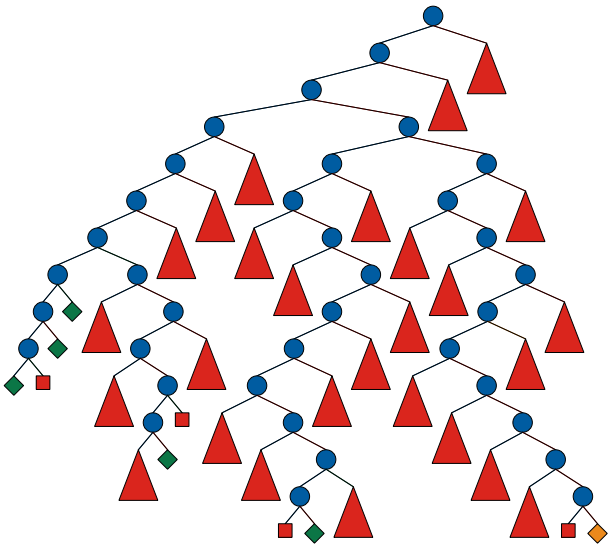


FIG. 1 – Arbre de recherche de l'analyse optimale de "Pierre mange la pomme"

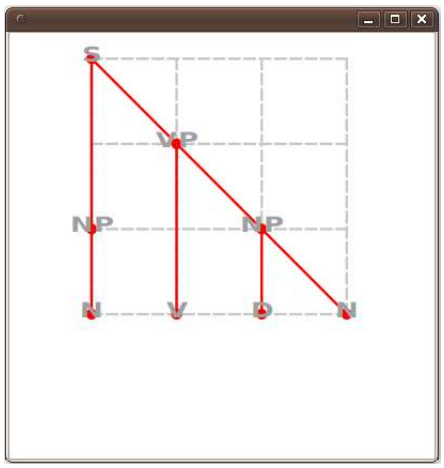


FIG. 2 – Arbre syntaxique optimal de "Pierre mange la pomme"

Notons que le but ici n'est pas tellement de développer un analyseur performant, en effet les analyseurs à base de modèles probabilistes sont actuellement bien plus rapides que les analyseurs à base de modèles symboliques. L'intérêt de notre approche est plutôt d'étudier les conséquences logiques d'une modélisation des grammaires de propriétés en théorie des modèles, et dans ce contexte, de pouvoir évaluer le degré de grammaticalité d'un énoncé.

Comme cela a été mentionné en section 5, les définitions de la pertinence $P(I)$ et satisfaction $S(I)$ d'instance de propriété sont représentées au moyen de contraintes réifiées. La recherche d'un arbre syntaxique optimal est réalisée en utilisant une stratégie de recherche de type *branch-and-bound* maximisant le ratio $|\mathcal{I}^+|/|\mathcal{I}^0|$.

La figure 1 donne un ordre d'idée de la taille de l'arbre de recherche de l'analyse syntaxique optimale pour l'exemple de la section 3 (grammaire de propriétés $\mathcal{G}$ et phrase "Pierre mange la pomme").[8] Cet arbre comporte près de 450 000 nœuds, correspondant à des points de choix lors de la recherche des valeurs à assigner aux variables de notre CSP. Parmi ces 450 000 nœuds, 6 sont des solutions (représentées sous forme de losanges dans l'arbre de recherche). La solution optimale est colorée en orange et correspond à l'arbre syntaxique de la figure 2.

Ce prototype est encore en cours de développement, en particulier il ne bénéficie pas encore d'une interface graphique d'entrée, ni d'un éditeur de grammaires de propriétés. Cependant, il est disponible sous licence GPL sur demande.

Actuellement, il est utilisé avec une grammaire jouet, néanmoins nous prévoyons de l'utiliser avec la grammaire du français de Prost [12].

7 Comparaison avec les travaux existants

Les grammaires de propriétés sont un formalisme relativement jeune. Néanmoins, nous ne sommes pas les premiers à nous être intéressés au problème de l'analyse syntaxique pour ce formalisme.

En particulier, nous pouvons citer les travaux de van Rullen [14], ou encore ceux cités précédemment de Prost [12]. La différence majeure avec notre approche réside dans le fait que nous nous basons sur une sémantique formelle des grammaires de propriétés en théorie des contraintes, permettant une modélisétion naturelle de l'analyse sous forme de CSP.

Cette analyse de GP sous forme de CSP n'est pas sans rappeler les travaux de Estratat et Henocque sur les grammaires de configuration [8]. Dans leurs travaux, les auteurs traduisent une grammaire de propriété décrivant un fragment de la langue naturelle, sous forme d'un modèle objet contraint, représenté avec le langage Z. La description grammaticale résultante est ensuite passée à un configurateur, dont le rôle est de calculer les analyses syntaxiques. Il convient de noter que les auteurs proposent un cadre grammatical général censé permettre l'analyse syntaxique pour plusieurs formalismes grammaticaux, pas uniquement les grammaires de propriétés.

La différence principale entre l'approche d'Estratat et Henocque et la notre, est que le passage par

[8] Cette représentation graphique de l'arbre de recherche est obtenue au moyen de l'outil *Gist* intégré à Gecode.

un configurateur traitant une description grammaticale permet uniquement l'analyse d'énoncés grammaticaux. L'un des atouts des grammaires de propriétés résidant dans la possibilité d'associer à un énoncé agrammatical une valeur de grammaticalité, il est préférable de pouvoir utiliser cette caractéristique du formalisme. La sémantique formelle de Duchier *et al* [7] à base de contraintes relâchées permet cela.

8 Conclusion

Duchier *et al* [7] ont défini formellement une sémantique en théorie des modèles pour les grammaires de propriétés. Dans cet article, nous avons présenté une modélisation en CSP de cette formalisation. Cette modélisation ouvre la voie à l'implantation d'un analyseur syntaxique à base de contraintes, calculant des arbres syntaxiques optimaux (en termes de grammaticalité d'énoncé). Un prototype d'analyseur a été développé, et a permis de commencer à expérimenter l'analyse d'énoncés grammaticaux et agrammaticaux.

Dans sa version actuelle, l'analyseur décrit ici manipule un grand nombre d'instances de propriétés, même sur des énoncés de taille réduite, ce qui se traduit par un arbre de recherche de grande taille. Dans ce contexte, nous souhaitons optimiser la recherche des solutions. L'une des pistes dans ce sens, correspond à paralléliser l'exploration de l'arbre de recherche des solutions du CSP, en utilisant par exemple les travaux de [5]. Une autre piste consisterait à coupler un reconnaisseur de constituants probabliste à l'analyseur pour réduire le nombre de constituants manipulés et ainsi le nombre de contraintes.

Enfin, nous allons travailler au développement d'un moteur de recherche de solutions dédié, car le moteur utilisé actuellement ne permet de conserver que la meilleure solution du CSP, or nous souhaiterions conserver toutes les meilleures solutions lorsqu'il y a plusieurs modèles de même score.

Remerciements

Merci à Sylvie Billot, Mathieu Lopez, Jean-Philippe Prost et Isabelle Tellier pour les interactions fructueuses sur ce travail.

Références

[1] Nicolas Beldiceanu, Pierre Flener, and Xavier Lorca. The tree constraint. In *CPAIOR*, pages 64–78, 2005.

[2] Philippe Blache. *Les Grammaires de Propriétés : des contraintes pour le traitement automatique des langues naturelles.* Hermès Sciences, 2001.

[3] Philippe Blache. Property Grammars : a Fully Constraint-Based Theory. In J. Villadsen H. Christiansen, P. Rossen Skadhauge, editor, *Constraint Solving and Language Processing*, volume 3438 of *Lecture Notes in Artificial Intelligence*, pages 1–16. Springer, 2004.

[4] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory IT-2*, 2(3) :113–124, 1956.

[5] Geoffrey Chu, Christian Schulte, and Peter J. Stuckey. Confidence-based work stealing in parallel constraint programming. In *15th International Conference on Principles and Practice of Constraint Programming*, volume 5732 of *LNCS*, pages 226–241, Lisbon, Portugal, 2009. Springer.

[6] Denys Duchier. Configuration of labeled trees under lexicalized constraints and principles. *Journal of Research on Language and Computation*, 1(3/4), September 2003.

[7] 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, June 2009.

[8] Mathieu Estratat and Laurent Henocque. Les grammaires de configuration : un cadre grammatical moderne. In *Troisièmes Journées Francophones de Programmation par Contraintes (JFPC07)*, Rocquencourt, France, 2007.

[9] Gecode Team. Gecode : Generic constraint development environment, 2010. Available from `http ://www.gecode.org`.

[10] Aravind Joshi, Leon Levy, and Masako Takahashi. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1) :136–163, 1975.

[11] Patrick Prosser and Chris Unsworth. Rooted tree and spanning tree constraints. In *17th ECAI Workshop on Modelling and Solving Problems with Constraints*, 2006.

[12] 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.

[13] Geoffrey Pullum and Barbara Scholz. On the Distinction Between Model-Theoretic and Generative-Enumerative Syntactic Frameworks. In *Logical Aspects of Computational Linguistics : 4th International Conference*, volume 2099 of *LNAI*, pages 17–43. Springer, 2001.

[14] Tristan van Rullen. *Vers une analyse syntaxique à granularité variable.* PhD thesis, Université de Provence, Aix-Marseille 1, France, 2005.