Associative Grammar Combination Operators for Tree-Based Grammars

Yael Sygal
Department of Computer Science
University of Haifa, Israel
yaelc@cs.haifa.ac.il

Shuly Wintner
Department of Computer Science
University of Haifa, Israel
shuly@cs.haifa.ac.il

Abstract

Polarized unification grammar (PUG) is a linguistic formalism which uses *polarities* to better control the way grammar fragments interact. The grammar combination operation of PUG was conjectured to be associative. We show that PUG grammar combination is *not* associative, and even attaching polarities to objects does not make it order-independent. Moreover, we prove that *no* non-trivial polarity system exists for which grammar combination *is* associative. We then redefine the grammar combination operator, moving to the powerset domain, in a way that guarantees associativity. The method we propose is general and is applicable to a variety of tree-based grammar formalisms.

1 Introduction

Development of large-scale grammars for natural languages is an active area of research in human language technology. Such grammars are developed not only for purposes of theoretical linguistic research, but also for natural language applications such as machine translation, speech generation, etc. Wide-coverage grammars are being developed for various languages in several theoretical frameworks. Grammar development is a complex enterprise: It is not unusual for a single grammar to be developed by a team including several linguists, computational linguists and computer scientists. The scale of grammars can be overwhelming and syntactic structures can be redundantly repeated throughout the grammar. As large-scale grammars introduce issues of modularity, various techniques were introduced to better control the way in which grammar fragments interact.

We focus on tree-based formalisms, inspired by and extending tree-adjoining grammars (TAG, Joshi, Levy, and Takahashi (1975)). A wide-coverage TAG may contain hundreds or even thousands of elementary trees, and syntactic structure can be redundantly repeated in many trees (XTAG Research Group, 2001; Abeillé, Candito, and Kinyon, 2000). Consequently, maintenance and extension of such grammars is a complex task. To address these issues, several high-level formalisms were developed (Vijay-Shanker, 1992; Candito, 1996; Duchier and Gardent, 1999; Kallmeyer, 2001). These formalisms take the *metagrammar approach*, where the basic units are tree descriptions (i.e., formulas denoting sets of trees) rather than trees. Parmentier et al. (2007) extend this approach as their basic units are descriptions of sets ot trees (i.e., forests) and not merely trees. Tree descriptions are constructed by a tree logic and combined through conjunction or inheritance (depending on the formalism). The set of minimal trees that satisfy the resulting descriptions are the TAG elementary trees. In this way modular construction of grammars is supported, where a module is merely a tree description and modules are combined by means of the control tree logic.

When trees are semantic objects, the denotation of tree descriptions, there can be various ways to refer to nodes in the trees in order to control the possible combination of grammar modules. In the metagrammar paradigm, where grammar fragments are tree *descriptions*, Candito (1996) associates with each node in a description a name, such that nodes with the same name must denote the same entity and therefore must be identified. The names of nodes are thus the only channel of interaction between two descriptions. Furthermore, these names can only be used to identify two nodes, but not to set nodes apart. Crabbé and Duchier (2004) propose to replace node naming by a *coloring* scheme, where nodes are colored black, white or red. When two trees are combined, a black node may be unified with zero, one or more white nodes and produce a black node; a white node must be unified with a black one producing a black node; and a red node cannot be unified with any other node. Furthermore, a satisfying model must be *saturated*, i.e., one in which all the nodes are either black or red. In this way some combinations can be forced and others prevented.

This mechanism is extended in *Interaction Grammars* (Perrier, 2000), where each node is decorated by a set of *polarity features*. A polarity feature consists of a feature, arbitrarily determined by the grammar writer, and a polarity, which can be either positive, negative or neutral. A positive value represents an available resource and a negative value represents an expected resource. Two feature-polarity pairs can combine only if their feature is identical and their polarities are opposite (i.e., one is negative and the other is positive); the result is a feature-polarity pair consisting of the same feature and the neutral polarity. Two nodes can be identified only if their polarity features can combine. A solution is a tree whose features are all neutralized. Bonfante, Guillaume, and Perrier (2004) use this method to optimize parsing where polarities are used to efficiently filter lexical selections.

The concept of polarities is further elaborated in *Polarized Unification Grammars* (PUG, Kahane (2006)). A PUG is defined over a *system of polarities* $(P, \cdot)$ where P is a set (of polarities) and '$\cdot$' is an associative and commutative product over P. A PUG generates a set of finite structures (e.g., trees) over objects (e.g., nodes) which are determined for each grammar separately. The objects are associated with polarities, and structures are combined by identifying some of their objects. The combination is sanctioned by polarities: objects can only be identified if their polarities are unifiable; the resulting object has the unified polarity. A non-empty, strict subset of the set of polarities, called the set of *neutral* polarities, determines which of the resulting structures are valid: A polarized structure is *saturated* if all its polarities are neutral, and the language generated by the grammar includes the saturated structures that result from all the possible combinations of elementary structures. PUG is a powerful and flexible formalism which was shown to be capable of simulating many grammar formalisms, including TAG, LFG, HPSG, etc.

PUG provides a more general polarity scheme than the mechanisms of polarity features and coloring, since the grammar designer has the freedom to define the system of polarities, whereas other systems pre-define it. Another difference is that while tree descriptions and Interaction Grammars are limited to trees or sets of trees, PUG manipulates arbitrary structures and objects.

In other tree-based grammars, if two nodes are identified then their predecessors must be identified as well (to maintain a tree structure). Even if sets of trees are allowed, identification of nodes is sanctioned to maintain a tree-based structure (e.g., a tree or a forest). Additionally, polarities are attached only to the tree nodes. In PUG, structures can be arbitrary (e.g., general graphs, directed or non-directed, DAGS etc.) and any two objects can be identified as long as their polarities are consistent. Furthermore, polarities are attached to all objects. For example, if the structures are graphs, then polarities are attached to both the nodes and the edges. However, unlike other tree-based formalisms, PUG does not take the metagrammar approach: the basic units are grammatical objects (e.g., trees or graphs) rather than grammatical descriptions (e.g., formulas describing grammatical objects).

The grammar combination operation of PUG was conjectured to be associative (Kahane and Lareau, 2005; Kahane, 2006). In this paper we show that it is not; even attaching polarities to objects does not render grammar combination order-independent. In section 2 we formalize the tree combination operation of PUG and set a common notation. We limit the discussion to the case of trees, rather than the arbitrary objects of PUG, for the sake of simplicity; all our results can easily be extended to arbitrary structures and objects (e.g., graphs and their nodes and edges). In section 3 we show that existing polarity systems do not guarantee associativity. This is not accidental: we prove that no non-trivial polarity system can guarantee the associativity of grammar combination. We analyze the reasons for this in section 4 and introduce new definitions, based on a move from trees to forests, which induce an associative grammar combination operator. The immediate contribution of this short paper is thus the identification—and correction—of a significant flaw in this otherwise powerful and flexible formalism. Moreover, the method we propose is general, and therefore applicable to a variety of formalisms. In section 5 we show that our results can be used to define an alternative semantics for XMG (Duchier, Le Roux, and Parmentier, 2004; Crabbé, 2005), a commonly used metagrammar formalism. We then conclude with suggestions for future research.

2 Tree combination in PUG

To the best of our knowledge, no formal definition of PUG was published and the formalism is only discussed informally (Kahane and Lareau, 2005; Kahane, 2006). We therefore begin by defining the formalism and its combination operation, both with and without polarities, to establish a common notation.

Definition 1. *A* ***tree*** *$\langle V, E, r\rangle$ is a connected, undirected, acyclic graph with vertices V, edges E and a unique root $r \in V$.*

Every pair of nodes in a tree is connected by a unique path, and the edges have a natural orientation, toward or away from the root. Let $\langle V, E, r\rangle$ be a tree and let $v \in V$. Any vertex u which is located on the single path from r to v is an **ancestor** of v, and v is a **descendant** of u. If the last arc on the path from r to v is (u, v) then u is the **parent** of v and v is the **child** of u. The meta-variable T ranges over trees and V, E, r over their components. The meta-variable $\mathcal{T}$ ranges over sets of trees.

Definition 2. *Two trees T_1, T_2 are* **disjoint** *if $V_1 \cap V_2 = \emptyset$. Two sets of trees $\mathcal{T}_1, \mathcal{T}_2$ are* **disjoint** *if for all $T_1 \in \mathcal{T}_1$, $T_2 \in \mathcal{T}_2$, $V_1 \cap V_2 = \emptyset$.*

Definition 3. *Two trees $T_1 = \langle V_1, E_1, r_1\rangle$, $T_2 = \langle V_2, E_2, r_2\rangle$ are* **isomorphic***, denoted $T_1 \sim T_2$, if there exists a total one to one and onto function $i : V_1 \to V_2$ such that $i(r_1) = r_2$ and for all $u, v \in V_1$, $(u, v) \in E_1$ iff $(i(u), i(v)) \in E_2$. Two sets of trees $\mathcal{T}_1, \mathcal{T}_2$ are* **isomorphic***, denoted $\mathcal{T}_1 \cong \mathcal{T}_2$, if there exist total functions $i_1 : \mathcal{T}_1 \to \mathcal{T}_2$ and $i_2 : \mathcal{T}_2 \to \mathcal{T}_1$ such that for all $T \in \mathcal{T}_1$, $T \sim i_1(T)$ and for all $T \in \mathcal{T}_2$, $T \sim i_2(T)$.*

Next, we define how two trees are combined. An equivalence relation over the nodes of the two trees states which nodes should be identified. In the result of the combination, nodes are equivalence classes of that relation and arcs connect nodes that are connected in their members. The equivalence relation is sanctioned in a way that guarantees that the resulting graph is indeed a tree.

Definition 4. *Let $T_1 = \langle V_1, E_1, r_1\rangle$, $T_2 = \langle V_2, E_2, r_2\rangle$ be two disjoint trees. An equivalence relation '$\overset{t}{\approx}$' over $V_1 \cup V_2$ is* **legal** *if all the following hold:*[1]

1. *for all $v_1, v_2 \in V_1 \cup V_2$, if $v_1 \overset{t}{\approx} v_2$ and $v_1 \neq v_2$ then either $v_1 \in V_1$ and $v_2 \in V_2$ or $v_1 \in V_2$ and $v_2 \in V_1$*

2. *for all $u_1, v_1, u_2, v_2 \in V_1 \cup V_2$, if $v_1 \overset{t}{\approx} v_2$, u_1 is the parent of v_1 and u_2 is the parent of v_2, then $u_1 \overset{t}{\approx} u_2$*

3. *there exists $v \in V_1 \cup V_2$ such that $|[v]_{\overset{t}{\approx}}| > 1$*

$Eq_t(T_1, T_2)$ is the set of legal equivalence relations over $V_1 \cup V_2$.

The first condition of definition 4 states that when two nodes are identified, they must belong to different trees. The second condition requires that when two nodes are identified, all their ancestors must identify as well. Finally, the last condition requires that at least two nodes (each from a different tree) be identified. The first two conditions guarantee that the resulting graph is acyclic and the third guarantees that it is connected.[2]

Definition 5. *Let $T_1 = \langle V_1, E_1, r_1\rangle$, $T_2 = \langle V_2, E_2, r_2\rangle$ be two disjoint trees and let '$\overset{t}{\approx}$' be a legal equivalence relation over $V_1 \cup V_2$. The* **tree combination** *of T_1, T_2* **with respect to '$\overset{t}{\approx}$'**, *denoted $T_1 +_{\overset{t}{\approx}} T_2$, is a tree $T = \langle V, E, r\rangle$, where:*

- $V = \{[v]_{\overset{t}{\approx}} \mid v \in V_1 \cup V_2\}$
- $E = \{([u]_{\overset{t}{\approx}}, [v]_{\overset{t}{\approx}}) \mid (u, v) \in E_1 \cup E_2\}$
- $r = \begin{cases} [r_1]_{\overset{t}{\approx}} & \textit{if } [r_1]_{\overset{t}{\approx}} = \{r_1\} \textit{ or } [r_1]_{\overset{t}{\approx}} = \{r_1, r_2\} \\ [r_2]_{\overset{t}{\approx}} & \textit{otherwise} \end{cases}$

[1]If '$\equiv$' is an equivalence relation then $[v]_\equiv$ is the equivalence class of v with respect to '$\equiv$'.

[2]The second condition is not an original requirement of PUG; it is added for the case in which the basic structures are trees to guarantee that the resulting graph is indeed a tree.

When two trees are combined, nodes belonging to the same equivalence class are identified. Since the equivalence relation is legal, the resulting graph is indeed a tree. Observe that since the equivalence relation is legal, either the two roots are identified; or one of them is identified with a non-root node and the other remains alone. In the former case, the root of the new tree is the node created from the identification of the two roots; in the latter case, the new root is the root whose equivalence class is a singleton. In definition 5, a systematic replacement of r_1 and r_2 in the definition of r would have yielded the same result.

Example 1. *Figure 1 depicts three trees, T_1, T_2, T_3. T and T' are tree combinations of T_1 and T_2. T is obtained by identifying q_1 with q_3 and q_2 with q_4. Notice that since q_2 is identified with q_4, q_1 must be identified with q_3 to maintain a tree structure (condition 2 of definition 4). T' is obtained by identifying q_2 with q_3. T'' is not a tree combination of T_2 and T_3 since it identifies q_6 with q_7, which belong to the same tree, T_3, in contradiction to condition 1 of definition 4.*

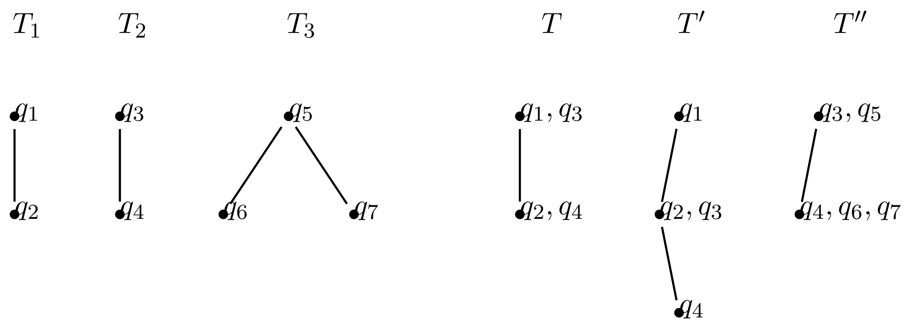

Figure 1: Tree combination

Definition 6. *Let $\mathcal{T}_1, \mathcal{T}_2$ be two disjoint sets of trees. The* **tree combination** *of $\mathcal{T}_1, \mathcal{T}_2$, denoted $\mathcal{T}_1 \overset{t}{+} \mathcal{T}_2$, is the set of trees*

$$\mathcal{T} = \bigcup_{\substack{T_1 \in \mathcal{T}_1, T_2 \in \mathcal{T}_2 \\ \overset{t}{\approx} \in Eq_t(T_1, T_2)}} T_1 +_{\overset{t}{\approx}} T_2$$

The tree combination operation takes as input two sets of trees and yields a set of trees which includes *all* the tree combinations of any possible pair of trees belonging to the two different sets with respect to *any* possible legal equivalence relations. Notice that the definitions allows multiple isomorphic trees in the same set of trees, which may result in inefficient processing. It is assumed that the grammar designer is responsible for avoiding such inefficiency.

Example 2. *The sets of trees defined by* $\{T_1\}\overset{t}{+}\{T_2\}$ *and* $\{T_2\}\overset{t}{+}\{T_3\}$ *(Figure 1) are depicted in Figures 2 and 3, respectively.*

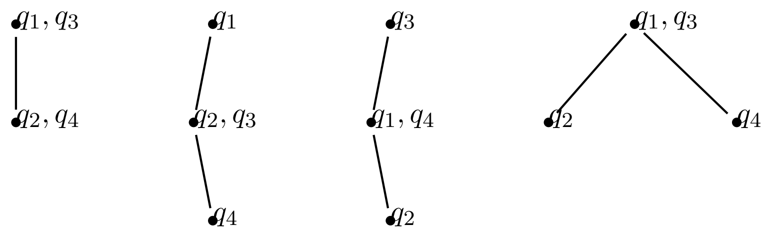

Figure 2: $\{T_1\}\overset{t}{+}\{T_2\}$

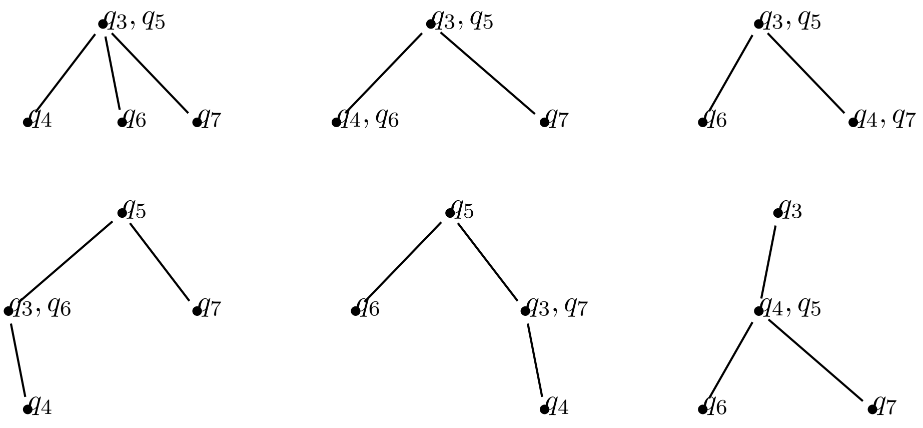

Figure 3: $\{T_2\}\overset{t}{+}\{T_3\}$

This combination operation is extended by attaching polarities to nodes (Crabbé and Duchier, 2004; Perrier, 2000; Kahane, 2006). In a polarized framework, an extra condition for the identification of two nodes is that their polarities combine; in this case a new node (obtained by identifying two nodes) has a polarity which is the product of the polarities of the two identified nodes.

Definition 7. *A system of polarities is a pair $(P, \cdot)$, where P is a non-empty set and '$\cdot$' is a commutative and associative product over $P \times P$.*

In the sequel, if $(P, \cdot)$ is a system of polarities and $a, b \in P$, $ab{\downarrow}$ means that the combination of a and b is defined and $ab{\uparrow}$ means that a and b cannot combine. For the following discussion we assume that a system of polarities $(P, \cdot)$ has been specified.

Definition 8. *A* **polarized tree** *$\langle V, E, r, p\rangle$ is a tree in which each node is assigned a polarity through a total function $p : V \to P$. If $\langle V, E, r, p\rangle$ is a polarized tree then $\langle V, E, r\rangle$ is its* **underlying tree**. *Two polarized trees are* **disjoint** *if their underlying trees are disjoint.*

Definition 9. *Two polarized trees $T_1 = \langle V_1, E_1, r_1, p_1\rangle$, $T_2 = \langle V_2, E_2, r_2, p_2\rangle$ are* **isomorphic**, *denoted $T_1 \sim T_2$, if their underlying trees are isomorphic and, additionally, for all $v \in V_1$, $p_1(v) = p_2(i(v))$. The definition of isomorphism of sets of trees is trivially extended to sets of polarized trees.*

Definition 10. *Let $T_1 = \langle V_1, E_1, r_1, p_1\rangle$, $T_2 = \langle V_2, E_2, r_2, p_2\rangle$ be two disjoint polarized trees. An equivalence relation '$\overset{t}{\approx}$' over $V_1 \cup V_2$ is* **legal** *if it is legal over the underlying trees of T_1 and T_2 and, additionally, for all $v_1 \in V_1$ and $v_2 \in V_2$, if $v_1 \overset{t}{\approx} v_2$, then $p_1(v_1) \cdot p_2(v_2){\downarrow}$. $Eq_t(T_1, T_2)$ is the set of legal equivalence relations over $V_1 \cup V_2$.*

Definition 11. *Let $T_1 = \langle V_1, E_1, r_1, p_1\rangle$, $T_2 = \langle V_2, E_2, r_2, p_2\rangle$ be two disjoint polarized trees and let '$\overset{t}{\approx}$' be a legal equivalence relation over $V_1 \cup V_2$. The* **polarized tree combination** *of T_1, T_2* **with respect to '$\overset{t}{\approx}$'**, *denoted $T_1 +_{\overset{t}{\approx}} T_2$, is a tree $T = \langle V, E, r, p\rangle$ where V, E and r are as in definition 5, and for all $[v]_{\overset{t}{\approx}} \in V$,*

$$p([v]_{\overset{t}{\approx}}) = \begin{cases} (p_1 \cup p_2)(v) & \textit{if } [v]_{\overset{t}{\approx}} = \{v\} \\ (p_1 \cup p_2)(v) \cdot (p_1 \cup p_2)(u) & \textit{if } [v]_{\overset{t}{\approx}} = \{v, u\} \textit{ and } u \neq v \end{cases}$$

Notice that since '$\overset{t}{\approx}$' is legal, p is well defined. The definition of tree combination of sets of trees, denoted '$\overset{t}{+}$', is trivially extended to sets of polarized trees.

The language of a PUG consists of the neutral structures obtained by combining the initial structure and a finite number of elementary structures. In the derivation process, elementary structures combine successively, each new elementary structure combining with at least one object of the previous result.

Definition 12. *A* **Polarized Unification Grammar (PUG)** *is a structure* $G = \langle T_0, \mathcal{T}, (P, \cdot)\rangle$ *where* $\mathcal{T}$ *is a set of polarized trees,* $T_0 \in \mathcal{T}$ *is the initial tree and* $(P, \cdot)$ *is the system of polarities over which the polarized tree combination is defined.*

Let A_i *be a sequence of tree sets where* $A_0 = \{T_0\} \overset{t}{+} \mathcal{T}$ *and for all* i*,* $i \geq 1$*,* $A_i = A_{i-1} \overset{t}{+} \mathcal{T}$*. The* **language** *generated by* G*, denoted* $L(G)$*, is* $L(G) = \bigcup\limits_{i \in \mathbb{N}} A_i$*.*

PUG is a powerful grammatical formalism that was shown to be capable of simulating various linguistic theories (Kahane, 2006). It can be instrumental for grammar engineering, and in particular for modular development of large-scale grammars, where grammar fragments are developed separately and are combined using the basic combination operation defined above. A pre-requisite for such an application is obviously that the grammar combination operation be associative: one would naturally expect that, if '$\circ$' is a grammar combination operator, then $G_1 \circ (G_2 \circ G_3) = (G_1 \circ G_2) \circ G_3$ for any three grammars (and, therefore, $L(G_1 \circ (G_2 \circ G_3)) = L((G_1 \circ G_2) \circ G_3)$).

The grammar combination operation of PUG was indeed conjectured to be associative (Kahane and Lareau, 2005; Kahane, 2006). The present paper makes two main contributions: In the next section we show that the combination operation defined above is *not* associative. In section 4 we introduce an alternative combination operation which we prove to be associative. We thus remedy the shortcoming of the original definition, and render PUG a more suitable formalism for modular grammar development.

3 Tree combination is not associative

In this section we show that tree combination as defined above, with or without polarities, is not associative. In the examples below, the relation which determines how polarities combine *is* indeed associative; it is the tree combination operation which uses polarities that is shown to be non-associative.

3.1 Non-polarized tree combination

Theorem 1. *(Non-polarized) tree combination is a non-associative operation: there exist sets of trees* $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ *such that* $\mathcal{T}_1 \overset{t}{+} (\mathcal{T}_2 \overset{t}{+} \mathcal{T}_3) \not\cong (\mathcal{T}_1 \overset{t}{+} \mathcal{T}_2) \overset{t}{+} \mathcal{T}_3$.

Proof. Consider again T_1, T_2, T_3 of Figure 1 and the sets of trees defined by $\{T_1\} \overset{t}{+} \{T_2\}$ and $\{T_2\} \overset{t}{+} \{T_3\}$, depicted in Figures 2 and 3, respectively. T_4 of Figure 4 is a member of $\{T_2\} \overset{t}{+} \{T_3\}$, obtained by identifying q_3 of T_2 and q_6 of T_3. Similarly, T_5 of Figure 4 is a member of $\{T_1\} \overset{t}{+} \{T_4\}$. Hence $T_5 \in \{T_1\} \overset{t}{+} (\{T_2\} \overset{t}{+} \{T_3\})$. However, T_5 (or any tree isomorphic to it) is not a member of $(\{T_1\} \overset{t}{+} \{T_2\}) \overset{t}{+} \{T_3\}$. $\square$

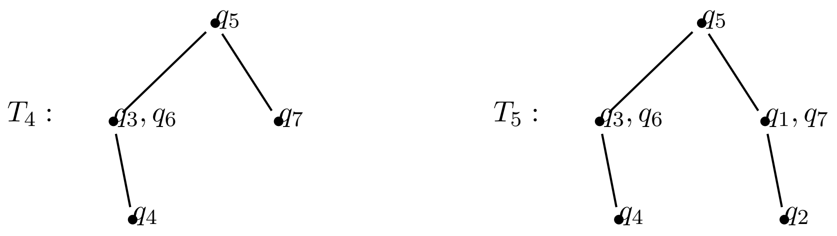

Figure 4: Non-polarized tree combination

3.2 Colors

Crabbé and Duchier (2004) use *colors* to sanction tree node identification. Their color combination table is presented in Figure 5. W, B and R denote white, black and red, respectively, and $\perp$ represents the impossibility to combine.

Theorem 2. *The color scheme of Figure 5 does not guarantee associativity: Let* $(P, \cdot)$ *be the system of Figure 5. Then there exist sets of trees* $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ *such that* $\mathcal{T}_1 \overset{t}{+} (\mathcal{T}_2 \overset{t}{+} \mathcal{T}_3) \not\cong (\mathcal{T}_1 \overset{t}{+} \mathcal{T}_2) \overset{t}{+} \mathcal{T}_3$.

$\cdot$	W	B	R
W	W	B	$\perp$
B	B	$\perp$	$\perp$
R	$\perp$	$\perp$	$\perp$

Figure 5: Color combination table

Proof. Consider Figure 6. The results of combining $\{T_9\}, \{T_{10}\}, \{T_{11}\}$ in different orders demonstrate that $(\{T_9\}\overset{t}{+}\{T_{10}\})\overset{t}{+}\{T_{11}\}\ncong\{T_9\}\overset{t}{+}(\{T_{10}\}\overset{t}{+}\{T_{11}\})$. $\square$

Notice that in Figure 6 all the intermediate and final solutions are saturated. Therefore, the saturation rule does not guarantee associativity.

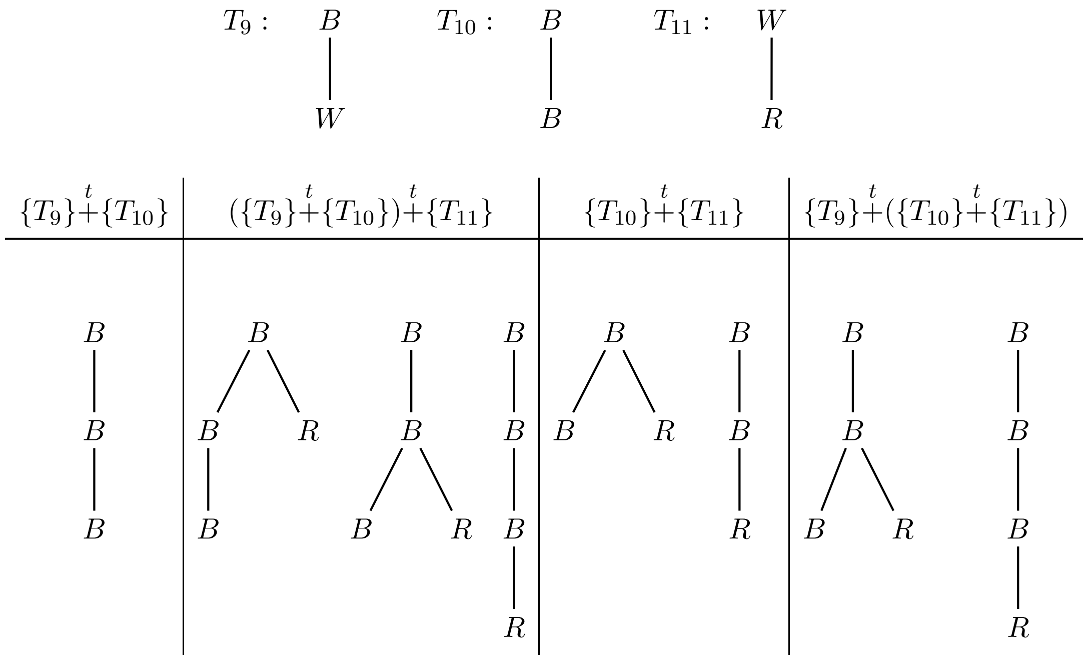
T_9 :
T_{10} :
T_{11} :
$\{T_9\}\overset{t}{+}\{T_{10}\}$
$(\{T_9\}\overset{t}{+}\{T_{10}\})\overset{t}{+}\{T_{11}\}$
$\{T_{10}\}\overset{t}{+}\{T_{11}\}$
$\{T_9\}\overset{t}{+}(\{T_{10}\}\overset{t}{+}\{T_{11}\})$

Figure 6: Tree combination with colors

3.3 Polarities

Kahane and Lareau (2005) and Kahane (2006) use two systems of polarities which are depicted in Figure 7. The first system includes three polarities, gray, white and black, where the neutral polarities are black and gray. A black node may be unified with 0, 1 or more gray or white nodes and produce a black node; a white node may absorb 0, 1 or more gray or white nodes but eventually must be unified with a black one producing a black node; and a gray node may be absorbed into a white or a black node. The second system extends the first by adding two more non-neutral polarities, plus and minus, which may absorb 0, 1 or more white or gray nodes but eventually a plus node must be unified with a minus node to produce a black node.

·	◍	○	●
◍	◍	○	●
○	○	○	●
●	●	●	⊥

·	◍	○	−	+	●
◍	◍	○	−	+	●
○	○	○	−	+	●
−	−	−	⊥	●	⊥
+	+	+	●	⊥	⊥
●	●	●	⊥	⊥	⊥

Figure 7: PUG polarity systems

Theorem 3. *PUG combination with either of the polarity systems of Figure 7 is not associative.*

Proof. Consider Figure 8. Clearly, $\{T_{12}\}\overset{t}{+}(\{T_{13}\}\overset{t}{+}\{T_{14}\})\not\cong(\{T_{12}\}\overset{t}{+}\{T_{13}\})\overset{t}{+}\{T_{14}\}$. □

3.4 General Polarity Systems

We showed above that some existing polarity systems yield non-associative grammar combination operators. This is not accidental; in what follows we show that the only polarity scheme that

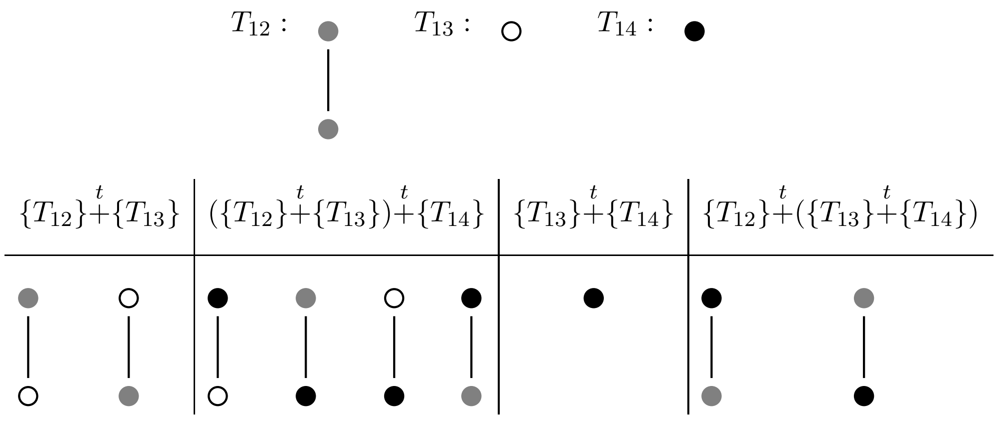
T_{12} :
T_{13} :
T_{14} :
$\{T_{12}\}\overset{t}{+}\{T_{13}\}$
$(\{T_{12}\}\overset{t}{+}\{T_{13}\})\overset{t}{+}\{T_{14}\}$
$\{T_{13}\}\overset{t}{+}\{T_{14}\}$
$\{T_{12}\}\overset{t}{+}(\{T_{13}\}\overset{t}{+}\{T_{14}\})$

Figure 8: Tree combination with polarities

induces associative tree combination is trivial: the one in which no pair of polarities are unifiable. This scheme is useless for sanctioning tree combination since it disallows any combination.

Definition 13. *A system of polarities $(P, \cdot)$ is trivial if for all $a, b \in P$, $ab\uparrow$.*

Theorem 4. *Let $(P, \cdot)$ be a system of polarities. If there exists $a \in P$ such that $aa\downarrow$ then the polarized tree combination based on $(P, \cdot)$ is not associative.*

Proof. Let $(P, \cdot)$ be a system of polarities and let $a \in P$ be such that $aa\downarrow$. Assume toward a contradiction that the polarized tree combination based on $(P, \cdot)$ is associative. Consider T_1, T_2, T_3 of Figure 1 and T_5 of Figure 4. Let T'_1, T'_2, T'_3, T'_5 be polarized trees obtained by attaching the polarity 'a' to all tree nodes of T_1, T_2, T_3, T_5, respectively. $T'_5 \in \{T'_1\}\overset{t}{+}(\{T'_2\}\overset{t}{+}\{T'_3\})$, but T'_5 (or any tree isomorphic to it) is not a member of $(\{T'_1\}\overset{t}{+}\{T'_2\})\overset{t}{+}\{T'_3\}$ (see the proof of theorem 1 for the complete details). Clearly $\{T'_1\}\overset{t}{+}(\{T'_2\}\overset{t}{+}\{T'_3\}) \ncong (\{T'_1\}\overset{t}{+}\{T'_2\})\overset{t}{+}\{T'_3\}$, a contradiction. □

Theorem 5. *Let $(P, \cdot)$ be a non-trivial system of polarities. Then the polarized tree combination based on $(P, \cdot)$ is not associative.*

Proof. Let $(P, \cdot)$ be a non-trivial system of polarities. If $|P| = 1$ then let $P = \{a\}$. Since P is non-trivial, $aa = a$. Then, by theorem 4, $(P, \cdot)$ is not associative. Now assume that $|P| > 1$. Assume toward a contradiction that the polarized tree combination based on $(P, \cdot)$ is associative. There are two possible cases:

1. There exists $a \in P$ such that $aa\downarrow$: Then from theorem 4 it follows that the resulting tree combination operation is not associative, a contradiction.

2. For all $a \in P$, $aa\uparrow$: Then since $(P, \cdot)$ is non-trivial and since $|P| > 1$, there exist $b, c \in P$ such that $b \neq c$, $bb\uparrow$, $cc\uparrow$ and $bc\downarrow$. Consider T_1, T_2, T_3 of Figure 9. Of all the trees in $(\{T_1\} \overset{t}{+} \{T_2\}) \overset{t}{+} \{T_3\}$ and $\{T_1\} \overset{t}{+} (\{T_2\} \overset{t}{+} \{T_3\})$, focus on paths of length 3. All possible instantiations of these trees are depicted in Figure 9 (we suppress the intermediate results). Notice that these trees are only candidate solutions; they are actually accepted only if the polarity combinations occurring in them are defined. Since $bb\uparrow$, $cc\uparrow$ and $bc\downarrow$, $(\{T_1\} \overset{t}{+} \{T_2\}) \overset{t}{+} \{T_3\}$ has no solutions and $\{T_1\} \overset{t}{+} (\{T_2\} \overset{t}{+} \{T_3\})$ has one accepted solution (the rightmost tree), a contradiction.

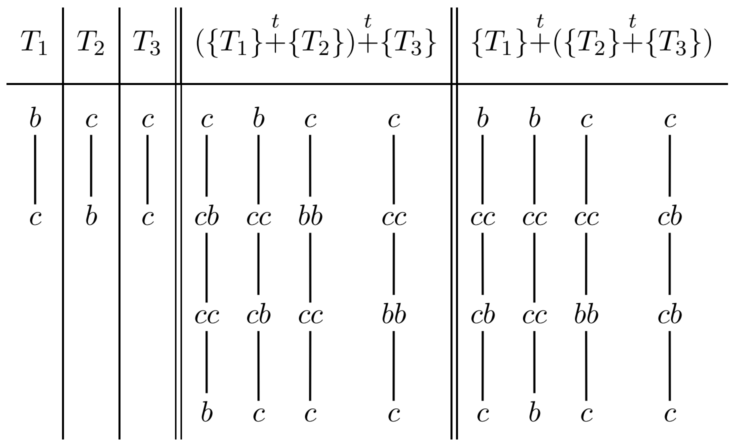

Figure 9: Candidate solutions for PUG tree combination

□

For the sake of completion, we also mention the reverse direction.

Theorem 6. *Let $(P, \cdot)$ be a trivial system of polarities. Then the polarized tree combination based on $(P, \cdot)$ is associative.*

Proof. If $(P, \cdot)$ is a trivial system of polarities then any combination of two sets of polarized trees results in the empty set (no solutions). Evidently, polarized tree combination based on $(P, \cdot)$ is associative. □

Corollary 7. *Let $(P, \cdot)$ be a system of polarities. Then polarized tree combination based on $(P, \cdot)$ is associative if and only if $(P, \cdot)$ is trivial.*

3.5 Practical consequences

Evidently, (polarized) tree combination induces a non-associative grammar combination for PUG. In some cases the result of the non-associativity is plain overgeneration: For example, in Figure 6, $(\{T_9\}\overset{t}{+}\{T_{10}\})\overset{t}{+}\{T_{11}\}$ strictly includes (and, consequently, overgenerates with respect to) $\{T_9\}\overset{t}{+}(\{T_{10}\}\overset{t}{+}\{T_{11}\})$. In general, however, non-associativity results in two non-equal sets: For example, consider Figure 9 and its candidate solutions for length-3 paths and assume that $cb = bc = bb = cc = b$. The length-3 solutions of this case are depicted in Figure 10. Clearly the resulting sets are not equal but none of them overgenerates with respect to the other. The non-associativity of the combination clearly compromises its usability for (modular) development of large-scale grammars: When the grammar designer wrongly assumes that the combination operation is associative, he or she can take advantage of this misconception to achieve a more efficient computation of the combination. This may lead to an incorrect result (which may sometimes over- or undergenerate with respect to the correct result). Such problems may be difficult to locate due to the size of the grammar.

When a combination *is* associative, the grammar designer is free to conceptualize about the combination of grammar fragments in any order; we trust that this makes the formalism more "friendly" to the grammar engineer, and hence easier to work with. In the next section we analyze the reasons for the non-associativity and introduce new definitions which induce an associative grammar combination operator.

4 From trees to forests

Let us now analyze the reasons for the non-associativity of tree combination. Consider again T_1, T_2, T_3 of Figure 1 and T_5 of Figure 4. T_5 is a member of $\{T_1\}\overset{t}{+}(\{T_2\}\overset{t}{+}\{T_3\})$ but not of

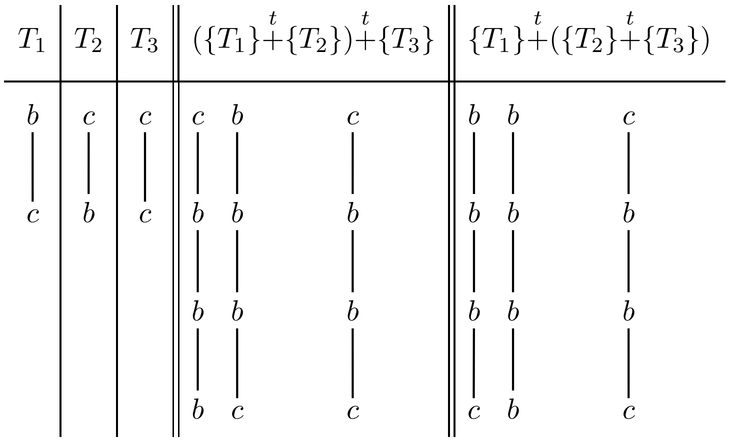
T_1
T_2
T_3
$(\{T_1\}\overset{t}{+}\{T_2\})\overset{t}{+}\{T_3\}$
$\{T_1\}\overset{t}{+}(\{T_2\}\overset{t}{+}\{T_3\})$

Figure 10: Length-3 paths solutions

$(\{T_1\}\overset{t}{+}\{T_2\})\overset{t}{+}\{T_3\}$. The reason is that in T_5, T_1 and T_2 are substructures separated by T_3. When T_2 and T_3 are combined first, T_2 connects to one of the nodes of T_3; then, when T_1 is added, it is connected to another node of T_3. However, when T_1 and T_2 combine first, they must be connected through a common node and cannot be separated as they are in T_5.

Similarly, considering again T_{12}, T_{13}, T_{14} of Figure 8 and their combinations, clearly

$$\{T_{12}\}\overset{t}{+}(\{T_{13}\}\overset{t}{+}\{T_{14}\})\not\cong(\{T_{12}\}\overset{t}{+}\{T_{13}\})\overset{t}{+}\{T_{14}\}$$

When T_{13} and T_{14} are combined, their two single nodes must identify in order to yield a tree. However, when T_{12} and T_{13} combine first, the single node of T_{13} can identify with either of the two nodes of T_{12}. Then, when the resulting tree is combined with T_{14}, the single node of T_{14} can be identified with the other node of T_{12} (the one that was not identified with the node of T_{13}). This is why $(\{T_{12}\}\overset{t}{+}\{T_{13}\})\overset{t}{+}\{T_{14}\}$ overgenerates with respect to $\{T_{12}\}\overset{t}{+}(\{T_{13}\}\overset{t}{+}\{T_{14}\})$.

The above cases exemplify the causes for the non-associativity of tree combination: When two trees are combined, at least two nodes (each from a different tree) must identify. Hence, the two trees must be connected in the resulting tree. However, other combination orders that allow two trees to be separated (by other trees) can yield results which cannot be obtained when the two trees are first combined together.

The solution we propose is inspired by Cohen-Sygal and Wintner (2006) who, in the context of typed unification grammars, move to the powerset domain in order to ensure associativity of grammar combination. Working in the powerset domain, rather than the original entities, enables the operator to 'remember' *all* the possibilities; then, after the combination, an extra stage is added in which the original entities are restored.

In the case of tree combination, the basic units should be forests rather than trees; and *forest combination* must be defined over sets of forests rather than sets of trees. Forest combination is defined in much the same way as above: two forests are combined by identifying some of their nodes. Again, if two nodes are identified then all their ancestors must be identified as well. We allow two forests to combine even if none of their nodes are identified. Furthermore, similarly to tree combination, two different nodes in the same forest represent different entities. Therefore, when two forests are combined, two nodes can be identified only if they belong to the two different forests.

Definition 14. *A* ***forest*** $\langle V, E, R\rangle$ *is a finite set of node-disjoint trees with vertices* V*, edges* E *and roots* R*. If* $\langle V, E, r\rangle$ *is a tree, then* $\langle V, E, \{r\}\rangle$ *is its* **corresponding forest***.*

The meta-variable F ranges over forests and V, E, R over their components. The meta-variable $\mathcal{F}$ ranges over sets of forests. The definition of disjointness is trivially extended to forests and set of forests.

Definition 15. *Two forests* $F_1 = \langle V_1, E_1, R_1\rangle$*,* $F_2 = \langle V_2, E_2, R_2\rangle$ *are* **isomorphic***, denoted* $F_1 {\sim} F_2$*, if there exists a total one to one and onto function* $i : V_1 \to V_2$ *such that for all* $u, v \in V_1$*,* $(u, v) \in E_1$ *iff* $(i(u), i(v)) \in E_2$*; and for all* $u \in V_1$*,* $u \in R_1$ *iff* $i(u) \in R_2$*.*

The definition of isomorphism of sets of trees is extended to sets of forests (using the above definition of forests isomorphism).

Definition 16. *Let* $F_1 = \langle V_1, E_1, R_1\rangle$*,* $F_2 = \langle V_2, E_2, R_2\rangle$ *be two disjoint forests. An equivalence relation '*$\overset{f}{\approx}$*' over* $V_1 \cup V_2$ *is* **legal** *if both:*

1. *for all* $v_1, v_2 \in V_1 \cup V_2$*, if* $v_1 \overset{f}{\approx} v_2$ *and* $v_1 \neq v_2$ *then either* $v_1 \in V_1$ *and* $v_2 \in V_2$ *or* $v_1 \in V_2$ *and* $v_2 \in V_1$*; and*

2. for all $u_1, v_1, u_2, v_2 \in V_1 \cup V_2$, if $v_1 \overset{f}{\approx} v_2$, u_1 is the parent of v_1 and u_2 is the parent of v_2, then $u_1 \overset{f}{\approx} u_2$.

$Eq_f(F_1, F_2)$ is the set of legal equivalence relations over $V_1 \cup V_2$.

Notice that in contrast to definition 4, a legal equivalence relation over forests permits a combination in which no nodes unify. Such a grammar combination amounts to a set union of the two forests.

Definition 17. *Let $F_1 = \langle V_1, E_1, R_1 \rangle$, $F_2 = \langle V_2, E_2, R_2 \rangle$ be two disjoint forests and let '$\overset{f}{\approx}$' be a legal equivalence relation over $V_1 \cup V_2$. The* **forest combination** *of F_1, F_2* **with respect to '$\overset{f}{\approx}$'**, *denoted $F_1 +_{\overset{f}{\approx}} F_2$, is a forest $F = \langle V, E, R \rangle$, where V and E are as in definition 5, and $R = \{[r]_{\overset{f}{\approx}} \mid$ for all $u \in [r]_{\overset{f}{\approx}}, u \in R_1 \cup R_2\}$.*

When two forests are combined, nodes in the same equivalence class are identified. Since the equivalence relation is legal, the resulting structure is indeed a forest.

Definition 18. *Let $\mathcal{F}_1, \mathcal{F}_2$ be two disjoint sets of forests. The* **forest combination** *of $\mathcal{F}_1, \mathcal{F}_2$, denoted $\mathcal{F}_1 \overset{f}{+} \mathcal{F}_2$, is the set of forests*

$$\mathcal{F} = \bigcup_{\substack{F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2 \\ \overset{f}{\approx} \in Eq_f(F_1, F_2)}} F_1 +_{\overset{f}{\approx}} F_2$$

Example 3. *Consider F_1, F_2 of Figure 11. Three members of $\{F_1\} \overset{f}{+} \{F_2\}$, namely F_3, F_4, F_5, are depicted in Figure 12. F_3 is obtained by identifying q_5 and q_6, F_4 is obtained by not identifying any of the nodes and F_5 is the result of identifying q_5 with q_6 and q_1 with q_7. Notice that in F_5, the two separated trees of F_1 are connected through the single tree of F_2. F_6 of Figure 12 is not a member of $\{F_1\} \overset{f}{+} \{F_2\}$ because it identifies q_1 and q_5 which belong to the same forest.*

Example 4. *Consider again T_1, T_2, T_3 of Figure 1 and T_5 of Figure 4. Let F_1, F_2, F_3, F_5 be their corresponding forests, respectively. F of Figure 13 is a member of $\{F_1\} \overset{f}{+} \{F_2\}$ which is obtained by not identifying any of the two forests nodes. F_5 is a member of $\{F\} \overset{f}{+} \{F_3\}$ which is*

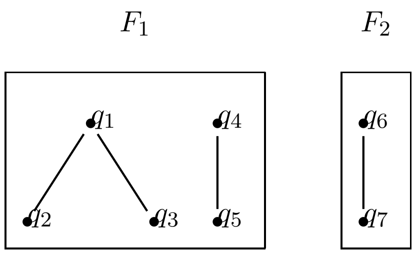

Figure 11: Two forests to be combined

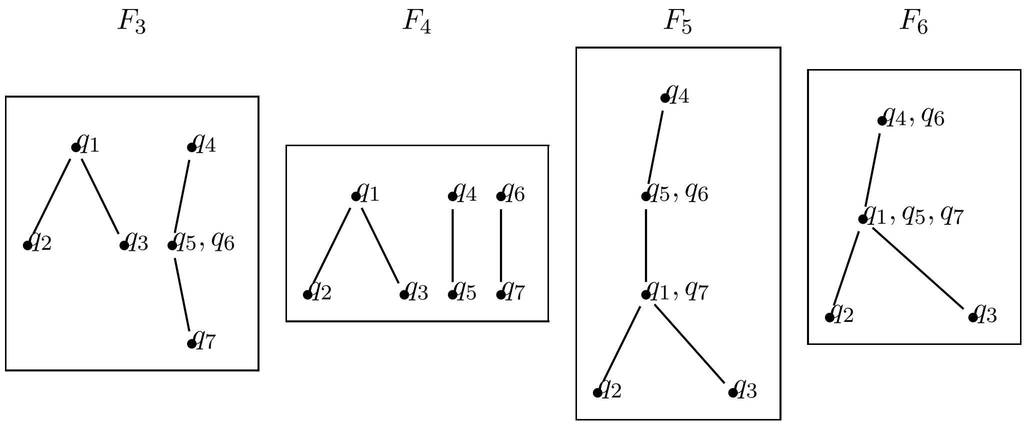

Figure 12: Legal and illegal combinations of F_1, F_2

obtained by identifying the two roots of F with the two leaves of F_3. Hence, F_5 is a member of both $\{F_1\}\overset{f}{+}(\{F_2\}\overset{f}{+}\{F_3\})$ and $(\{F_1\}\overset{f}{+}\{F_2\})\overset{f}{+}\{F_3\}$.

The forest combination operation can be easily extended to the polarized case. This is done in the same way tree combination is extended to polarized tree combination: Polarities are attached to nodes and an extra condition for the identification of two nodes is that their polarities combine; in that case the new node has the polarity which is the product of the two nodes polarities. The complete definitions are given in Appendix A.

Example 5. *Consider again the systems of polarities depicted in Figure 7 and T_{12}, T_{13}, T_{14} of Figure 8. Let F_{12}, F_{13}, F_{14} be their corresponding forests, respectively. The forest combination of*

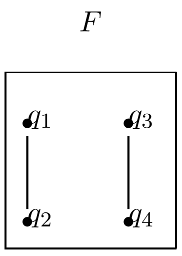

Figure 13: A forest combination of F_1 and F_2

$\{F_{12}\}, \{F_{13}\}, \{F_{14}\}$ *is depicted in Figures 14 (intermediate results) and 15. Here,*

$$(\{F_{12}\}\overset{f}{+}\{F_{13}\})\overset{f}{+}\{F_{14}\}\cong\{F_{12}\}\overset{f}{+}(\{F_{13}\}\overset{f}{+}\{F_{14}\})$$

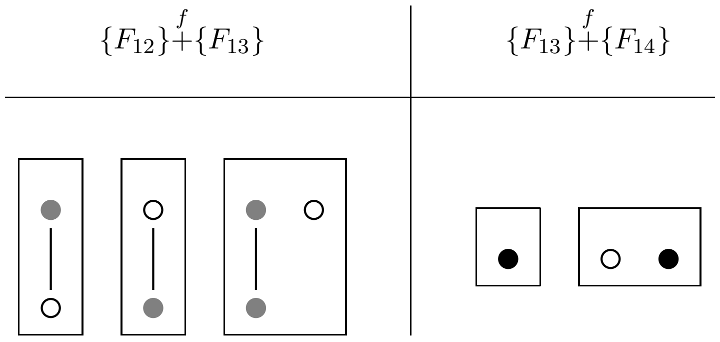
$\{F_{12}\}\overset{f}{+}\{F_{13}\}$
$\{F_{13}\}\overset{f}{+}\{F_{14}\}$

Figure 14: Intermediate results

In order to guarantee the associativity of tree combination we moved from trees to the powerset domain, i.e., to forests. However, our interest is in the trees rather than the forests. Therefore, after all the forests are combined, a resolution stage is required in which only desired solutions are retained. In our case, this is done by eliminating all forests which are not singletons. For example, executing the resolution stage over the forests of Figure 15, retains only the four forests of the upper row.

Theorem 8. *Forest combination is an associative operation: if* $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ *are disjoint sets of forests then* $((\mathcal{F}_1\overset{f}{+}\mathcal{F}_2)\overset{f}{+}\mathcal{F}_3)\cong(\mathcal{F}_1\overset{f}{+}(\mathcal{F}_2\overset{f}{+}\mathcal{F}_3))$. *This holds both for non-polarized and for polarized combination, as long as* $(P, \cdot)$ *is commutative.*

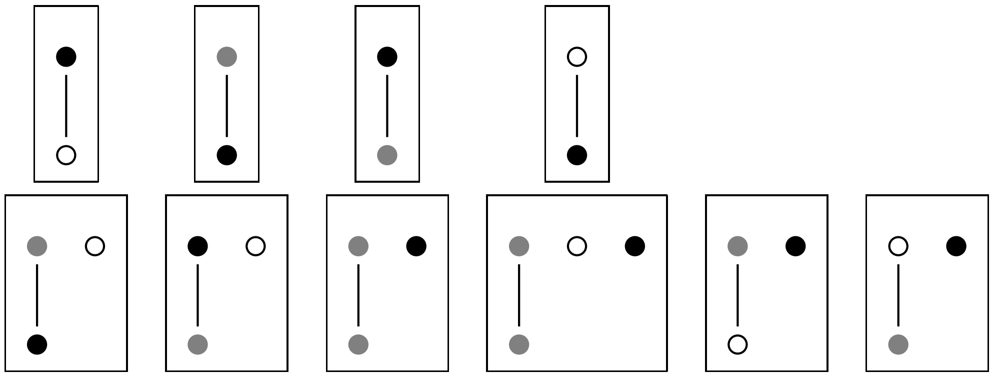

Figure 15: Forest combination of F_{12}, F_{13} and F_{14}: $(\{F_{12}\}\overset{f}{+}\{F_{13}\})\overset{f}{+}\{F_{14}\}\cong\{F_{12}\}\overset{f}{+}(\{F_{13}\}\overset{f}{+}\{F_{14}\})$

The proof is given in Appendix B.

Summing up, we showed how to redefine tree combination in PUG in order to guarantee the associativity of the operation. In this way, the combination operator can be implemented more flexibly, independently of the order of the arguments, which results in more efficient computation. In particular, we showed corresponding (but associative!) computations of all the (non-associative) examples of the previous sections.

5 Forest combination and XMG

The results of the previous section bear relevance to the metagrammar paradigm and specifically to XMG (Duchier, Le Roux, and Parmentier, 2004; Crabbé, 2005). In particular, the forest-based grammar combination operation can be instrumental for defining an alternative semantics for XMG, which we sketch in this section.

XMG provides the grammar writer with a tree-description logic, whose semantics is based on trees. A given formula denotes an infinite set of trees, each satisfying the conditions of the formula. This denotation is restricted by considering only the finite set of *minimal* trees satisfying the description (Duchier and Gardent, 1999; Duchier and Gardent, 2001). Conceptually, computation of the minimal tree models of a given formula consists of two stages: The first computes the (infinite set of) tree models of a formula and the second extracts from these models only the minimal ones.

The following definitions are based on Duchier and Gardent (1999) and Duchier and Gardent (2001).

Definition 19. *A* **formula** ϕ *is an arbitrary conjunction of dominance and labeling constrains*

$$\phi ::= \phi \wedge \phi' \mid x \lhd y \mid x = y \mid x \perp y$$

where x, y are taken from a set of variables.[3]

The semantics is given by interpretation over finite tree structures.

Definition 20. *Let V_ϕ be the set of variables occurring in a formula ϕ. A* **tree solution** *of ϕ is a pair (T, I) where $T = \langle V, E, r\rangle$ is a finite tree (a* **tree model***) and $I : V_\phi \mapsto V$ is a function (an* **interpretation***) that maps each variable in ϕ to a node in T. $x \lhd y$ means that, in the solution tree T , $I(x)$ must dominate $I(y)$; $x = y$ means that $I(x) = I(y)$; and $x \perp y$ means that $I(x) \neq I(y)$. The* **denotation** *of a formula ϕ, denoted $S_{xmg}(\phi)$ is the set of its tree solutions $\{(T, I) \mid (T, I)$ is a tree solution of $\phi\}$.*

If T is a tree model of ϕ, then every tree T' which contains T as a subtree is also a tree model of ϕ. Therefore, there are infinitely many tree models of any formula ϕ. To restrict the infinite set to desired trees, *minimal* (finite) models are considered. Any formula ϕ has only finitely many minimal tree models (up to isomorphism).

Definition 21. *A tree model T is a* **minimal tree model**[4] *of ϕ if all nodes in T interpret at least one variable in ϕ. Then $extract(S_{xmg}(\phi)) = \{T \mid (T, I) \in S_{xmg}(\phi)$ and T is a minimal tree model of $\phi\}$.*

We propose an alternative semantics, denoted S_{fc}, for tree descriptions, based on the forest combination operation of section 4. In S_{fc} a formula denotes the set of minimal forests satisfying

[3]Duchier and Gardent (1999) and Duchier and Gardent (2001) define several more operators (e.g., precedence and labeling). For the sake of simplicity we restrict ourselves to the list of operators presented in this definition, but all the results can be extended to the full list of operators.

[4]In Duchier and Gardent (1999) and Duchier and Gardent (2001), the definition of minimal tree models is based on the notion of D-trees (Rambow, Vijay-Shanker, and Weir, 1995). For the sake of simplicity we do not use this notion, but all the results can be easily extended to D-trees.

it. Forest combination operates directly on minimal forests in a way that corresponds to formula conjunction in the syntactic level: The denotation of a conjunction of formulas is the combination of the denotations of the conjuncts. Here, also, a resolution stage is required, to retain only forests which are singletons (i.e., trees).

Definition 22. *Let V_ϕ be the set of variables occurring in a formula ϕ. A* **forest solution** *of ϕ is a pair (F, I) where $F = \langle V, E, R\rangle$ is a finite minimal forest (a* **forest model***) and $I : V_\phi \mapsto V$ is an onto function (an* **interpretation***) that maps each variable in ϕ to a node in F such that all nodes in F interpret at least one variable in ϕ. $x \lhd y$ means that, in the solution forest F , $I(x)$ must dominate $I(y)$; $x = y$ means that $I(x) = I(y)$; and $x \bot y$ means that $I(x) \neq I(y)$. The* **denotation** *of a formula ϕ, denoted $S_{fc}(\phi)$, is the set of its forest solutions $\{(F, I) \mid (F, I)$ is a forest solution of $\phi\}$. Define $resolve(S_{fc}(\phi)) = \{F \mid (F, I) \in S_{fc}(\phi)$ and F is a singleton$\}$.*

Observe that in this semantics a formula can denote only finitely many forests (up to isomorphism). The two semantics, S_{xmg} and S_{fc}, coincide.

Theorem 9. *$extract(S_{xmg}(\phi)) = resolve(S_{fc}(\phi))$*

Proof. Assume $T \in extract(S_{xmg}(\phi))$. Then, there exists an interpretation I from the variables of ϕ to the nodes of T such that (T, I) is a tree solution of ϕ and T is a minimal tree model. Any tree is also a forest (a singleton) and therefore, T is also a forest model of ϕ. Hence, $(T, I) \in S_{fc}(\phi)$. Since T is a tree, it follows that $(T, I) \in resolve(S_{fc}(\phi))$.

Now assume that $F \in resolve(S_{fc}(\phi))$. Then, there exists an interpretation I from the variables of ϕ to the nodes of F such that (F, I) is a forest solution of ϕ and F is a tree. Since F is a tree, $(F, I) \in S_{xmg}(\phi)$. (F, I) is a forest solution of ϕ and therefore F is a minimal model. Hence, $(F, I) \in extract(S_{xmg}(\phi))$. □

Since the two semantics coincide, either one of them can be used in an implementation of XMG. Specifically, in the existing implementation of XMG the grammar designer is presented with finite trees only, and the infinite tree models are never explicit. S_{fc} offers the opportunity to use finite trees as the bona fide denotation of tree descriptions. This, however, comes with a cost: the number of tree fragments can grow very fast, and a sophisticated cashing mechanism will be necessary in any practical implementation.

Both approaches require a resolution stage; the resolution stage in the forest combination approach seems to be simpler, requiring only the extraction of singletons from a set. However, it could also be less efficient, due to the growth in the number of trees and the fact that resolution is deferred to the end of the computation.

To sum up, the forest combination semantics provides the grammar writer with a formally defined operation executed directly on the minimal models amounting to the conjunction operation in the syntactic level of tree descriptions. Whether or not it can be practically beneficial remains to be seen.

6 Conclusion

We have shown how the tree combination operation in PUG can be redefined to guarantee associativity, thus facilitating the use of this powerful and flexible formalism for grammar engineering and modular grammar development. The key to the solution is a *powerset-lift* of the domain and the corresponding operation: Rather than working with trees, manipulating forests provides means to ‘remember’ *all* the possible combinations of grammar fragments. Then, after all fragments are combined, a *resolution* stage is added to produce the desired results.

This method was used to guarantee associativity in a different domain, namely signature combination in the context of typed unification grammars (Cohen-Sygal and Wintner, 2006). We believe that this method is sufficiently general to be applicable to a variety of formalisms. In particular, it is applicable to the general case of PUG where arbitrary objects and structures are manipulated. In this case also, the move to the powerset domain by manipulating sets of objects, rather that the objects themselves, enforces associativity.

While the results presented in this paper are theoretical, they constitute the foundation for correctly implementing tree-based grammar combination operators in existing formalisms. The actual integration of these results in a grammar development environment is delegated to future work.

Acknowledgments

This research was supported by The Israel Science Foundation (grant no. 136/01). We are grateful to Yannick Parmentier for his help and support, including very useful comments on earlier versions of this paper. We with to thank Claire Gardent for giving us the opportunity to present some of these results at the XMG Workshop in LORIA Nancy in 2007. We also benefited from constructive comments by three anonymous referees. All remaining errors and misconceptions are, of course, our own.

Appendix

A Polarized forest combination

We extend the forest combination operation to the polarized case. This is done in the same way tree combination is extended to polarized tree combination: Polarities are attached to nodes and an extra condition for the identification of two nodes is that their polarities combine; in that case the new node has the polarity which is the product of the two nodes polarities. For the following discussion we assume that a system of polarities $(P, \cdot)$ is given.

Definition 23. *A* **polarized forest** $\langle V, E, R, p\rangle$ *is a forest in which each node is associated with a polarity through a total function* $p : V \to P$*. If* $\langle V, E, R, p\rangle$ *is a polarized forest then* $\langle V, E, R\rangle$ *is its* **underlying forest***.*

Definition 24. *Two polarized forests are* **disjoint** *if their underlying forests are disjoint.*

Definition 25. *Two polarized forest* $F_1 = \langle V_1, E_1, R_1, p_1\rangle$*,* $F_2 = \langle V_2, E_2, R_2, p_2\rangle$ *are* **isomorphic***, denoted* $F_1 {\sim} F_2$*, if their underlying forests are isomorphic and, additionally, for all* $v \in V_1$*,* $p_1(v) = p_2(i(v))$*.*

The definition of isomorphism of sets of forests is trivially extended to sets of polarized forests.

Definition 26. *Let $F_1 = \langle V_1, E_1, R_1, p_1 \rangle$, $F_2 = \langle V_2, E_2, R_2, p_2 \rangle$ be two disjoint polarized forests. An equivalence relation '$\overset{f}{\approx}$' over $V_1 \cup V_2$ is* **legal** *if it is legal over the underlying forests of F_1 and F_2 and, additionally, for all $v_1 \in V_1$ and $v_2 \in V_2$, if $v_1 \overset{t}{\approx} v_2$, then $p_1(v_1) \cdot p_2(v_2)\downarrow$.*

$Eq_f(F_1, F_2)$ is the set of legal equivalence relations over $V_1 \cup V_2$.

Definition 27. *Let $F_1 = \langle V_1, E_1, R_1, p_1 \rangle$, $F_2 = \langle V_2, E_2, R_2, p_2 \rangle$ be two disjoint polarized forests and let '$\overset{f}{\approx}$' be a legal equivalence relation over $V_1 \cup V_2$. The* **polarized forest combination** *of F_1, F_2* **with respect to '$\overset{f}{\approx}$'**, *denoted $F_1 +_{\overset{f}{\approx}} F_2$ is a forest $F = \langle V, E, R, p \rangle$ where:*

- *V, E and R are as in definition 17*
- *for all $[v]_{\overset{f}{\approx}} \in V$,* $p([v]_{\overset{f}{\approx}}) = \begin{cases} (p_1 \cup p_2)(v) & \textit{if } [v]_{\overset{f}{\approx}} = \{v\} \\ (p_1 \cup p_2)(v) \cdot (p_1 \cup p_2)(u) & \textit{if } [v]_{\overset{f}{\approx}} = \{v, u\} \textit{ and} \\ & u \neq v \end{cases}$

The definition of forest combination of sets of forests is trivially extended to sets of polarized forests.

B (Polarized) forest combination is associative

We now show that forest combination (both with and without polarities) is an associative operation. We begin by proving the associativity of the non-polarized case. To do so, we need to show that if $F \in ((\mathcal{F}_1 \overset{f}{+} \mathcal{F}_2) \overset{f}{+} \mathcal{F}_3)$ then $(\mathcal{F}_1 \overset{f}{+} (\mathcal{F}_2 \overset{f}{+} \mathcal{F}_3))$ includes an isomorphic forest of F. The isomorphism is required in order to ignore the irrelevant names of nodes. To be able to refer to any isomorphic tree of the combination result, we define *mutual combination*: If F_3 is a forest combination of F_1 and F_2 with respect to some legal equivalence relation then both F_1 and F_2 are substructures of F_3, and furthermore, F_3 contains no redundant information: Every arc and node in F_3 belongs to either of the substructures that are induced by F_1 and F_2. This property is common for all the isomorphic trees of F_3. Moreover, F_1 and F_2 induce in all these isomorphic trees the exact same substructures. F_3 and all its isomorphic trees are mutual combinations of F_1 and F_2.

Definition 28. *Let F_1, F_2, F_3 be disjoint forests. F_3 is a* **mutual combination** *of F_1 and F_2, denoted $F_1 \oplus F_2 \mapsto F_3$, if there exists a total function $f : V_1 \cup V_2 \to V_3$ (***a combination function***) such that all the following hold:*

- *f is onto*
- *for all $u, v \in V_1 \cup V_2$, if u is the parent of v (in either F_1 or F_2) then $f(u)$ is the parent of $f(v)$ in F_3*
- *for all $u, v \in V_3$, if u is the parent of v in F_3 then there exist $u', v' \in V_1 \cup V_2$ such that u' is the parent of v' (in either F_1 or F_2), $f(u') = u$ and $f(v') = v$*
- *for all $u, v \in V_1 \cup V_2$, if $f(u) = f(v)$ and $u \neq v$ then either $u \in V_1$ and $v \in V_2$ or $u \in V_2$ and $v \in V_1$*

The second condition guarantees that F_1 and F_2 are substructures of F_3. The first and third conditions guarantee that F_3 contains no redundant information. The last condition guarantees that two different nodes in the same forest (representing different entities) correspond to different nodes in F_3. Lemma 10 and theorem 11 show that indeed mutual combination corresponds to forest combination.

Lemma 10. *If F_1, F_2, F_3, F_4 are disjoint forests such that $F_1 \oplus F_2 \mapsto F_3$ and $F_3 \sim F_4$, then $F_1 \oplus F_2 \mapsto F_4$.*

Proof. Let F_1, F_2, F_3, F_4 be disjoint forests such that $F_1 \oplus F_2 \mapsto F_3$ and $F_3 \sim F_4$. Then there exist a combination function $f : V_1 \cup V_2 \to V_3$ and an isomorphism $i : V_3 \to V_4$. Define $h : V_1 \cup V_2 \to V_4$ where for all $v \in V_1 \cup V_2$, $h(v) = i(f(v))$. h is a combination function (the actual proof is suppressed) and hence, $F_1 \oplus F_2 \mapsto F_4$. $\square$

Theorem 11. *Let F_1, F_2, F_3 be disjoint forests. The following two conditions are equivalent:*

- $F_1 \oplus F_2 \mapsto F_3$

- *there exist a forest F_4 and a legal equivalence relation $\overset{f}{\approx} \in Eq_f(F_1, F_2)$ such that $F_4 = F_1 +_{\overset{f}{\approx}} F_2$ and $F_3 {\sim} F_4$*

Proof. Let F_1, F_2, F_3 be disjoint forests and assume that there exist a forest F_4 and a legal equivalence relation $\overset{f}{\approx} \in Eq_f(F_1, F_2)$ such that $F_4 = F_1 +_{\overset{f}{\approx}} F_2$ and $F_3 {\sim} F_4$. Observe that $V_4 = \{[v]_{\overset{f}{\approx}} | v \in v_1 \cup V_2\}$ and $E_4 = \{([u]_{\overset{f}{\approx}}, [v]_{\overset{f}{\approx}}) | (u, v) \in E_1 \cup E_2\}$. Define $h : V_1 \cup V_2 \to V_4$ where for all $v \in V_1 \cup V_2$, $h(v) = [v]_{\overset{f}{\approx}}$. h is a combination function and hence, $F_1 \oplus F_2 \mapsto F_4$. Since $F_4 {\sim} F_3$ and by lemma 10, $F_1 \oplus F_2 \mapsto F_3$.

Let F_1, F_2, F_3 be disjoint forests and assume that $F_1 \oplus F_2 \mapsto F_3$. Therefore, there exists a combination function $f : V_1 \cup V_2 \to V_3$. Define a relation '$\approx$' over $V_1 \cup V_2$ where for all $u, v \in V_1 \cup V_2$, $u \approx v$ iff $f(u) = f(v)$. Clearly, '$\approx$' is an equivalence relation. Furthermore, '$\approx$' is legal. Now, define $F_4 = F_1 +_{\approx} F_2$ and define $i : V_4 \to V_3$ where for all $[v]_{\approx} \in V_4$, $i([v]_{\approx}) = f(v)$. Notice that i is well defined because for all u, v such that $u \approx v$, $f(u) = f(v)$. i is an isomorphism of F_3 and F_4. □

Notice that since forest isomorphism is reflexive and by theorem 11, if $F_1 +_{\overset{f}{\approx}} F_2 = F_3$ then $F_1 \oplus F_2 \mapsto F_3$.

Theorem 12. *Forest combination is an associative operation: if $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ are disjoint sets of forests then $((\mathcal{F}_1 \overset{f}{+} \mathcal{F}_2) \overset{f}{+} \mathcal{F}_3) \cong (\mathcal{F}_1 \overset{f}{+} (\mathcal{F}_2 \overset{f}{+} \mathcal{F}_3))$*

Proof. Let $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ be disjoint sets of forests and assume that $F = \langle V, E, R \rangle \in (\mathcal{F}_1 \overset{f}{+} \mathcal{F}_2) \overset{f}{+} \mathcal{F}_3$. Then there exist $F' \in \mathcal{F}_1 \overset{f}{+} \mathcal{F}_2$, $F_3 \in \mathcal{F}_3$ and $\approx_1 \in Eq_f(F', F_3)$ such that $F' +_{\approx_1} F_3 = F$. Therefore by theorem 11, $F' \oplus F_3 \mapsto F$, and hence, there exists a combination function $f_1 : V' \cup V_3 \to V$. $F' \in \mathcal{F}_1 \overset{f}{+} \mathcal{F}_2$ and therefore there exist $F_1 \in \mathcal{F}_1$, $F_2 \in \mathcal{F}_2$ and $\approx_2 \in Eq_f(F_1, F_2)$ such that $F_1 +_{\approx_2} F_2 = F'$. Therefore by theorem 11, $F_1 \oplus F_2 \mapsto F'$ and hence, there exists a combination function $f_2 : V_1 \cup V_2 \to V'$. Define $f : V_1 \cup V_2 \cup V_3 \to V$ where:

$$f(v) = \begin{cases} f_1(v) & v \in V_3 \\ f_1(f_2(v)) & v \in V_1 \cup V_2 \end{cases}$$

Let F_4 be a graph defined by the restriction of f to $V_2 \cup V_3$, where:

- $V_4 = \{f(v) \mid v \in V_2 \cup V_3\}$
- $E_4 = \{(f(u), f(v)) \mid (u, v) \in E_2 \cup E_3\}$
- $R_4 = \{f(r) \mid r \in R_2 \cup R_3$ and for all $v \in V_2 \cup V_3$ such that $f(v) = f(r)$, $v \in R_2 \cup R_3\}$

F_4 is a forest and $f_{|V_2 \cup V_3}$ (the restriction of f to $V_2 \cup V_3$) is a combination function of F_2 and F_3 to F_4 (the actual proof is suppressed). Hence, $F_2 \oplus F_3 \mapsto F_4$ and therefore by theorem 11, there exist a forest F_5 and a legal equivalence relation $\approx_3 \in Eq_f(F_2, F_3)$ such that $F_5 = F_2 +_{\approx_3} F_3$ and $F_5 \sim F_4$. Hence, $F_5 \in \mathcal{F}_2 \overset{f}{+} \mathcal{F}_3$. Let $i : V_5 \to V_4$ be an isomorphism of F_5 and F_4. Define $h : V_5 \cup V_1 \to V$ where:

$$h(v) = \begin{cases} f(v) & v \in V_1 \\ i(v) & v \in V_5 \end{cases}$$

h is a combination function of F_1 and F_5 to F. Hence, $F_1 \oplus F_5 \mapsto F$, and therefore by theorem 11, there exists a forest F'' and a legal equivalence relation $\approx_4 \in Eq_f(F_1, F_5)$ such that $F'' = F_1 +_{\approx_4} F_5$ and $F'' \sim F'$. Hence, $F'' \in \mathcal{F}_1 \overset{f}{+} (\mathcal{F}_2 \overset{f}{+} \mathcal{F}_3)$ and $F'' \sim F'$.

The proof that if $F \in \mathcal{F}_1 \overset{f}{+} (\mathcal{F}_2 \overset{f}{+} \mathcal{F}_3)$ then there exists $F' \in (\mathcal{F}_1 \overset{f}{+} \mathcal{F}_2) \overset{f}{+} \mathcal{F}_3$ such that $F \sim F'$ is symmetric. $\square$

We now prove the associativity of polarized forest combination. The proof idea is similar to the proof of the non-polarized case. For the following discussion, assume that a system of polarities $(P, \cdot)$ has been specified.

Definition 29. *Let F_1, F_2, F_3 be disjoint polarized forests. F_3 is a* **mutual combination** *of F_1 and F_2, denoted $F_1 \oplus F_2 \mapsto F_3$, if there exists a total function $f : V_1 \cup V_2 \to V_3$ (***a combination function***) such that f is a combination function of the underlying forests, and, additionally, for all $v_1, v_2 \in V_1 \cup V_2$, if $f(v_1) = f(v_2)$ and $v_1 \neq v_2$ then $(p_1 \cup p_2)(v_1) \cdot (p_1 \cup p_2)(v_2){\downarrow}$ and $(p_1 \cup p_2)(v_1) \cdot (p_1 \cup p_2)(v_2) = p_3(f(v_1))$.*

Lemma 13. *If F_1, F_2, F_3, F_4 are disjoint polarized forests such that $F_1 \oplus F_2 \mapsto F_3$ and $F_3 \sim F_4$, then $F_1 \oplus F_2 \mapsto F_4$.*

Proof. Similar to the proof of lemma 10. □

Theorem 14. *Let F_1, F_2, F_3 be disjoint polarized forests. The following two conditions are equivalent:*

- *there exist a forest F_4 and a legal equivalence relation $\overset{f}{\approx} \in Eq_f(F_1, F_2)$ such that $F_4 = F_1 +_{\overset{f}{\approx}} F_2$ and $F_3 {\sim} F_4$*
- *$F_1 \oplus F_2 \mapsto F_3$*

Proof. Similar to the proof of theorem 11. □

Theorem 15. *Let $(P, \cdot)$ be a system of polarities. Then polarized forest combination based on $(P, \cdot)$ is an associative operation: if $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ are disjoint sets of forests then*

$$((\mathcal{F}_1 \overset{f}{+} \mathcal{F}_2) \overset{f}{+} \mathcal{F}_3) \cong (\mathcal{F}_1 \overset{f}{+} (\mathcal{F}_2 \overset{f}{+} \mathcal{F}_3))$$

Proof. Similar to the proof of theorem 12. □

References

Abeillé, Anne, Marie-Hélène Candito, and Alexandra Kinyon. 2000. FTAG: developing and maintaining a wide-coverage grammar for French. In Erhard Hinrichs, Detmar Meurers, and Shuly Wintner, editors, *Proceedings of the ESSLLI-2000 Workshop on Linguistic Theory and Grammar Implementation*, pages 21–32.

Bonfante, Guillaume, Bruno Guillaume, and Guy Perrier. 2004. Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation. In *Proceedings of the 20th international conference on Computational Linguistics (COLING 04)*, pages 303–309.

Candito, Marie-Hélène. 1996. A principle-based hierarchical representation of LTAGs. In *Proceedings of the 16th conference on Computational linguistics (COLING 1996)*, pages 194–199, Copenhagen, Denemark.

Cohen-Sygal, Yael and Shuly Wintner. 2006. Partially specified signatures: A vehicle for grammar modularity. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*, pages 145–152, Sydney, Australia, July.

Crabbé, Benoit. 2005. Grammatical development with XMG. In *Proceedings of the 5th International Conference on Logical Aspects of Computational Linguistics (LACL)*, Bordeaux, France, April.

Crabbé, Benoit and Denys Duchier. 2004. Metagrammar redux. In *Proceedings of the International Workshop on Constraint Solving and Language Processing (CSLP)*, Copenhagen, Denemark.

Duchier, Denys and Claire Gardent. 1999. A constraint-based treatment of descriptions. In *Third International Workshop on Computational Semantics (IWCS-3).*

Duchier, Denys and Claire Gardent. 2001. Tree descriptions, constraints and incrementality. In Harry Bunt, Reinhard Muskens, and Elias Thijsse, editors, *Computing Meaning, Volume 2*, volume 77 of *Studies In Linguistics And Philosophy.* Kluwer Academic Publishers, Dordrecht, December, pages 205–227.

Duchier, Denys, Joseph Le Roux, and Yannick Parmentier. 2004. The metagrammar compiler: An NLP application with a multi-paradigm architecture. In *Proceedings of the Second International Mozart/Oz Conference (MOZ 2004)*, Charleroi, Belgium, October.

Joshi, Aravind K., Leon S. Levy, and Masako Takahashi. 1975. Tree Adjunct Grammars. *Journal of Computer and System Sciences.*

Kahane, Sylvain. 2006. Polarized unification grammars. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*, pages 137–144, Sydney, Australia, July.

Kahane, Sylvain and Francois Lareau. 2005. Meaning-text unification grammar: modularity and polarization. In *Proceedings of the 2nd International Conference on Meaning-Text Theory*, pages 197–206, Moscow.

Kallmeyer, Laura. 2001. Local tree description grammars. *Grammars*, 4(2):85–137.

Parmentier, Yannick, Laura Kallmeyer, Timm Lichte, and Wolfgang Maier. 2007. Xmg: extending metagrammars to mctag. In *Actes de l'atelier sur les formalismes syntaxiques de haut niveau, Conf?rence sur le Traitement Automatique des Langues Naturelles, TALN 2007*, Toulouse, France, June.

Perrier, Guy. 2000. Interaction grammars. In *Proceedings of the 18th conference on Computational linguistics (COLING 2000)*, pages 600–606.

Rambow, Owen, K. Vijay-Shanker, and David Weir. 1995. D-tree grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 151–158.

Vijay-Shanker, K. 1992. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18(4):481–517.

XTAG Research Group. 2001. A lexicalized tree adjoining grammar for English. Technical Report IRCS-01-03, IRCS, University of Pennsylvania.