*Constituent Structure Sets**

HIROYUKI UCHIDA and DIRK BURY

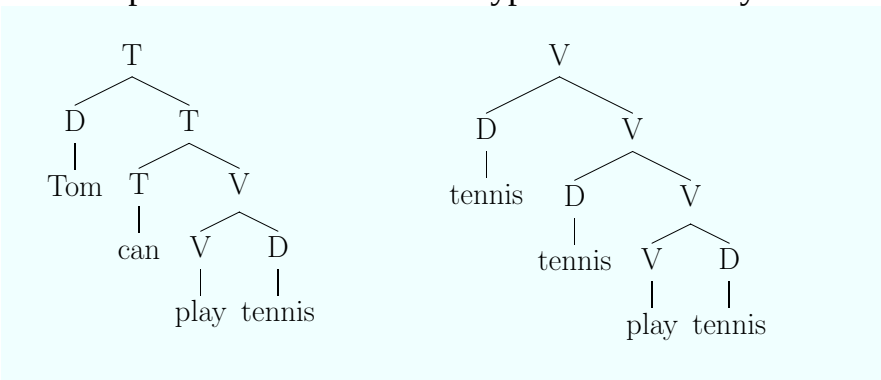
Abstract

We replace traditional phrase structure tree representations by a new type of set-based representations. In comparison to labelled trees in which we can potentially copy one category onto an infinite number of distinct nodes, the proposed representation system significantly restricts syntactic copying. Thus, we do not need to filter out copying possibilities by way of additional constraints. Each structure represents a set of constituents with structurally distinguished items. Compositional semantics is represented by using typed lambda expressions and PF linearization generates binary bracketed strings of phonological items with enough flexibility with regard to word order.

1 Introduction

This paper presents a system of set-based representations of constituent structures that takes over the purely structural tasks of Phrase Structure (PS) trees. If we use labelled trees to represent syntactic structures, we can potentially represent an infinite number of copying of one item. For example, in the bare phrase structure in (1a), the category V which is associated with the lexical item *play* is spread into two nodes, and similarly, the category T has three copies.[1]

(1) a. Bare phrase structure b. Hypothetical infinity



*Many thanks to Marcus Kracht, Neil Smith, Nicholas Allott and Robert Truswell for comments and discussions.

[1] For convenience, we use the word 'copying' to describe both the repetition of the same item into the mother node, as in (1a), and copying to a terminal node, as we can see with *D* in (1b).

In practice, different theories restrict the number of syntactic copying by additional means, and thus, the potential infinity of copying as shown in (1b) does not arise in the actual use of labelled trees in syntactic theories. For example, the copying of V may be restricted by the number of theta roles that the verb *play* assigns (i.e., two) and copying of D to a terminal node can be limited by the requirement of the existence of some functional head whose specifier position D can be copied into. However, our main goal is to investigate a structure representation system which simply cannot express an infinite number of syntactic copying. If the representational system cannot generate a structure of an infinite size (relative to a finite enumeration set), then we do not need to filter out potential infinity possibilities by way of some post-hoc constraints.

Related to this point, we investigate a syntactic system which does not have a rule that is dependent on the identity of the categories (or the items that are put into some structural configurations). In other words, given three categories A, B, C, our narrow syntax puts them into some structural configurations without considering whether A is V (i.e., the category for verbs) or B is D (i.e., the category for determiner phrases such as *Tom* and *that boy*). Our narrow syntax simply checks whether the configurations of the categories are structurally well-formed according to some restrictions definable without mentioning particular category names. Whether verbs and nouns are in the right configuration or whether selection requirements of some categories are satisfied appropriately will then be checked when we interpret the syntactic structures in the semantics, not within the narrow syntactic representations. Thus, our proposed system requires an essential use of the semantic interpretation rules external to the syntax. We also assume that phonological structures that can be generated by the narrow syntax rules together with some PF linearization rules will be evaluated by some phonological constraints that are external to the narrow syntax. In this way, we can concentrate on purely structural factors that narrow down possible natural language expressions abstracted away from interpretational factors.

Section 2 presents a relational structure defined with the set of lexically provided categories as the domain. This system cannot express copying of one item at all and because of this, it cannot represent the desirable sort of asymmetry in the syntax, unless we multiply one lexical category/item into distinct items at the level of lexical enumeration, so that those multiplied items count as distinct items in the relational structure. Though this rather stipulative specification works in application, we prefer to have the syntax tell us how many times each item can be copied via some general calculation rule, rather than stipulating the number of copies of one item at the level of lexical enumeration. Other than such comparison reasons, the definition of the reflexive dominance relation *R* as part of the relational structure in Section 2 facilitates the presentation of the reflexive containment relation that we define in our structure representation system in Section 3.

Section 3 introduces our structure representation system and shows how the system limits the maximal number of copies of one item via some simple calculation. This section also indicates that the German V2 data can be neatly explained by our structure representation system. Section 4 compares the proposed system with labelled tree representations in terms of their expressive powers. Section 5 shows that given the assumption that the enumeration set for each structure is finite and given that the total number of lexical items in the target language is finite, our system generates only an enumerably infinite set of well-formed structures, as desired. Section 6 and 7 provide some provisional sketch of PF linearization and LF interpretation rules respectively. Section 6 provides concluding remarks.

2 Relational Structures

This section presents a relational structure which defines its basic binary relation directly over lexically assigned category names. This system cannot represent either copying or projection of categories/items that are provided by lexical numeration. We argue that this extreme restrictiveness is not desirable in linguistic application. Thus, in the next section, we move on to another representation system in which we define the basic syntactic relation between syntactic constituents rather than categories/lexical items. However, some of the ideas in the relational structure presented in this section are maintained in our representation system. Also, the presentation of relational structures is less complex than the exposition of our representation systems and because of that, some fundamental ideas that we adopt can be explained more clearly with relational structures.

In this system, each structure *S* is a pair as in (2), where *Cat* is a set of categories and *R* is a binary relation between categories. Each *S* has a minimal element, as in (2b). The membership of *Cat* is fixed for each *S*. For the weak representational power of this structure representation system to lead to the restrictiveness of the syntactic theory that uses this system, it is important to have some external evidence to limit the number of items in *Cat*. Thus, we limit the number of items in *Cat* in each *S* by way of the number of overt language expressions appearing in the phonological string to be generated by the structure.[2] We often use category names to represent those phonological words for the sake of generality, but because there is one to one mapping between the number of items in *Cat* and the number of overt phonological words, it does not matter whether we represent each item by a category name such as V or a phonological word such as *smoke*.

[2] As we see in Section 3, we add the functional category T if some sign of its presence is recoverable in the phonological string, such as the presence of a tense suffix.

(2) a. Structure, $S := \langle Cat, R\rangle$, where $R \subseteq Cat \times Cat$
b. Minimal element, $\exists b \in Cat. \forall a \in Cat. Rba$

R is reflexive, transitive and antisymmetric. Each structure is upward non-branching.

(3) a. Reflexitivity: $\forall a \in Cat. Raa$
b. Transitivity: $\forall a, b, c \in Cat. [(Rab \,\&\, Rbc) \rightarrow Rac]$
c. Antisymmetry: $\forall a, b \in Cat. [(Rab \,\&\, Rba) \rightarrow (a=b)]$
d. Upward non-branching:
$\forall a, b, b' \in Cat. [(Rba \& Rb'a) \rightarrow (Rbb' \vee Rb'b)]\$$
e. Max binary branching:
$\forall a,b,c \in Cat. (\{a' \in Cat | Ra'a \& a' \neq a\} = \{b' \in Cat | Rb'b \& b' \neq b\}$
$= \{c' \in Cat | Rc'c \& c' \neq c\}) \rightarrow ((a=b) \vee (a=c) \vee (b=c)))$

(4) Closure (satisfied by (3a)):
$(\forall a \in Cat. \exists b \in Cat. Rab) \,\&\, (\forall b \in Cat. \exists a \in Cat. Rab)$

R corresponds to the reflexive dominance relation (RD) in syntactic trees. Crucially, however, the relation structure presented here defines R as a relation between category names, without using an additional notion of "tree nodes." As we have indicated above, each member of *Cat* corresponds to an overt phonological word in the PF string to be generated (which may include functional items, for example, for T, as long as there are phonological words that they may host in the string). Reflexivity in (3a) satisfies Closure in (4). Note that we could not close *Cat* in this way if R were irreflexive. Thus, immediate dominance (ID), which is inherently irreflexive, is not useful as the basic relation in this representation system.[3] Each set *Cat* is finite and has discrete members. Thus, each structure is finite.

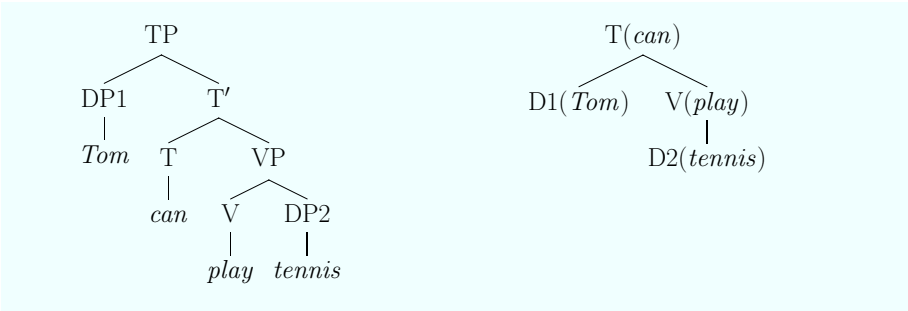
Note that the above restriction on the relational structure does not say anything about the identity of the minimal element. All the structural constraints are definable abstracted away from the identities of the category names. Thus, the restriction such that T for tense must asymmetrically dominates V for a verb is assigned at the level of semantic interpretations. The narrow syntax rules are made

[3] ID is a special case of RD and can be derived from RD without a disjunctive condition. Also, ID cannot always be maintained via every P-morphic mapping between structures, whereas RD can be, cf. Kurtonina (1994:32). These considerations suggest that RD is more basic in relational structures, though ID might be more basic in a derivational grammar presentation, cf. Cornell (1998). For reflexive transitive closure in terms of immediate dominance relation, see Kepser (2006).

out of only those rules which are definable without specifying the names of particular categories/lexical items. This property is preserved in our representation system in Section 3.

Relational structures as defined in (2)-(3) are free of categorial projection. Compare the system with Brody's projection free ``telescope trees" (cf. Brody (2000) for *Tom can play tennis*.

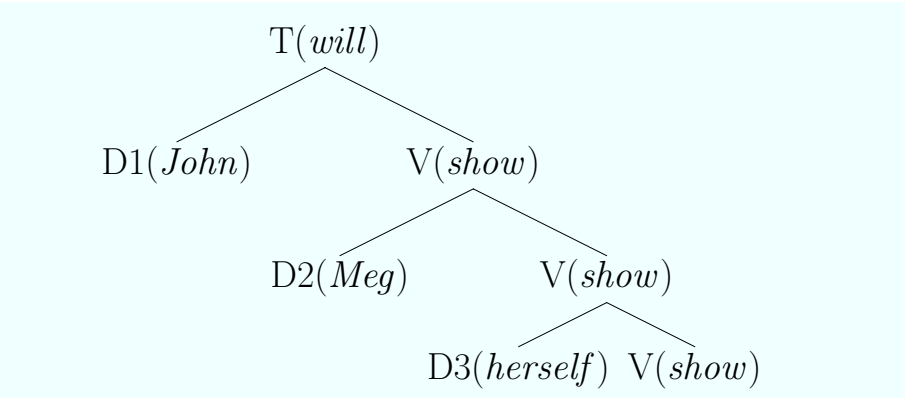
(5) a. b.



c. *Cat* = {*T, V, D1, D2*} (= {*can, play, tom, tennis*})
R={<*T,T*>, <*T,V*>, <*T,D1*>, <*T,D2*>, <*V,V*>, <*V,D2*>,<*D1,D1*>,<*D2,D2*>}
Minimal element (m)} = *T*

The telescope tree in (5b) reduces the two projection lines in the standard PS tree in (5a) to two single nodes, i.e., (i) TP-T′-T to T and (ii) VP-V to V. In the relational structure presented in this section, if we stipulate the ordering among the categories as in 1. T 2. V 3. D1 4. D2 (we later attribute this ordering requirement to the semantics), the structural representation is (5c), which is equivalent to (5b). *R* relates only the categories that are already in *Cat*. Thus we cannot project a member of *Cat* onto a new category (e.g. T projected onto T′ in (5a)), because T′ is not in *Cat*. We also cannot create a new copy of a member of *Cat*, because we cannot distinguish two copies in *R*. Thus, the relation structure defined as above is equivalent to `projection-free' telescope trees as in Brody (2000), though unlike Brody's theory that uses labelled trees for representing syntactic structures and therefore could copy items if necessary (see below for a hypothetical possibility), the relational structure presented here simply cannot represent any copying of one category/item. Thus, it follows as a necessary consequence that this representation system embodies a version of Chomskyan Lexical Inclusiveness. However, the system is too restrictive for linguistic application. Compare (6) with (7).

(6)



(7) *Cat* = {*T, V, D1, D2, D3*}

R ={<*T, T*>, <*T, V*>, <*T, D1*>, <*T, D2*>, <*T, D3*>, <*V, V*>, <*V, D1*>, < *V, D2*>, <*V, D3*>, <*D1, D1*>, <*D2, D2*>, <D3, *D3*>}

Contrary to Brody's assumption, suppose we can copy the same category onto different nodes in telescope trees, as long as their 'specifier positions[4]' are filled. That is, suppose we could extend the head categories (e.g., V in (6)) if the specifier position is filled. In linguistic analysis, we do want a structure as in (6), to express the well-known asymmetry between the two object positions with regard to reflexive pronoun binding. Unfortunately, the relational structure as in (7) cannot express the asymmetry between D2 and D3. We could use different category names V1 and V2 for the specs D2 and D3, violating the above mentioned constraint that the number of items in *Cat* corresponds to the number of overt phonological words in the string to be generated. However, as I have indicated above, multiplying the number of items beyond the number of phonologically visible words compromises the restrictiveness of the grammar system.

Restricted duplication of head categories such as V and T is linguistically useful, as we see in Section 4. In Section 3, we propose a representation system that can copy categories only in special cases. This system still has significantly weaker expressive power than Phrase Structure trees.

3 Definition of CSSs

This section introduces the proposed representation system. Unlike labelled tree structures, the system simply cannot represent the syntactic copying of one item in

[4] In (6), D1, D2, D3 are the specifier of T, V, V respectively

an infinite number of times. However, unlike the relational structure that we have presented in Section 2, our representation system can express a limited number of copies in the structure. We argue that this is more explanatory than accepting the multiplication of the items in *Cat* beyond the number of lexical items identifiable in the overt phonological string. We also indicate that German V2 phenomena can be neatly explained by the way our representational system restricts syntactic copying.

Following Bury (2003), we replace each tree by a Constituent Structure Set (CSS), which is a set of 'treelets.' In this representation system, each structure is given as in (8).

(8) *Structure* := <*Cat, CSS, RC*>

Cat is the **numeration set**, or the set of categories, which is isomorphic to the set of selected lexical items for the structure in our analysis. As with relational structures in Section 2, it is important to restrict the membership of *Cat* by some external evidence. Thus, we assume that for each structure the number of items in *Cat* basically corresponds to the number of overt PF expressions. As we have indicated before, in application, we assume that *Cat* may include T even if it does not host an overt word in the string to be generated, since it is linguistically well-motivated, for example, in terms of tense information which is often expressed as verbal morphology, some agreements between the verb and the subject (in the spec of T) and the special status of the external argument of the verb.

On the other hand, our syntactic system does not require each *Cat* to include particular categories/items. As with the relational structure in the previous section, no syntactic rule mentions particular category names. Thus, even though each CSS may have T in *Cat* without T hosting a PF word in it, we would still expect some eternal evidence for its presence.

We will soon find out that even with this restriction of not mentioning particular category names in the narrow syntax, we can still express significant amount of structural constraints. However, certain kinds of constraints/requirements which traditional syntactic theories use are not definable in our system, such as selection requirements. Such requirements will be expressed as type requirements in the semantics in our system, as we briefly see in Section 6. The requirement of tense information for propositional expressions and the correct ordering between functors and their arguments are also assigned at the level of the semantic interpretation.

For each $a \in Cat$, we have at least one treelet in the form as in (9a). Also, if a category *a* heads a treelet, the dominance set in the treelet must contain *a* as a member, as stated in (9b). *CSS* is a set of such treelets.

(9) Treelet:

a. In each structure, for each $a \in Cat$, CSS contains at least one treelet x such that $x = \{a, Dx\}$ and $\{a\} \subseteq Dx \subseteq Cat$.

b. In each CSS, for each $x \in CSS$ and for each $a \in Cat$, if a is the head of x, then $\{a\} \subseteq Dx \subseteq Cat$.

RC (mnemonic for `reflexive containment') in (8) is a binary relation between treelets, which is analogous to reflexive dominance, though unlike reflexive dominance which is defined between tree-nodes in labelled tree representations, *RC* is defined between treelets. Without further restriction on *RC*, the definition of treelets as in (9) is too generous and overgenerates CSSs for our purpose. Thus we restrict possible CSSs by defining the containment relation *RC* between treelets as a partial order. Before that, we explicitly require that the containment relation between treelets must be isomorphic to the containment relation between the dominance sets of those treelets, as shown in (10). For each treelet $x \in CSS$, Dx represents the dominance set of the treelet x. Intuitively, for each category a and for each treelet $x=\{a, Dx\}$, the dominance set Dx is the set of all categories that are reflexively dominated by a. However, this explanation is used for comparison reasons and unlike the relational structure presented in the previous section, and our system is not dependent on the notion of reflexive dominance defined over category names.

(10) a. Reflexive Containment (*RC*): $\forall x, y \in CSS. (RC(x, y) \Leftrightarrow (Dx \supseteq Dy))$

b. Immediate Containment (*IC*): $\forall x, y \in CSS.$

$(IC(x, y) \Leftrightarrow$

$(RC(x, y) \;\&\; x \neq y \;\&\; (\neg\exists z \in CSS.(RC(x,z) \;\&\; RC(z,y) \;\&\; z \neq x \;\&\; z \neq y))))$

The basic relation of our structural representation system is *RC* in (10a), but (10b) defines the derived relation of Immediate Containment *IC* which is useful for showing some proofs and also for the PF linearization in later sections.

Now, in each structure, the *CSS* has a unique ‘maximal’ treelet with regard to *RC*.

(11) Maximal treelet, $\exists x \in CSS. \forall y \in CSS. RC(x, y)$

Reflexive containment *RC* is a partial order, as in (12).

(12) a. Reflexitivity: $\forall x \in CSS. RC(x, x)$

b. Transitivity: $\forall x,y,z \in CSS.[(RC(x, y) \& RC(y, z) \rightarrow RC(x, z)]$

c. Antisymmetry: $\forall x,y \in CSS.[(RC(x, y) \& RC(y, x) \rightarrow (x=y)]$

Note that because of Antisymmetry in (12c) together with the definition of *RC* as in (10a), it follows that a CSS cannot contain two treelets that have the same dominance set but have different heads. To see this point, suppose that the dominance set Dx of $x \in CSS$ and the dominance set Dy of $y \in CSS$ contain exactly the same members of *Cat*, then $Dx \subseteq Dy$ & $Dy \subseteq Dx$. By (10a), $RC(x,y) \& RC(y,\underline{x})$. Then, by (12c), $x=y$. According to our interpretation of `=,' this means that x and y must be identical, which means that x and y must be the same in the dominance set and the head.

We also assume *RC* has the following up-ward non-branching property in (13a). The maximally binary branching constraint in (13b) is provisional, but it plays some non-trivial role when we define PF structures and interpretation of our CSSs as phonological strings.

(13) a. Upward non-branching:

$$\forall x,y,y' \in CSS.[(RC(y, x) \& RC(y', x) \rightarrow (RC(y, y') \vee RC(y', y))]$$

b. Maximally binary branching:

$$\forall x,y,z \in CSS. ((\{x' \in CSS \mid RC(x', x) \ \& \ x' \neq x\} = \{y' \in CSS \mid RC(y', y) \ \& \ y' \neq y\}$$
$$= \{z' \in CSS \mid RC(z', z) \ \& \ z' \neq z\}) \rightarrow ((x=y) \vee (x=z) \vee (y=z))$$

c. Unique splittablility: $\forall x,y \in CSS$.

$$((\{x' \in CSS \mid RC(x',x) \ \& \ x' \neq x\} = \{y' \in CSS \mid RC(y',y) \ \& \ y' \neq y\} \ \&$$
$$x \neq y) \rightarrow (Dx \cup Dy = \emptyset)).$$

The unique splittability prevents one CSS from containing two treelets such as $\{d, \{d, e\}\}$ and $\{c, \{c, e\}\}$. Together with the other conditions that we have introduced so far, it follows that if there exists a treelet $z = \{a, Dz\}$ such that $IC(z,x)$ and $IC(z,y)$, then, the dominance set Dz must be the union of Dx and Dy and $\{a\}$.

To prove this, suppose that Dz had two distinct categories a and b which are not members of Dx or Dy, where a is the head of z. Then by (9a), the CSS must have at least one treelet $w=\{b, \{b,...\}\}$. Now, since $b \in Dz$, because of the unique splittability in (13c) together with (9a), it follows that (a): $RC(z,w)$. The next paragraph proves this point.

If $RC(z,w)$ were not the case, because of the maximally binary branching constraint in (13b) and because of the presence of the maximal treelet as in (11), there could be only two subcases, Case A and Case B. In Case A, there would exist another treelet u such that $u \neq z$, $u \neq w$, $RC(u,z)$ and $RC(u,w)$. But this would violate the unique splittability in (13c) since u would immediately contain two treelets both of whose dominance sets contain b (that is, Dz contains b, and wherever w is placed

in the other branch, w would be reflexively contained in the maximal treelet in that branch which would be immediately contained by u, and thus the treelet that u immediately contains would have a dominance sent that contains b). In Case B, $RC(w,z)$ and $w \neq z$. But then, by definition of RC, Dw must contain at least one element which does not appear in Dz. Suppose there is exactly one element $e \in Cat$ such that $e \in Dw$ but $e \notin Dz$. Because of (9a), the CSS must have a treelet $v = \{e,\{e,...\}\}$, but since e does not appear in Dz, it must be either the case that there is another treelet s such that $s \neq v$, $s \neq w$, $RC(s,w)$ and $RC(s,v)$, in which case we would violate unique splittability in (13c), or that there is another treelet t which is not w and which immediately contains w and which does not immediately contain any other treelet. But in this latter case, again, because of the definition of RC, Dt must contain some element $f \in Cat$ which is not in Dw. However, since each Cat is finite, at a certain stage, we would have a situation such in which this extra element f could not be supplied. Suppose it were the case for Dt already (that is, $Cat = Dx \cup Dy \cup \{a, b, e\}$), then $Dt = Dw$ (i.e., $Dw \subseteq Dt$ and $Dt \subseteq Dw$). But remember that the head of t must be e since we have postulated t in order to satisfy the requirement in (9a). This would then violate the antisymmetry in (12c), since $RC(t,w)$ and $RC(w,t)$ but $w \neq t$ (that is, their dominance sets are the same, but their heads are different). We could have assumed that Dt contains some other elements than the elements in $Dx \cup Dy \cup \{a, b, e\}$, but as we mentioned above, since Cat is fine, and since those additional elements cannot occur in w or any treelets w reflexively dominates, we would have the same contradiction at a certain stage.

This concludes that $RC(z,w)$. But this would violate maximally binary branching constraint in (13b) since there would then be another treelet that was immediately contained by z other than x and y.

Since the above proof is based on the assumption that Dz contains an element $b \in Cat$ which is not the same as the head of z or a member of Dx or Dy, the same proof can be used when Dz contains another element that is not the same as a or a member of Dx or Dy. Thus, Dz must be the union of Dx and Dy and $\{a\}$ where $a \in Cat$ is the head of z.

Closure in (14) is automatically satisfied by (12a).

(14) Closure (satisfied by (12a)):
$(\forall x \in CSS.\ \exists y \in CSS.\ RC(x, y))\ \&\ (\forall y \in CSS.\ \exists x \in CSS.\ RC(x, y))$

As we have already seen in the main sections, each CSS is a set and so is each dominance set Dx. Thus, we interpret n occurrences of one item in each set as one, as shown in (15).

(15) Denotational interpretation of sets:

$\forall a \in Cat.\ \{a, a, a\} = \{a\}$

$\forall a, b \in Cat.\ \{\{a, \{a, b\}\}; \{a, \{a, b\}\}\} = \{\{a, \{a, b\}\}\}$ ... etc.

Given the restriction in (15), *CSS* can still contain more than one treelet for one category, say, $a \in Cat$, such as $\{a, \{a, b, c\}\}$, $\{a, \{a, b\}\}$ and $\{a, \{a\}\}$, where $Cat = \{a, b, c\}$. As we see in the next section closely, (15) means that CSSs cannot distinguish some of the copy structures that PS trees can represent as distinct. This inability to express syntactic copying is not stipulated in our system, it follows from the basic property of CSSs.

The set-based interpretation of CSSs as in (15) already assigns the maximal bound to the number of copies of an item in *Cat*. However, with linguistic application in mind, we add additional constraint to restrict copying possibilities even further, as in (16).

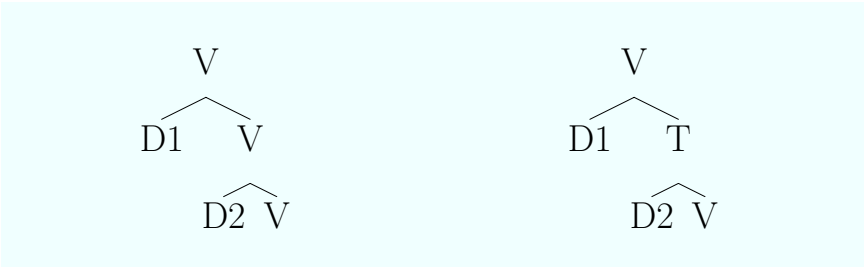
(16) $\forall a \in Cat.(\exists x, y \in CSS.(head'(x)=a\ \&\ head'(y)=a\ \&\ RC(x, y))$

$\rightarrow \neg\exists z \in CSS.(RC(x, z)\ \&\ RC(z, y)\ \&\ z \neq x\ \&\ z \neq y\ \&\ head'(z) \neq a))$

where $\forall a \in Cat, \forall x \in CSS$ such that $x = \{a, Dx\}$: $head'(x)=a$.

Translated in telescope trees, (16) requires that all the copies of one item enter into successive immediate dominance relation. Thus, with the restriction in (16), (17a) can be represented by a well-formed CSS in (18a), but the CSS in (18b), which is for (17b), is ill-formed.

(17) a. Well-formed in CSS: b. Ill-formed in CSS:



(18) a. *CSSa* (well-formed):

$\{\{V, \{V, D1, D2\}\}; \{V, \{V, D2\}\}; \{V, \{V\}\}; \{D1, \{D1\}\}; \{D2, \{D2\}\}\}$

b. *CSSb* (ill-formed):

$\{\{V, \{V, D1, T, D2\}\}; \{T, \{T, V, D2\}\}; \{V, \{V\}\}; \{D1, \{D1\}\}; \{D2, \{D2\}\}\}$

Unlike the inability of our CSSs to distinguish certain copy structures because of (15), CSSs could represent the structure in (17b) as (18b) if we did not have the restriction in (16). Thus, the restriction on copying based on (16) does not follow from the basic expressive power of our representation system itself. However, note that the restriction in (16) is stated without mentioning a particular category/item in the rule. (16) is a definable rule of our structure representation system.

Finally, we show that the CSS in (19) represents the asymmetry between D2 and D3 in (6), the kind of asymmetry that we could represent in the relational structure only by distinguishing each occurrence of V as a separate category, such as V1 and V2, where V1≠V2.

(19) *Cat*: $\{T, V, D1, D2, D3\}$

CSS: $\{\{T, \{T, V, D1, D2, D3\}\};\ \{V, \{D2, D3\}\};\ \{V, \{V, D3\}\};$
$\{V,\{V\}\};\ \{D1, \{D1\}\};\ \{D2, \{D2\}\};\ \{D3, \{D3\}\}\}$

Thus, the CSS system is more expressive than the relational structure in Section 2. However, CSSs are less expressive than PS trees, as we show in section 4.

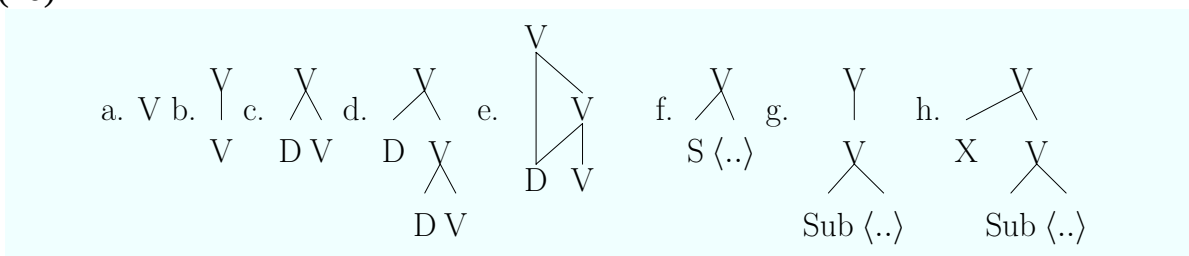
In this section, we have formally defined our structural representations as Constituent Structure Sets. Given a finite set of category (= lexical items), each CSS is a set of all constituents with distinguished members which we called heads. We have shown that we can still assign a significant amount of structural constraints without mentioning particular category names, such as T and V.

4 Representational collapsibility

This section compares the expressive power of the representation system defined in Section 3 with the expressive power of labelled tree representations, especially in terms of categorial copying.

In unordered sets, we cannot distinguish multiple occurrences of a category from one occurrence, as in $\{X, X\}=\{X\}$. Thus, CSS cannot distinguish certain structures that PS trees can. Compare (20) and (21).

(20)



CSSs in (21a)~(21e) represent the trees in (20a)~(20e) in our system.

(21)

a. $\{\{V,\{V\}\}\}$

b. $\{\{V,\{\underline{V},\underline{V}\}\};\ \{V,\{V\}\}\} = \{\underline{\{V,\{V\}\}};\underline{\{V,\{V\}\}}\} = \{\{V,\{V\}\}\}$=(1a)

c. $\{\{V,\{\underline{V},\underline{V},D\}\};\{V,\{V\}\};\{D,\{D\}\}\}$
$= \{\{V,\{V,D\}\};\{V,\{V\}\};\{D,\{D\}\}\}$

d. $\{\{V,\{\underline{V},\underline{V},\underline{V},\underline{D},\underline{D}\}\};\{V,\{\underline{V},\underline{V},D\}\};\{V,\{V\}\};\underline{\{D,\{D\}\}};\underline{\{D,\{D\}\}}\}$
$= \{\underline{\{V,\{V,D\}\}};\underline{\{V,\{V,D\}\}};\{V,\{V\}\};\{D,\{D\}\}\} = (1c)$

e. $\{\{V,\{\underline{V},\underline{V},\underline{V},D\}\};\{V,\{\underline{V},\underline{V},D\}\};\{V,\{V\}\};\{D,\{D\}\}\} = (1d) = (1c)$

Two tree structures in (20a, b) collapse into one CSS, as shown in (21a, b). Thus, projection of V is impossible without a filled specifier, that is, D in (c) which produces a different CSS in (18c). Also, in CSS, we cannot fill this spec position by copying a category from a lower position in the tree as in (20d). In CSS, (20d) is equivalent to (20c), as is shown in (21c, d). Moreover, as (21c~e) show, CSS cannot distinguish the multiple dominance structure (MDS) in (20e) from the copy-chain structure in (20d) or from the non-movement structure in (20c) (cf. Kracht (2001) shows that copy chains and MDS are formally equivalent).

In linguistic applications of CSS, ``self-attachment'' of a head (i.e. copying of V as in (20c)) is possible with a filled specifier, whereas movement/copying into a “specifier position” is not expressible. Thus, for A/A-bar movement phenomena, we must resort to either base generation analysis or use of distinct categories/lexical items that are related by way of the semantics, as we briefly explain in Section 6.

As supporting data for the restricted projection or ``remerge'' of (head) categories as in (21), we briefly discuss German V2 phenomenon. In the German V2 pattern, a fronted verb must be preceded by a single phrasal constituent, XP. Crucially, XP can be of any category and does not receive a uniform interpretation (cf. Haider (1993). Thus, an analysis abstracting away from category names and the interpretations of categories will be more explanatory. (The varying interpretations of the fronted constituent will be explained by the semantics/pragmatics).

In CSS, a tree structure where V is “remerged” without a spec, that is, (20g), is undistinguishable from (20f) and thus, without a filled specifier, it leads to the same PF order, Sub-V-<..>. If however a structure contains a remerged V with an additional specifier, that is, the tree structure in (20h), then its CSS will be distinct from the CSS for (20f). This means that a moved verb can only be pronounced in the PF position of a “remerged” category (i.e. the PF position of the higher V in

(20h)) if it has a filled specifier. This specifier's category or interpretation is irrelevant, as long as it isn't empty (although it may contain a null operator, as in yes/no questions, which are verb-initial). The basic V2 pattern is thus derived from structural principles, without the introduction of any features that lack an independent motivation.

This section has shown that our representation system is much weaker in expressive power than labelled tree representations. We have also suggested that our way of restricting copying of categories can provide some insight about German V2 phenomena. The next section compares CSSs with labelled tree representations from a different viewpoint.

5 Decidability

As is well known, the set of all the possible sets of positive integers is not enumerable, or undecidable in its membership, as is provable in a diagonal proof (cf. Boolos, Burgess, and Jeffrey 2002:16-20).

(22) The set of all the possible sets of positive integers: undecidable.

{ {1, 3, 5, 7, 9, … },
{2, 4, 6, 8, 10, … },
{3, 5, 7, 9, 11, … },
{4, 6, 8, 10, 12, …},
..... }

If we see a grammar of a language as a set of syntactic structures that are grammatical in that language, then it is easy to show that such a set is undecidable in its membership, if we represent each structure as a labelled tree and if we also assume that we can potentially copy any one label (i.e., one member of the finite numeration set) into an infinite number of nodes.[5]

In contrast, the set of all (well-formed) CSSs in a language is decidable in its membership, given a finite set of total lexical items.

[5] We assume that one lexical item in a language, such as *Meg* in English may occur as separate items in a numeration set Cat, to deal with a sentence, such as *Meg ran and Meg swam*, as we come back to later. Thus, the set of all the possible numeration sets is assumed to be enumerably/countably infinite relative to a finite set of total lexical items from which we can select the members of each numeration set.

(23) a. Proposition 1: Each CSS is a finite set of treelets.

Proof:

1. Each *Cat* is finite by our assumption. That is, for some finite number *n*, $Cat = \{A_1,..., A_n\}$.
2. Given 1, for each $A_i \in Cat$ (where $1 \leq i \leq n$), the number of treelets which A_i can head is finite (bounded by $|Cat| = n$). And therefore, the number of treelets in each *CSS* is finite. Q.E.D.

   (Cf. For each treelet, the dominance set *D* is finite.)

b. Proposition 2: The set of all (well-formed) *CSS*s is decidable in its membership (or the set of CSSs is enumerable), given the following two assumptions:

Premise 1: The set of total lexical items in the language, *Lex*, is finite.

Premise 2: Given a finite *Lex*, we can choose a potentially infinite number of different numeration sets, *Cat*. This is because one member in *Lex* (i.e., one lexical item in a language) may occur as different members in *Cat*, as in $Cat = \{Mary_1, Mary_2, and, smoke, ran\}$ for the sentence $Mary_1$ *smoked and* $Mary_2$ *ran*.

Informally, the proof of (23b) goes as in the following. Given a finite set *Cat*, each *CSS* is finite, as shown in (23a). Also, given a finite *Cat*, we can only have a finite number of distinct CSSs. Because the set of all the *Cats* (i.e. the set of all the numeration sets) in a language is enumerable, where we can have only a finite number of CSSs for each *Cat*, the set of all (well-formed) CSSs in a language is enumerable (relative to a finite set of lexical items from which we can select *Cats*). In other words, the membership of the set of all CSSs in a language is decidable in its membership.

In contrast, as we have indicated above, in tree representations making use of independent notions of nodes and labels, each tree structure can potentially be infinite in its size, even relative to a finite set of labels. Thus, if we can select an infinite number of different numeration sets (of labels) in such an analysis (relative to a finite set of total labels), then the set of well-formed labelled tree structures in a language becomes undecidable in its membership. Of course, we could add additional constraints to limit the number of the copies of one label in a tree structure, as is commonly done in application of labelled tree structures in linguistic analyses, but we argue that such restriction should better be assigned at the foundational level of the representational system in a systematic manner, rather than being added post-hoc independent of the maximal expressive power of the representational system. Just as an addition of a non-logical rule to a deductive

system makes the system incomplete with regard to the intended interpretation, such a post-hoc rule will make the grammar system incomplete.

This section has shown that our CSSs are decidable in their memberships with their numeration sets Cats being finite. The final two sections show how we can define the semantic interpretations and PF linearization, given the limited expressive power of our structure representation systems.

6 Semantics

In (19), we stipulated the spec-head asymmetry and the order among heads (T-V) (from which we can calculate the minimal element as T), so that the syntax would generate only the desired CSS, but these constraints do not have syntactic properties. Unlike the syntactic conditions in Section 3, the order among category names does not help distinguish one kind of structures from another. In our view, the spec-head asymmetry is an asymmetry between arguments and functors in the semantics, and the order between T and V is the selection order between the corresponding semantic functors. Thus, we attribute them to the semantics. To explain how it works, we show the interpretation process for *John can play tennis*.

(24) a. Lexical entries:

$</play/; \mathrm{V}; \lambda x.\lambda y.\mathrm{play}'_{e(et)}\, xy>; </can/; \mathrm{T}; \lambda P.\lambda z.\mathrm{can}'_{(et)(et)}\, Pz>;$

$< /ally/; \mathrm{D1}; ally'_{e}>; < /tennis/; \mathrm{D2}; tennis'_{e}>$

b. Identification (cf. Identity Axiom as in logical proofs):

$\{D1, \{D1\}\}: \{john', \{john'\}\}$ (cf. $john' \Rightarrow john'$)

c. Function application (with one argument):

$\{V, \{V, D2\}\};\ \{\lambda y.play'tennis'y, \{\lambda x.\lambda y.play'xy,\ tennis'\}\}$

d. Function application (with two arguments):

$\{T, \{T, D1, \underline{\{V, D2\}}\}\};$

$\{(can'(play'tennis'))ally', \{\lambda P.\lambda z.can'Pz,\ ally',\ \underline{\lambda y.play'tennis'y}\}\}$

Each lexical item has a triplet entry, <*PF item*; Category; logical expression>. In (24a), we associate T with *can* since T hosts the auxiliary in this string. Treelets in the form of $\{X,\{X\}\}$ correspond to lexical identification in the semantics, as in (24b). The logical expressions are all simply typed. To see how function application works, look at (25).

(25) a. Cat: {*T*, *V*, *D1*, *D2*}

b. CSS 1: {{*T*, {*T*, *V*, *D1*, *D2*}}; {*V*, {*V*, *D2*}};
{*V*, {*V*}}; {*D1*, {*D1*}}; {*D2*, {*D2*}}}

CSS 2 (Alt): {{*T*, {*T*, *V*, *D1*, *D2*}}; <u>{*D1*, {*D1*, *D2*}}</u>;
{*V*, {*V*}}; {*D1*, {*D1*}}; {*D2*,{*D2*}}}

c. Sem (for CSS1):

{{(*can′*(*play′tennis′*))*ally′*, {λ*P*.λ*z*.*can′Pz*, λ*y*.*play′tennis′y*, *ally′*}};
{λ*y*.*play′tennis′y*, {λ*x*.λ*y*.*play′xy*, *tennis′*}};
{λ*x*.λ*y*.*play′xy*, {λ*x*.λ*y*.*play′xy*}}; {*ally′*, {*ally′*}}; {*tennis′*, {*tennis′*}}}

After lexical identification[6], we successively apply functions to their arguments. Each function application must correspond to a treelet that contains a functor expression and one or two arguments of the right type(s), which is interpreted in the form of either (24c).[7] After each function application, the output is compiled into the treelet one step larger in terms of the constituent containment relation in CSS. For example, in (25c), *λy.play′tennis′y* as the output of the function application as in (24c) is compiled into the largest treelet where function application applies in the form of (24d) (note that in (24d), V and D2 together count as one argument of the functor *λP.λz.can′Pz* in the semantics). Because of this successive compilation and the types of the semantic items, we do not need to order the items in dominance sets.

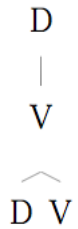
CSS2 in (25b) is syntactically well-formed, but the underlined treelet, {$ally'_e$, {$ally'_e$, $tennis'_e$}}, is not interpretable and the semantic composition does not converge. In other words, though the syntax itself does not distinguish functor categories such as T and V from argument categories such as D, the semantics does. We instantiate this semantic compositionality in terms of type compatibilities.

As another example, the CSS in (26) is semantically excluded because the underlined treelet is not interpretable. The D expression would have to be the functor over the V expression but we exclude Type Raising in the semantics.

[6] Functor categories such as V do not have to have identity treelets, though they can.

[7] Because of the max binary branching in (13b), the number of arguments in each *Da* set is maximally two. Thus, only one or two of the *n*-arguments of an *n*-ary functor can be saturated at each step, as is obvious in (24c) and (24d).

(26) a.



b. $\{\{\underline{\{D, \{D, V\}\}}\}; \{V, \{V, D\}\}; \{D, \{D\}\}; \{V, \{V\}\}\}$

We interpret all D(P) terms as type e expressions. For quantificational DPs such as *every boy*, we adopt the type e analysis which uses epsilon terms, as in Kempson et al. (2001: chapter 8).

In (24) and (25c), the functor $\lambda P.\lambda z.\text{can}'Pz$ inherits the external argument-slot of $\lambda x.\lambda y.\text{play}'xy$. We use this argument-slot percolation in terms of (higher) functors when we treat some A-movement phenomena in base generation analyses. Because our grammar cannot represent copying into a specifier (as opposed to copying onto the top of the "projection line"), we would like to explain the A movement phenomena in terms of percolation of type e argument-slots within the verbal projection line. Mostly, we can do this by modifying the lexical entries of the higher functors, such as auxiliary verbs (e.g. *can*), control verbs (e.g. *try*) and raising verbs (e.g. *seem*), so that we can percolate even the internal argument-slots of the selected lower verb. But in scrambling languages such as Japanese, we would need to permute DP arguments among themselves, by incorporating morphological case information. Though we do not provide the details here, the basic idea is similar to the one in directionless categorial grammar frameworks such as Muskens (2003). That is, as long as we permute the arguments at PF side and at LF side in the same way, we do not mix up the arguments in a wrong way. Because our syntactic system itself does not constrain the structural placement of D categories, incorporation of such permutation effects will be straightforward.

For A′-movement phenomena, we mostly rely on the use of distinct categories/lexical items that are related in the semantics (e.g. with a semantic identity function $\lambda X_a.X_a$ as the dependent element in a "trace" position in the traditional grammar, cf. Jacobson (1999)). Various islands to A′-movement are explained in terms of linking the dependent element with its operator in the semantic structure.[8]

[8] See Truswell (2007) for some examples of constraints on Wh-movement coming from event structures

7 PF linearization

PF mapping is still under development and we only provide a sketch.

For the convenience of presenting PF linearization rules, we first define the irreflexive containment relation *C* and the immediate containment relation *IC* between treelets in (28). *C* is simply a reflexive version of *RC* as we have defined in Section 3. (27) repeats the most basic part of the definition of *RC* in (10). For notation, for each treelet *x*, *Dx* represents *x*'s dominance set.

(27) Containment (*RC*): $\forall x, y \in CSS.(RC(x,y) \Leftrightarrow (Dx \supseteq Dy))$

When we linearize a CSS, we successively linearize the members of the dominance sets of the treelets according to the **(proper) constituent containment relation *C*** and **immediate constituent containment relation *IC*** between treelets as defined in (28). (28a) and (28b) define *C* and *IC* from the basic syntactic relation *RC*.

(28) a. Containment *C*:

$\forall x, y \in CSS. (C(x, y) \Leftrightarrow (RC(x, y) \ \& \ x \neq y))$

b. Immediate Containment (*IC*): $\forall x, y \in CSS.$

$(IC(x,y) \Leftrightarrow (RC(x,y) \& x \neq y \& (\neg \exists z \in CSS.(RC(x,z) \& RC(z,y) \& z \neq x \ \& z \neq y))))$

Since *C* and *IC* are relations derived from *RC*, they are indirectly constrained by the restrictions on *RC* as we have assigned in Section 3, though some of the properties that *RC* has might not apply because of the definitions in (28). For example, *IC* is no longer transitive whereas *RC* and *C* are. And *C* is irreflexive by definition whereas *RC* is reflexive. Note that each CSS is closed and partially ordered in terms of *RC*, with a maximal treelet. Thus, we can specify in which order we linearize the treelets in each CSS with *RC*, but it is easier to provide the order instruction with *C* since it is a strict partial order, as we see shortly.

With the triplet lexical entries as in (24a), we provide the set of PF lexical items as in (29a). The definition of PF strings is as in (29b).[9]

[9] The connective '·' is non-commutative and non-associative, but we sometimes omit the parentheses for presentation reasons. See footnote 12.

(29) a. Given *Cat* for *CSS*: *PFLex* is the set of phonological lexical items whose category entries are members of *Cat*.

b. Φ_{max}: the set of **potential** PF strings, given *PFLex*.

i) If $a \in PFLex$, then $a \in \Phi_{max}$

ii) If $a, b \in \Phi_{max}$, then $(a \cdot b) \in \Phi_{max}$

iii) The set of PF units: $\Phi_{CSS} \subseteq \Phi_{max}$

Given *CSS*, *Cat*, *PFLex* and (28), we successively linearize the Dominance Sets of the treelets, starting with the treelets that are the lowest in the order in terms of *C* in (28a) (i.e., the identity treelets) and finishing with the maximal treelet. That is, for all $x, y \in CSS$, if $C(x, y)$, linearize (the dominance set of) y before linearizing (the dominance set of) x. If $\neg C(x, y) \& \neg C(y, x)$, then we can linearize x and y in either order.

For the convenience of presenting the linearization rules, we change the notations of the variables for treelets from simple variables such as x and y in (27) to variables with additional subscript variables which specify the **heads** of the treelets. As we have discussed in Section 3, each treelet represents a structural constituent with a distinguished category, X, as in the form $\{X, D_X\}$, where D_X represents the set of categories that are reflexively dominated by X.[10] Again, we call X the head of the treelet. To explicitly show the head category of each treelet, for each $X \in Cat$, we let t_X represent the treelet headed by X. Some $X \in Cat$ can be heads of two or more treelets. Thus, sometimes we need to use subscripts on X, such as in t_{X1}, t_{X2}, t_{XId}, to distinguish distinct treelets headed by the same category X. The third treelet t_{XId} represents the `identity treelet' for X, that is, $\{X, \{X\}\}$. An identity treelet is represented in this way only if X heads at least one more treelet other than the identity treelet. The subscripts in category names are used for presentation reasons only and X_1, X_2, X_{Id} as appearing as subscripts in t_{Xi} (for $i = 1, 2, Id$) do not mean that X_1, X_2 and X_{Id} are three distinct categories in the CSS. For each category $X \in Cat$, D_X represents the dominance set of the treelet headed by X. The use of subscripts $1, \ldots, n, Id$ on X as in $D_{X1}, \ldots, D_{X2}, D_{Id}$ is the same as in its use in the meta-variables for treelets.

Now with these notations, we represent the linearization process of each treelet as in (30a)~(30b). We successively incorporate the output of each treelet t_X into the dominance set of the treelet t_Y that immediately contains t_X. The membership of Φ_{CSS} depends on the linearization of *CSS* and the definition of "PF units."

[10] Here, we refer to the common reflexive dominance relation between categories for convenience.

(30) a. {*output-unit*, {*input-unit*}}

b. E.g.:

$\{\{(c\cdot(a\cdot b)), \{c, (a\cdot b)\}\}; \{(a\cdot b), \{a, b\}\}; \{a,\{a\}\}; \{b,\{b\}\}; \{c,\{c\}\}$

For each treelet, the output PF string counts as **one PF unit**, as indicated by the pair of parentheses.[11] The PF unit status may change with the D_X set that is being linearized at each stage. In (30b), $(a \cdot b)$ counts as one PF unit in the D_X set of the maximal treelet, but in the D_X set of the second largest treelet, a and b are separate PF units. Φ_{CSS} in (29b) contains all the units that are derived at any stage of CSS linearization process as its members.[12] We incorporate the output of each treelet linearization to the D_X set of the treelet that immediately contains it. The linearization of the maximal treelet of CSS is the final output string.

Given (28) and (30), we provide the basic PF linearization principle of our system in (31). (31b)~(31c) are the formal implementations of the basic idea in (31a).

(31) a. Immediate Containment as PF adjacency (ICPA): Immediate containment between treelets in CSS corresponds to PF adjacency between the corresponding PF units.

b. $\forall t_X, t_Y \in CSS$, when we linearize t_X, t_Y:

If $IC(t_X, t_Y)$, then $(x\cdot y')$ or $(y'\cdot x)$ is the output PF unit of t_X, where x is the PF lexical item for the head category X and y' is the PF unit as the output of the linearization of t_Y.

c. $\forall t_X, t_Y, t_Z \in CSS$, when we linearize t_X, t_Y, t_Z:

If $(IC(t_X,t_Y)$ & $IC(t_X, t_Z))$, then $(y'\cdot(x\cdot z'))$ or $((z'\cdot x)\cdot y')$ or $((y'\cdot x)\cdot z')$ or $(z'\cdot(x\cdot y'))$ is the output PF unit of t_X, where x is the PF lexical item for the head of t_X and y' is the PF unit as the output of the linearization of t_Y and z' is the PF unit as the output of the linearization of t_Z

(31a) expresses the intuitive idea of our PF linearization of CSSs. With the maximally binary structure constraint on *RC*, in our CSSs, one treelet can

[11] Though we put the output string in the head position of the treelet, this is for notational convenience, and does not mean that the head category in the syntax corresponds to the derived PF output string.

[12] The membership of Φ_{CSS} depends on the way we linearize the CSS. Since the PF units are maximally binary branching, some of the unit statues during the linearization process can be read off the output unit of the maximal treelet in the CSS. However, we cannot always do that since some bracketing introduced via the rule in (31c) does not correspond to a PF-unit status at any stage

immediately contain one or two treelets. In this paper, we assume that PF structures are maximally binary and also, non-commutative and non-associative. Thus, we need the basic idea in (31a) stated separately for cases when a treelet immediately contains only one treelet (as stated in (31b)) and for cases when a treelet immediately contains two treelets (stated in (31c)). The idea behind (31c) is that PF adjacency is not sensitive to bracketing. For example, the PF items *x* and *y* count as being adjacent in the structure ($x\cdot(y\cdot z)$).

As we have indicated above, we could assume that PF structures are associative. Then, we would not need an additional rule in (31c), on top of the basic (31b), but for the moment, we maintain the bracketing in the PF structure.

(32) explains our treatment of syntactic copies in PF linearization and flexible pronunciation positions of verbal heads.

(32) a. $\forall X \in$ *Cat*, if the corresponding PF item *x* appears more than once in the generated PF string, *x* can potentially be pronounced in any of those positions, whereas the other positions are PF null.
b. A verbal head item, such as *play* of V, can be pronounced in the PF position of another head in the same "projection line" (such as T - V).

(31) together with (32) provide flexibility in terms of syntax-phonology mapping, which overgenerates PF strings without further constraints. However, the idea is that, as in the syntax-semantics interface, independent PF considerations can provide non-trivial constraints on PF linearization. For example, as we see briefly later, structural DP case assignment by a verbal head may require a certain PF configuration in a morphologically impoverished language such as English. We also assume a certain asymmetry between PF and LF, in the sense that PF only linearizes the CSSs which are syntactically and semantically well-formed. However, the linearization rules that refer to some semantic concepts (such as the notion of 'verbal heads' and 'projection line' in (32b) and the rules that only refer to the narrow syntactic elements abstracted away from particular category names are different in nature in the proposed system. Thus, we can calculate the PF objects that CSSs can generate via the interpretation rules in (31) abstracted away from the semantics.

We add further linearization constraints beyond (31)~(32). Such constraints are postulated either for maintaining the LF-PF correspondence as something external to the narrow syntactic rules, or for some well-formedness restrictions at the level of PF. As an example, we mention one candidate constraint which is for maintaining some LF-PF correspondence.

(33) The start of each propositional unit at LF must be clearly marked at PF.

We take a conservative view that each TP is the minimal propositional unit.[13] In the default case, the start of a propositional expression is marked by the spec item for the T head, that is, the subject DP. We provide a simple example.

(34) a. *Cat*: {*T*, *V*, *D1*, *D2*}, *PFLex*: {*can*, *play*, *ally*, *tennis*}

CSS:{{*T*, {*T*,*V*,*D1*,*D2*}}; {*V*, {*V*,*D2*}}; {*V*, {*V*}}; {*D1*, {*D1*}; {*D2*, {*D2*}}

b. Linearization:

t_T: {(*ally* · (can · (play · tennis))), {*can*, *ally*, (*play*· *tennis*)}}

t_V: {(*play* · *tennis*), {*play*, *tennis*}}

t_{VId}:{*play*,{*play*}}; t_{D1}:{*ally*,{*ally*}}; t_{D2}: {*tennis*, {*tennis*}}}

The category V occurs in two treelets as the head, and I distinguish the identity treelet for V as t_{Vid} from the other, which is t_V. This is for presentational convenience. When we linearize t_V, we could have generated (*tennis·play*), as may happen in German, but in English, structural case assignment by the verb requires the object DP to the right of the verb and this alternative is rejected. Because of (33), we first pronounce the specifier DP1, when we linearize t_T, and for English, *Ally·can·play·tennis* is the generated PF string (in German, the same CSS generates *Ally·can·tennis·play*, with further linguistic assumptions which we omit).

With regard to (33), when some element is overtly extracted to the left of the subject, as in *Tennis, Ally plays*, then *tennis* occupies the Spec of the C head, which is the highest head, and thus the spec is pronounced first, to mark the starting boundary of the propositional element. When the Spec of the highest head is PF null, we may pronounce the highest head first, as in *Did Ally play tennis*, where *did* is in the C position. We omit more complicated cases for space reasons.

Given a *CSS* = {{*can*, {*can*, *run*, *Meg*}}; {*run*, {*run*, *Meg*}}; {*run*, {*run*}}; {*Meg*, {*Meg*}}}, the flexible linearization rules in (31) can produce both ((*Meg·can*)· *smoke*) and ((*Meg·can*)·*smoke*), among two more PF structures. Now, ((*Meg·can*)·*smoke*) might not be very useful in linguistic analysis. Coordination as in *Meg can, but Avril cannot swim* does not preserve both the immediate containment relation between *can* and *run* and the IC between *can* and *swim*, and thus, does not require the internal structure as in ((*Meg·can*)·*smoke*), either. As another rule motivated for maintaining LF-PF consistency, we might require both the 'head copies' and the 'related heads' to be bracketed first, as stated in the rule in (35).

[13] We do not exclude potential incorporation of *phases* into our system, though.

(35) $(IC(t_X, t_Y)$ & $IC(t_X, t_Z)) \rightarrow (y'\cdot(x\cdot z'))$ or $((z'\cdot x)\cdot y')$ in the output PF unit of t_X when x and z' are related heads (such as *can* and *run*) or when $x=z'$.

The notion of 'related heads' are describable only at LF in our system and thus, (35) cannot be stated at the level of linearization rules of CSSs abstracted away from the semantics.[14] In that sense, (35) is fundamentally different from the rules in (31) which are the only basic linearization rules that we postulate, where (32) will be stated at the level of how to pronounce the generated PF structures.

In our system, one category $x \in Cat$ can head two treelets t_{X1} and t_{X2} in such a way that '$IC(t_{X1}, t_{X2})$' holds. Then, the whole output of the linearization of t_{X2} is either preceded or followed by x. With (32a), x will then be pronounced in one of the two positions of x in the PF string.

With (35), suppose that x and z' are related heads, such as *can* and *drive*, where *can* is the head of the maximal treelet t_{can} which immediately contains the treelet headed by *drive*. Suppose also that the other treelet immediately contained by t_{can} is t_{meg} headed by *meg*. If these are the case, then the bracketing will become either (*Meg*·(*can*·*smoke*)) or ((*smoke*·*can*)·*Meg*). The latter one will then be excluded because of (33). Note that one treelet can immediately contain maximally two treelets, because of the maximally binary branching constraint on *RC*. With (35), the PF string *Meg can Jack kick* can still be generated from a CSS in which the treelet t_{can} is the maximal treelet which immediately contains both t_{Meg} = {*Meg*, {*Meg*}} and t_{kick1} = {*kick*, {*kick*, *Jack*}}, since z' as the output of the treelet t_Z in (35) may have an internal structure, as in (*Meg*·(*can*·(*Jack*·*kick*))).

On the other hand, the word order in *(that) Meg smoke can*, which may appear in an embedded clause in German, would require a different CSS, such as CSS_g = {{*can*, {*can*, *meg*, *smoke*}}; {*smoke*, {*smoke*, *meg*}};{*meg*, {*meg*}}}, in which the maximal treelet t_{can} does not immediately contain the identity treelet t_{Meg}. With CSS_g and (35), we would be able to generate the string, ((*smoke*·*meg*)·*can*), and it is not clear if such a word order is attested in some language data, but again, we can exclude this PF by way of (33) above. The restriction in (33) needs to be stated more accurately, but we leave a better definition together with its status for future research.

Note that adjacency as in (31a) is a symmetrical (and non-transitive) relation. Because of that, the mapping of CSSs onto PF strings is not even homomorphism. Nevertheless, a more flexible syntax-PF mapping (which can be further constrained by specific system external constraints such as PF case checking) can be more explanatory than a rigid one that requires a more expressive syntax that treats PF concepts in syntactic terms.

---

[14] The rule in (35) when x=z is definable without the semantics.

8 Conclusion

Our structure representation system has two characteristics, (i) it only represents structured configurations of category names, without referring to the (semantic or phonological) interpretations of the category names, and (ii) it can express copying of categories only in special cases. Thus, we can focus on purely structural elements of natural language. On the other hand, this purely structural syntax requires further investigation about the matching semantics and PF linearization.

References

Boolos, G. S., J. P. Burgess, and R. C. Jeffrey (2002). *Computability and Logic*. Cambridge: Cambridge University Press.

Brody, M. (2000). Mirror theory: Syntactic representation in perfect syntax. *Linguistic Inquiry 4*, 29–52.

Bury, D. (2003). *Phrase Structure and Derived Heads*. Ph. D. thesis, University College London, London.

Cornell, T. (1998). Derivational and representational views of minimalist grammar. In

A. Lecomte, F. Lamarche, and G. Perrier (Eds.), *Logical Aspects of Computational Linguistics*, Heidelberge, pp. 92–111. Springer-Verlag.

Haider, H. (1993). *Deutshce Syntax Generativ*. Tubingen: Gunter Narr.

Jacobson, P. (1999). Towards a variable free semantics. *Linguistics and Philosophy 22*, 117–185.

Kepser, S. (2006). Properties of binary transitive closure logic over trees. In P. Monachesi,

G. Penn, G. Satta, and S. Winter (Eds.), *Proceedings of 11th Conference on Formal Grammar, Stanford*, pp. 77–90. CSLI Publications.

Kracht, M. (2001). Syntax in chains. *Linguistics and Philosophy 24*, 467–529.

Kurtonina, N. (1994). *Frames and Labels: a Modal Analysis of Categorial Inference.* PhD Thesis, Utrecht University, Utrecht.

Muskens, R. (2003). Language, lambdas and logic. In A. Lecomte, F. Lamarche, and G. Perrier (Eds.), *Resource Sensitivity in Binding and Anaphora*, pp. 23–54. Dordrecht: Kluwer.

Truswell, R. (2007). A semantic constraint on wh-movement. In *Proceedings of ConSOLE XV*.