Constituent Structure Sets

In this talk, we discuss a structure representation system that replaces Phrase Structure (PS) trees with set-based representations of relational structures. The goal is to restrict the expressive power of the grammar at the foundational level. In this respect, it is the same sort of endeavour as Seki's Multiply Context Free Grammar, MCFG (Seki et al, 1991, Stabler 2004). As in MCFG, we discuss the syntax only as structured configurations of category names, without using further information (e.g. features) encoded with the categories. This enables us to focus on purely structural factors abstracted away from the constraints that should more naturally be attributed to the semantic composition or PF linearization. Unlike MCFG, however, which achieves restrictiveness in terms of the limited generative power of the derivation rules, our grammar achieves restrictiveness in terms of its limited power to represent structures. More specifically, the system cannot represent copying or projection of categories except for some special cases, which has interesting implications for various movement phenomena.

Because our syntax does not incorporate LF/PF elements, it under-specifies the interpretations at the two interfaces. Thus, as in the system sketched in Kracht (2004), the relation between the syntax and the semantics becomes different from syntactic theories in which the semantic structures can be read off the syntactic structures (e.g. UTAH-based Minimalism or Categorial grammar). As in Kracht's system, the semantic compositionality plays a non-trivial role.

Section 1 introduces the relational structures that underlie our set-based representations, which we explain in section 2. Section 3 shows the restrictiveness of our system with linguistic implications. Sections 4 and 5 briefly sketch the semantics and the PF linearization. Section 6 is the conclusion.

1. Relational structure

We represent syntactic trees as relational structures as in Landman (1991). Each structure is a pair as in (1), where *Cat* is a set of categories and *R* is a binary relation between categories. Each *S* has a minimal element, as in (1b). The membership of *Cat* is fixed for each *S*.

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

For each *S*, *R* is reflexive, transitive and antisymmetric. Each structure is upward non-branching.

(2) a. Reflexivity: $\forall a \in Cat. Raa$
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: $\forall a,b,c \in Cat. ((\{a' \in Cat \mid Ra'a \,\&\, a' \neq a\} = \{b' \in Cat \mid Rb'b \,\&\, b' \neq b\} = \{c' \in Cat \mid Rc'c \,\&\, c' \neq c\}) \rightarrow (a=b \vee a=c \vee b=c))$
f. Closure (satisfied by 2a): $(\forall a \in Cat. \exists b \in Cat. Rab) \,\&\, (\forall b \in Cat. \exists a \in Cat. Rab)$

R corresponds to the reflexive dominance relation (RD) in syntactic trees. But crucially, we define *R* as a relation between category names, without using an additional notion of 'tree nodes.'[1] Each member of *Cat* represents a lexical item (which may include functional items, e.g., for T and *v*). Reflexivity in (2a) satisfies Closure in (2f). 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.[2] Each set *Cat* is finite and has discrete members. Thus, each structure is finite.

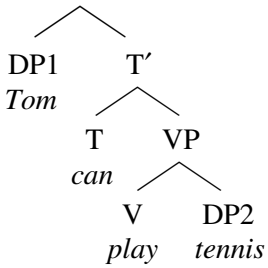
---

[1] One could see our 'category names' as lexically provided 'individuated nodes', instead.

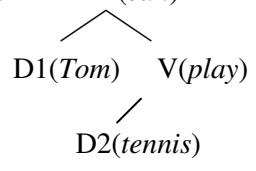
[2] ID is a special case of RD and can be derived from RD without a disjunctive condition. Also, ID cannot always be maintained via every P-morphic mapping between structures, whereas RD can be, cf. Kurtonina (1994:32). These considerations suggest that RD is more basic in relational structures, though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

The relational structures in (1)~(2) are free of categorical projection. Compare the system with Brody's (almost) projection free 'telescope' trees (cf. Brody 2000), with regard to *Tom can play tennis.*

(3) a.



b.



c. *Cat* = {T, V, D1, D2} or equivalently,
(= {*can*, *play*, *tom*, *tennis*})

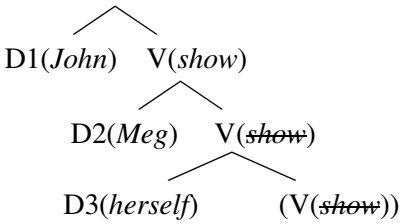
R = {<T, T>, <T, V>, <T, D1>,
<T, D2>, <V, V>, <V, D2>,
<D1, D1>, <D2, D2>}

Minimal element (m) = T

The telescope tree in (3b) reduces the two projection lines in the standard PS tree in (3a) to a single node, i.e., (i) TP-T′-T to T and (ii) VP-V to V. In our relational structure system, given the ordering among T, V, D1 and D2 (which we later attribute to the semantics), the structural representation is (3c), which is equivalent to (3b). Because the membership of *Cat* is fixed as is lexically provided, we cannot project the tree structure in (3a). The system cannot express copying/duplication of categories.

This representation system provides some formal support to the Chomskyan assumption of Lexical Inclusiveness. However, the system is too restrictive for linguistic application. Compare (4a) with (4b).

(4) a.



b. *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>}

Minimal element (m) = T

Telescope trees in Brody can extend head categories (e.g. V in (4a)) if the specifier position is filled (D1, D2, D3 are the specifier of T, V, V respectively). We want a structure as in (4a), to express the well-known asymmetry between the two object positions with regard to reflexive binding. Unfortunately, the relational structure as in (4b) cannot express the asymmetry between D2 and D3.

Restricted projection of head categories such is linguistically useful (see section 3). In section 2, we develop a representation system that can copy categories only in special cases. The system still has significantly weaker expressive power than Phrase Structure tree representations.

2. Constituent Structure Sets

Following Bury (2003), we replace each PS tree by a *Constituent Structure Set* (CSS). The relational structure in section 1 underlies CSS. Each CSS is a set of 'treelets,' where each treelet has the form in (5a). For each $a \in Cat$, $\{a, Da\}$ represents a constituent with a distinguished category a, where Da is the set of the categories that are reflexively dominated by a. The reflexive dominance relation is R as is defined in (1)~(2). We call Da a dominance set.

(5) a. Treelet$_a$: $\{a, Da\}$
b. CSS: $\{\{a, Da\}; \{b, Db\}; \{c, Dc\}; \{d, Dd\}\ldots\}$
c. $a, b, c, d\ldots \in Cat.$
d. $\forall a \in Cat.\ Da := \{b \in Cat \mid Rab\}$

The CSS in (6) represents the asymmetry between D2 and D3 in (4a) above.

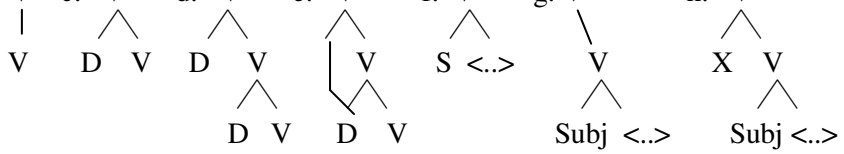
(6) *Cat*: {T, V, D1, D2, D3}
CSS: { {T, {T, V, D1, D2, D3}}; **{V, {V, D2, D3}}**; **{V, {V, D3}}**; ({V, {V}});
{D1, {D1}}; {D2, {D2}}; {D3, {D3}} }

Thus, the CSS system is stronger in its expressive power than the relational structure system in section 1. However, CSSs are less expressive than Phrase Structure trees, as we show in section 3.

3. Representational collapsibility and its linguistic implications

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 (7) and (8).

(7) a. V b. V c. V d. V e. V f. V g. V h. V



(8) CSS with $R \subseteq Cat \times Cat$ as in (1)~(2) (i.e. Reflexive Dominance relation)

a. {{V, {V}}}

b. {{V, {**V**, **V**}}; {V, {V}}} = {**{V, {V}}**; **{V, {V}}**} = {{V, {V}}} = (8a)

c. {{V, {**V**, **V**, D}}; {V, {V}}; {D, {D}}} = {{V, {V, D}}; {V, {V}}; {D, {D}}}

d. {{V, {**V**, **V**, **V**, **D**, **D**}}; {V,{**V**, **V**, D}}; {V, {V}}; **{D, {D}}**; **{D, {D}}**}
= {**{V, {V, D}}**; **{V, {V, D}}**; {V, {V}}; {D, {D}}} = (8c)

e. {{V, {**V**, **V**, **V**, D}}; {V, {**V**, **V**, D}}; {V, {V}}; {D, {D}}} = (8d) = (8c)

Two tree structures in (7a, b) collapse into one CSS in (8a). Thus, projection of V is impossible without a filled specifier, that is, D in (7c) which produces a different CSS in (8c). Also, in CSS, we cannot fill this spec position by copying a category from a lower position in the tree as in (7d). In CSS, (7d) is equivalent to (7c), as is shown in (8c, d). Also, as (8c~e) show, CSS cannot distinguish the Multiple dominance structure (MDS) in (7e) from the non-movement structure in (7c) (cf. Kracht 2001 shows that copy chains and MDS are formally equivalent).

In linguistic applications of CSS, head movement is possible with a filled specifier, whereas movement/copying into a 'terminal node' 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 4.

As supporting data for the projection or "remerge" of (head) categories as in (7), 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). Thus, an analysis abstracted away from category names or 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 (i.e. (7g)) is undistinguishable from (7f) and thus, without a filled specifier, it leads to the same PF order, Subj-V-<..>. If however a structure contains a remerged V with an additional specifier (cf. the tree structure in (7h)), its CSS will be distinct from (7f). This means that a moved verb can only occur in the PF position of a "remerged" category (i.e. the PF position of the higher V in (7h)) 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.

4. Semantics

In (6), 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 (1)~(2), the order among category names does not help distinguish one kind of (relational) structure from another. In our view,

the spec-head asymmetry is an asymmetry between arguments and functors in the semantics, and the order among T-(*v*)-V is the selection order among semantic functors. Thus, we attribute them to the semantics. To explain how it works, we show the interpretation process for *John can play tennis.*

(9) a.</play/; V; λ*x*.λ*y*.play′*xy*>; </can/; T ; λ*P*.λ*z*.can′*Pz*>; </meg/; D1;m′>;</tennis/; D2; ten′>
b. Identification, {D, {D}}: {john′, {john′}} (cf. john′ ⇒ john′)
c. Function application, {V, {V, D}}: {λ*y*.play′tennis′*y*, {λ*x*.λ*y*.play′*xy*, tennis′}}

Each lexical item has a triplet entry, <*PF item*; category; logical expression>. Treelets in the form of {X,{X}} correspond to lexical identification in the semantics, as in (9b). The logical expressions are all simply typed. To see how function application works in a treelet of the form as in (9c), look at (10).

(10) 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}}; **{D1, {D1, D2}}**; {V, {V}}; {D1, {D1}}; {D2, {D2}}}
c. Sem (for CSS1)
{ {(can′(play′tennis′))john′, {λ*P*.λ*z*.can′*Pz*, λ*y*.play′tennis′*y*, john′}};
{λ*y*.play′tennis′*y*, {λ*x*.λ*y*.play′*xy*, tennis′}};
{λ*x*.λ*y*.play′*xy*, {λ*x*.λ*y*.play′*xy* }}; {john′, {john′}}; {tennis′, {tennis′}} }

After lexical identification,[3] we successively apply functions to their arguments. Each function application must correspond to a treelet that contains the *n*-ary function and the *n* argument(s) of the right type(s), which is interpreted as in **{output, {functor^n, argument$_1$…argument$_n$}}** (because of the compilation mechanism below, and the max binary branching in (2e), *n* is maximally two). After each function application, the output is compiled into the treelet one step larger in terms of the constituent containment relation in CSS. In (10c), λ*y*.play′tennis′*y* as the output of the second treelet is compiled into the first treelet (i.e., in D_T, V and D2 counts as one argument of the functor **can′**). Because of this successive compilation and the semantic types, we do not need to order the items in dominance sets.

As for CSS2 in (10b), though it is syntactically well-formed, **{D1, {D1, D2}}** is not interpretable and the semantic composition does not converge.

In (10c), the functor λ*P*.λ*z*.can′*Pz* 'percolates' the external argument-slot of λ*x*.λ***y***,play′*x****y***. We use this argument-slot percolation in terms of (higher) functors when we treat some A movement phenomena in base generation analyses. 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 identify function λ*x*.*x* as the dependent element in an argument position of a verb, cf. Jacobson 1999).

5. PF linearization

PF linearization is still under development, but the basic Syntax-PF mapping rule is in (11).

(11) Immediate Containment as PF adjacency (ICPA): Immediate containment relation between treelets in CSS correspond to PF adjacency between the corresponding PF units.

Exactly how (11) works with our CSSs depends on the definition of the containment relation between treelets and some other formal details, which we explain in our talk. This abstract only provides an informal PF linearization example in (12). In some sense, our PF linearization strategy has a similar effect to treating (standard PS) binary trees as a mobile-like structure in which one can PF-linearize the two sisters at each binary branch (in the given syntactic structure) either left-to-right or right-to-left.

[3] The functor expressions for T and V do not have to have an identity treelet in the syntax (though they can); the closure condition in (1b) is satisfied in the treelet {V, {V,D,.}}. CSS1 in (10b) contains only {V,{V}}, but not {T, {T}}. The reason comes from the semantics, though we omit it for space in this paper. We also omit the semantic treatments of adjuncts and higher order operators on heads.

(12) a. {a PF-output-unit, {PF input units}}
b. E.g. $CSSX_{PF}$= {{($c \cdot a \cdot b$), {c, **($a \cdot b$)**}}; {**($a \cdot b$)**, {a, b}}; {b, {b}}} (N.B. a possible linearization)

We linearize the PF item(s) in the dominance set of each treelet, starting with the smallest treelet. In (12b), there are three PF lexical items, *a, b, c*, though CSSX has only one identity treelet, {b, {b}}. We first linearize this identify treelet, which is just the idenitification of *b*, which is incorporated into the dominance set (D set) of the second largest treelet in (12b). At this stage, we can either linearize the two PF units *a* and *b* as ($a \cdot b$) or ($b \cdot a$). Let us choose ($a \cdot b$). ($a \cdot b$) then counts as **one (complex) PF unit** when it is incorporated into the D set of the largest treelet (i.e. the unit status changes spontaneously). Finally, the D-set of the largest treelet can either be linearized as (($a \cdot b$)$\cdot c$) or ($c \cdot$($a \cdot b$)).[4] If we let *c* be S, *a* be V and *b* be O with the same CSS, without copying in syntax, we can generate the orders, SVO, SOV, VOS, OVS but not VSO or OSV. We argue that this flexible linearization strategy has some linguistic justification. We also provide some ideas about how we can constrain potential overgeneration caused by our flexible strategy by using specific Syntax external constraints such as PF case checking.

6. Conclusion

PS trees define structural relations (such as dominance) over tree nodes, and then decorate these nodes with labels (such as category names). Our syntax defines reflexive dominance directly over (lexically provided) category names (rather than using categories as labels of independently existing nodes). This eliminates the syntactic copying of lexically provided category names. Our CSS system allows us to make limited use of syntactic copying, but still, CSS is more restrictive than PS trees. Though our syntactic system has a merit in terms of its restrictiveness at the base level, it cannot incorporate semantic factors as labels of syntactic units (which makes it purely structural). This requires further investigation of the matching interpretation modules, which we mostly leave for further investigation.

References

Brody, M. (2000) Mirror theory: Syntactic representation in perfect syntax. *Linguistic Inquiry* 31: 29-56

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

Chomsky, N. (2005) Three Factors in Language Design, *Linguistic Inquiry 36, No 1*: 1-22

Cornell, T. (1998) Derivational and Representational Views of Minimalist Transformational Grammar. In A. Lecomte, F. Lamarche and G. Perrier (Eds.) *Logical Aspects of Computational Linguistics,* pp. 92-111. Heidelberg: Springer-Verlag.

Haider, H. (1993) *Deutsche Syntax Generativ*. Tübingen: 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 the 11th Conference on Formal Grammar,* pp. 77-90. Stanford: CSLI Publications.

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

Kracht, M. (2004) The Emergence of Syntactic Structure, *Talk at the Foundations of natural language grammar, ESSLLI'05 workshop.*

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

Landman, F. (1991) *Structures for Semantics,* Kluwer, Dordrecht.

Seki, H, T Matsumura M Fujii, and T Kasami. (1991) ON multiple context-free grammars. *Theoretical computer science,* 88:191-229.

Stabler, E. (2004) Varieties of crossing dependencies. *Cognitive Science* 28(5): 699-720.

[4] The two units, *c* and ($a \cdot b$), are adjacent to each other in either case. The connective · is non-commutative. There is some complication about its associativity, which we ignore in this abstract.