Theoretical Computer Science 579 (2015) 33–63

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Categorial dependency grammars

Michael Dekhtyar [a,*,1], Alexander Dikovsky [b,2], Boris Karlov [a,1]

[a] Dept. of CS, Tver State University, Tver, 170000, Russia
[b] LINA UMR CNRS 6241, Université de Nantes, France

ARTICLE INFO

Article history:
Received 26 February 2014
Accepted 23 January 2015
Available online 11 February 2015
Communicated by Z. Esik

Keywords:
Formal grammar
Dependency grammar
Categorial dependency grammar
Push-down automata with independent counters

ABSTRACT

Categorial Dependency Grammars (CDGs) are classical categorial grammars extended by oriented polarized valencies. At the same time, CDGs represent a class of completely lexicalized dependency grammars. They express both projective and non-projective dependencies. CDGs generate non-context-free languages. At that, they are parsed in polynomial time under realistic conditions. CDGs possess a normal form that is analogous to Greibach normal form for cf-grammars. CDG-languages are closed under almost all AFL operations and are accepted by a special class of push-down automata with independent counters.

1. Introduction

Categorial grammars. Categorial grammars (CG) is the eldest class of formal grammars. They go back to the work Ajdukewicz [1] whose idea was to define a grammar G as a mapping λ from a dictionary W (a finite set of words) to finite sets of formulas. The language $L(G)$ generated by G is then defined in terms of a system of derivation rules R and of a derivability relation $\vdash_R$ in the way that $L(G)$ is the set of all strings $w_1 \dots w_n$ over W such that from a string of formulas $\tau_1 \dots \tau_n$, where $\tau_i \in \lambda(w_i)$, $1 \le i \le n$, is derivable a special primitive formula S: $\tau_1 \dots \tau_n \vdash_R S$. In modern terminology the grammars defined through a mapping of the dictionary are called `lexicalized`. Ajdukewicz used the propositional formulas called `categories` which were either propositional letters or expressions $(\alpha/\beta_1, \dots, \beta_k)$ where $\alpha, \beta_1, \dots, \beta_k$ are formulas and / is the (unique) right connector. The derivation rules in his grammars were of the form $(A/B_1, \dots, B_k)B_1, \dots, B_k \vdash A$ where $A, B_1, \dots, B_k$ are meta-variables to be replaced by categories.

Later, in the papers of Bar-Hillel and his co-authors [3,4], the categorial grammars of Ajdukewicz were extended with left connector \. At the same time, the two connectors have become binary (i.e. the form of the categories has become (α/β) or $(\alpha\backslash\beta)$). These categorial grammars are called `classical` (or AB). Finally, Lambek (see [33,34]) has defined a Gentzen style calculus of AB-categories (traditionally called `Lambek calculus` and denoted **L**) giving rise to the `Lambek grammars`. After some partial results of Buszkowski (see [7]) Pentus [43] proved that Lambek grammars are equivalent to context-free grammars. In the 80-ies other variants of the Lambek calculus (and respectively, of Lambek grammars) were defined. Starting from this period (especially after the appearance of the semantics of Montague (see [37])), the categorial grammars have become one of the most popular instruments of mathematical analysis of natural language.

* Principal corresponding author.
E-mail addresses: Michael.Dekhtyar@tversu.ru (M. Dekhtyar), bnkarlov@gmail.com (B. Karlov).

[1] This work was supported by the RFBR grants 13-01-00643 and 13-01-00382.

[2] 1945–2014.

http://dx.doi.org/10.1016/j.tcs.2015.01.043
0304-3975/

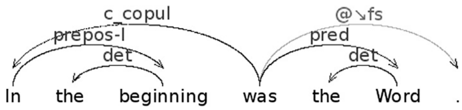

Fig. 1. A projective dependency tree with inverted WO.

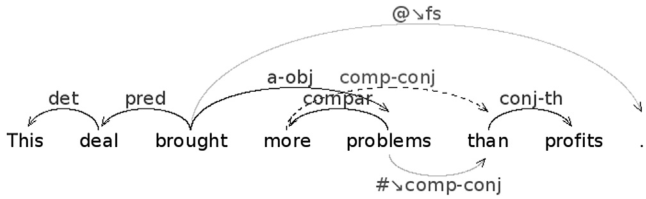

Fig. 2. Non-projective dependency in English *more .. than* comparison construction.

The categorial grammars may be naturally considered as grammars assigning constituent structures to the generated strings: if $\beta_1 \dots \beta_n \in \lambda(w_1 \dots w_n)$ and $\alpha \vdash_G \beta_1 \dots \beta_n$, then $w_1 \dots w_n$ is a constituent of category α. Meanwhile, in this paper it will be a matter of categorial grammars with a different interpretation, assigning to the generated strings not constituent, but dependency structures.

Dependency structure. A dependency structure of a sentence is a graph whose nodes are the words of the sentence and the arcs are labeled by the names of (binary) syntactic relations. In general, the dependency structures are trees (dependency trees) or at least are cycle-free. If a word w_1 is in relation d with a word w_2, which is denoted $w_1 \xrightarrow{d} w_2$, then w_1 is a governor of w_2 and w_2 is a subordinate of w_1 (through dependency d). The dependency relations are anti-reflexive, anti-symmetric and anti-transitive. So is also the immediate dominance relation $w_1 \Rightarrow w_2 \equiv \exists d(w_1 \xrightarrow{d} w_2)$. Its reflexive-transitive closure $\Rightarrow^*$ is called dominance. The set $proj(w) = \{w' \mid w \Rightarrow^* w'\}$ of the words dominated by w is a projection of w. A dependency structure D of a string x is not always considered together with the linear order $<$ of precedence of words in x, but when it is linearly ordered, one may express very important properties of D in terms of projections. Namely, a dependency $w_1 \xrightarrow{d} w_2$ in D is projective if every word w of x in the interval $[w_1, w_2]$ when $w_1 \le w \le w_2$, or in $[w_2, w_1]$ when $w_2 \le w \le w_1$, is dominated by w_1 (i.e. $w \in proj(w_1)$). Otherwise $w_1 \xrightarrow{d} w_2$ is non-projective. D is projective if every dependency in D is projective (or, equivalently, every projection $proj(w)$ is an interval of x).[3] Clearly, if D is projective, then every two projections either have no words in common, or one is a subset of the other: $w_1 \Rightarrow^* w_2$ iff $proj(w_2) \subseteq proj(w_1)$. In other words, if a dependency tree of x is projective, then the set of projections forms a constituent structure of x.

For instance, in Fig. 1, one may see a projective dependency tree. Indeed, the projections of all words in this tree are intervals.

Linguistic theories treat the precedence order in different ways. In historically the first theory of L. Tesnière (see [49]), the dependency trees (called there stemmas) only partially reflect the surface precedence order. I. Mel'čuk, in [36], distinguishes between the deep dependency trees and the surface dependency trees. The former do not reflect the precedence order and are in fact similar to the Tesnière's stemmas. The latter are linearly ordered by the precedence order. On the other hand, in his Word grammar (see Hudson [26]), R. Hudson, presumes that every correct dependency tree must be projective. This assumption is doubtful (and is not followed by many researchers in the domain of NLP) because in many well-known languages (seemingly, in all languages) there are regular syntactic constructions using non-projective dependencies. For instance, in Figs. 2, 3, we see two typical examples of non-projective dependencies in English.

The dependency tree in Fig. 2 is non-projective because the projection $proj(more)$ contains *problems* which is not dominated by *more*.

The dependency tree in Fig. 3 is non-projective because the projection $proj(pilot)$ contains the root *lost* which is not dominated by *pilot*.

Dependency grammars. The grammars assigning dependency structures to correct sentences are called dependency grammars. There are two main frames of dependency grammars: constraint grammars and tree generating grammars.

Historically the first dependency grammars Hays [23,24], Gaifman [20] were constraint grammars. They encoded the elementary formulas: “a word belongs to a grammatical class”, “a word immediately precedes a word”, “a word governs

[3] This is equivalent to the fact that no two dependencies cross and none of them, $w_1 \xrightarrow{d} w_2$, contains the tree root strictly between w_1 and w_2.