Transition-Based Coding and Formal Language Theory for Ordered Digraphs

Yli-Jyrä, Anssi

The Association for Computational Linguistics
2019-09-23

Yli-Jyrä , A 2019 , Transition-Based Coding and Formal Language Theory for Ordered Digraphs . in H Vogler & A Maletti (eds) , The 14th International Conference on Finite-State Methods and Natural Language Processing : Proceedings of the Conference . Proceedings of the International Conference on Finite-State Methods and Natural Language Processing , þÿ T h e A s s o c i a t i o n f o r C o m p u t a t i o n a l L i n g u i s t i c s , S t r o u d s b u r g , p p . 1 1 8 Conference on Finite State Methods and Natural Language Processing , Dresden , Germany , 23/09/2019 . https://doi.org/10.18653/v1/w19-3115

http://hdl.handle.net/10138/306880
https://doi.org/10.18653/v1/w19-3115

Transition-Based Coding and Formal Language Theory for Ordered Digraphs

Anssi Yli-Jyrä
University of Helsinki, P.O. Box 4, FIN-00014, Finland
anssi.yli-jyra@helsinki.fi

Abstract

Transition-based parsing of natural language uses transition systems to build directed annotation graphs (digraphs) for sentences. In this paper, we define, for an arbitrary ordered digraph, a unique decomposition and a corresponding linear encoding that are associated bijectively with each other via a new transition system. These results give us an efficient and succinct representation for digraphs and sets of digraphs. Based on the system and our analysis of its syntactic properties, we give structural bounds under which the set of encoded digraphs is restricted and becomes a context-free or a regular string language. The context-free restriction is essentially a superset of the encodings used previously to characterize properties of noncrossing digraphs and to solve maximal subgraphs problems. The regular restriction with a tight bound is shown to capture the Universal Dependencies v2.4 treebanks in linguistics.

1 Introduction

Transition systems have been a widely used mechanism in language understanding, cognitive modelling of natural language processing and syntactic and semantic parsing of sentences. The combination of high parsing speed of transition systems with the accuracy of the attached statistical models have paved the way for practical applications of parsing and similar data transformations enabled by these systems (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004; Zhang and Nivre, 2011; Chen and Manning, 2014; Dyer et al., 2015; Andor et al., 2016; Kiperwasser and Goldberg, 2016; Shi et al., 2017). Transition systems may also be used to encode dependency trees, DAGs and other ordered digraphs, and to connect these to the classical formal language theory and to the problems of *graph representation* (Turán, 1984; Farzan and Munro, 2013; Yli-Jyrä, 2019), *graph enumeration* (Pólya, 1937; Conte et al., 2018; Yli-Jyrä, 2019), *integer sequence discovery* (Hoppe and Petrone, 2016; Yli-Jyrä and Gómez-Rodríguez, 2017), *maximum subgraph inference* (Conte et al., 2019; Yli-Jyrä and Gómez-Rodríguez, 2017), *algebraic representations of graph queries* (Courcelle, 1990; Ogawa, 2004; Yli-Jyrä and Gómez-Rodríguez, 2017), *encoder-decoder parsing* (Vinyals et al., 2015; Strzyz et al., 2019) and *parsing as sequence labeling* (Gómez-Rodríguez and Vilares, 2018).

The study of transition systems in general ranges from Turing complete transition systems (Woods, 1970; Goldin et al., 2004; Thomas, 2002) to well-understood transition systems that build projective dependency structures, context-free parse trees and noncrossing graphs (Nivre, 2003, 2004; Goldberg and Elhadad, 2010; Kuhlmann et al., 2011; Sagae and Tsujii, 2008; Honnibal and Johnson, 2015). The need to model nonlocal dependencies and crossing edges in parses have motivated the study of transition systems that balance the computational complexity and the coverage of the possible outputs. Many of these systems are extensions of stack-based transition systems (Attardi, 2006; Nivre, 2009; Gómez-Rodríguez and Nivre, 2013; de Lhoneux et al., 2017; Qi and Manning, 2017; Gómez-Rodríguez et al., 2018) but there are also some proposals for transition systems that are based, solely or additionally, on some other memory model, such as a shack, a list, registers, a set, or a cache (Kornai and Tuza, 1992; Covington, 2000;

Proceedings of the 14th International Conference on Finite-State Methods and Natural Language Processing, pages 118–131
Dresden, Germany, September 23-25, 2019.

Choi and McCallum, 2013; Pitler and McDonald, 2015; Fernández-González and Gómez-Rodríguez, 2018; Gildea et al., 2018; Vilares and Gómez-Rodríguez, 2018; Coavoux and Cohen, 2019).

In this paper, we present a transition system that implements an efficient, invertible function between its action sequences and arbitrary ordered digraphs. The action sequences of the system can also viewed as strings of balanced brackets, constituting formal languages that have elegant Chomsky-Schützenberger representations and many desirable characteristics of input-driven languages. The transition-based transformation between the relational and sequential representations of digraphs opens a possibility to apply classical formal language theory of subsets of free monoids to the classes of digraphs.

Our transition system takes advantage of a new kind of decomposition of a digraph: the *rope decomposition* views the underlying graph as a union of subgraphs what we call *ropes*. The longest edge of a rope shares exactly one endpoint with each of the other edges in it.

The theoretical notions of rope decompositions and the new transition system are introduced in Sections 3-4. The action sequences of the transition system are related to formal languages in Sections 5-6. Sections 7-8 contain corpus-based empirical evaluation and a discussion that argues that the developed encoding for digraphs contributes to the work in some related and important research areas in graph theory and computer science.

2 Basic Definitions

Denote the empty string with ϵ. Denote transpose of a binary relation X as X^T. Define the composition of two binary relations X, Y as $X \circ Y = \{(x, z) \mid (x, y) \in X, (y, z) \in Y\}$. Abbreviate an assignment $S \leftarrow S \cup T$ as $S \overset{\cup}{\leftarrow} T$. Let V_n denote the finite set of integers $\{1, ..., n\}$. Let the parameter $d \in \{<, >, <>\}$ indicate the choice between leftward, rightward and bidirectional orientation of arcs in actions that produce these arcs.

A *(finite ordered) graph* is a pair (V_n, E) where V_n is a finite set of ordered vertices and $E \subseteq \{(u, v) \in V_n \times V_n \mid u < v\}$ is a set of edges. For each edge $(i, j) \in E$, we call i the left index and j the right index of the edge. A *(finite ordered) digraph* is a pair (V_n, A) where V_n is a set of vertices and $A \subseteq \{(u, v) \in V_n \times V_n \mid u \neq v\}$ is a set of arcs. The underlying graph of a digraph (V_n, A) is the graph (V_n, E_A), where $E_A = \{(i, j) \mid (i, j) \in A \cup A^T, i < j\}$.

3 New Notions

3.1 Rope Cover

Definition 3.1. Let (V_n, E) be an ordered graph. In this graph, edge (h, k), where $h < k$, is a *(properly-longer shared-endpoint) covering edge* for a shorter edge (i, j) if either $h = i$ and $i < j < k$, or $j = k$ and $h < i < j$. Denote this situation by $(h, k) : (i, j)$.

Definition 3.2. A subset $R \subseteq E$ is a *rope cover* of the graph (V_n, E) if, for every edge $e \in E \backslash R$, there is an edge $c \in R$ such that $c : e$. Moreover, R is a *proper rope cover (PRC)* if is there is no edges $c_1, c_2 \in R$ s.t. $c_1 : c_2$.

Proposition 3.1. *Any element in a PRC can be identified by specifying either its left index or its right index.*

Proposition 3.2. *Every graph has a PRC.*

Theorem 3.3. *The PRC is unique.*

Proof. Let (V_n, E) be an arbitrary graph and let $R, R' \subseteq E$ be two PRCs of the graph. Assuming that $R \neq R'$ and that there is $(x_0, y_0) \in R \backslash R'$, we show, by induction, that there is an infinite sequence of distinct edges $(x_0, y_0), (x_2, y_2), (x_4, y_4), ... \in R \backslash R'$ and $(x_1, y_1), (x_3, y_3), (x_5, y_5), ... \in R' \backslash R$ where $(x_{i+1}, y_{i+1}) : (x_i, y_i)$ for every $i \geq 0$. To prove the required induction steps, there is, by the definition of a PRC, a covering edge (x_{i+1}, y_{i+1}) in $R' \backslash R$ for every edge $(x_i, y_i) \in R \backslash R'$, and there is a covering edge $(x_{i+1}, y_{i+1}) \in R \backslash R'$ for every edge $(x_i, y_i) \in R' \backslash R$. Such an infinite sequence of distinct edges requires E to be infinite. By contradiction, the PRC of the graph is unique. □

Definition 3.3. For graph (V_n, E) with a PRC R, the *rope-thickness of a vertex* $i \in V_{n-1}$ is the number of edges $(h, j) \in R$ satisfying $h \leq i < j$. The *rope-thickness of the graph* is the maximum over the rope-thicknesses of all vertices $i \in V_{n-1}$ in the graph.

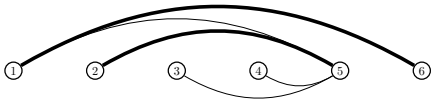
Lemma 3.4. *For any graph of n vertices, its PRC is constructed in $O(n^3)$.*

Proof. Let (V_n, E) be a graph. To construct the PRC, start with $R_0 = \emptyset$ and $E_0 = E$. Given R_i and E_i, $i \geq 0$, construct R_{i+1} as the set all edges in E_i that do not have a covering edge in E_i, and E_{i+1} as the set of all edges in E_i that do not have a covering edge in R_{i+1}. Each such iteration is computed in $O(n^2)$. Clearly, $E_{i+1} \subsetneq E_i$ unless $E_i = \emptyset$, and $E_{\lfloor n/2 \rfloor} = \emptyset$. The PRC of the graph is the set $R = R_1 \cup R_2 \cup ... \cup R_{\lfloor n/2 \rfloor}$, and it is constructed in $O(n^3)$ time. □

Corollary 3.5. *The rope thickness of a graph can be computed in cubic time.*

Example (4) is a graph $([6], \{(1,6), (1,5), (2,5), (3,5), (4,5)\})$ for which we obtain sets

$$R_0 = \{\} \qquad E_0 = E \tag{1}$$
$$R_1 = \{(1,6)\} \quad E_1 = \{(2,5), (3,5), (4,5)\} \tag{2}$$
$$R_2 = \{(2,5)\} \quad E_2 = \{\} \tag{3}$$



(4)

Convention 3.4 ("Left Index")**.** We will sometimes refer to the edges in a PRC by their left indices (see Proposition 3.1).

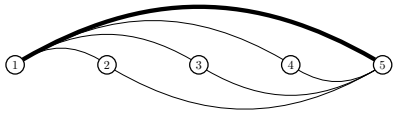
Convention 3.5 ("Indirect Edges")**.** When an edge (i, j) has a covering edge (h, j), $h < i < j$, we refer to the edge (i, j) indirectly, via the pair (i, h) where h is the LEFT INDEX of the covering edge.

The PRC of the graph (4) comprises the edges $\{(1,6), (2,5)\}$, while the remaining edges are covered by these. Edge $(1,5)$ is a usual edge, and there are INDIRECT EDGES, $(3,2)$ and $(4,2)$, that we draw under the vertices. The rope thickness of vertices 2 – 4 is two, which is also the maximum for the whole graph.

3.2 Ropes

Definition 3.6. An ordered graph (V_n, E) is called a *rope* if $n = 1$ or the PRC of the graph is $\{(1, n)\}$. This is *complete* if $E = \{(i, n) \mid 1 \leq i < n\} \cup \{(1, j) \mid 1 < j \leq n\}$.

Ropes can be used in algorithms that construct graphs while processing vertices. Example (5) shows a digraph whose underlying graph is a complete rope. Some of its edges would cross one another if the edges were drawn above a line containing the vertices.



(5)

Two-way algorithms can build a complete rope in two passes with a vertex counter and one memory unit that contains a reference to one index of the covering edge. After memorizing the left index of the covering edge to variable x, such an algorithm processes vertices 2 to 5 in Example (5) and builds edges $(x, 2)$, $(x, 3)$, $(x, 4)$ and $(x, 5)$. During the backward pass over the vertices, the algorithm saves the right index of the covering edge to variable x and builds edges $(4, x)$, $(3, x)$, and $(2, x)$.

One-way algorithms process each vertex only once as their output can represent the edges (i, n), $2 \leq i \leq n - 1$, in Example (5) indirectly, by a reference to the left index of the respective covering edge. In the output, the edge $(2, 5)$ is represented as an INDIRECT EDGE $(2, 1)$ where 1 identifies the LEFT INDEX of the covering edge $(1, 5)$. After processing the vertices, the composition of $(2, 1)$ and $(1, 5)$ is computed to obtain the actual edge $(2, 5)$.

3.3 Rope Assignment

Theorem 3.6. *There are graphs where an edge has two distinct covering edges in the PRC.*

Proof. The PRC of the graph of Example (6) is $\{(1,3), (2,4)\}$.

(6)

The edge $(2,3)$ is covered by the edge $(1,3)$ with which it shares the right index, and by the edge $(2,4)$ with which it shares the left index. □

When an edge has two covering edges, we need a consistent policy for treating them. In (6), we can *assign* the edge $(2,3)$ to either of the covering edges $(1,3)$, $(2,4)$, or both.

Convention 3.7 ("EARLIEST")**.** We adopt a convention according to which the ambiguity between two possible covering edges is resolved by assigning the edge to the earliest available covering edge.

By Convention 3.7, the arc $(2,3)$ in Example (6) will be assigned to the EARLIEST covering edge $(1,3)$, which can be identified by its LEFT INDEX 1, by Convention 3.4. The covered arc $(2,3)$ is thus represented as an INDIRECT EDGE $(2,1)$ by Convention 3.5.

3.4 Rope Decomposition

This section introduces a new representation for digraphs. The idea is to start from the PRC of the underlying graph, and then assign the remaining arcs to covering edges.

Definition 3.8. A *rope decomposition* of an (ordered) digraph (V_n, A) is a tuple $(V_n, R, \underset{<}{A}, \underset{>}{A}, \underset{<}{I}, \underset{>}{I})$ satisfying the following conditions:

1. edges $R \subseteq E_A$ constitute a PRC of the underlying graph (V_n, E_A),
2. $\underset{<}{A} \subseteq \{(i,j)\in A^T \mid (i,k)\in R, i<j\leq k\}$ and $\underset{>}{A} \subseteq \{(i,j)\in A \mid (i,k)\in R, i<j\leq k\}$ are, respectively, left and right arcs whose left index coincides with the left index of the covering edge,
3. $\underset{<}{I}\subseteq\{(i,h)\mid (i,j)\in A^T, (h,j)\in R, h<i<j\}$ and $\underset{>}{I}\subseteq\{(i,h)\mid (i,j)\in A, (h,j)\in R, h<i<j\}$ are, respectively, indirect representations for arcs whose right index coincides with the respective covering edge but is represented indirectly, via the left index of the respective covering edge.
4. The four sets of arcs represent together the original set of arcs: $A = \underset{<}{A}^T \cup (\underset{<}{I} \circ R)^T \cup \underset{>}{A} \cup (\underset{>}{I} \circ R)$.

Lemma 3.7. *Under the "*EARLIEST*" convention, the relation between digraphs and rope decompositions is a bijection.*

Proof. (⇒): Let $G = (V_n, A)$ be a digraph and $G' = (V_n, E_A)$ its underlying graph. By Theorem 3.3, G' has a unique PRC $R \subseteq E_A$. By the "EARLIEST" convention, we choose the earliest available covering edge for each edge and first construct the sets of indirect arcs $\underset{<}{I} = \{(i,h) \mid (i,j)\in A^T, (h,j)\in R, h<i<j\}$ and $\underset{>}{I} = \{(i,h) \mid (i,j)\in A, (h,j)\in R, h<i<j\}$. After this, we construct the sets of arcs whose left index coincides with the covering edges: $\underset{<}{A} = \{(i,j)\in A^T\setminus(\underset{<}{I} \circ R) \mid (i,k)\in R, i<j\leq k\}$ and $\underset{>}{A} \subseteq \{(i,j)\in(A\setminus(\underset{>}{I} \circ R) \mid (i,k)\in R, i<j\leq k\}$.

(⇐): Let $(G = V_n, R, \underset{<}{A}, \underset{>}{A}, \underset{<}{I}, \underset{>}{I})$ be a rope decomposition. We obtain the corresponding graph as $(V_n, \underset{<}{A}^T \cup (\underset{<}{I} \circ R)^T \cup \underset{>}{A} \cup (\underset{>}{I} \circ R))$. □

4 A New Transition System

By Lemma 3.7, there is a bijection between (a class of) rope decompositions and digraphs. We complement this result by relating each rope decomposition bijectively to a sequence of actions. The actions are controlled by a transition system.

Our transition system has a *buffer* $\beta \in \mathbb{N}$, a *main stack* $\sigma \in \mathbb{N}^*$ and an *auxiliary stack* $\tau \in \mathbb{N}^*$, each containing vertex indices. The tuple (σ, τ, β) of these three structures forms the core of the configurations between which the transition system moves. As an *input*, the system takes a sequence of actions that tell how to build a rope decomposition in an incremental manner. The possible types of actions of the transition system are listed in Table 1.

Initially, both the stacks are empty and the buffer β consists of the list of positive integers. The final configurations of the system consists of all those configurations $(\epsilon, \epsilon, \beta)$ where both stacks are empty, and β contains a suffix $[n, n+1, ...]$ of the list of positive integers. When the system reaches a final configuration, it has produced a relational structure $(V_n, R, \underset{<}{A}, \underset{>}{A}, \underset{<}{I}, \underset{>}{I})$ of a rope decomposition. By doing so, the transition system maps the input sequence of actions to a rope decomposition that represents a digraph. It is not too difficult to define the inverse of this function, but we suppress the details in the interest of space.

4.1 Main Actions

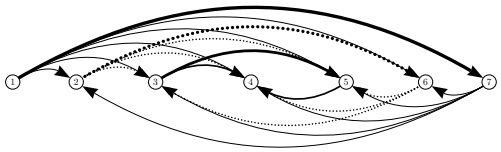
The most important actions of the transition system create the set of edges in the PRC R. By a *shift* (SH) action, the system removes a

Table 1: Transition system for rope decompositions.

action	transition between configurations		effect on the decomposition
SH	$\sigma, \epsilon, i\beta, [\_, \_, \_]$	$\Rightarrow \sigma i, \epsilon, \beta, [-\text{INS}, -\text{RE}, -\underset{0}{\text{PASS}}]$	
NX	$\sigma, \epsilon, i\beta, [\_, \_, \_]$	$\Rightarrow \sigma, \epsilon, \beta, [-\text{INS}, -\text{RE}, -\underset{0}{\text{PASS}}]$	
$\underset{d}{\text{RE}}$	$\sigma i, \epsilon, j\beta, [-\text{INS}, -\text{RE}, \_]$	$\Rightarrow \sigma, \epsilon, j\beta, [-\text{INS}, +\text{RE}, -\underset{0}{\text{PASS}}]$	$R \overset{\cup}{\leftarrow} \{(i,j)\}, \underset{d}{A} \overset{\cup}{\leftarrow} \{(i,j)\}$
$\underset{0}{\text{PASS}}$	$\sigma i, \tau, j\beta, [-\text{INS}, \alpha\ \text{RE}, \_]$	$\Rightarrow \sigma, i\tau, j\beta, [-\text{INS}, \alpha\ \text{RE}, +\underset{0}{\text{PASS}}]$	
$\underset{d}{\text{PASS}}$	$\sigma i, \tau, j\beta, [-\text{INS}, \_, \_]$	$\Rightarrow \sigma, i\tau, j\beta, [-\text{INS}, +\text{RE}, -\underset{0}{\text{PASS}}]$	$\underset{d}{A} \overset{\cup}{\leftarrow} \{(i,j)\}$
$\underset{0}{\text{INS}}$	$\sigma, i\tau, j\beta, [\_, \_, -\underset{0}{\text{PASS}}]$	$\Rightarrow \sigma i, \tau, j\beta, [+\text{INS}, +\text{RE}, -\underset{0}{\text{PASS}}]$	
$\underset{d}{\text{INS}}$	$\sigma, i\tau, j\beta, [\_, \_, \_]$	$\Rightarrow \sigma i, \tau, j\beta, [+\text{INS}, +\text{RE}, -\underset{0}{\text{PASS}}]$	$\underset{d}{I} \overset{\cup}{\leftarrow} \{(j,i)\}$

vertex index i from the front of the buffer and places it to the top of the stack in order to prepare for a future situation where the index i is the left index of an edge in the PRC, i.e., $\exists j.R(i,j)$. When the index j becomes available in the front of the buffer, the system creates the edge $(i,j) \in R$ by a *reduce* (RE) action and removes the index i from the stack. The specifier $d \in \{<, >, <>\}$ of the action tells whether the corresponding arc (i,j) is to be added to $\underset{<}{A}$, $\underset{>}{A}$ or both. Only one *reduce* action is allowed in a row. By a *next* (NX) action, the system removes an index i from the front of the buffer to secure the situation where no covering edge has vertex i as its left index, i.e., $\neg\exists j.R(i,j)$.

Example (7) is a digraph whose underlying graph is a complete graph with an edge between every pair of vertices. The PRC of this graph is $\{(1,7),(2,6),(3,5)\}$.



(7)

The PRC R, and the corresponding arcs in $\underset{<}{A}$ and $\underset{>}{A}$ of the rope decomposition of this digraph are created by the action sequence:

$$\begin{aligned}(\epsilon,\epsilon,[1..]) &\overset{\text{SH}}{\Rightarrow} ([1],\epsilon,[2..]) \overset{\text{SH}}{\Rightarrow} ([1,2],\epsilon,[3..]) \overset{\text{SH}}{\Rightarrow} \\ ([1..3],\epsilon,[4..]) &\overset{\text{NX}}{\Rightarrow} ([1..3],\epsilon,[5..]) \overset{\underset{>}{\text{RE}}}{\Rightarrow} ([1,2],\epsilon,[5..]) \overset{\text{NX}}{\Rightarrow} \\ ([1,2],\epsilon,[6..]) &\overset{\underset{>}{\text{RE}}}{\Rightarrow} ([1],\epsilon,[6..]) \overset{\text{NX}}{\Rightarrow} ([1],\epsilon,[7..]) \overset{\underset{>}{\text{RE}}}{\Rightarrow} (\epsilon,\epsilon,[7..])\end{aligned} \tag{8}$$

The main actions involved in this example do not use the auxiliary stack. The edges in R and the corresponding arcs are created by the reduce actions as follows:

configuration	action	effect
$([1..3],\epsilon,[5..])$	$\underset{>}{\text{RE}}$	$R \overset{\cup}{\leftarrow} \{(3,5)\}, \underset{>}{A} \overset{\cup}{\leftarrow} \{(3,5)\}$
$([1,2],\epsilon,[6..])$	$\underset{>}{\text{RE}}$	$R \overset{\cup}{\leftarrow} \{(2,6)\}, \underset{>}{A} \overset{\cup}{\leftarrow} \{(2,6)\}$
$([1],\epsilon,[7..])$	$\underset{>}{\text{RE}}$	$R \overset{\cup}{\leftarrow} \{(1,7)\}, \underset{>}{A} \overset{\cup}{\leftarrow} \{(1,7)\}$

(9)

4.2 Intermediate Actions

In order to create arcs whose underlying edges do not belong to the PRC, some additional, intermediate actions are needed. Such intermediate actions allow the front of the buffer to form arcs with non-top elements of the main stack.

For instance, the first visit to configuration $([1..3],\epsilon,[4..])$ allows actions that move the elements of the main stack temporarily to the auxiliary stack before restoring the original configuration:

$$\begin{aligned}\ldots \Rightarrow ([1..3],\epsilon,[4..]) &\overset{\underset{>}{\text{PASS}}}{\Rightarrow} ([1,2],[3],[4..]) \overset{\underset{>}{\text{PASS}}}{\Rightarrow} ([1],[2,3],[4..]) \\ \overset{\underset{>}{\text{PASS}}}{\Rightarrow} (\epsilon,[1..3],[4..]) &\overset{\underset{>}{\text{INS}}}{\Rightarrow} ([1],[2,3],[4..]) \overset{\underset{>}{\text{INS}}}{\Rightarrow} ([1,2],[3],[4..]) \\ &\overset{\underset{>}{\text{INS}}}{\Rightarrow} ([1..3],\epsilon,[4..]) \Rightarrow \ldots\end{aligned} \tag{10}$$

The *pass* (PASS) and *insert* (INS) actions create arcs whose underlying edges do not belong to the PRC:

configuration	action	effect
$([1..3],\epsilon,[4..])$	$\underset{>}{\text{PASS}}$	$\underset{>}{A} \overset{\cup}{\leftarrow} \{(3,4)\}$
$([1,2],[3],[4..])$	$\underset{>}{\text{PASS}}$	$\underset{>}{A} \overset{\cup}{\leftarrow} \{(2,4)\}$
$([1],[2,3],[4..])$	$\underset{>}{\text{PASS}}$	$\underset{>}{A} \overset{\cup}{\leftarrow} \{(1,4)\}$
$(\epsilon,[1..3],[4..])$	$\underset{>}{\text{INS}}$	$\underset{>}{I} \overset{\cup}{\leftarrow} \{(4,1)\}$
$([1],[2,3],[4..])$	$\underset{>}{\text{INS}}$	$\underset{>}{I} \overset{\cup}{\leftarrow} \{(4,2)\}$
$([1,2],[3],[4..])$	$\underset{>}{\text{INS}}$	$\underset{>}{I} \overset{\cup}{\leftarrow} \{(4,3)\}$

(11)

To prevent multiple re-entry to the same configuration and repeating the intermediate actions, the detailed configurations of the transition system include control variables that restrict the available actions in different phases of the transition system. As the *insert* actions set +INS, the *reduce* and *pass* astions become blocked until the next *shift*/*next* action.

Some intermediate actions do not create arcs. These intermediate actions have an important role in allowing other actions to access

the non-top elements of the main stack. Yet the sequence $\underset{0}{\text{PASS}}\ \underset{0}{\text{INS}}$ is blocked by the control variable $\underset{0}{\text{PASS}}$ as a useless sequence.

A *reduce* action can be immediately preceded by several $\underset{0}{\text{PASS}}$ actions. Example (6) has a PRC $\{(1,3),(2,4)\}$ that is created through a sequence that involves intermediate actions. The preceding *pass* and the following *insert* actions allow the first *reduce* action to get access to a non-top element of the main stack and to remove it from the stack:

$$\begin{array}{c}(\epsilon,\epsilon,[1..]) \overset{\text{SH}}{\Rightarrow} ([1],\epsilon,[2..]) \overset{\text{SH}}{\Rightarrow} ([1,2],\epsilon,[3..]) \overset{\underset{0}{\text{PASS}}}{\Rightarrow} \\ ([1],[2],[3..]) \overset{\underset{>}{\text{RE}}}{\Rightarrow} (\epsilon,[2],[3..]) \overset{\underset{0}{\text{INS}}}{\Rightarrow} ([2],\epsilon,[3..]) \overset{\text{NX}}{\Rightarrow} \\ ([2],\epsilon,[4..]) \overset{\underset{>}{\text{RE}}}{\Rightarrow} (\epsilon,\epsilon,[4..])\end{array} \tag{12}$$

The configuration $([1],\epsilon,[2..])$ allows a combination of a *pass* and *insert* actions. The *insert* action puts, to $\underset{<}{I}$, the pair $(2,1)$ representing the arc $(3,2)$ in Example (6).

$$\ldots \overset{\text{SH}}{\Rightarrow} ([1],\epsilon,[2..]) \overset{\underset{0}{\text{PASS}}}{\Rightarrow} (\epsilon,[1],[2..]) \overset{\underset{<}{\text{INS}}}{\Rightarrow} ([1],\epsilon,[2..]) \overset{\text{SH}}{\Rightarrow} \ldots \tag{13}$$

4.3 Correctness

Lemma 4.1. *For every digraph, there is a unique action sequence that creates the edges of its PRC.*

Proof sketch. By Proposition 3.1, no vertex needs to start or finish more than one edge in the PRC. The transition system allows to start one covering edge per vertex with a *shift* action and finish any previously started cover edge with a *reduce* action. A crossing cover edge can be finished by accessing the non-topmost stack elements with $\underset{0}{\text{PASS}}$ and $\underset{0}{\text{INS}}$ actions in their appropriate time.

Theorem 4.2. *The transition system is able to produce every possible rope decomposition, capturing every digraph.*

Proof sketch. By Lemma 4.1, we have a way to create the PRC and the corresponding arcs. The set of arcs is extended to represent the rest of the arcs via *pass* and *insert* actions that create the arcs that are properly covered by the edges in the PRC.

5 Linear Encoding

The action sequences of the transition system can be seen as linearisations for the digraphs. In particular, the undirected graph in Example (6) is encoded as the action sequence in (14). To make the linearisation more convenient for eyes, we replace actions with brackets and other symbols in (15).

$$\underset{}{\text{SH}}\ \underset{0}{\text{PASS}}\ \underset{<>}{\text{INS}}\ \underset{}{\text{SH}}\ \underset{0}{\text{PASS}}\ \underset{<>}{\text{RE}}\ \underset{0}{\text{INS}}\ \underset{}{\text{NX}}\ \underset{<>}{\text{RE}} \tag{14}$$

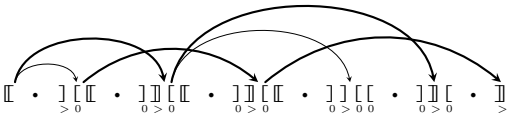
$$[\![\ \bullet\ ]\!]_{0}\ \ [_{<>}\ \ [\![\ \bullet\ ]\!]_{0}\ \ ]\!]_{<>}\ \ [_{0}\ \ \bullet\ \ ]\!]_{<>} \tag{15}$$

Convention 5.1. By convention, the bracketing scheme renames the actions of the transition system as follows:

NX	SH	PASS $\underset{0}{}$	PASS $\underset{d}{}$	INS $\underset{0}{}$	INS $\underset{d}{}$	RE $\underset{d}{}$
$\bullet$	$[\![\ \bullet$	$]_{0}$	$]_{d}$	$[_{0}$	$[_{d}$	$]\!]_{d}$

(16)

The convention improves the readability of action sequence and gives compact action sequences: especially, the digraph in Example (7) is encoded as string $[\![\ \bullet\ ]_{>}\ [_{<}\ [\![\ \bullet\ ]_{>}\ ]_{>}\ ]_{<}\ [_{<}\ [_{<}\ [\![\ \bullet\ ]_{>}\ ]_{>}\ ]_{>}\ ]_{<}\ [_{<}\ [_{<}\ [\![\ \bullet\ ]_{>}\ ]_{>}\ ]_{>}\ ]_{<}\ [_{<}\ [\![\ \bullet\ ]_{>}$. Example (17) demonstrates how the bracket(s) now correspond almost iconically to the represented arcs.



(17)

Convention 5.1 benefits us when we analyse the formal, language theoretic properties of such action sequences altough the encoding is not otherwise meant for human inspection and its direct manipulation by hand is prone to errors. The convention borrows ideas from Yli-Jyrä (2017) but differs from it in four important aspects:

1. The vertices are separated with a bullet symbol ($\bullet$) instead of curly brackets ({}).
2. The left brackets $\{[\![, [_{}, [_{0}\}$ match the right brackets $\{]\!],]_{},]_{0}\}$.
3. Weak brackets may be stacked (like in $]_{}]_{}]_{}[_{}[_{}[_{}$) in order to deal with stacked pass and insert actions.
4. A strong closing bracket may occur inside a stack of weak brackets (like in $]_{0}]_{0}]\!][_{}[_{}$).

These changes are necessary to deal with crossing arcs and the actions that operate on two stacks.

5.1 Syntactic Properties

In order to analyse the formal properties of the transition system, we need to understand how the actions, or the corresponding brackets, form strings that correspond to ordered graphs in a bijective manner. We induce the following principles:

1. Each vertex corresponds to a sequence of closing brackets followed by a sequence of opening brackets.
2. Between "$\rrbracket$" and the closest preceding "$\bullet$", there can be only "$\underset{0}{]}$"-brackets. This corresponds to the convention 3.7 that always chooses the "EARLIEST" covering edge in the PRC of the underlying graph. We do not need "]"-brackets to produce arcs as we have an opportunity to produce same arcs with "["-brackets earlier.
3. The unnecessary pair of pass and insert actions, marked with bracket substring "$\underset{0}{]}\,\underset{0}{[}$" do not occur in the strings.
4. The strong opening bracket "$\llbracket$" always occurs right before the vertex boundary "$\bullet$".
5. The maximum number of brackets per vertex is $n - 1$.
6. Left brackets $B_L = \{\llbracket, \underset{0}{[}, \underset{d}{[}\}$ match right brackets $B_R = \{\underset{d}{\rrbracket}, \underset{0}{]}, \underset{d}{]}\}$.
7. The rope thickness of the (di)graph is the maximum number of momentarily open brackets in its encoding.

According to the principles 1-4, the bracket substrings that correspond to different vertices constitute a context-free language W that is generated by the grammar G_W:

$$\begin{array}{llll} S \to T^? \mid T^? \llbracket & T^? \to T \mid \epsilon \\ R' \to \underset{<}{\rrbracket} \mid \underset{>}{\rrbracket} \mid \underset{<>}{\rrbracket} & L' \to L \mid \underset{0}{[} \\ R \to \underset{<}{]} \mid \underset{>}{]} \mid \underset{<>}{]} & L \to \underset{<}{[} \mid \underset{>}{[} \mid \underset{<>}{[} \\ T \to R'\, T^? \mid R\, T^?\, L' \mid \underset{0}{]}\, T^?\, L \mid \underset{0}{]}\, T\, \underset{0}{[} \end{array} \tag{18}$$

Lemma 5.1. *The action sequences that conform to the seven principles allow only one way to represent each ordered digraph.*

Proof. The strong brackets are crucial for encoding all arcs of the digraph and the PRC of its underlying graph in particular. Every digraph (V_n, A) has a unique PRC, and, due to the principles 1-4, it is not possible to build the same PRC with the correct arc orientations in two different ways. By the second principle, all non-PRC arcs are assigned to a unique covering edge. There is thus only one moment when the right combination of indices is available in the configuration for constructing each arc, and there is only one action sequence that can construct any given digraph. □

We also observe that string concatenation of two action sequences gives an action sequence that produces a digraph concatenation of two ordered digraphs with one shared vertex.

Proposition 5.2. *The encoding from digraphs to strings is a mapping that preserves the structure of the digraph concatenation monoid and sends it the structure of a string concatenation monoid.*

6 Formal Language Theory

Chomsky-Schützenberger (CS) parsing (Yli-Jyrä, 2005, 2012; Hulden, 2009; Yli-Jyrä and Gómez-Rodríguez, 2017; Ruprecht and Denkinger, 2019) combines a particular kind of language representations with weighted automata techniques. A prototypical CS style language representation $h(\mathcal{L} \cap \mathcal{D})$ involves a homomorphic mapping (h) applied to an intersection of a a regular language component $\mathcal{L}$ and a Dyck language $\mathcal{D}$. Yli-Jyrä and Gómez-Rodríguez (2017) used this kind of language representations to show that their encoding for the noncrossing digraphs ($L_{\text{NC-DIGRAPH}}$) is a context-free language and admits an efficient algorithm for finding maximal constrained subdigraphs in a weighted complete digraph.

This section gives a CS style representation for the language of all encoded digraphs (L_{DIGRAPH}) by relaxing the requirement that the $\mathcal{L}$ component of the language representation is a regular language. The represented language is then not context-free, but the representation is *loosely speaking* of "the CS style". The similarity becomes more obvious when we derive a context-free approximation of it.

Lemma 6.1. *There is a CS style representation for the language* L_{DIGRAPH}.

Proof. We start by defining a Dyck language $\mathcal{D}$ that checks for balanced bracketing. Let the internal alphabet of the representation be $\Sigma = \{\text{\{}, \text{\}}\} \cup \{ [_{(l,r)},]_{(l,r)} \mid l \in B_L, r \in B_R\}$. Let $\mathcal{D}$ be the language generated by the grammar

$$S \to \epsilon \mid SS \mid \text{\{}S\text{\}} \mid [_{(l,r)} S]_{(l,r)} \tag{19}$$

where $l \in B_L$ and $r \in B_R$. Let $h : \Sigma^* \to (\{ \bullet \} \cup B_L \cup B_R)^*$ be a homomorphism defined in such a way that, for all $l \in B_L, r \in B_R$,

$$\begin{aligned} h(ab) &= h(a)h(b) & h(\epsilon) &= \epsilon \\ h([_{(l,r)}) &= l & h(\text{\{}) &= \epsilon \\ h(]_{(l,r)}) &= r & h(\text{\}}) &= \bullet . \end{aligned} \tag{20}$$

Instead of the usual regular component of CS representations, we use a marked concatenation closure of a context-free language: let $\mathcal{L}$ be the context-free language $W(\bullet W)^{n-1}$, whose inverse homomorphims $h^{-1}(\mathcal{L})$ is also a marked concatenation closure of a context-free language.

The set of encoded digraphs is now given as $L_{\text{DIGRAPH}} = h(h^{-1}(\mathcal{L}) \cap \mathcal{D}) = \mathcal{L} \cap h(\mathcal{D})$. □

Lemma 6.2. *The subset of the encoded digraphs* L_{DIGRAPH}*, where the number of* brackets per vertex *is bounded by* k*, is context-free.*

Proof. We start from the CS style representation (the proof of lemma 6.1) for L_{DIGRAPH} and replace the context-free language $\mathcal{L}$, with a regular approximation $\mathcal{L}_{<k} = W_{<k}(\bullet W_{<k})^*$ where $W_{<k}$ is a finite subset of W restricted to contain at most k nested brackets in the strings. This gives a more prototypical CS represention $L_{\text{DIGRAPH},k} = h(h^{-1}(\mathcal{L}_{<k}) \cap D) = \mathcal{L}_{<k} \cap h(\mathcal{D})$, which yields a context-free subset of L_{DIGRAPH}. □

Lemma 6.3. *The subset of encoded graphs* L_{DIGRAPH}*, where the rope thickness of the encoded digraphs is bounded by* t*, is regular.*

Proof. As the rope thickness is bounded by t, the number of brackets per vertex is bounded by $2t$. Thus, we start from the CS style representation (the proof of Lemma 6.2) for encoded graphs where the number of brackets per vertex is bounded. By the bound t for the rope thickness, we replace the context-free language $\mathcal{D}$ with a regular subset $\mathcal{D}_t \subset \mathcal{D}$ that can contain t levels of nested brackets. The $2t, t$-bounded set of encoded graphs is given by $L_{\text{DIGRAPH},2t,t} = h(h^{-1}(\mathcal{L}_{<2t}) \cap \mathcal{D}_t) = \mathcal{L}_{<2t} \cap h(\mathcal{D}_t)$. By the closure properties of regular languages, $L_{\text{DIGRAPH},2t,t}$ is regular. □

7 Evidence for Linguistic Relevance

To assess the linguistic relevance of the currently presented encoding, we carried out a small experiment where we computed the rope-thickness of dependency trees in the Universal Dependencies v 2.4 treebanks (Nivre et al., 2019). The compacted results are presented in Table 2. The results indicate that a very high proportion of the observations is captured when rope-thickness is 4 or higher.

According to our preliminary experiments on graph banks, a very similar distribution of rope-thickness is observed in more general annotation graphs.

8 Discussion

Among the earliest encoding schemes for graphs are the Prüfer sequences for labeled trees (Prüfer, 1918) that have been extened to DAGs (Steinsky, 2003). More recently, Turán (1984) introduced the problem of *graph representation* given an adjacency matrix. There are now some efficient representations for unlabled and labeled graphs (Turán, 1984; Naor, 1990; Farzan and Munro, 2013). Our representation for digraphs is also *efficient*: it has a cubic-time encoder and a linear-time decoder.

The currently presented encoding for digraphs is a generalisation of an earlier representation (Yli-Jyrä, 2017, 2019) that is itself an optimized alternative for the balanced bracketing proposed for weighted dependency parsing in (Yli-Jyrä, 2012). Several edge-weighted parsing algorithms have been presented earlier (Dixon et al., 1992; Charniak et al., 1998; Sasano et al., 2000; Kuhlmann and Jonsson, 2015), but these newer methods apply to up to 50 families of dependency graphs and the currently presented encoding is expected to help in their generalization. It would be also interesting to study how rope graphs relate to 1-endpoint crossing graphs (Pitler et al., 2013; Kurtz and Kuhlmann, 2017).

language	trees	rope-thickness 1	2	3	4	5	6	7	8
Arabic	28,402	4.9%	20.5%	63.5%	94.0%	99.64%	99.986%	100.000%	100.000%
Czech	127,507	13.3%	48.3%	89.3%	98.8%	99.91%	99.992%	100.000%	100.000%
Dutch	20,735	21.3%	63.4%	92.9%	98.9%	99.92%	99.986%	100.000%	100.000%
English	34,601	15.6%	54.9%	92.3%	99.3%	99.98%	99.997%	100.000%	100.000%
Finnish	34,641	37.0%	78.7%	96.8%	99.6%	99.97%	99.994%	100.000%	100.000%
German	191,757	10.1%	49.2%	90.5%	99.2%	99.96%	99.997%	100.000%	100.000%
Hindi	19,545	2.9%	44.4%	85.3%	98.3%	99.87%	99.980%	100.000%	100.000%
Italian	34,057	5.6%	49.9%	90.3%	98.8%	99.89%	99.997%	100.000%	100.000%
Korean	34,702	15.5%	65.6%	94.7%	99.5%	99.95%	99.994%	100.000%	100.000%
Latvian	12,558	12.7%	50.0%	89.8%	98.8%	99.90%	99.992%	99.992%	100.000%
Russian	87,377	15.0%	55.5%	90.7%	98.9%	99.91%	99.994%	99.999%	100.000%
Chinese	18,628	46.1%	73.7%	91.9%	98.6%	99.89%	99.989%	100.000%	100.000%
all 83 languages	1,232,262	15.1%	54.8%	90.5%	99.0%	99.93%	99.995%	100.000%	100.000%

Table 2: The cumulative coverage of bounded rope-thickness in the UD v2.4 dataset from which we purged the trees containing ellipsis.

The languages of encoded graphs have applications to constrained graph enumeration problems. Hoppe and Petrone (Hoppe and Petrone, 2016) have exhaustively enumerated all simple, connected graphs of a finite order and computed a selection of invariants over the sets in order to discover and add 141 new integer sequences to the Online Encyclopedia of Integer Sequences (OEIS). Our previous encoding scheme (Yli-Jyrä and Gómez-Rodríguez, 2017) gave context-free characterisations for some graph properties. This led to the discovery of dozens of known and new integer sequences by graph enumeration. These new computational methods complement the research that spans from the "Abzählsatz" of (Pólya, 1937) to more recent work on graph enumeration (Wormald, 1979; Mckay, 1983; Kapoor and Ramesh, 2000; Acuña et al., 2012; Conte et al., 2018; Equi et al., 2019).

The language L_{DIGRAPH} is not only context-sensitive but even an indexed language (Aho, 1968): it is possible to construct an indexed grammar that generates the same set of strings. However, the existence of a simple transition system, a finite representation, and a finite indexed grammar for the encoded digraphs should not be confused with condition under which digraphs themselves become finitely generated. Ogawa (2004) has presented a complete, infinite set of generators for the graphs. We also need an infinite set of generators for the language L_{DIGRAPH}, because the the paths that take the transition system from one final configuration to another final configuration constitute an infinite set of code words over which the encoded digraphs are generated. This set remains infinite even for digraphs with bounded rope thickness, but the context-free and regular subsets of L_{DIGRAPH} may have some other ways to motivate finite algebraic axiomatisations.

9 Conclusion

This paper contributes to the research on graph representations (Turán, 1984) by developing a linear-time decodable encoding for arbitrary labeled digraphs that we preferred to call ordered digraphs. The particular design of our linear encoding is motivated by the success of similar representations (Yli-Jyrä, 2005, 2012; Yli-Jyrä and Gómez-Rodríguez, 2017) in the characterisations of several families of noncrossing digraphs and by the effectiveness of the recently improved representation (Yli-Jyrä, 2017, 2019). Evidently, both kinds of graph representations have potential applications in graph enumeration (Yli-Jyrä and Gómez-Rodríguez, 2017) and weighted Chomsky-Schützenberger parsing (Yli-Jyrä, 2005, 2012; Hulden, 2009; Yli-Jyrä and Gómez-Rodríguez, 2017; Ruprecht and Denkinger, 2019).

More specifically, the paper contributes a general transition system that decodes arbitrary digraphs from linear action sequences. Crucial notions – the proper rope cover (PRC) and the related rope decomposition – are defined and used in this transition system. The first PRC-based measure for the complexity of the graphs is introduced. Context-free and regular approximations of the encoded graphs are defined and shown to contain the dependency annotations of the UD 2.4 treebanks.

References

Vicente Acuña, Etienne Birmelé, Ludovic Cottret, Pierluigi Crescenzi, Fabien Jourdan, Vincent Lacroix, Alberto Marchetti-Spaccamela, Andrea Marino, Paulo Vieira Milreu, Marie-France Sagot, and Leen Stougie. 2012. Telling stories: Enumerating maximal directed acyclic graphs with a constrained set of sources and targets. *Theoretical Computer Science*, 457:1 – 9.

Alfred V. Aho. 1968. Indexed grammars – an extension of context-free grammars. *Journal of the ACM*, 15(4):647–671.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 166–170. Association for Computational Linguistics.

Jill Burstein, Christy Doran, and Thamar Solorio, editors. 2019. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics.

Eugene Charniak, Sharon Goldwater, and Mark Johnson. 1998. Edge-based best-first chart parsing. In *Sixth Workshop on Very Large Corpora, VLC@COLING/ACL 1998, Montreal, Quebec, Canada, August 15-16, 1998*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1052–1062, Sofia, Bulgaria. Association for Computational Linguistics.

Maximin Coavoux and Shay B. Cohen. 2019. Discontinuous constituency parsing with a stack-free transition system and a dynamic oracle. *CoRR*, abs/1904.00615.

Alessio Conte, Roberto Grossi, Andrea Marino, and Romeo Rizzi. 2018. Efficient enumeration of graph orientations with sources. *Discrete Applied Mathematics*, 246:22 – 37.

Alessio Conte, Roberto Grossi, Andrea Marino, and Luca Versari. 2019. Listing maximal subgraphs satisfying strongly accessible properties. *SIAM Journal on Discrete Mathematics*, 33(2):587–613.

Bruno Courcelle. 1990. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75.

Michael A. Covington. 2000. A fundamental algorithm for dependency parsing. In *In Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102, Athens, GA. Association for Computing Machinery.

Brandon Dixon, Monika Rauch, and Robert Endre Tarjan. 1992. Verification and sensitivity analysis of minimum spanning trees in linear time. *SIAM Journal on Computing*, 21(6):1184–1192.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Massimo Equi, Roberto Grossi, Veli Mäkinen, and Alexandru I. Tomescu. 2019. On the complexity of string matching for graphs. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, volume 132 of *LIPIcs*, pages 55:1–55:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Arash Farzan and J. Ian Munro. 2013. Succinct encoding of arbitrary graphs. *Theoretical Computer Science*, 513:38 – 52.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2018. Non-projective dependency parsing with non-local transitions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 693–700, New Orleans, Louisiana. Association for Computational Linguistics.

Daniel Gildea, Giorgio Satta, and Xiaochang Peng. 2018. Cache transition systems for graph parsing. *Computational Linguistics*, 44(1):85–118.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 742–750. Association for Computational Linguistics.

Dina Q. Goldin, Scott A. Smolka, Paul C. Attie, and Elaine L. Sonderegger. 2004. Turing machines, transition systems, and interaction. *Information and Computation*, 194(2):101 – 128. Special Issue Commemorating the 50th Birthday Anniversary of Paris C. Kanellakis.

Carlos Gómez-Rodríguez and Joakim Nivre. 2013. Divisible transition systems and multiplanar dependency parsing. *Computational Linguistics*, 39(4):799–845.

Carlos Gómez-Rodríguez, Tianze Shi, and Lillian Lee. 2018. Global transition-based non-projective dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2664–2675, Melbourne, Australia. Association for Computational Linguistics.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1314–1324. Association for Computational Linguistics.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Travis Hoppe and Anna Petrone. 2016. Integer sequence discovery from small graphs. *Discrete Appl. Math.*, 201(C):172–181.

Mans Hulden. 2009. Parsing CFGs and PCFGs with a Chomsky-Schützenberger representation. In *Human Language Technology. Challenges for Computer Science and Linguistics - 4th Language and Technology Conference, LTC 2009, Poznan, Poland, November 6-8, 2009, Revised Selected Papers*, volume 6562 of *Lecture Notes in Computer Science*, pages 151–160. Springer.

S. Kapoor and H. Ramesh. 2000. An algorithm for enumerating all spanning trees of a directed graph. *Algorithmica*, 27(2):120–130.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

András Kornai and Zsolt Tuza. 1992. Narrowness, pathwidth, and their application in natural language processing. *Discrete Applied Mathematics*, 36(1):87–92.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 673–682. Association for Computational Linguistics.

Marco Kuhlmann and Peter Jonsson. 2015. Parsing to noncrossing dependency graphs. *TACL*, 3:559–570.

Robin Kurtz and Marco Kuhlmann. 2017. Exploiting structure in parsing to 1-endpoint-crossing graphs. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 78–87, Pisa, Italy. Association for Computational Linguistics.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017. Arc-hybrid non-projective dependency parsing with a static-dynamic oracle. In *Proceedings of the 15th International Conference on Parsing Technologies, IWPT 2017, Pisa, Italy, September 20-22, 2017*, pages 99–104. Association for Computational Linguistics.

Brendan D. Mckay. 1983. Applications of a technique for labelled enumeration. *Congressus Numerantium*, 40:207–221.

Moni Naor. 1990. Succinct representation of general unlabeled graphs. *Discrete Applied Mathematics*, 28(3):303 – 307.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, IncrementParsing '04, pages 50–57. Association for Computational Linguistics.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 351–359. Association for Computational Linguistics.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Gabrielė Aleksandravičiūtė, Lene Antonsen, Katya Aplonova, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, John Bauer, Sandra Bellato, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Agnė Bielinskienė, Rogier Blokland, Victoria Bobicev, Loïc Boizou, Emanuel Borges Völker, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Kristina Brokaitė, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Bamba Dione, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Hanne Eckhoff, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Kazunori Fujita, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta González Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Johannes Heinecke, Felix Hennig, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Takumi Ikeda, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Andre Kaasen, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Arne Köhn, Kamil Kopacewicz, Natalia Kotsyba, Jolanta Kovalevskaitė, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Yuan Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Sarah McGuinness, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Tomohiko Morioka, Shinsuke Mori, Shigeki Moro, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Yoshihiro Nikaido, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Angelika Peljak-Łapińska, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Daria Petrova, Slav Petrov, Jussi Piitulainen, Tommi A Pirinen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Erika Rimkutė, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Alessio Salomoni, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Dage Särg, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Hiroyuki Shirasu, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Shingo Suzuki, Zsolt Szántó, Dima Taji, Yuta Takahashi, Fabio Tamburini, Takaaki Tanaka, Isabelle Tellier, Guillaume Thomas, Liisi Torga, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Abigail Walsh, Jing Xian Wang, Jonathan North Washington, Maximilan Wendt, Seyi Williams, Mats Wirén, Christian Wittern, Tsegay Woldemariam, Taksum Wong, Alina Wróblewska, Mary Yako, Naoki Yamazaki, Chunxiao Yan, Koichi Yasuoka, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2019. Universal Dependencies 2.4. LINDAT/CLARIN digital library at the Institute of Formal and Ap-

plied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *COLING 2004, 20th International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2004, Geneva, Switzerland*.

Mizuhito Ogawa. 2004. Complete axiomatization of an algebraic construction of graphs. In *Functional and Logic Programming*, pages 163–179, Berlin, Heidelberg. Springer Berlin Heidelberg.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2013. Finding optimal 1-endpoint-crossing trees. *TACL*, 1:13–24.

Emily Pitler and Ryan McDonald. 2015. A linear-time transition system for crossing interval trees. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 662–671, Denver, Colorado. Association for Computational Linguistics.

Heinz Prüfer. 1918. Neuer Beweis eines Satzes über Permutationen (New proof of a theorem on permutations). *Archiv der Mathematik und Physik*, 27(3):142–144.

G. Pólya. 1937. Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen. *Acta Math.*, 68:145–254.

Peng Qi and Christopher D. Manning. 2017. Arc-swift: A novel transition system for dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 110–117. Association for Computational Linguistics.

Thomas Ruprecht and Tobias Denkinger. 2019. Implementation of a Chomsky-Schützenberger n-best parser for weighted multiple context-free grammars. In (Burstein et al., 2019), pages 178–191.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of COLING 2008, the 22nd International Conference on Computational Linguistics*, pages 753–760, Manchester, UK. Coling 2008 Organizing Committee.

Isao Sasano, Zhenjiang Hu, Masato Takeichi, and Mizuhito Ogawa. 2000. Make it practical: a generic linear-time algorithm for solving maximum-weightsum problems. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000.*, pages 137–149. Association for Computing Machinery.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 12–23, Copenhagen, Denmark. Association for Computational Linguistics.

B. Steinsky. 2003. Efficient coding of labeled directed acyclic graphs. *Soft Computing*, 7(5):350–356.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. Viable dependency parsing as sequence labeling. In (Burstein et al., 2019), pages 717–723.

Wolfgang Thomas. 2002. A short introduction to infinite automata. In *Developments in Language Theory*, pages 130–144, Berlin, Heidelberg. Springer Berlin Heidelberg.

György Turán. 1984. On the succinct representation of graphs. *Discrete Applied Mathematics*, 8(3):289 – 294.

David Vilares and Carlos Gómez-Rodríguez. 2018. A transition-based algorithm for unrestricted AMR parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 142–149, New Orleans, Louisiana. Association for Computational Linguistics.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2773–2781.

W. A. Woods. 1970. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606.

Nicholas C. Wormald. 1979. Enumeration of labelled graphs II: Cubic graphs with a given connectivity. *Journal of the London Mathematical Society*, s2-20(1):1–7.

H. Yamada and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector machines. In *The 8th International Workshop of Parsing Technologies (IWPT2003)*.

Anssi Yli-Jyrä. 2005. Approximating dependency grammars through intersection of star-free regular languages. *International Journal of Foundations of Computer Science*, 16(3):565–579.

Anssi Yli-Jyrä. 2012. On dependency analysis via contractions and weighted FSTs. In *Shall We Play the Festschrift Game?, Essays on the Occasion of Lauri Carlson's 60th Birthday*, pages 133–158. Springer.

Anssi Yli-Jyrä. 2017. Bounded-depth high-coverage search space for noncrossing parses. In *Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing, FSMNLP 2017, Umeå, Sweden, September 2017*, pages 30–40. Association for Computational Linguistics.

Anssi Yli-Jyrä. 2019. How to embed noncrossing trees in Universal Dependencies treebanks in a low-complexity regular language. *Journal of Language Modelling*. Forthcoming.

Anssi Yli-Jyrä and Carlos Gómez-Rodríguez. 2017. Generic axiomatization of families of noncrossing graphs in dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1745–1755. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA - Short Papers*, pages 188–193. The Association for Computer Linguistics.