Hierarchical Bracketing Encodings for Dependency Parsing as Tagging

Ana Ezquerro

David Vilares

Anssi Yli-Jyrä

Carlos Gómez-Rodríguez

UNIVERSIDADE DA CORUÑA

INDITEX UDC Cátedra de IA en Algoritmos Verdes

citic

Funded by the European Union

erc European Research Council

MINISTERIO DE CIENCIA E INNOVACIÓN

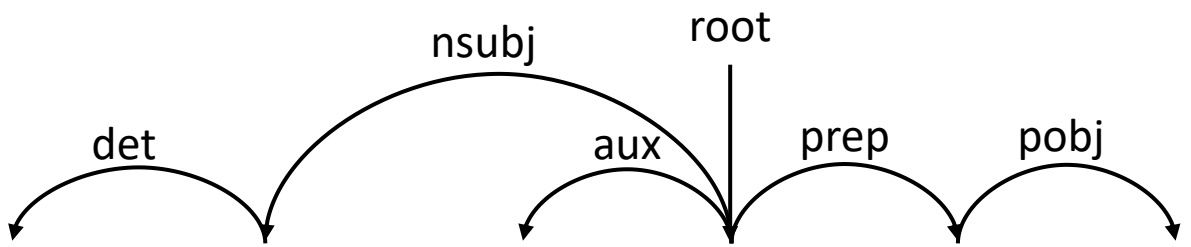
UNIÓN EUROPEA

FONDO EUROPEO DE DESARROLLO REGIONAL

AGENCIA ESTATAL DE INVESTIGACIÓN

Motivation: Dependency Parsing as Sequence Labeling (SL)

SL reframes the dependency parsing task (graph-based prediction) as token-level classification.



	The	ACL-2025	is	located	in	Vienna	**LABEL SPACE:**
Strzyz et al. (2020): *brackets*	<	\<	<	\\/	>/	>	unbounded
Gómez-Rodríguez et al. (2023): *4-bit (and 7-bit)*	0100	0110	0000	1111	1101	1100	bounded (16)
Amini et al. (2023): *hexatag*	↗↗^{R}	↖↗^{R}	↖↗^{R}	↖↗^{L}	↗ ↖^{L}	↖	bounded (8)
Hierarchical (ours):	<	\<	<	>/	>/	>	**bounded (12)**

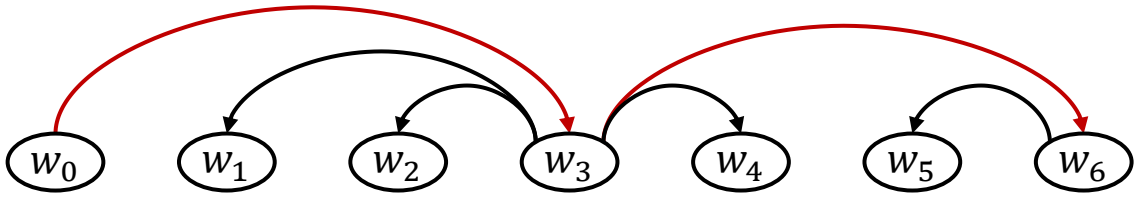
super vs auxiliary brackets

Optimal Hierarchical Bracketing Encodings (OHB)

- Contribution: Novel SL approach for dependency trees (bounded label space).
- Key concepts: (1) *rope cover* (2) *superbrackets* (3) *auxiliary brackets.*

How to encode?
1. Identify the *proper rope cover*.
2. Assign superbrackets (>, \, /, <).
3. Assign auxiliary brackets (>, <).

Projective dependency tree



Bracketing:	<	<	\\>//	>	<	\>
Superbrackets:			>/			>
Auxiliary brackets:	<	<	>/	>	<	>

The head of the auxiliary arcs is paired with a superbracket

Step 1: Given a dependency tree $G = (W, A)$, the *proper rope cover* is $R \subseteq A$ such that:
1. Every $a \in A - R$ leans on some $a' \in R$.
2. No $a \in R$ leans on any other $a' \in R - \{a\}$.

Step 2: Encode R with superbrackets.

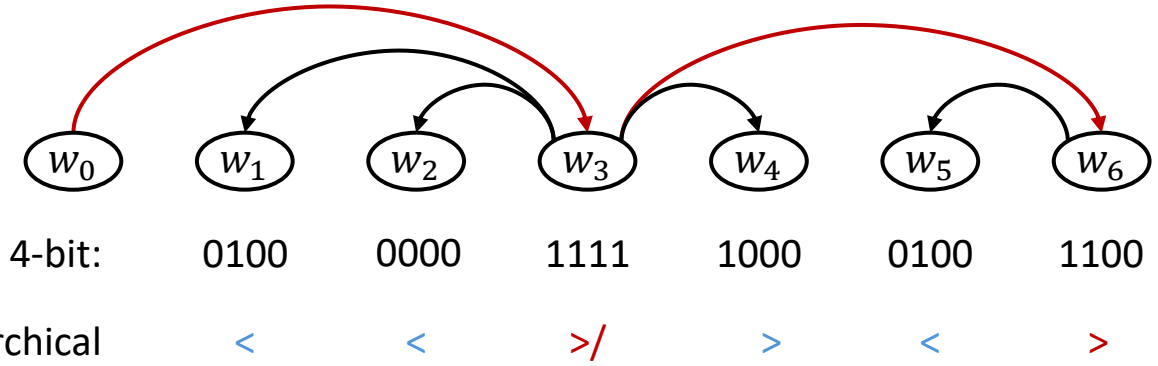
Step 3: Encode $A - R$ with auxiliary brackets.

The Bounded Space of Optimal Hierarchical Brackets (OHB)

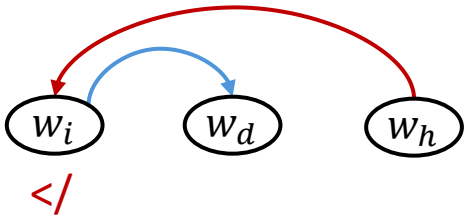
- The rope cover is the set of arcs corresponding to the highest hierarchy level.
- OHB compresses the brackets of sub-dependencies with the superbrackets of the rope cover.
- There are a total of 12 labels. **How?**

Proof from 4-bit encoding (2^4 = 16 labels)

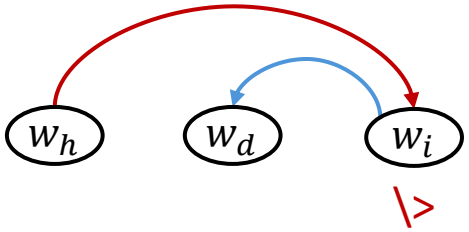
- $b_0 = w_i$ has a left head.
- $b_1 = w_i$ is the outermost dependent.
- $b_2 = w_i$ has left dependents.
- $b_3 = w_i$ has right dependents.



b_0b_1 \ b_2b_3	00	01	10	11
00	<	< /	\ <	\ < /
01	<	< /	\ <	\ < /
10	>	> /	\ >	\ > /
11	>	> /	\ >	\ > /



Never happens since $(w_i \rightarrow w_d)$ is auxiliary.



Never happens since $(w_d \leftarrow w_i)$ is auxiliar.

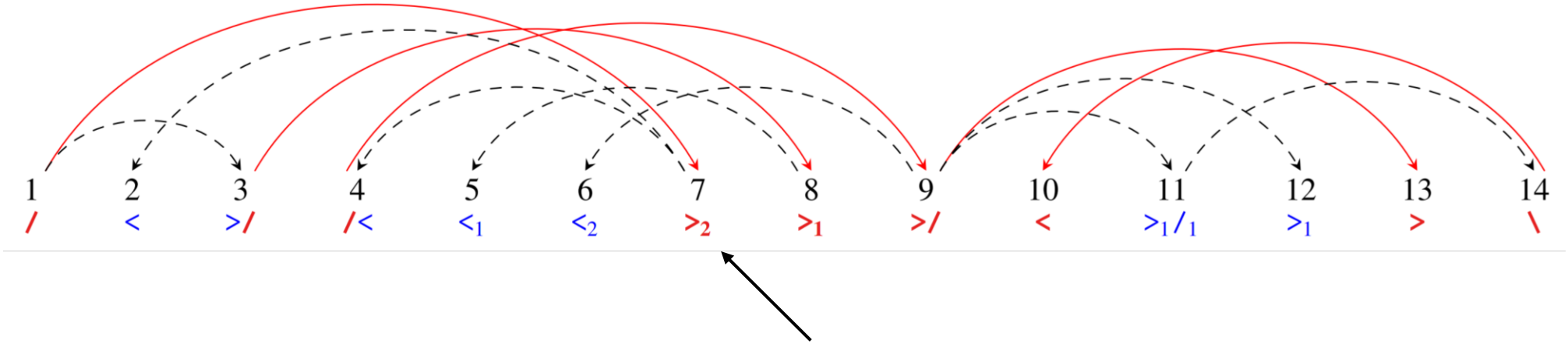
But { \</, \>/, </, \> } **cannot** occur in OHB since no $a \in R$ leans on $a' \in R - \{a\}$.

Expanding Hierarchical Brackets to Non-Projective Dependency Trees

- Hexatagging, 4-bit and HB encodings do not support non-projective trees.

We expand HB with **subindices** to support non-projective trees (unbounded space).

- Only closing superbrackets (>, \) have subindices.
- More auxiliary brackets is expanded (>, \, /, <).



Subindices indicate the number of superbrackets that need to be skipped in the decoding process until matching.

Experiments and Results

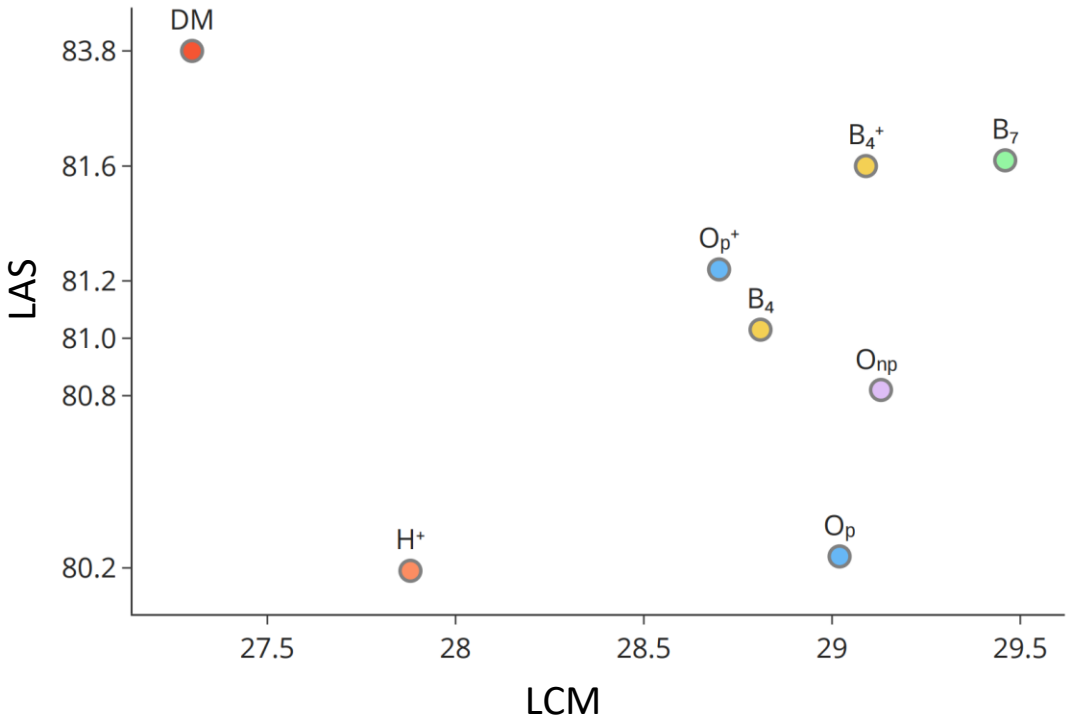
- PTB and UD (9 languages) with XLM✧/XLNet ☐.
- Baselines: Hexatagging (H^+) and Biaffine (DM).
- 4-bit (B_4) and projective OHB (O_p) with pseudo-projectivity (+).
- 7-bit (B_7) and non-projective OHB (O_{np}).

Close performance:

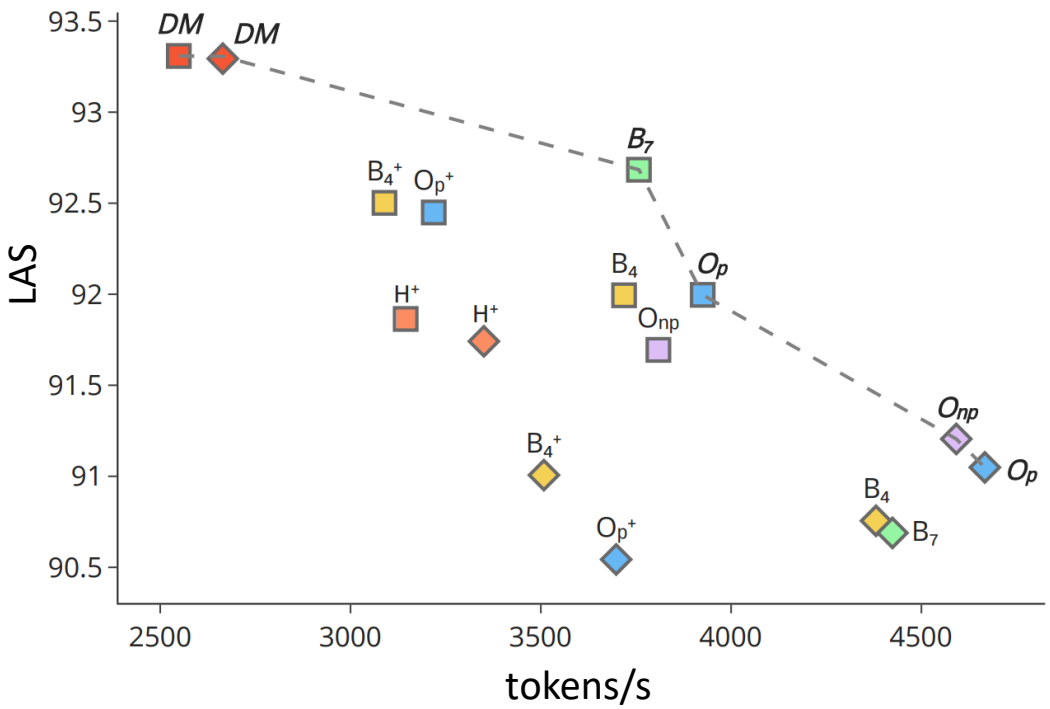
- LAS: $O_p < O_{np} < B_4 < O_p^+ < B_4^+ < B_7$.
- LCM: $O_p^+ < B_4 < O_p < B_4^+ < O_{np} < B_7$.

But O_p/O_{np} are faster (compact label space).

Performance (average)



Pareto Front (English-EWT)



Code and Materials: https://github.com/anaezquerro/separ.

UNIVERSIDADE DA CORUÑA

Hierarchical Bracketing Encodings for Dependency Parsing as Tagging

Thank you for listening!

Ana Ezquerro

David Vilares

Anssi Yli-Jyrä

Carlos Gómez-Rodríguez

We acknowledge grants SCANNER-UDC (PID2020-113230RB-C21) funded by MICIU/AEI/10.13039/501100011033; GAP (PID2022-139308OA-I00) funded by MICIU/AEI/10.13039/501100011033/ and ERDF, EU; LATCHING (PID2023-147129OB-C21) funded by /AEI/10.13039/501100011033 and ERDF, EU; and TSI-100925-2023-1 funded by Ministry for Digital Transformation and Civil Service and ``NextGenerationEU'' PRTR; as well as funding by Xunta de Galicia (ED431C 2024/02), and CITIC, as a center accredited for excellence within the Galician University System and a member of the CIGUS Network, receives subsidies from the Department of Education, Science, Universities, and Vocational Training of the Xunta de Galicia. Additionally, it is co-financed by the EU through the FEDER Galicia 2021-27 operational program (Ref. ED431G 2023/01). We also extend our gratitude to CESGA, the supercomputing center of Galicia, for granting us access to its resources.Furthermore, we acknowledge the Faculty of Agriculture and Forestry of the University of Helsinki, as well as projects "Theory of Computational Logics" (352420) and "XAILOG" (345612, 345633) funded by the Research Council of Finland for the continued support of the third author during the multistage writing process.

Funded by the European Union

European Research Council

INDITEX UDC

Cátedra de IA en Algoritmos Verdes

MINISTERIO DE CIENCIA E INNOVACIÓN

UNIÓN EUROPEA

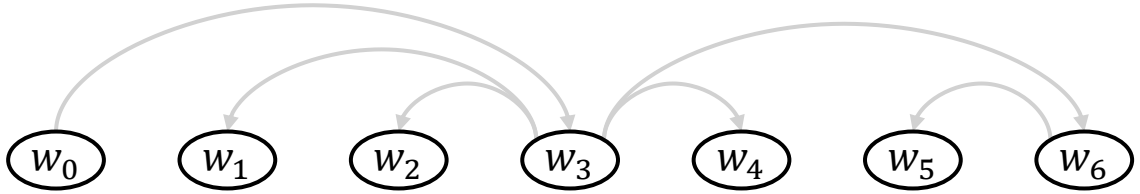
FONDO EUROPEO DE DESARROLLO REGIONAL

AGENCIA ESTATAL DE INVESTIGACIÓN

Decoding with Hierarchical Brackets

Stack-based system where:

- Auxiliary brackets only match superbrackets.
- Superbrackets match any bracket.



Stack

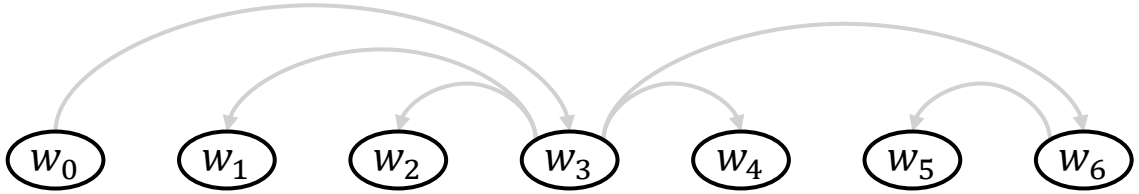
Buffer

(label indices are marked subscripts)

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack

$/_0$

Buffer

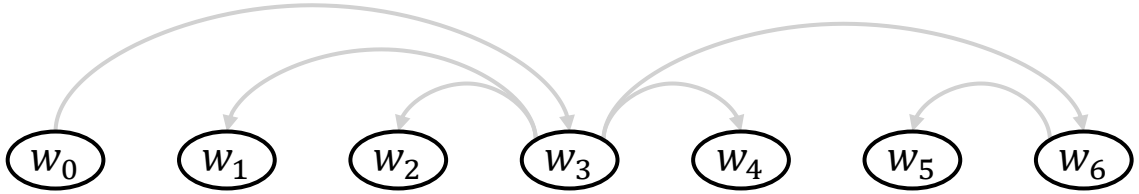
$<_1$ $<_2$ $>_3 /_3$ $>_4$ $<_5$ $>_6$

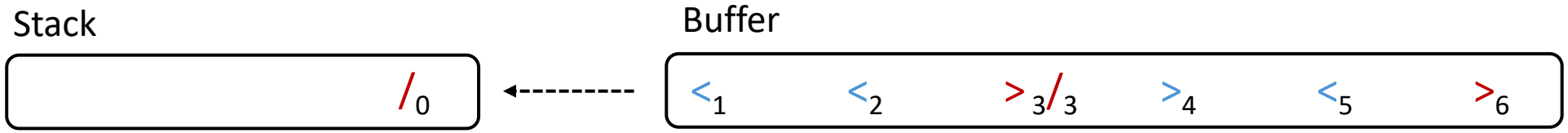
(label indices are marked subscripts)

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



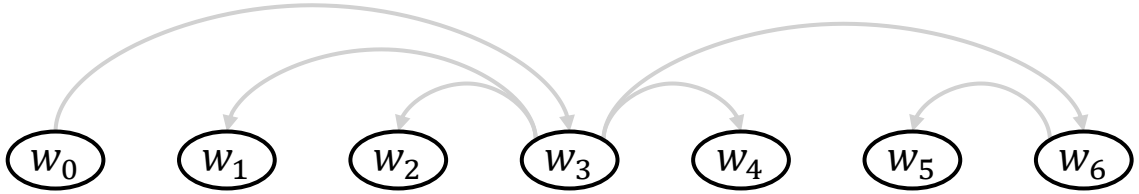


Add $<_1$ to the stack.

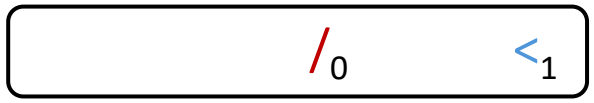
Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack



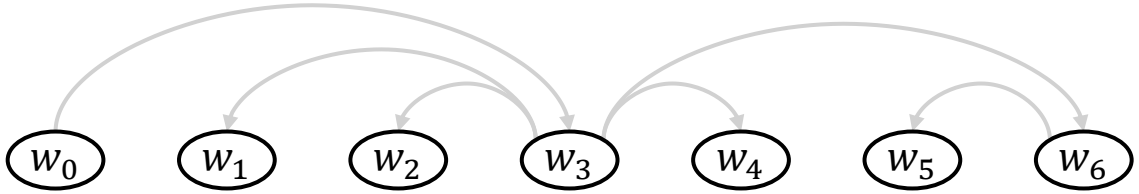
Buffer

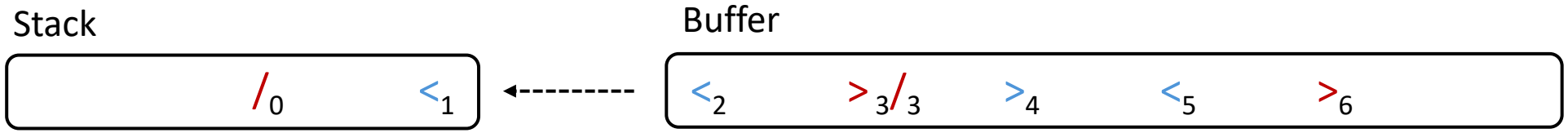
Add $<_1$ to the stack.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



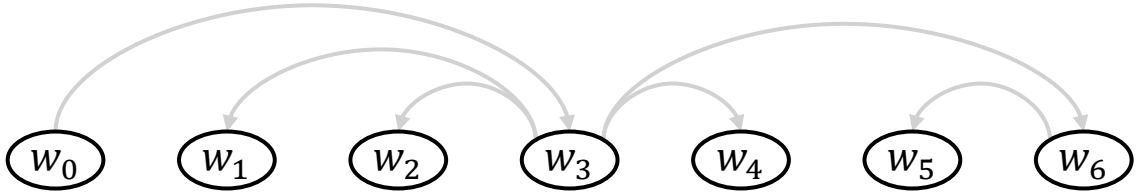


Add $<_2$ to the stack.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack

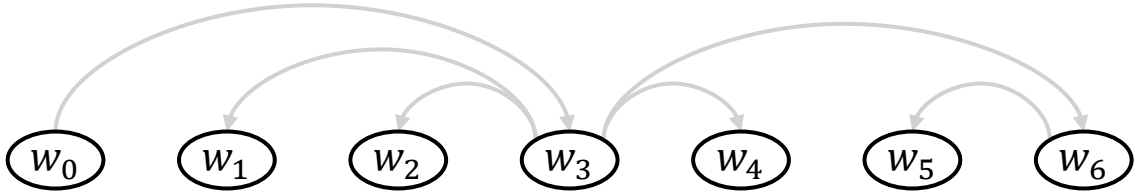
Buffer

Add $<_2$ to the stack.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack

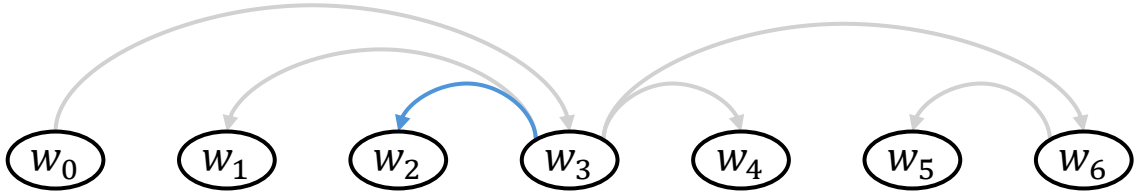
Buffer

Match $>_3$ with the elements in the stack.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack

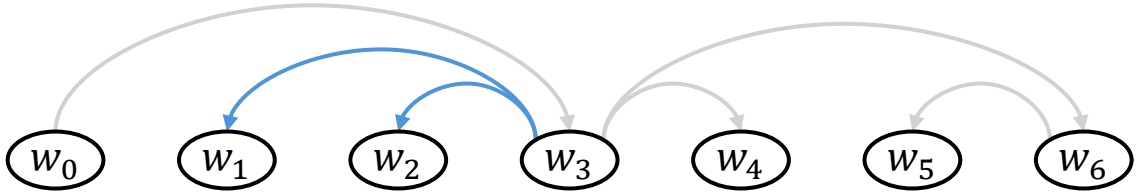
Buffer

Match $>_3$ with the elements in the stack.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack

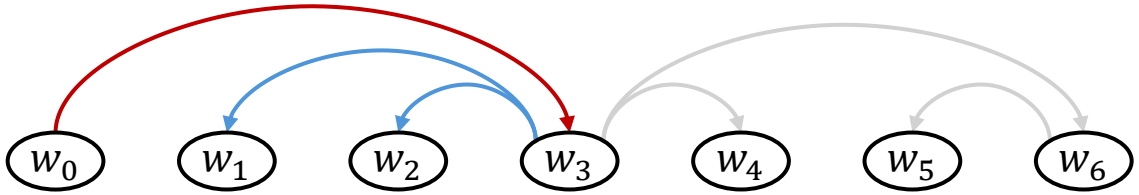
Buffer

Match $>_3$ with the elements in the stack.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack

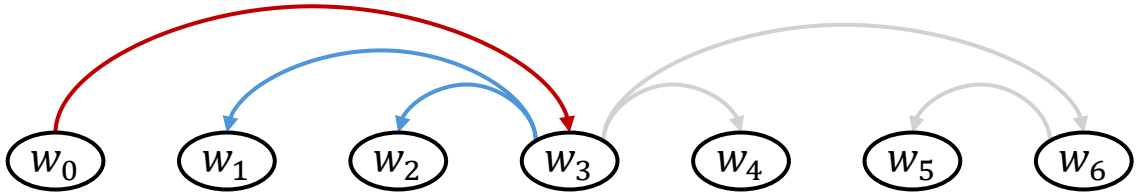
Buffer

Stop when a superbracket is found.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack

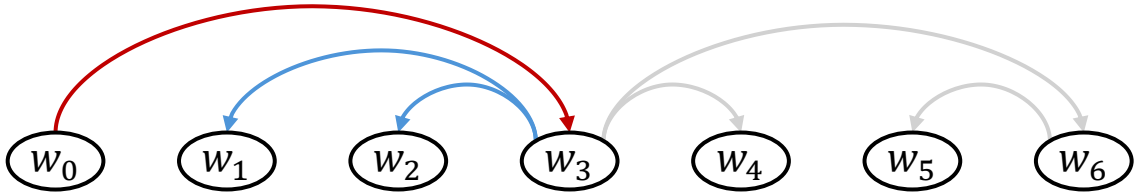
Buffer

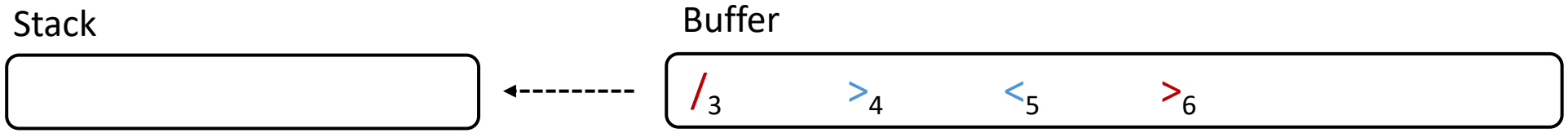
Stop when a superbracket is found.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



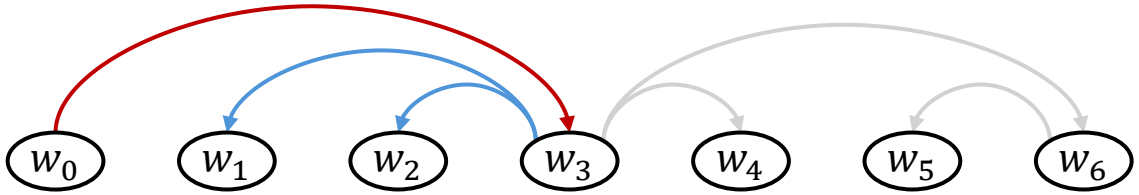


Add $/_3$ to the stack.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack

$/_3$

Buffer

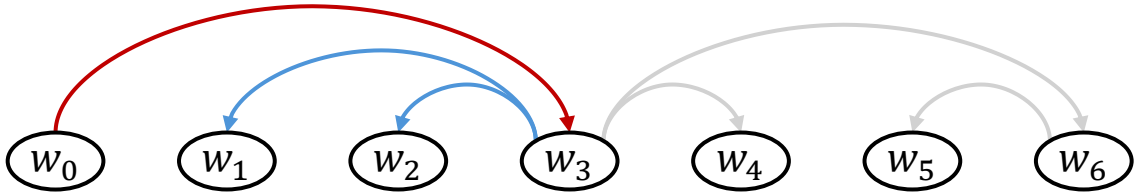
$>_4$ $<_5$ $>_6$

Add $/_3$ to the stack.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack

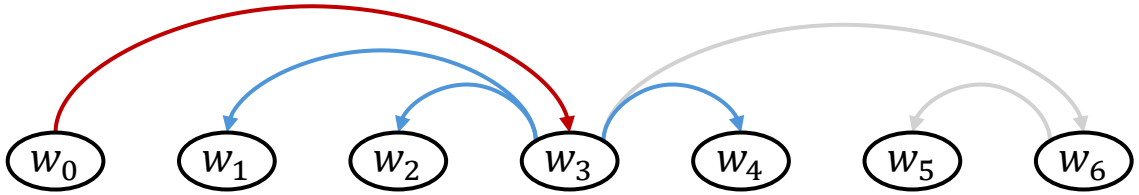
Buffer

Match $>_4$ with the superbracket at the stack.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack

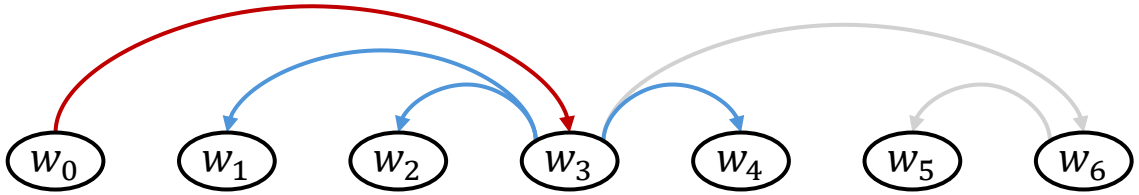
Buffer

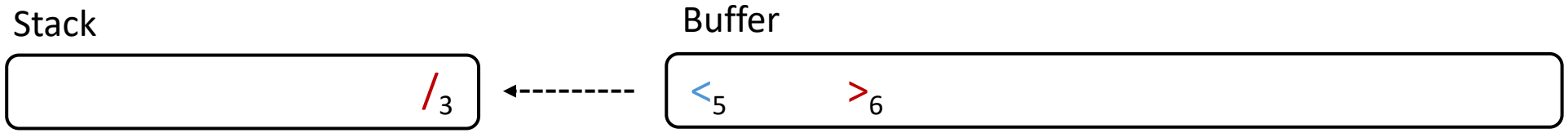
Match $>_4$ with the superbracket at the stack.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



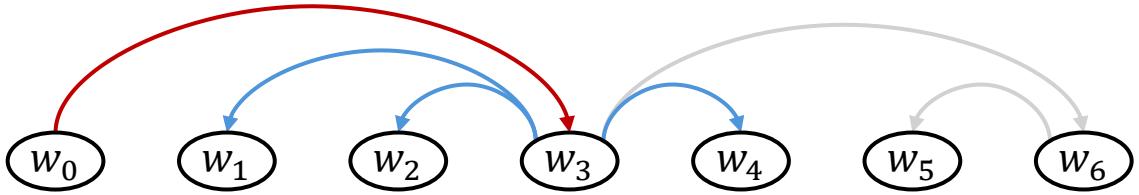


Add $<_5$ to the stack.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack

$/_3$ $<_5$

Buffer

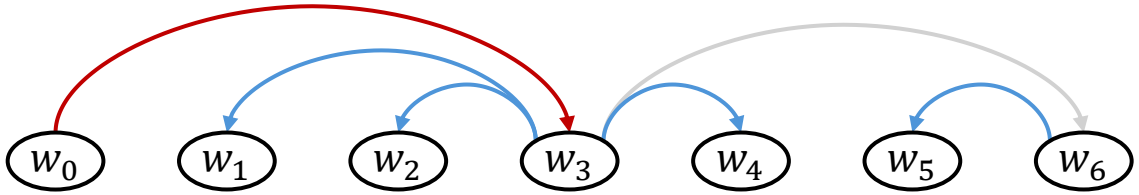
$>_6$

Match $>_6$ with the elements stack.

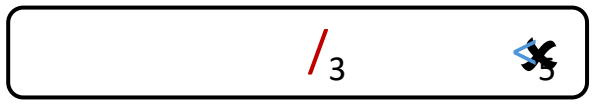
Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack



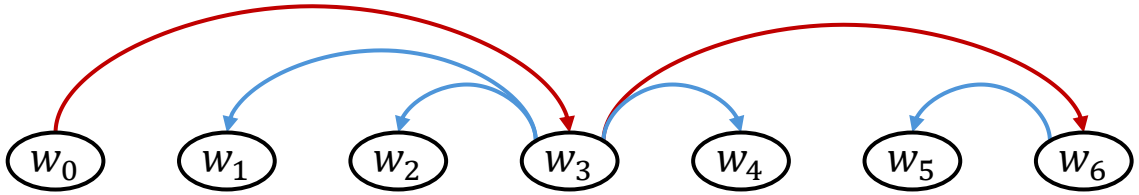
Buffer

Match $>_6$ with the elements stack.

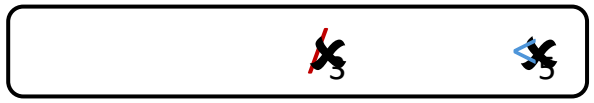
Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack



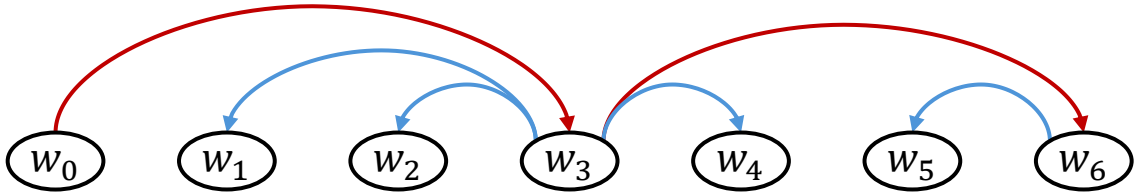
Buffer

Stop iteration when a superbracket is found.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



Stack

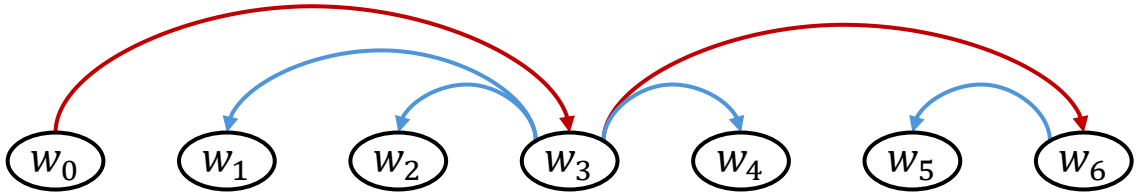
Buffer

Stop iteration when a superbracket is found.

Decoding with Hierarchical Brackets

Stack-based system where:

- Super right dependent (/) matches super-left head (>).
- Super right head (<) matches super-left dependent (\).
- Auxiliary left dependent (<) matches > and \.
- Auxiliary right dependent (>) matches < and /.



All arcs are recovered!

Stack

Buffer