Training Deterministic Parsers Using Non-Deterministic Oracles

Joakim Nivre

Uppsala University
Department of Linguistics and Philology
joakim.nivre@lingfil.uu.se

Joint work with Yoav Goldberg, Bar-Ilan University
Introduction

He sent her a letter.

Deterministic dependency parsing:
- Very fast: $10^5$ words per second
- Fairly accurate: 2–3% below the state of the art

How can we improve accuracy without losing speed?
Introduction

- Transition-based dependency parsing:
  - Define a transition system for dependency parsing
  - Train a classifier for predicting the next transition
  - Use the classifier to do deterministic parsing

- Current practice:
  - Train classifier on derivations produced by an oracle
  - Leads to error propagation at parsing time
  - Can be mitigated by using beam search – slowdown

- Novel idea:
  - Explore a larger search space during training
  - Allow the parser to make mistakes and recover
  - Requires a new type of oracle
Introduction

Outline

1. Transition-based dependency parsing
2. Old oracles – and why they are a problem
3. New oracles – and why they should help
4. Experiments
Dependency Trees

- A dependency tree is a labeled directed tree $T$ with:
  - a set $V$ of nodes, labeled with words
  - a set $A$ of arcs, labeled with dependency types

- Notation:
  - Arc $(w_i, d, w_j)$ links head $w_i$ to dependent $w_j$ with label $d$
  - Shorthand: $w_i \xrightarrow{d} w_j \iff (w_i, d, w_j) \in A$

He sent her a letter.
A parser configuration is a triple \( c = (S, B, A) \), where

- \( S \) = a stack \([\ldots, w_i]_S\) of partially processed words,
- \( B \) = a buffer \([w_j, \ldots]_B\) of remaining input word,
- \( A \) = a set of labeled arcs \((w_i, d, w_j)\).

- **Initialization:**
  \(([\ ]_S, [w_1, \ldots, w_n]_B, \{ \})\)

- **Termination:**
  \((S, [\ ]_B, A)\)
Transition System: Transitions

Left-Arc($d'$) \[
(\ldots, w_i)_S, [w_j, \ldots]_B, A) \\
(\ldots)_S, [w_j, \ldots]_B, A \cup \{(w_j, d, w_i)\}) \neg \text{HEAD}(w_i)
\]

Right-Arc($d$) \[
(\ldots, w_i)_S, [w_j, \ldots]_B, A) \\
(\ldots, w_i, w_j)_S, [\ldots]_B, A \cup \{(w_i, d, w_j)\})
\]

Reduce \[
(\ldots, w_i)_S, B, A) \\
(\ldots)_S, B, A) \text{HEAD}(w_i)
\]

Shift \[
(\ldots)_S, [w_i, \ldots]_B, A) \\
(\ldots, w_i)_S, [\ldots]_B, A)
\]}
Parse Example

Transitions:

Stack       Buffer       Arcs
[ ]_S       [He, sent, her, a, letter, .]_B

He    sent    her    a    letter    .

Training Deterministic Parsers Using Non-Deterministic Oracles
Parse Example

Transitions: SH

Stack       Buffer       Arcs

[He]_S  [sent, her, a, letter, .]_B
Parse Example

Transitions: SH-LA

Stack | Buffer | Arcs
------|--------|------
[ ]_S | [sent, her, a, letter, .]_B | He ←_{SBJ} sent
Parse Example

Transitions: SH-LA-SH

Stack | Buffer | Arcs
---|---|---
[sent]$_S$ | [her, a, letter, .]$_B$ | He $\xleftarrow{\text{SBJ}}$ sent
Parse Example

Transitions: SH-LA-SH-RA

Stack       Buffer       Arcs
\[\text{sent, her}]_S \quad \text{[a, letter, .]}_B \quad \text{He} \quad \text{sent} \quad \text{sent} \quad \text{her}

He \quad \text{sent} \quad \text{her} \quad \text{a} \quad \text{letter} \quad .
Parse Example

**Transitions:** SH-LA-SH-RA-SH

**Stack**

[sent, her, a]$_S$

**Buffer**

[letter, .]$_B$

**Arcs**

He $\overset{\text{SBJ}}{\leftarrow}$ sent

sent $\overset{\text{IOBJ}}{\rightarrow}$ her

He sent her a letter.
Parse Example

Transitions: SH-LA-SH-RA-SH-LA

Stack
[send, her]_S

Buffer
[letter, .]_B

Arcs
He \text{SBJ} sent
sent \text{IOBJ} her
a \text{DET} letter

He sent her a letter.
Parse Example

Transitions: SH-LA-SH-RA-SH-LA-RE

Stack: [sent]$_S$
Buffer: [letter, .]$_B$
Arcs:
- He $\leftarrow_{SBJ}$ sent
- sent $\rightarrow_{IOBJ}$ her
- a $\leftarrow_{DET}$ letter

He sent her a letter.
Parse Example

**Transitions:** SH-LA-SH-RA-SH-LA-RE-RA

**Stack**  
[sent, letter]$_S$  

**Buffer**  
[.]$_B$

**Arcs**  
He $\leftarrow$ SBJ sent  
sent $\rightarrow$ IOBJ her  
a $\leftarrow$ DET letter  
sent $\rightarrow$ DOBJ letter

He sent her a letter.
Parse Example


Stack

Buffer

Arcs

He \leftarrow_{SBJ} sent
sent \rightarrow_{IOBJ} her
a \leftarrow_{DET} letter
sent \rightarrow_{DOBJ} letter

He sent her a letter.
Parse Example


Stack  Buffer  Arcs

\[ \text{He} \rightarrow \text{sent} \rightarrow \text{her} \rightarrow \text{a} \rightarrow \text{letter} \rightarrow . \]
To guide the parser we use a (linear) classifier:

\[ t^* = \arg\max_t w \cdot f(c, t) \]

History-based feature representation \( f(c, t) \):
- Features over input tokens relative to \( S \) and \( B \)
- Features over the (partial) dependency tree defined by \( A \)
- Features over the (partial) transition sequence

Weight vector \( w \) learned from treebank data
Deterministic Parsing

\[
\text{PARSE}(w_1, \ldots, w_n, w)
\]

1. \( c \leftarrow ([], [w_1, \ldots, w_n]_B, \{\}) \)
2. \( \text{while } B_c \neq [] \)
3. \( t^* \leftarrow \text{argmax}_t w \cdot f(c, t) \)
4. \( c \leftarrow t^*(c) \)
5. \( \text{return } T = (\{w_1, \ldots, w_n\}, A_c) \)
Online Learning with an Oracle

\[
\text{LEARN(}\{T_1, \ldots, T_N\})
\]

1. \(w \leftarrow 0.0\)
2. \(\text{for } i \text{ in } 1..K\)
3. \(\text{for } j \text{ in } 1..N\)
4. \(c \leftarrow ([\ ]_s, [w_1, \ldots, w_n]_B, \{\})\)
5. \(\text{while } B_c \neq []\)
6. \(t^* \leftarrow \text{argmax}_t w \cdot f(c, t)\)
7. \(t_o \leftarrow o(c, T_i)\)
8. \(\text{if } t^* \neq t_o\)
9. \(w \leftarrow w + f(c, t_o) - f(c, t^*)\)
10. \(c \leftarrow t_o(c)\)
11. \(\text{return } w\)
Online Learning with an Oracle

\[ \text{\textsc{Learn}}(\{T_1, \ldots, T_N\}) \]

1. \( w \leftarrow 0.0 \)
2. \( \text{for } i \in 1..K \)
3. \( \quad \text{for } j \in 1..N \)
4. \( \quad c \leftarrow ([S], [w_1, \ldots, w_{n_j}]B, \{\}) \)
5. \( \quad \text{while } B_c \neq [] \)
6. \( \quad t^* \leftarrow \text{argmax}_t w \cdot f(c, t) \)
7. \( \quad t_0 \leftarrow o(c, T_i) \)
8. \( \quad \text{if } t^* \neq t_0 \)
9. \( \quad w \leftarrow w + f(c, t_0) - f(c, t^*) \)
10. \( c \leftarrow t_0(c) \)
11. \( \text{return } w \)

Oracle \( o(c, T_i) \) returns the optimal transition for \( c \) and \( T_i \)
Standard Oracle for Arc-Eager Parsing

\[ o(c, T) = \begin{cases} 
  \text{Left-Arc} & \text{if } \text{top}(S_c) \leftarrow \text{first}(B_c) \text{ in } T \\
  \text{Right-Arc} & \text{if } \text{top}(S_c) \rightarrow \text{first}(B_c) \text{ in } T \\
  \text{Reduce} & \text{if } \exists w < \text{top}(S_c) : w \leftrightarrow \text{first}(B_c) \text{ in } T \\
  \text{Shift} & \text{otherwise}
\end{cases} \]

- **Correct:**
  - Derives \( T \) in a configuration sequence \( C(o, T) = c_0, \ldots, c_m \)

- **Problems:**
  - Deterministic: Ignores other derivations of \( T \)
  - Incomplete: Valid only for configurations in \( C(o, T) \)
Non-Determinism

Transitions:
SH-LA-SH-RA-SH-LA-RE-RA-RE-RA

Stack
[ ]s

Buffer
[He, sent, her, a, letter, .]B

Arcs

He  sent  her  a  letter  .
Non-Determinism

Transitions:
SH-LA-SH-RA-SH-LA-RE-RA-RE-RA

SH

Stack
[He]_S

Buffer
[send, her, a, letter, .]_B

Arcs

He  sent  her  a  letter  .

SBJ  IOBJ  DET  DOBJ  PUNC
Non-Determinism

SH-LA

Stack [ ]s
Buffer [sent, her, a, letter, .]B
Arcs He ^SBJ sent

He sent her a letter .
Non-Determinism

Transitions:

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA
SH-LA-SH

Stack
[\text{sent}]_S

Buffer
[\text{her, a, letter, .}]_B

Arcs
\text{He} \xleftarrow{\text{SBJ}} \text{sent}
Non-Determinism

Transitions:

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA

SH-LA-SH-RA

Stack

[sent, her]_S

Buffer

[a, letter, .]_B

Arcs

He ← SBJ sent
sent → IOBJ her

He sent her a letter.

Training Deterministic Parsers Using Non-Deterministic Oracles
Non-Determinism

Transitions:

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA

SH-LA-SH-RA-RE

Stack

[sent]_S

Buffer

[a, letter, .]_B

Arcs

He \xleftarrow{SBJ} sent

sent \xrightarrow{IOBJ} her

He \xrightarrow{SBJ} sent

sent \xrightarrow{IOBJ} her

He sent her a letter.
Non-Determinism

Transitions:

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA
SH-LA-SH-RA-RE-SH

Stack

[ sent, a ]

Buffer

[ letter, . ]

Arcs

He \( ^{SBJ} \) sent
sent \( ^{IOBJ} \) her

He \( \rightarrow^{SBJ} \) sent
sent \( \rightarrow^{IOBJ} \) her

He \( \leftarrow\) sent
sent \( \rightarrow \) her

a letter
Non-Determinism

**Transitions:**

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA

SH-LA-SH-RA-RE-SH-LA

**Stack**

[sent]$_S$

**Buffer**

[letter, .]$_B$

**Arcs**

He $\leftarrow^{SBJ}$ sent

sent $\rightarrow^{IOBJ}$ her

a $\leftarrow^{DET}$ letter

He sent her a letter.
Non-Determinism

**Transitions:**

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA

SH-LA-SH-RA-RE-SH-LA-RA

**Stack**

[sent, letter]_S

**Buffer**

[.]_B

**Arcs**

He ← SBJ sent

sent IOBJ her

a DET letter

sent DOBJ letter
Non-Determinism

Transitions:
SH-LA-SH-RA-SH-LA-RE-RA-RE-RA
SH-LA-SH-RA-RE-SH-LA-RA-RE

Stack
[sent]_S

Buffer
[.]_B

Arcs
He \(\xleftarrow{\text{SBJ}}\) sent
sent \(\xrightarrow{\text{IOBJ}}\) her
\(a \xleftarrow{\text{DET}}\) letter
sent \(\xrightarrow{\text{DOBJ}}\) letter
Non-Determinism

Transitions:

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA

SH-LA-SH-RA-RE-SH-LA-RA-RE-RA

Stack  Buffer  Arcs

[sent, .]_S  [ ]_B

He  sent  her  a  letter  .

He  \(\xleftarrow{SBJ}\) sent

sent  \(\xrightarrow{IOBJ}\) her

a  \(\xleftarrow{DET}\) letter

sent  \(\xrightarrow{DOBJ}\) letter

sent  \(\xrightarrow{PUNC}\) .
Non-Optimality

Transitions:

Stack | Buffer | Arcs
---|---|---
[ ]ₜ | [He, sent, her, a, letter, .]ₜ | 

He sent her a letter .

SBJ | IOBJ | DET
---|---|---
He | sent | her

PUNC

DOB
Non-Optimality

Transitions:

Stack

$[\text{He}]_S$

Buffer

$[\text{sent, her, a, letter, .}]_B$

Arcs

He sent her a letter .
Non-Optimality

Transitions:

Stack

Buffer

[ ]s

[sent, her, a, letter, .]_B

Arcs

He \(\overset{\text{SBJ}}{\leftarrow}\) sent

He sent her a letter .
Non-Optimality

Transitions: SH-LA-SH

Stack

Buffer

Arcs

He  sent

He  sent  her  a  letter  .
Non-Optimality

Transitions: SH-LA-SH-SH

Stack [sent, her]_S
Buffer [a, letter, .]_B
Arcs He ← sent

He sent her a letter .
Non-Optimality

Transitions:

SH-LA-SH-SH-SH

Stack

[ sent, her, a ]

Buffer

[ letter, . ]

Arcs

He  sent

He    sent    her    a     letter    .
Non-Optimality

Transitions:

SH-LA-SH-SH-SH-LA

Stack

[sent, her]_S

Buffer

[letter, .]_B

Arcs

He SBJ sent
a DET letter

He sent her a letter.
Non-Optimality

Transitions:
SH-LA-SH-SH-SH-LA-SH

Stack
[sent, her, letter]$_S$

Buffer
[.]$_B$

Arcs
He $\leftarrow^{SBJ}$ sent
a $\leftarrow^{DET}$ letter

He sent her a letter.
Non-Optimality

Transitions:


Stack

Buffer

[sent, her, letter, .]_S  [ ]_B

Arcs

He ←− sent  a ←− letter

He sent her a letter .
Non-Optimality

Transitions:


Stack

[ ]

Buffer

[He, sent, her, a, letter, .]_B

Arcs

He  sent  her  a  letter  .
Non-Optimality

Transitions:


SH

Stack

[He]_{S}

Buffer

[sent, her, a, letter, .]_{B}

Arcs

He  sent  her  a  letter  .

PUNC

DOB

DET

IOB

SBJ
Non-Optimality

Transitions:

SH-LA-SH-SH-SH-LA-SH-SH \[2/5\]
SH-LA

Stack

Buffer

Arcs

He $\text{sent}$ sent

[ ]$_s$

[sent, her, a, letter, .]$_B$

He $\text{sent}$ sent
Non-Optimality

Transitions:

SH-LA-SH

Stack

[ sent ]_S

Buffer

[ her, a, letter, . ]_B

Arcs

He ←− sent
Non-Optimality

Transitions:

SH-LA-SH-SH

Stack

[ sent, her ]_S

Buffer

[ a, letter, . ]_B

Arcs

He \( \overset{\text{SBJ}}{\leftarrow} \) sent

He sent her a letter .
Non-Optimality

Transitions:


SH-LA-SH-SH-SH

Stack

[sent, her, a]$_S$

Buffer

[letter, .]$_B$

Arcs

He $\leftarrow$ sent

He sent her a letter.
Non-Optimality

Transitions:


SH-LA-SH-SH-SH-LA

Stack

[sent, her]$_S$

Buffer

[letter, .]$_B$

Arcs

He $\xleftarrow{\text{SBJ}}$ sent

a $\xleftarrow{\text{DET}}$ letter

He sent her a letter.
Non-Optimality

Transitions:

SH-LA-SH-SH-SH-LA-LA

Stack
[\text{sent}]_S

Buffer
[\text{letter, .}]_B

Arcs
He \overset{\text{SBJ}}{\leftarrow} \text{sent}
\text{a} \overset{\text{DET}}{\leftarrow} \text{letter}
\text{her} \overset{\times}{\leftarrow} \text{letter}
Non-Optimality

Transitions:

SH-LA-SH-SH-SH-LA-LA-RA

Stack
[sent, letter]_S

Buffer
[.]_B

Arcs
He SBJ sent
a DET letter
her X letter
sent DOBJ letter

He sent her a letter .
Non-Optimality

Transitions:
SH-LA-SH-SH-SH-LA-LA-RA-RE

Stack
[sent]$_S$

Buffer
[.]$_B$

Arcs
He $\xleftarrow{SBJ}$ sent
a $\xleftarrow{DET}$ letter
her $\xleftarrow{X}$ letter
sent $\xrightarrow{DOBJ}$ letter
Non-Optimality

Transitions:


Stack

[ sent, . ]\textsubscript{S}

Buffer

[ ]\textsubscript{B}

Arcs

He \xleftarrow{SBJ} sent

a \xleftarrow{DET} letter

her \xleftarrow{X} letter

sent \xrightarrow{DOBJ} letter

sent \xrightarrow{PUNC} .
Rethinking Oracles

- New idea:
  - A transition is optimal if the best tree remains reachable
  - Best tree = $\text{argmin}_{T'} \mathcal{L}(T, T')$

- New view of oracle:
  - Boolean function $o(c, t, T) = \text{true}$ if $t$ is optimal for $c$ and $T$
  - Non-deterministic: More than one transition can be optimal
  - Complete: Correct for all configurations

- New problem:
  - How do we know which trees are reachable?
Reachability for Arcs and Trees

- **Arc reachability:**
  - An arc $w_i \rightarrow w_j$ is reachable in $c$ iff $w_i \rightarrow w_j \in A_c$, or $w_i \in S_c \cup B_c$ and $w_j \in B_c$ (same for $w_i \leftarrow w_j$)

- **Tree reachability:**
  - A (projective) tree $T$ is reachable in $c$ iff every arc in $T$ is reachable in $c$

- **Arc-decomposable system:**
  - Tree reachability reduces to arc reachability
  - Holds for some transition systems but not all
A New Oracle

\[ \mathcal{R}(c) \equiv \{ a \mid a \text{ is an arc reachable in } c \} \]

\[ o(c, t, T) = \begin{cases} 
\text{true} & \text{if } [\mathcal{R}(c) - \mathcal{R}(t(c))] \cap T = \emptyset \\
\text{false} & \text{otherwise} 
\end{cases} \]
Case by Case

- Notation: \( s \) = word on top of stack, \( b \) = first word in buffer

\[
\begin{align*}
o(c, LA, T) &= \begin{cases} \text{false} & \text{if } \exists w \in B_c : s \leftrightarrow w \in T \text{ (except } s \leftarrow b) \\ \text{true} & \text{otherwise} \end{cases} \\
o(c, RA, T) &= \begin{cases} \text{false} & \text{if } \exists w \in S_c : w \leftrightarrow b \in T \text{ (except } s \rightarrow b) \\ \text{true} & \text{otherwise} \end{cases} \\
o(c, RE, T) &= \begin{cases} \text{false} & \text{if } \exists w \in B_c : s \rightarrow w \in T \\ \text{true} & \text{otherwise} \end{cases} \\
o(c, SH, T) &= \begin{cases} \text{false} & \text{if } \exists w \in S_c : w \leftrightarrow b \in T \\ \text{true} & \text{otherwise} \end{cases}
\end{align*}
\]
A New Learning Algorithm

\textbf{LEARN}(\{T_1, \ldots, T_N\})

1. \( w \leftarrow 0.0 \)
2. \textbf{for} \( i \) in 1..K 
3. \textbf{for} \( j \) in 1..N 
4. \( c \leftarrow ([\_], [w_1, \ldots, w_{n_j}]_B, \{\}) \)
5. \textbf{while} \( B_c \neq [] \)
6. \( t^* \leftarrow \arg\max_t \ w \cdot f(c, t) \)
7. \( t_o \leftarrow \arg\max_{t \in \{t \mid o(c, t, T_i)\}} \ w \cdot f(c, t) \)
8. \textbf{if} \( t^* \neq t_o \)
9. \( w \leftarrow w + f(c, t_o) - f(c, t^*) \)
10. \( c \leftarrow \text{CHOICE}(t_o(c), t^*(c)) \)
11. \textbf{return} \( w \)
A New Learning Algorithm

\textbf{LEARN}(\{T_1, \ldots, T_N\})

1. \( w \leftarrow 0.0 \)
2. \textbf{for} \( i \) \textbf{in} 1..K
3. \quad \textbf{for} \( j \) \textbf{in} 1..N
4. \quad \quad c \leftarrow ([ ]_S, [w_1, \ldots, w_n]_B, \{ \})
5. \quad \textbf{while} \ B_c \neq []
6. \quad \quad t^* \leftarrow \text{argmax}_t w \cdot f(c, t)
7. \quad \quad t_o \leftarrow \text{argmax}_{t \in \{t | o(c, t, T_i)\}} w \cdot f(c, t)
8. \quad \quad \textbf{if} t^* \neq t_o
9. \quad \quad \quad w \leftarrow w + f(c, t_o) - f(c, t^*)
10. \quad \quad c \leftarrow \text{CHOICE}(t_o(c), t^*(c))
11. \textbf{return} w
Experimental Evaluation

- Data sets:
  - English treebanks: WSJ, Brown, BNC, Google Web
  - Multilingual: CoNLL 2007 Shared Task

- Settings:
  - Greedy: Old learning algorithm
  - Greedy + Ambiguity: \( \text{CHOICE}(t_0(c), t^*(c)) = t_0(c) \)
  - Greedy + Exploration: Random \( \text{CHOICE}(t_0(c), t^*(c)) \)
Experiments

English Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Greedy</th>
<th>Greedy + Ambiguity</th>
<th>Greedy + Exploration</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSJ22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSJ23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brown</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Questions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BNC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Answers / web</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emails / web</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Groups / web</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reviews / web</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blogs / web</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Football</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parsing Accuracy

77  80  83  86  89  92
Experiments

Multilingual Results

<table>
<thead>
<tr>
<th>Language</th>
<th>Greedy</th>
<th>Greedy + Ambiguity</th>
<th>Greedy + Exploration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basque</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Catalan</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chinese</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Czech</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greek</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hungarian</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Italian</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turkish</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parsing Accuracy

70 74 78 82 86 90 94
Conclusion

- Exploring a larger search space at training time helps
  - Allowing non-canonical derivations (spurious ambiguity)
  - Learning optimal transitions in non-optimal configurations
- Requires a new type of oracle
  - Non-deterministic: more than one transition may be optimal
  - Complete: optimality defined for all configurations
- Parsing remains deterministic (fast)