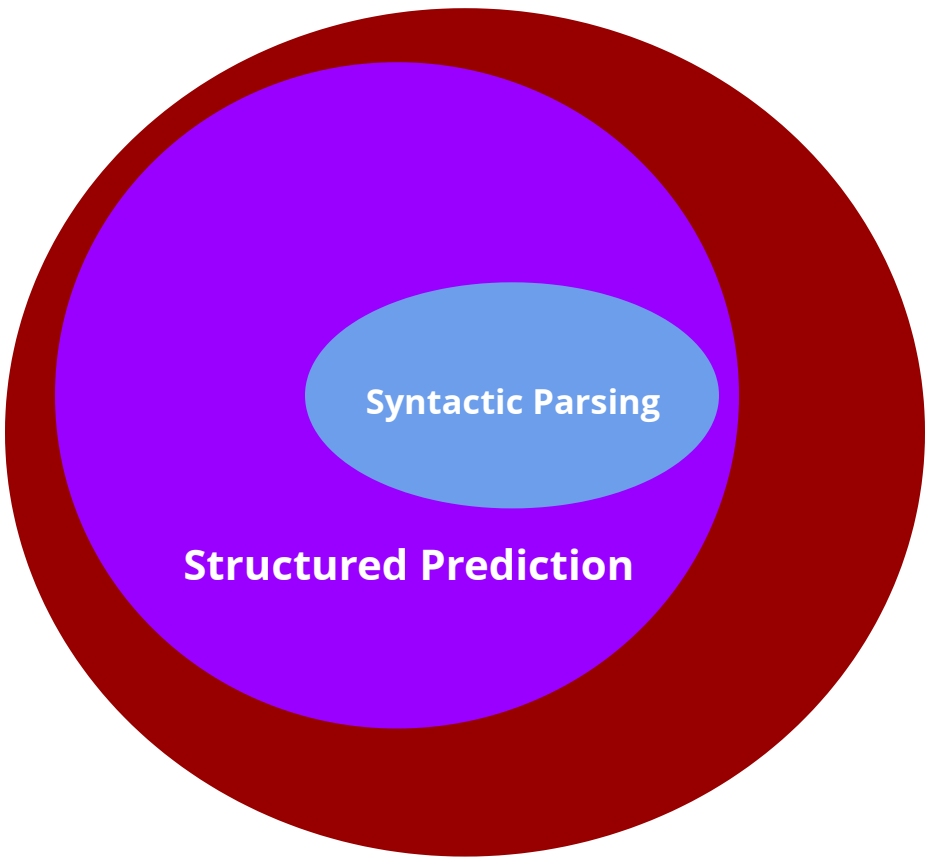
50.040
Natural Language Processing

Lu, Wei

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Tasks in NLP



Syntactic Parsing

The parsers that we have built so far are based on dynamic programming algorithms on top of some charts.

Such algorithms are "efficient", but still invovles either a cubic time or a quadratic time complexity.

Practical challenge: processing data at web-scale may be too slow!

Question

Can we perform effective parsing in linear time?

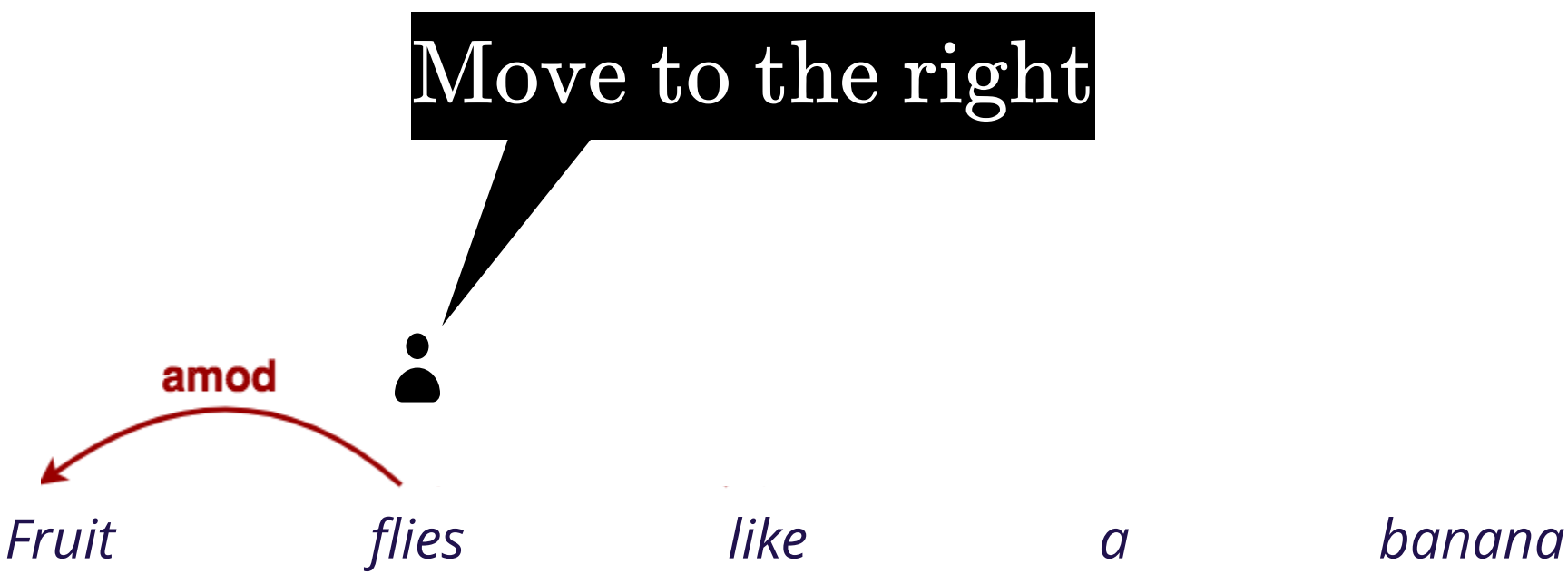
Incremental Parsing

Fruit *flies* *like* *a* *banana*

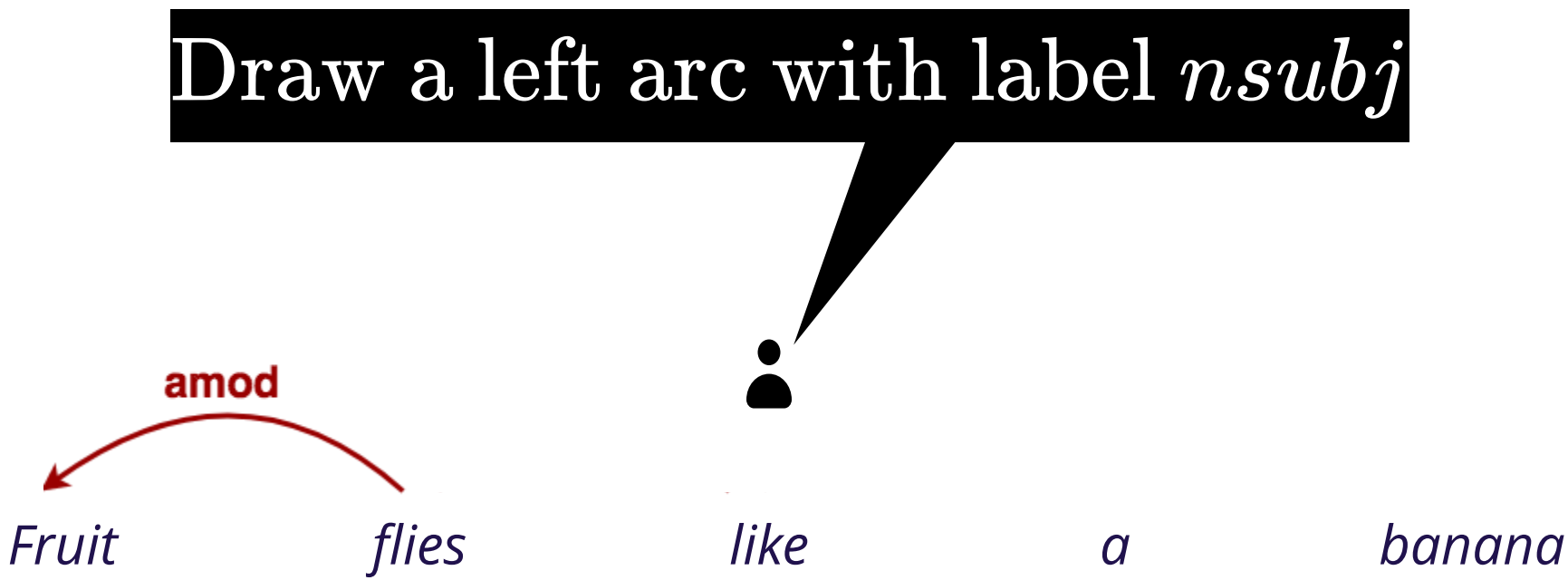
Incremental Parsing

Fruit *flies* *like* *a* *banana*

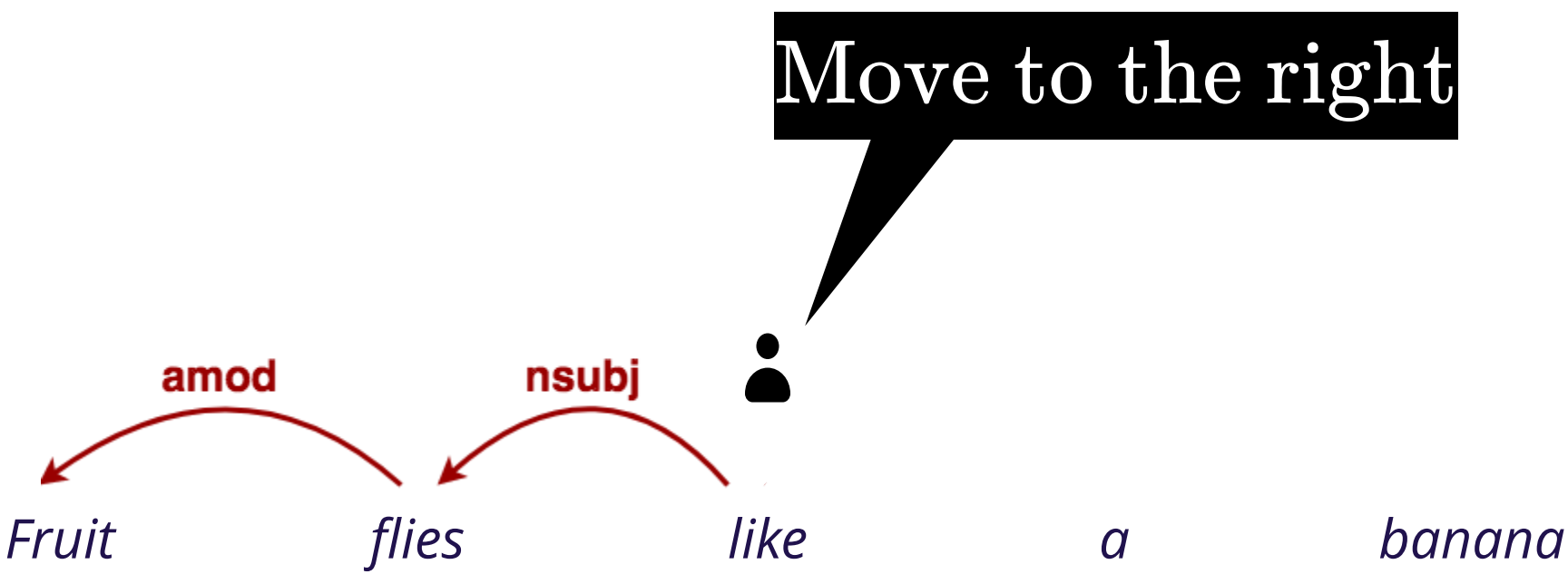
Incremental Parsing



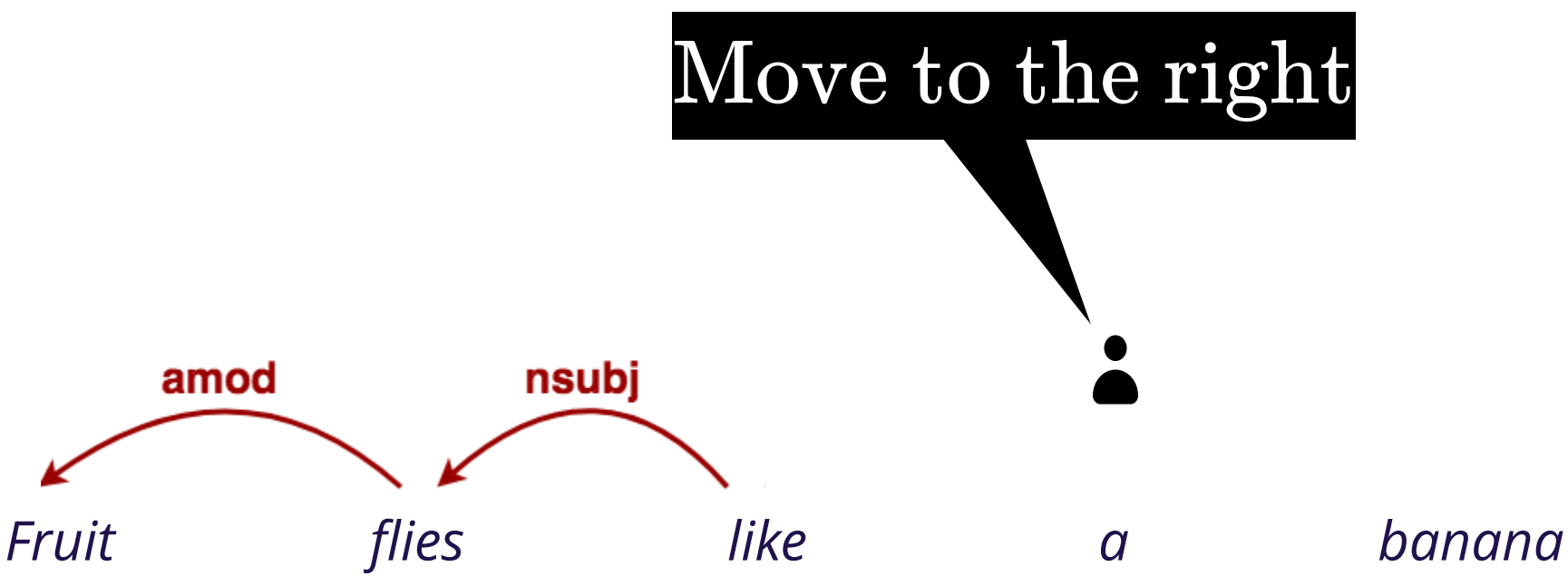
Incremental Parsing



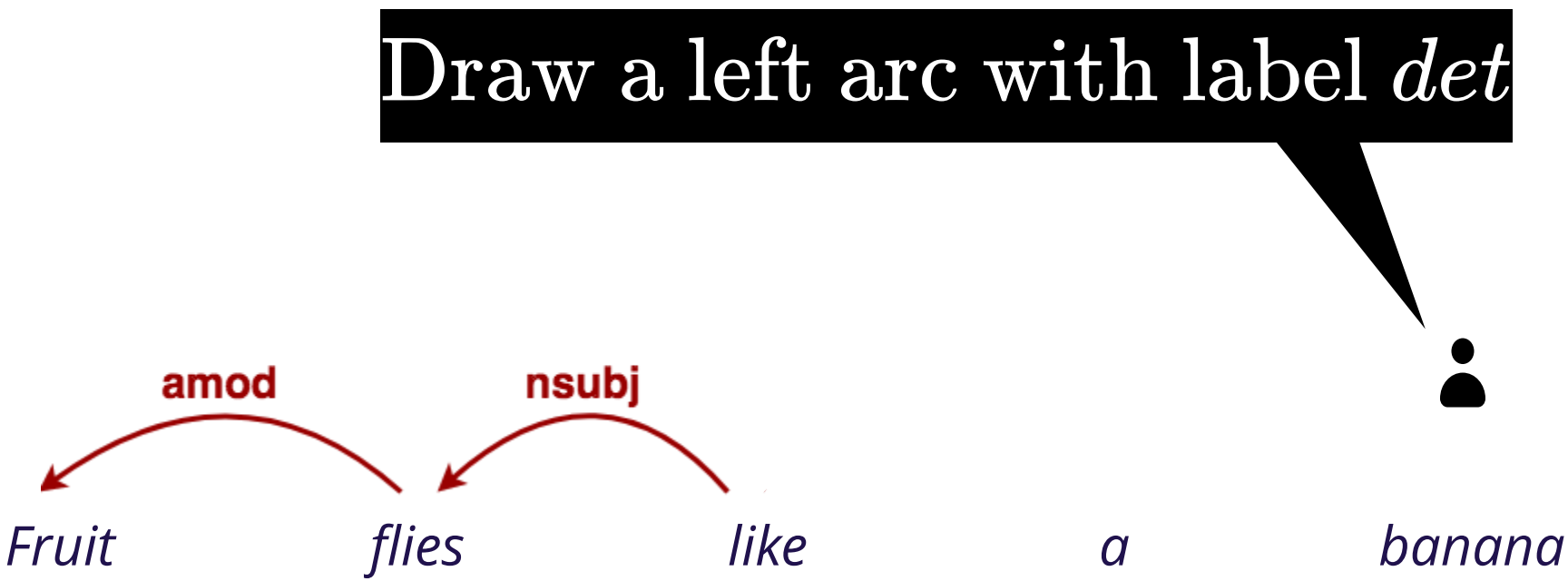
Incremental Parsing



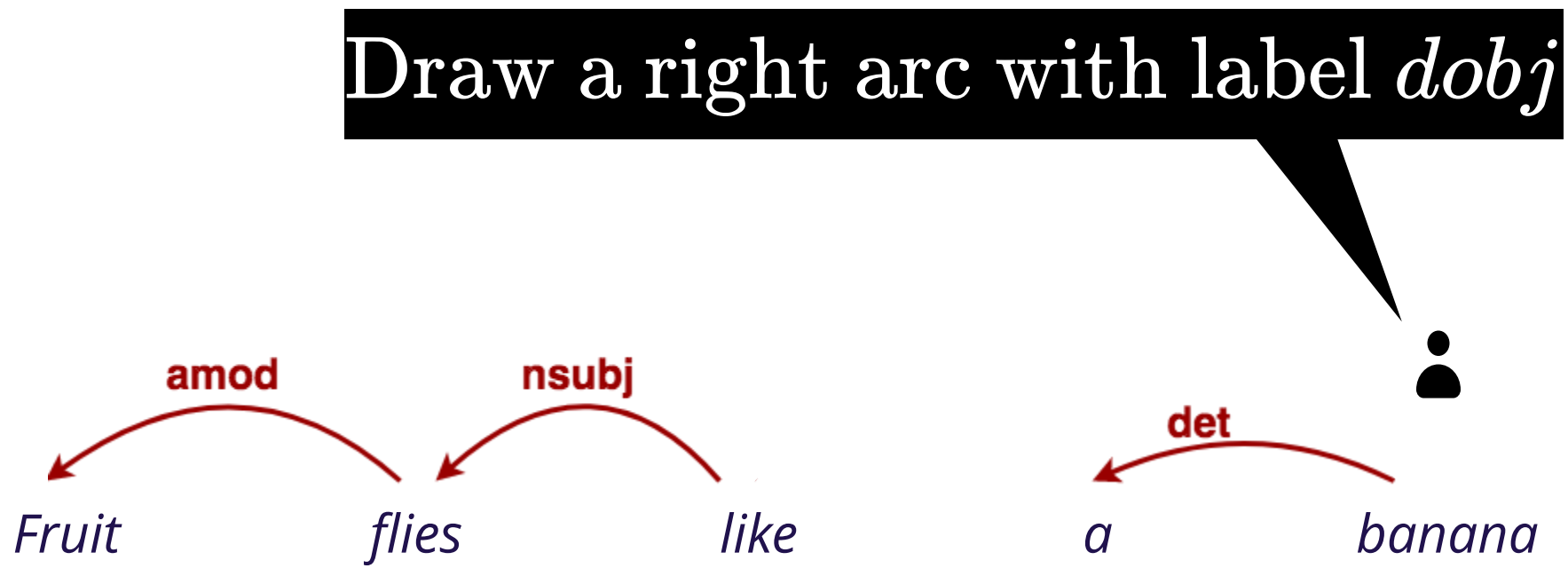
Incremental Parsing



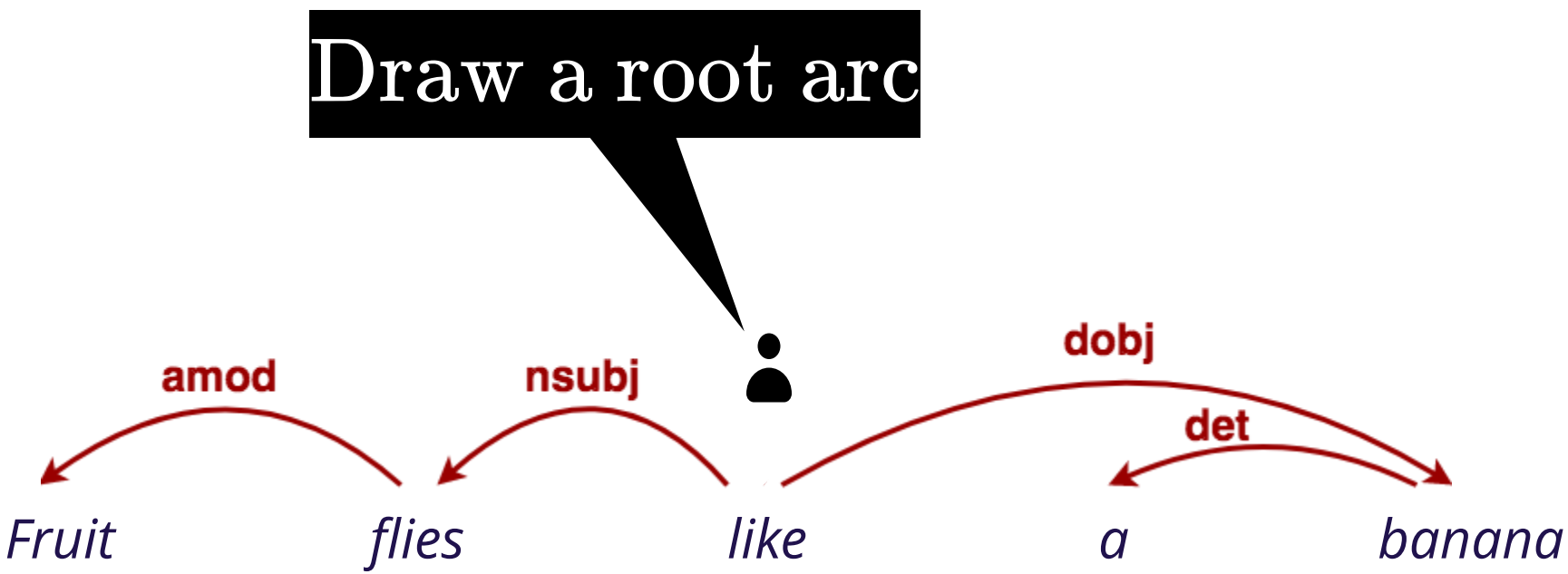
Incremental Parsing



Incremental Parsing

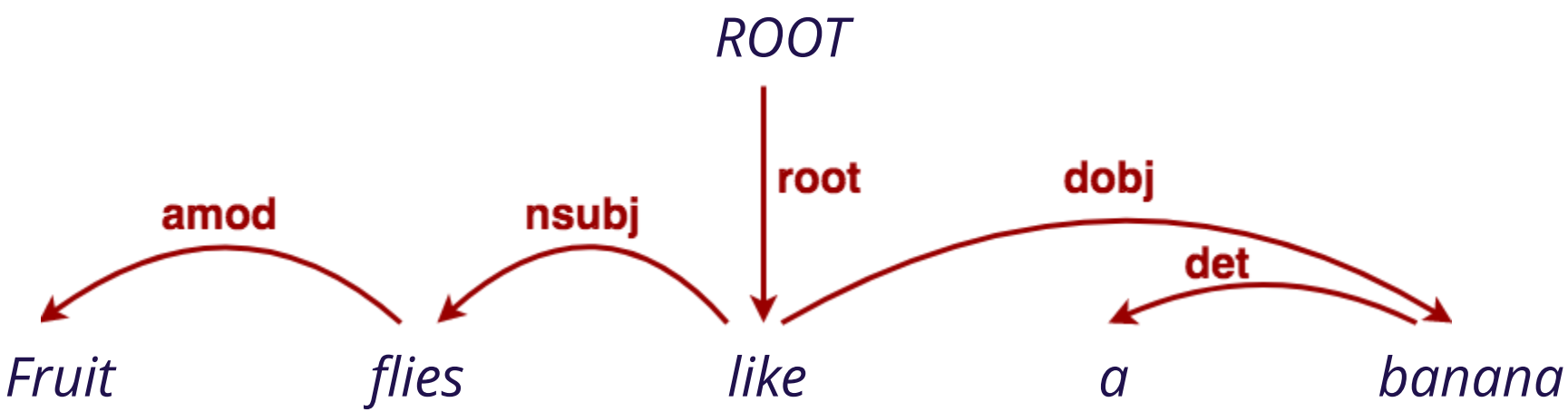


Incremental Parsing



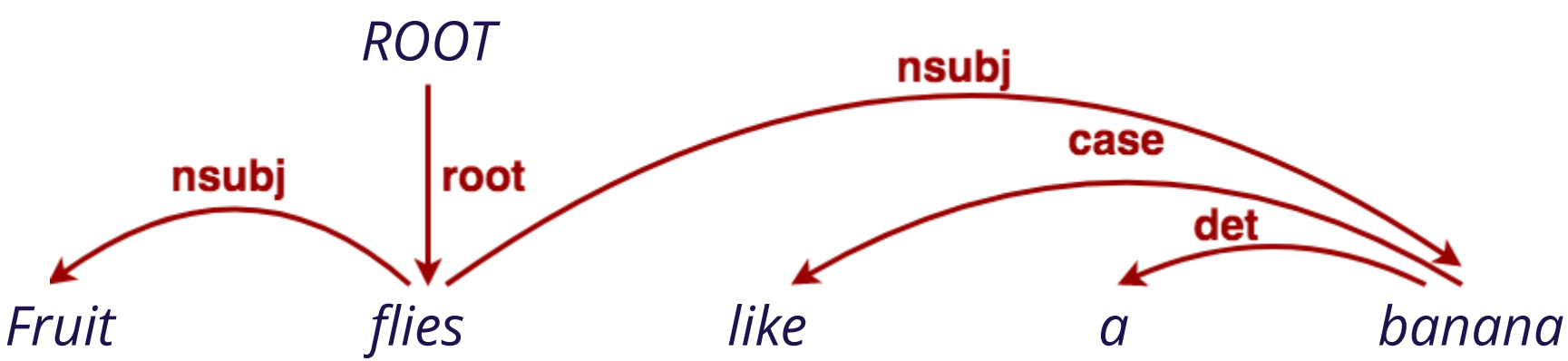
Incremental Parsing

We have successfully constructed a tree with a incremental approach from the left to the right, and … this procedure runs in **linear time**!



Incremental Parsing

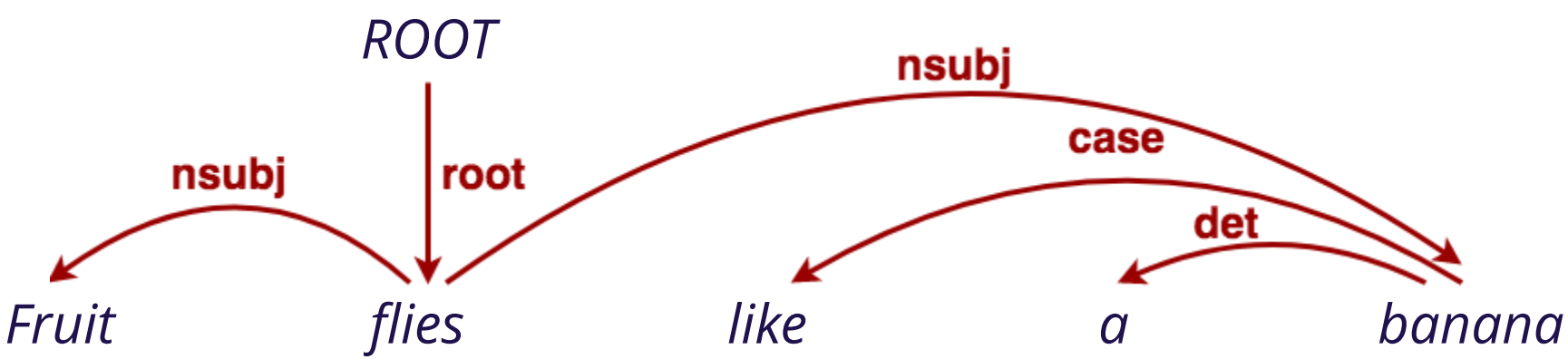
Following another sequence of actions, we will be able to arrive at a different parse tree.



Incremental Parsing

Key Observations

1. We need to design the set of actions.
2. We need to find a way to measure the quality of an action at any time.



Incremental Parsing (Ratnaparkhi, 1997)

A Linear Observed Time Statistical Parser Based on Maximum Entropy Models

Adwait Ratnaparkhi•
Dept. of Computer and Information Science
University of Pennsylvania
200 South 33rd Street
Philadelphia, PA 19104-6389
adwait@unagi.cis.upenn.edu

Abstract

This paper presents a statistical parser for natural language that obtains a parsing accuracy—roughly 87% precision and 86% recall—which surpasses the best previously published results on the Wall St. Journal domain. The parser itself requires very little human intervention, since the information it uses to make parsing decisions is specified in a concise and simple manner, and is combined in a fully automatic way under the maximum entropy framework. The observed running time of the parser on a test sentence is linear with respect to the sentence length. Furthermore, the parser returns several scored parses for a sentence, and this paper shows that a scheme to pick the best parse from the 20 highest scoring parses *could* yield a dramatically higher accuracy of 93% precision and recall.

1 Introduction

This paper presents a statistical parser for natural language that finds one or more scored syntactic parse trees for a given input sentence. The parsing accuracy—roughly 87% precision and 86% recall—surpasses the best previously published results on the Wall St. Journal domain. The parser consists of the following three conceptually distinct parts:

1. A set of procedures that use certain actions to

3. A search heuristic which attempts to find the highest scoring parse tree for a given input sentence.

The maximum entropy models used here are similar in form to those in (Ratnaparkhi, 1996; Berger, Della Pietra, and Della Pietra, 1996; Lau, Rosenfeld, and Roukos, 1993). The models compute the probabilities of actions based on certain syntactic characteristics, or *features*, of the current context. The features used here are defined in a concise and simple manner, and their relative importance is determined automatically by applying a training procedure on a corpus of syntactically annotated sentences, such as the Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1994). Although creating the annotated corpus requires much linguistic expertise, creating the feature set for the parser itself requires very little linguistic effort.

Also, the search heuristic is very simple, and its observed running time on a test sentence is linear with respect to the sentence length. Furthermore, the search heuristic returns several scored parses for a sentence, and this paper shows that a scheme to pick the best parse from the 20 highest scoring parses *could* yield a dramatically higher accuracy of 93% precision and recall.

Sections 2, 3, and 4 describe the tree-building procedures, the maximum entropy models, and the search heuristic, respectively. Section 5 describes experiments with the Penn Treebank and section 6 compares this paper with previously published works.

Incremental Parsing

Three passes, with actions defined below.

Table 1. Tree-building procedures of parser.

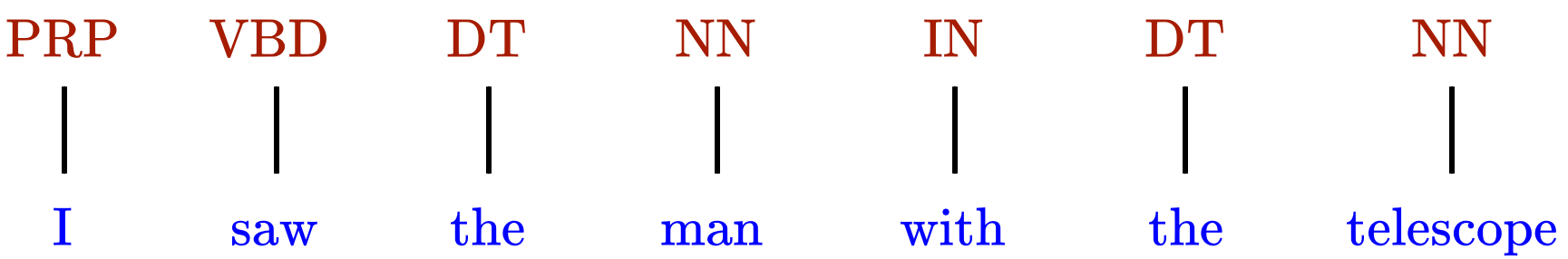
Pass	Procedure	Actions	Description
First Pass	TAG	A POS tag in tag set	Assign POS Tag to word
Second Pass	CHUNK	`Start X`, `Join X`, `Other`	Assign Chunk tag to POS tag and word
Third Pass	BUILD	`Start X`, `Join X`, where `X` is a constituent label in label set	Assign current tree to start a new constituent, or to join the previous one
	CHECK	`Yes`, `No`	Decide if current constituent is complete

Incremental Parsing

I saw the man with the telescope

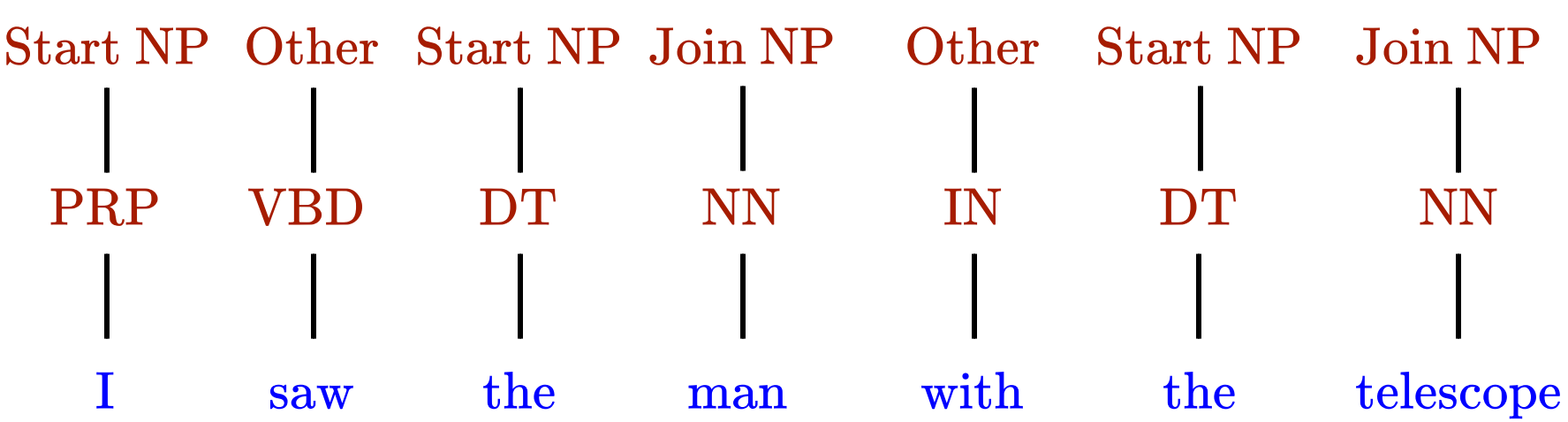
Incremental Parsing

Assign POS Tags



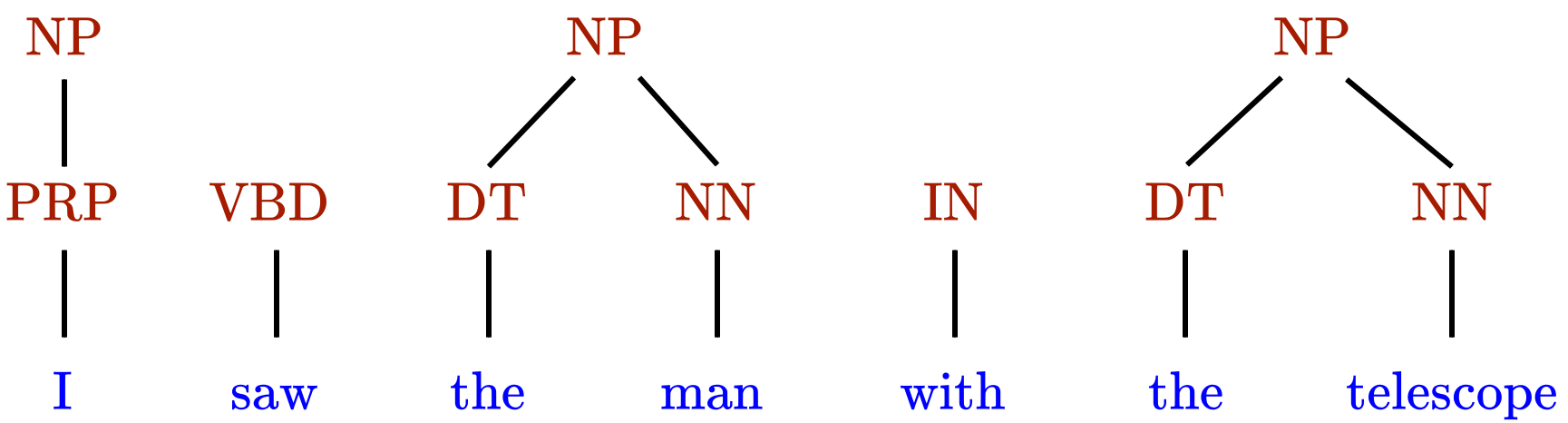
Incremental Parsing

Assign Chunk Labels



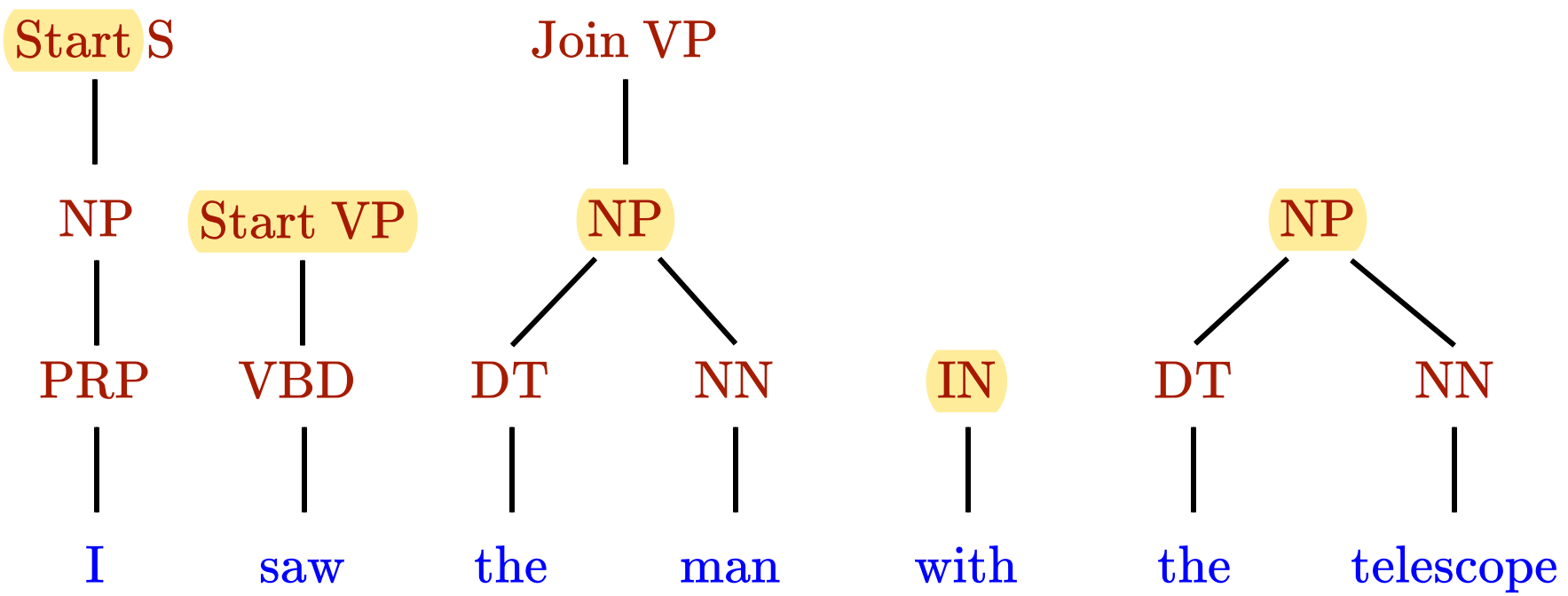
Incremental Parsing

Build constituents based on chunk labels.



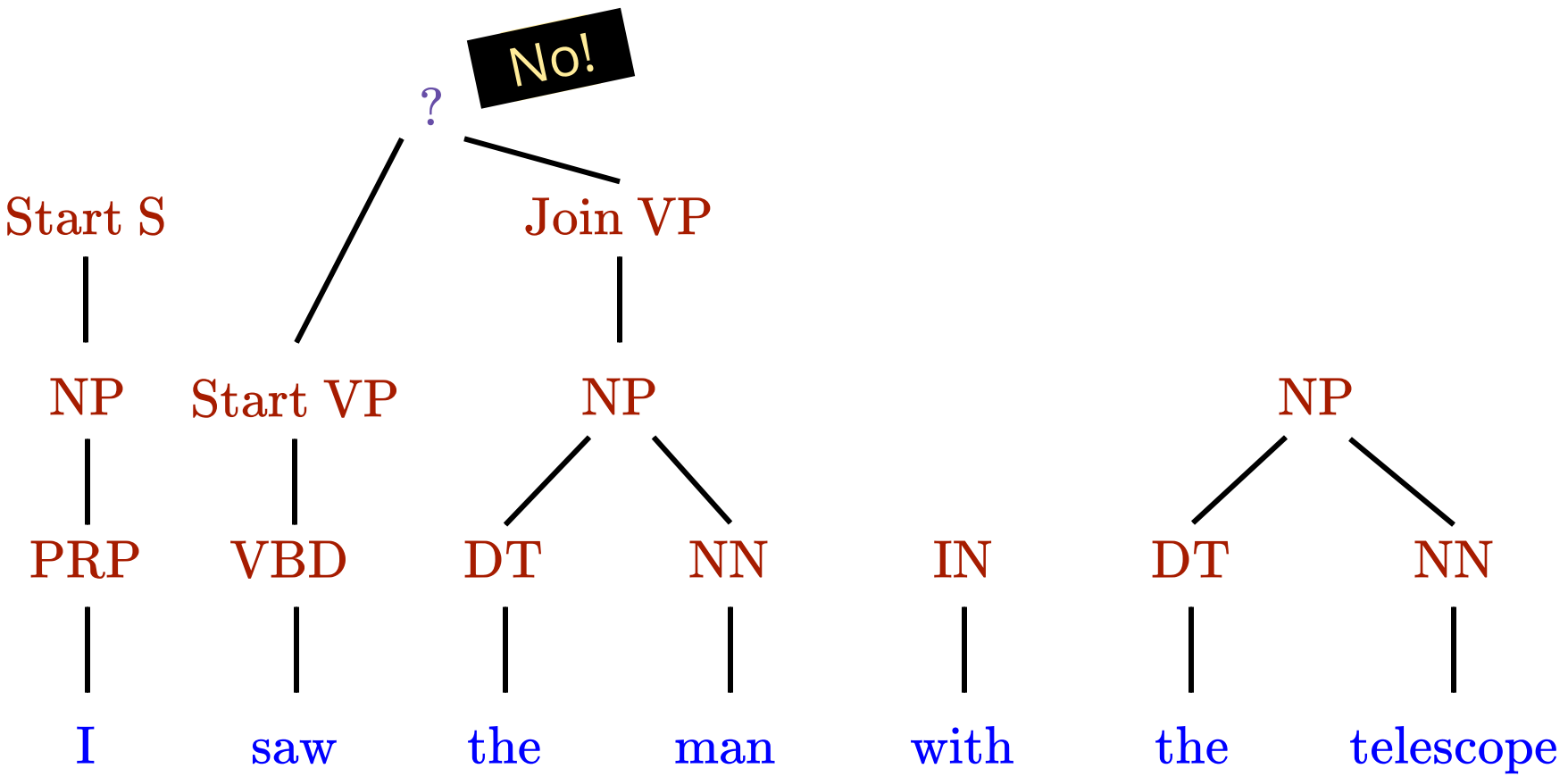
Incremental Parsing

Assign constituent labels.
Assume we just finished
BUILD with action "Join VP"



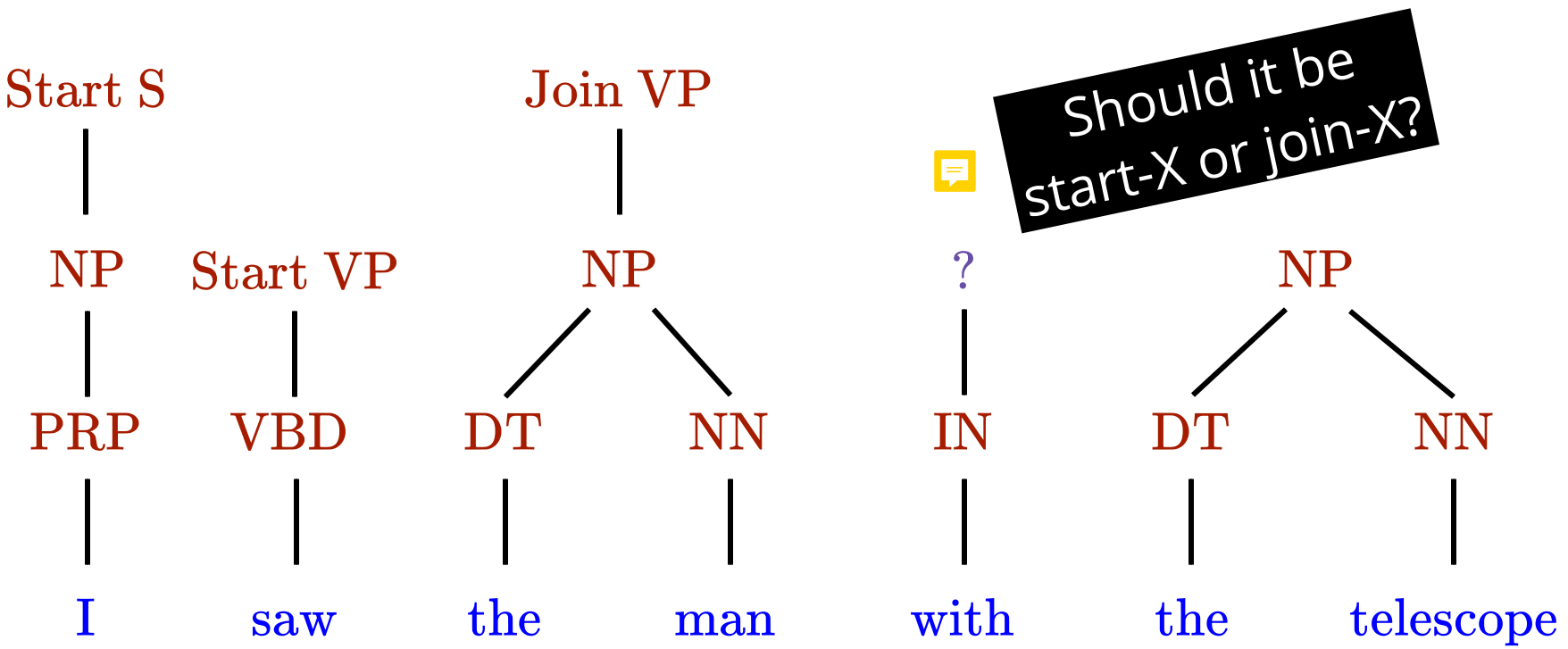
Incremental Parsing

CHECK if the constituent is complete?



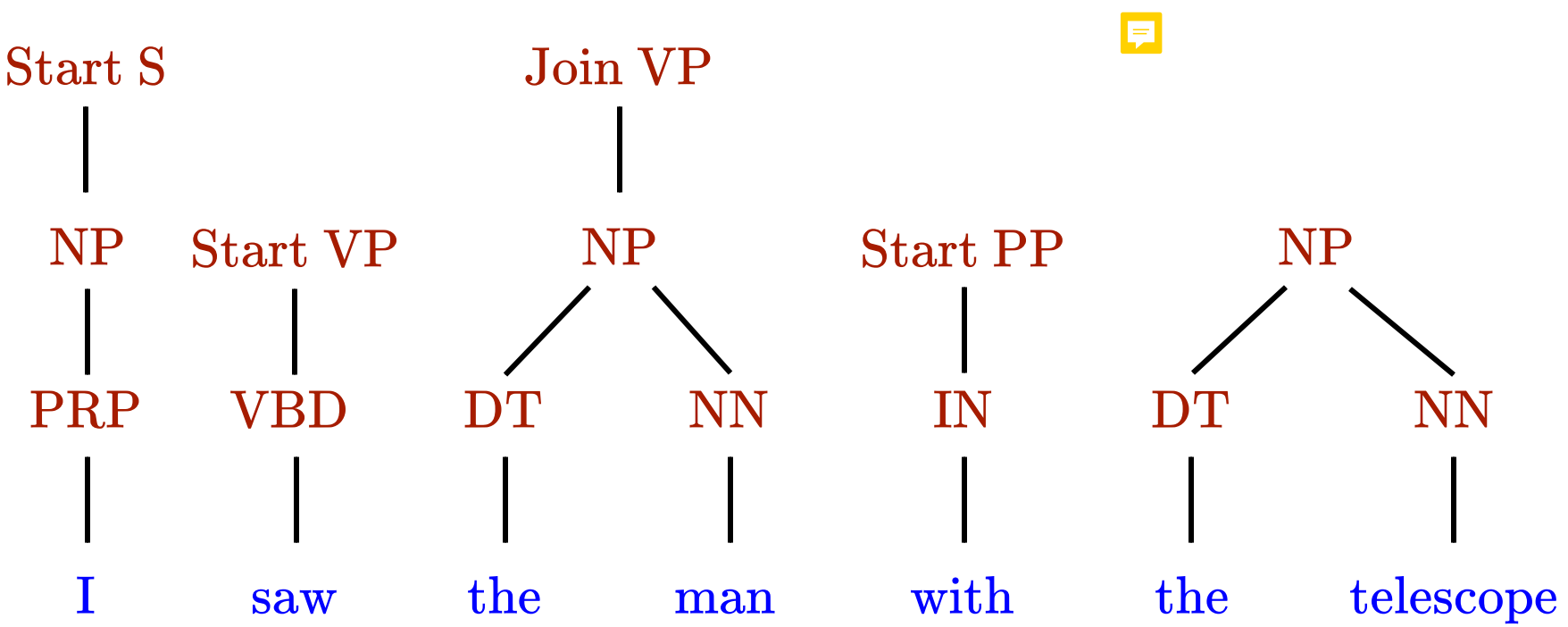
Incremental Parsing

Ok, if no, then move on to BUILD.



Incremental Parsing

We continue the process until a complete tree is constructed.



Question

How do we know which action to take in the process?

How do you know that it should be a 'No' in the check?

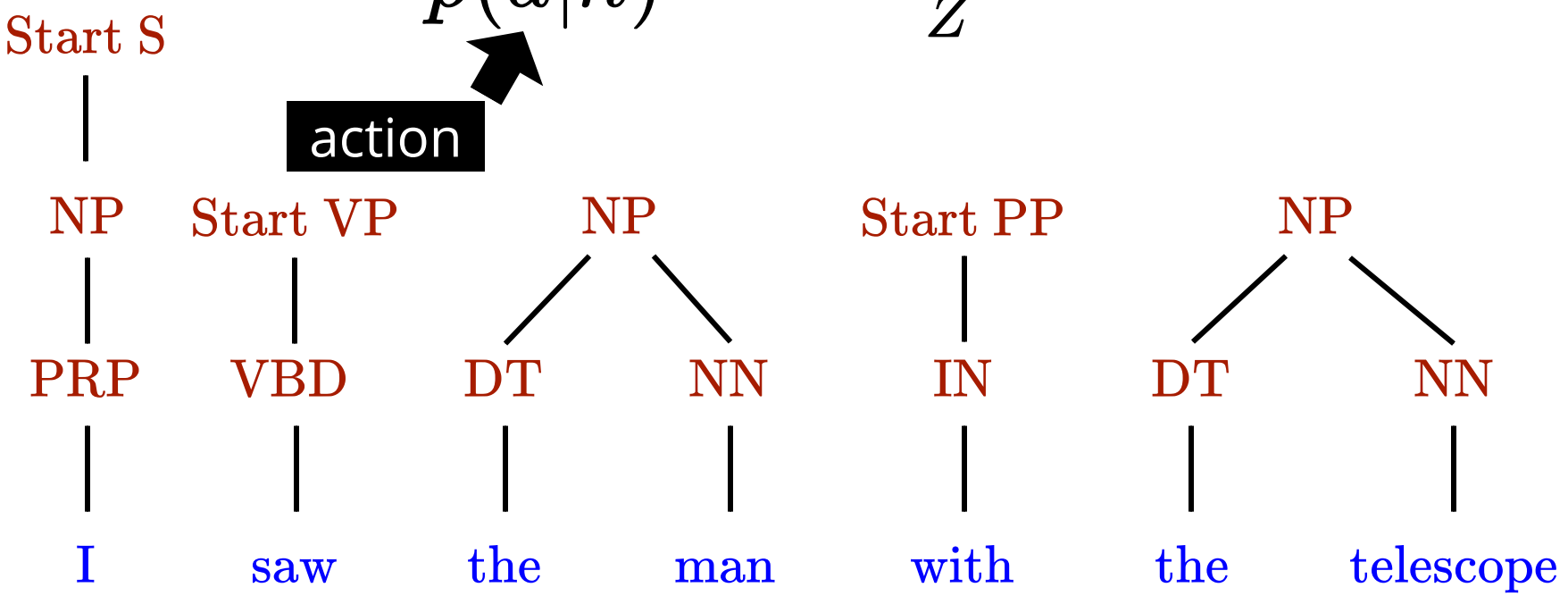
Incremental Parsing

Actions are scored with maximum entropy classifiers.

parse history

$$p(a|h) = \frac{\exp(\mathbf{f}(h,a))}{Z}$$

action



Optional

Incremental Parsing
(Collins & Roark, 2004)

Incremental Parsing with the Perceptron Algorithm

Michael Collins
MIT CSAIL
mcollins@csail.mit.edu

Brian Roark
AT&T Labs - Research
roark@research.att.com

Abstract

This paper describes an incremental parsing approach where parameters are estimated using a variant of the perceptron algorithm. A beam-search algorithm is used during both training and decoding phases of the method. The perceptron approach was implemented with the same feature set as that of an existing generative model (Roark, 2001a), and experimental results show that it gives competitive performance to the generative model on parsing the Penn treebank. We demonstrate that training a perceptron model to combine with the generative model during search provides a 2.1 percent F-measure improvement over the generative model alone, to 88.8 percent.

1 Introduction

In statistical approaches to NLP problems such as tagging or parsing, it seems clear that the representation used as input to a learning algorithm is central to the accuracy of an approach. In an ideal world, the designer of a parser or tagger would be free to choose any features which might be useful in discriminating good from bad structures, without concerns about how the features interact with the problems of training (parameter estimation) or decoding (search for the most plausible candidate under the model). To this end, a number of recently proposed methods allow a model to incorporate "arbitrary" global features of candidate analyses or parses. Examples of such techniques are Markov Random Fields (Ratnaparkhi et al., 1994; Abney, 1997; Della Pietra et al., 1997; Johnson et al., 1999), and boosting or perceptron approaches to reranking (Freund et al., 1998; Collins, 2000; Collins and Duffy, 2002).

A drawback of these approaches is that in the general case, they can require exhaustive enumeration of the set of candidates for each input sentence in both the training and decoding phases[1]. For example, Johnson et al. (1999) and Riezler et al. (2002) use all parses generated

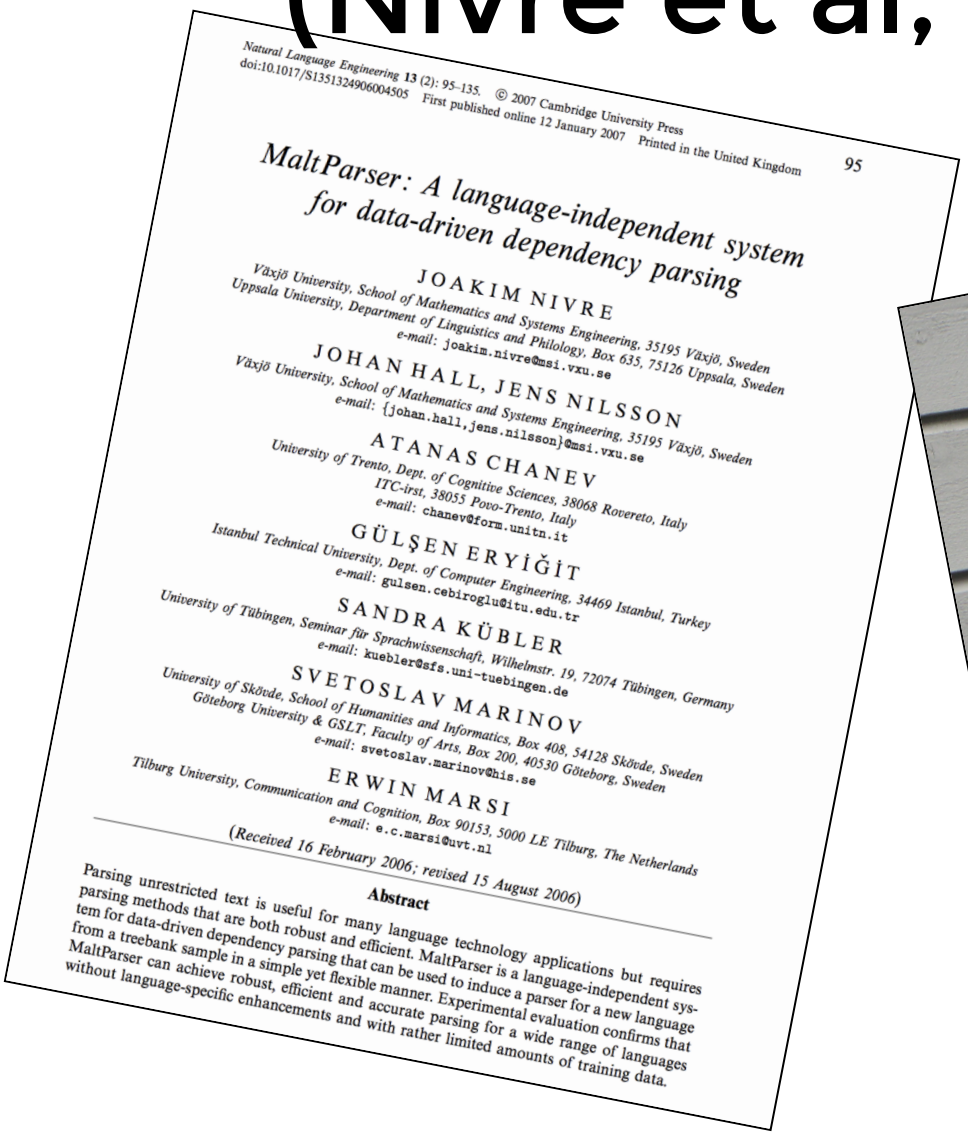
presupposes that there is an existing baseline model with reasonable performance. Many of these baseline models are themselves used with heuristic search techniques, so that the potential gain through the use of discriminative re-ranking techniques is further dependent on effective search.

This paper explores an alternative approach to parsing, based on the perceptron training algorithm introduced in Collins (2002). In this approach the training and decoding problems are very closely related – the training method decodes training examples in sequence, and makes simple corrective updates to the parameters when errors are made. Thus the main complexity of the method is isolated to the decoding problem. We describe an approach that uses an incremental, left-to-right parser, with beam search, to find the highest scoring analysis under the model. The same search method is used in both training and decoding. We implemented the perceptron approach with the same feature set as that of an existing generative model (Roark, 2001a), and show that the perceptron model gives performance competitive to that of the generative model on parsing the Penn treebank, thus demonstrating that an unnormalized discriminative parsing model can be applied with heuristic search. We also describe several refinements to the training algorithm, and demonstrate their impact on convergence properties of the method.

Finally, we describe training the perceptron model with the negative log probability given by the generative model as another feature. This provides the perceptron algorithm with a better starting point, leading to large improvements over using either the generative model or the perceptron algorithm in isolation (the hybrid model achieves 88.8% f-measure on the WSJ treebank, compared to figures of 86.7% and 86.6% for the separate generative and perceptron models). The approach is an extremely simple method for integrating new features into the generative model: essentially all that is needed is a definition of feature-vector representations of entire

Similar idea. Also applied to constituency parsing, but makes use of structured perceptron with partial n-best outputs.

Incremental Parsing (Nivre et al, 2007)



Natural Language Engineering 13 (2): 95–135. © 2007 Cambridge University Press
doi:10.1017/S1351324906004505 First published online 12 January 2007 Printed in the United Kingdom

MaltParser: A language-independent system for data-driven dependency parsing

JOAKIM NIVRE
Växjö University, School of Mathematics and Systems Engineering, 35195 Växjö, Sweden
Uppsala University, Department of Linguistics and Philology, Box 635, 75126 Uppsala, Sweden
e-mail: joakim.nivre@msi.vxu.se

JOHAN HALL, JENS NILSSON
Växjö University, School of Mathematics and Systems Engineering, 35195 Växjö, Sweden
e-mail: {johan.hall,jens.nilsson}@msi.vxu.se

ATANAS CHANEV
University of Trento, Dept. of Cognitive Sciences, 38068 Rovereto, Italy
ITC-irst, 38055 Povo-Trento, Italy
e-mail: chanev@form.unitn.it

GÜLŞEN ERYİĞİT
Istanbul Technical University, Dept. of Computer Engineering, 34469 Istanbul, Turkey
e-mail: gulsen.cebiroglu@itu.edu.tr

SANDRA KÜBLER
University of Tübingen, Seminar für Sprachwissenschaft, Wilhelmstr. 19, 72074 Tübingen, Germany
e-mail: kuebler@sfs.uni-tuebingen.de

SVETOSLAV MARINOV
University of Skövde, School of Humanities and Informatics, Box 408, 54128 Skövde, Sweden
Göteborg University & GSLT, Faculty of Arts, Box 200, 40530 Göteborg, Sweden
e-mail: svetoslav.marinov@his.se

ERWIN MARSI
Tilburg University, Communication and Cognition, Box 90153, 5000 LE Tilburg, The Netherlands
e-mail: e.c.marsi@uvt.nl

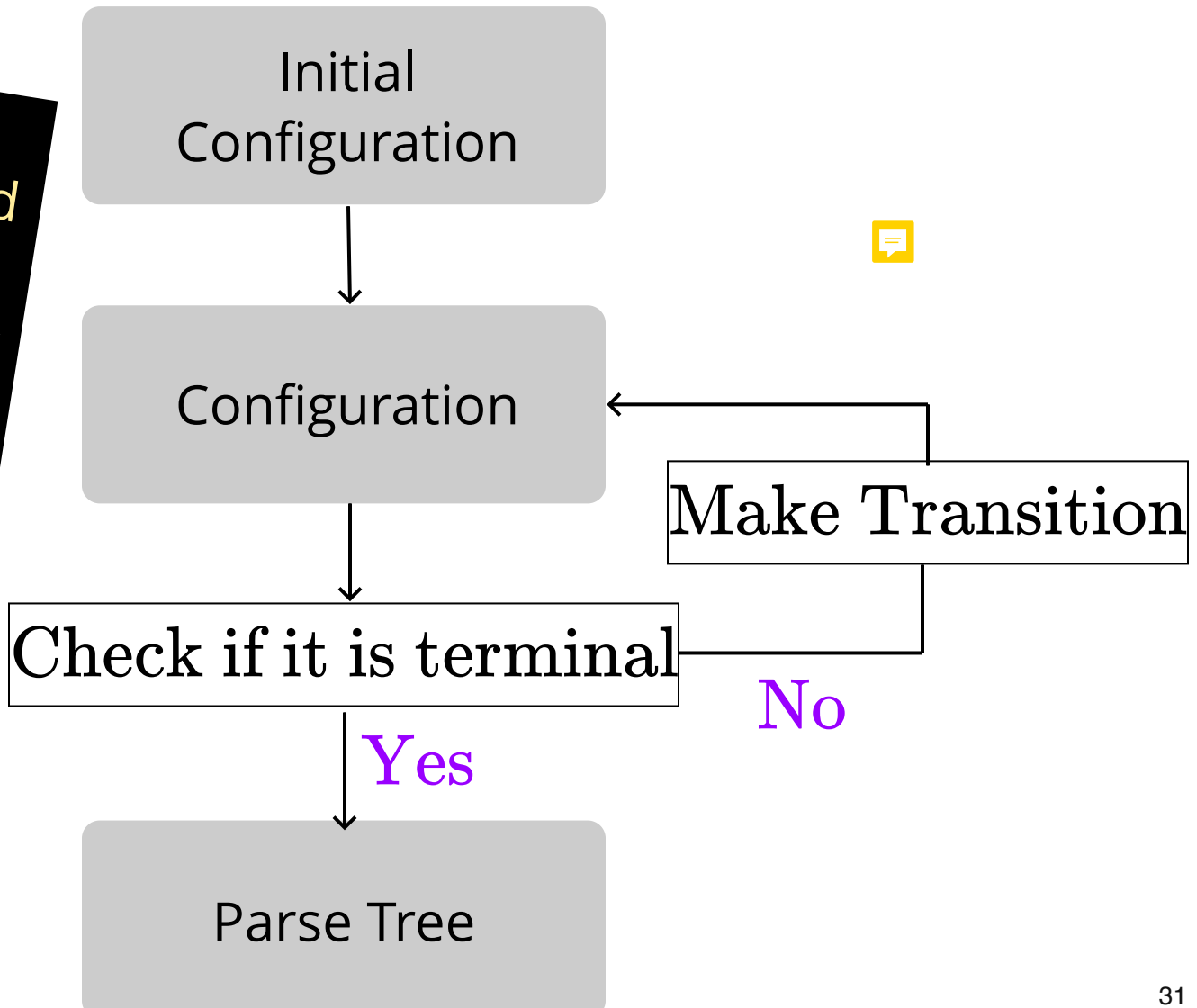
(Received 16 February 2006; revised 15 August 2006)

Abstract

Parsing unrestricted text is useful for many language technology applications but requires parsing methods that are both robust and efficient. MaltParser is a language-independent system for data-driven dependency parsing that can be used to induce a parser for a new language from a treebank sample in a simple yet flexible manner. Experimental evaluation confirms that MaltParser can achieve robust, efficient and accurate parsing for a wide range of languages without language-specific enhancements and with rather limited amounts of training data.

Incremental Parsing

Also called "Transition-based Parsing". The resulting parser is also similar to a "Shift-Reduce" Parser



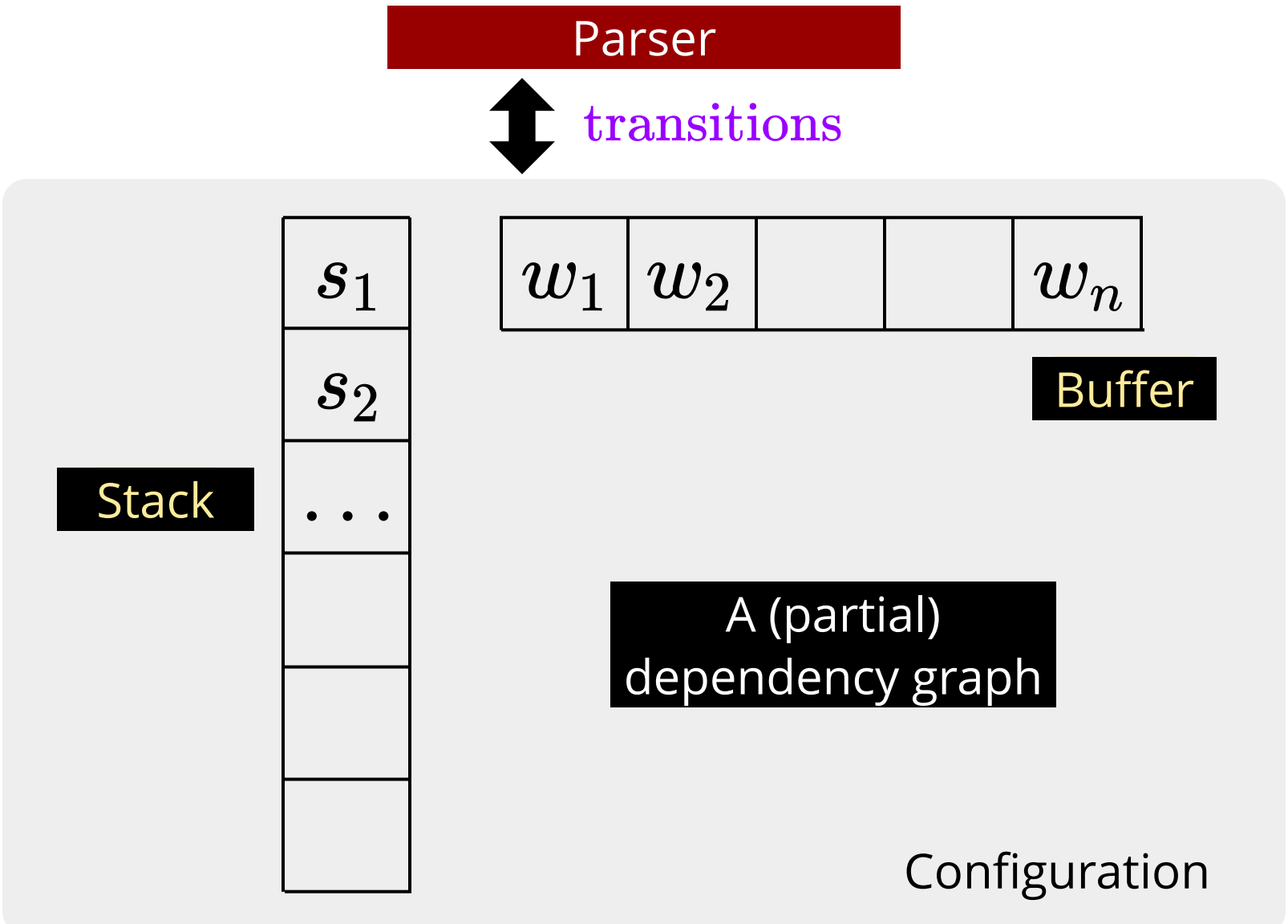
Transition-based Parsing

Configuration

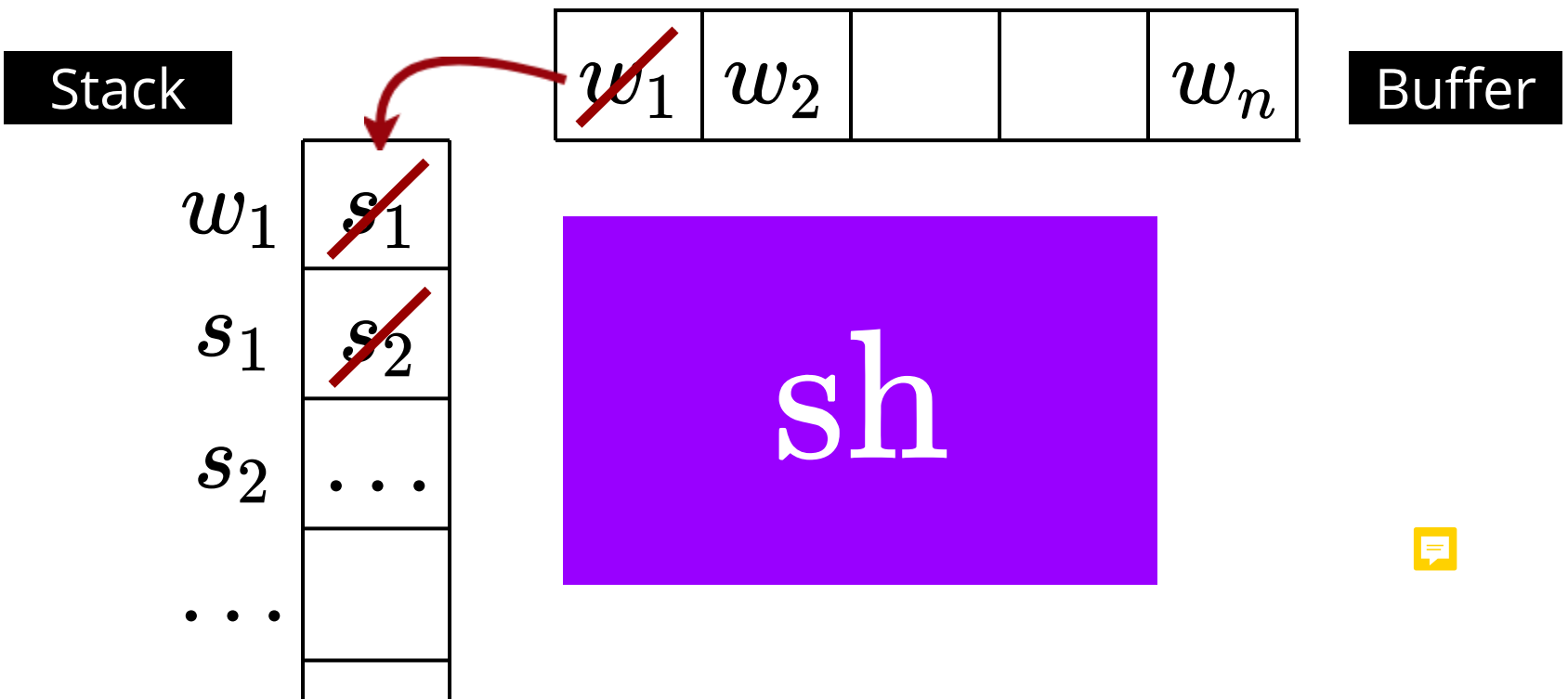
A state in the parsing process.

Now let us consider a specific algorithm
for dependency parsing

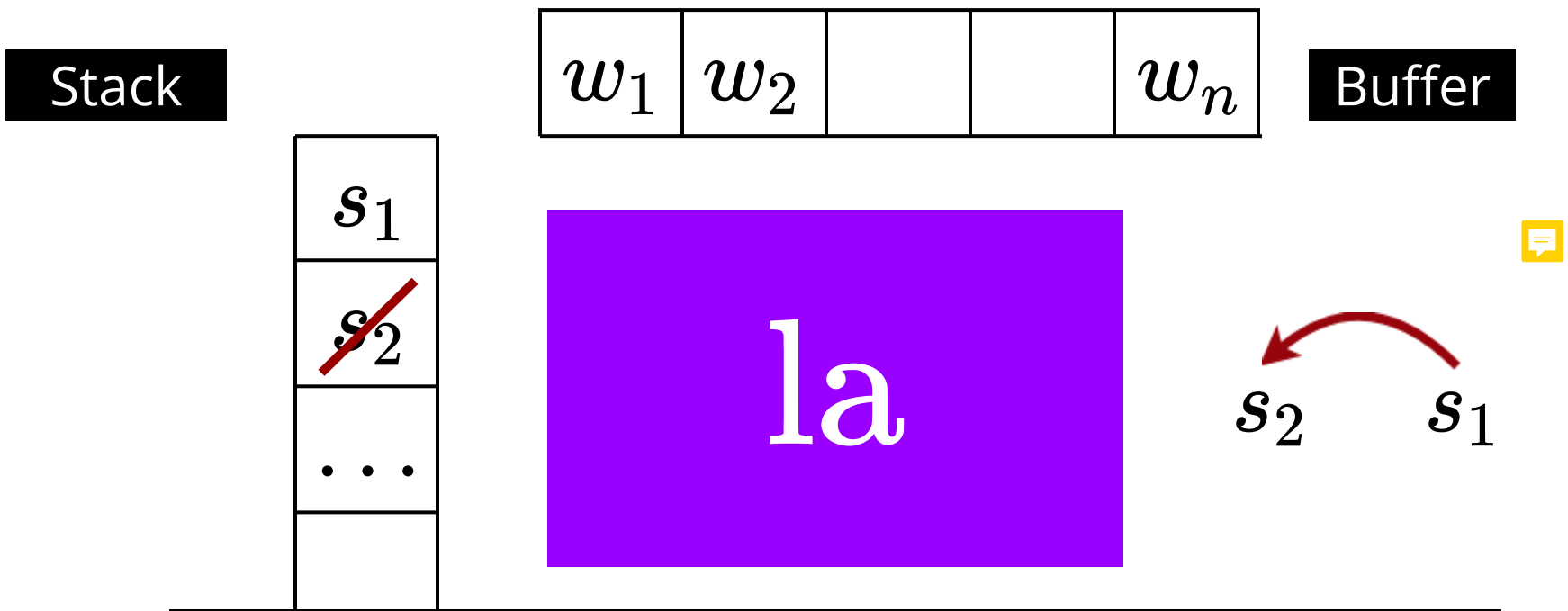
Transition-based Parsing



Transitions



Transitions



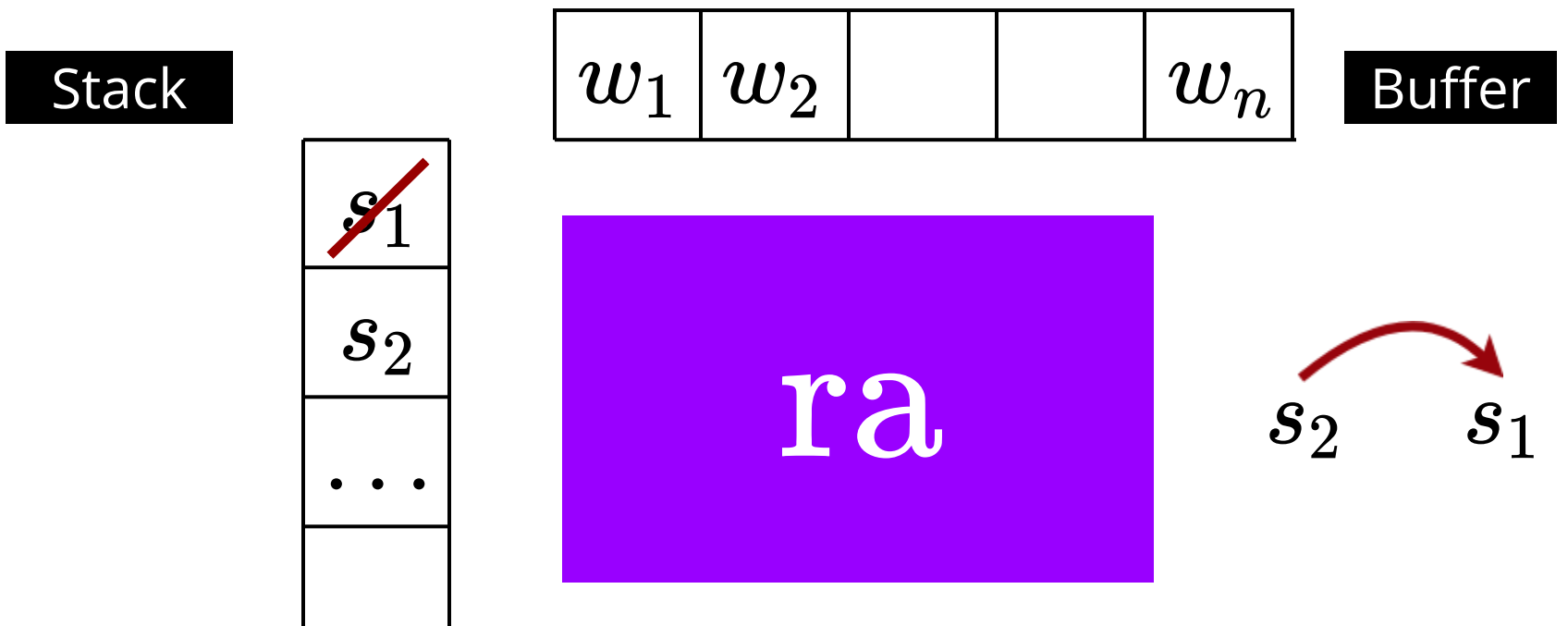
(left-arc)

add an arc

from the topmost word on the stack, s_1

to the second-topmost word, s_2, and pop s_2

Transitions



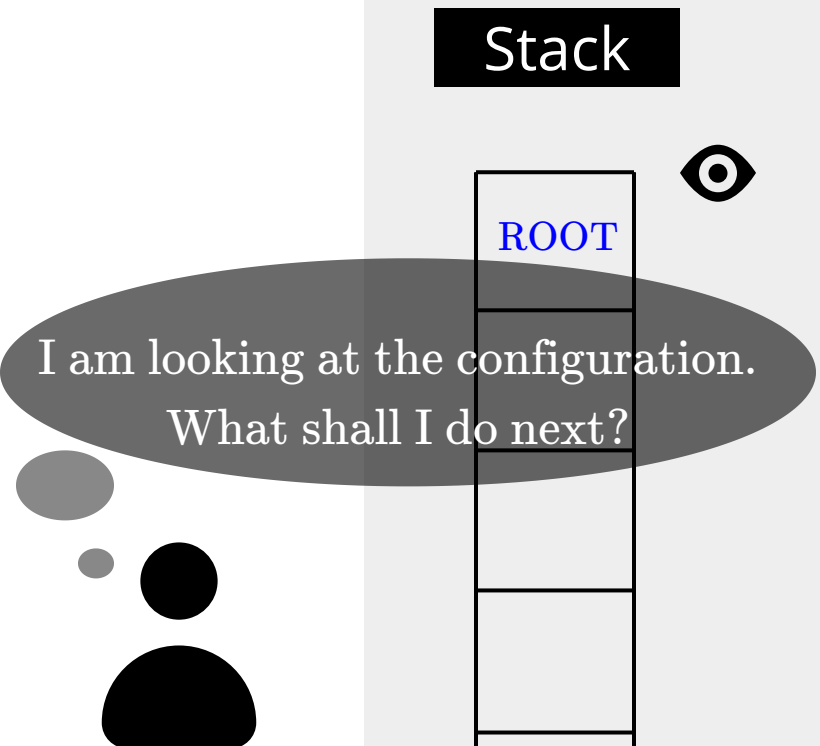
(right-arc)

add an arc

from the second-topmost word on the stack, s_2

to the topmost word, s_1, and pop s_1

Transition-based Parsing



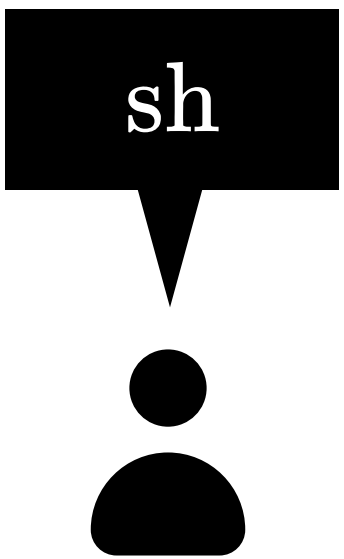
Stack

Buffer

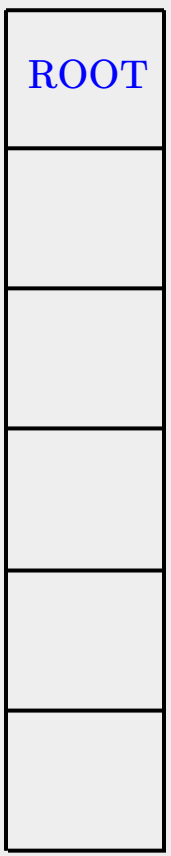
Book	me	the	morning	flight

Book me the morning flight

Transition-based Parsing



Stack

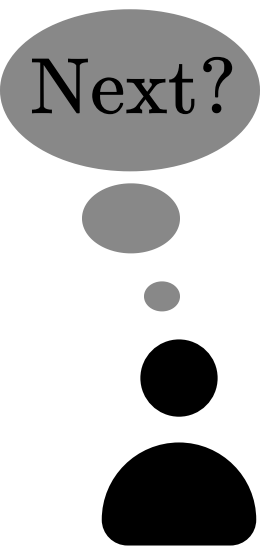


Buffer

Book	me	the	morning	flight

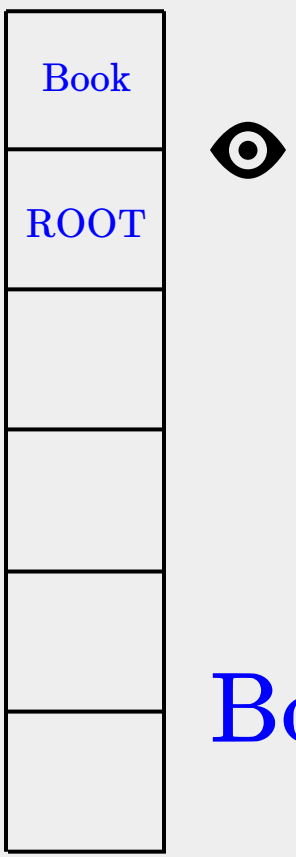
Book me the morning flight

Transition-based Parsing



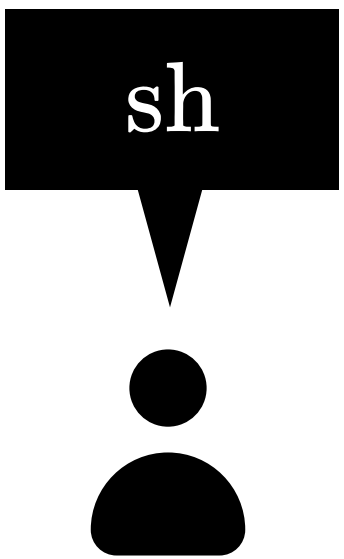
Stack

Buffer

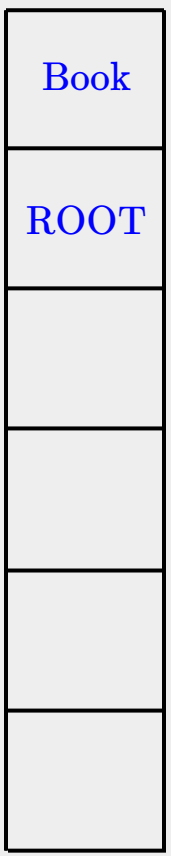


Book me the morning flight

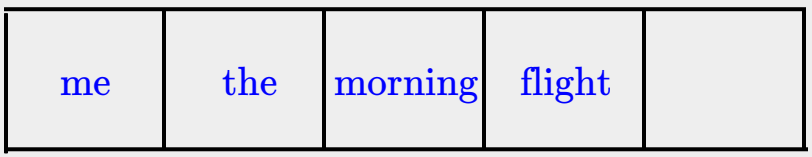
Transition-based Parsing



Stack

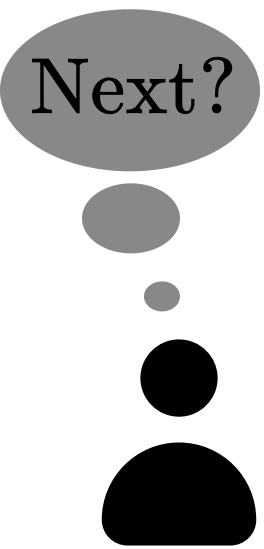


Buffer

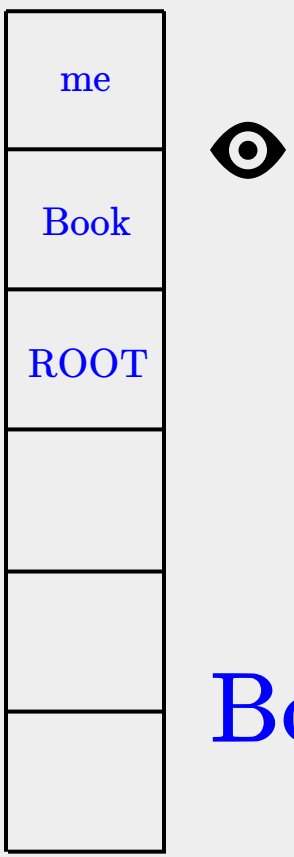


Book me the morning flight

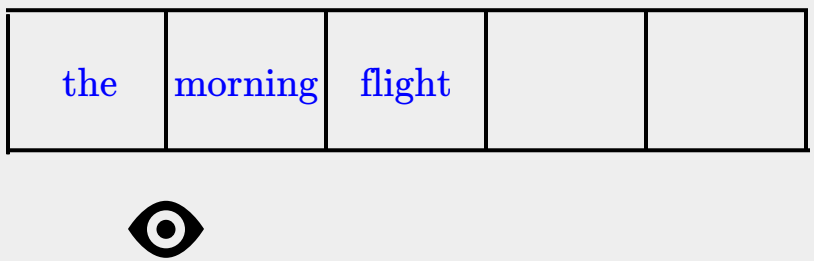
Transition-based Parsing



Stack

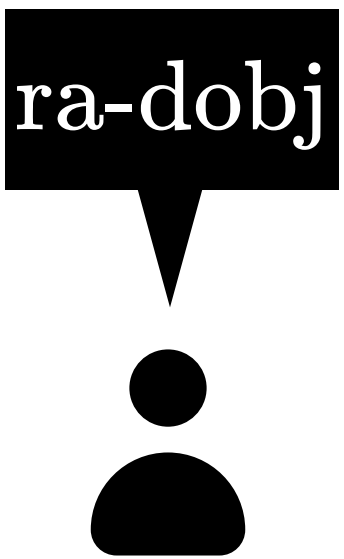


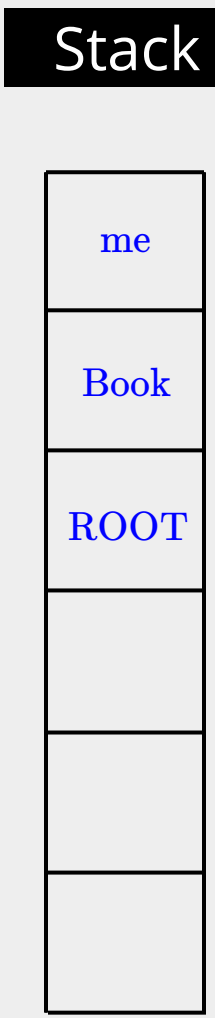
Buffer

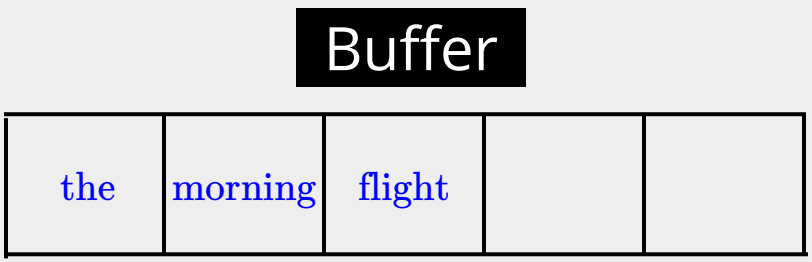


Book me the morning flight

Transition-based Parsing

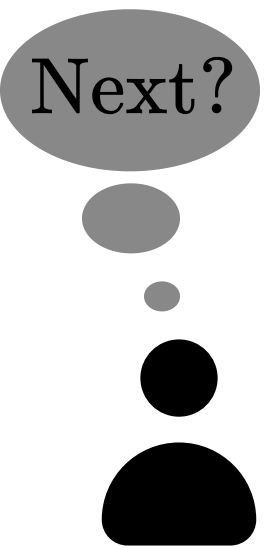






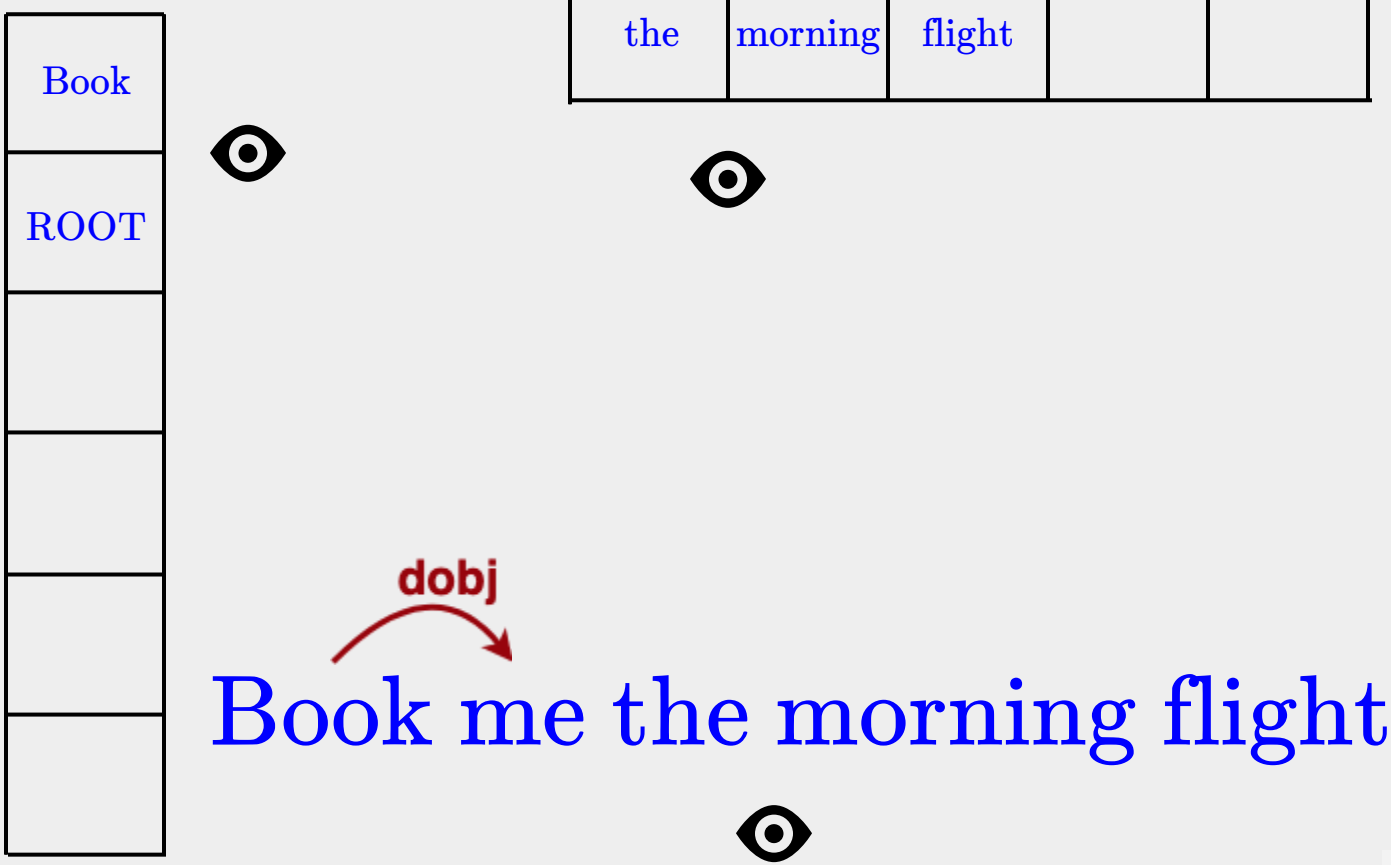
Book me the morning flight

Transition-based Parsing

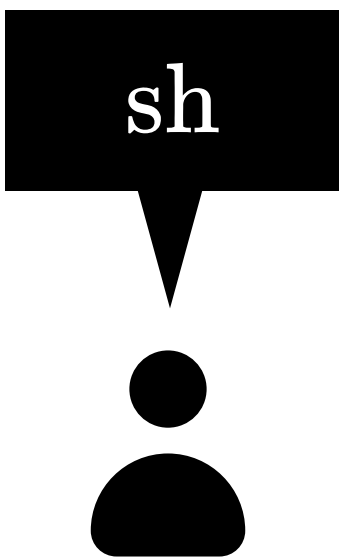


Stack

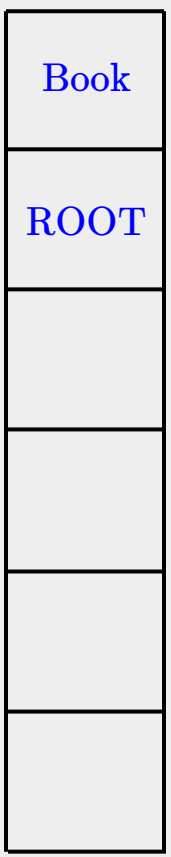
Buffer



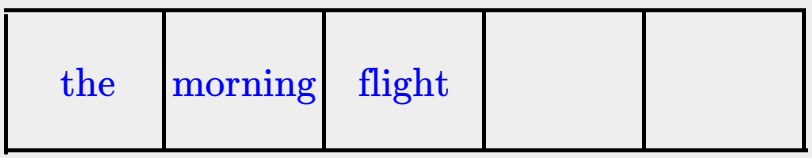
Transition-based Parsing



Stack

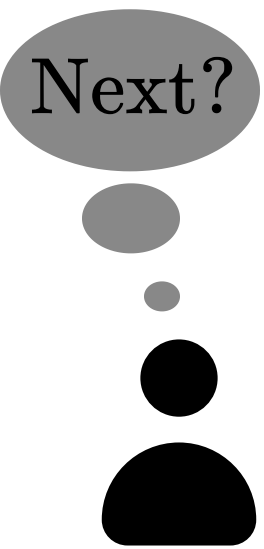


Buffer



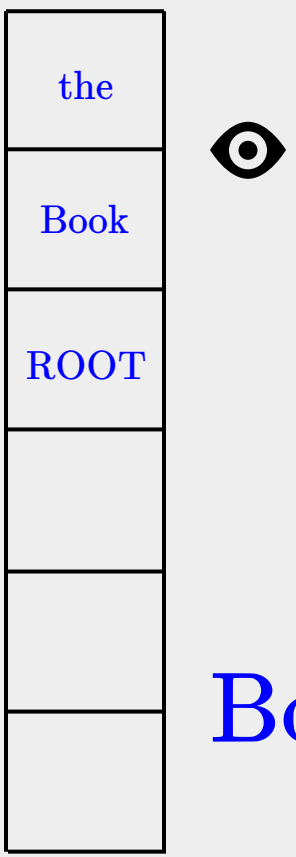
Book me the morning flight

Transition-based Parsing



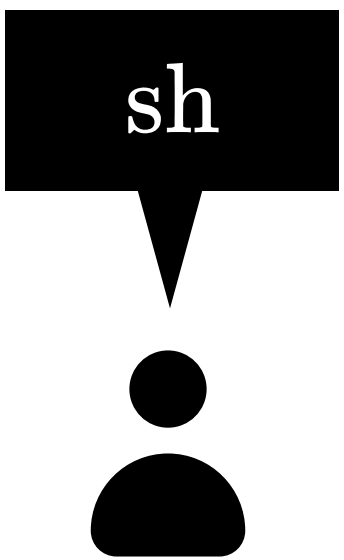
Stack

Buffer

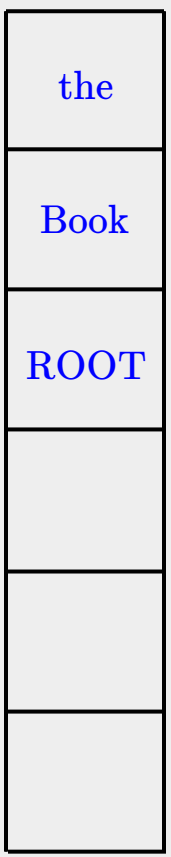


Book me the morning flight

Transition-based Parsing



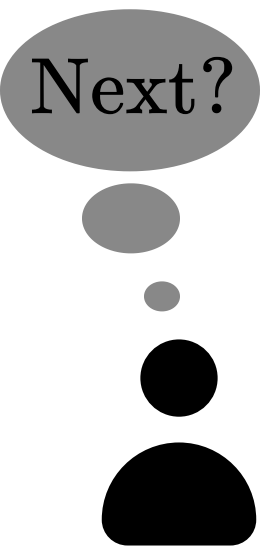
Stack



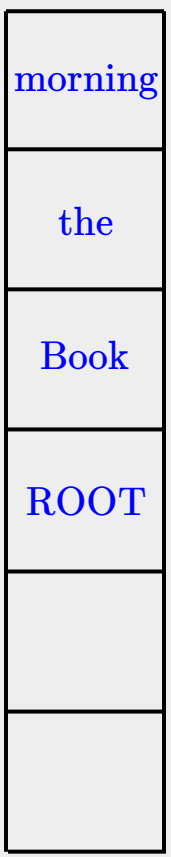
Buffer

Book me the morning flight

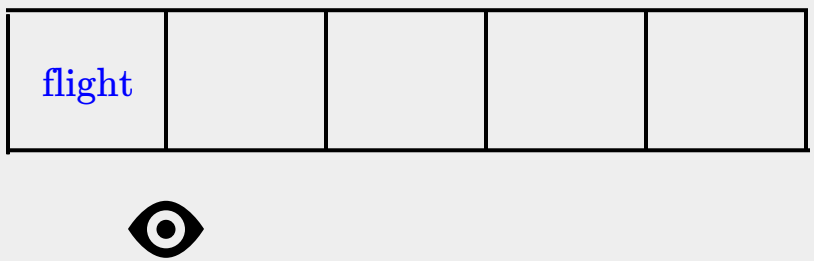
Transition-based Parsing



Stack



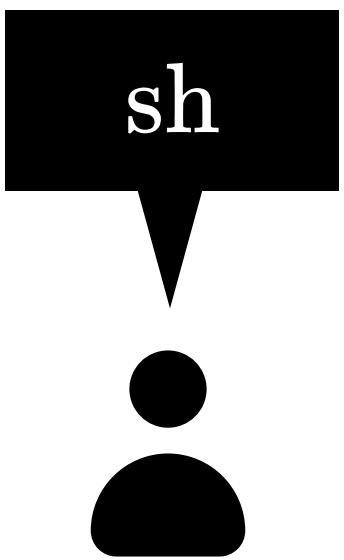
Buffer



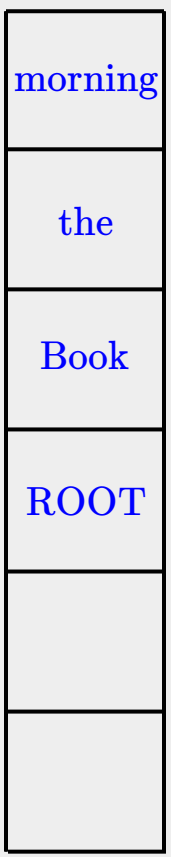
dobj

Book me the morning flight

Transition-based Parsing



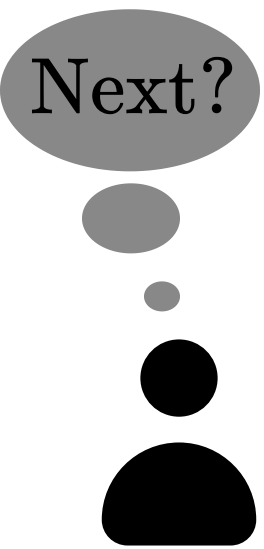
Stack



Buffer

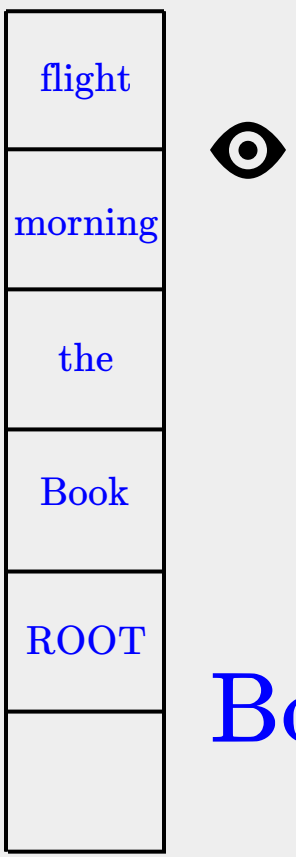
Book me the morning flight

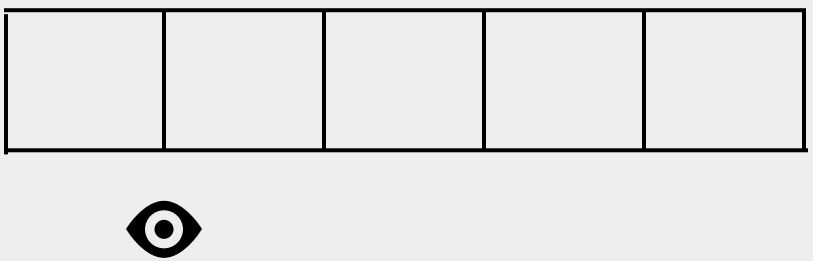
Transition-based Parsing



Stack

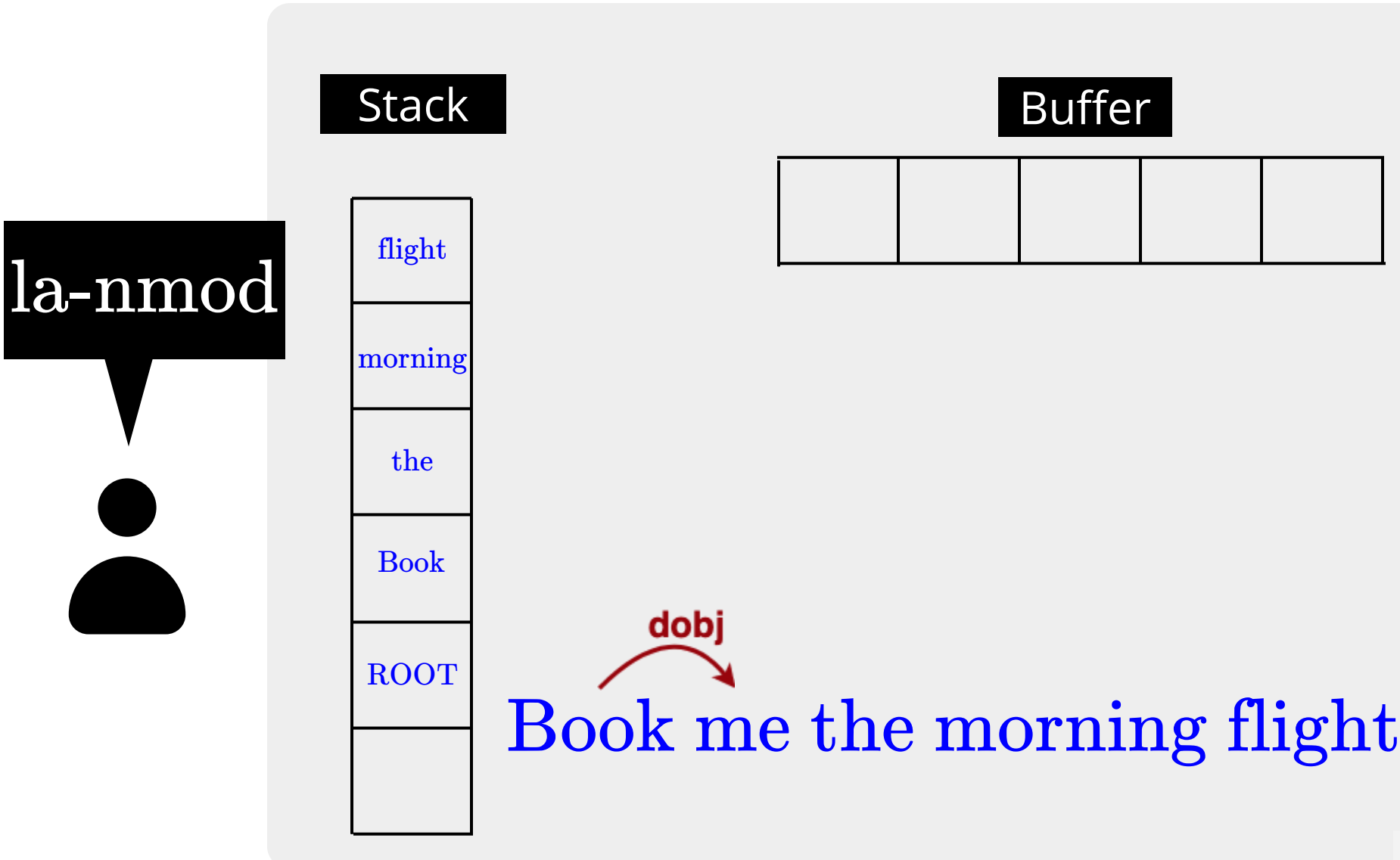
Buffer



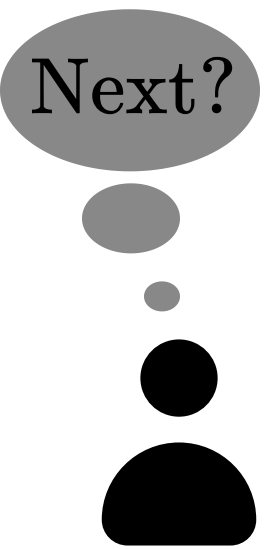


Book me the morning flight

Transition-based Parsing

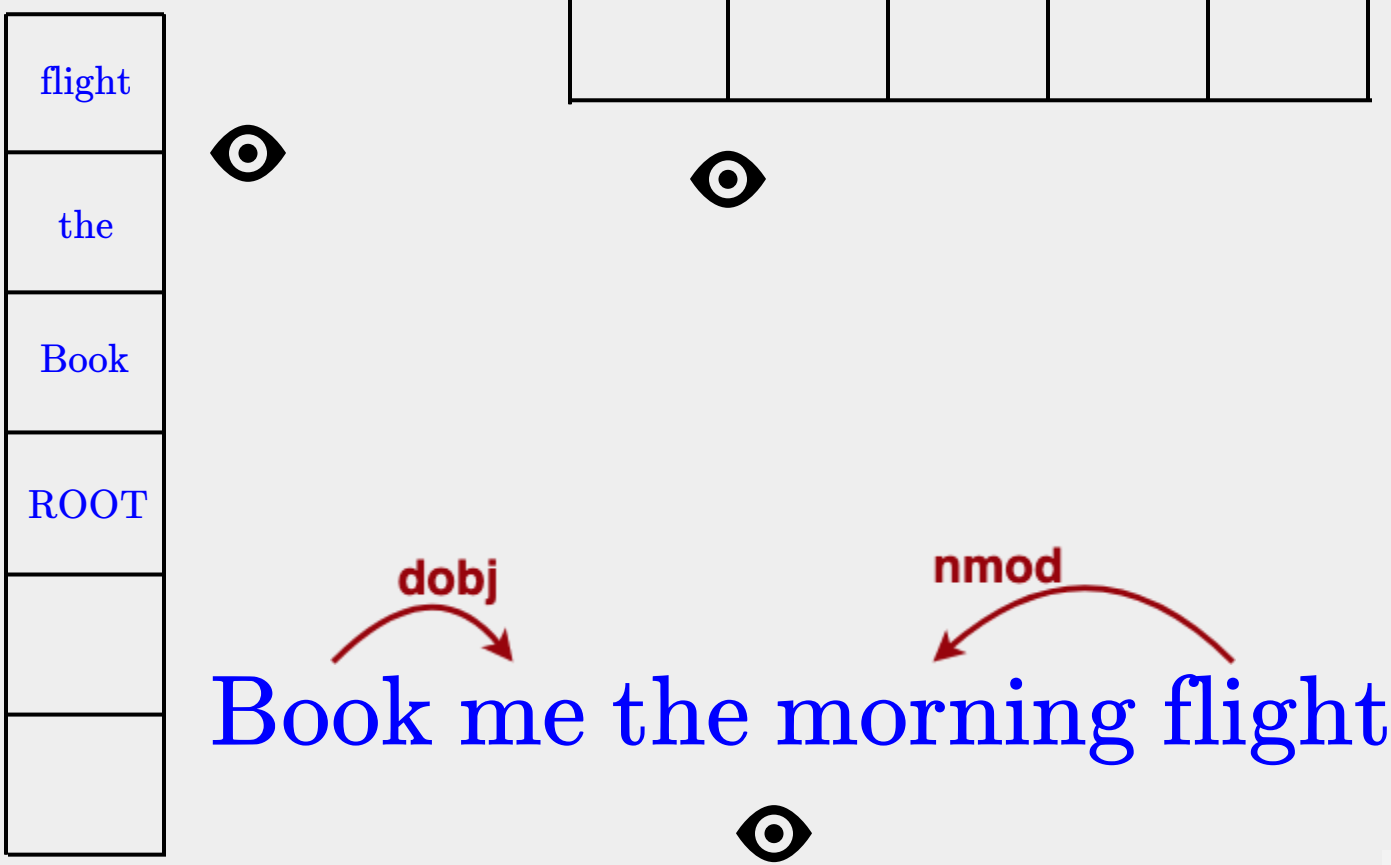


Transition-based Parsing

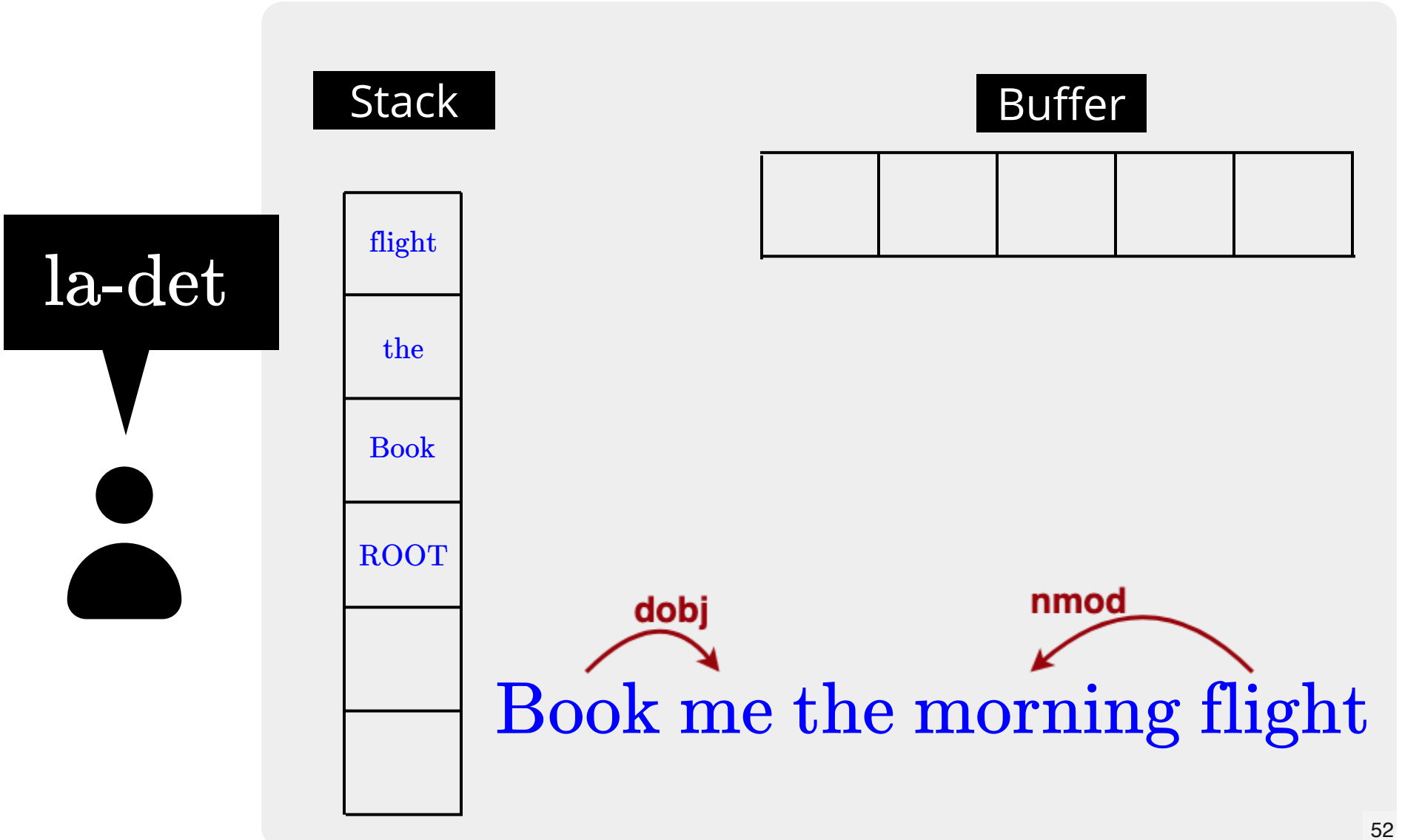


Stack

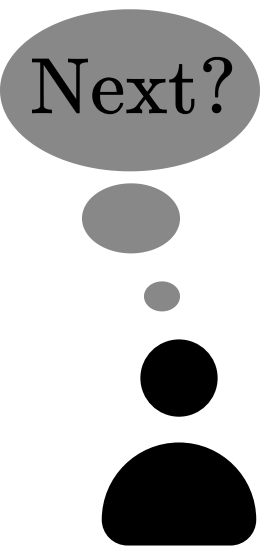
Buffer

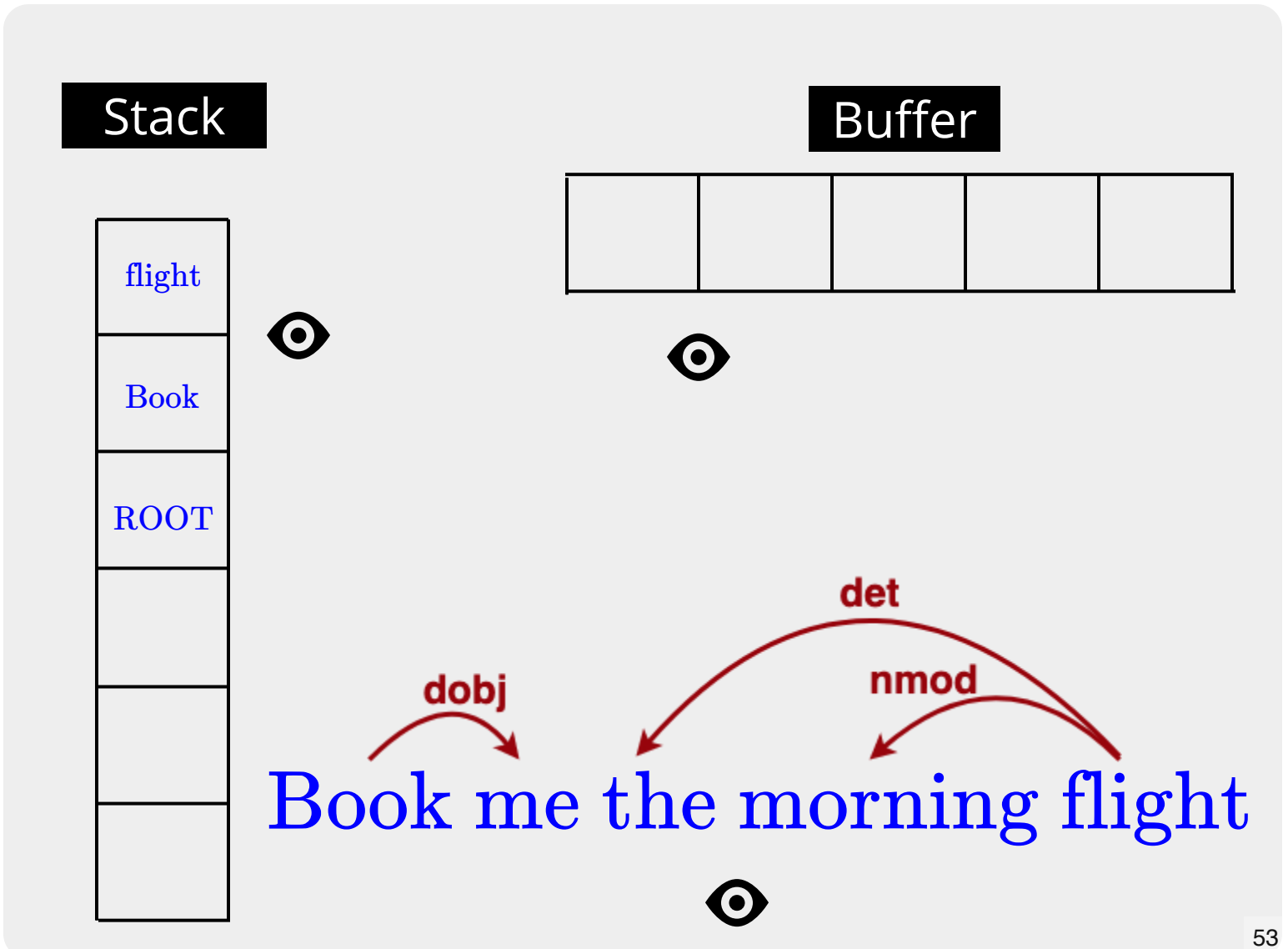


Transition-based Parsing

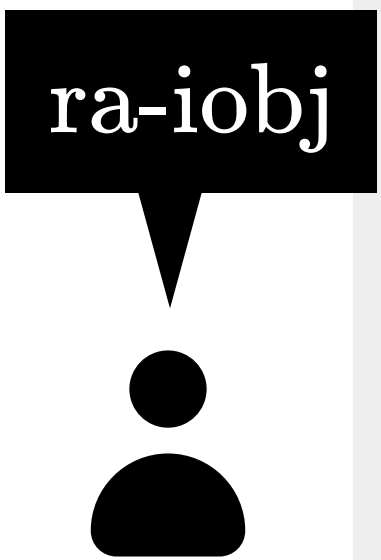


Transition-based Parsing



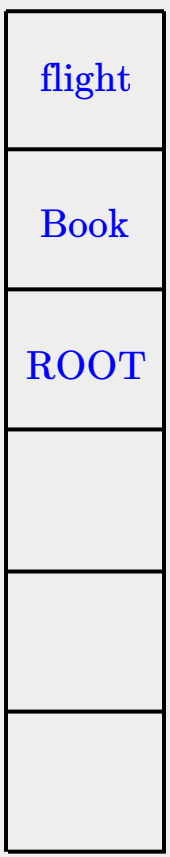


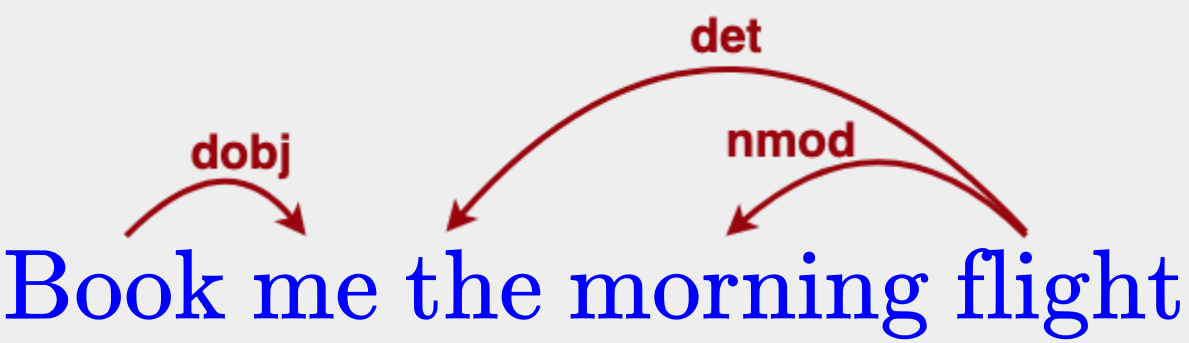
Transition-based Parsing



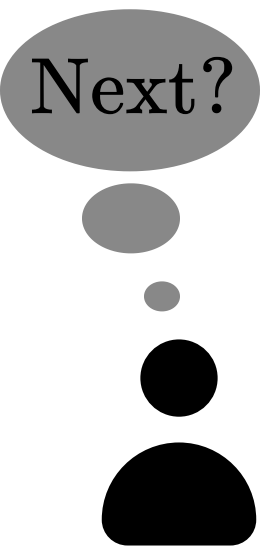
Stack

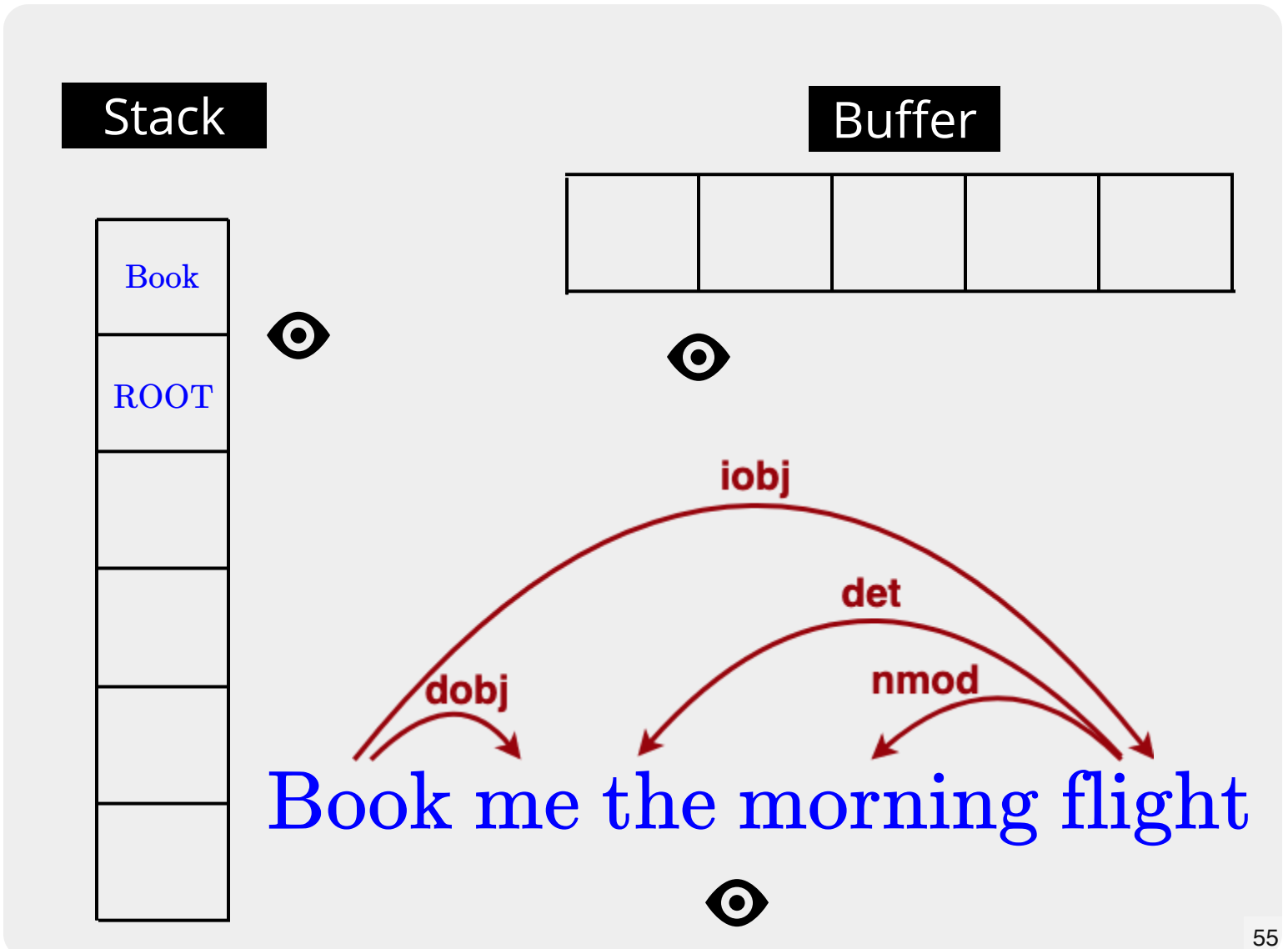
Buffer



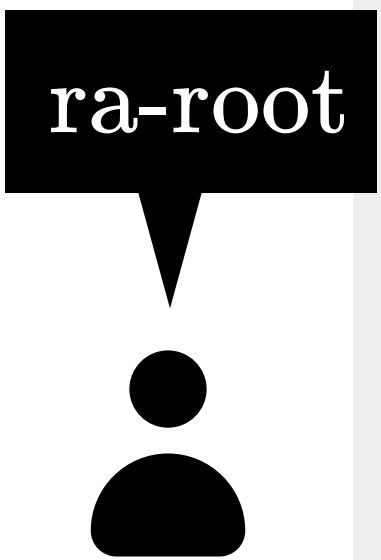


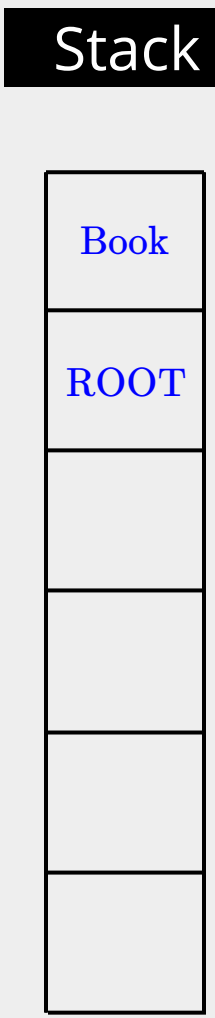
Transition-based Parsing

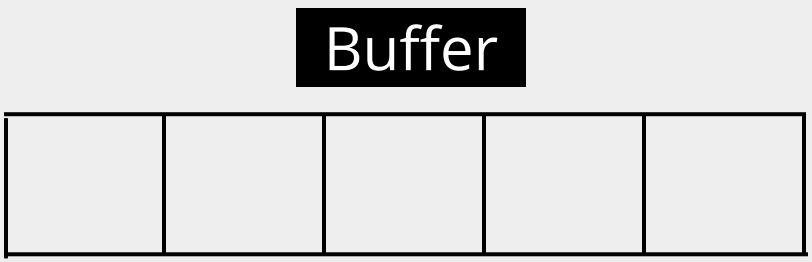


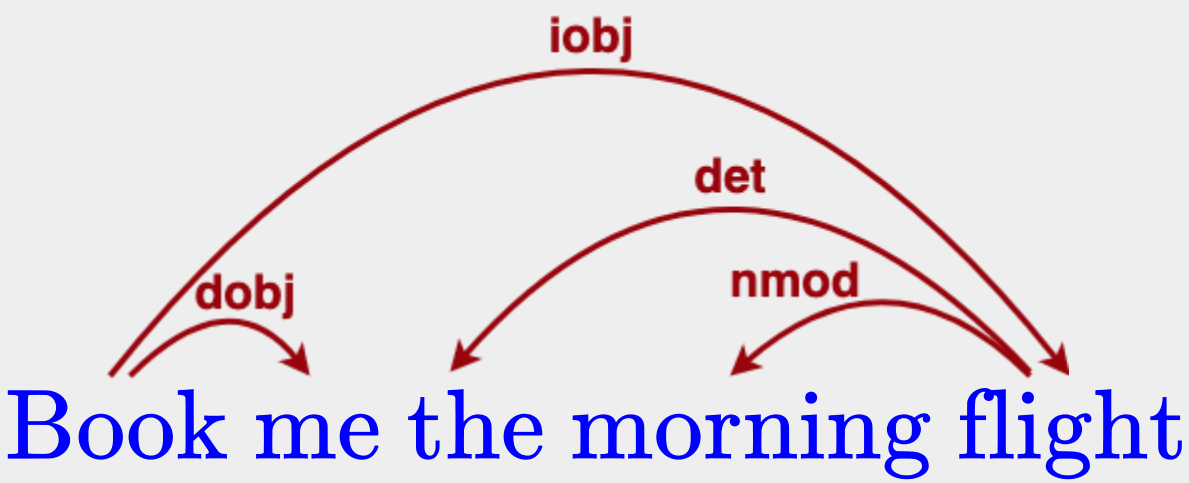


Transition-based Parsing

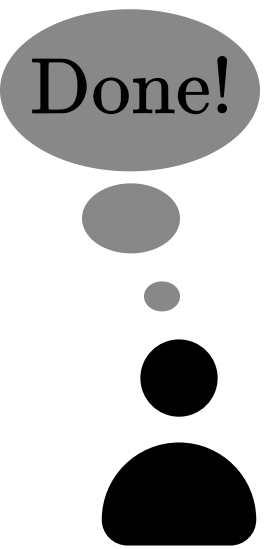


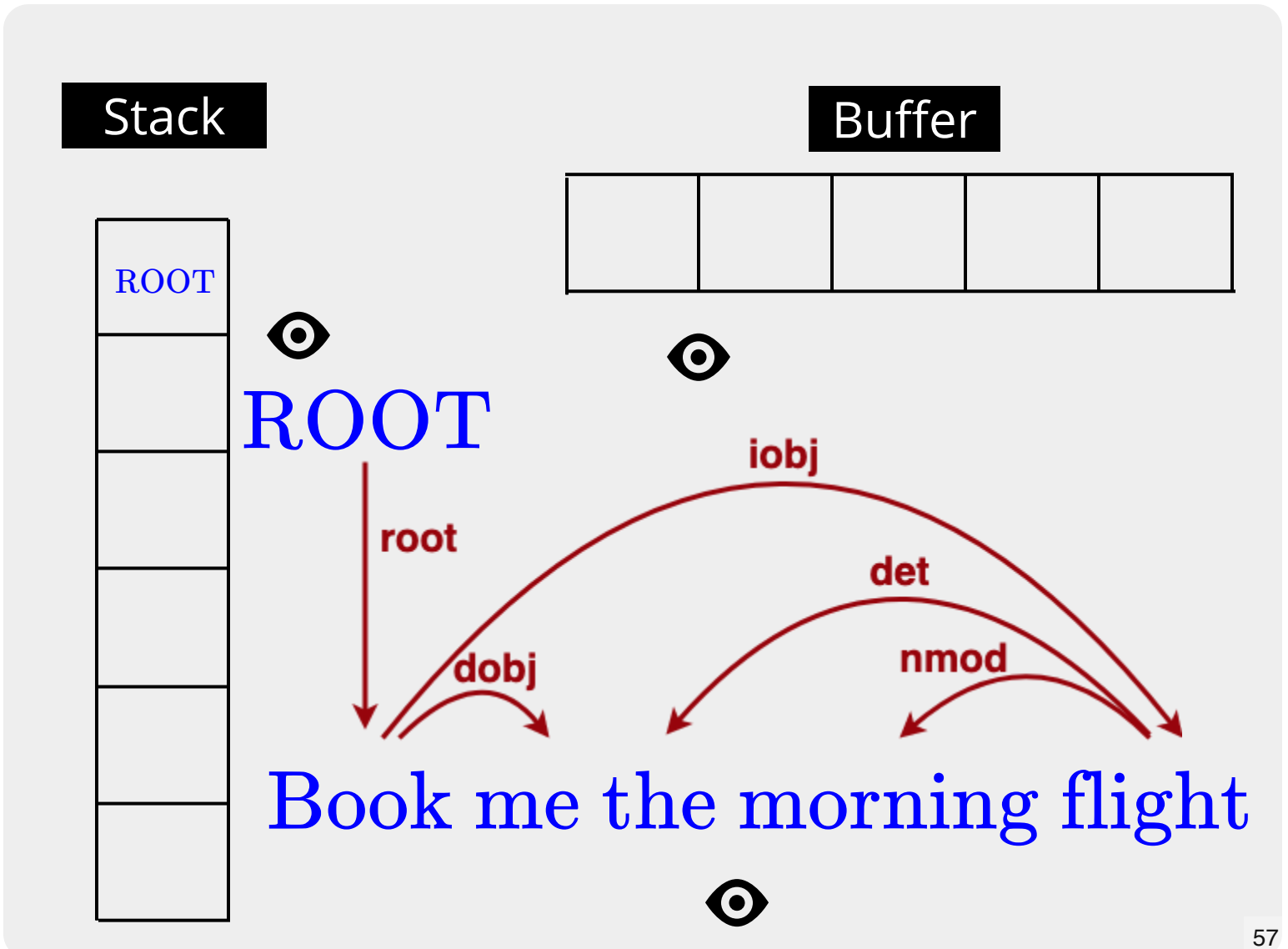






Transition-based Parsing

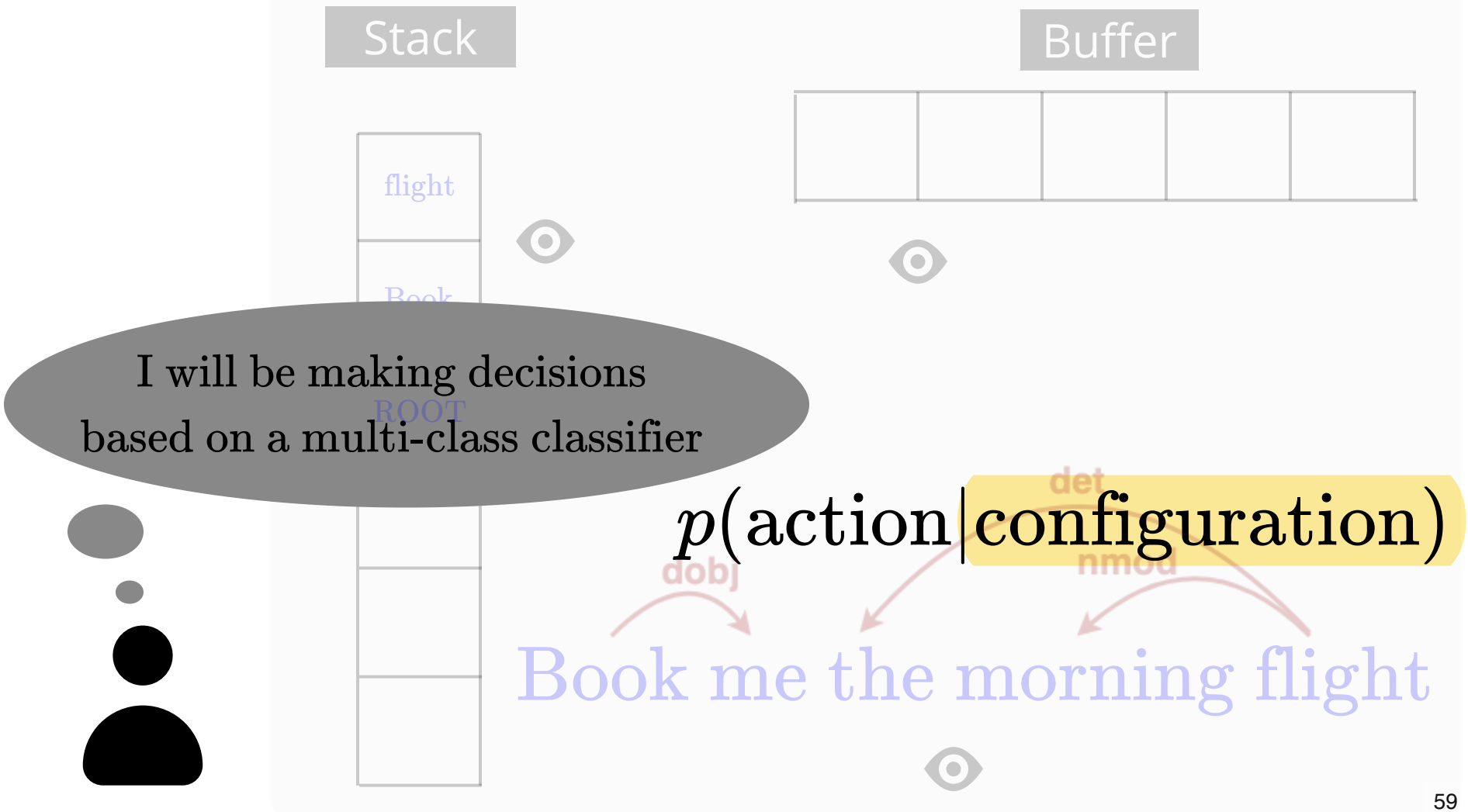




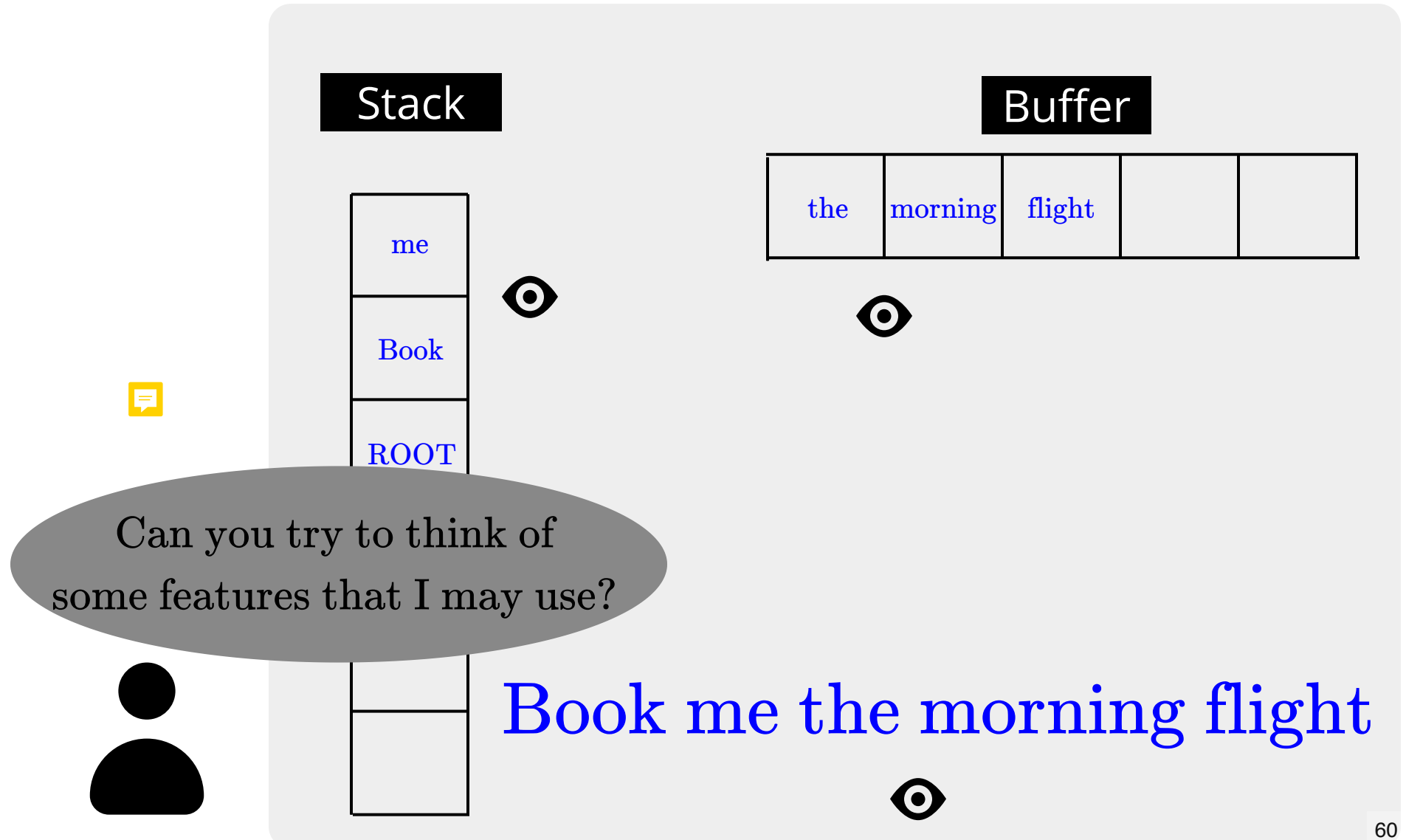
Question

How to know which action to take?

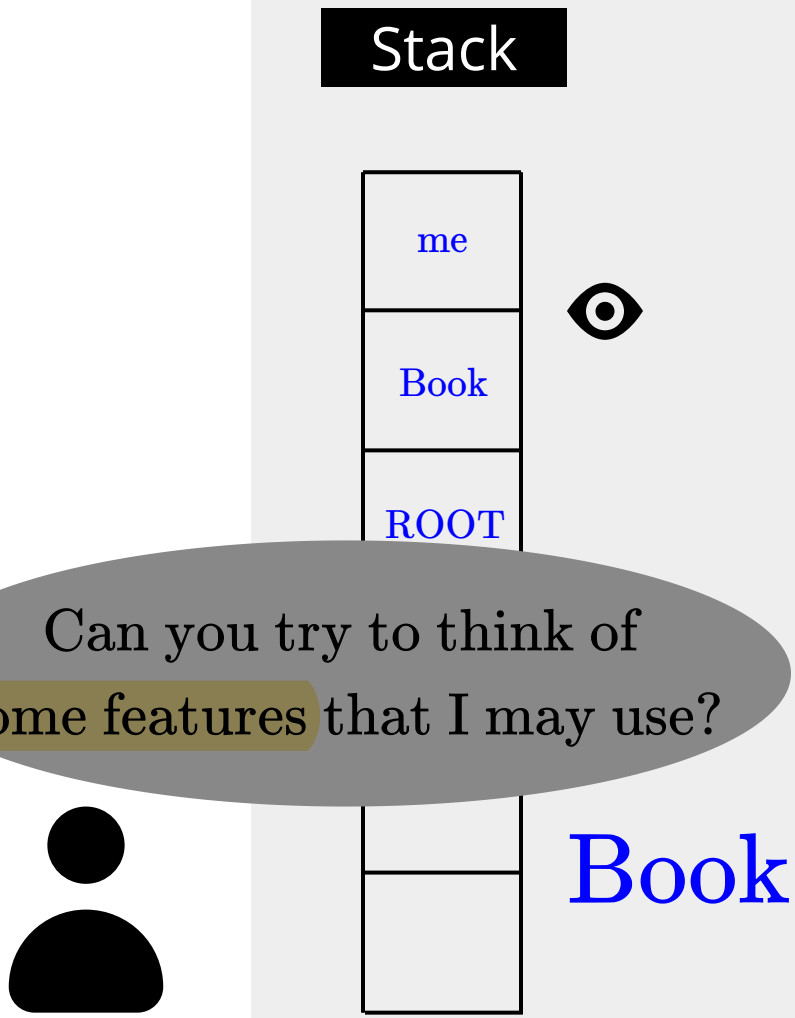
Transition-based Parsing

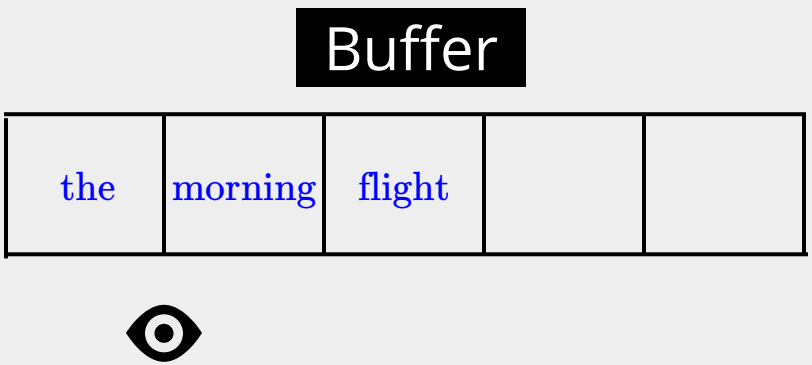


Transition-based Parsing



Transition-based Parsing





Empirically, we use word forms, POS tags and their combinations for words that appear at the top of stack and at head of the buffer. May also consider the existing dependency arcs attached to such words as part of feature.

Can you try to think of some features that I may use?

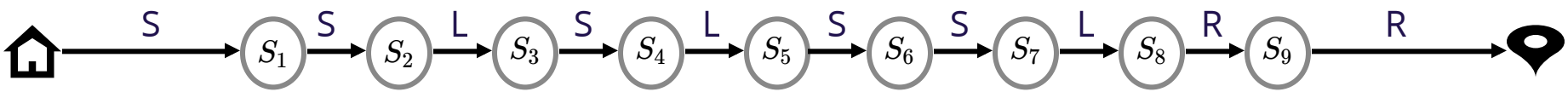
Book me the morning flight

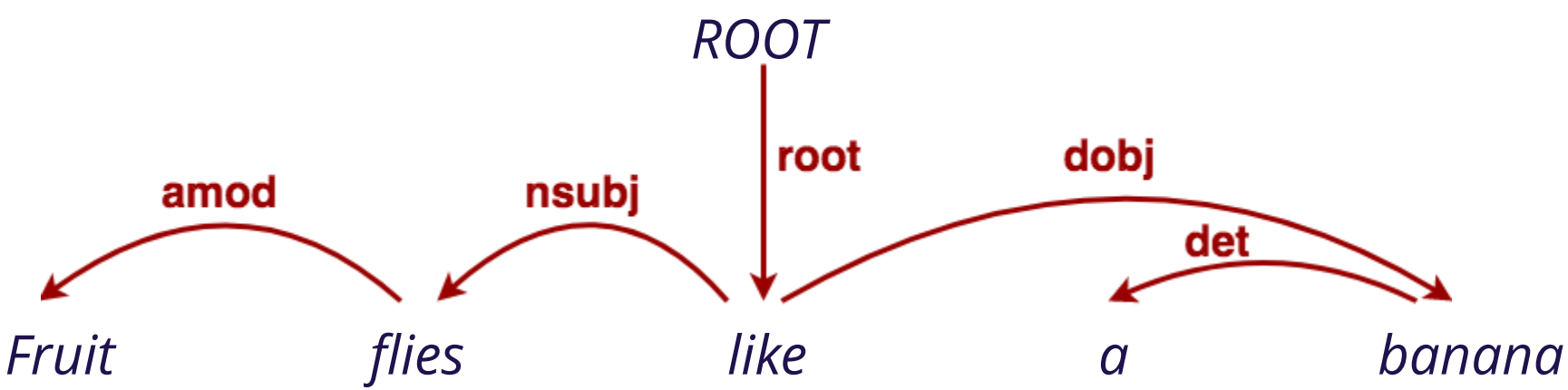
Summary

1. The algorithm comes with a linear time complexity.
2. The model is greedy, and therefore may have error-propagation issue.
3. The distinctive advantage of the model is being able to exploit rich non-local features.
4. Alternative designs exist. Possible to extend it to support non-projective parsing (with SWAP actions, and higher worst case complexity)

Optional

Graph vs Transition

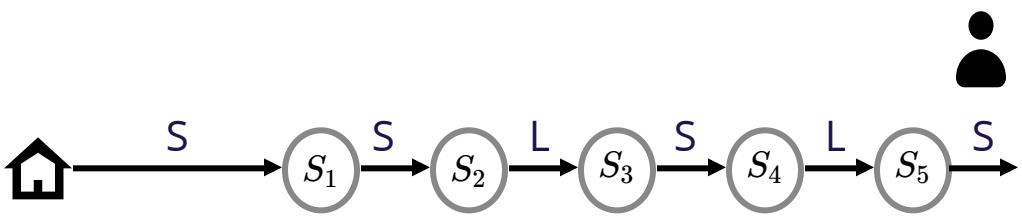


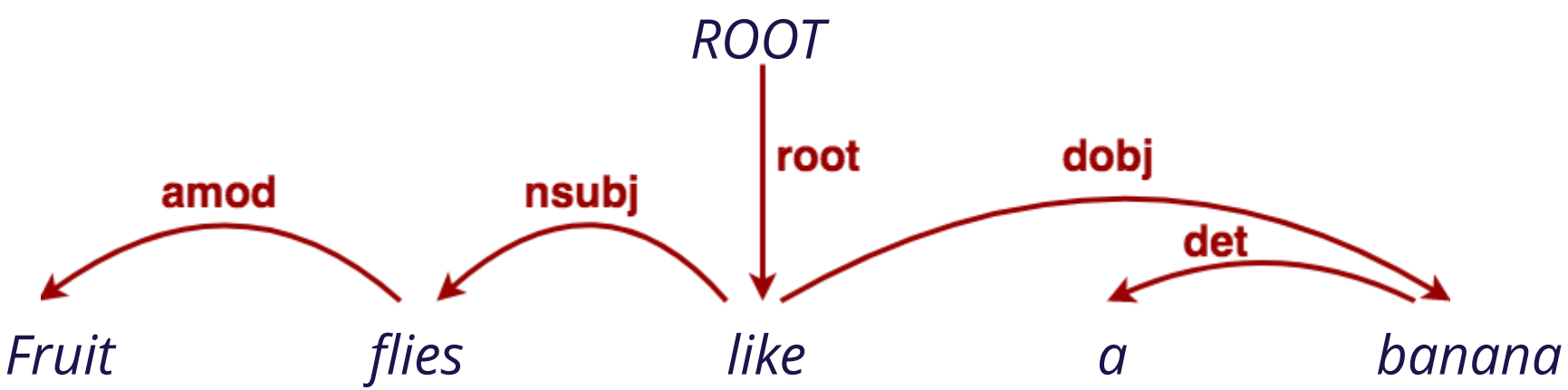


Optional

Transitions, States

$$S : \text{sh}, L : \text{la}, R : \text{ra}$$

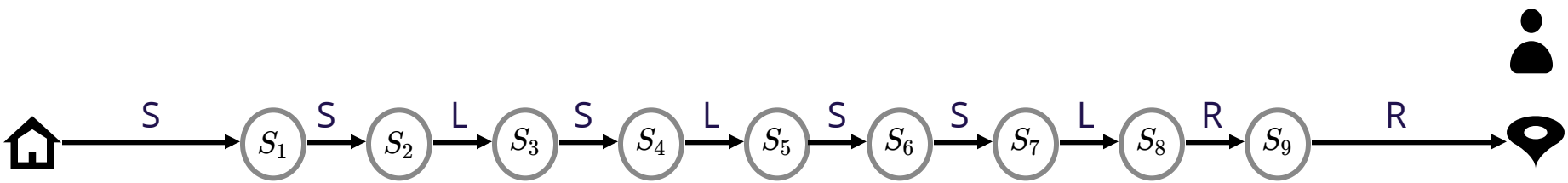


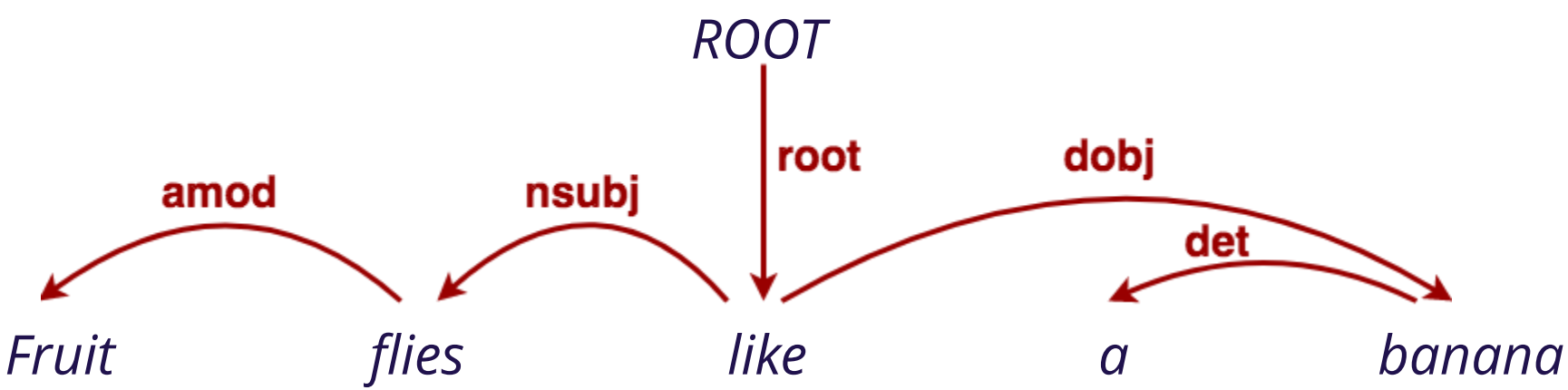


Optional

Transitions, States

$$S : \text{sh}, L : \text{la}, R : \text{ra}$$

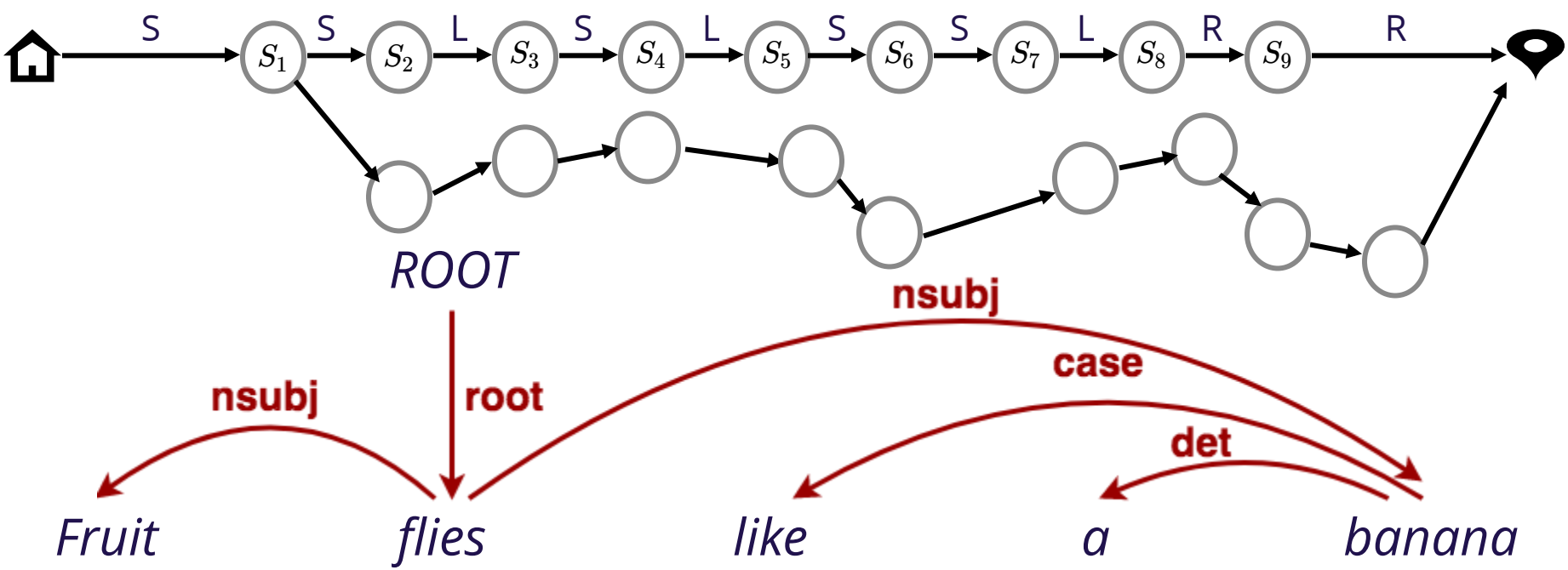




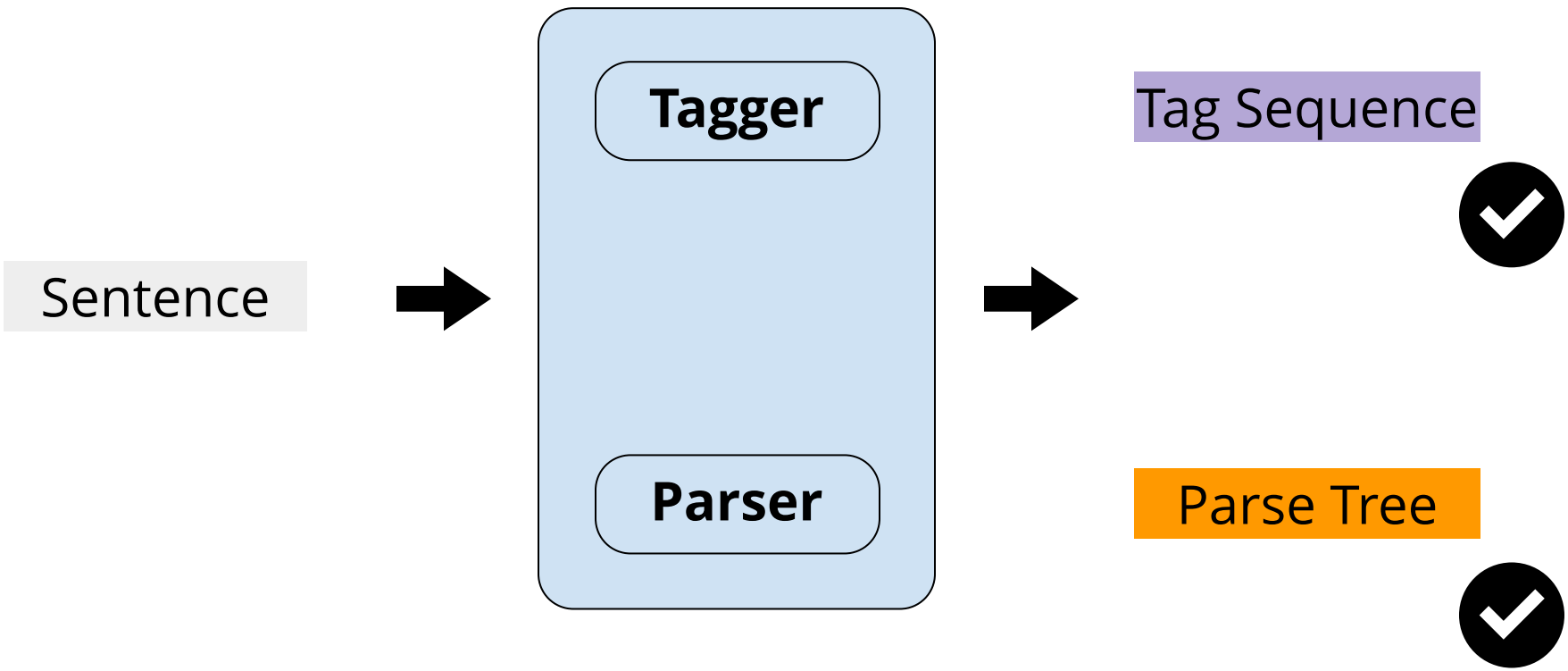
Optional

Transitions, States

The transition-based approach has an underlying connection with what we discussed in previous lectures which are graph-based approaches.



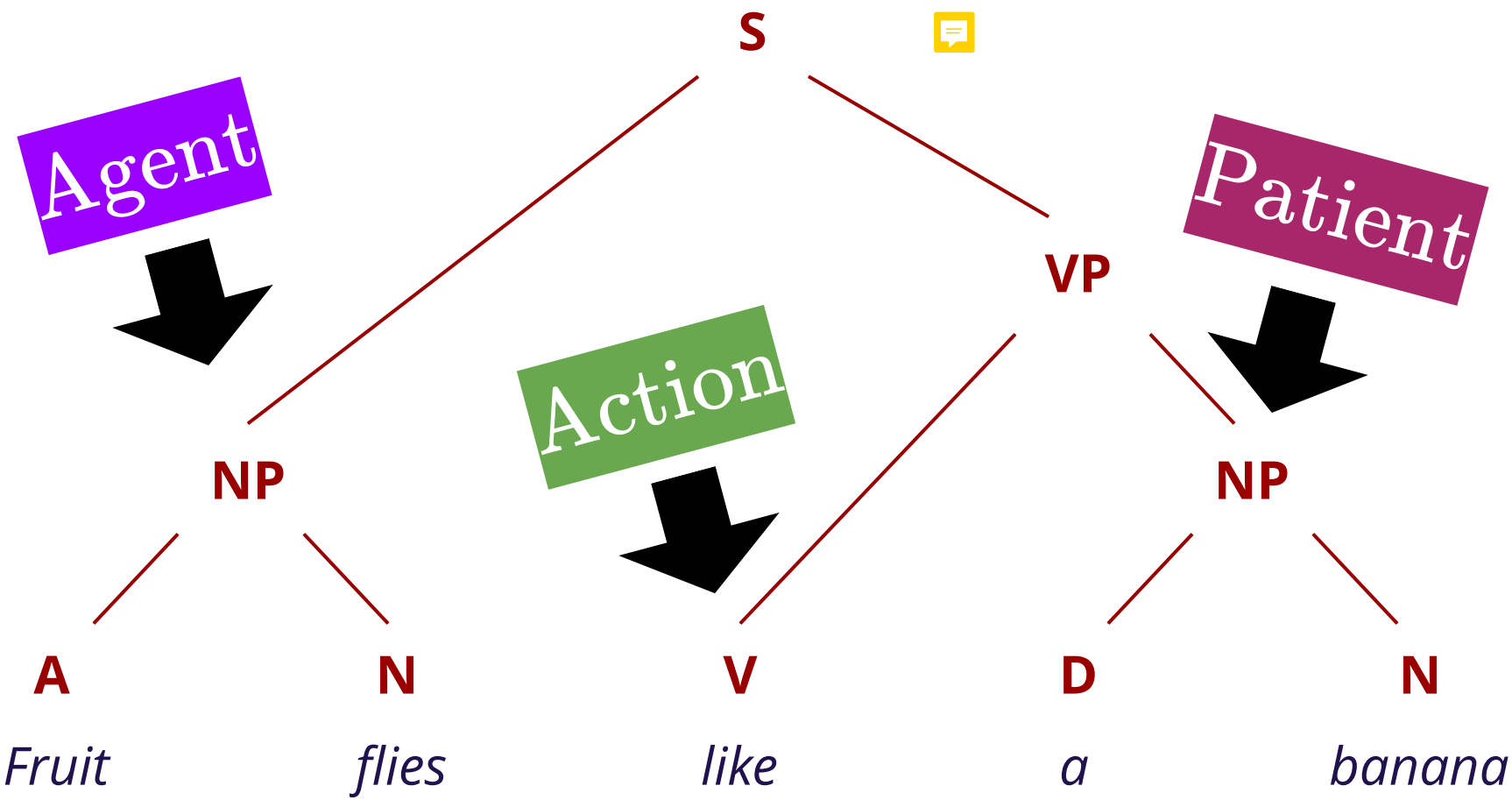
Structured Prediction



Structured Space $\mathcal{S}_i$ | Structured Prediction Model M | Structured Space $\mathcal{S}_o$

$\mathsf{M} : \mathcal{S}_i \rightarrow \mathcal{S}_o$

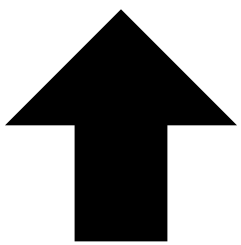
Semantic Roles

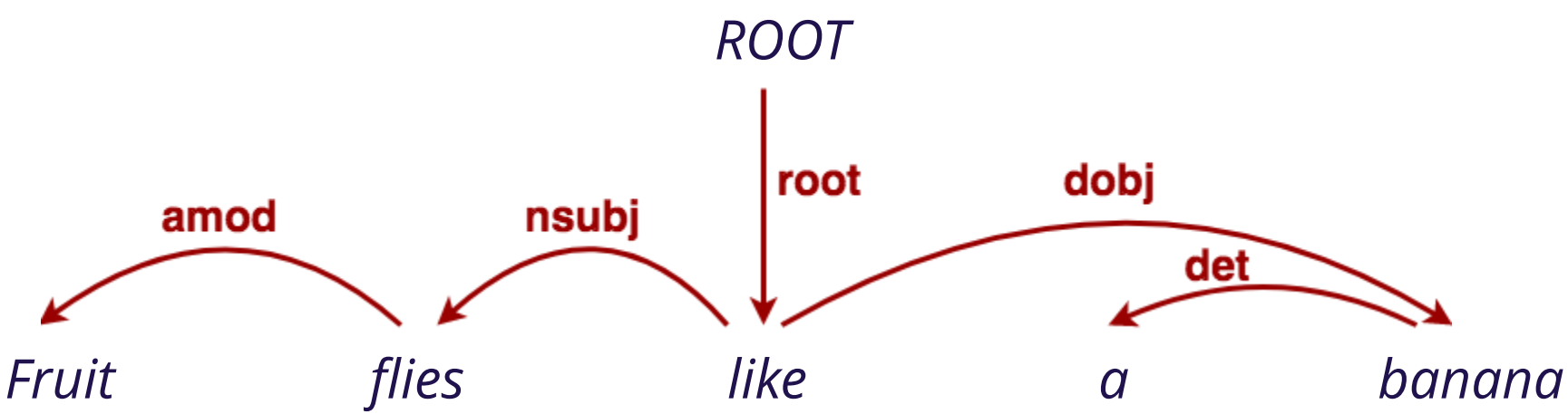


Semantic Forms

$$\text{like}(\text{“}Fruit\ Flies\text{”}, \text{“}a\ banana\text{”})$$

Logical form!





Tasks in NLP

We will move to semantics next!