Notes on Sequence Parsing

G.A. Jarrad

July 1, 2015

1 Introduction

The purpose of *sequence parsing* is to provide a hierarchically structured interpretation of a sequence of tokens, $\vec{\tau} = (\tau_1, \tau_2, ..., \tau_n)$. Our primary example is the English sentence $\vec{\tau} = (\texttt{The}, \texttt{cat}, \texttt{sat}, \texttt{on}, \texttt{the}, \texttt{mat})$, the parse of which has a variety of representations, as shown by (but not restricted to) Figures 1.1–1.3. It is important to note, however, that some natural languages are not so strictly ordered, and so a general sequence parse need not necessarily follow the same ordering as the token sequence.

```
{ {The cat} {sat {on {the mat} } } }
```

Figure 1.1: The parse represented as a hierarchical partitioning of the tokens.

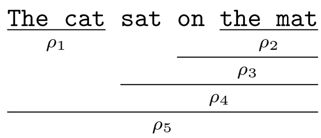

Figure 1.2: The parse represented as an ordered set of combination rules.

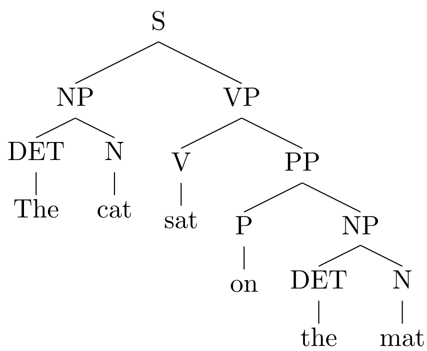

Figure 1.3: The parse represented as a tree of nodes with part-of-speech categories.

Sequence parsing typically comprises two distinct stages: (i) *token analysis*, discussed more fully in Section **??**; and (ii) *structure analysis*, discussed in

Section **??**. Briefly, the purpose of token analysis is to deduce, for each token τ_i, the set $\Lambda(\tau_i)$ of *leaf nodes* that represent plausible interpretations of the token. For example, in natural language understanding the leaf nodes might include categories such as part-of-speech, as shown in Figure 1.3. The purpose of structure analysis is then to deduce a set $\Pi(\vec{\tau})$ of rules that recursively combine sequences of nodes into higher-order *derived nodes*, until a single derived node spans the entire token sequence $\vec{\tau}$, again as shown in Figure 1.3.

In summary, $\Pi(\vec{\tau})$ may be characterised as a parse tree graph with a set $\mathcal{V}$ of nodes (both leaf and derived), and a set $\mathcal{R}$ of rules linking these nodes. Each leaf node represents known information about the corresponding token, including the position of that token in the token sequence. Likewise, each derived node represents a sequence of leaf and/or derived nodes, which includes knowledge of the positions of all corresponding tokens. In general, therefore, every node $\nu \in \mathcal{V}$ *spans* a set of tokens. Specifically, we define the span $\sigma(\nu)$ of node ν to be the ordered set of indices of the underlying tokens. Consequently, each rule $\rho \in \mathcal{R}$ then takes the form

$$\nu_{i_1}\ \nu_{i_2}\ \cdots\ \nu_{i_m} \quad \stackrel{\rho}{\to} \quad \nu_*\,, \tag{1.1}$$

where the derived node $\nu_* = \delta(\rho)$ combines the *predecessor* nodes $\vec{\pi}(\rho) = (\nu_{i_1}, \ldots, \nu_{i_m})$ with a resulting ordered span of

$$\sigma(\delta(\rho)) = \bigcup_{\nu \in \vec{\pi}(\rho)} \sigma(\nu)\,. \tag{1.2}$$

For example, if $\sigma(\nu_1) = \{2, 1\}$ and $\sigma(\nu_2) = \{3\}$, then the rule $\nu_1\ \nu_2 \to \nu_3$ implies that $\sigma(\nu_3) = \{2, 1, 3\}$.