Formal Models of Language

Paula Buttery

Dept of Computer Science & Technology, University of Cambridge

Shift-reduce parsers are useful for **deterministic** languages

LR(k) Shift-reduce parsers are most useful for recognising the strings of **deterministic** languages (languages where no string has more than one analysis) which have been described by an unambiguous grammar.

Quick reminder:

- The parsing algorithm has two actions: SHIFT and REDUCE
- Initially the input string is held in the buffer and the stack is empty.
- Symbols are **shifted** from the buffer to the stack
- When the top items of the stack match the RHS of a rule in the grammar then they are **reduced**, that is, they are replaced with the LHS of that rule.
- k refers to the look-ahead.

Reminder: shift-reduce parsing using a **deterministic** CFG

Shift-reduce parse for the string *abcd* generated using $G_{cfg} = (\Sigma, \mathcal{N}, s, \mathcal{P})$:

$$\begin{array}{lcl} \Sigma & = & \{a, b, c\} \\ \mathcal{N} & = & \{S, A, B, C, D\} \\ s & = & \mathsf{S} \\ \mathcal{P} & = & \{S \rightarrow A\ B, \\ & & A \rightarrow a, \\ & & B \rightarrow b\ C, \\ & & C \rightarrow c\ D, \\ & & D \rightarrow d\} \end{array}$$

STACK	BUFFER	ACTION
	abcd	SHIFT
a	bcd	REDUCE
A	bcd	SHIFT
Ab	cd	SHIFT
Abc	d	SHIFT
Abcd		REDUCE
AbcD		REDUCE
AbC		REDUCE
AB		REDUCE
S		

Reminder: properties of **Deterministic** CFLs

Deterministic context-free languages:

- are a proper subset of the context-free languages
- are accepted by deterministic push-down automata
- can be modelled by an unambiguous grammar
- can be parsed in linear time
- parser can be automatically generated from the grammar

CFGs used to model natural language are **not** deterministic

- Natural languages (with all their inherent ambiguity) are not well suited to shift-reduce parsers which operate deterministically recognising a single derivation without backtracking
- However, natural language parsing can be achieved deterministically by selecting parsing actions using a machine learning classifier (more on this next time).
- All CFGs (including those exhibiting ambiguity) can be recognised in polynomial time using chart parsing algorithms.

Ambiguous grammars derive a **parse forest**

Number of binary trees is proportional to the Catalan number

$$\texttt{Num of trees for sentence length n} = \prod_{k=2}^{n-1} \frac{(n-1)+k}{k}$$

sentence length	number of trees	sentence length	number of trees
3	2	8	429
4	5	9	1430
5	14	10	4862
6	42	11	16796
7	132	12	58786

We need parsing algorithms that can efficiently store the parse forest and not derive shared parts of tree more than once—chart parsers

The **Earley parser** is a **chart parsing** algorithm

The **Earley parser** is a dynamic programming algorithm that records partial derivations in a CHART (a table).

- Uses a top-down approach to explore the whole search space, recovering multiple derivations where they exist.
- The progress of the algorithm is encoded in something called a **dotted rule** or **progress rule**:

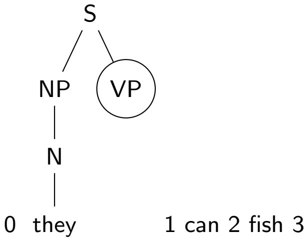
  $A \rightarrow {}_{\bullet}\alpha\beta \mid \alpha_{\bullet}\beta \mid \alpha\beta_{\bullet}$ where $A \rightarrow \alpha\beta \in \mathcal{P}$.
- Rules of the form $A \rightarrow {}_{\bullet}\alpha\beta$ have all symbols still to be explored;
- Rules of the form $A \rightarrow \alpha\beta_{\bullet}$ have been completely *used up* deriving a portion of the string.

Partial derivations are recorded in a **chart**

- By convention, each row of the chart is referred to as an **edge**.
- An edge in the chart records a dotted rule, and its **span**.
- The span refers to the portion of the input string which is consistent with the partial tree.
- If we wish to discover the structure of a parse, an edge must also record the derivation **history** of the immediately previous partial tree(s) that made the current partial tree possible.

Partial derivations are recorded in a **chart**

For an illustration, consider the partial tree below which has been derived when attempting to parse the sentence *they can fish*:



ID	RULE	[start, end]	HIST
⋮			
e_i	$S \rightarrow NP_{\bullet}VP$	$[0, 1]$	h_k

Partial derivations are recorded in a **chart**

For input string $u = a_1 ... a_n$ and grammar $G_{cfg} = (\mathcal{N}, \Sigma, S, \mathcal{P})$:

An edge $A \rightarrow \alpha_\bullet\beta\ [i,j]$ is added if

$S \underset{G^*}{\Longrightarrow} a_1 ... a_i A\gamma$ where γ are symbols in u yet to be parsed

and $\alpha \underset{G^*}{\Longrightarrow} a_{i+1} ... a_j$

- The chart is **initialised** with the edge $S \rightarrow {}_\bullet\alpha\beta\ [0,0]$;
- The input string $u = a_1 ... a_n$ is **recognised** when we add the edge $S \rightarrow \alpha\beta_\bullet\ [0,n]$.

Today's toy grammar

We will parse the sentence *they can fish* using $G_{cfg} = (\mathcal{N}, \Sigma, S, \mathcal{P})$ where:

$$
\begin{array}{lcl}
\mathcal{N} & = & \{S, NP, VP, PP, N, V, P\} \\
\Sigma & = & \{can, fish, in, rivers, they \ldots\} \\
S & = & S \\
\mathcal{P} & = & \{S \rightarrow NP\ VP \\
 & & NP \rightarrow N\ PP \mid N \\
 & & PP \rightarrow P\ NP \\
 & & VP \rightarrow VP\ PP \mid V\ VP \mid V\ NP \mid V \\
 & & N \rightarrow can \mid fish \mid rivers \mid \ldots \\
 & & P \rightarrow in \mid \ldots \\
 & & V \rightarrow can \mid fish \mid \ldots\ \}
\end{array}
$$

0 they 1 can 2 fish 3

Initialise the chart

The chart is initialised with $S \rightarrow {\bullet}\alpha\beta\ [0,0]$.

ID	RULE	[start, end]	HIST
e_0	$S \rightarrow \bullet\ NP\ VP$	$[0,0]$	

In rule induction notation:

$$\frac{}{S \rightarrow {\bullet}\alpha\beta\ [0,0]}\ \text{(induction step)}$$

Three steps of the Earley parser: **predict** step

This step adds new edges to the chart and can be thought of as expanding tree nodes in the top-down derivation.

ID	RULE	[start, end]	HIST
e_0	$S \rightarrow \bullet\ NP\ VP$	$[0, 0]$	
e_1	$NP \rightarrow \bullet\ N$	$[0, 0]$	
e_2	$NP \rightarrow \bullet\ N\ PP$	$[0, 0]$	

In rule induction notation:

$$\frac{A \rightarrow \alpha_\bullet B\beta\ [i, j]}{B \rightarrow {}_\bullet\gamma\ [j, j]}\ \text{(predict step) where } B \rightarrow \gamma \in \mathcal{P}$$

Three steps of the Earley parser: **scan** step

This step allows us to check if we have a node that is consistent with the input sentence. If the input sentence is $u = a_1 ... a_n$ we can add a new edge if $A \rightarrow \bullet a \ [i, j-1]$ and $a = aj$.

ID	RULE	[start, end]	HIST
e_0	$S \rightarrow \bullet \ NP \ VP$	$[0, 0]$	
e_1	$NP \rightarrow \bullet \ N$	$[0, 0]$	
e_2	$NP \rightarrow \bullet \ N \ PP$	$[0, 0]$	
e_3	$N \rightarrow they \bullet$	$[0, 1]$	

In rule induction notation:

$$\frac{A \rightarrow \bullet a \ [i, j-1]}{A \rightarrow a \bullet \ [i, j]} \text{ (scan step) when } a = a_j$$

Three steps of the Earley parser: **scan** step

- For natural language sentence parsing tasks, Σ can be the finite set of words in the language (a very large set).
- when carrying out the predict step from a rule like $NP \rightarrow \bullet N$ we would end up adding a new edge for every *noun* in the language.
- To save us from creating all these edges we can privilege a set of the non-terminals and perform a forward look-up of the next a_j to see whether it will be consistent.
- In our example this set would be $\mathcal{N}_{PofS} = \{N, V, P\}$, that is, all the non-terminal symbols that represent the **parts-of-speech** of the language (such as *nouns*, *verbs*, *adjectives*...).
- During the scanning step, we find edges containing non-terminals in $\mathcal{N}_{PofS}$ with a dot on their LHS and check if the upcoming word is consistent with the part-of-speech. Iff it is consistent then we add an edge to the chart.

Three steps of the Earley parser: **complete** step

This step propagates fully explored tree nodes in the chart.

ID	RULE	[start, end]	HIST
e_0	$S \rightarrow \bullet\ NP\ VP$	$[0,0]$	
e_1	$NP \rightarrow \bullet\ N$	$[0,0]$	
e_2	$NP \rightarrow \bullet\ N\ PP$	$[0,0]$	
e_3	$N \rightarrow \mathit{they}\ \bullet$	$[0,1]$	
e_4	$NP \rightarrow N\ \bullet$	$[0,1]$	e_3
e_5	$NP \rightarrow N\ \bullet\ PP$	$[0,1]$	e_3
e_6	$S \rightarrow NP\ \bullet\ VP$	$[0,1]$	e_4

In rule induction notation:

$$\frac{A \rightarrow \alpha_\bullet B\beta\ [i,k] \qquad B \rightarrow \gamma_\bullet\ [k,j]}{A \rightarrow \alpha B_\bullet \beta\ [i,j]} \text{ (complete step)}$$

ID	RULE	[start, end]	HIST	word n
e_0	$S \rightarrow \bullet\ NP\ VP$	$[0,0]$		**word 0**
e_1	$NP \rightarrow \bullet\ N$	$[0,0]$		**word 1**
e_2	$NP \rightarrow \bullet\ N\ PP$	$[0,0]$		
e_3	$N \rightarrow they\ \bullet$	$[0,1]$		
e_4	$NP \rightarrow N\ \bullet$	$[0,1]$	(e_3)	
e_5	$NP \rightarrow N\ \bullet\ PP$	$[0,1]$	(e_3)	
e_6	$S \rightarrow NP\ \bullet\ VP$	$[0,1]$	(e_4)	
e_7	$PP \rightarrow \bullet\ P\ NP$	$[1,1]$		**word 2**
e_8	$VP \rightarrow \bullet\ V$	$[1,1]$		
e_9	$VP \rightarrow \bullet\ V\ NP$	$[1,1]$		
e_{10}	$VP \rightarrow \bullet\ V\ VP$	$[1,1]$		
e_{11}	$VP \rightarrow \bullet\ VP\ PP$	$[1,1]$		
e_{12}	$V \rightarrow can\ \bullet$	$[1,2]$		
e_{13}	$VP \rightarrow V\ \bullet$	$[1,2]$	(e_{12})	
e_{14}	$VP \rightarrow V\ \bullet\ NP$	$[1,2]$	(e_{12})	
e_{15}	$VP \rightarrow V\ \bullet\ VP$	$[1,2]$	(e_{12})	
e_{16}	$S \rightarrow NP\ VP\ \bullet$	$[0,2]$	(e_4, e_{13})	
e_{17}	$VP \rightarrow VP\ \bullet\ PP$	$[1,2]$	(e_{13})	
e_{18}	$NP \rightarrow \bullet\ N$	$[2,2]$		**word 3**
e_{19}	$NP \rightarrow \bullet\ N\ PP$	$[2,2]$		
e_{20}	$VP \rightarrow \bullet\ V$	$[2,2]$		
e_{21}	$VP \rightarrow \bullet\ V\ NP$	$[2,2]$		

The run time of the Earley parser is **polynominal**

- The **complete** step dominates run time $O(n^2)$
- Running time of the Earley parser is $O(n^3)$
- Run time is reduced in various scenarios, e.g. when the grammar is unambiguous or left-recursive .[1]

So what makes a sentence **complex** for a human to process?

[1] See `https://homepages.cwi.nl/~jve/lm2005/earley.pdf` for a full discussion

The term **complexity** can be used to describe human processing difficulty

The term **complexity** is also used to describe the perceived human processing difficulty of a sentence: work in this area is generally referred to as **computational psycholinguistics**.

Complexity within this domain can refer to:

- the **time and space requirements** of the algorithm that your brain is posited to require while processing a sentence.
- the **information theoretic content** of the sentence itself in isolation from the human processor (more in later lectures on this)

The term **complexity** can be used to describe human processing difficulty

Traditional work in this area has looked mainly at parsing algorithms to discover whether they exhibit properties that correlate with measurable predictors of complexity in human linguistic behaviour.

Two general assumptions are made in this work:

1) Sentences will take **longer to process** if they are more complicated for the human parser.
    - Processing time is usually measured as the time it takes to read a sentence.
    - This can be done with eye-tracking machines which also identify whether the subject reread any parts of a sentence.
    - Researchers also use neuro-imaging techniques (MEG, fMRI)

The term **complexity** can be used to describe human processing difficulty

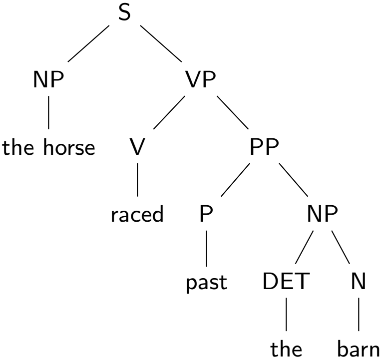
2) Sentences **will not occur frequently in the spoken language** if they are complicated to produce or comprehend.
   - Frequencies are calculated by counting constructions of interest in spoken language corpora.

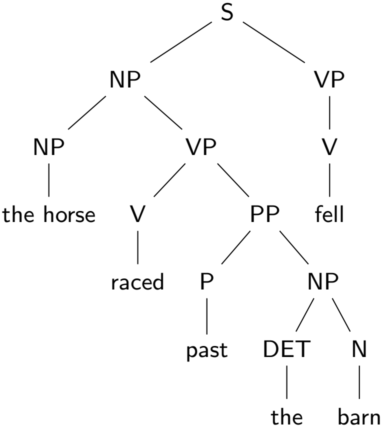
The assumption then is that one (or both) of the two measurements of perceived complexity above will correlate with time and space requirements of the parsing algorithm.

What makes a sentence expensive to process?

Example: long distance syntactic dependencies (e.g. garden-paths)

- The horse raced past the barn
- The horse raced past the barn fell—comparatively slow reading time





Hale — Earley parser as a model of sentence processing

Using predictability as a measure of difficulty

- The cognitive effort associated with a word in a sentence can be measured by the word's **surprisal** (negative log conditional probability): $\log \frac{1}{P(w_i|w_{1\cdots i-1})}$ (more on this in later lectures)
- The suggestion is that probabilistic context-free grammars (PCFGs) can be used to model human language processing.

  $G_{pcfg} = (\Sigma, \mathcal{N}, S, \mathcal{P}, q)$ where q is a mapping from rules in $\mathcal{P}$ to a probability and $\sum_{A \to \alpha \in \mathcal{P}} q(A \to \alpha) = 1$
- A probabilistic Earley parser is used as a model of online eager sentence processing.

Hale — Earley parser as a model of sentence processing

- The probabilistic Earley parser computes all parses of its input.
- As a psycholinguistic theory it is one of **total parallelism** (as opposed to a reanalysis theory)
- Calculate **prefix probabilities** i.e. probabilities of partially derived trees.
- Hypothesis is that the cognitive effort expended to parse a given prefix is **proportional to the total probability** of all the structural analyses which are not compatible with the prefix.
- Generates predictions about word-by-word reading times by comparing the total effort expended before some word to the total effort after.
- The explanation for garden-pathing is then **the reduction in the probability** of the new tree set compared with the previous tree set.
- The model accounts successfully for reading times.

Hale — Earley parser as a model of sentence processing

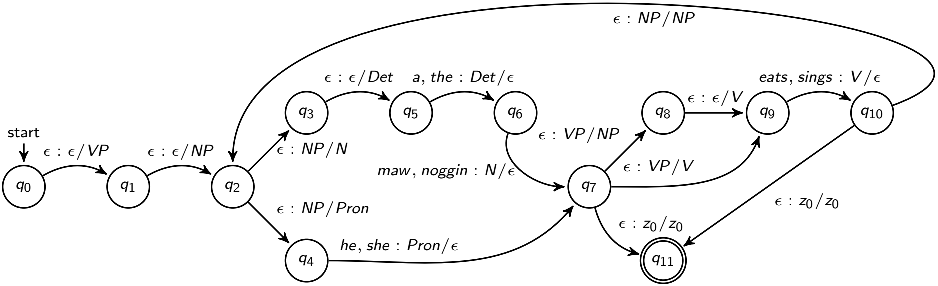
Toy grammar with probabilities

S	$\rightarrow$	NP VP	1
NP	$\rightarrow$	N PP	0.2
NP	$\rightarrow$	N	0.8
PP	$\rightarrow$	P NP	1
VP	$\rightarrow$	VP PP	0.1
VP	$\rightarrow$	V VP	0.2
VP	$\rightarrow$	V NP	0.4
VP	$\rightarrow$	V	0.3
N	$\rightarrow$	{it, fish, rivers, December, they}	0.2
P	$\rightarrow$	{in}	1
V	$\rightarrow$	{can, fish}	0.5

Hale — Earley parser as a model of sentence processing

$edge_n$	DOTTED RULE	[S, W]	HIST	**Prob**	**MaxProb**
e_0	S → ● NP VP	[0,0]			P(S → NP VP)=1
e_1	NP → ● N	[0,0]			P(e_0)P(NP → N)=1*0.8=0.8
e_2	NP → ● N PP	[0,0]			P(e_0)P(NP → N PP)=1*0.2=0.2
e_3	N → they ●	[0,1]		P(N → they)=0.2	
e_4	NP → N ●	[0,1]	(e_3)	P(e_3)P(NP → N) =0.2*0.8 =0.16	
e_5	NP → N ● PP	[0,1]	(e_3)		
e_6	S → NP ● VP	[0,1]	(e_4)		
e_7	PP → ● P NP	[1,1]			P(N → they)P(e_2)P(PP → P NP) =0.2*1*0.2*1=0.04
e_8	VP → ● V	[1,1]			P(N → they)P(e_1)P(VP → V) =0.2*1*0.8*0.3=0.048
e_9	VP → ● V NP	[1,1]			P(N → they)P(e_1)P(VP → V NP) =0.2*1*0.8*0.4=0.064
e_{10}	VP → ● V VP	[1,1]			P(N → they)P(e_1)P(VP → V VP) =0.2*1*0.8*0.2=0.032
e_{11}	VP → ● VP PP	[1,1]			P(N → they)P(e_1)P(VP → VP PP) =0.2*1*0.8*0.1=0.0016
e_{12}	V → can ●	[1,2]		P(V → can)=0.5	
e_{13}	VP → V ●	[1,2]	(e_{12})	P(e_{12})P(VP → V) =0.5*0.3 =0.15	
e_{14}	VP → V ● NP	[1,2]	(e_{12})		
e_{15}	VP → V ● VP	[1,2]	(e_{12})		
e_{16}	S → NP VP ●	[0,2]	(e_4,e_{13})	P(e_4)P(e_{13})P(S → NP VP) =0.2*0.8*0.5*0.3*1 =0.024	
e_{17}	VP → VP ● PP	[1,2]	(e_{13})		

Yngve—PDA as a model of sentence processing



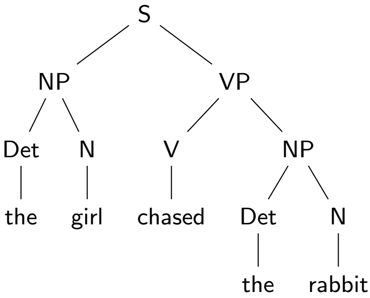
- **Hypothesis**: the size of the stack correlates with working memory load.
- **Prediction**: sentences which require many items to be placed on the stack will be difficult to process and also less frequent in the language.
- **Prediction**: when multiple parses are possible we should prefer the one with the minimised stack.

Yngve—PDA as a model of sentence processing

- Yngve formulated the problem as interaction between:
  - a **register** (which holds the current node) and
  - the **stack** (which contains all the nodes left to explore)
- Sentences are constructed top-down and left-to-right.
- Under these circumstances the size of the stack is hypothesised to correlate with working memory load.

Hypothesis: stack correlates with **working memory load**

S→NP VP
NP→Det N
VP→V NP
Det→the
N→girl
N→rabbit
V→chased



Register	Stack
S	
NP	VP
Det	N VP
the	N VP
N	VP
girl	VP
VP	
V	NP
chased	NP
NP	
Det	N
the	N
N	
rabbit	

Hypothesis: stack correlates with **working memory load**

Yngve's model makes **predictions** about centre embedding:

- Consider:

  This is the malt that the rat that the cat that the dog worried killed ate.

  STACK: N VP VP VP

- as opposed to:

  This is the malt that was eaten by the rat that was killed by the cat that was worried by the dog.

- Yngve evaluated his predictions by looking at frequencies of constructions in corpus data.