School of Computing and Information Systems
The University of Melbourne
COMP90042
WEB SEARCH AND TEXT ANALYSIS (Semester 1, 2017)

Sample Earley solution: Week 4

Discussion

2. Consider the following simple **context–free grammar**:

```
S -> NP VP
VP -> V NP | V NP PP
PP -> P NP
V -> "saw" | "walked"
NP -> "John" | "Bob" | Det N | Det N PP
Det -> "a" | "an" | "the" | "my"
N -> "man" | "cat" | "telescope" | "park"
P -> "on" | "by" | "with"
```

   (a) What changes need to be made to the grammar to make it suitable for **CYK parsing**?

   (b) Using the ~~CYK strategy and the above grammar in CNF~~ Earley strategy, parse the following sentences:

      (ii) "an park by Bob walked an park with Bob"

      - This is the most interesting example; we will work through it in detail.
      - For Earley, the chart is much simpler[1] than CYK: there is a chart position for each token in the sentence, plus one for the start of the sentence. Each chart position contains a list of rules (edges), where each rule (edge) has the following structure:

        `A->B·C [m]`

        Where `A` is a non-terminal, `B` and `C` are zero or more terminals or non-terminals[2], and `m` is the chart index where the edge was created[3].
      - We initialise chart position 0 (for the start of the sentence) with the following dummy rule[4]:

        γ`->·S [0]`

        This rule (edge) indicates that we are going to start at the start of the sentence (position 0) and look for an `S` — an entire sentence.
      - Typically[5], we mark up each chart position with the possible parts–of–speech (POS) for each token in the sentence that the chart position corresponds to, with the following edge:

[1] In fact, with a little bookkeeping, the chart can be represented by a one-dimensional array.

[2] It is typically more efficient to transform the grammar so that the right-hand side of each rule consists of only non-terminals, or a single terminal. This is a simpler process than the conversion into Chomsky Normal Form.

[3] Sometimes this is written as `[m,n]`, where `n` is the (redundant) chart index of where the edge is currently stored.

[4] Sometimes, this step is skipped, and we go straight to expanding the start symbol, in particular, when there is only a single start symbol.

[5] This step can be skipped, at the cost of generating many trivially false hypotheses; for example, there is no need to posit that any of the (perhaps tens of thousands of) nouns could occur at a given chart position — we can observe immediately which lexical item is present.

`A->x· [p-1]`

where `x` is terminal corresponding to the p^{th} token in the sentence, and `A` is a **pre-terminal** corresponding to the part–of–speech[6] of `x`.

- For the given sentence, the chart is consequently initialised as follows:

0	1	2	3
`γ->·S [0]`	`Det->"an"· [0]`	`N->"park"· [1]`	`P->"by"· [2]`
4	5	6	7
`NP->"Bob"· [3]`	`V->"walked"· [4]`	`Det->"an"· [5]`	`N->"park"· [6]`
8	9		
`P->"with"· [7]`	`NP->"Bob"· [8]`		

- We will fill the chart from position 0 (the "left" side of the chart) to position 9 (the "right"), in a top-down manner. To do this, we will process each edge, one–by–one, and then add more edges to the chart as required, according to either the Predictor or Completer[7]:
  - **Predictor**: If we are processing an edge whose dot precedes a non-terminal, we add edges to this chart position according to the rules in the grammar with that non-terminal on the left-hand side (LHS), which have one or more non-terminals on the right-hand side[8] (RHS).
  - **Completer**: If we are processing an edge where nothing follows the dot, we have completed the non-terminal on the LHS of this rule. We then find the edge(s), located at the chart position indicated in brackets, which have the dot directly preceding that non-terminal, and move the dot along. (It sounds confusing, but the example below will help.)
- In this case, we begin at chart position 0, which currently only contains the dummy rule γ->· S [0]; the dot precedes a non-terminal (`S`), so we Predict according to the rule(s) in our grammar with `S` on the LHS. Here, there is only one such rule: `S->NP VP`, so we add this rule (as an edge) to our chart position 0 as follows:

  `S->·NP VP [0]`

  Whenever we Predict edges, the dot is placed at the beginning of the RHS (non-terminals), and the index in brackets is whichever chart position we are currently working on.
- We have now processed the edge `γ->·S [0]`; we then proceed to the next edge in this chart position, namely, the edge we just added. Here, the dot proceeds `NP`, so we Predict according to the rule(s) in our grammar with `NP` on the LHS: this time, there are two:

  `NP->·Det N [0]`
  `NP->·Det N PP [0]`

[6]However, sometimes grammars are simplified to abstract away from some of the sentence structure; as you can see here, *Bob* is identified as an `NP`, which is really a simplification of the fact that a proper noun (`NNP`) can form an `NP` according to the rule `NP->NNP`.

[7]Assuming that we have transformed the grammar as described in footnote 2, and didn't skip initialising the entire chart with the (dotted) tokens from the sentence, as described in footnote 5, the dot can never precede a terminal, so we don't have to worry about the **Scanner** — in effect we have already applied all of the Scan operations when we initialised the chart.

[8]If we haven't initialised the table with our Scanner edges, or we haven't transformed the grammar according to footnote 2, then we also need to add edges for rules with terminals on the right-hand side.

- We then process these two edges (with the Predictor), but in both cases, the `Det` rules only have terminals on the RHS. Now we have run out of edges in this chart position, so we will go on to the chart position 1 with the following chart:

0	1	2	3

`γ->·S [0]`
`S->·NP VP [0]`
`NP->·Det N [0]`
`NP->·Det N PP [0]`
	`Det->"an"· [0]`	`N->"park"· [1]`	`P->"by"· [2]`

4	**5**	**6**	**7**
`NP->"Bob"· [3]`	`V->"walked"· [4]`	`Det->"an"· [5]`	`N->"park"· [6]`
8	**9**		
`P->"with"· [7]`	`NP->"Bob"· [8]`		

- Chart position 1 initially contains the edge `Det->"an"·[0]`: we apply our Completer as follows:
  - The index is [0], so we will look in chart position 0
  - The non-terminal on the LHS is `Det`, so we will look for edges with the dot directly preceding `Det`
  - There are two such edges: `NP->·Det N [0]` and `NP->·Det N PP [0]`
- For each rule that we find, we move the dot so that it follows the `Det`, and then add this edge to the current chart position:
  `NP->Det·N [0]`
  `NP->Det·N PP [0]`
- We then process these two rules: the dot precedes `N` (in both cases), but the rules where `N` is on the LHS only have terminals on the RHS. So, we're finished with this chart position:

0	1	2	3

`γ->·S [0]`
`S->·NP VP [0]`
`NP->·Det N [0]`
`NP->·Det N PP [0]`
	`Det->"an"· [0]`
	`NP->Det·N [0]`
	`NP->Det·N PP [0]`
		`N->"park"· [1]`	`P->"by"· [2]`

4	**5**	**6**	**7**
`NP->"Bob"· [3]`	`V->"walked"· [4]`	`Det->"an"· [5]`	`N->"park"· [6]`
8	**9**		
`P->"with"· [7]`	`NP->"Bob"· [8]`		

- Chart position 2 initially contains the edge `N->"park"·[1]`: we apply our Completer as follows:
  - The index is [1], so we will look in chart position 1
  - The non-terminal on the LHS is `N`, so we will look for edges with the dot directly preceding `N`
  - There are two such edges: `NP->Det·N [0]` and `NP->Det·N PP [0]`
- We move the dot along, and then add the following two edges to this chart position: `NP->Det N· [0]` and `NP->Det N·PP [0]`
- For the first of those edges, the dot is at the end, so we apply our Completer:

- The index is [0], so we will look in chart position 0
  - The non-terminal on the LHS is `NP`, so we will look for edges with the dot directly preceding `NP`
  - There is one such edge: `S->·NP VP [0]`
- We now add the edge `S->NP·VP [0]` to this position.
- Next, we have the edge `NP->Det N·PP [0]` — the dot precedes `PP`, so we Predict according to the grammar's only rule with `PP` on the LHS to add the edge `PP->·P NP [2]`
- Now, we process the edge `S->NP·VP [0]` to Predict `VP->·V NP [2]` and `VP->·V NP PP [2]`
- The remaining edges don't allow us to Predict anything (`P` and `V` are only on the LHS of rules with terminals on the RHS). So, we are done with this chart position:

0	1	2	3

`γ->·S [0]`
`S->·NP VP [0]`
`NP->·Det N [0]`
`NP->·Det N PP [0]`
	`Det->"an"· [0]`
	`NP->Det·N [0]`
	`NP->Det·N PP [0]`
		`N->"park"· [1]`
		`NP->Det N· [0]`
		`NP->Det N·PP [0]`
		`S->NP·VP [0]`
		`PP->·P NP [2]`
		`VP->·V NP [2]`
		`VP->·V NP PP [2]`
			`P->"by"· [2]`

4	**5**	**6**	**7**
`NP->"Bob"· [3]`	`V->"walked"· [4]`	`Det->"an"· [5]`	`N->"park"· [6]`
8	**9**		
`P->"with"· [7]`	`NP->"Bob"· [8]`		

- Before we move on, let's look at what the chart is telling us: we believe that we might have just completed an `NP` — in which case, we've completed the first part of our `S`, and now we're looking for a `VP` which must begin with a `V`; or, we haven't finished our `NP` yet, because it might also have a `PP` — in which case, we're looking for a `P` next.
- Chart position 3 initially contains the edge `P->"by"·[2]:` we apply our Completer as follows:
  - The index is [2], so we will look in chart position 2
  - The non-terminal on the LHS is `P`, so we will look for edges with the dot directly preceding `P`
  - There is one such edge: `PP->·P NP [2]`
- Before we move on, you might like to observe that the edges from chart position 2 that were expecting a `V` will never generate any more edges later in the chart; this procedure (using the tokens that we've actually seen to reject hypothetical analyses) helps to keep the chart from exploding in size.
- After moving the dot after the `P` in the edge above, we Predict the following two rules: `NP->·Det N [3]` and `NP->·Det N PP [3]`
- The current chart follows overleaf.
- Chart position 4 initially contains the edge `NP->"Bob"·[3]:`

0	1	2	3
γ->·S [0] S->·NP VP [0] NP->·Det N [0] NP->·Det N PP [0]	Det->"an"· [0] NP->Det·N [0] NP->Det·N PP [0]	N->"park"· [1] NP->Det N· [0] NP->Det N·PP [0] S->NP·VP [0] PP->·P NP [2] VP->·V NP [2] VP->·V NP PP [2]	P->"by"· [2] PP->P·NP [2] NP->·Det N [3] NP->·Det N PP [3]
4	5	6	7
NP->"Bob"· [3]	V->"walked"· [4]	Det->"an"· [5]	N->"park"· [6]
8	9		
P->"with"· [7]	NP->"Bob"· [8]		

  - We will look in chart position 3, for edges with the dot directly preceding `NP`
  - There is one such edge: `PP->P·NP [2]`
- Moving the dot after the `NP` allows us to Complete that `PP` edge, which in turn allows us to Complete the edge `NP->Det N PP [0]` at position 2.
- Now we have completed the initial `NP` in our `S` (`S->NP·VP [0]`), so we Predict `VP->·V NP [4]` and `VP->·V NP PP [4]`, to arrive at the following chart:

0	1	2	3
γ->·S [0] S->·NP VP [0] NP->·Det N [0] NP->·Det N PP [0]	Det->"an"· [0] NP->Det·N [0] NP->Det·N PP [0]	N->"park"· [1] NP->Det N· [0] NP->Det N·PP [0] S->NP·VP [0] PP->·P NP [2] VP->·V NP [2] VP->·V NP PP [2]	P->"by"· [2] PP->P·NP [2] NP->·Det N [3] NP->·Det N PP [3]
4	5	6	7
NP->"Bob"· [3] PP->P NP· [2] NP->Det N PP· [0] S->NP·VP [0] VP->·V NP [4] VP->·V NP PP [4]	V->"walked"· [4]	Det->"an"· [5]	N->"park"· [6]
8	9		
P->"with"· [7]	NP->"Bob"· [8]		

- Chart position 5 initially contains the edge `V->"walked"·[4]`:
  - We will look in chart position 4, for edges with the dot directly preceding `V`
  - There are two such edges: `VP->·V NP [4]` and `VP->·V NP PP [4]`
- For the first of these edges, moving the dot along allows us to Predict an `NP`, and add the following edges to the chart: `NP->·Det N [5]` and `NP->·Det N PP [5]`
- For the second of those edge, moving the dot means that we **also** Predict an `NP`, which would mean that we would add exactly the same rules to the chart for this edge. By convention, we only store a single

copy of each edge in the chart[9] — this means that later, when we are recovering the parse(s) from the chart, we will need to check all of the edges in each chart position to determine which one(s) generated each edge in the parse.

- So, the chart now (suitably interpretting the two `NP` edges in position 5) looks like:

0	1	2	3

`γ->·S [0]`
`S->·NP VP [0]`
`NP->·Det N [0]`
`NP->·Det N PP [0]`
	`Det->"an"· [0]`
	`NP->Det·N [0]`
	`NP->Det·N PP [0]`
		`N->"park"· [1]`
		`NP->Det N· [0]`
		`NP->Det N·PP [0]`
		`S->NP·VP [0]`
		`PP->·P NP [2]`
		`VP->·V NP [2]`
		`VP->·V NP PP [2]`
			`P->"by"· [2]`
			`PP->P·NP [2]`
			`NP->·Det N [3]`
			`NP->·Det N PP [3]`

4	**5**	**6**	**7**

`NP->"Bob"· [3]`
`PP->P NP· [2]`
`NP->Det N PP· [0]`
`S->NP·VP [0]`
`VP->·V NP [4]`
`VP->·V NP PP [4]`
	`V->"walked"· [4]`
	`VP->V·NP [4]`
	`VP->V·NP PP [4]`
	`NP->·Det N [5]`
	`NP->·Det N PP [5]`
		`Det->"an"· [5]`	`N->"park"· [6]`

8	**9**		
`P->"with"· [7]`	`NP->"Bob"· [8]`		

- Chart position 6 initially contains the edge `Det->"an"·[5]`:
  - We will look in chart position 5, for edges with the dot directly preceding `Det`
  - There are two such edges: `NP->·Det N [5]` and `NP->·Det N PP [5]` (each has two copies, but we ignore that for now)
- We add the following two edges to the chart: `NP->Det·N [5]` and `NP->Det·N PP [5]`
- Chart position 7 initially contains the edge `N->"park"·[6]`:
  - We will look in chart position 6, for edges with the dot directly preceding `N`
  - There are two such edges: `NP->Det·N [5]` and `NP->Det·N PP [5]`
- For the first of those edges, we Complete the `NP` from chart position 5, which means that we add the following edges: `VP->V NP· [4]` and `VP->V NP·PP [4]`
- For the second of those edges, we Predict the `PP` to add the following edge: `PP->·P NP [7]`
- For the next three edges, the first (`VP->V NP· [4]`) allows us to Complete the `VP` at chart position 4, giving us the edge `S->NP VP· [0]` — however, we don't want to process the dummy γ because we haven't seen all of the sentence input yet. (Effectively, we've observed that the proper prefix "an park by Bob walked an park" is a sentence according to this grammar.)

[9] Alternatively, we could store multiple copies of an edge, with some indication of which edge in the chart caused its creation. This means that the parses for highly ambiguous sentences can be found more efficiently — unfortunately, it is exactly that property that causes the chart to grow very quickly. Generally, the space–for–time tradeoff is seen as undesirable here.

- The second allows us to Predict a `PP` here, which would duplicate the edge `PP->·P NP [7]`.
- Predicting from that edge doesn't give us any new edges (because `P` is on the LHS of rules with only terminals on the RHS). The chart is now:

0	1	2	3

`γ->·S [0]`
`S->·NP VP [0]`
`NP->·Det N [0]`
`NP->·Det N PP [0]`
	`Det->"an"· [0]`
	`NP->Det·N [0]`
	`NP->Det·N PP [0]`
		`N->"park"· [1]`
		`NP->Det N· [0]`
		`NP->Det N·PP [0]`
		`S->NP·VP [0]`
		`PP->·P NP [2]`
		`VP->·V NP [2]`
		`VP->·V NP PP [2]`
			`P->"by"· [2]`
			`PP->P·NP [2]`
			`NP->·Det N [3]`
			`NP->·Det N PP [3]`

4	**5**	**6**	**7**

`NP->"Bob"· [3]`
`PP->P NP· [2]`
`NP->Det N PP· [0]`
`S->NP·VP [0]`
`VP->·V NP [4]`
`VP->·V NP PP [4]`
	`V->"walked"· [4]`
	`VP->V·NP [4]`
	`VP->V·NP PP [4]`
	`NP->·Det N [5]`
	`NP->·Det N PP [5]`
		`Det->"an"· [5]`
		`NP->Det·N [5]`
		`NP->Det·N PP [5]`
			`N->"park"· [6]`
			`NP->Det N· [5]`
			`NP->Det N·PP [5]`
			`VP->V NP· [4]`
			`VP->V NP·PP [4]`
			`PP->·P NP [7]`
			`S->NP VP· [0]`

8	**9**		
`P->"with"· [7]`	`NP->"Bob"· [8]`		

- Chart position 8 initially contains the edge `P->"with"·[7]`:
  - We will look in chart position 7, for edges with the dot directly preceding `P`
  - There is one such edge: `PP->·P NP [7]`
- We then Predict the `NP`, and add the following edges: `NP->·Det N [8]` and `NP->·Det N PP [8]`
- Chart position 9 initially contains the edge `NP->"Bob"·[8]`:
  - We will look in chart position 8, for edges with the dot directly preceding `NP`
  - There is one such edge: `PP->P·NP [7]`
- We Complete the `PP` from chart position 7, which was generated by two different rules there: one an `NP` (`NP->Det N PP· [5]`), and one a `VP` (`VP->V NP PP· [4]`).
- The `NP`, in turn, adds the following edges from chart position 5: `VP->V NP· [4]` and `VP-> V NP·PP [4]`
- The `VP` (from completing the `PP` at 7) gives us an entire `S` (`S->NP VP· [0]`), which will correspond to a parse for this sentence.
- The `VP` (from completing the `NP` here) gives us a second parse.
- We also Predict a `PP` (`PP->·P NP [9]`), but it won't go anywhere because we've run out of input.
- We can also Complete the dummy rule `γ->S·` for completeness.
- The full chart follows below. Recovering the parse(s), with this representation of edges, is somewhat horrible, so we will not go into detail.

0	1	2	3

`γ->·S [0]`
`S->·NP VP [0]`
`NP->·Det N [0]`
`NP->·Det N PP [0]`
	`Det->"an"· [0]`
	`NP->Det·N [0]`
	`NP->Det·N PP [0]`
		`N->"park"· [1]`
		`NP->Det N· [0]`
		`NP->Det N·PP [0]`
		`S->NP·VP [0]`
		`PP->·P NP [2]`
		`VP->·V NP [2]`
		`VP->·V NP PP [2]`
			`P->"by"· [2]`
			`PP->P·NP [2]`
			`NP->·Det N [3]`
			`NP->·Det N PP [3]`

4	**5**	**6**	**7**

`NP->"Bob"· [3]`
`PP->P NP· [2]`
`NP->Det N PP· [0]`
`S->NP·VP [0]`
`VP->·V NP [4]`
`VP->·V NP PP [4]`
	`V->"walked"· [4]`
	`VP->V·NP [4]`
	`VP->V·NP PP [4]`
	`NP->·Det N [5]`
	`NP->·Det N PP [5]`
		`Det->"an"· [5]`
		`NP->Det·N [5]`
		`NP->Det·N PP [5]`
			`N->"park"· [6]`
			`NP->Det N· [5]`
			`NP->Det N·PP [5]`
			`VP->V NP· [4]`
			`VP->V NP·PP [4]`
			`PP->·P NP [7]`
			`S->NP VP· [0]`

8	**9**		

`P->"with"· [7]`
`PP->P·NP [7]`
`NP->·Det N [8]`
`NP->·Det N PP [8]`
	`NP->"Bob"· [8]`
	`PP->P NP· [7]`
	`NP->Det N PP· [5]`
	`VP->V NP PP· [4]`
	`VP->V NP· [4]`
	`VP->V NP·PP [4]`
	`S->NP VP· [0]`
	`PP->·P NP [9]`
	`γ->S· [0]`		

- You might like to compare the (coloured) edges on the chart with the parses below to confirm the structure:

```
(S   (NP   (Det an)
           (N park)
           (PP          (P by)
                        (NP Bob)
           )
     )
     (VP   (V walked)
           (NP          (Det an)
                        (N park)
                        (PP        (P with)
                                   (NP Bob)
                        )
           )
     )
)
```

```
(S   (NP   (Det an)
           (N park)
           (PP          (P by)
                        (NP Bob)
           )
     )
     (VP   (V walked)
           (NP          (Det an)
                        (N park)
           )
           (PP          (P with)
                        (NP Bob)
           )
     )
)
```

Table 1: The two trees for the table above; left corresponds to the red colour; right to blue