Parsing (part 2)

chart parsing

Chapter 13 J&M'09

(Passive) Chart-parsing

The chart contains all phrases that have been recognized.

Each chart entry contains labeled edges (arcs), e.g.:

(0,1,the) $= \underbrace{{}_0\,\text{the}\,{}_1}$

(1,5,VP) $= \underbrace{{}_1\text{VP}_5}$

(3,5,NP) $= \underbrace{{}_3\text{NP}_5}$

The numbers correspond to input token spans.

Algorithm: for each token in the input, place it as an edge on the chart (shift) and apply every grammar rule to the existing edges (complete): for all $Z \rightarrow X_1 ... X_n$ rules such that $\underbrace{{}_p\, X_1 ... X_n\, {}_q}$ are adjacent edges in the chart, add edge (p, q, Z) to the chart if edge not already in the chart.

Stop when no more rules can be applied.

Parsing

Input: $_0$ Robin $_1$ called $_2$ Sam $_3$

Chart	**Action**
–	–
(0,1,Robin)	shift
(0,1,NP) (0,1,PN) (0,1,Robin)	complete
(1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	shift
(1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	complete
(2,3,Sam) (1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	shift
(0,3,S) (1,3,VP) (2,3,NP) (2,3,PN) (2,3,Sam) (1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	complete

Parsing

Input: $_0$ Robin $_1$ called $_2$ a $_3$ friend $_4$ from $_5$ Australia $_6$

Chart	**Action**
–	–
(0,1,Robin)	shift
(0,1,NP) (0,1,PN) (0,1,Robin)	complete
(1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	shift
(1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	complete
(2,3,a) (1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	shift
(2,3,DT) (2,3,a) (1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	complete
(3,4,friend) (2,3,DT) (2,3,a) (1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	shift
(0,4,S) (1,4,VP) (2,4,NP) (3,4,N) (3,4,friend) (2,3,DT) (2,3,a) (1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	complete

Parsing

Input: $_0$ Robin $_1$ called $_2$ a $_3$ friend $_4$ from $_5$ Australia $_6$

Chart	**Action**
(0,4,S) (1,4,VP) (2,4,NP) (3,4,N) (3,4,friend) (2,3,DT) (2,3,a) (1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	
(4,5,from) (0,4,S) (1,4,VP) (2,4,NP) (3,4,N) (3,4,friend) (2,3,DT) (2,3,a) (1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	shift
(4,5,P) (4,5,from) (0,4,S) (1,4,VP) (2,4,NP) (3,4,N) (3,4,friend) (2,3,DT) (2,3,a) (1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	complete
(5,6,Australia) (4,5,P) (4,5,from) (0,4,S) (1,4,VP) (2,4,NP) (3,4,N) (3,4,friend) (2,3,DT) (2,3,a) (1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	shift
(0,6,S) (2,6,NP) (3,6,N) (1,6,VP) (4,6,PP) (5,6,NP) (5,6,PN) (5,6,Australia) (4,5,P) (4,5,from) (0,4,S) (1,4,VP) (2,4,NP) (3,4,N) (3,4,friend) (2,3,DT) (2,3,a) (1,2,TV) (1,2,called) (0,1,NP) (0,1,PN) (0,1,Robin)	complete

Prolog implementation

Use Prolog database to represent the chart:

```
scan(0,1,robin).
scan(1,2,called).
scan(2,3,sam).

arc(0,1,pn).
arc(0,1,np).
arc(1,2,tv).
arc(2,3,pn).
arc(2,3,np).
arc(1,3,vp).
arc(0,3,s).
```

Passive Bottom-Up Chart Parser (part 1):

```
1  chart_recognize_bottomup(Input) :-
2          cleanup,
3          initialize_chart(Input, 0),
4          process_chart_bottomup,
5          length(Input, N),
6          arc(0, N, s).
7
8  cleanup :-
9  retractall(scan(_,_,_)),
10 retractall(arc(_,_,_)).
11
12 initialize_chart([], _).
13 initialize_chart([Word|Input], From) :-
14         To is From + 1,
15         asserta(scan(From, To, Word)),
16         initialize_chart(Input, To).
```

Passive Bottom-Up Chart Parser (part 2):

```
1  % Apply all possible rules to input
2  process_chart_bottomup :-
3           doall( (scan(From, To, Word),
4                lex(Cat,Word),
5                add_arc(arc(From, To, Cat)))    ).
6
7  doall(Goal) :-  Goal, fail.
8  doall(_) :- true.
9
10 % Add new arcs to memory
11 add_arc(Arc) :-
12  \+ Arc,
13   assert(Arc),
14    new_arcs(Arc).
```

Passive Bottom-Up Chart Parser (part 3):

```
1  % Find new arcs to add from previous arcs + rules
2  new_arcs(arc(J, K, Cat)) :-
3           doall(  (rule(LHS,RHS),
4                       append(Before, [Cat], RHS),
5                       path(I, J, Before),
6                       add_arc(arc(I, K, LHS)))  ).
7
8  % Find the category for a previously parsed  span
9  path(I, I, []).
10 path(I, K, [Cat|Cats]) :-
11          arc(I, J, Cat),   J =< K,  path(J, K, Cats).
```

Parsing

How much better is this parser?

(1) The captain said you claimed the sergeant denied the antenna in the hangar broke in the attack near the gate.

(14 parses)

Parser	**Steps**	**Time**
Shift-reduce	19129	0.01 secs
Left-corner	5453	0.01 secs
Passive BUP Chart	12157	0 secs

(on a 2.4 Ghz Intel Core 2 Duo with 4 GB 1067 MHz DDR4)

Active Chart-parsing (Earley Parsing)
A chart is a record of all phrases that have been recognized (passive edges) and of all phrases that have been partially recognized (active edges).

Examples of active edges:

$\underbrace{{}_1\,\text{Robin}\,{}_2}_{\text{S}\rightarrow\text{NP}\bullet\text{VP}}$ (found NP, may be an S if there is an adjacent VP)

$\underbrace{{}_3\,\text{said}\,{}_4\,\text{Tom}_5}_{\text{VP}\rightarrow\text{SV}\bullet\text{S}}$ (found SV, may be a VP if there is an adjacent S)

$\overset{0}{\underbrace{}}$ (may be an S if there are adjacent NP VP)

$\text{S}\rightarrow\bullet\text{NP VP}$

The Fundamental Rule (of active parsing): allows us to combine active edges with passive edges:

From
$(n_1, n_2, [X \rightarrow W_0 ... W_i \bullet Y\, K_0 ... K_j])$
$(n_2, n_3, [Y \rightarrow Z_1 ... Z_p \bullet])$

create new edge:
$(n_1, n_3, [X \rightarrow W_0 ... W_i\, Y \bullet K_0 ... K_j])$

For example, from the edges

$(4, 8, [\text{VP} \rightarrow \text{DTV NP} \bullet \text{PP}])$
$(8, 10, [\text{PP} \rightarrow \text{P NP} \bullet])$
create:
$(4, 10, [\text{VP} \rightarrow \text{DTV NP PP} \bullet])$

Parsing

Active edges are initially stored in an **agenda**, not in the chart.

Active edges are removed from the agenda, added to the chart, and used to create new edges.

- if the agenda is treated as a stack, the parser employs a depth-first strategy
- if the agenda is treated as a queue, the parser employs a breadth-first strategy
- if the elements are ordered in the agenda by probability, then the parser will employs a hybrid search strategy.

Active Chart-parsing (Earley Parsing)

1. Initialize chart (find all possible edges for the input)
2. Initialize agenda (introduce all S rules)
3. Repeat until no other action can be done:
    1. Remove an edge X from agenda;
    2. Add edge X to chart if X is not already on the chart;
    3. If possible, use the fundamental rule to X and other edges in the chart to create new edges in the agenda. (**complete**)
    4. If possible, make new active edges based on X and the grammar rules, then add new active edges to the agenda. (**predict**)
4. If the chart contains a passive edge from the first node to the last node that has the label 'S' then succeed. Otherwise, fail.

Parsing

Input: 'Robin sneezed'

Chart	**Agenda**	**Action**
(1,2,IV) (0,1,PN)	(0,0. S → • NP VP)	Initialize
(0,0, S → • NP VP) (1,2,IV) (0,1,PN)	(0,0, NP → • PN) (0,0, NP → • DT N)	Predict
(0,0, NP → • PN) (0,0, S → • NP VP) (1,2,IV) (0,1,PN)	(0,1, NP → PN•) (0,0, NP → • DT N)	Complete
(0,1, NP → PN •) (0,0, NP → • PN) (0,0, S → • NP VP) (1,2,IV) (0,1,PN)	(0,1, S → NP • VP) (0,0, NP → • DT N)	Complete
(0,1, S → NP•VP) (0,1, NP → PN•) (0,0, NP → • PN) (0,0, S → • NP VP) (1,2,IV) (0,1,PN)	(1,1, VP → •IV) (1,1, VP → •TV NP) (0,0, NP → • DT N)	Predict
(1,1, VP → •IV) (0,1, S → NP•VP) (0,1, NP → PN•) (0,0, NP → • PN) (0,0, S → • NP VP) (1,2,IV) (0,1,PN)	(1,2, VP → IV•) (1,1, VP → •TV NP) (0,0, NP → • DT N)	Complete

Parsing

Chart	**Agenda**	**Action**
(1,1, VP → •IV) (0,1, S → NP•VP) (0,1, NP → PN•) (0,0, NP → • PN) (0,0, S → • NP VP) (1,2,IV) (0,1,PN)	(1,2, VP → IV•) (1,1, VP → •TV NP) (0,0, NP → • DT N)	Complete
(1,2, VP → IV•) (1,1, VP → •IV) (0,1, S → NP•VP) (0,1, NP → PN•) (0,0, NP → • PN) (0,0, S → • NP VP) (1,2,IV) (0,1,PN)	(0,2, S → NP VP•) (1,1, VP → •TV NP) (0,0, NP → • DT N)	Complete
(1,1, VP → •TV NP) **(0,2, S → NP VP•)** (1,1, VP → •IV) (0,1, S → NP•VP) (0,1, NP → PN•) (0,0, NP → • PN) (0,0, S → • NP VP) (1,2,IV) (0,1,PN)	(0,0, NP → • DT N)	
(0,0, NP → • DT N) (1,1, VP → •TV NP) **(0,2, S → NP VP•)** (1,1, VP → •IV) (0,1, S → NP•VP) (0,1, NP → PN•) (0,0, NP → • PN) (0,0, S → • NP VP) (1,2,IV) (0,1,PN)		

Parsing

```
1  % main predicate
2  active_chart_recognize(Input) :-
3          cleanup,
4          initialize_chart_topdown(Input, 0),
5          initialize_agenda_topdown(Agenda),
6          process_agenda(Agenda),
7          length(Input, N),
8          arc(0, N, s, _, []).
9
10 % delete any scans and arcs from previous parses
11 cleanup :-
12         retractall(scan(_,_,_)),
13         retractall(arc(_,_,_,_,_)).
```

Parsing

```
1  % Shift input to chart and  create agenda
2  initialize_chart_topdown([], _).
3
4  initialize_chart_topdown([Word|Input], From) :-
5         To is From + 1,
6         assert(scan(From, To, Word)),
7         doall( (lex(Cat,Word), assert(arc(From,To,Cat,[Word],[]))
              ) ),
8         initialize_chart_topdown(Input, To).
9
10 % Add all S rules to agenda
11 initialize_agenda_topdown(Agenda) :-
12        findall(arc(0, 0, s, [], RHS), rule(s,RHS), Agenda).
```

Parsing

```
1  % Handle agenda
2  process_agenda([]).
3
4  process_agenda([Arc|Agenda]) :-
5          \+ Arc,!,
6          assert(Arc),
7          make_new_arcs_topdown(Arc, NewArcs),
8          append(NewArcs, Agenda, NewAgenda),
9         process_agenda(NewAgenda).
10
11 % Descend through agenda
12 process_agenda([_|Agenda]) :-
13       process_agenda(Agenda).
```

Parsing

```
1 make_new_arcs_topdown(Arc, NewArcs) :-
2         Arc = arc(_,_,_,_,[_|_]),
3         apply_fundamental_rule(Arc, NewArcs1),
4         predict_new_arcs_topdown(Arc, NewArcs2),
5         append(NewArcs1,NewArcs2,NewArcs).
6
7 make_new_arcs_topdown(Arc, NewArcs) :-
8       Arc = arc(_,_,_,_,[]),
9       apply_fundamental_rule(Arc, NewArcs).
```

Parsing

```
1  apply_fundamental_rule(arc(I, J, Cat, Done, [SubCat|SubCats]),
       NewArcs) :-
2       findall(arc(I, K, Cat, [SubCat|Done], SubCats),
3                   arc(J, K, SubCat, _, []),
4                  NewArcs ).
5
6  apply_fundamental_rule(arc(J, K, Cat, _, []), NewArcs) :-
7       findall(arc(I, K, SuperCat, [Cat|Done], Cats),
8                   arc(I, J, SuperCat, Done, [Cat|Cats]),
9                  NewArcs ).
10
11 predict_new_arcs_topdown(arc(_, J, _, _, [ToFindCat|_]), NewArcs)
       :-
12       findall(arc(J, J, ToFindCat, [], RHS),
13                   rule(ToFindCat,RHS),
14                   NewArcs ).
```

J&M09's version of Earley parsing: 'book that flight'

Chart[0]	S0	$\gamma \rightarrow \bullet S$	[0,0]	Dummy start state
	S1	$S \rightarrow \bullet NP\ VP$	[0,0]	Predictor
	S2	$S \rightarrow \bullet Aux\ NP\ VP$	[0,0]	Predictor
	S3	$S \rightarrow \bullet VP$	[0,0]	Predictor
	S4	$NP \rightarrow \bullet Pronoun$	[0,0]	Predictor
	S5	$NP \rightarrow \bullet Proper\text{-}Noun$	[0,0]	Predictor
	S6	$NP \rightarrow \bullet Det\ Nominal$	[0,0]	Predictor
	S7	$VP \rightarrow \bullet Verb$	[0,0]	Predictor
	S8	$VP \rightarrow \bullet Verb\ NP$	[0,0]	Predictor
	S9	$VP \rightarrow \bullet Verb\ NP\ PP$	[0,0]	Predictor
	S10	$VP \rightarrow \bullet Verb\ PP$	[0,0]	Predictor
	S11	$VP \rightarrow \bullet VP\ PP$	[0,0]	Predictor
Chart[1]	S12	$Verb \rightarrow book \bullet$	[0,1]	Scanner
	S13	$VP \rightarrow Verb \bullet$	[0,1]	Completer
	S14	$VP \rightarrow Verb \bullet NP$	[0,1]	Completer
	S15	$VP \rightarrow Verb \bullet NP\ PP$	[0,1]	Completer
	S16	$VP \rightarrow Verb \bullet PP$	[0,1]	Completer
	S17	$S \rightarrow VP \bullet$	[0,1]	Completer
	S18	$VP \rightarrow VP \bullet PP$	[0,1]	Completer
	S19	$NP \rightarrow \bullet Pronoun$	[1,1]	Predictor
	S20	$NP \rightarrow \bullet Proper\text{-}Noun$	[1,1]	Predictor
	S21	$NP \rightarrow \bullet Det\ Nominal$	[1,1]	Predictor
	S22	$PP \rightarrow \bullet Prep\ NP$	[1,1]	Predictor

J&M09's version of Earley parsing

Chart[2]	S23	*Det* → *that* •	[1,2]	Scanner
	S24	*NP* → *Det* • *Nominal*	[1,2]	Completer
	S25	*Nominal* → • *Noun*	[2,2]	Predictor
	S26	*Nominal* → • *Nominal Noun*	[2,2]	Predictor
	S27	*Nominal* → • *Nominal PP*	[2,2]	Predictor
Chart[3]	S28	*Noun* → *flight* •	[2,3]	Scanner
	S29	*Nominal* → *Noun* •	[2,3]	Completer
	S30	*NP* → *Det Nominal* •	[1,3]	Completer
	S31	*Nominal* → *Nominal* • *Noun*	[2,3]	Completer
	S32	*Nominal* → *Nominal* • *PP*	[2,3]	Completer
	S33	*VP* → *Verb NP* •	[0,3]	Completer
	S34	*VP* → *Verb NP* • *PP*	[0,3]	Completer
	S35	*PP* → • *Prep NP*	[3,3]	Predictor
	S36	*S* → *VP* •	[0,3]	Completer
	S37	*VP* → *VP* • *PP*	[0,3]	Completer

Parsing

This top-down parser has no problems with recursion.

Input 'A cat sneezed'

Chart	Agenda	
(0,1,DT) (1,2,N) (2,3,IV)	(0,0, S → • NP VP)	
(0,0, S → • NP VP) ...	(0,0, NP → • DT N)	
(0,0, NP → • DT N) ...	(0,1, NP → DT• N)	
(0,1, NP → DT• N) ...	(1,1, N → • N PP) (0,2, NP → DT N •)	
(1,1, N → • N PP) ...	(1,1, N → N • PP) (0,2, NP → DT N •)	
(1,1, N → N • PP) ...	(1,1, PP → • P NP) (0,2, NP → DT N •)	← This yields no more rules

Parsing

Naive top-down parsing falls prey to infinite loops

Input	**Stack**	**Actions**
a cat sneezed	S	initial state
a cat sneezed	NP VP	replace S with NP VP
a cat sneezed	DT N VP	replace NP with DT N
cat sneezed	N VP	match DT
cat sneezed	N PP VP	replace N with N PP
cat sneezed	N PP PP VP	replace N with N PP
cat sneezed	N PP PP PP VP	replace N with N PP
cat sneezed	N PP PP PP PP VP	replace N with N PP
...	...	...

Grammar:

S → NP VP NP → DP N

N → N PP PP → P NP

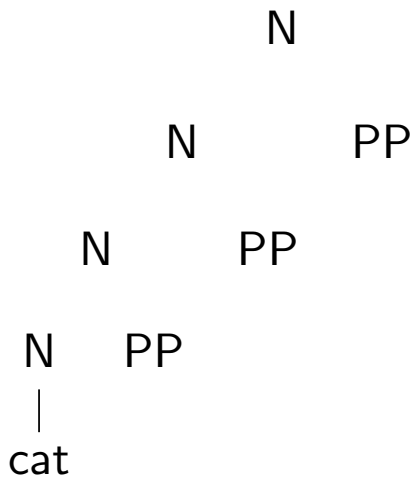
P → in IV → sneezed

Parsing

Standard solution: eliminate left-recursive rules.

$N \rightarrow N\ PP$

$N \rightarrow cat$



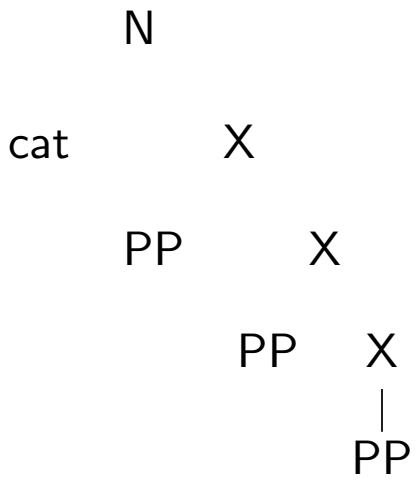
Grammar re-writing:

$N \rightarrow cat$

$N \rightarrow cat\ X$

$X \rightarrow PP\ X$

$X \rightarrow PP$



The Earley active top-down chart parser can be bottom-up:

1. Make initial chart and agenda as before.
2. Repeat until agenda is empty:
    1. Remove an edge X from agenda;
    2. Add X it to chart if X it is not already on the chart;
    3. If possible, use the fundamental rule to combine X and other edges to create new edges in the agenda (**complete**)
    4. If possible, make new active edges about new constituents formed with X and the rules of the grammar, add these to the agenda. (**predict**)
3. See if the chart contains a passive edge from the first node to the last node that has the label S.

Parsing

How much better are these parsers?

(2) The captain said you claimed the sergeant denied the antenna in the hangar broke in the attack near the gate.

(14 parses)

Parser	**Steps**	**Time**
Shift-reduce	19129	0.01 secs
Left-corner	5453	0.01 secs
Passive BUP Chart	12157	0 secs
Active TPD Chart	13599	0 secs
Active BUP Chart	12946	0 secs

(on a 2.4 Ghz Intel Core 2 Duo with 4 GB 1067 MHz DDR4)

... we need a bigger sample.

Parsing

SR:

Steps	**Time**	**Parses**
819	0	3
1754	0.01	4
19129	0.02	14
359378	0.38	165
7099488	7.56	2002
142121092	151.25	25194
?	?	$\pm$230000

LC:

Steps	**Time**	**Parses**
445	0	3
881	0	4
5453	0.01	14
70067	0.05	165
989642	0.68	2002
14396672	9.98	25194
?	?	$\pm$230000

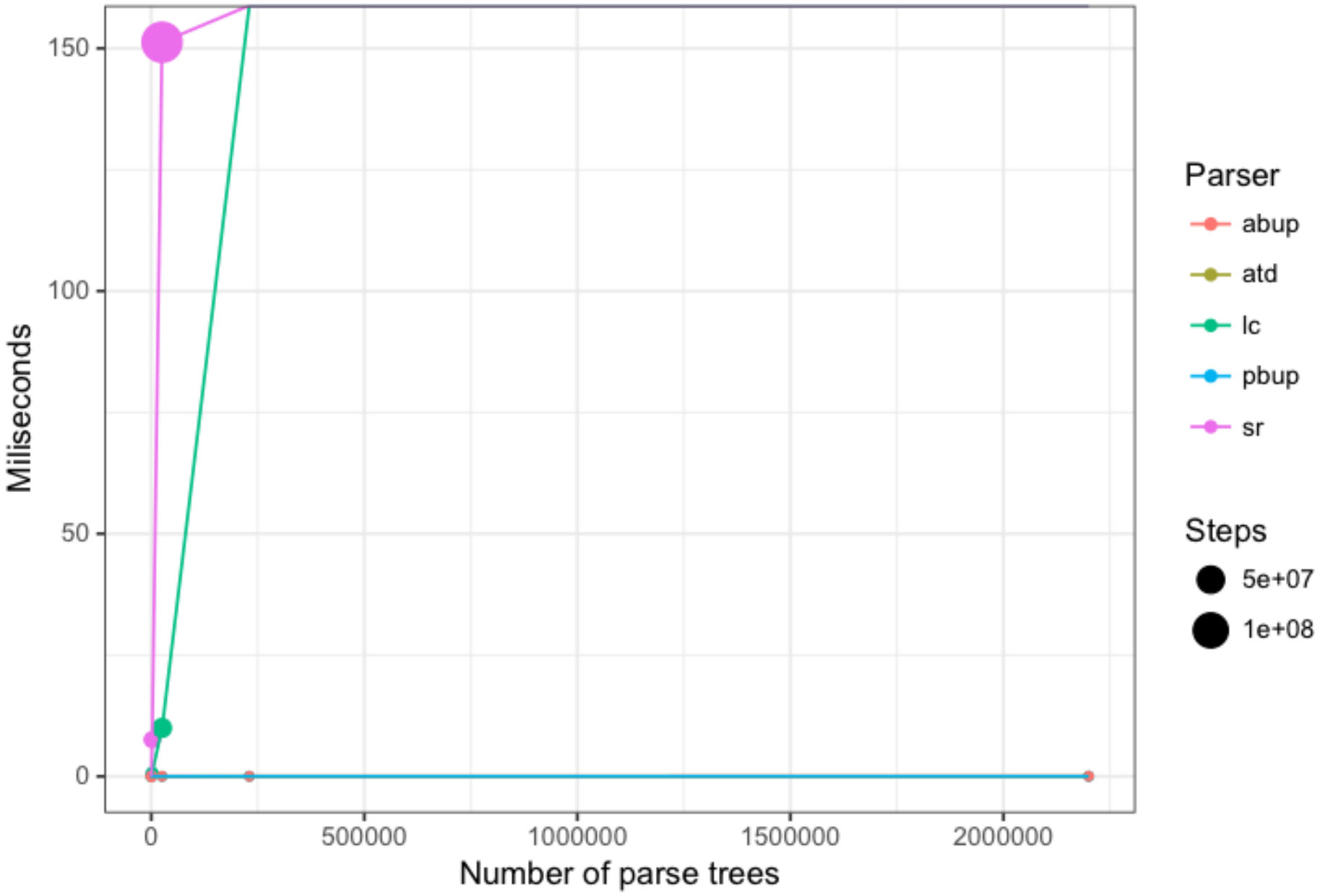
PassiveBUP:

Steps	**Time**	**Parses**
1234	0	3
1613	0.01	4
2904	0	14
5332	0	165
8972	0.01	2002
14192	0.01	25194
21424	0.02	$\pm$230000
31164	0.02	$\pm$2800000

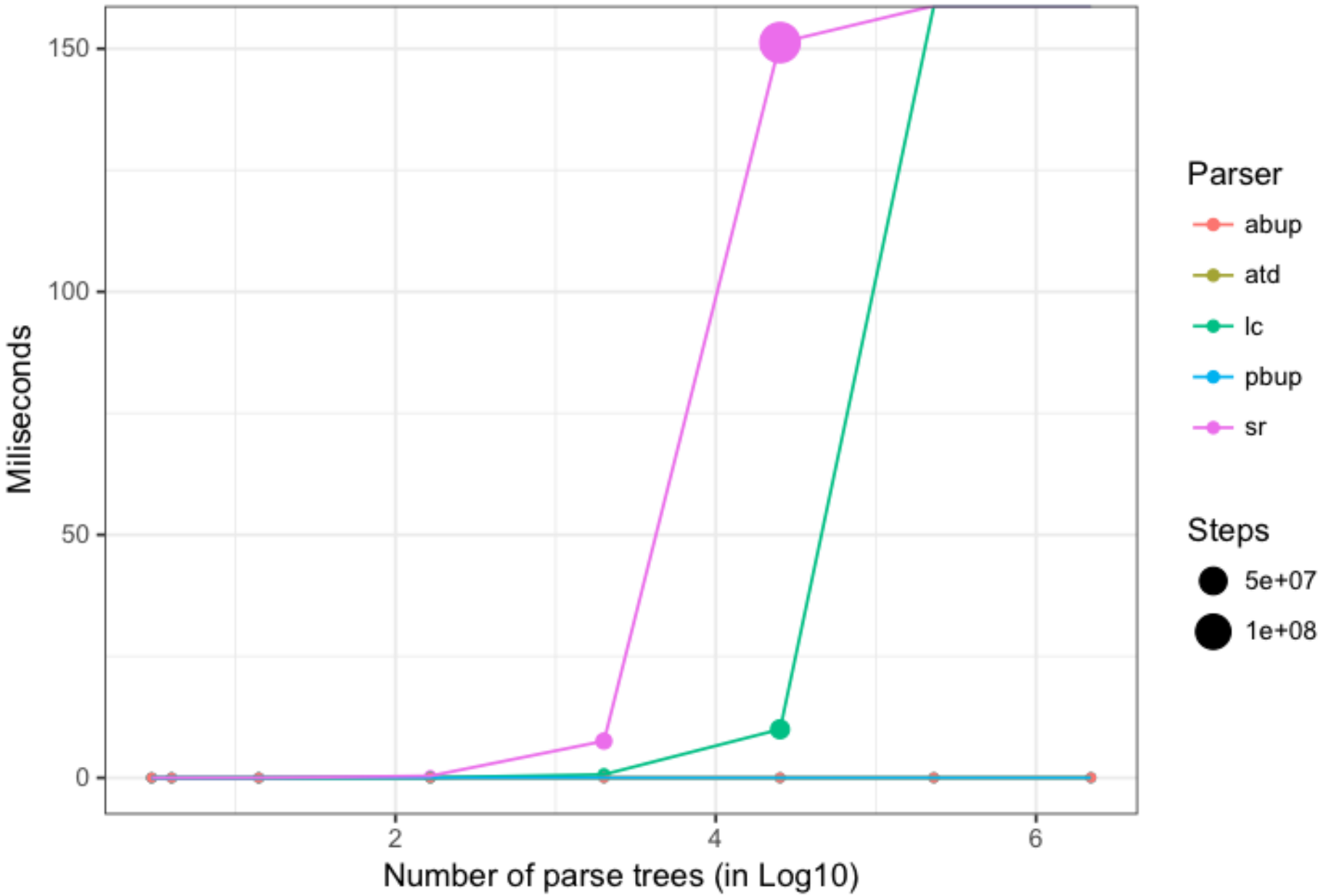
ActiveBUP:

Steps	**Time**	**Parses**
1781	0.01	3
2312	0	4
3839	0	14
6408	0.01	165
9705	0.01	2002
13770	0.02	25194
18643	0.02	$\pm$230000
24364	0.04	$\pm$2800000

Parsing


150
100
50
0
Miliseconds
0
500000
1000000
1500000
2000000
Number of parse trees
Parser
abup
atd
lc
pbup
sr
Steps
5e+07
1e+08

Parsing


150
100
50
0
Miliseconds
2
4
6
Number of parse trees (in Log10)
Parser
abup
atd
lc
pbup
sr
Steps
5e+07
1e+08

Parsing

Standard for grammar profiling: [incr tsdb()] package
See pages 24 and forward of the manual.