— I'm looking for an S.
— To get an S, I need an NP and a VP.
— To get an NP, I need a D and an N.
— To get a D, I can use *the* ... Got it.
— To get an N, I can use *dog* ... Got it.
— That completes the NP.
— To get a VP, I need a V.
— To get a V, I can use *barked* ... Got it.
— That completes the VP.
— That completes the S.

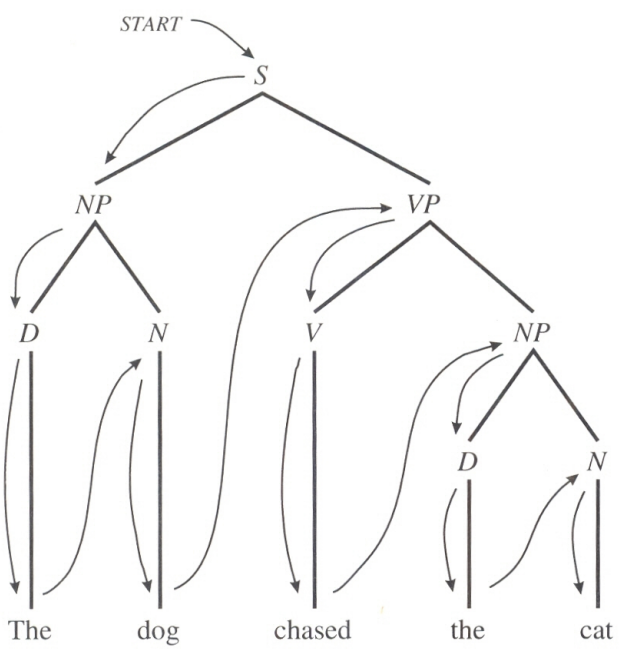

Figure 6.2 Top-down parsing. The parser discovers the nodes of the tree in the order shown by the arrows.

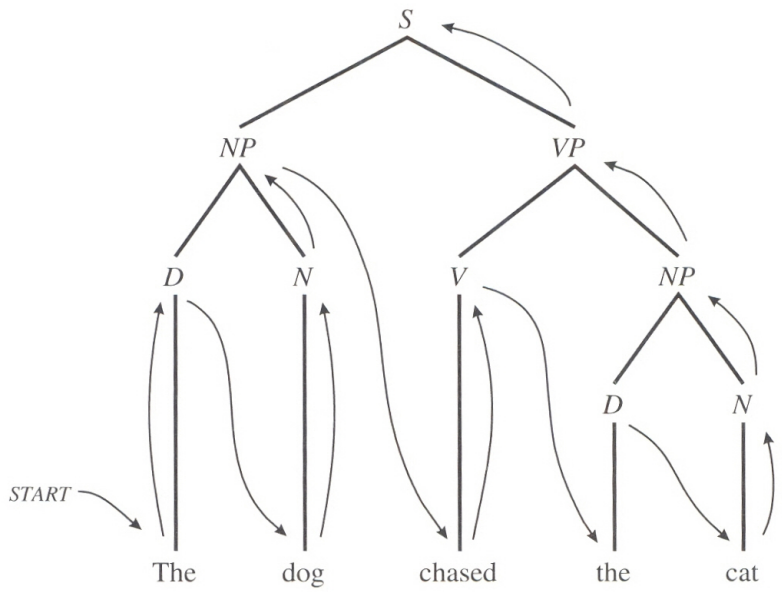

Figure 6.3 Bottom–up parsing. The parser discovers the nodes of the tree in the order shown by the arrows.

Because its actions are triggered only by words actually found, this parser does not loop on left-recursive rules. But it has a different limitation: it cannot handle rules like

$$D \rightarrow \emptyset$$

because it has no way of responding to a null (empty, missing) constituent. It can only respond to what's actually there.

Step	Action	Stack	Input string
	(Start)		*the dog barked*
1	Shift	*the*	*dog barked*
2	Reduce	*D*	*dog barked*
3	Shift	*D dog*	*barked*
4	Reduce	*D N*	*barked*
5	Reduce	*NP*	*barked*
6	Shift	*NP barked*	
7	Reduce	*NP V*	
8	Reduce	*NP VP*	
9	Reduce	*S*	

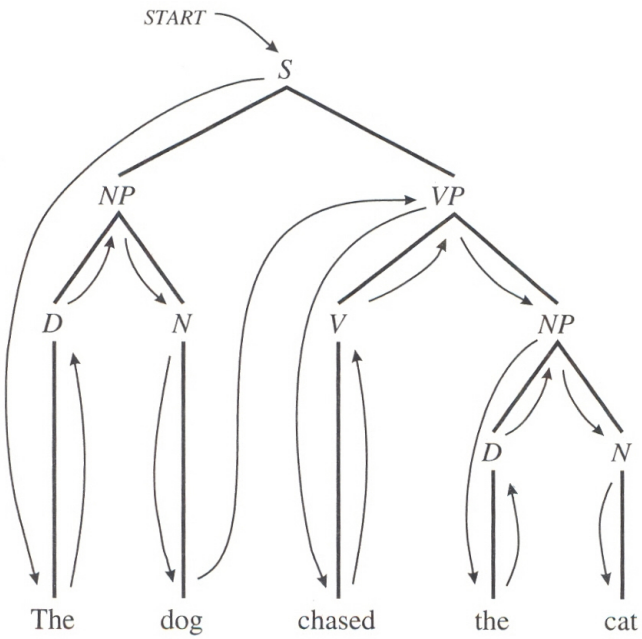

Figure 6.5 Left-corner parsing. Note that some nodes are visited twice, once working top-down and once working bottom-up.

1. **Accept** a word from the input string and determine its category. Call its category W.
2. **Complete** C. If W = C, you're done. Otherwise,

   - Look at the rules and find a constituent whose expansion begins with W. Call that constituent P (for "phrase"). For example, if W is Determiner, use the rule $NP \rightarrow D\ N$ and let P be Noun Phrase.
   - Recursively left-corner-parse all the remaining elements of the expansion of P. (This is the top-down part of the strategy.)
   - Last, put P in place of W, and go back to the beginning of step 2 (i.e., start over trying to complete C).

Earley's Algorithm

Start with: `chart(start,[the,dog,chases,the,cat],[s],[the,dog,chases,the,cat] ).`

Predict: Use $S \rightarrow NP\ VP$ and $NP \rightarrow D\ N$.
`chart(s,[the,dog,chases,the,cat],[ np,vp],[the,dog,chases,the,cat]).`
`chart(np,[the,dog,chases,the,cat],[d,n],[the,dog,chases,the,cat]).`

Scan: Accept *the*.
`chart(np,[the,dog,chases,the,cat],[n],[dog,chases,the,cat]).`

Complete: Nothing to do; no phrase has been completed.

Predict: Nothing to do; there are no rules expanding N.

Scan: Accept *dog*.
`chart(np,[the,dog,chases,the,cat],[],[chases,the,cat]).`

Complete: An NP has now been parsed.
`chart(s,[the,dog,chases,the,cat],[vp],[chases,the,cat]).`

Predict: Use $VP \rightarrow V$ and $VP \rightarrow V\ NP$.
`chart(vp,[chases,the,cat],[v],[chases,the,cat]).`
`chart(vp,[chases,the,cat],[v,np],[chases,the,cat]).`

Scan: Accept *chases*.
`chart(vp,[chases,the,cat],[],[the,cat]).`
`chart(vp,[chases,the,cat],[np],[the,cat]).`

Complete: According to $VP \rightarrow V$, a VP and hence the S have now been parsed.
`chart(s,[the,dog,chases,the,cat],[],[the,cat]).`
`chart(start,[the,dog,chases,the,cat],[],[the,cat]).`

Predict: The other VP rule is still looking for an NP. Expand it...
`chart(np,[the,cat],[d,n],[the,cat]).`

Scan: Accept *the*.
`chart(np,[the,cat],[n],[cat]).`

Complete: Nothing to do—no phrase has been completed.

Predict: Nothing to do—there are no rules expanding N.

Scan: Accept *cat*.
`chart(np,[the,cat],[],[]).`

Complete: Now the NP, and hence the VP and S, have been parsed.
`chart(vp,[chases,the,cat],[],[]).`
`chart(s,[the,dog,chases,the,cat],[],[]).`
`chart(start,[the,dog,chases,the,cat],[],[]).`
All done.