An Introduction to Minimalist Grammars

Gregory M. Kobele
Humboldt-Universität zu Berlin
University of Chicago

Jens Michaelis
Universität Bielefeld

ESSLLI 2009
Bordeaux

Outline

- Use MGs to introduce you to something like a *status quo* in Minimalism
- Will investigate:
  - raising to subject
  - raising to object
  - passive
  - Expletive-*it*
- While details of these analyses might not coincide with those of every other minimalist analysis of these phenomena, they will at least be easily recognizable in these
- Sometimes, as with *it*-insertion, the formal precision of MGs allows us to formulate proposals which have yet to gain wide acceptance in the linguistic literature

Head movement in the auxiliary system

- We begin with simple intransitive sentences, such as the below.

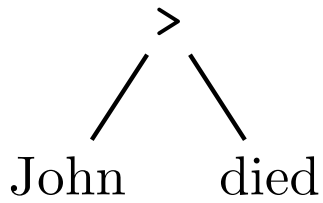
1. John died.
2. John will die.
3. John had died.
4. John has been dying.

⋮

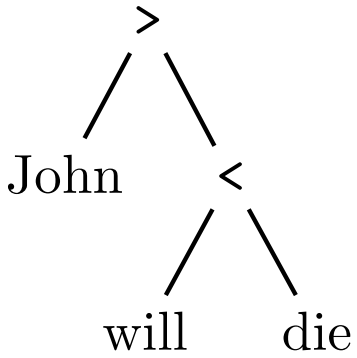
Head movement in the auxiliary system

- We treat these sentences as being divided into a subject (*John*), and a predicate (the rest). The predicate is treated as right branching, with elements to the left projecting over those to their right.

1. John died.



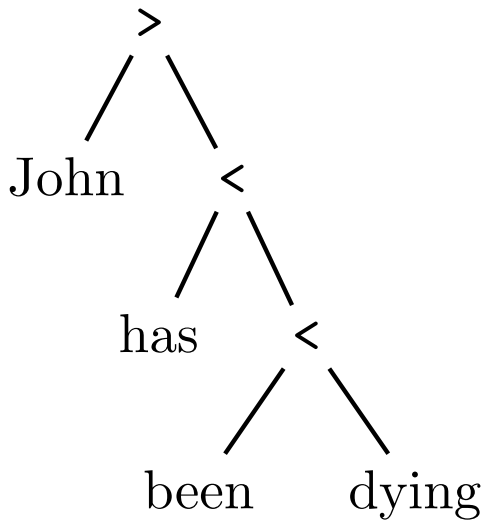
2. John will die.



Head movement in the auxiliary system

- A slightly bigger example. . .

4. John has been dying.

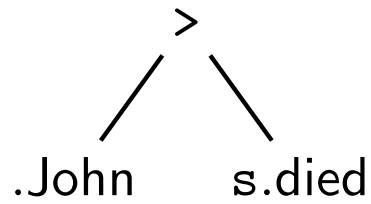


Head movement in the auxiliary system

- We want our grammar to generate these expressions.
- Recall: to specify a grammar, we need to specify four things:
  1. The features
     (which features we will use in our grammar)
  2. The lexicon
     (which syntactic feature sequences are assigned to which words)
  3. The grammatical operations
     (here, these will always be **move**, **merge**, and **agree**, so I will leave them implicit in the following)
  4. The start category
     (what is the category of complete sentences)
     - Breaking with tradition, I will call the start category `s` – it reminds me of sentence, as well as start!
- Thus, all that is left is to determine the features we will use, and the lexical items we have

Head movement in the auxiliary system

- Given a sentence like the below, we know that its head must have category s, and that no other leaves may have syntactic features.



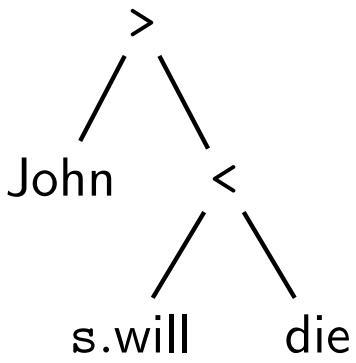
- What features must *John* and *died* have in order to combine into the structure above of category s?
- We can only build the above structure from lexical items of the following shape:

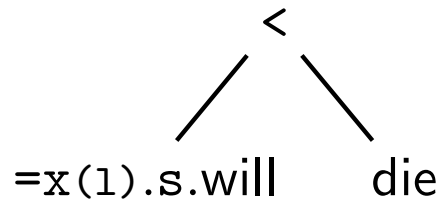
x.John =x(l).s.died

- What should 'x' be? It doesn't matter! All that matters is whether two features match, not what they are called. Let's take 'x' to be 'd' (for 'DP'), as a nod to tradition.

Head movement in the auxiliary system

- We can perform the same line of reasoning on the structure on the left below, too.





- The structure on the left must be the result of merging a lexical item x.John with the structure on the right
- This righthand structure then must be the result of merging the following two lexical items.

=y(r).=x(l).s.will y.die

- As feature names don't matter, lets call 'y' 'v', and 'x' 'd'.

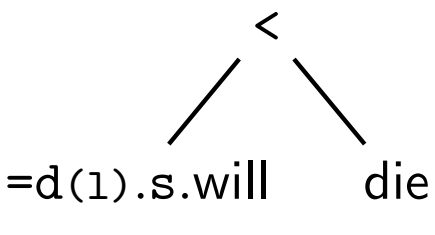
d.John =v(r).=d(l).s.will v.die

Is this right? – A sanity check

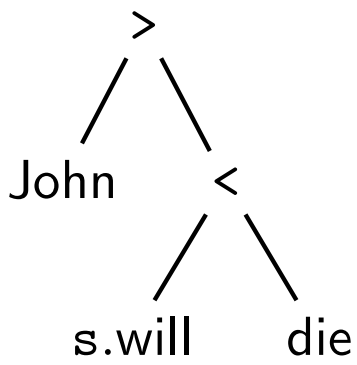
- So we have decomposed the tree we assigned to the sentence *John will die* into the three lexical items below – Let's make sure they allow us to derive this sentence!

d.John =v(r).=d(l).s.will v.die

1. **merge**(=v(r).=d(l).s.will, v.die) =

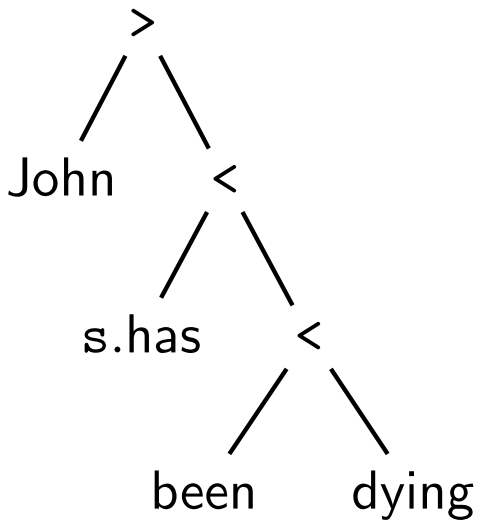


2. **merge**(1, d.John) =



Head movement in the auxiliary system

- In the same way, from a structure like that below, we obtain the following lexical items:



d.John	=perf(r).=d(l).s.has
=prog(r).perf.been	prog.dying

Head movement in the auxiliary system

- In this way, from the sentences below, we arrive at the following set of lexical items, which determine a grammar.

John dies
John died
John will die
John has died
John had died
John is dying

John was dying
John has been dying
John had been dying
John will be dying
John will have died
John will have been dying

v.die	=v(r).=d(l).s.will	=prog(r).=d(l).s.is
perf.died		=prog(r).=d(l).s.was
prog.dying	=perf(r).v.have	=prog(r).v.be
=d(l).s.died	=perf(r).=d(l).s.has	=prog(r).perf.been
=d(l).s.dies	=perf(r).=d(l).s.had	

Head movement in the auxiliary system

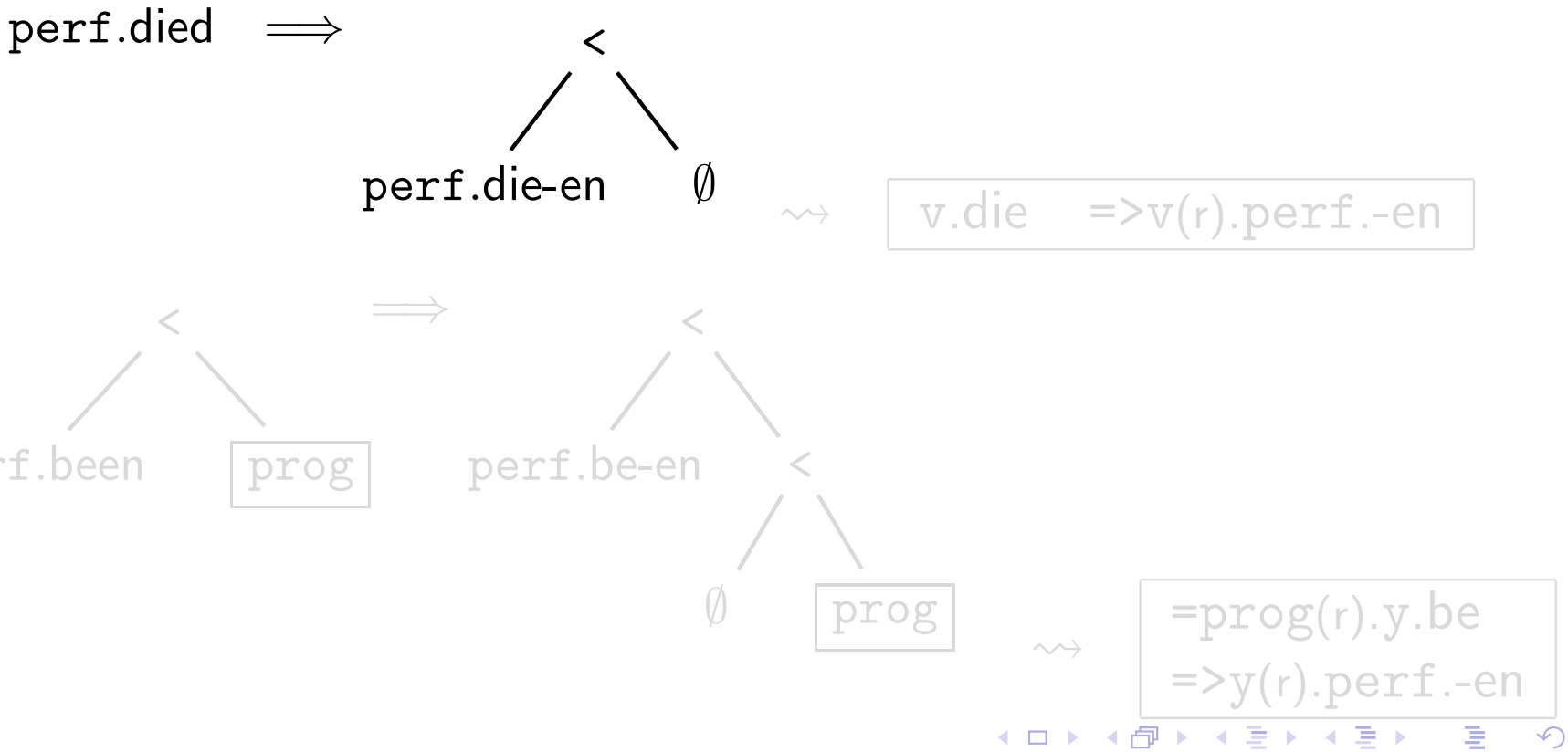
v.die	=v(r).=d(l).s.will	=prog(r).=d(l).s.is
perf.died		=prog(r).=d(l).s.was
prog.dying	=perf(r).v.have	=prog(r).v.be
=d(l).s.died	=perf(r).=d(l).s.has	=prog(r).perf.been
=d(l).s.dies	=perf(r).=d(l).s.had	

- These lexical items are highly redundant:
  1. all of the *be* forms select for something in the progressive
  2. all the *have* forms something in the perfective
  3. all and only the tensed forms (*died*,*dies*, *has*, *had*, . . .) select an argument
- Whenever a new verb is added to the language, we need to add five new lexical items:

v.laugh	perf.laughed
prog.laughing	=d(l).s.laughed
=d(l).s.laughs	

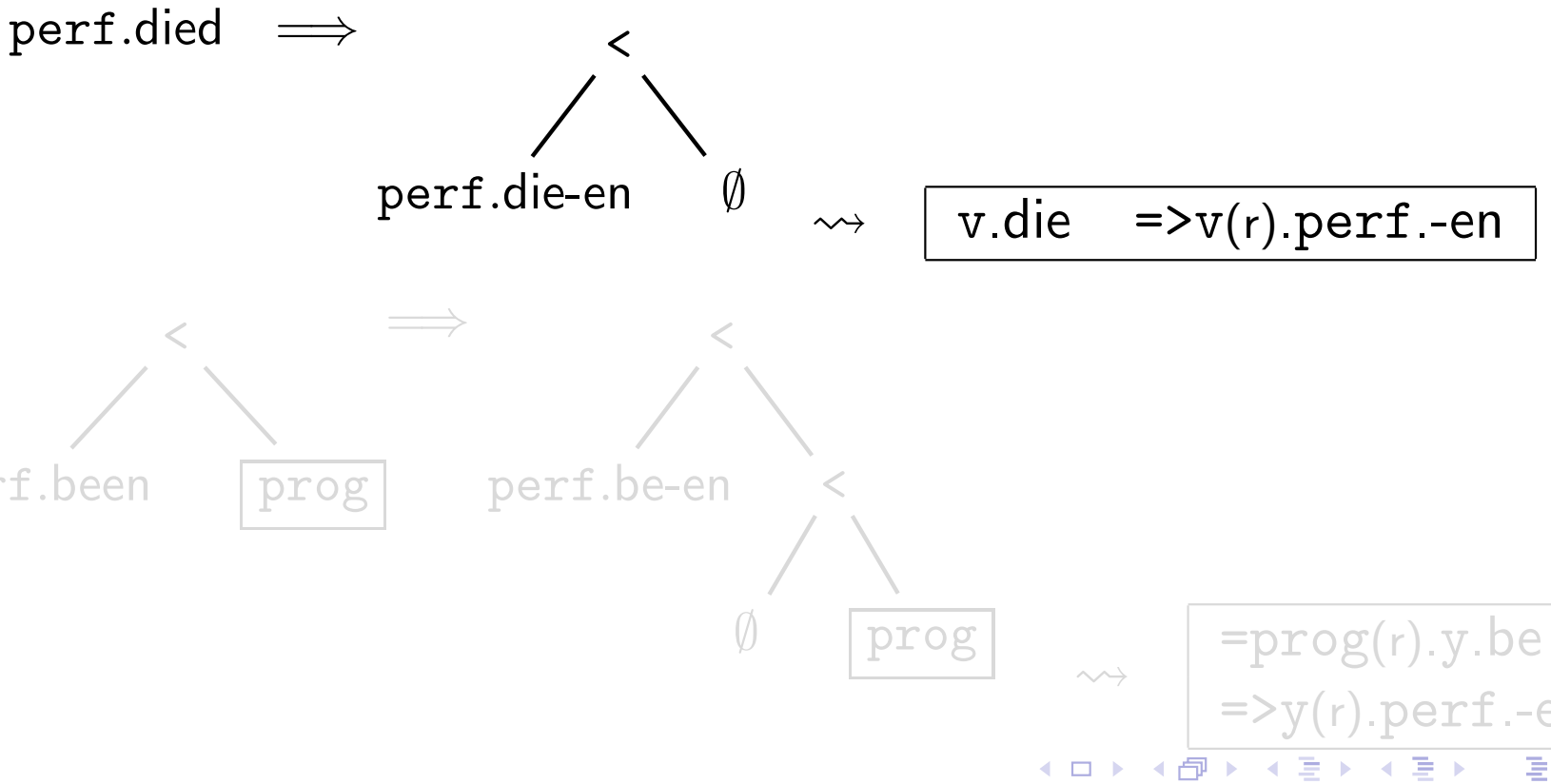
Head movement in the auxiliary system

- Let's begin by looking at lexical items of category `perf` (*died* and *been*, but also *broken*,. . .)
- Instead of looking at these expressions as lexical items, let's think of them as containing the perfective suffix *-en* as well as a verb (*die*) or auxiliary (*be*)



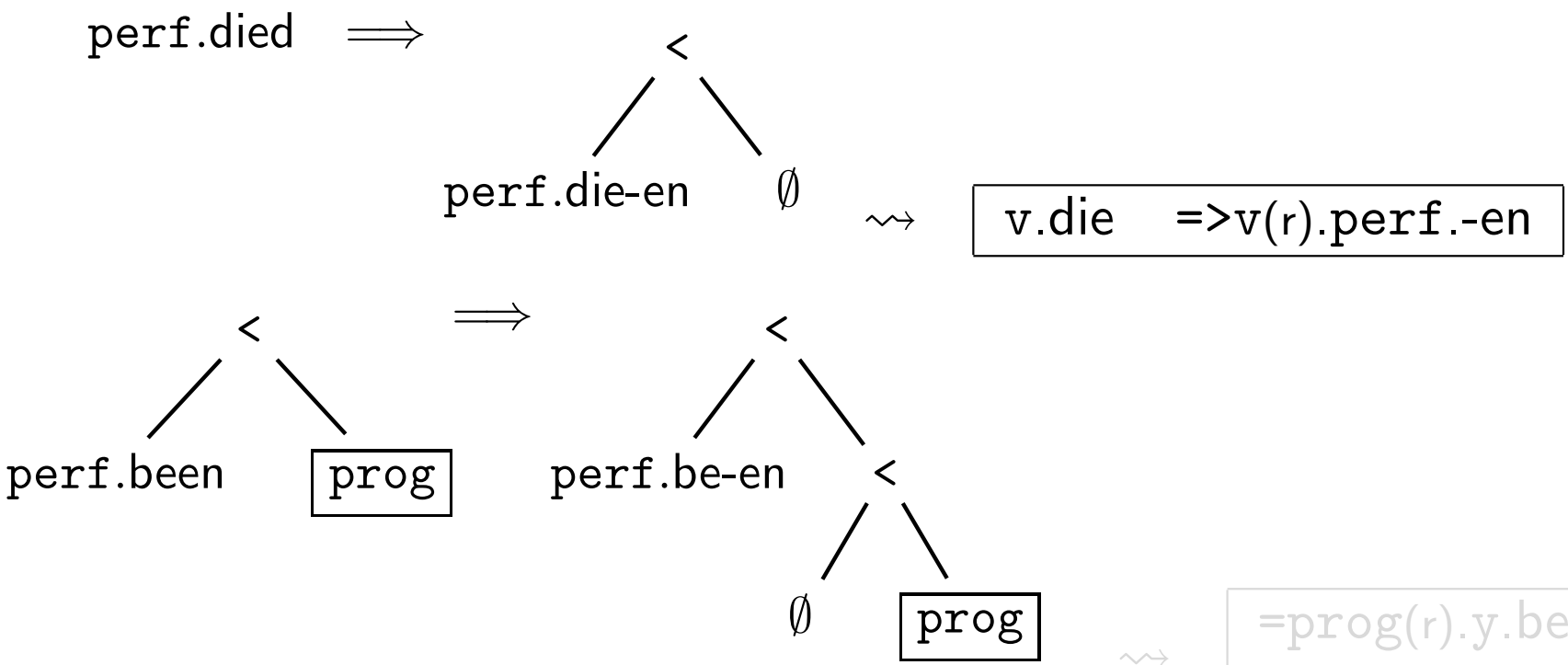
Head movement in the auxiliary system

- Let's begin by looking at lexical items of category `perf` (*died* and *been*, but also *broken*,. . .)
- Instead of looking at these expressions as lexical items, let's think of them as containing the perfective suffix *-en* as well as a verb (*die*) or auxiliary (*be*)



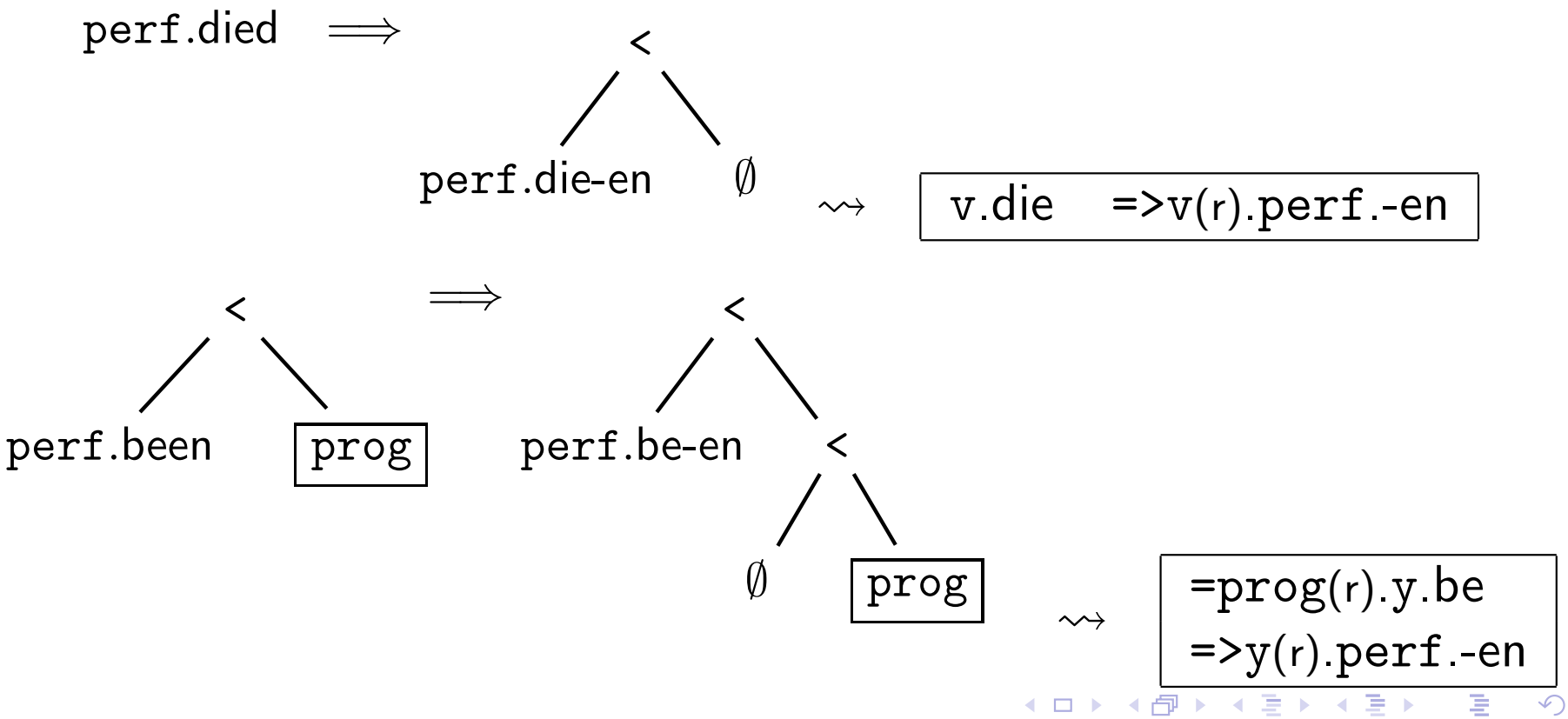
Head movement in the auxiliary system

- Let's begin by looking at lexical items of category `perf` (*died* and *been*, but also *broken*,. . .)
- Instead of looking at these expressions as lexical items, let's think of them as containing the perfective suffix *-en* as well as a verb (*die*) or auxiliary (*be*)



Head movement in the auxiliary system

- Let's begin by looking at lexical items of category `perf` (*died* and *been*, but also *broken*,. . .)
- Instead of looking at these expressions as lexical items, let's think of them as containing the perfective suffix *-en* as well as a verb (*die*) or auxiliary (*be*)



Head movement in the auxiliary system

- Similarly, we decompose
  - =>perf(r).=d(l).s.has $\rightsquigarrow$
    =perf(r).x.have
    and =>x(r).=d(l).s.-s
  - =>perf(r).=d(l).s.had $\rightsquigarrow$
    =perf(r).x.have
    and =>x(r).=d(l).s.-ed
  - =>prog(r).=d(l).s.is $\rightsquigarrow$
    =prog(r).y.be
    and =>y(r).=d(l).s.-s
  - =>prog(r).=d(l).s.was $\rightsquigarrow$
    =prog(r).y.be
    and =>y(r).=d(l).s.-ed

Head movement in the auxiliary system

- Note though that now we have two versions each of the present and past tense morphemes:

=>x(r).=d(l).s.-s	=>x(r).=d(l).s.-ed
=>y(r).=d(l).s.-s	=>y(r).=d(l).s.-ed

- There are three options:
  1. collapse x and y into a third category (perhaps v):

=>v(r).=d(l).s.-s	=>v(r).=d(l).s.-ed

  2. allow an **isa**-relationship to obtain between x and y (x is a y):

=>y(r).=d(l).s.-s	=>y(r).=d(l).s.-ed
=>x(r).y.∅	

  3. allow an **isa**-relationship to obtain between x and y (y is a x):

=>x(r).=d(l).s.-s	=>x(r).=d(l).s.-ed
=>y(r).x.∅	

Head movement in the auxiliary system

- Note that whenever *have* and *be* occur together, *have* always precedes *be*:
  - John has been dying
  - *John is having died
  - John will have been dying
  - *John will be having died
- and that, whenever *be* occurs incorporated into *-s* or *-ed*, *have* is not present:
  - John is dying
  - *John is having died
  - John was dying
  - *John was having died
- These facts argue against the first option (treating *have* and *be* as having the same category)

Head movement in the auxiliary system

- We have the same difficulty with the perfective *-en*!

=>v(r).perf.-en	=>y(r).perf.-en

- There are again three options:
  1. collapse v and y together:

=>v(r).perf.-en

  2. allow an **isa**-relationship to obtain between v and y (v is a y):

=>v(r).perf.-en =>v(r).y.∅

  3. allow an **isa**-relationship to obtain between v and y (y is a v):

=>y(r).perf.-en =>y(r).v.∅

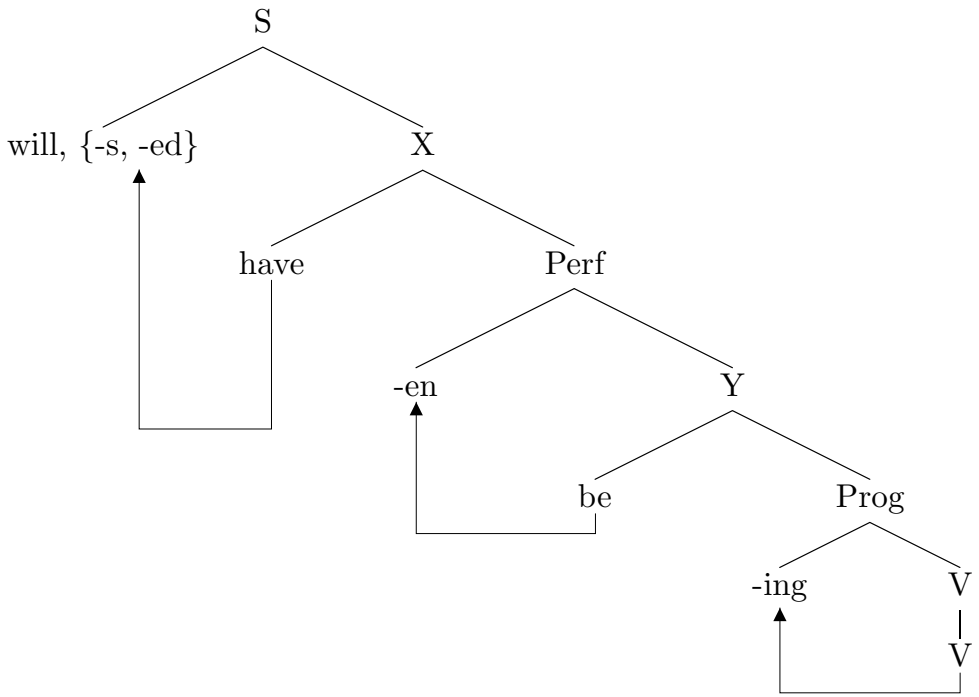
Head movement in the auxiliary system

- Note that whenever *be* and *die* occur together, *be* always precedes *die*:
  - John has been dying
  - *John has died be
  - John will have been dying
  - *John will have died be
- and that, whenever *die* occurs incorporated into *-en*, *be* is not present:
  - John has died
- The first option again is seen to be incorrect
- Note that if we assume that `v` **isa** `y`, and that `y` **isa** `x`, then we predict that *die* can incorporate into *-s* and *-ed*!
  - John dies
  - John died

Head movement in the auxiliary system

- Following similar reasoning, we arrive at the lexicon below:

=x(r).=d(l).s.will	=perf(r).x.have	=prog(r).y.be	v.die
=>x(r).=d(l).s.-s	=>y(r).perf.-en	=>v(r).prog.-ing	
=>x(r).=d(l).s.-ed	=>y(r).x.∅	=>v(r).y.∅	



- To add a new verb, we add just a single lexical item: v.laugh

Raising to Subject

- Verbs like *seem* allow for the following alternation:
  1. It seems that John died
  2. John seems to have died
- Observations:
  1. *it* as main clause subject requires finite *that*-complement
  2. DP as main clause subject forbids finite *that*-complement
- New lexical items:

$$\texttt{=x(r).i.to} \qquad \texttt{=i(r).v.seem}_1$$
$$\texttt{=s(r).c.that} \qquad \texttt{=c(r).v.seem}_2$$

- but how to enforce the dependency between the choice of lexical item (*seem*$_1$ vs *seem*$_2$) and the choice of subject type?
  Possibilities:
  1. Semantic type; $[\![it]\!] = \texttt{id}$, and $[\![seem_1]\!](P)(i) = [\![seem_2]\!](P(i))$
  2. Syntactic feature percolation

$$\texttt{=c(r).v}^{\texttt{expl}}\texttt{.seem}_2 \qquad \texttt{=x}^{\texttt{expl}}\texttt{(r).=d(l).s.will}$$

Raising to Subject

- The strategy pursued in minimalism (all the way back to generative semantics) is to satisfy dependencies ASAP, and then to transformationally re-arrange structures, as appropriate
- As *seem* doesn't semantically select for a subject, we assign it the following category:

  =i(r).v.seem

- The embedded predicate, *die*, on the other hand, *does* select a semantic argument:

  =d(r).v.die

- But now we have a problem: subjects surface in front of the auxiliaries, not after them! Accordingly, we assign to DPs the type:

  d.-k

  this encodes that the base position of the DP (d) is different from its surface position (-k).

Raising to Subject

- The =d(l) feature on the lexical items *will*, *-s*, and *-ed* were originally intended to introduce the predicate's argument in its surface position. Now the argument is already present, but not in its surface position.
- We thus assign the tense lexical items the type:

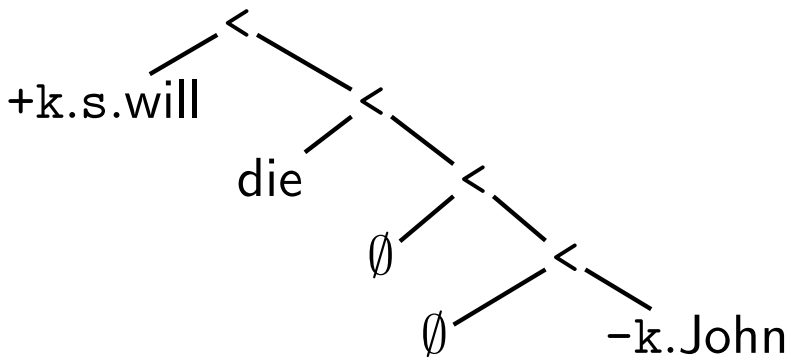
$$=\mathtt{x}(\mathsf{r}).+\underline{\mathtt{k}}.\mathtt{s}$$

  This indicates that a lexical item like *will* provides a *surface* position (for something with a -k feature, like a DP)
- Crucially, *to* doesn't provide a surface position:

$$=\mathtt{x}(\mathsf{r}).\mathtt{i}.\mathtt{to}$$

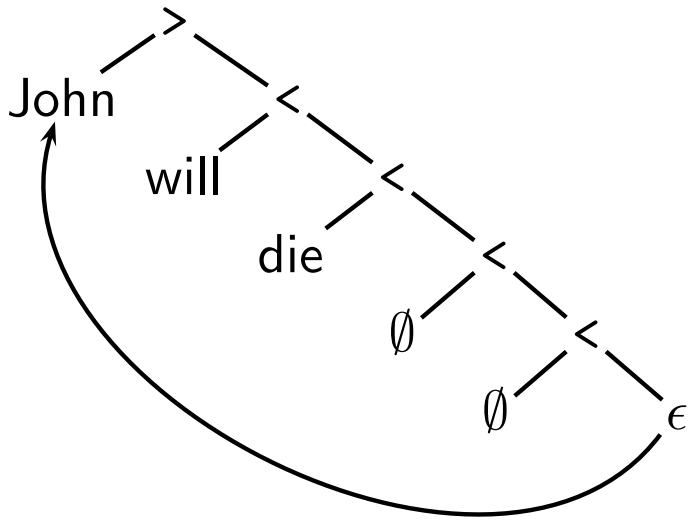
Raising to Subject

- Thus, surface subjects in simple intransitive sentences raise to this position from within the vP:



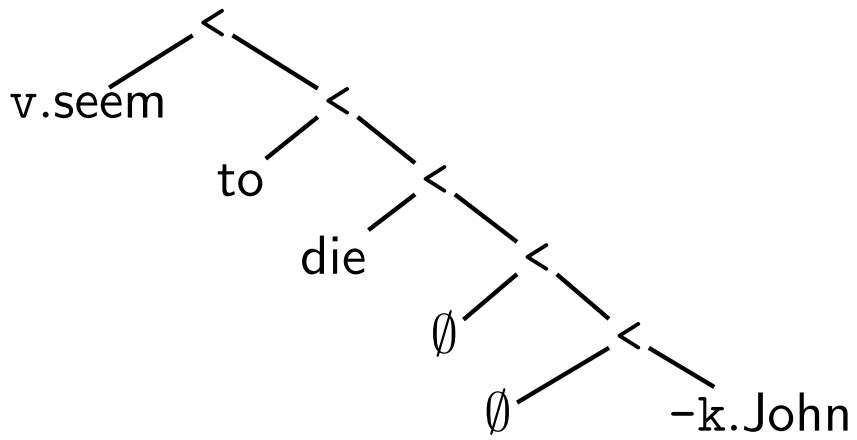
Raising to Subject

- Thus, surface subjects in simple intransitive sentences raise to this position from within the vP:



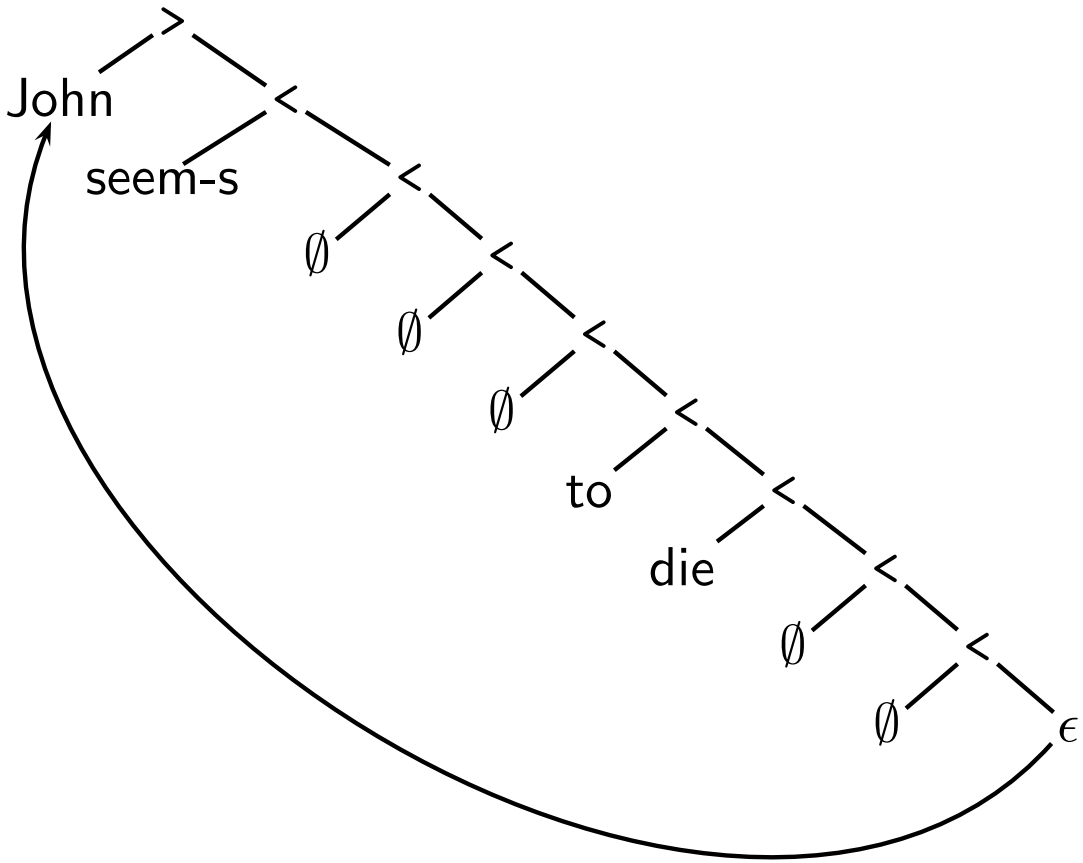
Raising to Subject

- The same is true of surface subjects of the matrix verb *seem*



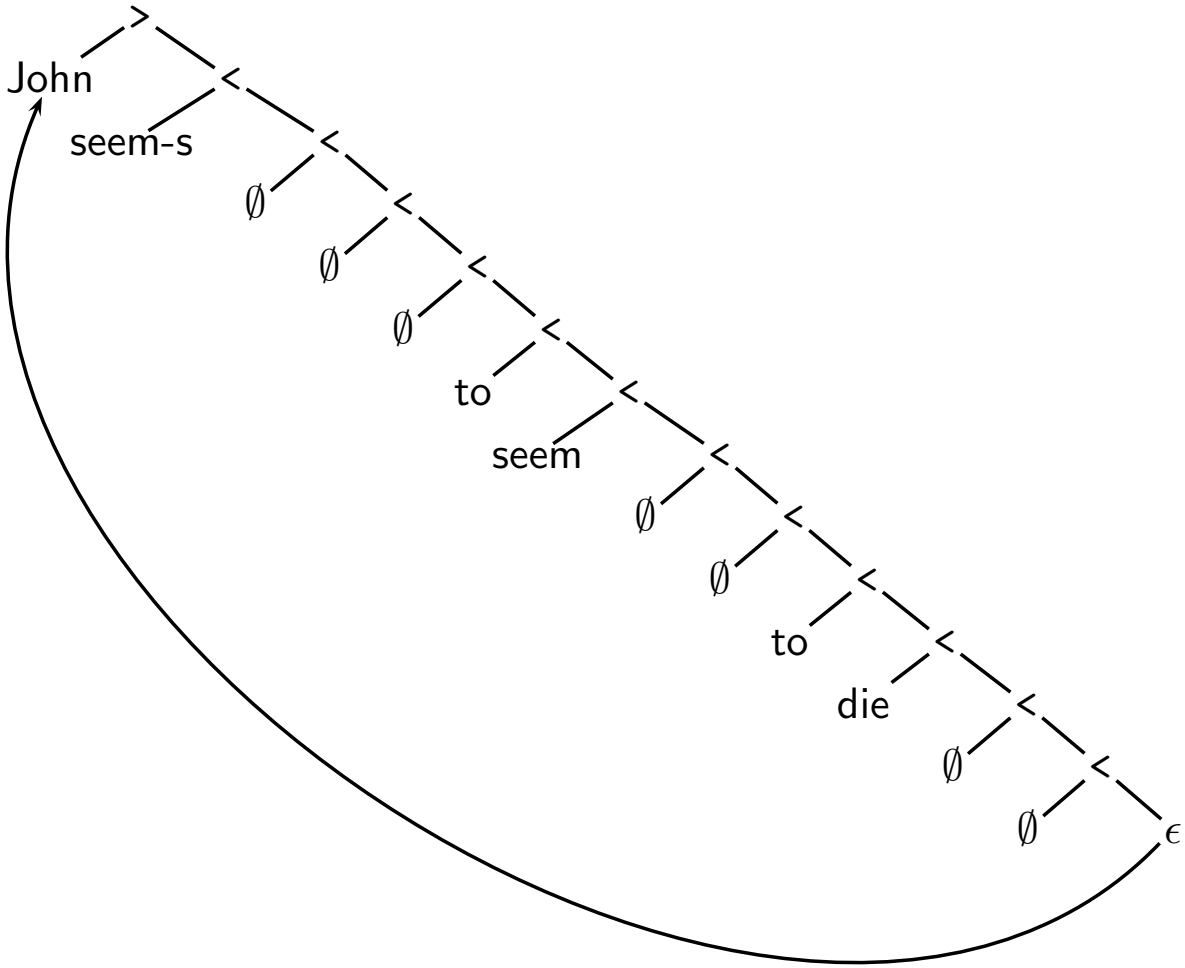
Raising to Subject

- The same is true of surface subjects of the matrix verb *seem*



Raising to Subject

- Note that we can add as many *seem to*'s as we want; only after we add a tense item do we trigger raising of the embedded DP:



Raising to Subject

- How do we deal with the alternation?
  1. It seems that John died
  2. John seems to have died
- We observe that *it* appears as the subject of tensed clauses without semantic subjects:
  - *it* seems. . .
  - *it* rains

  From the perspective of our analysis thus far, *it* appears whenever we have a +k feature with nothing available to check it.
- Thus, within the present framework, we need to assign *it* a complex feature sequence ending in -k.
- As *it* doesn't have the same distribution as a regular DP, we don't give it the same category:

expl.-k.it

Raising to Subject

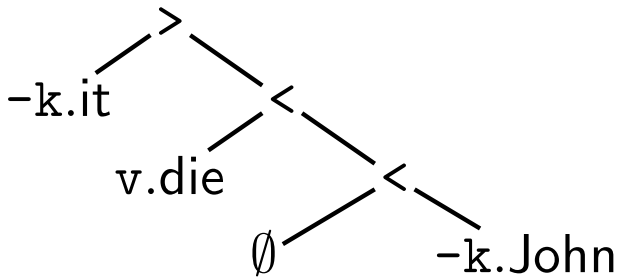
- We implement the 'last-resort' distribution of *it* (it appears only when nothing else can check the +k feature) as follows:
  1. treat it as a vP adjunct, allowing it to appear anywhere. Formally, a vP is something which can optionally select an `expl`:

     $$=>\texttt{v}(\texttt{r}).=\texttt{expl}(\texttt{l}).\texttt{v}.\emptyset$$

  2. This allows *it* to occur even when it isn't wanted!
  3. However, the SMC causes to crash any derivation in which *it* is merged while there is a DP still looking to check its `-k` feature.

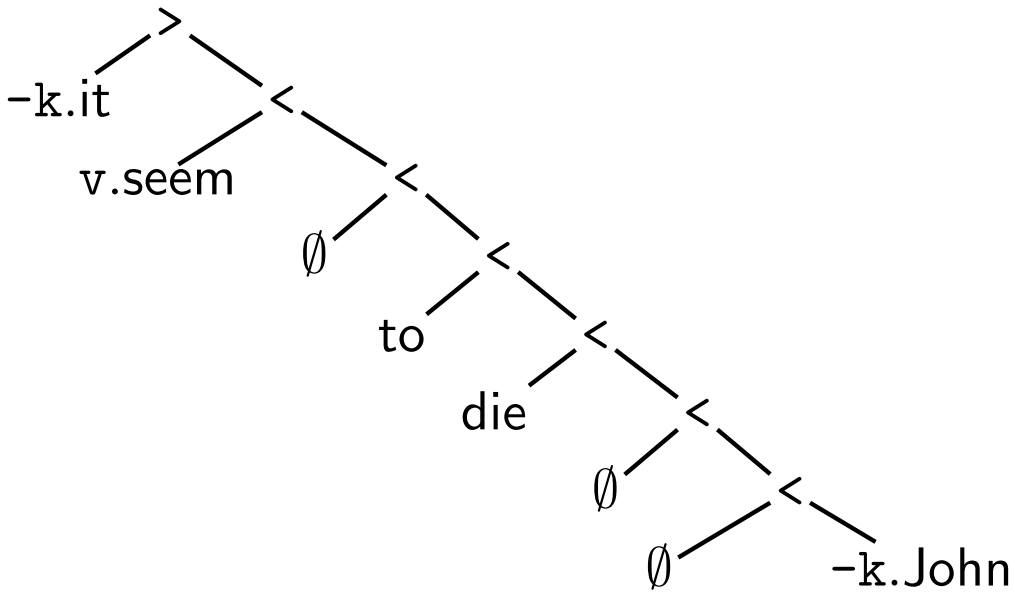
Raising to Subject

- This expression is generated by our grammar
- Note that it has two subtrees displaying -k! It can never become a complete expression of category s.



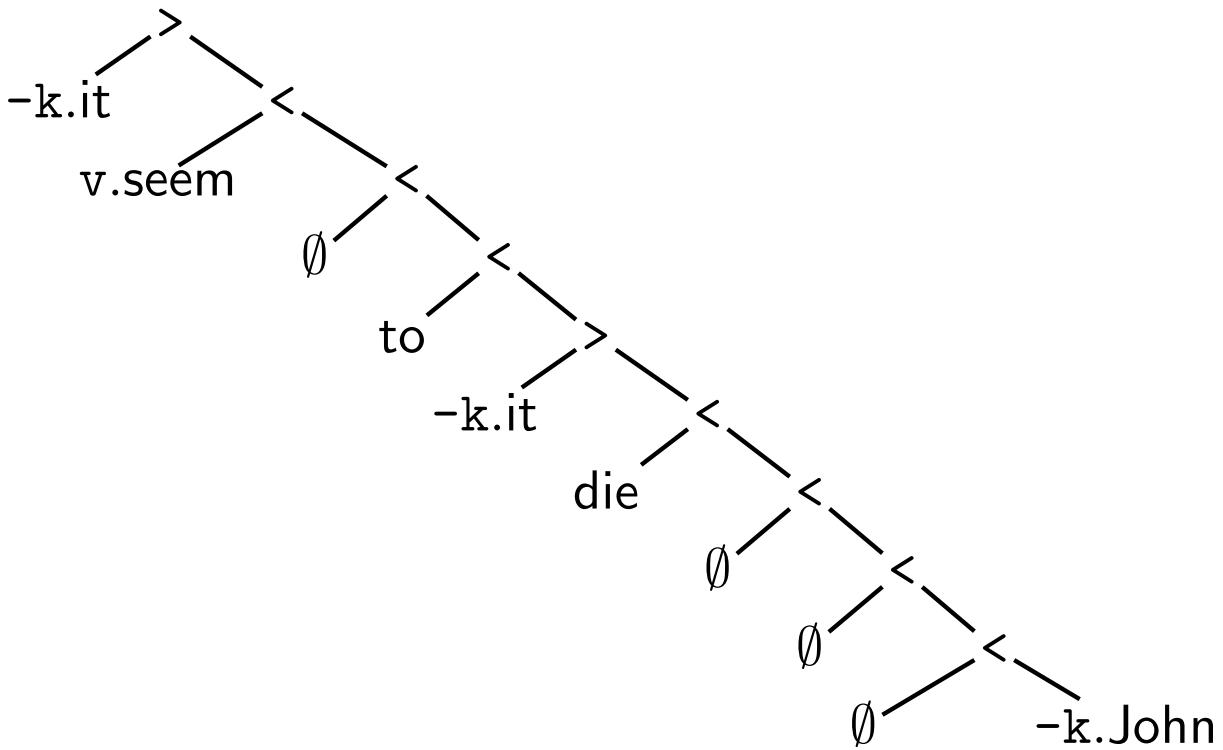
Raising to Subject

- This expression is generated by our grammar
- Note that it has two subtrees displaying -k! It can never become a complete expression of category s.



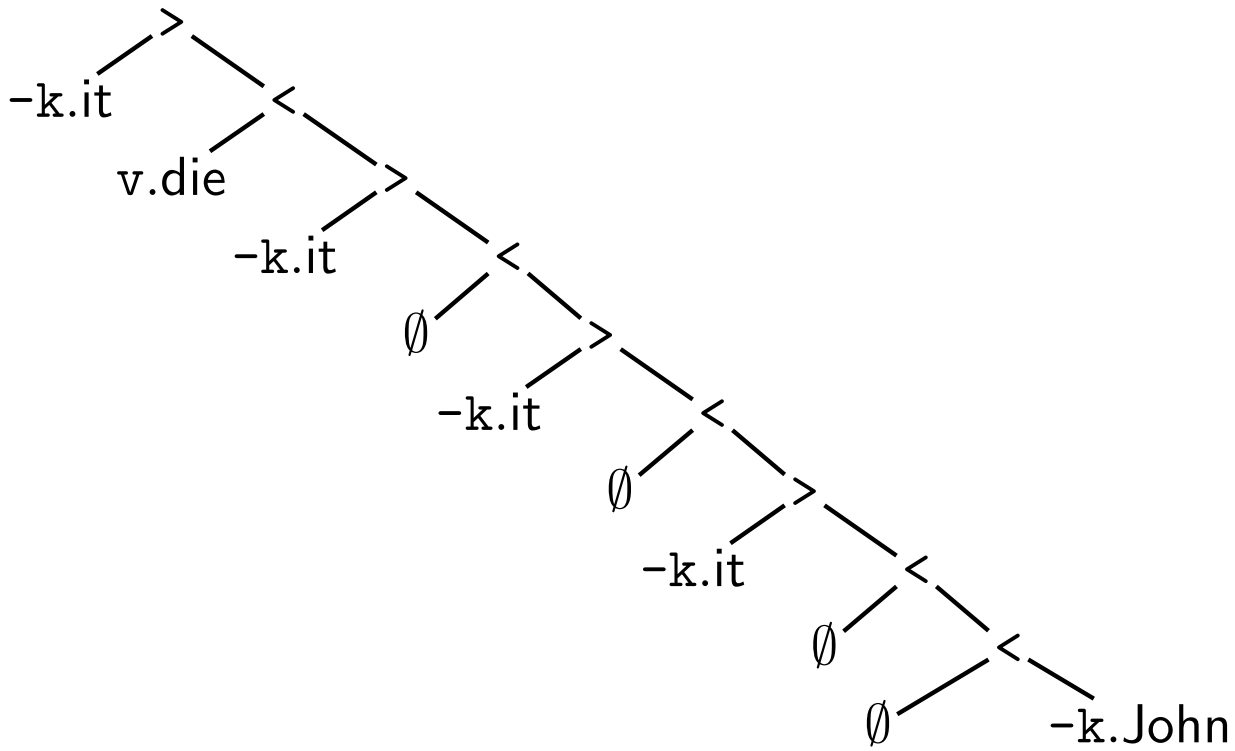
Raising to Subject

- Even crazier things are now in the closure of our lexicon under the generating functions.
- They are all blocked by the SMC from ever becoming complete expressions.



Raising to Subject

- Even crazier things are now in the closure of our lexicon under the generating functions.
- They are all blocked by the SMC from ever becoming complete expressions.



Raising to Subject

- We still have two lexical entries for *seem*:

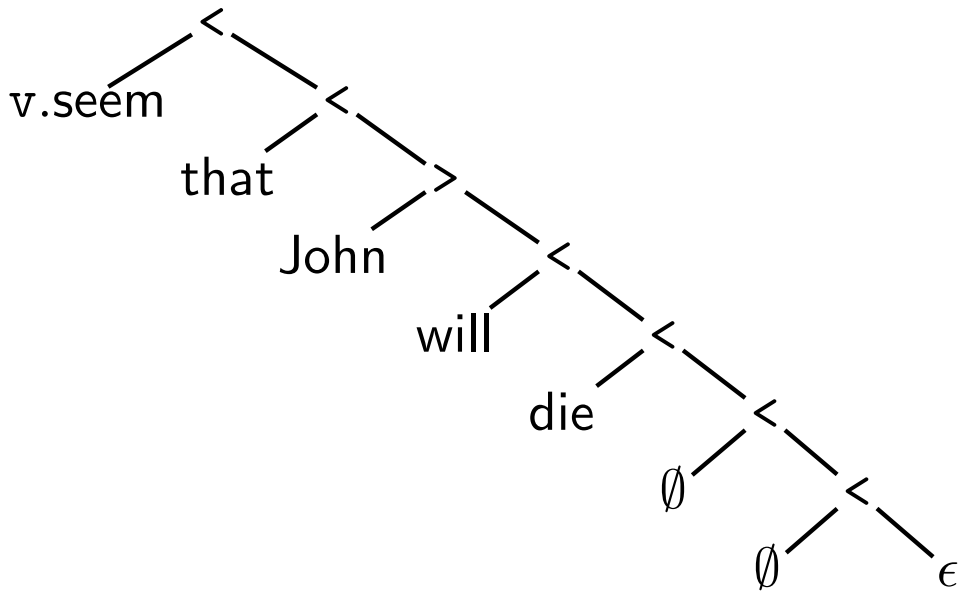
$$\texttt{=c(r).v.seem}_2 \quad \texttt{=i(r).v.seem}_1$$

- However, there is no point to the distinction between `i` and `c` in our grammar. We unify these categories throughout our lexicon:

=x(r).+k.s.will	=perf(r).x.have	=prog(r).y.be	=d(r).v.die
=>x(r).+k.s.-s	=>y(r).perf.-en	=>v(r).prog.-ing	d.-k.John
=>x(r).+k.s.-ed	=>y(r).x.$\emptyset$	=>v(r).y.$\emptyset$	
=s(r).c.that	=x(r).c.to	expl.-k.it	=c(r).v.seem

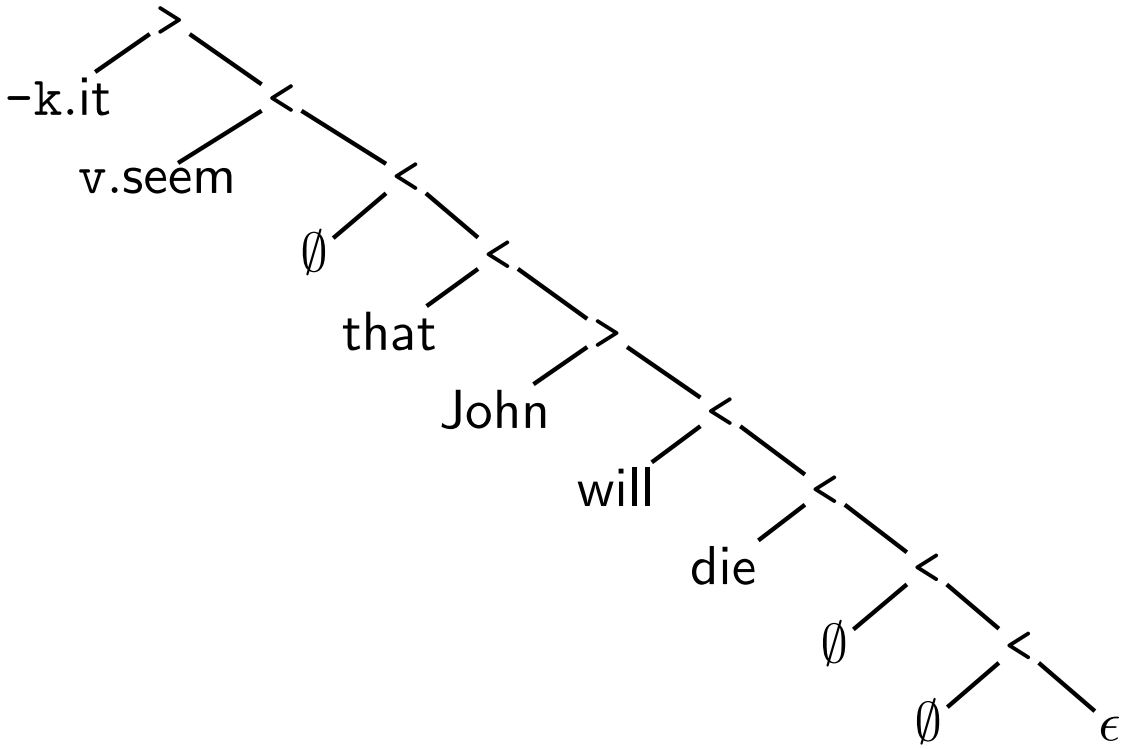
Raising to Subject

- We assign the *it*-sentence the following structure:



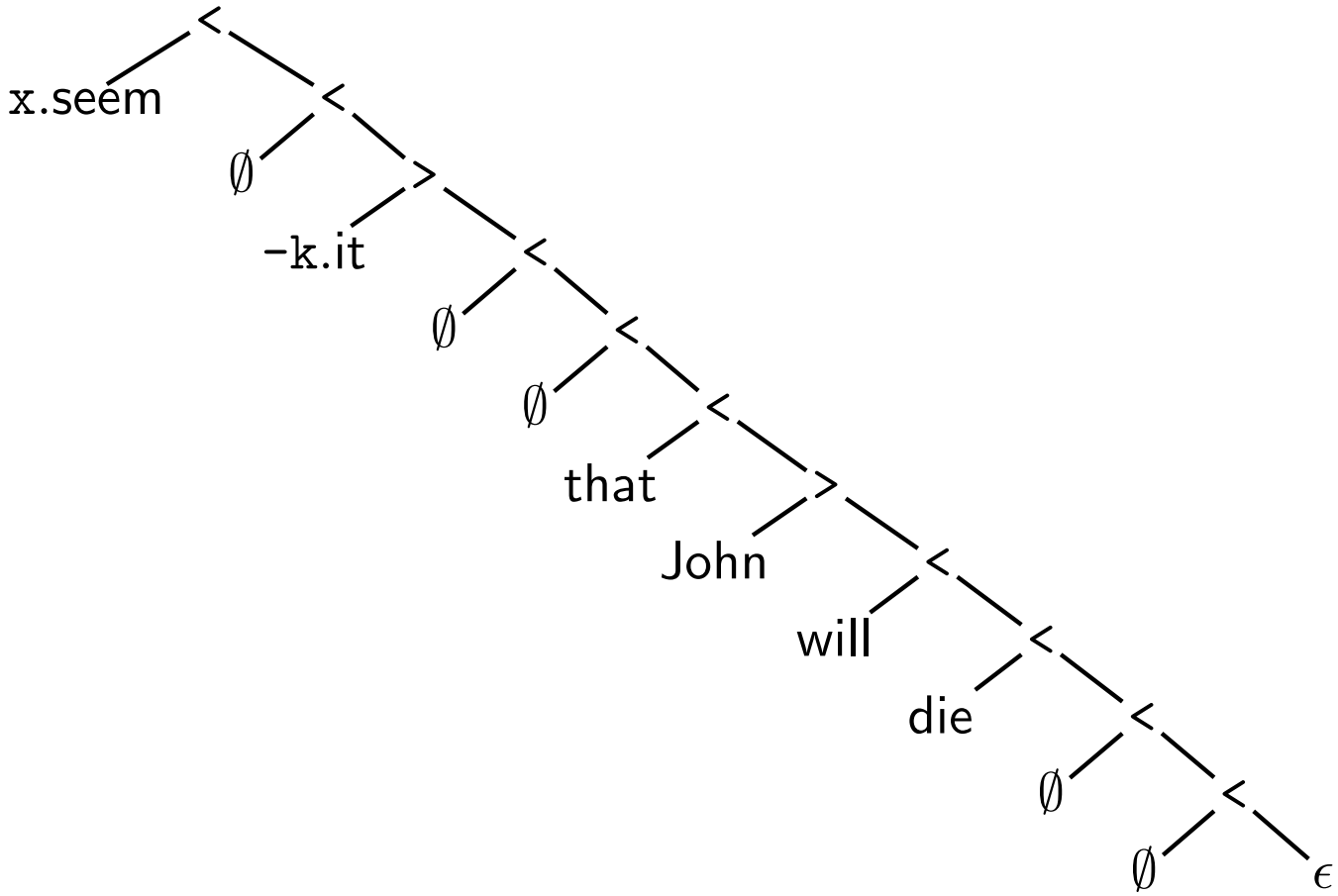
Raising to Subject

- We assign the *it*-sentence the following structure:



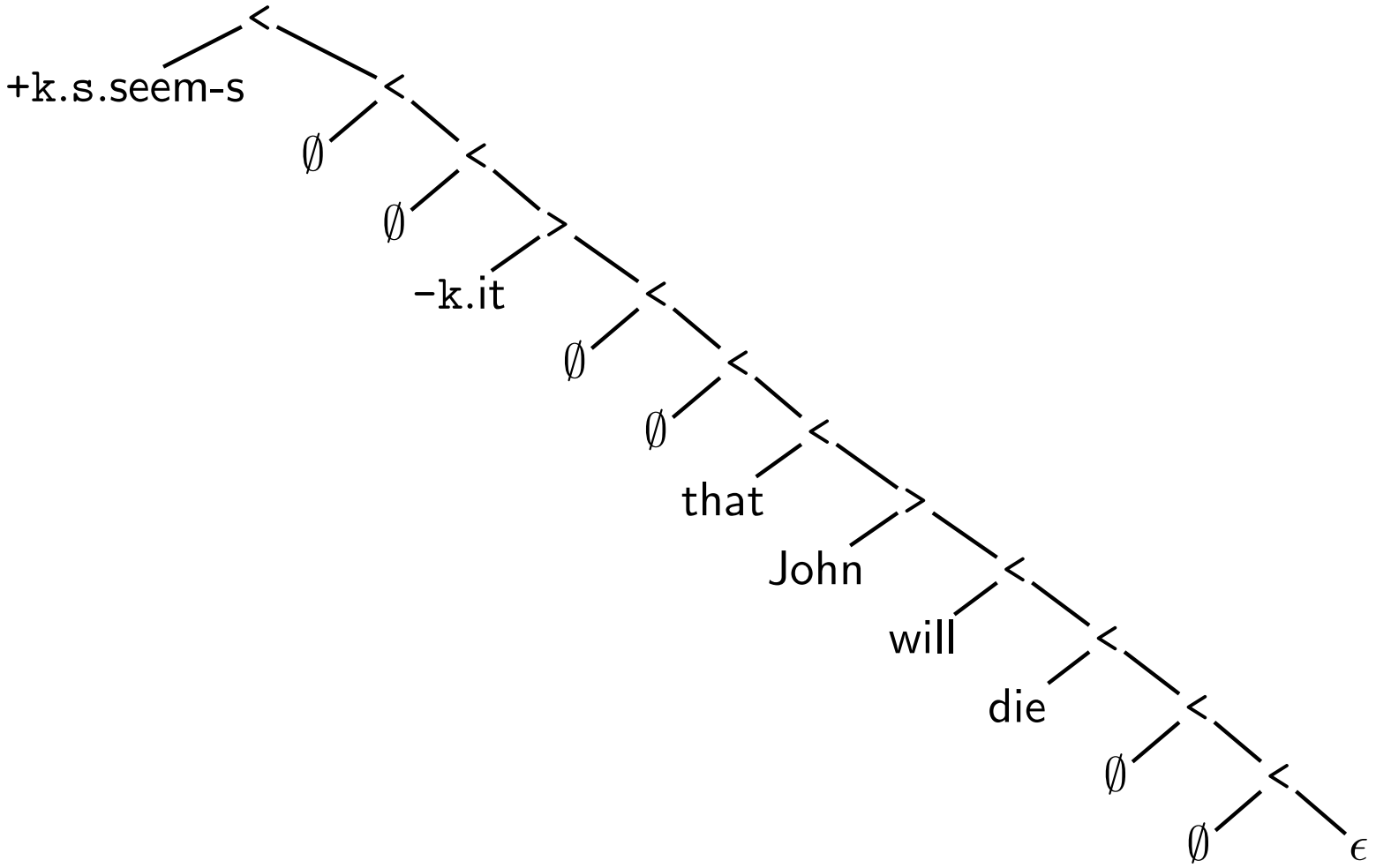
Raising to Subject

- We assign the *it*-sentence the following structure:



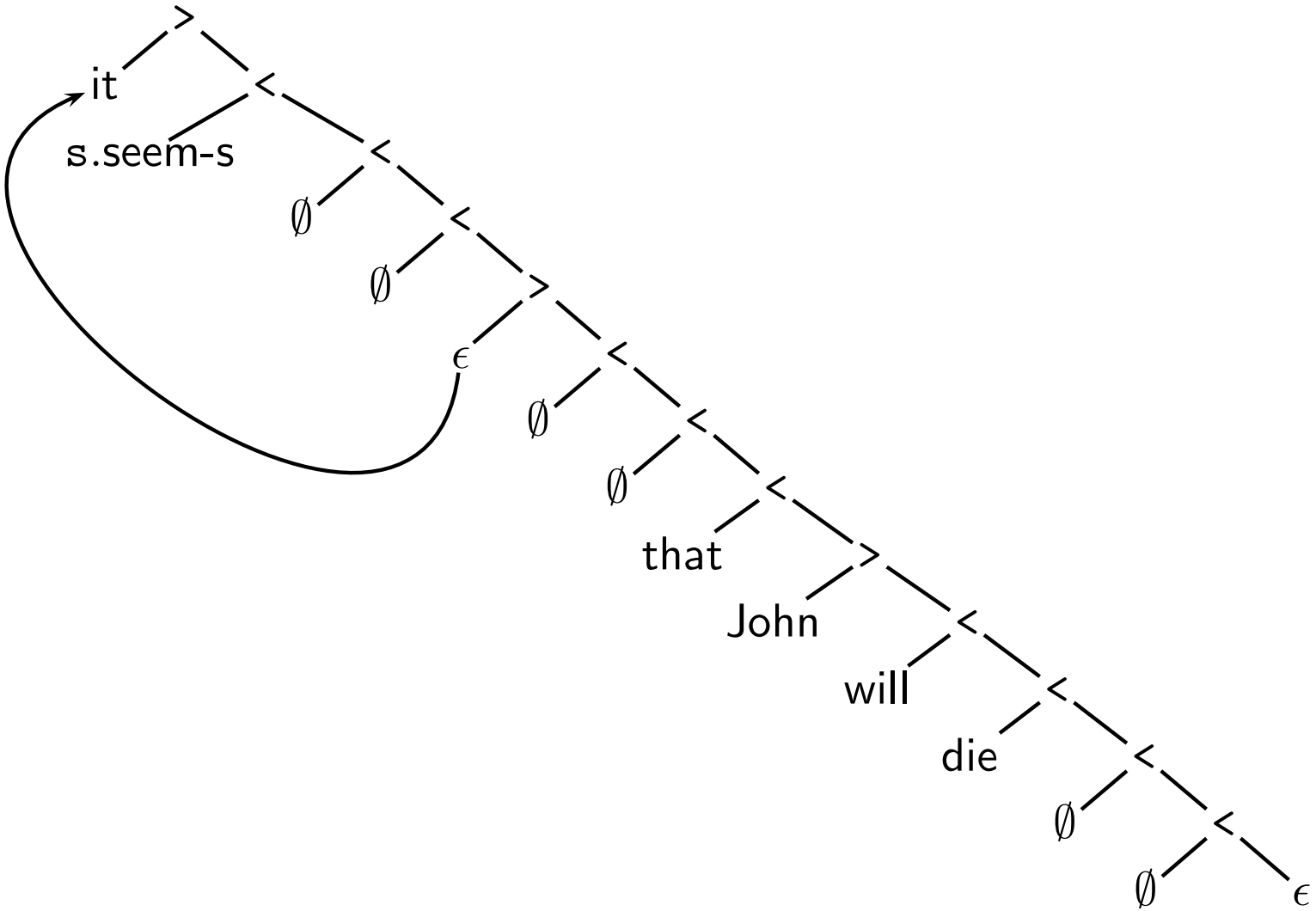
Raising to Subject

- We assign the *it*-sentence the following structure:



Raising to Subject

- We assign the *it*-sentence the following structure:



Raising to Subject

- Verbs like *rain*, or *snow* are represented as the below, allowing for *it*-insertion:

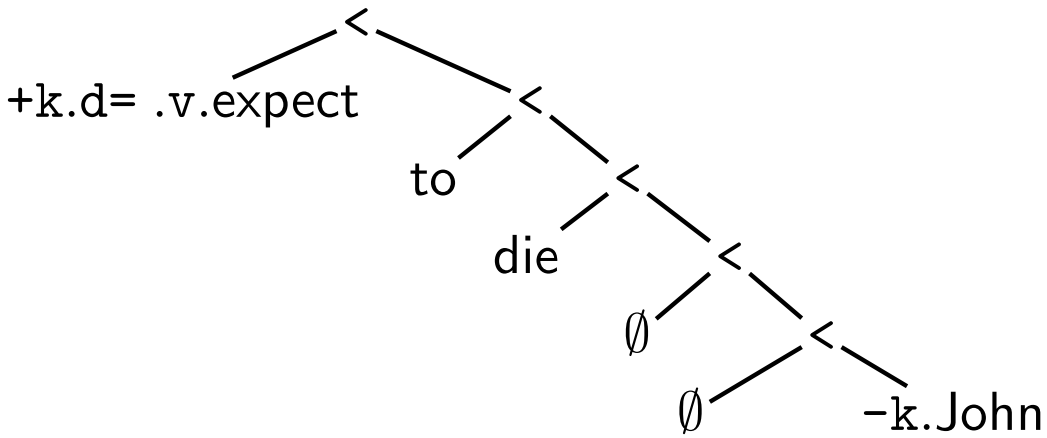
$$\text{v.rain}$$

- We can then derive the following sentences:
  1. It is raining.
  2. It seems to be raining.
  3. It seems that it is raining.

Raising to Object

- Raising to object, as in:
  1. Bill expects John to die.
  2. Bill expects that John will die.

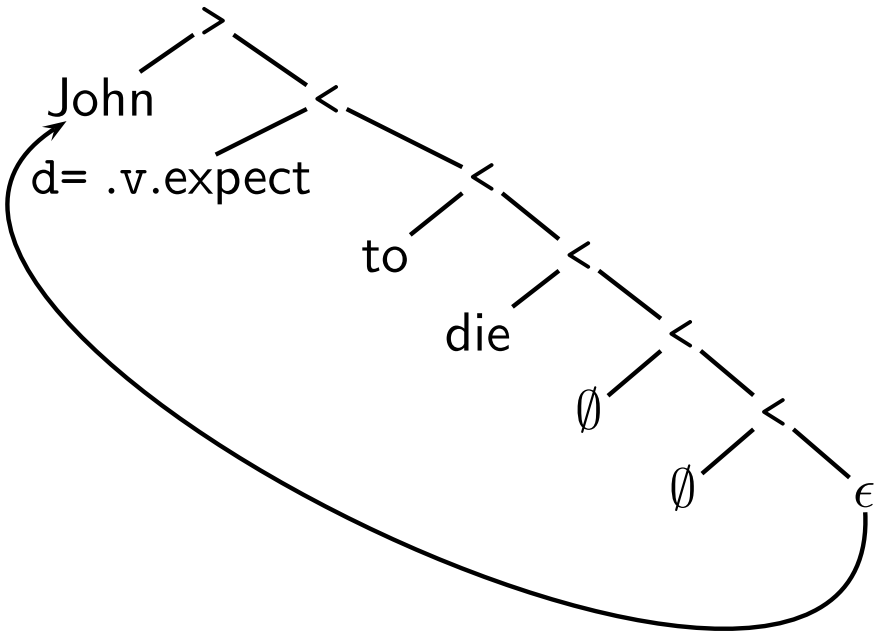
  can be accommodated by assigning *expect* the types below:
  1. =c(r).+k.=d(l).v.expect
  2. =c(r).=d(l).v.expect



Raising to Object

- Raising to object, as in:
  1. Bill expects John to die.
  2. Bill expects that John will die.

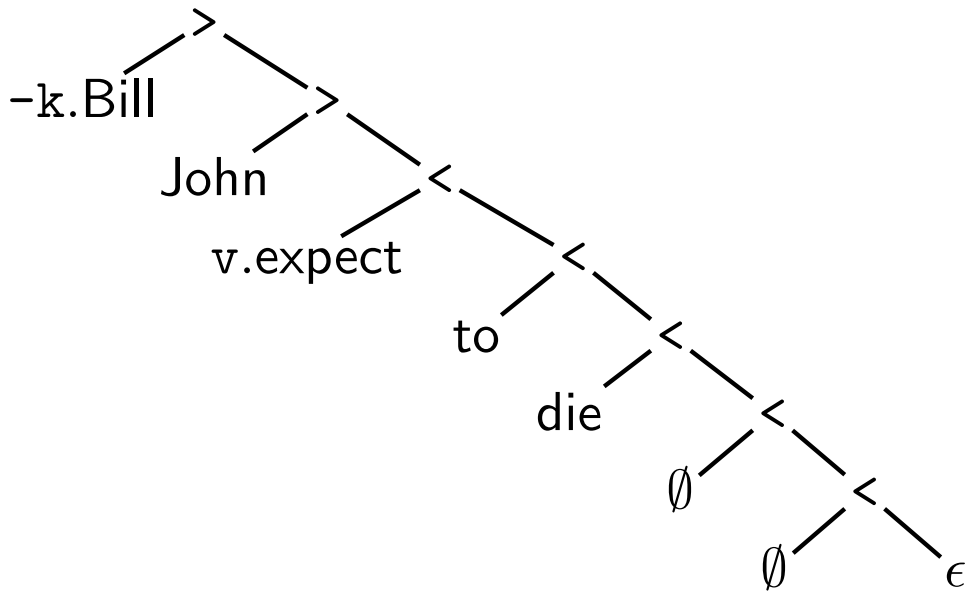
  can be accommodated by assigning *expect* the types below:
  1. =c(r).+k.=d(l).v.expect
  2. =c(r).=d(l).v.expect



Raising to Object

- Raising to object, as in:
  1. Bill expects John to die.
  2. Bill expects that John will die.

  can be accommodated by assigning *expect* the types below:
  1. =c(r).+k.=d(l).v.expect
  2. =c(r).=d(l).v.expect



Passive

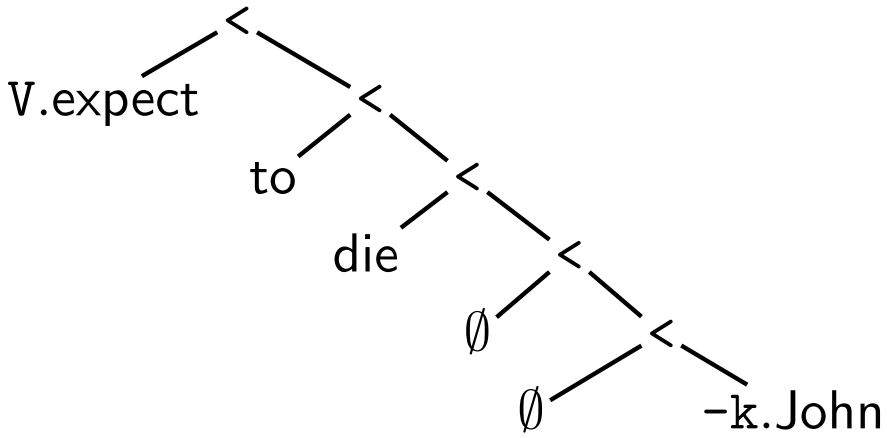
- Using the idea that DPs have distinct deep and surface positions lets us use our current technology to account for passivization:
  1. Bill expects John to die.
  2. John is expected to die.
  3. Bill expects that Mary will die.
  4. It is expected that Mary will die.
- In the first case, the +$\underline{k}$ of the surface position of the object and the =d(l) of the deep position of the subject are suppressed:

  =c(r).+$\underline{k}$.=d(l).v.expect =c(r).pass.expected

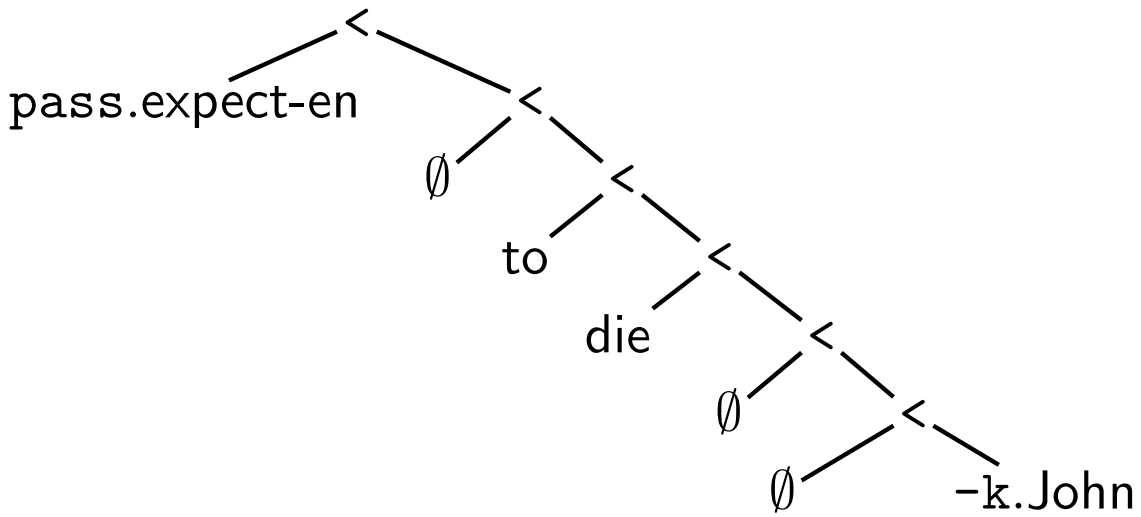
  =pass(r).v.be
- We again see regularities lurking beneath the surface:

=c(r).pass.expected	$\rightsquigarrow$	=c(r).V.expect,	=>V(r).pass.-en
=c(r).+$\underline{k}$.=d(l).v.expect	$\rightsquigarrow$	=v(r).V.expect,	=>V(r).+$\underline{k}$.=d(l).$\emptyset$

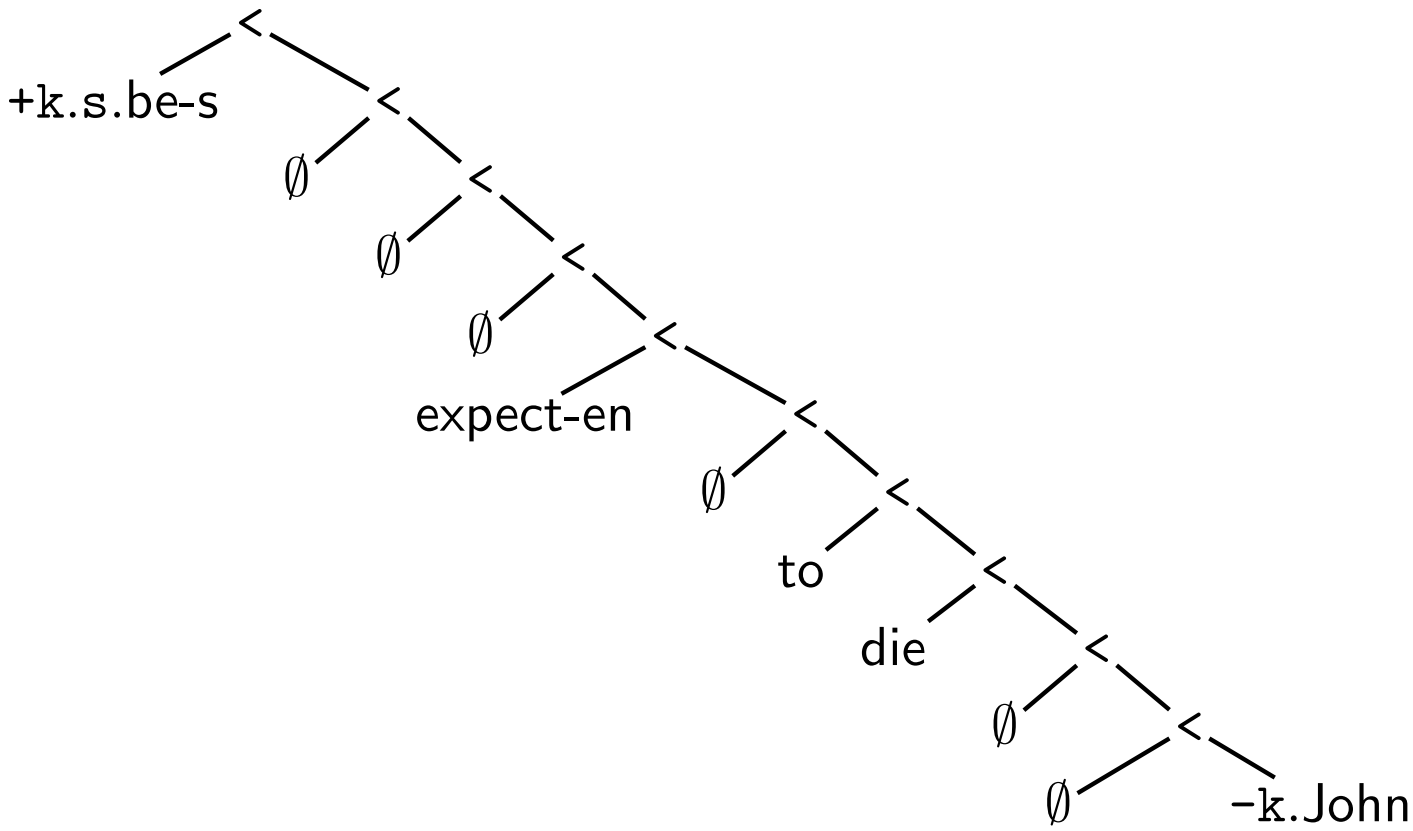
Passive



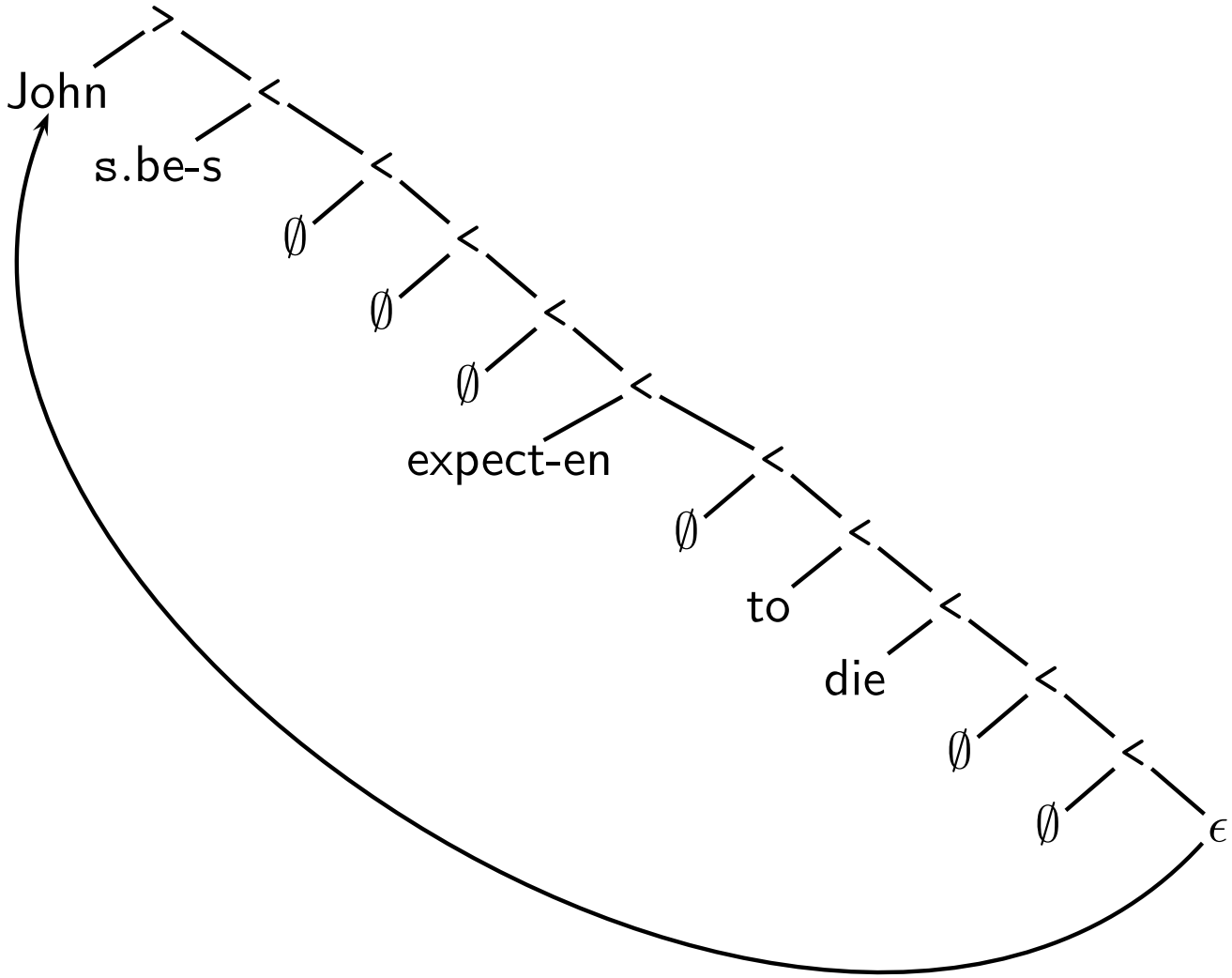
Passive



Passive

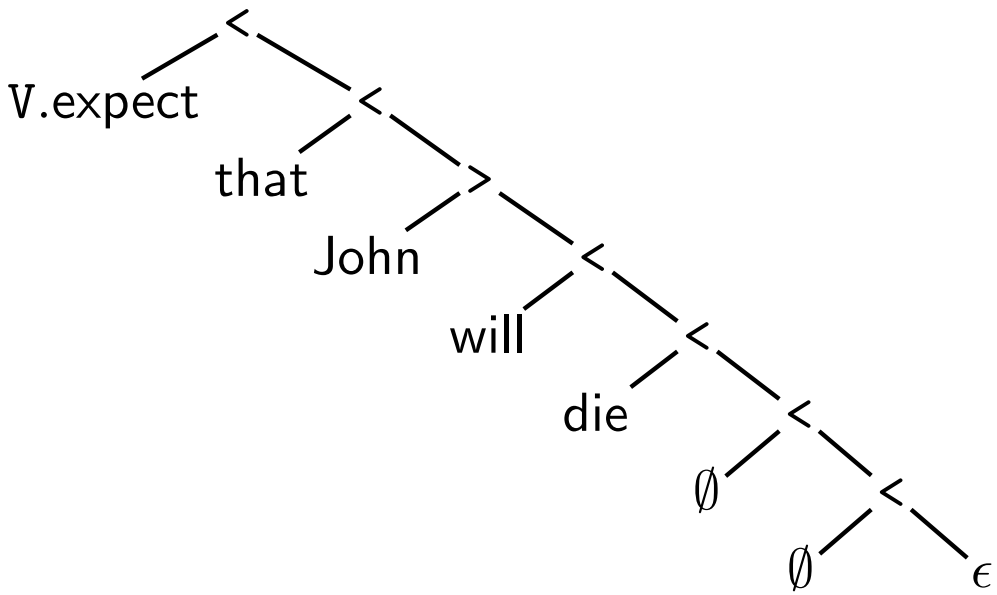


Passive



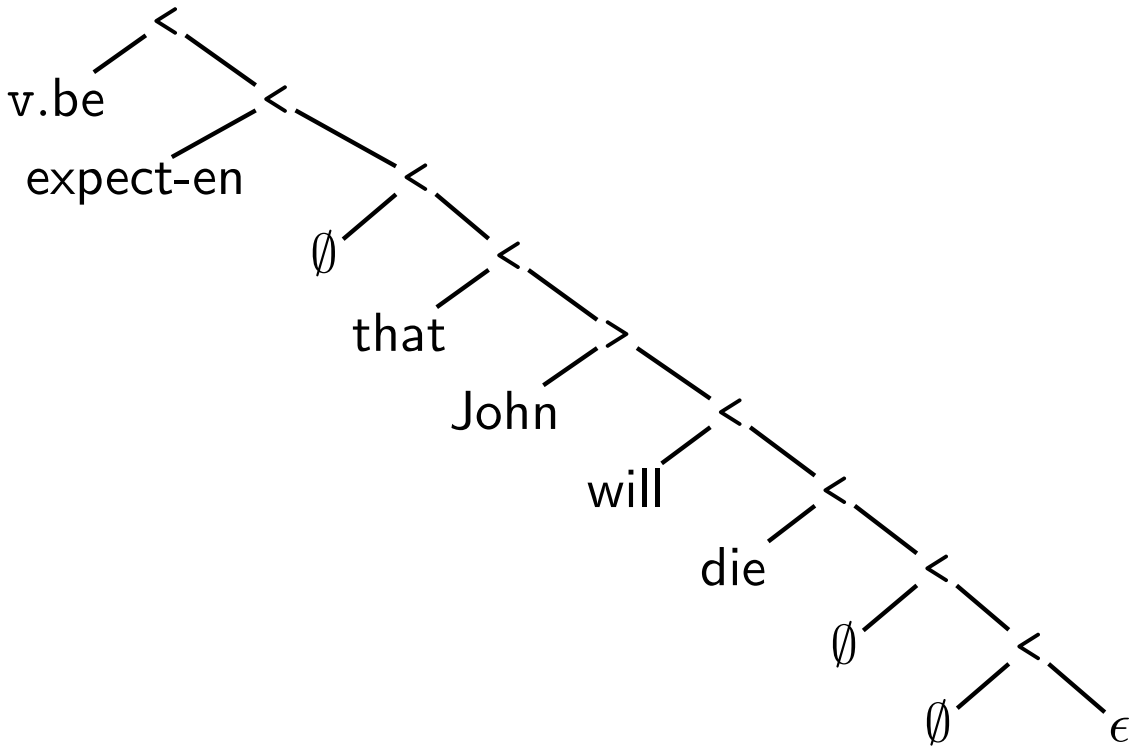
Passive

- With these lexical entries, we already derive both passive forms:
  1. John is expected to die
  2. It is expected that John will die



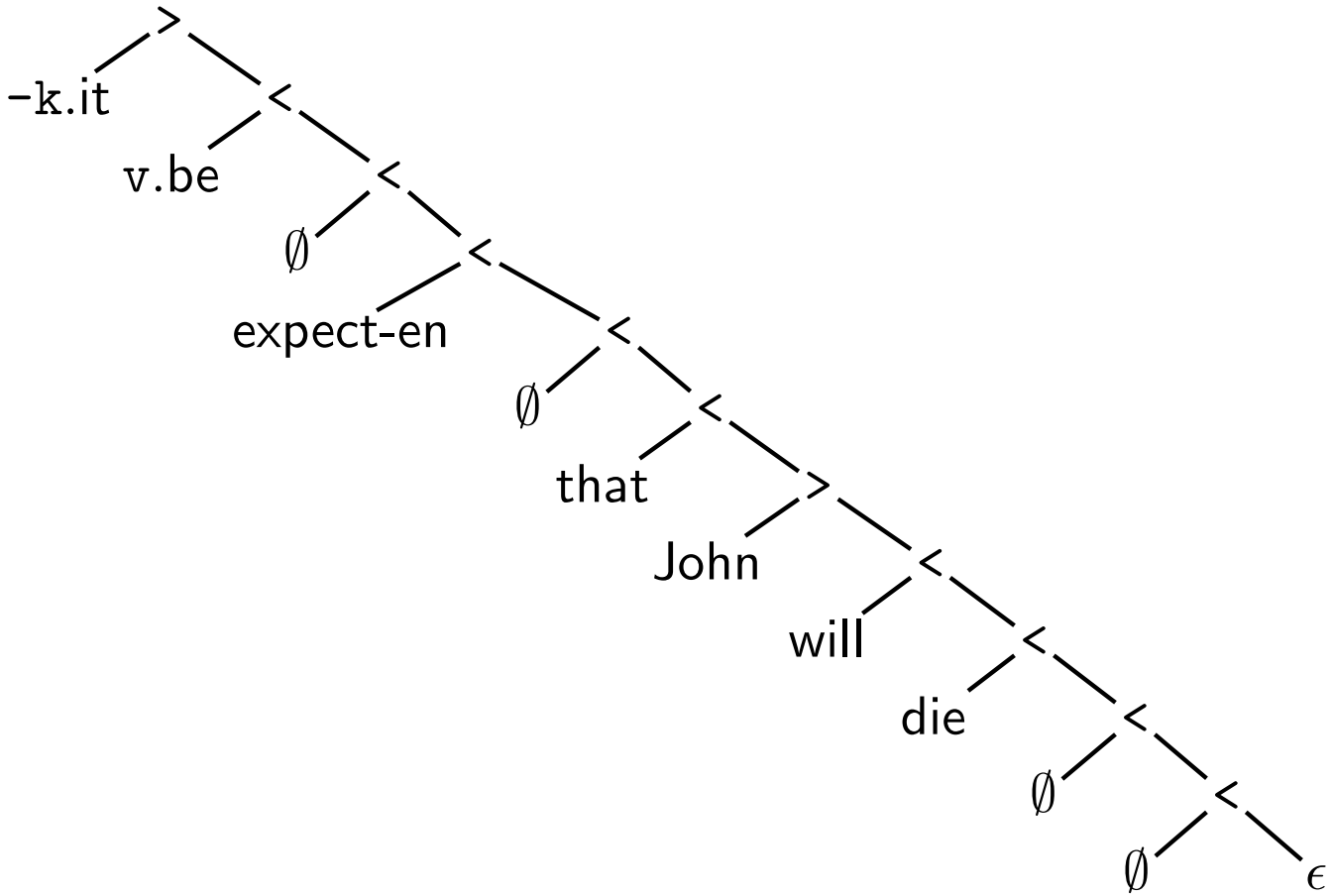
Passive

- With these lexical entries, we already derive both passive forms:
  1. John is expected to die
  2. It is expected that John will die



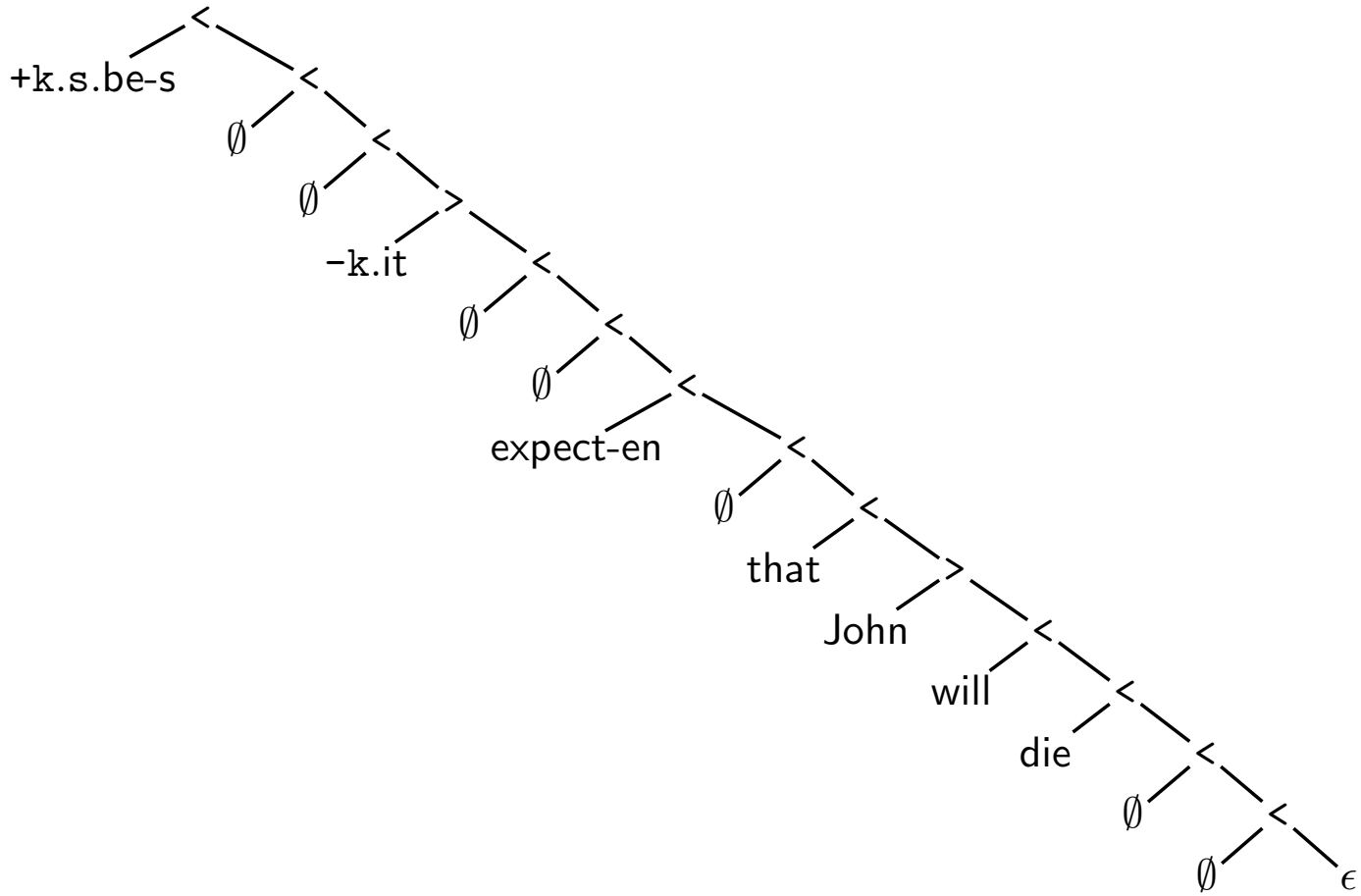
Passive

- With these lexical entries, we already derive both passive forms:
  1. John is expected to die
  2. It is expected that John will die



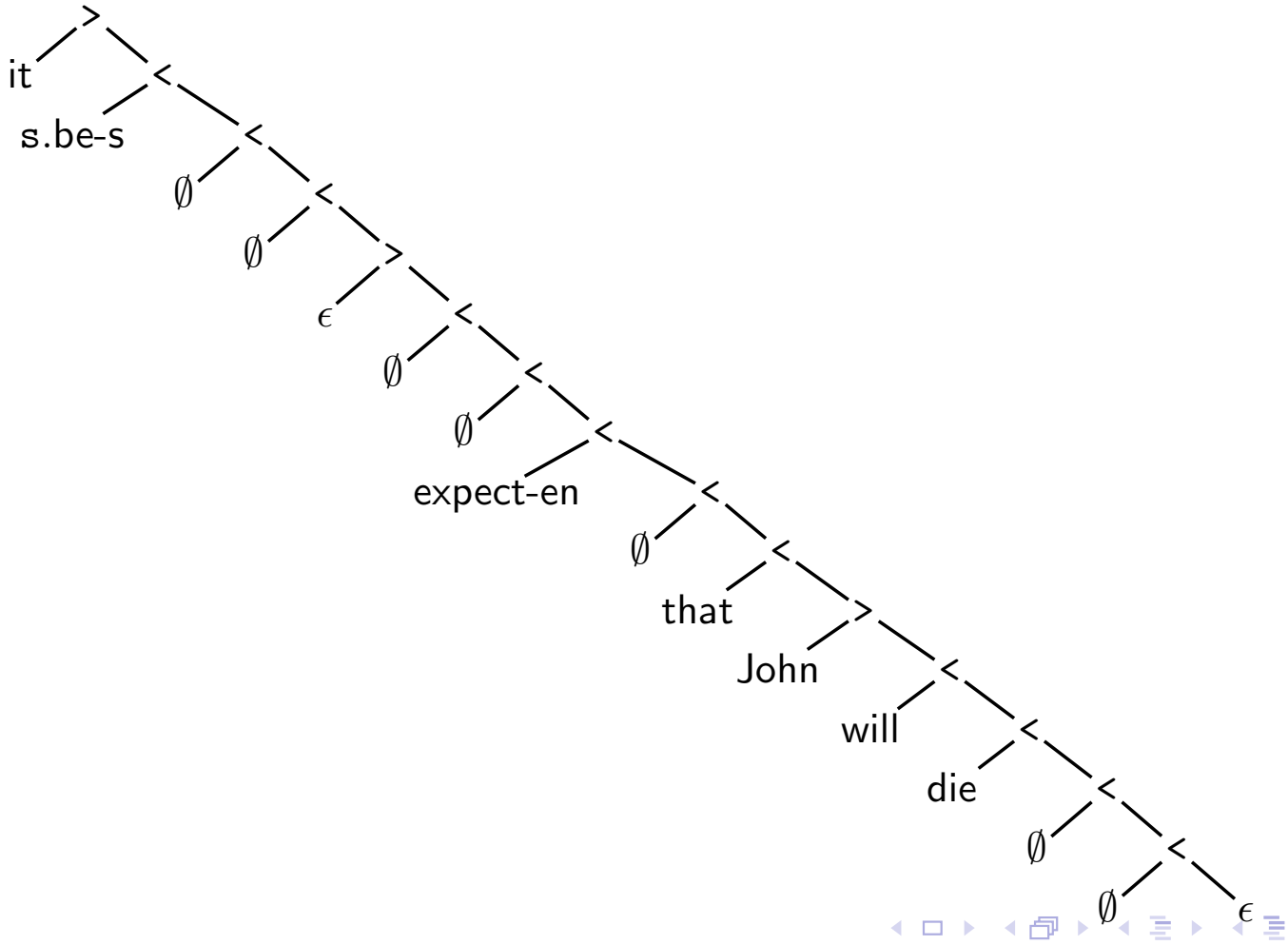
Passive

- With these lexical entries, we already derive both passive forms:
  1. John is expected to die
  2. It is expected that John will die



Passive

- With these lexical entries, we already derive both passive forms:
  1. John is expected to die
  2. It is expected that John will die



Raising to Object

- What can we say about the two lexical entries for *expect*?
  1. =c(r).V.expect
  2. =c(r).=d(l).v.expect
- The latter we can decompose into

$$\text{=c(r).V.expect} \qquad \text{=>V(r).=d(l).v.}\emptyset$$

- The element on the right looks similar to our 'active voice head': =>V(r).+$\underline{\text{k}}$.=d(l).v.$\emptyset$
- We decompose once more, extracting case assignment and external argument selection:

$$\text{=c(r).V.expect} \quad \text{=>V(r).+}\underline{\text{k}}\text{.agr0.}\emptyset \quad \text{=>agr0(r).=d(l).v.}\emptyset$$
$$\text{=>V(r).agr0.}\emptyset$$

Raising to Object

- Our lexicon looks as follows:

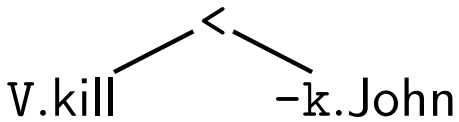
=x(r).+k.s.will =>x(r).+k.s.-s =>x(r).+k.s.-ed	=perf(r).x.have =>y(r).perf.-en =>y(r).x.∅	=prog(r).y.be =>v(r).prog.-ing =>v(r).y.∅	=d(r).v.die d.-k.John
=s(r).c.that	=x(r).c.to	expl.-k.it	=c(r).v.seem
=>agr0(r).=d(l).v.∅ =>V(r).pass.-en	=>V(r).+k.agr0.∅ =pass(r).v.be	=>V(r).agr0.∅	v.rain =c(r).V.expect

Transitivity

- A simple transitive verb looks as follows:

$$=\mathtt{d}(\mathtt{r}).\mathtt{V}.\mathrm{kill}$$

- The SMC ensures that, in the active voice, `agr0` must check the object's case

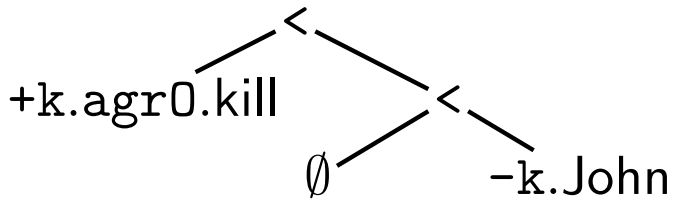


Transitivity

- A simple transitive verb looks as follows:

$$=\mathrm{d}(\mathrm{r}).\mathtt{V}.\mathrm{kill}$$

- The SMC ensures that, in the active voice, agr0 must check the object's case

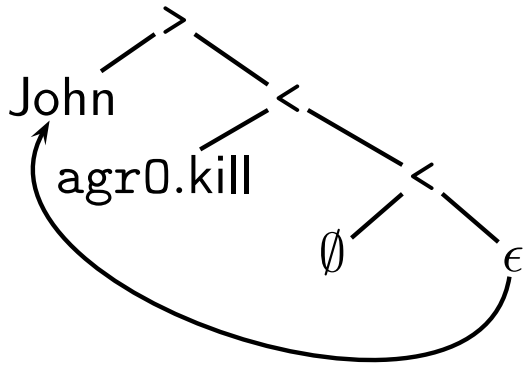


Transitivity

- A simple transitive verb looks as follows:

$$=\mathtt{d}(\mathsf{r}).\mathtt{V}.\mathsf{kill}$$

- The SMC ensures that, in the active voice, agr0 must check the object's case

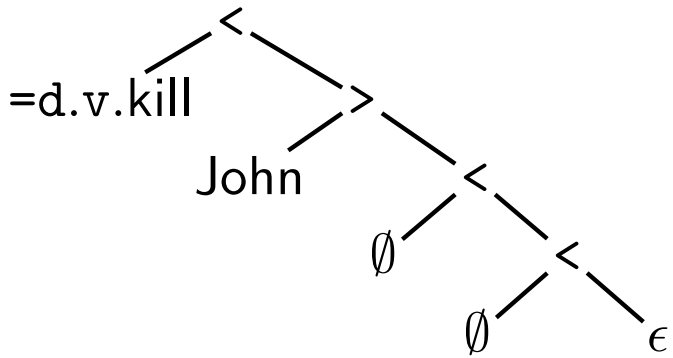


Transitivity

- A simple transitive verb looks as follows:

$$=\mathtt{d}(\mathtt{r}).\mathtt{V}.\mathtt{kill}$$

- The SMC ensures that, in the active voice, agr0 must check the object's case



Obligatory Raising to Object

- Some raising to object verbs do not allow for *that*-complements
  1. Bill caused John to die.
  2. *Bill caused that John died.
- In order to describe verbs like these, we need to reimplement a distinction between finite and non-finite complements (`c` and `i`)

$$\texttt{=i(r).V.cause} \qquad \texttt{=x(r).i.to}$$

- However, in order to continue to be able to describe the distribution of *seem* with a single lexical item, we want to say that there is a relation between `i` and `c`; namely, that `i` **isa** `c`:

$$\texttt{=>i(r).c.}\emptyset$$

Obligatorily Passive

- Some verbs *only* appear in the passive:
  1. It is rumored that John died.
  2. John is rumored to have died.
  3. *Bill rumors that John died.
  4. *Bill rumors John to have died.
- If we view this as a grammatical fact (as opposed to one of frequency), we can assign such verbs the following type:

=c(r).pass.rumored

What is not under the sun

- What this analysis doesn't really allow to be stated elegantly:
  - a sentential complement taking verb which
    1. is passivizable
    2. takes *only* a *that*-complement in the active
    3. but takes either a *that*-complement or a *to*-complement in the passive
- Like *think*?
  1. Bill thinks that John is dying
  2. *Bill thinks John to be dying.
  3. It is thought that John is dying.
  4. John is thought to be dying.
- But:
  1. "you might think him to be a reasonable person" (``think him to'', 29k Google hits)
  2. "she's not the person they think her to be." (``think her to'', 84k Google hits)
  3. "I am not as powerful as you think me to be." (``think me to'', 266k Google hits)