### Movement

1. a. I can solve this problem.
   
   b. This problem, I can solve.

2. **solve**: [— NP]

3. **Topicalization**
   Move XP and attach it as the leftmost constituent of S.

   
   ![Diagram of movement](image)

4. a. I can solve this problem.
   
   b. This problem, I can solve.

5. ![Diagram of topicalization](image)

6. ![Diagram of trace](image)

7. a. [PP to John], Mary gave the book \( t_{[PP]} \).
   
   b. [VP fix the car], I wonder whether she will \( t_{[VP]} \).

8. a. \([S \{NP this problem\}, I believe \{S that \{S I can solve \{NP I can solve \} \}] \].
   
   b. \([S I believe \{S that \{S \{NP this problem\} I can solve \{NP I can solve \} \}] \].

### Trace convention

Movement transformations leave a trace behind.

9. a. I want to read this novel.
   
   b. I wanna read this novel.

10. **wanna** contraction: **want** **to** → **wanna**.

11. a. This novel, I want to read.
   
   b. This novel, I want to be considered for a prize.

12. a. This novel, I wanna read.
   
   b. * This novel, I wanna be considered for a prize.

13. a. \([NP this novel], I want to read \( t_{[NP]} \).\]
   
   b. * \([NP this novel], I want \( t_{[NP]} \) to be considered for a prize.\]

### Some history…

14. Some student read every chapter.
   
   ‘There is a student \( x \) such that for each chapter \( x, x \ read y.\)’
   
   ‘For every chapter \( y, there is a student \( x \) such that \( x \ read y.\)’

15. a. Some student read every chapter.
   
   b. \([NP Every chapter], some student read \( t_{[NP]} \).\]

### The structure of the grammar:

\[
\text{DS} \leftarrow \text{phrase structure rules} \quad \leftarrow \text{movement rules} \\
\text{surface structure (abstract)} \rightarrow \text{SS} \leftarrow \text{more movement rules} \\
\text{“phonetic form”} \rightarrow \text{PF} \leftarrow \text{“logical form”} \text{ (meaning)}
\]
**Wh-movement**

<table>
<thead>
<tr>
<th>Q. Which problem did you solve?</th>
<th>A. I solved number 18.</th>
</tr>
</thead>
</table>

(17) solve [— NP].

(18) [which problem], did you solve \( t_i \)?

(19) I wonder [which problem will solve \( t_i \)].

(20) I wonder...

```
Comp

\[ S' \]

which

problem

NP

Aux

will

VP

N

Mary

V

solve

t[sp]
```

Comp = complementizer. E.g., *that*.

(21) I know \([s_t[\text{this problem}], I can solve t_c]\). |

(22) a. * I wonder [which problem], that Mary solved \( t_i \).

b. * I wonder that [which problem], Mary solved \( t_i \).

**Wh-movement**

Move \(wh\)-XP to Comp.

(23) a. I wonder \([s_t[\text{how difficult }], [s_t\text{ the problem was } t_c]\).

b. I wonder \([s_t[\text{ when }], [s_t\text{ Mary solved the problem } t_i]\).

c. I wonder \([s_t[\text{ to whom }], [s_t\text{ Mary gave the book } t_i]\).

d. I wonder \([s_t[\text{ who(m) }], [s_t\text{ Mary gave the book } t_i]]\).

**Conditions on wh-movement**

Not all S-bars are created equal:

(25) a. I wonder which problem Mary solved.

b. I wonder if Mary solved this problem.

c. * I believe which problem Mary solved.

d. * I believe if Mary solved this problem.

(26) a. * I wonder that Mary solved this problem.

b. I believe that Mary solved this problem.

(27) wonder \([— S':[+Q]]\)

believe \([— S':[-Q]]\)

(28) a. I know that Mary solved the problem.

b. I know if Mary solved the problem.

c. I know which problem Mary solved.

(29) know \([— S']\)

So, sentences are either [+Q] or [-Q], and wh-movement only works for [+Q] sentences.

**Wh-movement** (statement two)

Move \(wh\)-XP to Comp when Comp is [+Q].

Echo questions: actually [-Q]:

(30) a. Mary solved which problem?

b. You believe Mary solved which problem?

c. * You wonder Mary solved which problem?

**Questioning out of questions:** The wh-island condition

(31) a. How, do you think \([s_t[\text{(that) }]\), \([s_t\text{ Mary solved the problem } t_i]\)?

b. * How, do you wonder \([s_t[\text{ whether }], [s_t\text{ Mary solved the problem } t_i]\)?

(Ans.: I wonder whether Mary solved the problem *with her Palm Pilot.*)

b'. How, do you wonder \( t_i \), \([s_t[\text{ whether }], [s_t\text{ Mary solved the problem } t_i]\)?

(Ans.: I wonder *desperately* whether Mary solved the problem.)

(32) a. [Which way], do you think \([s_t[\text{(that) }]\), \([s_t\text{ John went } t_i\text{ (to fix his car)}]\)?

b. * [Which way], do you wonder \([s_t[\text{ why} ]], [s_t\text{ John went } t_i t_j]\)?

(Ans: I wonder why John went *north.*)
A *wh*-island is a [+Q] S′; an island to wh-movement (things inside are “stranded”).

**Wh-movement** (statement three)
Move wh-XP to Comp when:
(i) Comp is [+Q]
(ii) the wh-XP is not inside a [+Q] S′ (except the one containing the Comp).

**Complex Noun Phrase Condition (CNPC)**

Complex noun phrase: the claim that John went north, the hat I gave to John.

**Wh-movement** (statement four)
Move wh-XP to Comp when:
(i) Comp is [+Q]
(ii) the wh-XP is not inside a [+Q] S′ (except the one containing the Comp).
(iii) the wh-XP is not inside a complex NP.

**Successive-cyclicity**

**Wh-movement** (statement five)
Move wh-XP to Comp when:
(i) Comp is [+Q]
(ii) the wh-XP is not inside a [+Q] S′ (except the one containing the Comp).
(iii) the wh-XP is not inside a complex NP.
(iv) there is no nearer Comp.

**Relative clauses**

**Restrictive relatives** restrict the reference of the head noun. The head noun and the wh-phrase are co-indexed (meaning they share the same referent).

**Appositive relatives** don’t restrict the reference, but provide additional information.

**Free relatives** involve -ever and don’t modify a head noun.

The Transformational Cycle — each S′ is self-contained.
You apply all the rules inside S′ before moving on to the higher S′.

Another way to think about this: wh-movement can only go to the closest Comp. (A locality condition…)

(33) a. [Which way], do you think John went t_i ?
b. * [Which way], do you wonder [s_i if [s John went t_i ]] ?

(Ans: I wonder if John went *north.*)

(34) a. You believed [NP the claim that John went north].
b. * [Which way], did you believe [NP the claim that John went t_i ] ?

(35) a. Bill stole [NP the hat I gave to John].
b. * [To who(m)], did Bill steal [NP the hat I gave t_i ] ?

(36) a. [which problem], did John say [s_i Mary heard [s Robin solved t_i ]] ?
b. Which problem did John say Mary believed Robin heard Susan solved t_i ?

(37) a. [s_i John knows [s_i [which problem], [s_i Mary solved t_i ]] ].
b. [s_i [which problem], does [s_i John know [s_i s_i Mary solved t_i ]] ] ?

We see that there’s a “place to stop” below know:

(38) [s_i [which problem], does [s_i John know [s_i t_i s_i Mary solved t_i ]] ] ?

The Transformational Cycle — each S′ is self-contained.

Another way to think about this: wh-movement can only go to the closest Comp.

(A locality condition…)

(40) Bill heard [NP the speech [s_i which, [s_i Mary made t_i ]]].

(41) a. Mary, who you met yesterday, just bought a house.
b. Mary, [s_i who [s_i you met t_i yesterday ] ].

(42) a. I will buy [whatever you sell t_i ].
b. Whoever just arrived unplugged my lamp.

(43) a. Bill heard the speech [which Mary made].
b. Bill heard the speech [that Mary made].
c. Bill heard the speech [Mary made].

(44) made: [ — NP ] (in the sense of made a speech).
(45) Bill heard [\textit{NP the speech}, [\textit{s Op, [\textit{s Mary made \textit{ti} ]]}]].

(46) a. …the speech that Mary made. 
   b. …[\textit{NP the speech}, [\textit{s Op, that [\textit{s Mary made \textit{ti} ]]}]]
   c. Comp 
   \hspace{1cm} Op 
   \hspace{1cm} that

(47) a. * I know the way \textit{Op}, (that) John wonders [\textit{wh-island why Bill went \textit{ti}}].
   b. * I know the way \textit{Op}, that John made [\textit{CNP the claim that Bill went \textit{ti}}].

(48) a. * I know the way which, John wonders [\textit{wh-island why Bill went \textit{ti}}].
   b. * I know the way which, John made [\textit{CNP the claim that Bill went \textit{ti}}].

So: • If relative clauses can have a null \textit{wh}-phrase, why can’t \textit{wh}-questions?
   • Why can you have \textit{that} with \textit{Op} but not with \textit{which}?
   *I heard the speech, which, that Mary made \textit{ti} .

Recoverability Condition
The content of a null category must be recoverable (from a co-indexed overt category in the sentence).

(49) a. When did Mary buy the book? 
   b. Where did Mary buy the book? 
   c. How did Mary buy the book? 
   d. * \textit{Op}, did Mary buy the book?

(50) a. * \textit{Op}, did Mary buy \textit{ti}? 
   b. * \textit{Op}, did Mary give a book \textit{ti}?

(51) Bill heard the speech, [\textit{s Op, that [\textit{s Mary made \textit{ti} ]]}].

Doubly Filled Comp Filter
*\{Comp \textit{wh-XP that}, if \textit{wh-XP} is overt (non-null).

Filters. Exclude sentences which the PS rules and transformations otherwise generate.
(instead of encoding the restriction in the transformations).

NP-movement

Passive.

(52) a. Mary solved the problem.
   b. The problem was solved (by Mary).

(53) \textit{solve} [\textit{— NP}]

(54) S 
   \hspace{1cm} NP 
   \hspace{1cm} Aux 
   \hspace{1cm} VP 
   \hspace{1cm} tense 
   \hspace{1cm} V 
   \hspace{1cm} NP 
   \hspace{1cm} PP 
   \hspace{1cm} the problem 
   \hspace{1cm} was solved 
   \hspace{1cm} \textit{ti} 
   \hspace{1cm} by Mary

NP-movement
Move NP to an empty subject position

Raising constructions.

(55) a. It seems (that) Mary has solved the problem.
   b. Mary seems to have solved the problem.

(56) [\textit{s [\textit{NP e} ] seems [\textit{s [\textit{NP Mary } ] to have solved the problem} ]}].

(57) a. It appears (that) Mary has solved the problem.
   b. Mary appears to have solved the problem.
   c. It is likely that Mary will solve the problem.
   d. Mary is likely to solve the problem.

(58) a. * It is confident (that) Mary will win.
   b. * Mary is confident to win.
   c. Mary is confident (that) she will win.
Tensed S condition

(59) a. Mary seems [to have solved the problem].
   b. * Mary seems [has solved the problem].
   c. * Mary seems [will solve the problem].

Tensed clauses have [+Tense] in their Aux element,
Untensed clauses have [–Tense] in their Aux element,
pronounced as to (infinitive marker)

(60) [\[
\begin{array}{c}
\text{S} \\
\text{NP}_1 \\
\text{Aux} \\
\quad \text{VP} \\
\quad \quad \text{V} \\
\quad \quad \text{seems} \\
\quad \quad \text{NP} \\
\quad \quad \quad \text{Aux} \\
\quad \quad \quad \quad \text{VP} \\
\quad \quad \quad \quad \text{V} \\
\quad \quad \quad \quad \text{to} \\
\quad \quad \quad \quad \text{VP} \\
\quad \quad \quad \quad \text{have solved} \\
\quad \quad \quad \quad \text{the problem}
\end{array}\]
\]

NP-movement (second statement)
Move NP to an empty subject position,
unless NP is contained in a Tensed S (other than the target S).

Specified Subject Condition

(61) a. Mary appears \([t_1 \text{ to be likely } t_t \text{ to win}].\)
    b. * Mary appears \([t_t \text{ to be likely } t_1 \text{ to win}].\)

(62) a. John is believed \([t_t \text{ to have beaten Bill}].\)
    b. * Bill is believed \([\text{John to have beaten } t_t].\)

(63) a. \([\text{NP} \quad \text{is believed } \text{[S [NP John } \text{ to have beaten Bill]}.\]
   b. * \([\text{NP } \quad \text{is believed } \text{[S John to have beaten [NP Bill }]}.\]

NP-movement (third statement)
Move NP to an empty subject position, as long as
(i) NP is not contained in a Tensed S (other than the target S).
(ii) NP is not separated from the targeted position by a specified subject.

Extraposition and Heavy NP shift

(64) a. [Details of a secret plan to finance the rebels] have emerged.
    b. Details have emerged of a secret plan to finance the rebels.

(65) \([\text{NP } \quad \text{Details [NP } t_t ] \quad \text{have emerged [NP } \text{of a secret plan to finance the rebels}].\]

Extraposition (a rightward movement rule—unlike the rest we’ve seen)
Move XP and attach it as the rightmost constituent of VP.

(66) a. A man \(t_t\) has come forward \([t_t \text{ who claims to be the culprit}].\)

Heavy NP shift (a subcase of Extraposition)
Move a ‘heavy’ NP and attach it as the rightmost constituent of VP.

(67) a. Mary read \([\text{NP all the books she had borrowed}].\)
    b. ? Mary returned \([\text{NP all the books she had borrowed}] \text{ to the library}.\)
    c. Mary returned \(t_t \text{ to the library } \text{[NP all the books she had borrowed].}\)

Movement of terminal categories

(68) a. Mary read \text{Aspects}.
    b. Mary returned \text{Aspects} to the library.
    c. * Mary returned \(t_t \text{ to the library } \text{[NP Aspects]}.\)

Affix hopping.

(69) \text{Aux } \rightarrow \text{Tense (Modal) (Neg)}

(70) Mary solved the problem.
Mary

Tense

solve

the problem

VP

Aux

NP

S

(71) S

NP

Aux

VP

N

Mary

[+Past]

solve

the problem

VN P

Tense

(72) S

NP

Aux

VP

Mary

solve

Tense

-ed

the problem

VN P

Tense

(73) S

NP

Aux

VP

Mary

solve

Tense

-ed

the problem

VN P

Tense

(74) VP → (Adv) V … (Adv)

(75) a. John [VP cleverly avoided Bill].
  b. John [VP rarely visited Bill].
  c. * John avoided [VP cleverly Bill].
  d. * John visited [VP rarely Bill].

(76) John might cleverly avoid Bill.

(77) John did not avoid Bill.
    (78) John -ed not avoid Bill.

Affix Hopping (second statement)
Move Tense (from Aux) to V provided Aux does not dominate a Modal or Neg.

Do-support
Insert *do* to support stranded Tense.

(79) a. John should not avoid Bill.
    b. John cannot eat bacon.

Auxiliary verbs *have* and *be*:

(80) a. John is not happy.
    b. John has not visited Bill.
    c. * John did not be happy.
    d. * John did not have visited Bill.

To distinguish *have* and *be* from other verbs, we will say they are [+AUX],
whereas other verbs are [–AUX].

(81) a. John must be happy.
    b. John might have visited Bill.

V-raising
Raise V to Aux, provided:
  a) V has the feature specification [+AUX]
  b) Aux does not dominate a modal.

Affix Hopping
Move Tense (from Aux to V), provided:
  a) V has the feature specification [–AUX]
  b) Aux does not dominate a modal or neg.

(82) a. * Mary be worked at home.

Ouhalla adds to Affix Hopping
  iii) VP does not dominate a V with the feature specification [+AUX].
Goal: Support Tense.

Preference: Prefer V-raising to Affix Hopping, Affix Hopping to Do-support.

• If Tense is already supported (if there is a modal in Aux), goal is met.
• If Tense is still unsupported and V-raising can be applied (there is a [+AUX] V), apply it; goal is met.
• If Tense is still unsupported and Affix Hopping can be applied (Aux doesn’t dominate Neg), apply it; goal is met.
• If Tense is still unsupported, apply Do-support (as a “last resort”).

V-Raising
Raise V to Aux provided V has the feature specification [+AUX].

Affix Hopping
Move Tense (from Aux to V), provided Aux does not dominate Neg.

(83) a. * John did avoid Bill.
    b. John did avoid Bill.

(84) a. Mary will solve the problem.
    b. Will Mary solve the problem?

(85) a. You can drive my car.
    b. Can you drive my car?

(86)

Aux-raising
Move Aux to Comp, when Comp has the feature specification [+Q].

(87) a. Which problem can Bill solve?
    b. DS: [S'[+Q] [S [Aux can] [VP solve which problem]]]
    c. SS: [S' [NP which problem], [Aux can], [S Bill t, [VP solve t]]]

(88) a. I wonder which problem Bill can solve.
    b. I wonder if Bill can solve this problem.

Aux-raising (second statement)
Move Aux to Comp, provided
i) Comp has the feature specification [+Q]
ii) Comp is situated in a root clause.

(89) a. Was John at home?
    b. DS: [S'[+Q] [S [Aux Tense] [VP be at home]]]
    c. VR: [S'[+Q] [S [Aux [V be], Tense] [VP t, at home]]]
    d. AR: [S' [Aux [V be], Tense] [S John t, [VP t, at home]]]

(90) a. Did John solve the problem?
    b. DS: [S'[+Q] [S [Aux Tense] [VP solve the problem]]]
    c. AR: [S'[Aux Tense], [S John t, [VP solve the problem]]]
    d. do: [S'[Aux Tense [do], [S John t, [VP solve the problem]]]
**Affix Hopping**

Move Tense (from Aux) to V provided

i) Aux does not dominate a Modal or Neg

ii) V has the feature specification [-AUX]

iii) VP does not dominate a V with the feature specification [+AUX]

iv) Aux is adjacent to VP which dominates the targeted V

**or**

**Affix Hopping**

Move Tense (from Aux to V), provided

i) Aux does not dominate Neg.

ii) Aux is adjacent to VP which dominates the targeted V

We could even conflate these two if Neg counts as making Aux non-adjacent to VP.

More soon…