CAS LX 522
Syntax I

12

Case, agreement, and the passive
(chapter 6 continues)

Case

- Recall that pronouns in English show distinctions in case:
  - Subject pronouns are in nominative case
  - Object pronouns are in accusative case
- How can we ensure the correlation?
  1) I saw her.
  2) She saw me.
  3) They saw him.

[ucase:nom]

- Nominative subjects generally appear in the specifier of a *finite* T.
  - Finite T is pretty much any kind of T except the infinitive.
- We can treat case like we treated tense inflection:
  - Suppose T also has a [*u*case:nom] feature.
  - Suppose nominative NPs have a [*u*case:] feature.
  - Suppose the [*u*case:nom] on T can value [*u*case:] on the NP, checking **both**.
  - So T needs a nom NP, and a nom NP needs T.

[ucase:acc]

- Subjects check nominative case with T. Objects have accusative case, which we can treat in the same kind of way.
  - Suppose *v* has [*u*case:acc].
  - Suppose accusative NPs have [*u*case]
  - Suppose the [*u*case:acc] on *v* can value the [*u*case:] feature on the NP, checking both.
- Nominative case is a relation between (finite) T and an NP, accusative case is a relation between *v* and an NP.

Pronouns

- Nominative case is associated with *finite* T.
  - She will charm snakes.
  - I want her to charm snakes.
  - I expect her to charm snakes
    - Non-finite T is not associated with nominative case. It's not actually associated with accusative case either, but we'll come back to that later.
- Because NPs have an *unvalued* [*u*case:] feature, we can suppose that pronouns always enter the numeration the same way, and are valued based on where they are Merged.
  - pronoun [N, *u*case:, …]

NPs need case

- Although in English we only see the morphological effect of case on pronouns, we assume that all NPs have an unvalued [*u*case:] feature.
  - Plenty of languages other than English show case on all NPs, not just on pronouns. Case is something that goes with being an NP. It's just something you often don't hear in English.
- **Notational shortcuts:**
  - [nom] is used for [*u*case:nom] (on T, or NP when checked)
  - [acc] is used for [*u*case:acc] (on *v*, or NP when checked)
  - [case] is used for [*u*case:] (on an NP)

Subject-verb agreement

- Recall that in English, the φ-features of the subject have an effect on the morphology of the verb:

  1) Fans were rioting on Comm Ave.

  2) A fan was rioting on Comm Ave.

- While we're here, we might as well account for this too. It is also an agreement relation, between the subject and, eventually, the verb (or auxiliary, if there is one).

Subject-verb agreement

- What we're after is this:
  The subject (the thing that's getting nominative case) should share/check φ-features with the thing that gets inflection from tense.

  - The φ-features are on the NP that checks nominative case with T.
  - The relevant inflection is valued by T.
  - Maybe it's "passed" from the NP to T, then from T to the uInfl: below.

1) Fans were rioting on Comm Ave.
2) A fan was rioting on Comm Ave.
3) Fans riot on Comm Ave.
4) A fan riots on Comm Ave.

Subject-verb agreement

- So. The verb gets its tense inflection specified by T when, e.g., the [tense:pres] feature of T values the [uInfl:] feature of *v*.

- Since the subject already agrees with T (the [nom] feature of T checks the [case] feature of the subject), we'll incorporate subject agreement into this process.

  [$u$case:nom]

  [$u$case:]

- Notice that we still want this agreement to be mediated by T (sometimes it values, e.g., Perf):

  1) They have been reading novels.

  2) She has been reading novels.

Subject-verb agreement

- Suppose then that T has a [uφ:] feature as well.

- The subject has (interpretable) φ-features that value the [uφ:] feature of T.

  - Fans were rioting on Comm Ave.

  - T [T, uN*, uφ:, nom]   [$u$case:nom]

  - fans [N, φ:pl, case]   [$u$case:]

- So, once T is in the structure, c-commanding *fans* in Spec*v*P, we get:

  - T [T, uN*, ~~uφ:pl~~, ~~nom~~]   [~~$u$case:nom~~]

  - fans [N, φ:pl, ~~nom~~]   [~~$u$case:nom~~]

Subject-verb agreement

- Finally, we suppose that the (checked) [~~uφ:pl~~] feature of T, also values a [uInfl:] feature on a lower *v* (or Perf, or Prog).

- The rules of pronunciation will tell us that a *v* with the verb *riot* adjoined to it sounds like:

  - "riots" if *v* has the feature [uInfl:pres,sg]

  - "riot" if *v* has the feature [uInfl:pres,pl]

- Notice that T values a [uInfl:] feature all at once, with any relevant feature(s) it has (so, tense and φ-features both).
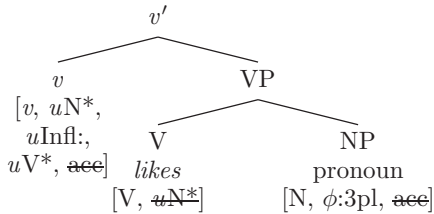
She likes them

- So, let's walk through it.

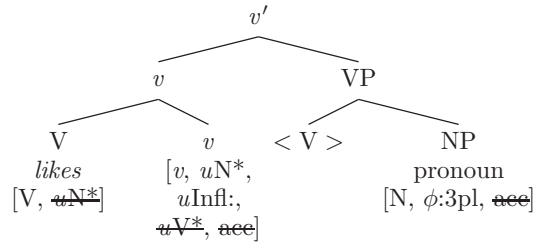- We start by merging *like* and the 3pl pronoun.

```
            VP
           /  \
          V    NP
       likes   pronoun
   [V, u̶N̶*̶]  [N, φ:3pl, case, . . . ]
```
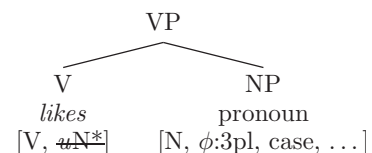
She likes them

- *v* [*v*, *u*N*, *u*Infl:, *u*V*, acc]

- We Merge *v* with VP (HoP).

- The [acc] on *v* matches, values, and checks the [case] on the pronoun, checking itself as well.

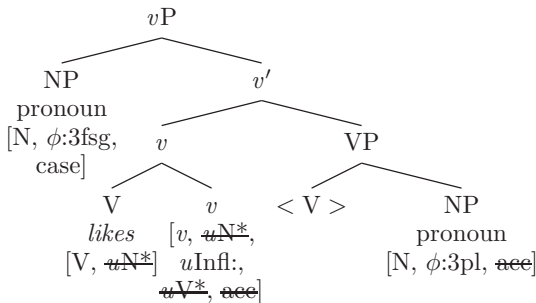 - Agree is lazy, we can do this without any further Merging or Moving.

```
              v′
          ┌────┴────┐
          v         VP
      [v, uN*,    ┌──┴──┐
       uInfl:,    V      NP
       uV*, acc]  likes  pronoun
                  [V, uN*] [N, φ:3pl, acc]
```

She likes them

- The V moves up to adjoin to *v* to check the [*u*V*] feature of *v*.

```
                 v′
          ┌──────┴──────┐
          v             VP
       ┌──┴──┐       ┌──┴──┐
       V     v      < V >   NP
      likes [v, uN*,       pronoun
    [V, uN*] uInfl:,      [N, φ:3pl, acc]
             uV*, acc]
```
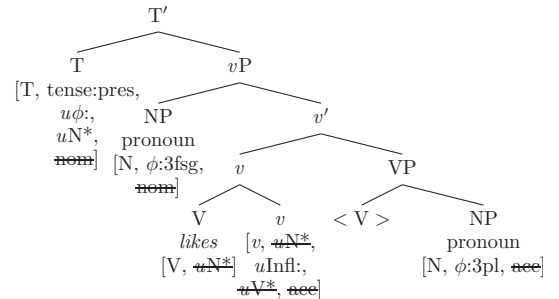
She likes them

- The V moves up to adjoin to *v* to check the [*u*V*] feature of *v*.

- The 3sg feminine pronoun is Merged to check the [*u*N*] feature of *v*.

```
                 vP
          ┌──────┴──────┐
         NP             v′
       pronoun     ┌────┴────┐
      [N, φ:3fsg,  v         VP
       case]    ┌──┴──┐   ┌──┴──┐
               V     v   < V >   NP
             likes [v, uN*,     pronoun
           [V, uN*] uInfl:,    [N, φ:3pl, acc]
                   uV*, acc]
```
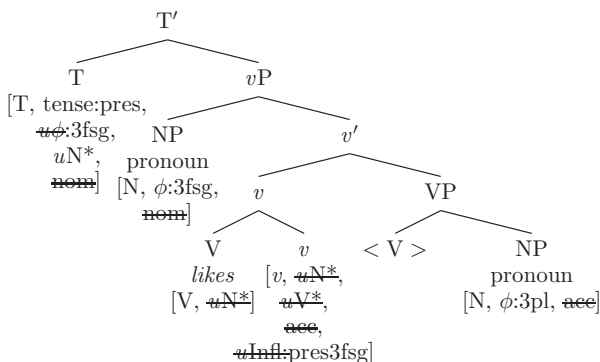
She likes them

- The T is Merged with *v*P (HoP).

- The [nom] feature of T matches, values, and checks the [case] feature of the pronoun, checking itself in the process.
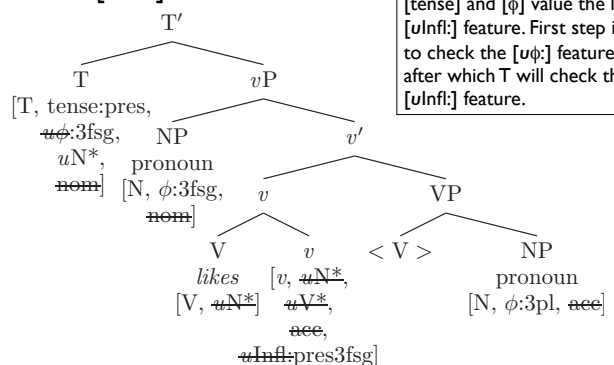
```
                    T′
            ┌───────┴────────┐
            T                vP
    [T, tense:pres,    ┌─────┴─────┐
       uφ:,           NP           v′
       uN*,         pronoun    ┌───┴───┐
       nom]      [N, φ:3fsg,   v       VP
                   nom]     ┌──┴──┐  ┌──┴──┐
                           V     v < V >  NP
                         likes [v, uN*,  pronoun
                       [V, uN*] uInfl:, [N, φ:3pl, acc]
                               uV*, acc]
```

She likes them

- The [φ:3fsg] feature of NP values and checks the [*u*φ:] feature of T.

```
                    T′
            ┌───────┴────────┐
            T                vP
    [T, tense:pres,    ┌─────┴─────┐
       uφ:3fsg,       NP           v′
       uN*,         pronoun    ┌───┴───┐
       nom]      [N, φ:3fsg,   v       VP
                   nom]     ┌──┴──┐  ┌──┴──┐
                           V     v < V >  NP
                         likes [v, uN*,  pronoun
                       [V, uN*] uV*,    [N, φ:3pl, acc]
                               acc,
                               uInfl:pres3fsg]
```
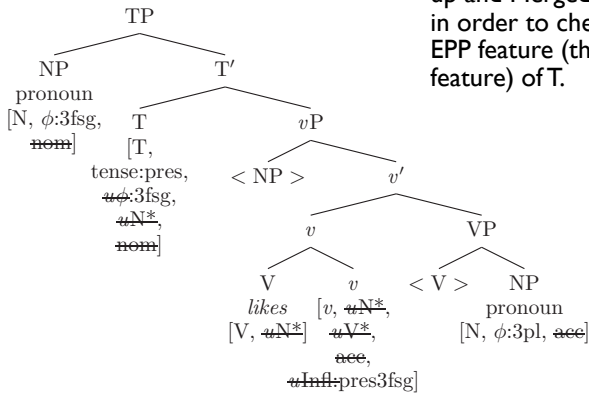
She likes them

- The [*u*φ:3fsg] and [tense:pres] features of T value and check the [*u*Infl:] feature of *v*.

> From now on: (Finite) T can only value a lower [*u*Infl:] feature once T itself has a value for [φ]. Both [tense] and [φ] value the lower [*u*Infl:] feature. First step is always to check the [*u*φ:] feature on T, after which T will check the lower [*u*Infl:] feature.

```
                    T′
            ┌───────┴────────┐
            T                vP
    [T, tense:pres,    ┌─────┴─────┐
       uφ:3fsg,       NP           v′
       uN*,         pronoun    ┌───┴───┐
       nom]      [N, φ:3fsg,   v       VP
                   nom]     ┌──┴──┐  ┌──┴──┐
                           V     v < V >  NP
                         likes [v, uN*,  pronoun
                       [V, uN*] uV*,    [N, φ:3pl, acc]
                               acc,
                               uInfl:pres3fsg]
```
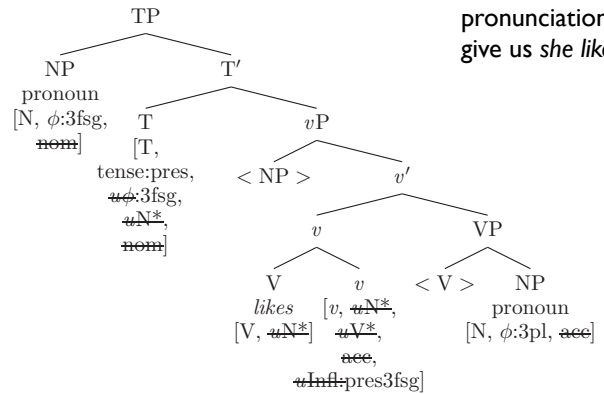
She likes them

- Finally, the NP is moved up and Merged with T′ in order to check the EPP feature (the [*u*N*] feature) of T.

```
              TP
         ┌────┴────┐
        NP         T′
      pronoun  ┌───┴───┐
   [N, φ:3fsg,  T       vP
     nom̶]    [T,    ┌───┴───┐
           tense:pres, < NP >   v′
            u̶φ̶:3fsg,       ┌────┴────┐
             u̶N̶*̶,          v          VP
             nom̶]      ┌───┴───┐  ┌────┴────┐
                      V    v   < V >    NP
                    likes [v, u̶N̶*̶,      pronoun
                  [V, u̶N̶*̶]  u̶V̶*̶,   [N, φ:3pl, a̶c̶c̶]
                          a̶c̶c̶,
                        u̶I̶n̶f̶l̶:pres3fsg]
```

She likes them

- All uninterpretable features are checked, the pronunciation rules give us *she likes them*.

```
              TP
         ┌────┴────┐
        NP         T′
      pronoun  ┌───┴───┐
   [N, φ:3fsg,  T       vP
     nom̶]    [T,    ┌───┴───┐
           tense:pres, < NP >   v′
            u̶φ̶:3fsg,       ┌────┴────┐
             u̶N̶*̶,          v          VP
             nom̶]      ┌───┴───┐  ┌────┴────┐
                      V    v   < V >    NP
                    likes [v, u̶N̶*̶,      pronoun
                  [V, u̶N̶*̶]  u̶V̶*̶,   [N, φ:3pl, a̶c̶c̶]
                          a̶c̶c̶,
                        u̶I̶n̶f̶l̶:pres3fsg]
```

The case of prepositional objects

- Consider the case of the object of a preposition:
  - Computers break near me.

- Now that we've incorporated case into our system, we're stuck with it. Noun phrases come with case. *Computers* has case (nominative) and *me* has case (accusative).

  - The question is: How is the case of *me* checked?

Computers break near me

- *Computers break* is unaccusative; there's no agent, and *computers* is the Theme/Patient, it is the affected object.

- Thus, we have in our numeration:
  - *break* [V, *u*N*]
  - *v*unaccusative[*v*, *u*Infl:, *u*V*]
  - *computers* [N, φ:3pl, case]
  - T [T, *u*φ:, pres, nom, *u*N*]

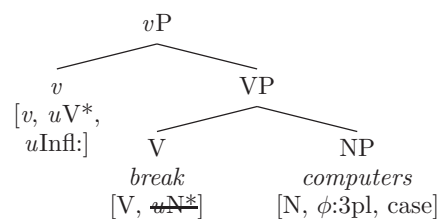- As well as *near* and *me*, which we'll get to in a moment.

Computers break

- First, let's just do *computers break*.
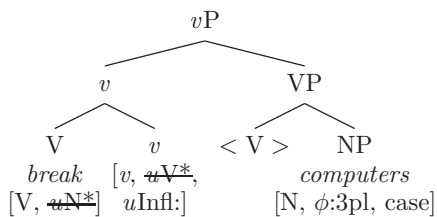- We start by merging *break* and *computers*.

```
              VP
         ┌────┴────┐
         V         NP
       break     computers
     [V, u̶N̶*̶]  [N, φ:3pl, case]
```
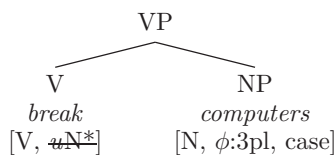
Computers break

- *v* [*v*, *u*Infl:, *u*V*]
- We Merge *v* with VP (HoP).

```
                vP
          ┌─────┴─────┐
          v           VP
      [v, uV*,    ┌────┴────┐
       uInfl:]    V         NP
                break     computers
              [V, u̶N̶*̶]  [N, φ:3pl, case]
```

Computers break

- The V moves up to adjoin to *v* to check the [*u*V*] feature of *v*.

$$vP$$

```
          vP
        /    \
       v      VP
      / \    /   \
     V   v  < V >  NP
   break [v, uV*,    computers
   [V, uN*] uInfl:]  [N, φ:3pl, case]
```
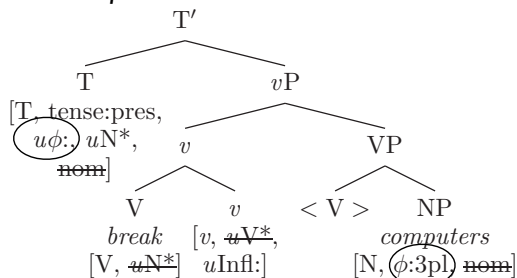
Computers break

- The T is Merged with *v*P (HoP).

- T has the features: [T, pres, *u*φ:, *u*N*, nom].

- The [nom] feature of T can now match the [case] feature of *computers*.

```
              T'
            /    \
           T      vP
    [T, tense:pres,  /  \
     uφ:, uN*,      v    VP
      (nom)        / \   /   \
                  V   v < V >  NP
                break [v, uV*,  computers
                [V, uN*] uInfl:] [N, φ:3pl,(case)]
```
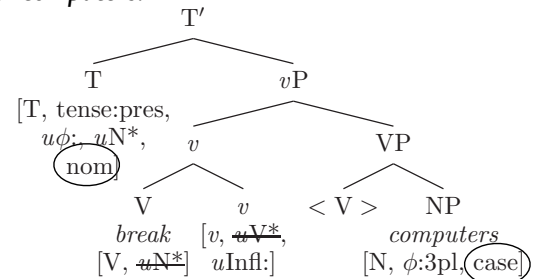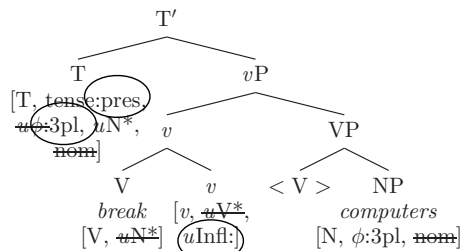
Computers break

- The [nom] feature of T matches, values, and checks the [case] feature of *computers*, checking itself in the process.

- The [*u*φ:] feature of T can also match the [φ:3pl] feature of *computers*.

```
              T'
            /    \
           T      vP
    [T, tense:pres, /  \
     (uφ:) uN*,    v    VP
      nom]        / \   /   \
                 V   v < V >  NP
               break [v, uV*,  computers
               [V, uN*] uInfl:] [N,(φ:3pl) nom]
```
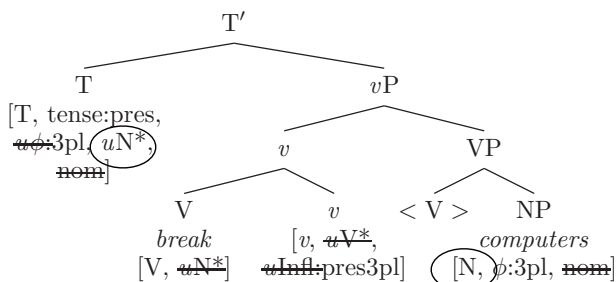
Computers break

- The [φ:3pl] feature of *computers* matches, values, and checks the [*u*φ:] feature of T.

- The [tense:pres] feature of T matches the [*u*Infl:] feature of *v*, which will be valued by both the tense and φ-features of T.

- It's [tense:pres] that matches the [*u*Infl:] feature, but the φ-features "come along" when the [*u*Infl:] feature is valued.

```
              T'
            /    \
           T      vP
    [T, tense:pres, /  \
     uφ:3pl, uN*,  v    VP
      nom]        / \   /   \
                 V   v < V >  NP
               break [v, uV*,  computers
               [V, uN*] (uInfl:) [N, φ:3pl, nom]
```
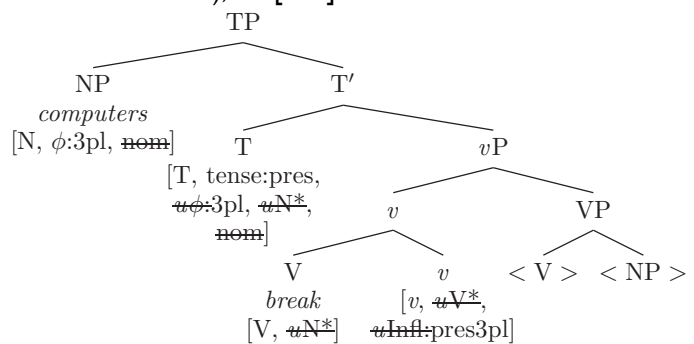
Computers break

- The [*u*N*] feature of T matches the [N] feature of *computers*. This is not sufficient to check the [*u*N*] feature because they are not local, so *computers* is moved up to SpecTP.
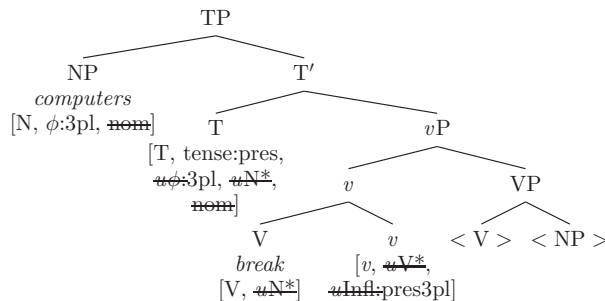
```
              T'
            /    \
           T      vP
    [T, tense:pres, /  \
     uφ:3pl,(uN*)  v    VP
      nom]        / \   /   \
                 V   v < V >  NP
               break [v, uV*,  computers
               [V, uN*] uInfl:pres3pl] [N, φ:3pl, nom]
```

Computers break

- Once the [N] feature of computers is a sister to the T′ that has the [*u*N*] feature (the feature projects from T to T′—it's the same feature), the [*u*N*] feature is checked.

```
              TP
            /    \
          NP      T'
       computers /   \
    [N, φ:3pl, nom]  T      vP
            [T, tense:pres, /  \
             uφ:3pl, uN*,  v    VP
              nom]        / \   /    \
                         V   v < V > < NP >
                       break [v, uV*,
                       [V, uN*] uInfl:pres3pl]
```
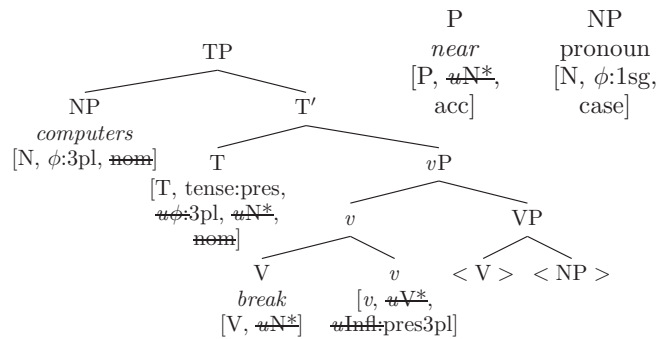
Computers break near me

- Now, let's consider *Computers break near me*.

- *Me* is clearly accusative. There's nothing here that can value a case feature as accusative. That's why I chose *break*. All we're adding to this is *me* (which has accusative case) and the P *near*.
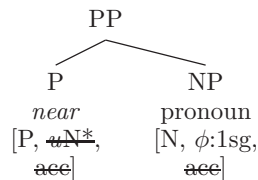
```
                    TP
           ┌─────────┴─────────┐
          NP                   T′
       computers        ┌───────┴───────┐
    [N, φ:3pl, nom]     T               vP
                   [T, tense:pres,  ┌────┴────┐
                    uφ:3pl, uN*,    v         VP
                     nom]        ┌──┴──┐   ┌───┴───┐
                                 V     v  < V >  < NP >
                              break  [v, uV*,
                           [V, uN*]  uInfl:pres3pl]
```

Computers break near me

- **Conclusion**: It must be *near* that is responsible for the accusative case on *me*.

```
                                      P            NP
                    TP             near         pronoun
           ┌─────────┴─────────┐  [P, uN*,    [N, φ:1sg,
          NP                   T′   acc]         case]
       computers        ┌───────┴───────┐
    [N, φ:3pl, nom]     T               vP
                   [T, tense:pres,  ┌────┴────┐
                    uφ:3pl, uN*,    v         VP
                     nom]        ┌──┴──┐   ┌───┴───┐
                                 V     v  < V >  < NP >
                              break  [v, uV*,
                           [V, uN*]  uInfl:pres3pl]
```
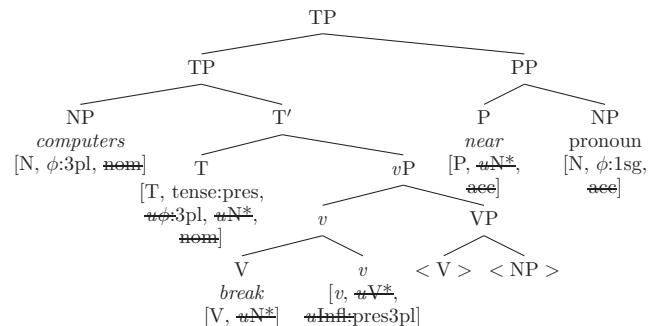
Computers break near me

- Merge *near* and *me* (1sg pronoun). The [N] feature of *me* checks the [uN*] feature of *near*. The [acc] feature of *near* values and checks the [case] feature of *me* (checking itself in the process).

```
              PP
        ┌──────┴──────┐
        P             NP
      near         pronoun
   [P, uN*,     [N, φ:1sg,
    acc]          acc]
```

Near me computers break

- The last step: Adjoin the PP to the TP.
  - To the TP? *Near me* can appear on either side of TP, not *v*P.
  - Computers near me break

```
                            TP
                ┌────────────┴────────────┐
                TP                        PP
        ┌────────┴────────┐         ┌──────┴──────┐
       NP                 T′        P             NP
    computers        ┌─────┴─────┐ near        pronoun
 [N, φ:3pl, nom]     T           vP [P, uN*,  [N, φ:1sg,
                [T, tense:pres, ┌──┴──┐ acc]     acc]
                 uφ:3pl, uN*,   v      VP
                  nom]       ┌──┴──┐ ┌──┴──┐
                             V    v < V > < NP >
                          break  [v, uV*,
                        [V, uN*]  uInfl:pres3pl]
```

P checks accusative

- So, *in general*: A preposition P...
  - Has a [P] category feature
  - Has a [uN*] feature, motivating a Merge with its object.
  - Has an [acc] feature, valuing and checking the [case] feature of its object.
  - T has [T], [uN*] (EPP), [uφ:], [nom]
  - *v* has [*v*], [uInfl:], [uV*], and, if *v* assigns a θ-role, it has [uN*] and [acc].

Double-object constructions

- We've by now covered the sentence

  1) Pat gave books to Chris.

- Pat, books, and *Chris* are all noun phrases, they all need case.
  - *Pat* gets (nom) case from T.
  - *books* gets (acc) case from *v*.
  - *Chris* gets (acc) case from P (*to*).

- What about *Pat gave Chris books*?
  - The "have" kind of "give" must have an [acc] feature.

Adverbs

- Before today, we'd always drawn adjuncts as adjoined to *v*P. This explains why *sloppily* can be either to the left or to the right of *v*P:
    1) Pat sloppily ate lunch.
    2) Pat ate lunch sloppily.
    3) Pat has sloppily eaten lunch.
    4) Pat has eaten lunch sloppily.

- *Sloppily* also seems to be able to adjoin to PerfP or ProgP, at least marginally.
    1) ?Pat might sloppily have eaten lunch.
    2) ?Pat should sloppily be eating lunch.

- But it can't be between a subject and T:
    1) *Pat sloppily might eat lunch.

Manner vs. propositional adverbs

- *sloppily*, *slowly*, *quickly*—all describe the *manner* in which an action takes place. These are **manner adverbs**. They adjoin to *v*P.

- There are other kinds of adverbs as well, however. One such kind are propositional adverbs: *perhaps*, *fortunately*, *interestingly*. These express a kind of attitude on the part of the speaker toward the content of the sentence.

Propositional & temporal adverbs

- **Propositional adverbs** seem to adjoin to TP.
    1) Fortunately, Pat ate lunch.
    2) Pat ate lunch, fortunately.
    3) ?Pat fortunately ate lunch.
    4) ?Pat might have fortunately eaten lunch.

- **Temporal adverbs** also seem to adjoin high.
    1) Today Pat ate lunch.
    2) Pat ate lunch today.
    3) *Pat today ate lunch.

Adverb positions

- Generally speaking, where an adverb attaches depends on its meaning.
    - *v*P for manner adverbs, TP for temporal adverbs, …
- Notice that we predict this now:
    1) Yesterday [Pat completely [finished lunch]].
    2) Yesterday [Pat [finished lunch] completely].
    3) Pat [[finished lunch] completely] yesterday.
    4) Pat [completely [finished lunch]] yesterday.
    5) *Pat [[finished lunch] yesterday completely.
- Later, perhaps, we'll consider additional complexity in adverb placement.

Passives

- The **passive construction** is one where:
    - The original subject disappears (or becomes a *by*-phrase)
    - The original object becomes the subject.
    - The verb appears as *be*+passive participle.
        - The passive participle in English sounds just like the perfective participle.
- Pat took pretzels.   **active**
- Pretzels were taken (by Pat).   **passive**

Passives

- Pat stole books.
- Books were stolen (by Pat).

- In both cases, *books* is getting the Theme/Patient θ-role. By UTAH, it must be originally Merged as NP daughter of VP, in both the active and the passive.

- In fact, the passive is a lot like the unaccusative. An "underlying object" becomes the subject.

Passives

- All we need is the passive auxiliary Pass.
  - *be* [Pass, *u*Infl:] selects a *v*~unaccusative~.
- By selecting for *v*~unaccusative~, the passive auxiliary "removes" an Agent.
  - Not allowed for intransitives, an open mystery.
  - *It was danced (by Pat)
- The passive auxiliary works like other auxiliaries: Pass can value a lower [*u*Infl:] feature, if Pass' own [*u*Infl:] feature is valued by a [tense] feature, it is strong.
  - Lunch was not eaten.
- **Pass is the last auxiliary in the HoP:**
  - Lunch may not have been being eaten.
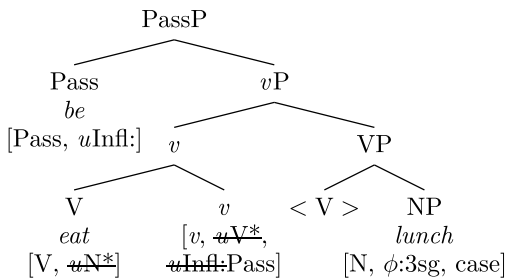 - T > (Neg) > (M) > (Perf) > (Prog) > (Pass) > *v* > V

Lunch was eaten

- For *Lunch was eaten*, we Merge *eat* and *lunch* to build the VP, then Merge an unaccusative *v*…

vP
- v [v, uV*, uInfl:]
- VP
  - V *eat* [V, uN*]
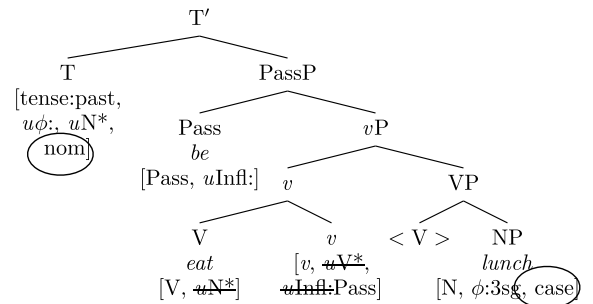 - NP *lunch* [N, ϕ:3sg, case]
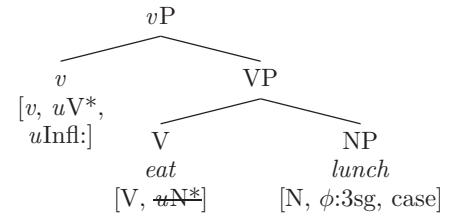
Lunch was eaten

- The V moves up to adjoin to *v* to check the [*u*V*] feature of *v*.
- The Pass auxiliary is Merged (HoP).
  - [Pass] matches, values, checks [*u*Infl:] on v.

PassP
- Pass *be* [Pass, uInfl:]
- vP
  - v
    - V *eat* [V, uN*]
    - v [v, uV*, uInfl:Pass]
  - VP
    - < V >
    - NP *lunch* [N, ϕ:3sg, case]

Lunch was eaten

- T is Merged (HoP).
  - [nom] on T matches, values, checks [case] on *lunch*.
  - [ϕ:3sg] on *lunch* matches, values, checks [$u\phi$:] on T.
  - [past] on T matches, values [*u*Infl:] on Pass.

T′
- T [tense:past, $u\phi$:, uN*, (nom)]
- PassP
  - Pass *be* [Pass, uInfl:]
  - vP
    - v
      - V *eat* [V, uN*]
      - v [v, uV*, uInfl:Pass]
    - VP
      - < V >
      - NP *lunch* [N, ϕ:3sg, (case)]
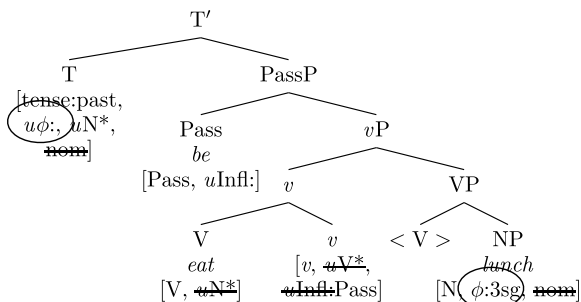

Lunch was eaten

- T is Merged (HoP).
  - [nom] on T matches, values, checks [case] on *lunch*.
  - [ϕ:3sg] on *lunch* matches, values, checks [$u\phi$:] on T.
  - [past] on T matches, values [*u*Infl:] on Pass.

T′
- T [tense:past, $u\phi$:, uN*, (nom)]
- PassP
  - Pass *be* [Pass, uInfl:]
  - vP
    - v
      - V *eat* [V, uN*]
      - v [v, uV*, uInfl:Pass]
    - VP
      - < V >
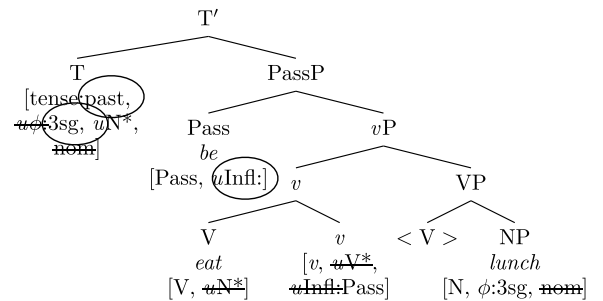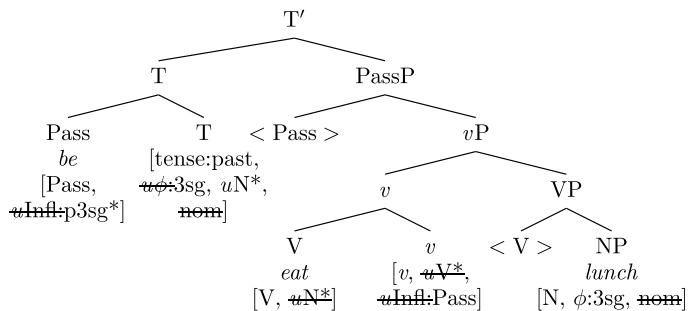   - NP *lunch* [N (ϕ:3sg), nom]

Lunch was eaten

- T is Merged (HoP).
  - [nom] on T matches, values, checks [case] on *lunch*.
  - [ϕ:3sg] on *lunch* matches, values, checks [$u\phi$:] on T.
  - [past] on T matches, values [*u*Infl:] on Pass.
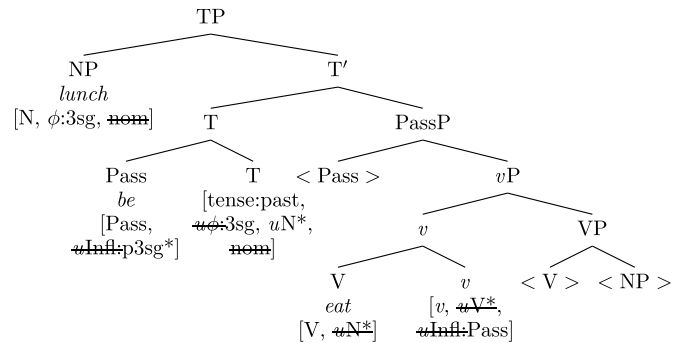
T′
- T [tense:past, ($u\phi$:3sg, uN*), nom]
- PassP
  - Pass *be* [Pass, (uInfl:)]
  - vP
    - v
      - V *eat* [V, uN*]
      - v [v, uV*, uInfl:Pass]
    - VP
      - < V >
      - NP *lunch* [N, ϕ:3sg, nom]

Lunch was eaten

- Pass moves to T (checks [uInfl:past*] on Pass).

```
              T′
         ┌─────┴─────┐
         T          PassP
      ┌──┴──┐    ┌────┴────┐
    Pass    T  < Pass >    vP
     be  [tense:past,    ┌──┴──┐
   [Pass,  u̶φ̶:3sg, uN*,  v     VP
  u̶I̶n̶f̶l̶:p3sg*]   n̶o̶m̶]  ┌─┴─┐  ┌──┴──┐
                     V    v < V >  NP
                    eat [v, u̶V̶*,    lunch
                  [V, u̶N̶*] u̶I̶n̶f̶l̶:Pass] [N, φ:3sg, n̶o̶m̶]
```

Lunch was eaten

- *Lunch* moves to SpecTP (checks [uN*] on T).

```
                    TP
           ┌────────┴────────┐
          NP                 T′
         lunch          ┌────┴────┐
      [N, φ:3sg, n̶o̶m̶]   T        PassP
                     ┌──┴──┐   ┌────┴────┐
                   Pass    T < Pass >    vP
                    be  [tense:past,  ┌──┴──┐
                  [Pass, u̶φ̶:3sg, uN*, v     VP
                 u̶I̶n̶f̶l̶:p3sg*]  n̶o̶m̶]  ┌─┴─┐  ┌──┴──┐
                                    V    v < V > < NP >
                                   eat [v, u̶V̶*,
                                 [V, u̶N̶*] u̶I̶n̶f̶l̶:Pass]
```

Ditransitive passives

- Consider again *Pat gave Chris books.*
  - Chris was given books.
  - *Books were given Chris.

- *Pat gave books to Chris.*
  - Books were given to Chris.
  - *Chris was given books to.

Where does the by-phrase attach?

- Adverb tests can give us a hint…
  - The sandwich was eaten by Pat today at noon
  - The sandwich was eaten by Pat at noon today
  - The sandwich was eaten today _ by Pat _ at noon
  - The sandwich was eaten at noon _ by Pat _ today

  - The dishes were washed by Pat _ poorly _ yesterday
  - The dishes were washed poorly by Pat yesterday
  - The sandwich was eaten by Pat _ sloppily _ at noon
  - The sandwich was eaten sloppily by Pat at noon

- Conclusion?

Japanese Numeral Quantifiers

- Gakusei ga   hon o   4-satu katta
  students nom book acc 4-cl   bought
  'The students bought four books.'
- ?*Gakusei ga   hon o   4-nin katta
   students nom book acc 4-cl   bought
- Gakusei ga   4-nin hon o   katta
  students nom 4-cl   book acc bought
  'Four students bought books.'
- Gakusei ga   kyoo 3-nin kita
  students nom today 3-cl   came
  'Three students came today.'
- Hon o   Taroo ga 2-satu katta
  books acc T nom   2-cl   bought
  'Books, Taroo bought two.'
- Yuube,   kuruma ga doroboo ni 2-dai nusum-are-ta
  last night cars nom   thief by   2-cl steal-pass-past
  'Last night, two cars were stolen by a thief.' (Miyagawa 1989)

Italian *ne*-cliticization

- Maria ha visto Gianni. Maria lo ha visto.
  M   has seen G.   M him has seen.
- Gianni trascorrerà   tre settimane a Milano.
  G   spend.fut3sg 3 weeks   in M
- Gianni ne   trascorrerà   tre (*ne) a Milano.
  G of-them spend.fut3sg 3   in M.
- Alcuni {persone/*ne} trascorreranno tre settimane a Milano
  some   people/of-them spend.fut3pl   3 weeks   in M.
- Telefoneranno tre persone domani
- *Ne telefoneranno tre domani
- Ne arriveranno tre domani
- Ne furono arrestati molti.