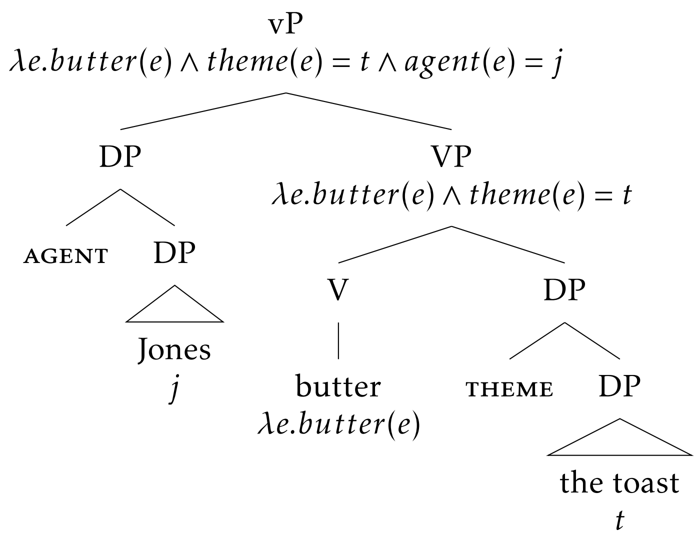
Event semantics

EGG 2024 in Braşov
Deniz Özyıldız, Universität Konstanz
https://deniz.fr/summers/egg2024/

3. Introducing events

Exercise from yesterday

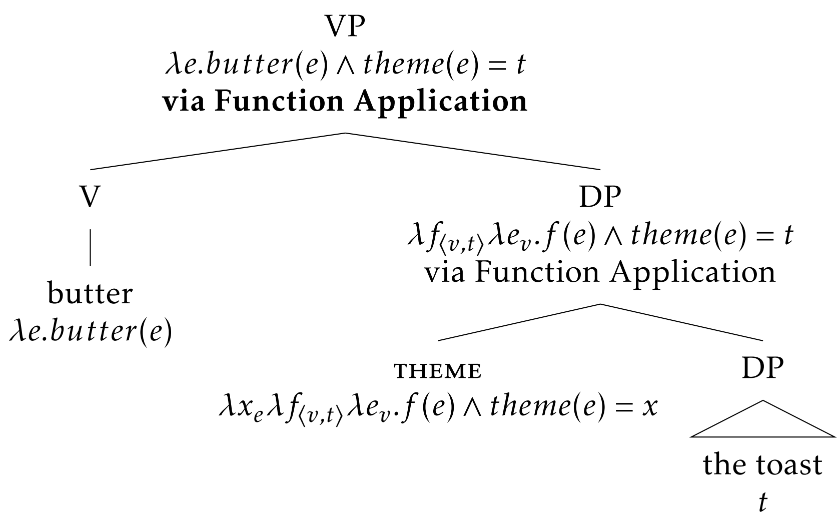
(42)



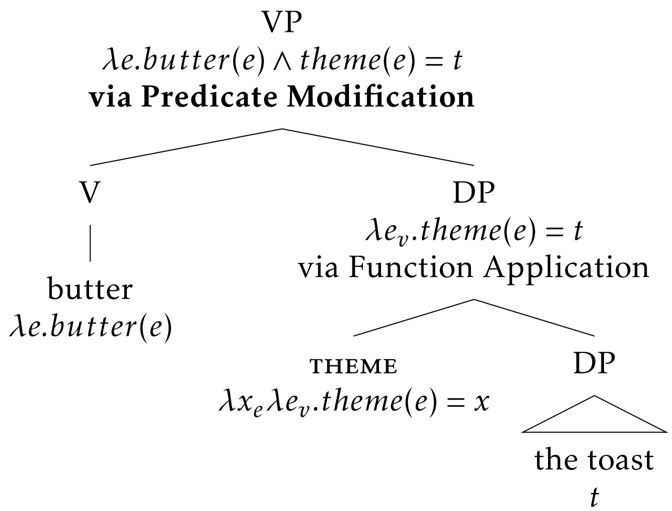
Exercises (+ solutions):

- Define THEME and AGENT.

  Two equally valid options came up:



OR



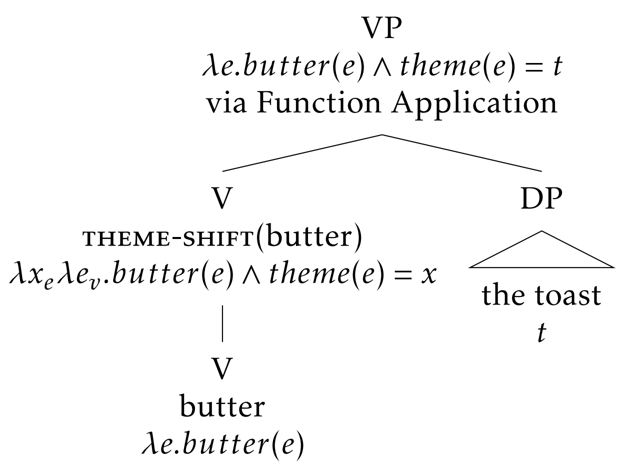
Notice the similarity with our discussion of treating adverbs as verb → verb functions vs. intersective modifiers. The second option is more in keeping with our decision to settle on intersective modification for event semantics.

- There are alternatives to free floating thematic heads in the syntax.

  Can you think of any?

  One could imagine operating on the meaning of the verb instead of the meaning of nominal arguments.

  $$\text{THEME-SHIFT} \rightsquigarrow \lambda f_{\langle v,t \rangle} \lambda x_e \lambda e_v . f(e) \wedge theme(e) = x$$



  Disclaimer: My intention here was to draw an analogy between this operation and *type shifting* (Partee 1987, Heim & Kratzer 1998: ch. 7).

  But the analogy isn't perfect, and we should think of the difference between this and the last bullet point as: Should we manipulate DP denotations or V(P) denotations in introducing thematic role information.

  Pending empirical or other kinds of evidence, the two options seem equally valid.

✧

Type shifting refers to operations that change the type of an expression silently and in situ to help them compose with surrounding expressions when they otherwise couldn't have.

Clasically, at least, type shifting doesn't add *accidental information*, like $theme(e) = x$.

A famous example case that requires type shifting might be:

(i) Kajsa and every Turk rejoiced.

a. Kajsa $\rightsquigarrow$ k   to be shifted

b. every Turk $\rightsquigarrow \lambda P_{\langle e,t\rangle}.\forall x : turk(x) \rightarrow P(x)$

Problem! We can't conjoin something of type e with something of type $\langle\langle e,t\rangle,t\rangle$. And there is no (easy) way of construing *every Turk* as anything of type e.

We settle on the following definition for 'and,' which 'Kajsa' won't fit into.

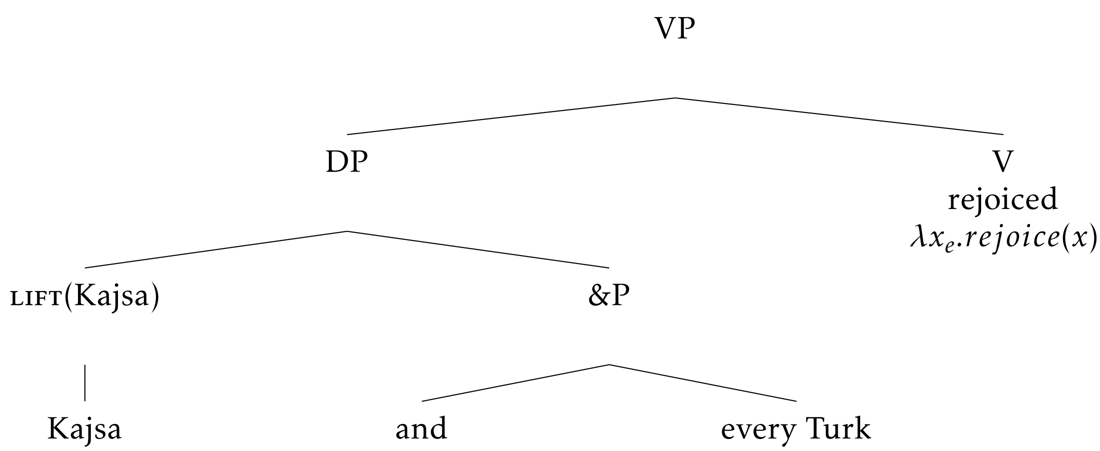
(ii) $\text{and}_{\langle\langle e,t\rangle,t\rangle} \rightsquigarrow \lambda P_{\langle\langle e,t\rangle,\rangle}\lambda Q_{\langle\langle e,t\rangle,t\rangle}\lambda f_{\langle e,t\rangle}.P(f) \wedge Q(f)$

Solution: "Lift" the type of Kajsa so that it becomes something of type $\langle\langle e,t\rangle,t\rangle$.

(iii) $\textsc{lift}$(Kajsa) $\rightsquigarrow \lambda R_{\langle e,t\rangle}.R(k)$

The set of properties of type $\langle e,t\rangle$ that Kajsa satisfies.

The following structure should now be interpretable. Compute the denotation of VP. Focus on the intermediate steps and don't take shortcuts. (Events are not involved.)



3.4. Champollion (2015)

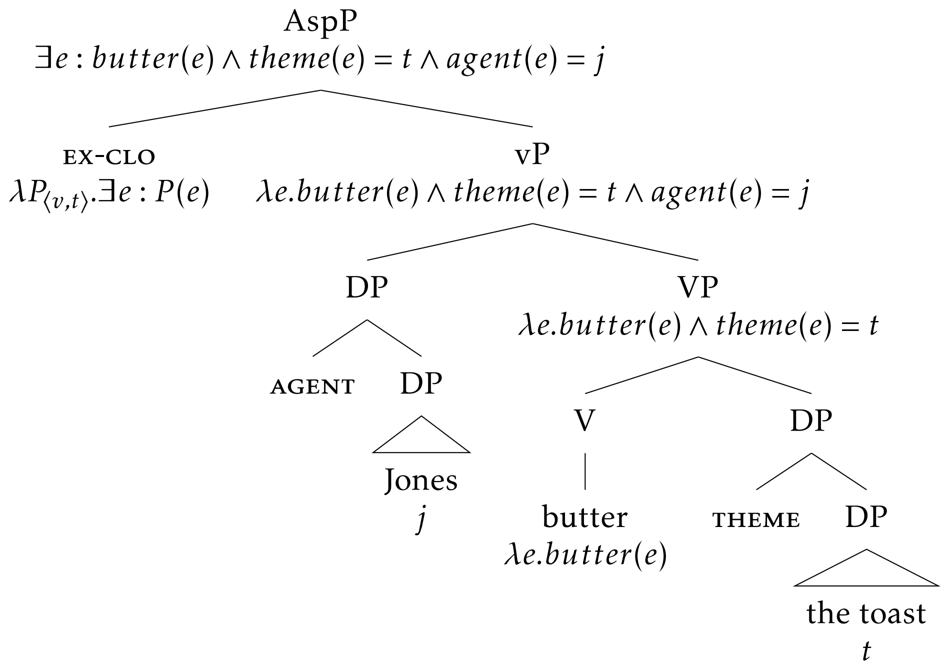
3.4.1. Preliminaries

Existential closure Above, we've been moving freely between *predicates of events*, like $\lambda e.run(e,j)$, and *existential statements*, like $\exists e : run(e,j)$.

In one tradition, that existential quantifier is introduced in the syntax by grammatical aspect (on which, more later).

A simplification that doesn't introduce the details of aspect is (51):

(51)



Introducing negation Assume the following definition for negation, of type $\langle t,t\rangle$:

(52) not $\rightsquigarrow \lambda p_t.\neg p$

There is only one node in the tree above that has an expression of type t: AspP. That is then the only place that we can introduce negation. That yields (53).

(53)

- NegP: $\neg\exists e : butter(e) \wedge theme(e) = t \wedge agent(e) = j$
  - not: $\lambda p_t.\neg p$
  - AspP: $\exists e : butter(e) \wedge theme(e) = t \wedge agent(e) = j$
    - EX-CLO: $\lambda P_{\langle v,t\rangle}.\exists e : P(e)$
    - vP: $\lambda e.butter(e) \wedge theme(e) = t \wedge agent(e) = j$
      - Jones buttered the toast

This is good. When we say (54a), what we mean is that no event of buttering the toast took place, which is what (54b) says.

(54) a. Jones didn't butter the toast.
b. $\neg\exists e : butter(e) \wedge theme(e) = t \wedge agent(e) = j$:)

An alternative would have been to have $\neg$ scope below $\exists e$. These truth conditions are too weak, made true by almost any event.

(55) $\exists e : \neg[butter(e) \wedge theme(e) = t \wedge agent(e) = j]$ can't derive ⇒ :)

But, given classical negation, we can't derive these truth conditions anyway.

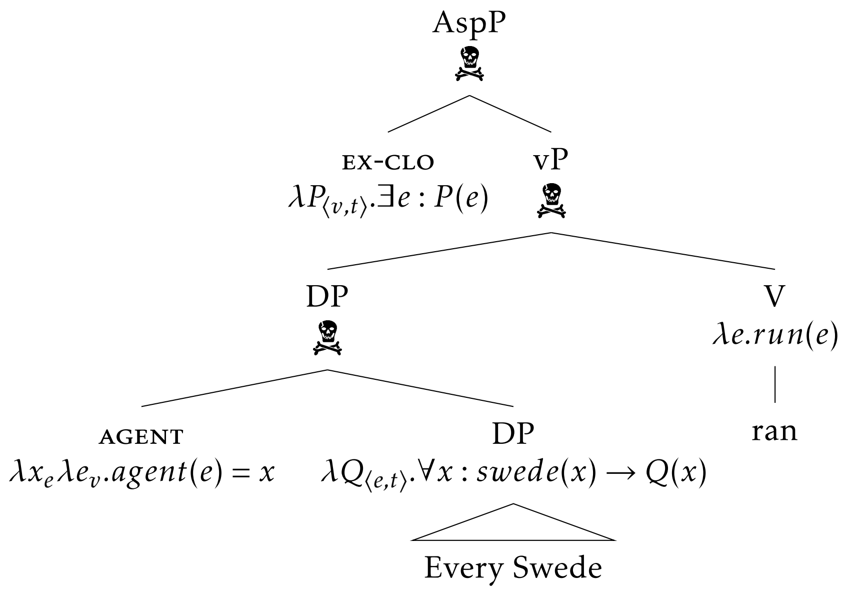
Introducing quantifier phrases In one tradition, quantifier phrases are analyzed as expressions of type $\langle\langle e,t\rangle,t\rangle$.

(56) Illustration in a system sans events

- VP: $\forall x : swede(x) \rightarrow run(x)$
  - DP: $\lambda Q_{\langle e,t\rangle}.\forall x : swede(x) \rightarrow Q(x)$
    - every Swede
  - V: $\lambda x_e.run(x)$
    - ran

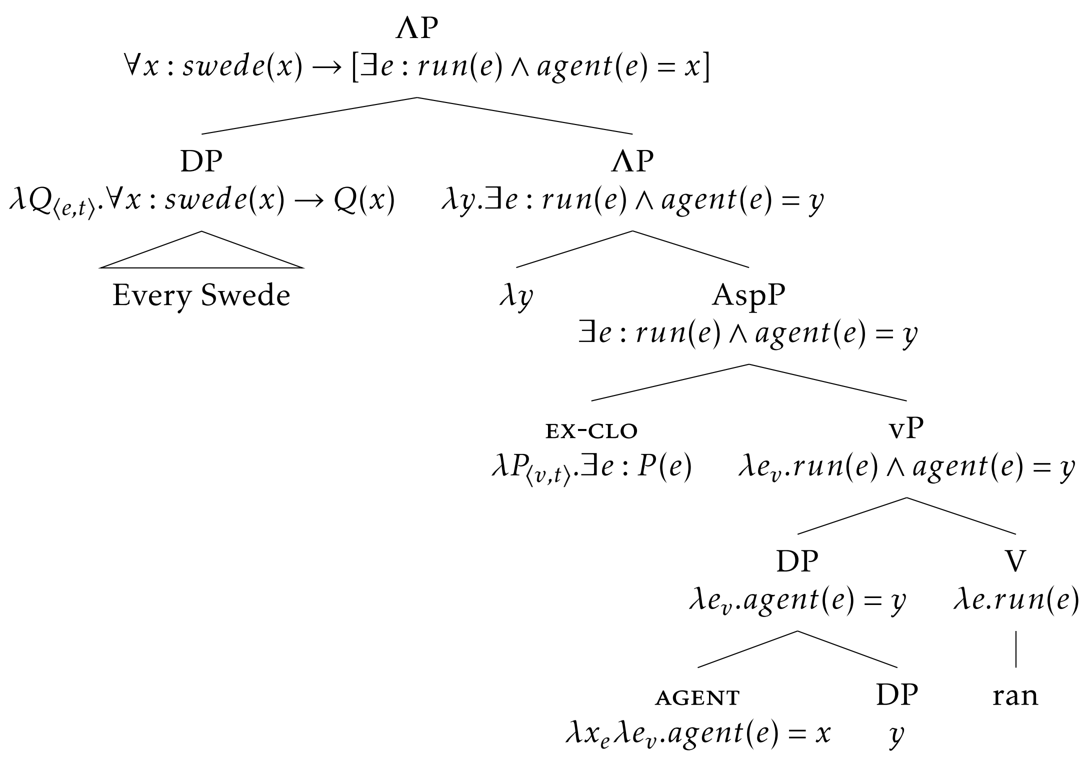
We can try to insert "Every Swede" in a tree that incorporates events... But that'll give rise to a type mismatch.

(57)



Except that we know how to handle this precise kind of type mismatch: **Quantifier Raising** (again, in one tradition).

(58)



Here again, these truth conditions, in (59b), approximate well the meaning of a sentence like (59a): For every Swede, there is a different run that they're the agent of.

(59) a. Every Swede ran.
b. $\forall x : swede(x) \rightarrow [\exists e : run(e) \wedge agent(e) = x]$:)

And again, an alternative would have been to swap the scope of $\forall x$ and $\exists e$, as in (60):

(60) $\exists e : \forall x : swede(x) \rightarrow [run(e) \wedge agent(e) = x]$ can't derive ⇒ :)

This is problematic at least in two places: There's at least one event that is a run (problem #1) and whose single agent is every Swede (problem #2).

But because Quantifier Raising can only target a node of type t, and the only such available node is above EX-CLO, we don't have the option of deriving this scope configuration.

Introducing conjunction Assume that 'party' and 'exercise' are both neo-Davidsonian event predicates, in (61).

(61) a. exercises $\rightsquigarrow \lambda e_v.exercise(e)$
b. parties $\rightsquigarrow \lambda e_v.party(e)$

How would you define "and," as it's used in (62)?

(62) Chiara parties and exercises.

Try to keep minimal silent material that you assume.

3.4.2. Transition

The assumptions of system presented above (EX-CLO, its position, $\langle\langle e,t\rangle,t\rangle$ QPs, quantifier raising) are but some among many different world views.

With even slightly different assumptions, it is possible to generate (unwanted) truth conditions like:

(63) a. Jones didn't butter the toast.
b. $\exists e : \neg[butter(e) \wedge theme(e) = t \wedge agent(e) = j]$

(64) a. Every Swede ran.
b. $\exists e : \forall x : swede(x) \rightarrow [run(e) \wedge agent(e) = x]$

✧

Even with the assumptions that we were making, we will run into problems at some point:

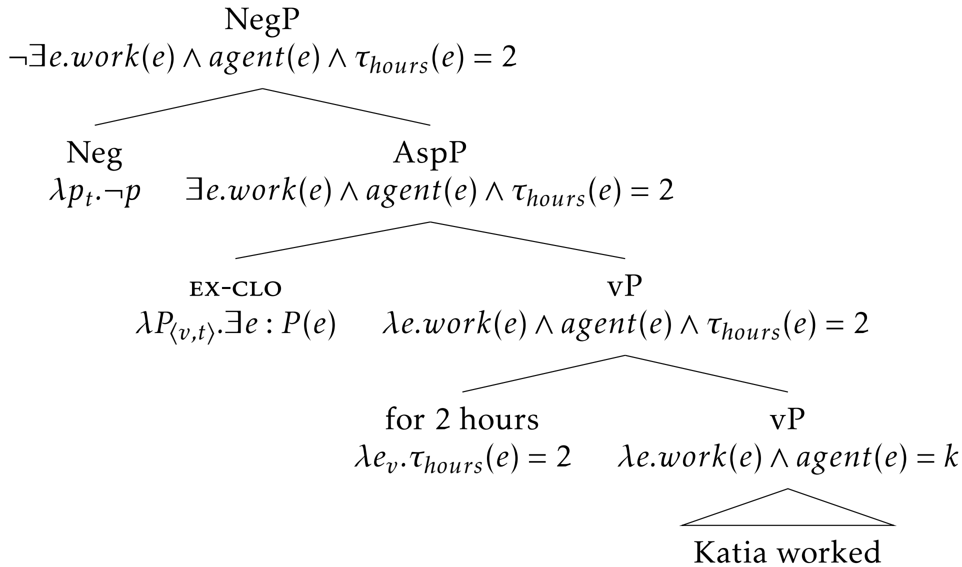
(65) Katia didn't work for two hours.
a. It wasn't the case that Katia worked for two hours.
b. For two hours, it wasn't the case that Katia worked.

Let's define "for two hours" as follows:

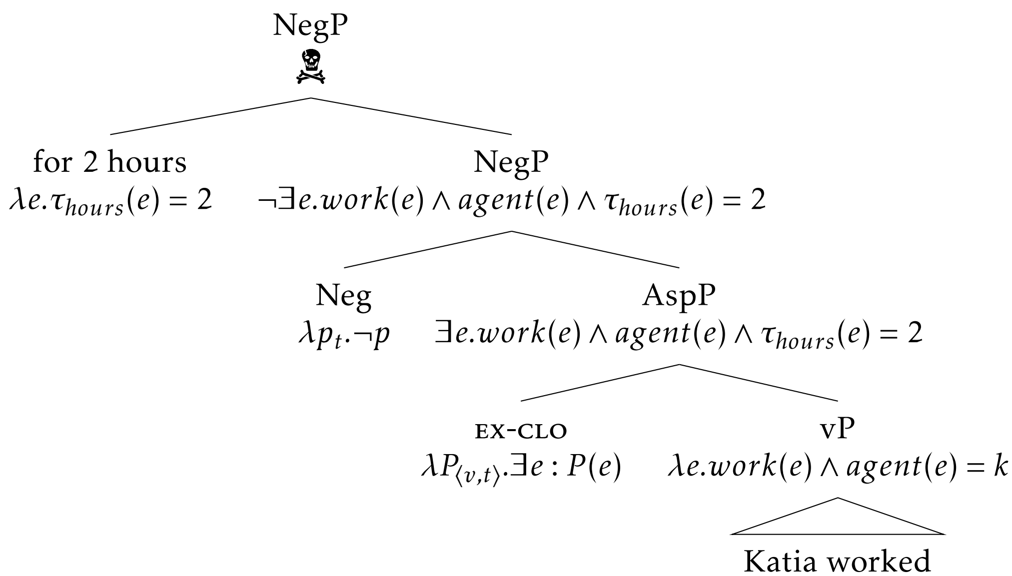
(66) for two hours $\rightsquigarrow \lambda e_v.\tau_{hours}(e) = 2$

Becaues this is an event predicate, it's possible to introduce it below EX-CLO, so long as an event variable is open:

(67) It wasn't the case that Katia worked for two hours.



(68) For two hours, it wasn't the case that Katia worked. (unsuccessful attempt)



In light of examples like this... one is tempted to change our assumptions about, e.g., negation (Krifka 1989).