LING 510: Midterm review

October 30, 2026

Shaunak Phadnis
sphadnis@umass.edu
Office hours: W 11–1PM, N421H | Zoom

Plan. In Part 1 you work in groups: each group builds one piece of a big sentence, John's cat definitely won a competition at school, and then we put the pieces together. Part 2 gives extra practice with presuppositions.

Part 1. Group work, then the big derivation: 's, definitely, won, at; FA, NN, TN, PM; checking presuppositions

Part 2. A presupposition trigger (stop): presupposition tests, a lexical entry, and a derivation

Composition rules

Terminal Nodes (TN). If α is a terminal node, $[\![\alpha]\!]$ is given by the lexicon.

Non-Branching Nodes (NN). If α is a node with sole daughter β, then $[\![\alpha]\!] = [\![\beta]\!]$.

Function Application (FA). If α is a branching node with daughters β and γ, and $[\![\beta]\!]$ is a function whose domain contains $[\![\gamma]\!]$, then $[\![\alpha]\!] = [\![\beta]\!]([\![\gamma]\!])$.

Predicate Modification (PM). If α is a branching node with daughters β and γ, and both $[\![\beta]\!]$ and $[\![\gamma]\!]$ are of type $\langle e, t\rangle$, then

$$[\![\alpha]\!] = \left[\lambda x_e.\, T \text{ iff } [\![\beta]\!](x) = T \text{ and } [\![\gamma]\!](x) = T\right].$$

Presuppositions in lambda notation: three things to remember

1. A lexical presupposition is written as a **domain condition**: the part between the colon and the period.

$$[\lambda x : \underbrace{x \in D_e \text{ and } x \text{ bla}}_{\text{domain condition (presupposition)}} . \underbrace{T \text{ iff } x \text{ is mla}}_{\text{value description (assertion)}}]$$

2. If the argument does not meet the domain condition, you **cannot** β**-reduce**. The result is **undefined**. Undefined is *not* the same as false.

3. Undefinedness **passes up the tree**: if a daughter's denotation is undefined, the mother's is too.

1 Group work: building a big derivation

Our goal is the truth conditions (and presuppositions!) of this sentence:

(1) John's cat definitely won a competition at school.

That is a lot at once. So we split it up. Each group gets **one small sentence** that contains **one piece** of (1), together with a situation to test it on. Once every group has its piece, we put them together as a class.

Group	*Sentence*	*New piece*	*Skill*
A	John's mother smokes.	's	a determiner with a presupposition
B	Mia definitely bakes.	definitely	adding an adverb (the four-step recipe)
C	Blake won a competition.	won	a transitive verb with a presupposition
D	Ana is at home.	at home	a PP of type $\langle e, t\rangle$

Recipe for a new lexical entry

Step 1. **Guess the truth conditions** of a simple sentence with the word, in plain English. Test your guess on a situation.

Step 2. **Reason out the type** from the tree: what is the word's sister, and what must the mother be?

Step 3. **Write the lexical entry.** Start from the type: one λ per argument, in the order the word combines with them. Then fill in the value description so it matches Step 1. If the word has a presupposition, put it in a domain condition.

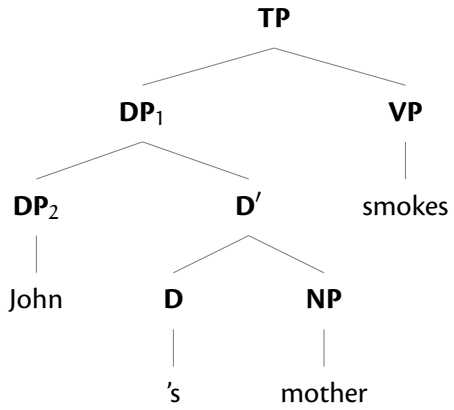
Step 4. **Derive** the sentence, and check that you get back exactly the truth conditions from Step 1.

Lexicon for this part. Each group gets its own situation.

$$[\![\text{John}]\!] = \text{John} \quad [\![\text{Mia}]\!] = \text{Mia} \quad [\![\text{Blake}]\!] = \text{Blake} \quad [\![\text{Ana}]\!] = \text{Ana}$$
$$[\![\text{home}]\!] = \text{h} \quad [\![\text{school}]\!] = \text{sch} \quad [\![\text{competition}]\!] = \text{c}$$
$$[\![\text{a}]\!] = [\lambda x \in D_e.\ x] \qquad \text{(the identity function on individuals)}$$
$$[\![\text{mother}]\!] = [\lambda x \in D_e.\ \text{T iff x is a mother}] \qquad [\![\text{cat}]\!] = [\lambda x \in D_e.\ \text{T iff x is a cat}]$$
$$[\![\text{smokes}]\!] = [\lambda x \in D_e.\ \text{T iff x smokes}] \qquad [\![\text{bakes}]\!] = [\lambda x \in D_e.\ \text{T iff x bakes}]$$
$$[\![\text{is}]\!] = [\lambda f \in D_{\langle e,t\rangle}.\ f]$$
$$[\![\text{'s}]\!] = \ ?? \qquad [\![\text{definitely}]\!] = \ ?? \qquad [\![\text{won}]\!] = \ ?? \qquad [\![\text{at}]\!] = ??$$

Group A: John's mother smokes

(2) John's mother smokes.



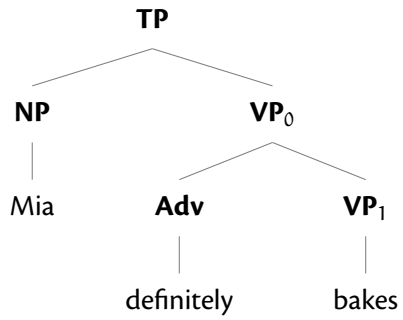
Group A

Situation s_A. Sue is John's mother, and she smokes. Lee is Ana's mother, and she doesn't smoke. Pebble is a rock (rocks have no mother). John has two sisters, Kate and Liz. Both smoke.

1. John's mother picks out an individual (Sue). So DP_1 is type e. Use the tree to work out the type of 's.
2. Run the negation and question tests on (2). What does it presuppose about John?
3. Write a lexical entry for 's. *Hint:* John's mother means roughly "the mother of John". Put the presupposition from question 2 in a domain condition.
4. Derive (2). Where do you check the presupposition?
5. True, false, or undefined in s_A? (a) (2) (b) Ana's mother smokes. (c) Pebble's mother smokes. (d) John's sister smokes.

Group B: Mia definitely bakes

(3) Mia definitely bakes.



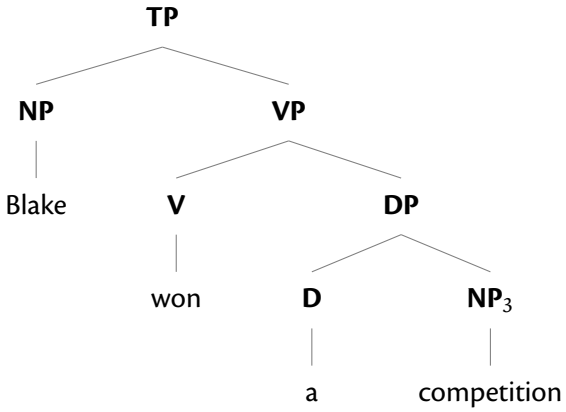
Group B

Situation s_B. Mia bakes bread every Sunday. Everyone has seen her do it. Ben also bakes, but nobody has seen him. People only think so because his kitchen smells of bread. Cal never bakes.

1. **Step 1.** When is (3) true? Is Ben bakes true in s_B? What about Ben definitely bakes? What does definitely add?
2. **Step 2.** What type must definitely be? Why can't it combine with bakes by PM?
3. **Step 3.** Write a lexical entry for definitely.
4. **Step 4.** Derive (3).
5. Does (3) *presuppose* that Mia bakes, or only *entail* it? Use the negation test.
6. True, false, or undefined in s_B? (a) (3) (b) Ben definitely bakes. (c) Cal definitely bakes.

Group C: **Blake won a competition**

(4) Blake won a competition.



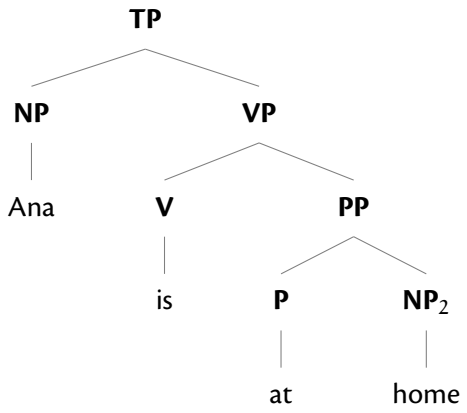
Group C

Situation s_C. The competition (c) is the school bake-off. Blake and Dana entered it. Blake won. Eve did not enter.

1. What type must won be?
2. Run the negation and question tests on (4). What does it presuppose? What does it assert?
3. Write a lexical entry for won. *Hint:* the presupposition is about the winner *and* the competition. Which λ does it go on?
4. Derive (4). Where do you check the presupposition?
5. True, false, or undefined in s_C? (a) (4) (b) Dana won a competition. (c) Eve won a competition.

Group D: Ana is at home

(5) Ana is at home.

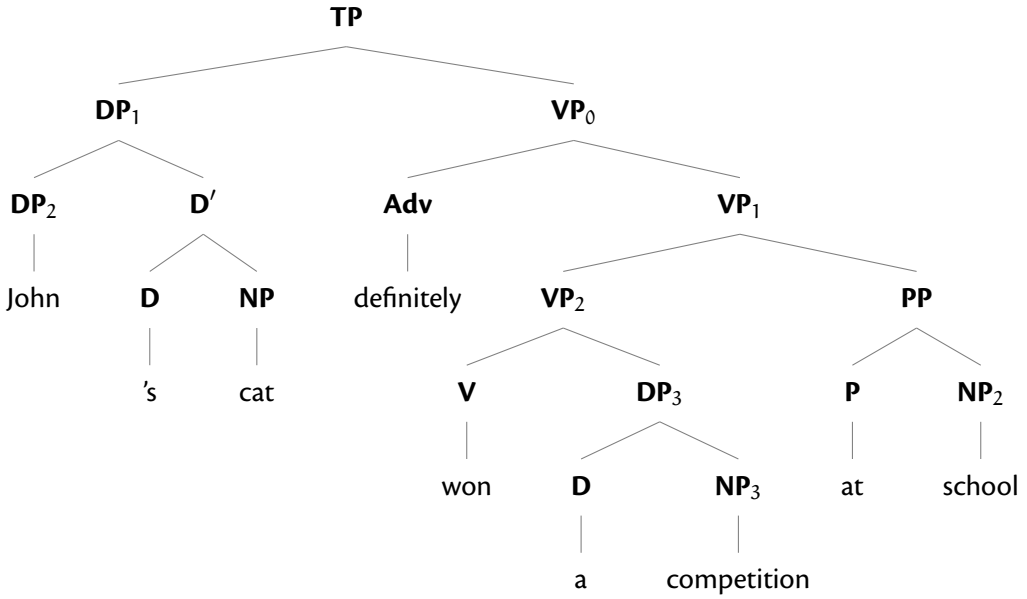


Group D

Situation s_D. Ana is at home. Rex is at the park. Pebble (a rock) is at home.

1. Annotate the tree with types and rules.
2. Derive (5). What is ⟦PP⟧? What does is contribute?
3. True, false, or undefined in s_D? (a) (5) (b) Rex is at home. (c) Pebble is at home.
4. Write ⟦at school⟧. What type is it?

Putting it together: John's cat definitely won a competition at school



Whole class: the big sentence

Situation *s*. John owns exactly one cat, Whiskers. Ana owns a cat called Felix. The competition (c) is the cat show held at school. Whiskers and Felix both entered and were at school for it. The judge announced that Whiskers won, so it is certain.

1. Match each part of the tree to the group that already built it. Which node did *no* group build?
2. That node is VP_1. What are the types of its two daughters? Which rule combines them?
3. Annotate the whole tree with types and rules.
4. Derive (1). Reuse the groups' results as subproofs. Do only the new step in full.
5. List every presupposition of (1). Which word does each come from, and at which node is it checked?
6. True, false, or undefined? Say which presupposition fails, if any.
   (a) In *s* as described.
   (b) John also owns Felix.
   (c) Whiskers never entered the show. Felix won.
   (d) Whiskers won, but the result is under appeal, so nobody is certain.
   (e) The show was held online. Whiskers won from home.

2 A presupposition trigger: stop

In Part 1 the presuppositions came from 's and won. Here is one more classic presupposition trigger: the change-of-state verb stop.

The situation for this part

The situation s. There are five individuals. Locations do not change over time. Times come in order: t_1, then t_2, then t_3. The time we are talking about ("now") is t_2.

				asleep at…		
Individual	*Kind*	*Animate?*	*Location*	t_1	t_2 (now)	t_3
Mia	cat	✓	home	✓	✓	✓
Tom	cat	✓	home	✗	✓	✗
Rex	dog	✓	park	✓	✗	✓
Ana	person	✓	park	✗	✓	✓
Pebble	rock	✗	home	–	–	–

Lexicon.

$[\![\text{Mia}]\!] = \text{Mia}$ $[\![\text{Tom}]\!] = \text{Tom}$ $[\![\text{Rex}]\!] = \text{Rex}$ $[\![\text{Ana}]\!] = \text{Ana}$ $[\![\text{Pebble}]\!] = \text{Pebble}$

$[\![\text{sleeps}]\!]^1 = ??$ $[\![\text{stopped}]\!] = ??$

You will fill in both ?? entries in this part.

(6) Rex stopped sleeping. (said about time t_2)

"Just before t_2" means t_1. We treat sleeping as meaning the same as sleeps.

Step A: what is asserted, what is presupposed?

Here are three things you might infer from (6):

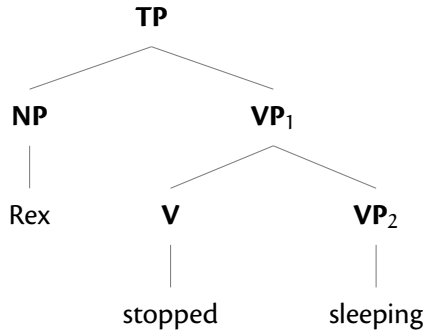
(7) a. Rex is not asleep at t_2.
b. Rex was asleep just before t_2.
c. Rex is animate.

Practice 5A: presupposition tests on stop

1. Apply the negation test and the question test to (6). For each of (7-a)–(7-c): does it survive?
2. Now apply the negation test to Rex isn't sleeping. Does (7-b) survive there? What does this tell you about *which word* the presupposition (7-b) comes from?
3. Where does (7-c) come from?

[1] "asleep" means asleep at the time we are talking about. Unless we say otherwise, that time is now (t_2).

Steps B–D: type, lexical entry, derivation



Practice 5B: stop

1. **Type.** Annotate the tree with types. What type must stopped be? Which other word in this handout has the same type?
2. **Lexical entry.** Write an entry for stopped. It should
   - take its arguments in the order the tree gives them,
   - put (7-b) in a domain condition,
   - make the value description match (7-a).

   Hint: the presupposition is about x *and* f. Which λ does the domain condition have to go on?
3. **Derivation.** Derive (6) (at t_2). Check *every* presupposition along the way.
4. **Model check.** True, false, or undefined (at t_2, unless stated)? Say which presupposition fails, if any.
   (a) Mia stopped sleeping.
   (b) Tom stopped sleeping.
   (c) Pebble stopped sleeping.
   (d) Tom stopped sleeping, said about t_3.

Your turn: started

(a) Run the negation and question tests on Ana started sleeping. What does it presuppose? What does it assert?

(b) Write a lexical entry for started. (Hint: it is the mirror image of stopped.)

(c) True, false, or undefined? (i) Ana started sleeping at t_2. (ii) Rex started sleeping at t_2. (iii) Rex started sleeping at t_3.

Midterm checklist

Lambdas

- Can I name the variable, the domain condition, and the value description of a lambda term?
- Do I β-reduce one λ at a time, outermost first? Do I check the domain condition *before* reducing?
- Do I rename variables when two lambdas use the same letter?
- Can I go between a table, a set, and a lambda term? Do I know that a partial function is a table with missing rows?

Types

- Can I read the type off a lambda term (variable type $\rightarrow$ value type)?
- Can I find a missing type in a tree, starting from e (names) and t (sentences)?
- Do I know when PM applies instead of FA (two $\langle e, t \rangle$ sisters)?

New words (like the adverb)

- Guess truth conditions in plain English first, and test the guess on a situation.
- Get the type from the tree, then write one λ per argument, in the order the word meets them.
- After the derivation, check you got back the truth conditions you guessed.

Derivations

- One rule per line, and label it: TN, NN, FA, PM, or β-reduction.
- Break long derivations into subproofs, bottom up, and reuse them.
- Every time a function with a domain condition gets applied: **stop and check the presupposition.**

Presuppositions

- Negation test and question test: does the inference survive?
- A presupposition goes in the **domain condition**. The assertion goes in the **value description**.
- Presupposition failure means **undefined**, not false. Undefinedness passes up the tree.
- If a presupposition mentions more than one variable, put it on the λ where all of them have been introduced.
- One sentence can have presuppositions from several words. Check each one.