Ling 566
Oct 27, 2016

Lexical Types

Overview

- Motivation for lexical hierarchy
- Default inheritance
- Tour of the lexeme hierarchy
- The Case Constraint
- *pos* vs. *lexeme*
- Reading Questions

Motivation

- We've streamlined our grammar rules...
- ...by stating some constraints as general principles
- ...and locating lots of information in the lexicon.
- Our lexical entries currently stipulate a lot of information that is common across many entries and should be stated only once.

- Examples?

- Ideally, particular lexical entries need only give phonological form, the semantic contribution, and any constraints truly idiosyncratic to the lexical entry.

Lexemes and Words

- **Lexeme**: An abstract proto-word which gives rise to genuine words. We refer to lexemes by their 'dictionary form', e.g. 'the lexeme *run*' or 'the lexeme *dog*'.
- **Word:** A particular pairing of form and meaning. *Running* and *ran* are different words

Lexical Types & Lexical Rules

- Lexemes capture the similarities among *run*, *runs*, *running*, and *run*.
- The lexical type hierarchy captures the similarities among *run*, *sleep*, and *laugh*, among those and other verbs like *devour* and *hand*, and among those and other words like *book*.

  Q: What do *devour* and *book* have in common?

  A: The SHAC
- Lexical rules capture the similarities among *runs*, *sleeps*, *devours*, *hands*,...

Default Inheritance

Q: Why do we have default inheritance?

A: Generalizations with exceptions are common:

- Most nouns in English aren't marked for CASE, but pronouns are.
- Most verbs in English only distinguish two agreement categories (*3sing* and *non-3sing*), but *be* distinguishes more.
- Most prepositions in English are transitive, but *here* and *there* are intransitive.
- Most nominal words in English are 3rd person, but some (all of them pronouns) are 1st or 2nd person.
- Most proper nouns in English are singular, but some (mountain range names, sports team names) are plural.

Default Inheritance, Technicalities

If a type says ARG-ST /< NP >, and one of its subtypes says ARG-ST < >, then the ARG-ST value of instances of the subtype is < >.

If a type says ARG-ST < NP >, and one of its subtypes says ARG-ST < >, then this subtype can have no instances, since they would have to satisfy contradictory constraints.

Default Inheritance, More Technicalities

- If a type says MOD / < S >, and one of its subtypes says MOD <[SPR < NP>] >, then the ARG-ST value of instances of the subtype is what?

$$\left[\text{MOD} \left\langle \left[\begin{array}{ll}\text{HEAD} & / \; \textit{verb} \\ \text{SPR} & / \; \langle \text{NP} \rangle \\ \text{COMPS} & / \; \langle\, \rangle\end{array}\right] \right\rangle\right]$$

- That is, default constraints are ‘pushed down’

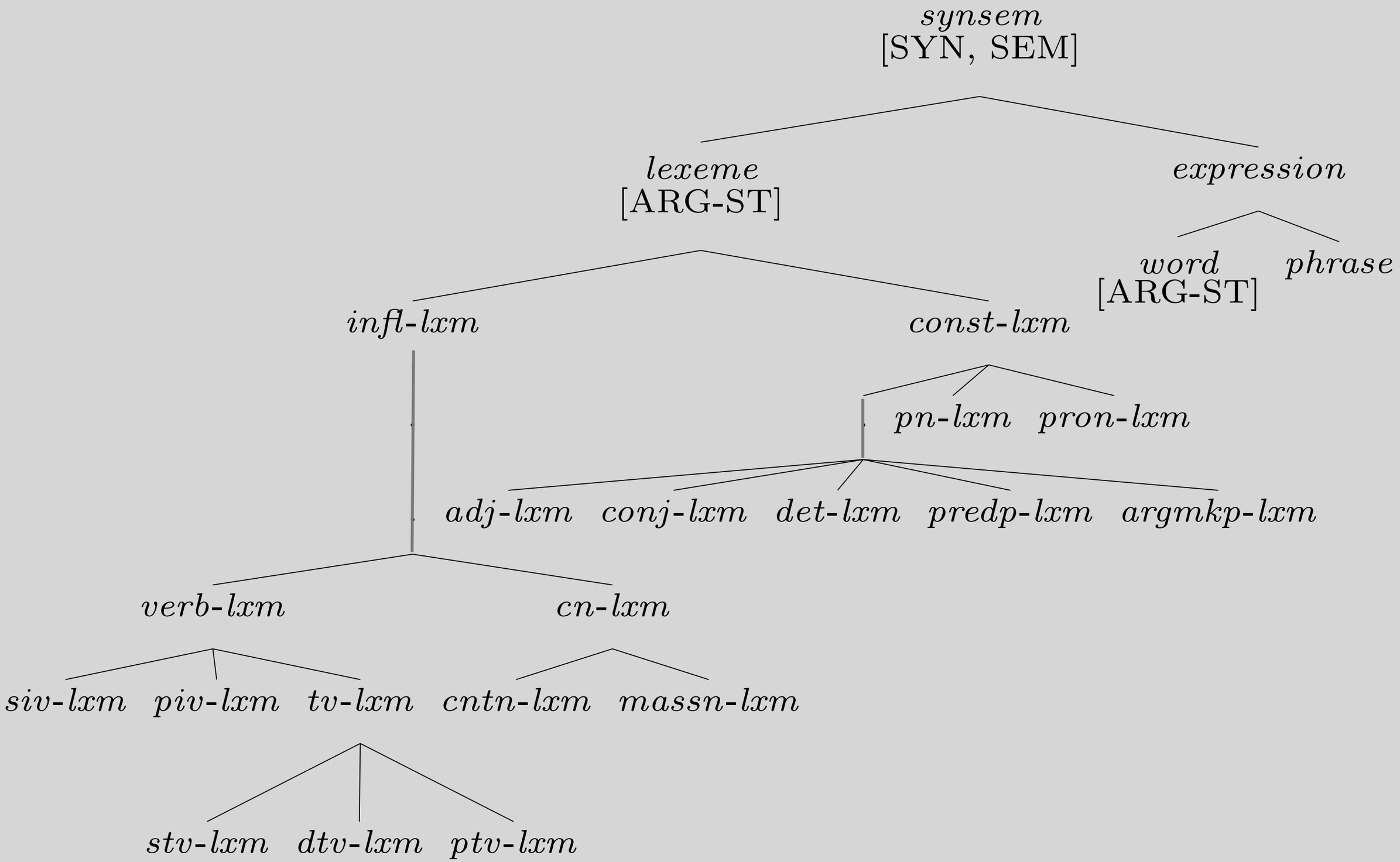
Question on Default Inheritance

Q: Can a grammar rule override a default constraint on a word?

A: No. Defaults are all ‘cached out’ in the lexicon.

- Words as used to build sentences have only inviolable constraints.

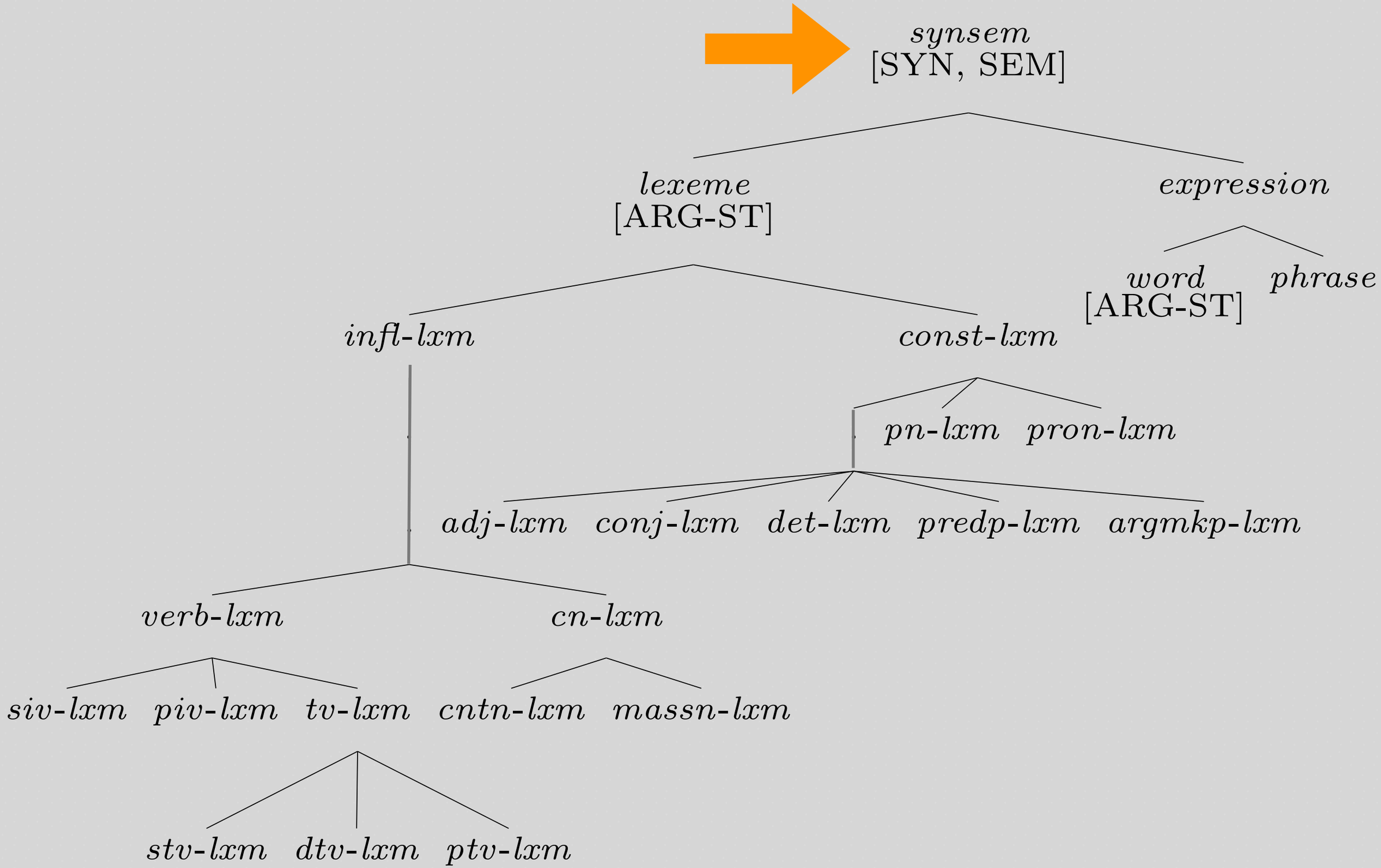
Our Lexeme Hierarchy



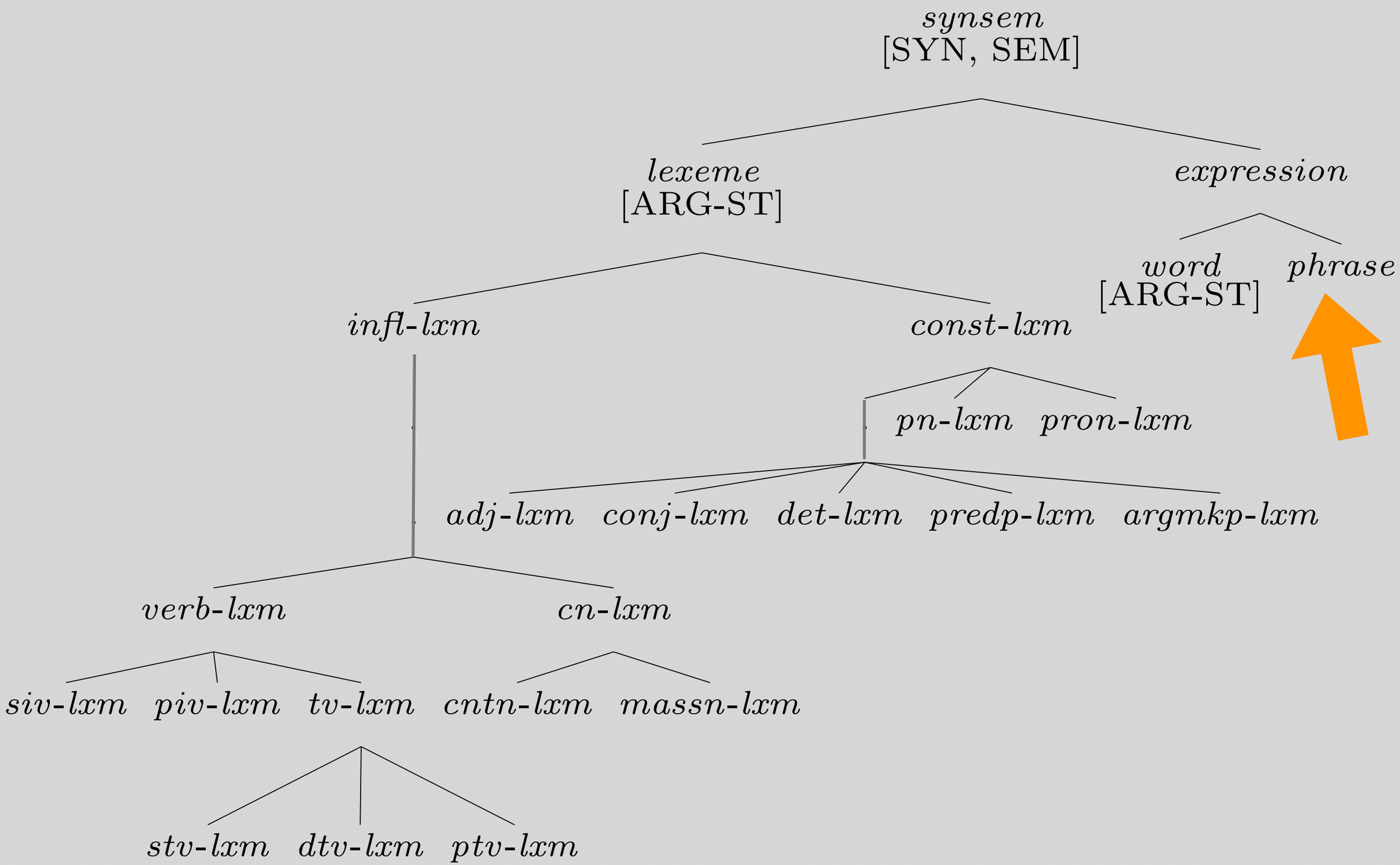
Functions of Types

- Stating what features are appropriate for what categories
- Stating generalizations
- Constraints that apply to (almost) all instances
- Generalizations about selection -- where instances of that type can appear

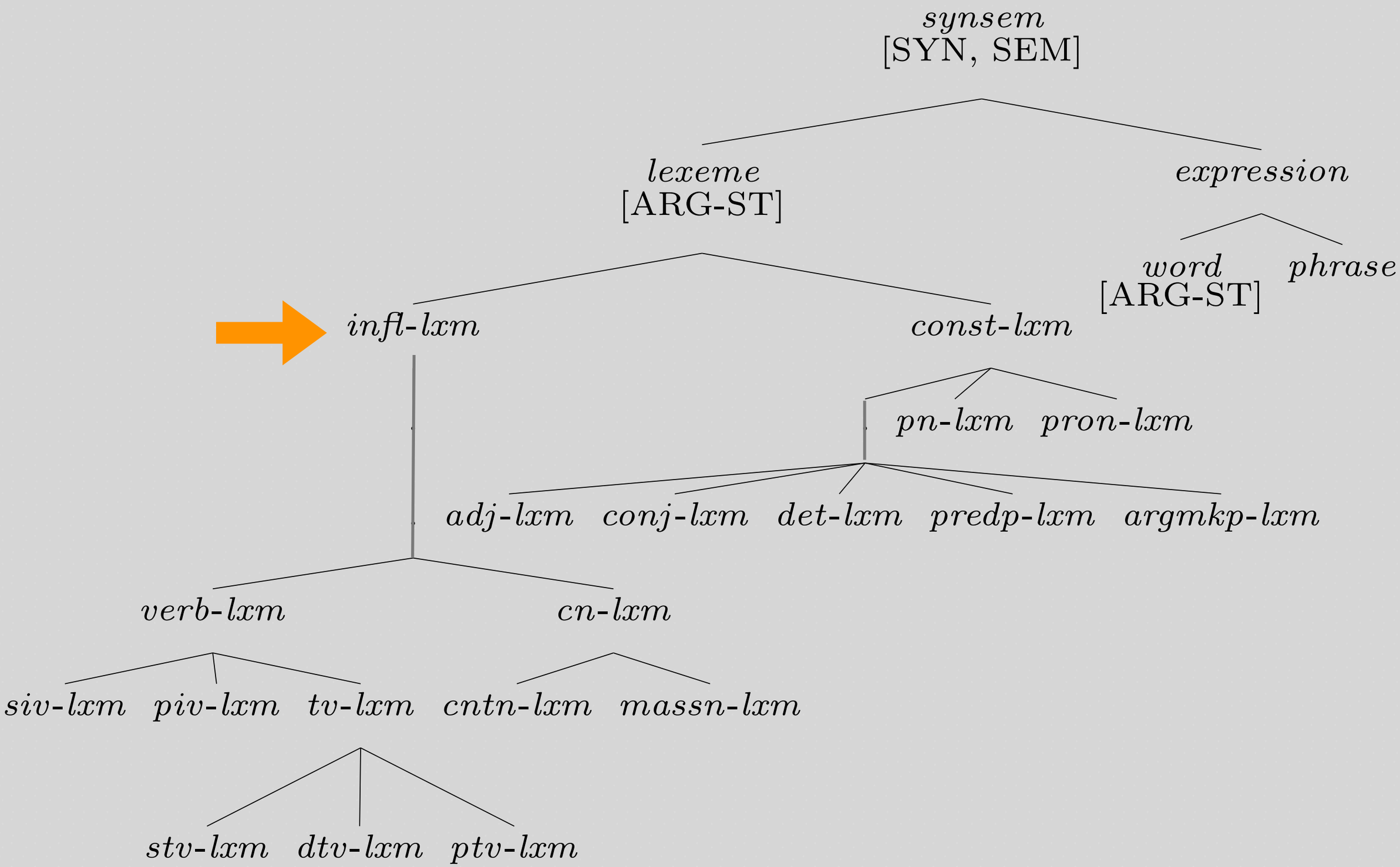
Every *synsem* has the features SYN and SEM



No ARG-ST on *phrase*



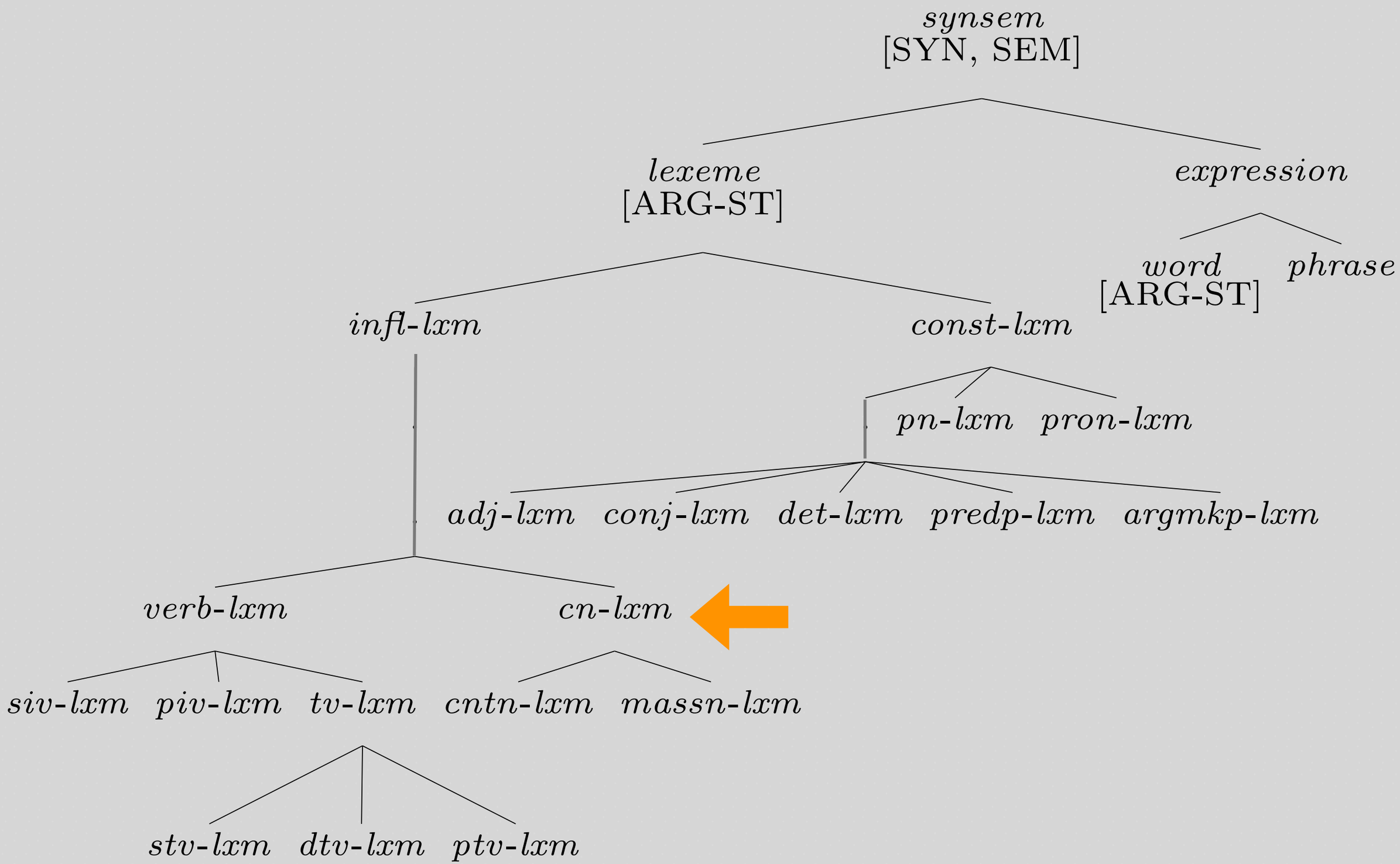
A Constraint on *infl-lxm*: the SHAC



A Constraint on *infl-lxm*: the SHAC

$$\textit{infl-lxm}: \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{VAL} & \begin{bmatrix} \text{SPR} & \left\langle [\text{AGR} \quad \boxed{1}\,] \right\rangle \end{bmatrix} \\ \text{HEAD} & [\,\text{AGR} \quad \boxed{1}\,] \end{bmatrix} \end{bmatrix}$$

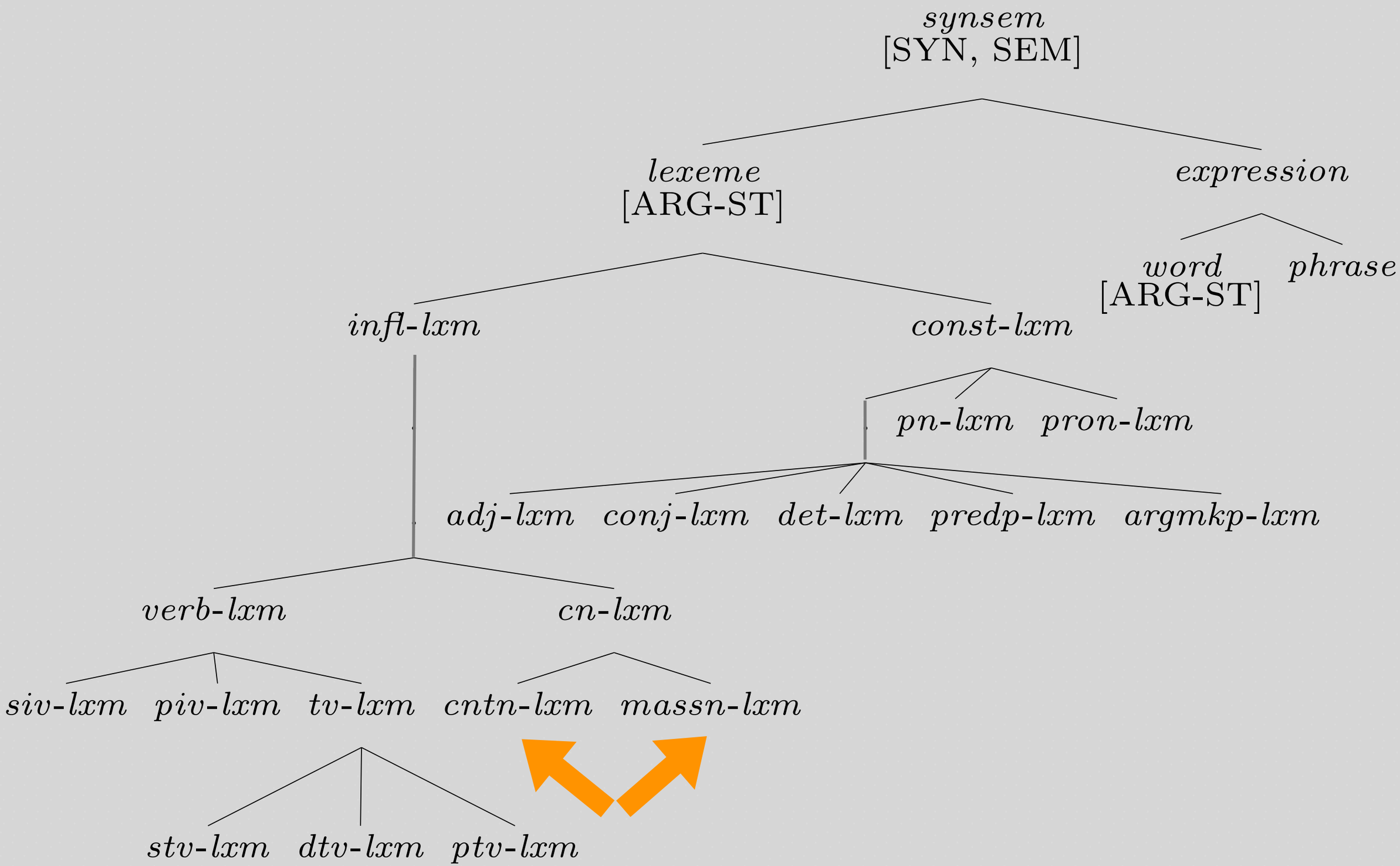
Constraints on *cn-lxm*



Constraints on *cn-lxm*

$$
\textit{cn-lxm}: \left[\begin{array}{ll}
\text{SYN} & \left[\begin{array}{ll}
\text{HEAD} & \left[\begin{array}{ll} \textit{noun} & \\ \text{AGR} & [\text{PER 3rd}] \end{array}\right] \\
\text{VAL} & \left[\text{SPR} \quad \left\langle \left[\begin{array}{ll} \text{HEAD} & \text{det} \\ \text{INDEX} & i \end{array}\right] \right\rangle\right]
\end{array}\right] \\
\text{SEM} & \left[\begin{array}{ll} \text{MODE} & / \text{ ref} \\ \text{INDEX} & i \end{array}\right] \\
\text{ARG-ST} & \langle \text{X} \rangle \oplus /\langle\ \rangle
\end{array}\right]
$$

Formally Distinguishing Count vs. Mass Nouns

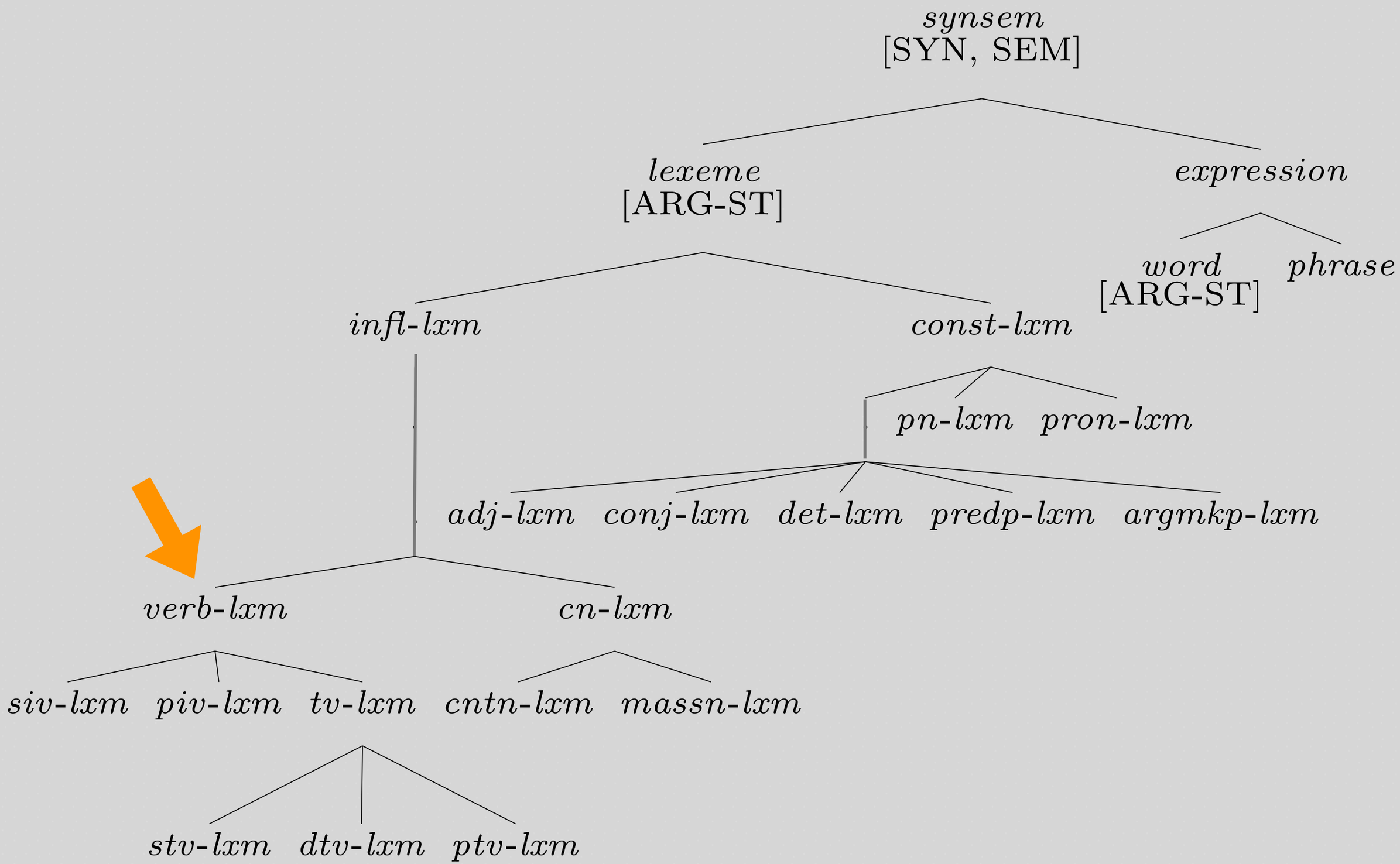


Formally Distinguishing Count vs. Mass Nouns

$$\textit{cntn-lxm} : \left[\text{SYN} \quad \left[\text{VAL} \quad \left[\text{SPR} \quad \langle\, [\text{COUNT} +] \,\rangle\right]\right]\right]$$

$$\textit{massn-lxm} : \left[\text{SYN} \quad \left[\text{VAL} \quad \left[\text{SPR} \quad \langle\, [\text{COUNT} -] \,\rangle\right]\right]\right]$$

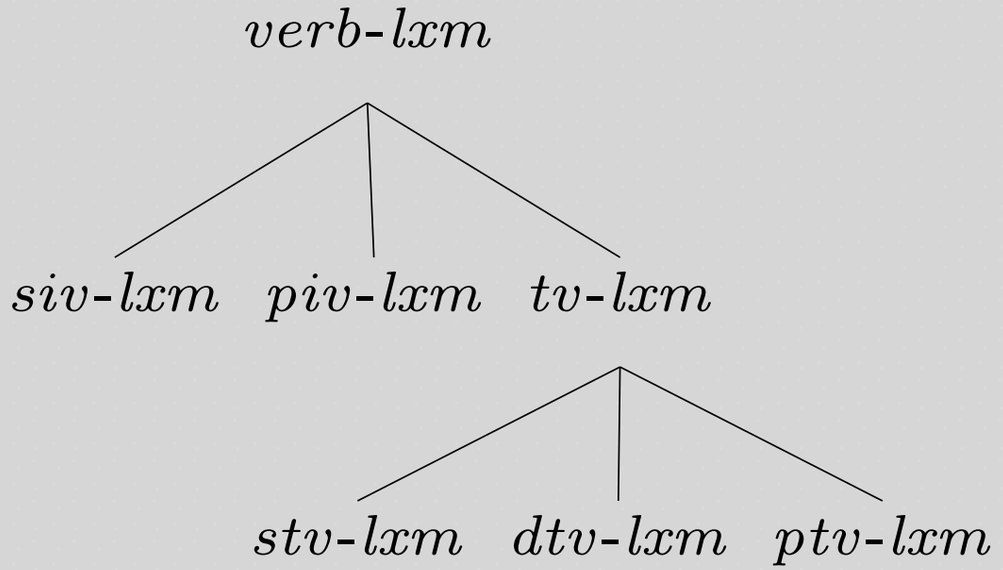
Constraints on *verb-lxm*



Constraints on *verb-lxm*

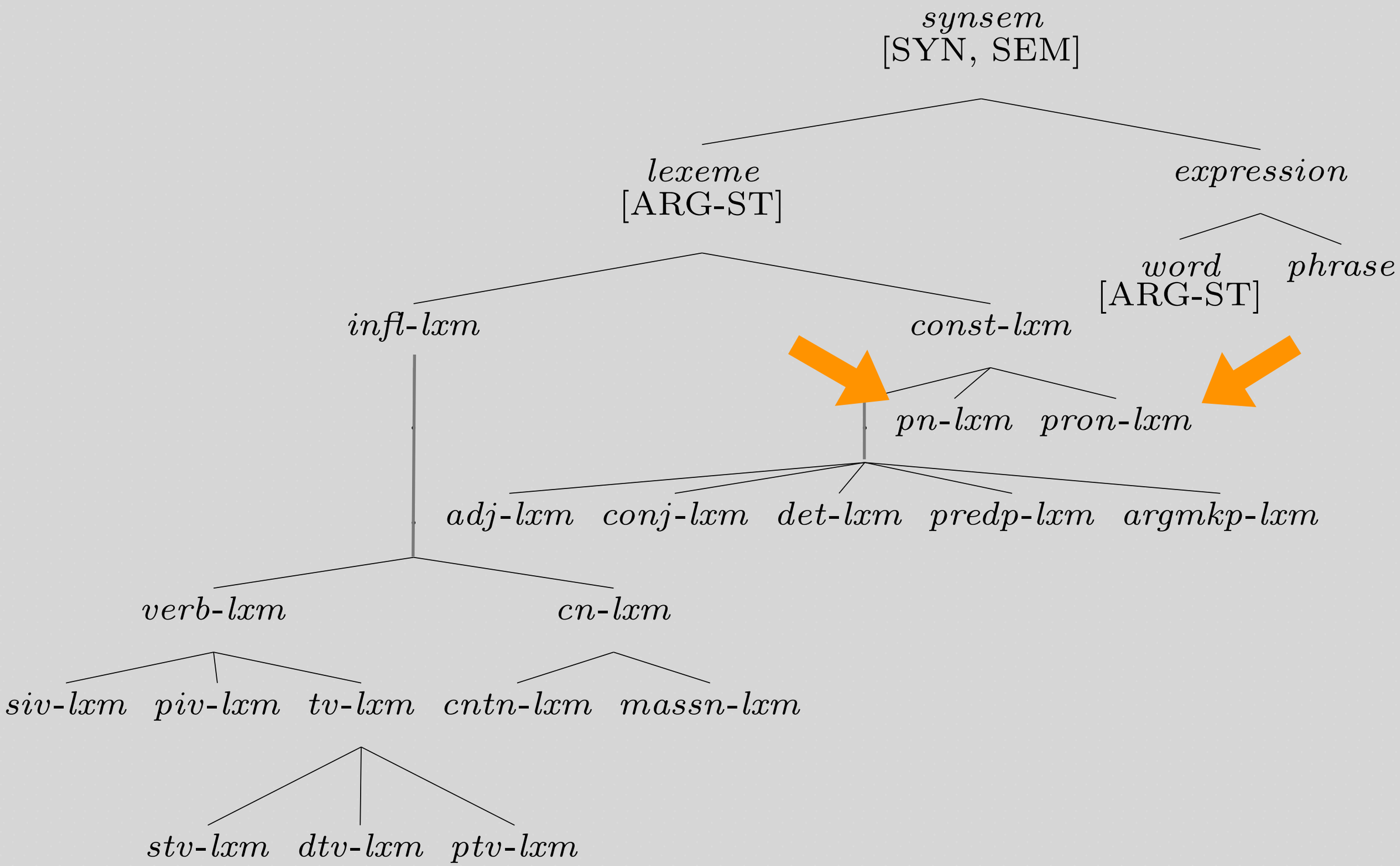
$$\textit{verb-lxm}: \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{HEAD} & \textit{verb} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{MODE} & \text{prop} \end{bmatrix} \\ \text{ARG-ST} & / \langle\ \text{NP}, \ldots\ \rangle \end{bmatrix}$$

Subtypes of *verb-lxm*



- *verb-lxm*: [ARG-ST / < NP, ... >]
  - *siv-lxm*: [ARG-ST / < NP >]
  - *piv-lxm*: [ARG-ST / < NP, PP >]
  - *tv-lxm*: [ARG-ST / < NP, NP, ... >]
    - *stv-lxm*: [ARG-ST / < NP, NP, >]
    - *dtv-lxm*: [ARG-ST / < NP, NP, NP >]
    - *ptv-lxm*: [ARG-ST / < NP, NP, PP >]

Proper Nouns and Pronouns



Proper Nouns and Pronouns

$$\textit{pn-lxm}: \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{noun} & \\ \text{AGR} & \begin{bmatrix} \text{PER} & \text{3rd} \\ \text{NUM} & \text{/ sg} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{MODE} & \text{ref} \end{bmatrix} \\ \text{ARG-ST} & \text{/ } \langle\, \rangle \end{bmatrix}$$

$$\textit{pron-lxm}: \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{HEAD} & \textit{noun} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{MODE} & \text{/ ref} \end{bmatrix} \\ \text{ARG-ST} & \langle\, \rangle \end{bmatrix}$$

The Case Constraint

An outranked NP is [CASE acc].

- object of verb ✓
- second object of verb ✓
- object of argument-marking preposition ✓
- object of predicational preposition (✓)

The Case Constraint, continued

An outranked NP is [CASE acc].

- Subjects of verbs
  - Should we add a clause to cover nominative subjects?
    - No.

      We expect them to leave. (Chapter 12)

    - Lexical rules for finite verbs will handle nominative subjects.
- Any other instances of case marking in English?
- Does it apply to case systems in other languages?

  No: The Case Constraint is an English-specific constraint.

Apparent redundancy

- Why do we need both the *pos* subhierarchy and lexeme types?
- *pos*:
  - Applies to words and phrases; models relationship between then
  - Constrains which features are appropriate (no AUX on *noun*)
- *lexeme*:
  - Generalizations about combinations of constraints

Lexical Types & Lexical Rules

- Lexemes capture the similarities among *run*, *runs*, *running*, and *run*.

- The lexical type hierarchy captures the similarities among *run*, *sleep*, and *laugh*, among those and other verbs like *devour* and *hand*, and among those and other words like *book*.

- Lexical rules capture the similarities among *runs*, *sleeps*, *devours*, *hands*,...

Overview

- Motivation for lexical hierarchy
- Default inheritance
- Tour of the lexeme hierarchy
- The Case Constraint
- *pos* vs. *lexeme*
- Reading Questions

Reading Questions

- If *verb-lxm* is specified as inviolable [MODE prop], how do we handle imperatives, which are specified as [MODE dir] and presumably fall under *verb-lxm*?

- Why is there a separate type for prepositional-intransitive-verb-lexeme? How does it differ from the strict-transitive lexeme since it takes one complement?

- What's to be done with verbs that fit into more than one lexeme subtype? Put the entry under *verb-lxm*? Multiple entries?

Reading Questions

- Why do we treat proper nouns as non-inflecting? That about *Keeping up with the Joneses*? It seems like the unifying feature of proper nouns & pronouns is that they don't take specifiers. Why not use that as the distinction between types?

- It seems odd to me to classify pronoun lexemes as type *const-lxm*. They're the only things in English that take case, after all, and unless we treat case as a derivational rule, that would seem to mean they inflect.

Reading Questions

- Also, I don't really understand the usage of X in the lexeme hierarchy in (35). If *verb-lxm* has an inviolable ARG-ST which begins with NP, why do we write all of its descendants as having ARG-STs beginning with X rather than NP?

- Why is the type constraint for *adj-lxm* written with an item called X in the SPR list, but an item called NP beginning the ARG-ST list? Shouldn't they match? (p. 244)

Reading Questions

- What are some ways to read the slash notation introduced in Chapter 8? Would "unless otherwise specified" (e.g., "the list of MOD is empty unless otherwise specified...") be applicable in this case?
- In the feature structure for *det-lxm* at the bottom of page 244, does the slash in the value of SPR just mean SPR is not always going to be an empty list for a determiner?

Reading Questions

- I am confused by example (21) on page 235. It seems to suggest that the value of: [TEL / [1]] and [CHAIR [TEL / [1]] are different. If they are distinct then why do they have the same index [1]?
- On page 245 it says " *'s* is a determiner that exceptionally takes an obligatory NP specifier.", but exactly what in the constraints on the type *det-lxm* specify that? Does this have to do with the / notation after SPR?

Reading Questions

- Why is MOD / < > specified in the feature structure for *lexeme* on page 237? Would every lexeme (or the subtypes of lexeme) be required to specify MOD < > if this wasn't part of the definition of *lexeme*? Why don't we also state COMPS / < >? It seems to me like there will be many lexemes that don't take complements, and it would be easier to have the default be an empty list.

Reading Questions

- Is there a way to indicate that something has, in fact, been overridden? I'm wondering how far down these subtype trees can go to reach the leaf level and if the path is far it may be difficult to remember what the supertype's defeasable values were. In that case it might be nice to have something to show that the leaf has significantly changed from an antecedent supertype.

Reading Questions

- Regarding choosing when to put the '/' before something--would we theoretically leave it out until we come across an examples that shows we require it? So assume that nothing is defeasible until we find a counterexample showing that it is?
- How can we determine whether a constraint is defeasible or not? Is it simply to see if there's any exception in real usage?

Reading Questions

- Are there defeasible-constraint/exception actions that preclude words from being included in a lexical class for a given language model, or could I hypothetically define, say, transitive verbs as a subtype of common nouns and then just have it override pretty much every constraint imparted by the common noun supertype? Is the only thing stopping that from happening a matter of smarter "engineering" decisions in constructing a model?

Reading Questions

- Is it ever possible for a subtype to "un-value" a feature that one of its supertypes values? e.g., if a lexical supertype defeasibly stated that COMPS needs to be empty for all its members, could a subtype make an exception and change it not to say that COMPS contains a single NP or whatever, but rather to say that the subtype doesn't impart any particular value for the COMPS list onto its members? How would we represent that?

Reading Questions

- Lexemes are useful for lexical entries but won't work in trees where the specific form of the word must be represented, right?
- Why don't we just use morphemes instead of lexemes?
- I am still unclear on the distinctions between lexeme, lexical sequence, and lexical entry. Looking forward to the discussion of them.

Reading Questions

lexical sequence Ordered pairs that can serve as the INPUT and OUTPUT values of lexical rules [q.v.] are called lexical sequences. They consist of a phonological form and a fully resolved feature structure.

lexical entry Information about individual words [q.v.] that must be stipulated is put into the lexicon [q.v.] in the form of descriptions that we call lexical entries. They are ordered pairs, consisting of a phonological form (description) and a partial feature structure description. Fully resolved lexical sequences [q.v.] consistent with lexical entries can serve as the INPUT values of lexical rules [q.v.].

lexeme The term 'word' is used ambiguously to mean either a particular form, such as *sees*, or a set of related forms such as *see*, *sees*, *saw*, *seen*, and *seeing*. To avoid this ambiguity, linguists sometimes posit an abstract entity called a 'lexeme' that gives rise to a family of related words. *See also* word.

Reading Questions

- Assuming that the lexical entry is what gets "stored" in the lexicon -- say, encoded in the neurons, just for concreteness -- does the type of the lexeme get stored with it? For example the lexical entry for devour is of type word, but does the type stv-lxm form part of its lexical entry somewhere as well?

- The descriptions of lexemes feels a bit like inheritance in OOP. Is it possible for a lexeme to have multiple inheritance? Is there any time this might be useful?

Reading Questions

- Do we still need lexemes/a lexical type hierarchy in isolating languages?
- Where do the lexical rules and the lexeme types belong, to the grammar rules or to the lexicon?

Reading Questions

- Are there applications where it helps to label text with lexeme classes before processing it, similar to the way POS tags are often useful features?
- Are fronted arguments still outranked by the same things? *On the table, I put the book.*
- I'd like some clarification on argument making and predicational prepositions. Is the former equivalent with PP as complements and the latter PP as modifiers?