

Constraint Language for Lambda Structures (CLLS): intuition 1

- CLLS (Egg, Koller, and Niehren 2001) allows an underspecified representation of structural ambiguities, e.g., with scope, ellipsis, and anaphora
- formally, it is a language for the (partial) description of **trees**
- underlying observation:
 - it is not that straightforward to give an underspecified representation of λ -terms
 - but this is rather straightforward for tree structures
 - i.e., representing λ -terms as tree structures opens up an **easy way of representing them in an underspecified way**



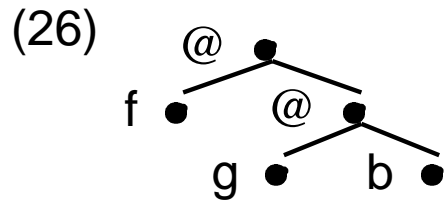
CLLS: intuition 2

- λ -terms are modelled as an extension of a tree structure, viz., a **lambda structure**
- λ -structures are node-labelled trees enriched by additional edges to model λ -binding
- CLLS expressions obtain partial descriptions of λ -terms by partial descriptions of λ -structures
- partial descriptions model **ambiguity**



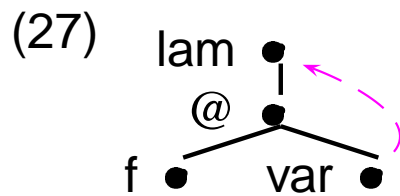
CLLS: intuition 3

- step 1: represent λ -terms as trees, e.g., $f(g(b))$:



- functional application is represented by a node labelled '@'

- step 2: represent λ -terms as trees decorated with λ -links, the so-called **λ -structures**, e.g., $\lambda x.f(x)$:



- λ -abstraction is represented by a λ -link between a var and a lam node

CLLS: example 1

(28) Every linguist attends a workshop

$\exists x.\mathbf{workshop}'(x) \wedge$

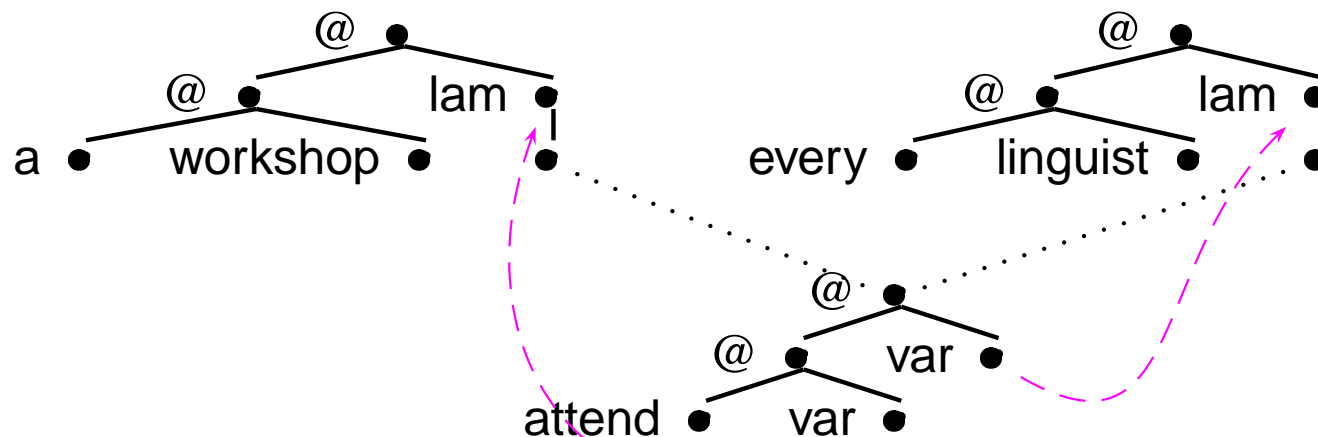
$\forall y.\mathbf{linguist}'(y) \rightarrow$

(29) (a) $\forall y.\mathbf{linguist}'(y) \rightarrow$
 $\mathbf{attend}'(y, x)$

(b) $\exists x.\mathbf{workshop}'(x) \wedge$
 $\mathbf{attend}'(y, x)$

- description of the $\hat{\lambda}$ -structures for these λ -terms in a single CLLS constraint:

(30)



- here 'a' abbreviates $\lambda Q\lambda P\exists x.Q(x) \wedge P(x)$, similarly for 'every'



CLLS: example 2

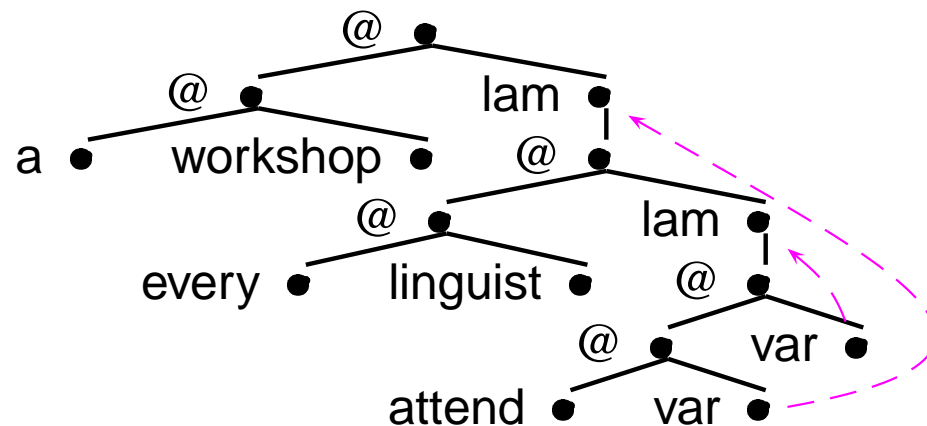
- the scope of the quantifier fragments in (30) is unspecified
- but, since λ -structures are trees, one fragment must eventually dominate the other (no upward branching)
- partial descriptions use **dominance relations** between nodes (dotted lines)
- such graphs have a formal meaning as **constraints over λ -structures**
- distinguish graphs for λ -structures from graphs for constraints on λ -structures
 - only the latter may incorporate dominance constraints
 - the latter describe **node variables**, the former **nodes**



CLLS: example 3

- (30) is satisfied by all λ -structures into which the graph can be embedded without overlap
- e.g., (31), which models (29a), the λ -term for one of the two readings of (28):

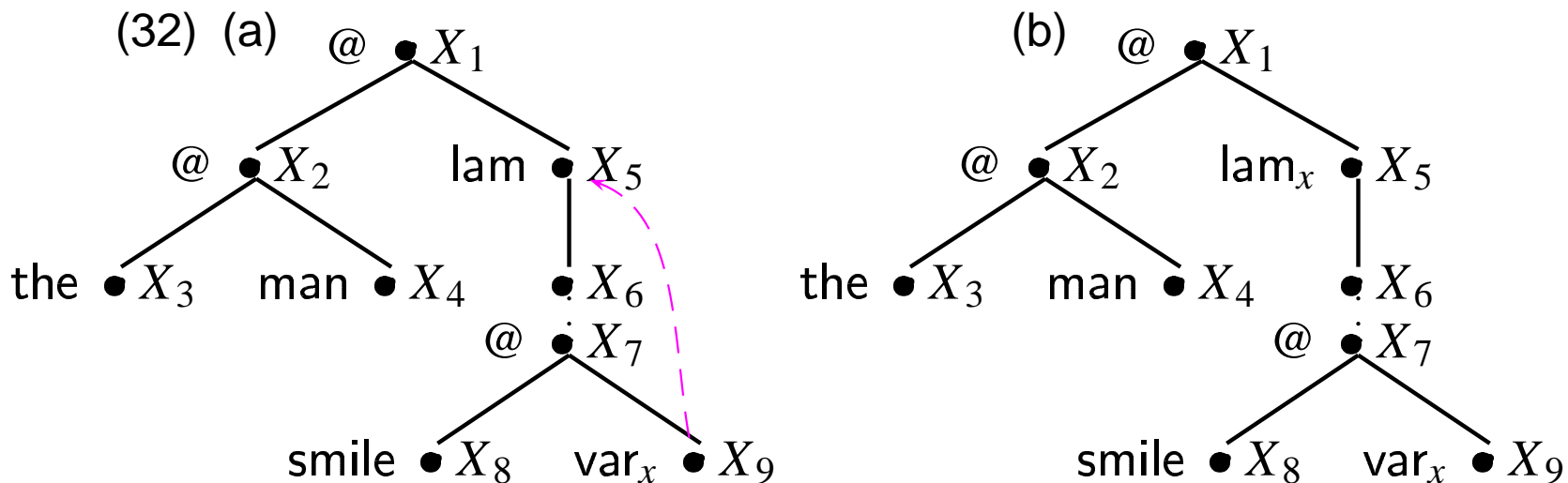
(31)



- this solution is **constructive** in that it only comprises material explicitly mentioned in the constraint

CLLS: the capturing problem 1

- representing λ -abstraction by a λ -link between a var and a lam node avoids the capturing problem
- compare λ -links to an alternative representation that uses explicitly annotated nodes for variables and their binders:



(33) the man smiles



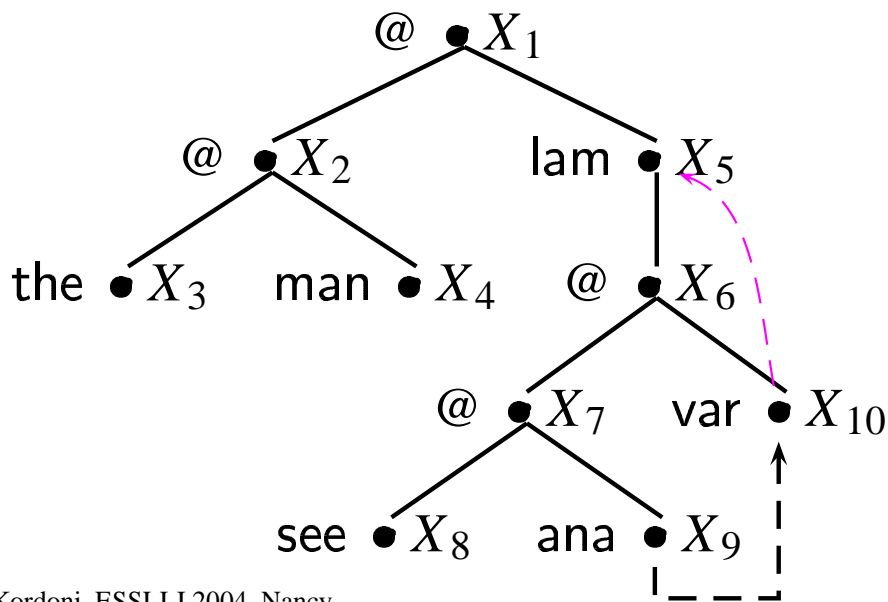
CLLS: the capturing problem 2

- in (32b), additional material intervening between X_6 and X_7 could comprise a binder for X_9 (another node labelled by lam_x)
- no such thing can happen in (32a), due to the functional nature of $\hat{\lambda}$
 - there can be only one binder node for a variable node
 - this relation cannot be cancelled or replaced by material that comes in between X_6 and X_7
- $\hat{\lambda}$ -links resemble ‘rubber bands’ that can be stretched by intervening material without breaking
- CLLS handles the capturing problem without having to resort to explicit bookkeeping devices for variables



CLLS: anaphoric binding

- anaphoric binding is covered in CLLS, too
 - an anaphor results in a node labelled *ana*
 - its binder is indicated by an ante-edge from the anaphor node to the antecedent node
- example: *the man sees himself*



CLLS: formal definition

- CLLS constraints are a conjunction of atomic literals:

$\varphi ::= X:f(X_1, \dots, X_n) \mid X \triangleleft^* Y \mid \lambda(X) = Y \mid \text{ante}(X) = Y \mid \varphi \wedge \varphi'$

- X, Y , etc. are variables which denote nodes in a lambda structure
- parallelism is not yet included here
- they are satisfied by a lambda structure and a variable assignment iff all their literals are satisfied:
 - a **labelling literal** $X:f(X_1, \dots, X_n)$ is satisfied iff X denotes a node with label f and children that are denoted by X_1, \dots, X_n
 - a **dominance literal** $X \triangleleft^* Y$ is satisfied iff X denotes a (reflexive, transitive) ancestor of Y in the lambda structure
 - **binding literals** $\lambda(X) = Y$ and $\text{ante}(X) = Y$ are satisfied iff the lambda structure contains corresponding lambda or anaphoric binding edges



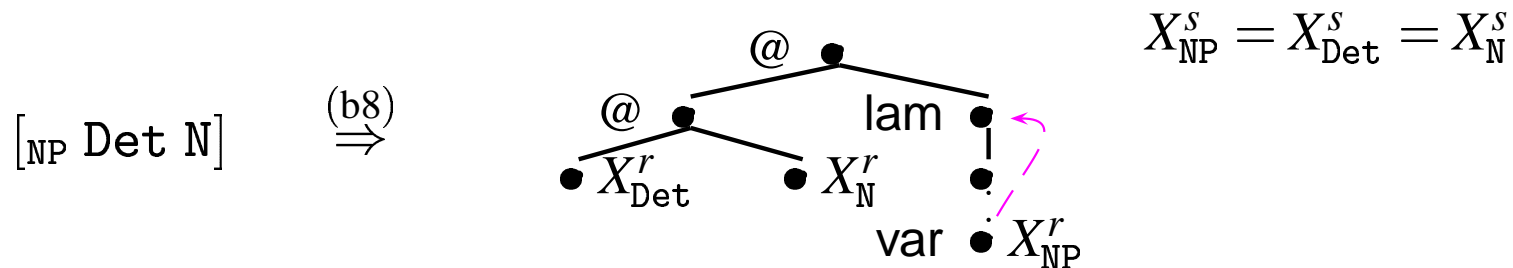
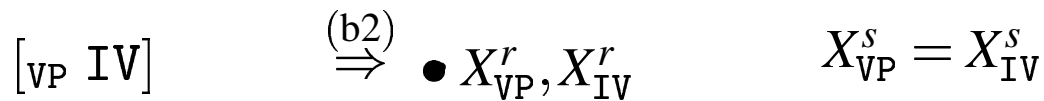
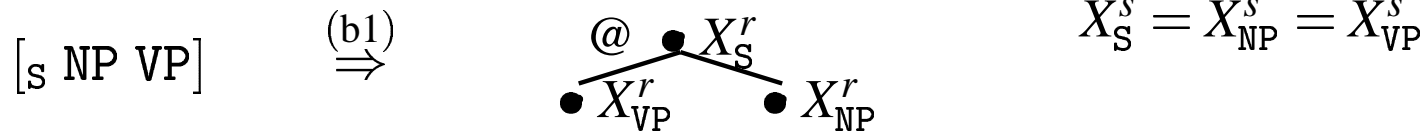
CLLS: semantic construction 1

- most of the constraint construction is **factored out** into the interface
- basic lexical entries contribute just a labelled node variable in their semantics
- constituents inherit the constraints of their immediate constituents
- syntax-semantics interface rules '**glue them together**' by identifying specific node variables
- two node variables are made visible for each constituent C
 - its **topnode** X_C^r (the root node of the respective constraint)
 - its **scopenode** X_C^s (similar to the global top, models scope islands)



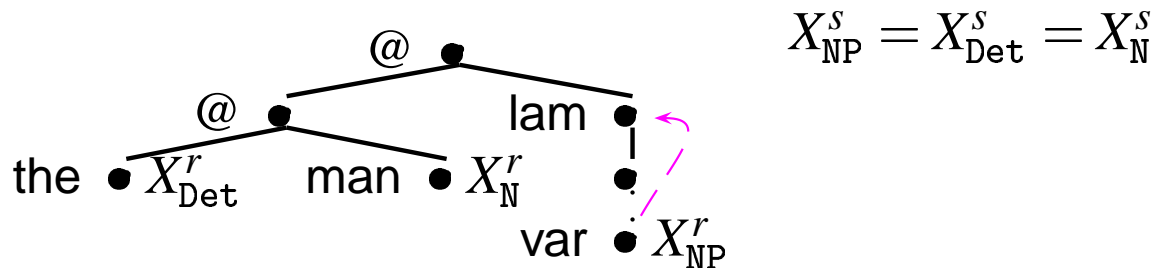
CLLS: semantic construction 2

- sample rules of the syntax-semantics interface

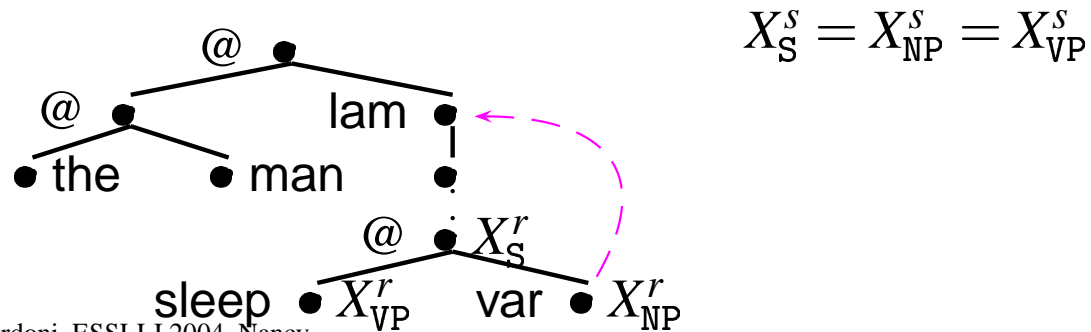


CLLS: semantic construction 3

- example: *the man sleeps*
- the lexical entries for the words are labelled nodes, e.g., for *sleeps*: sleep • X_1
- rule **(b8)** constructs the semantics of the NP *the man*:

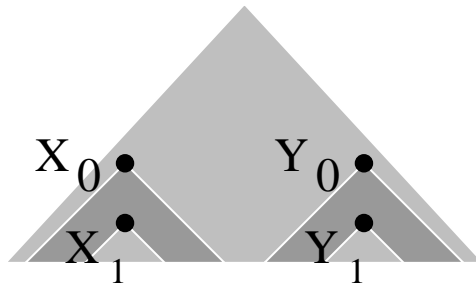


- due to rule **(b2)**, the semantics of the VP *sleeps* is inherited from its head
- rule **(b8)** constructs the semantics of the whole sentence



Parallelism in CLLS 1

- **parallelism literals** state that two pieces of a λ -structure have the same structure
- $X_0/X_1 \sim Y_0/Y_1$ says that ‘ X_0 upto X_1 ’ looks the same as ‘ Y_0 upto Y_1 ’



(34) parallelism

- X_0 and X_1 designate a tree with a hole (a **segment**), and so do Y_0/Y_1
- the parallelism literal states that the segments have the same structure

Parallelism in CLLS 2

- parallelism literals can be used to model ellipsis

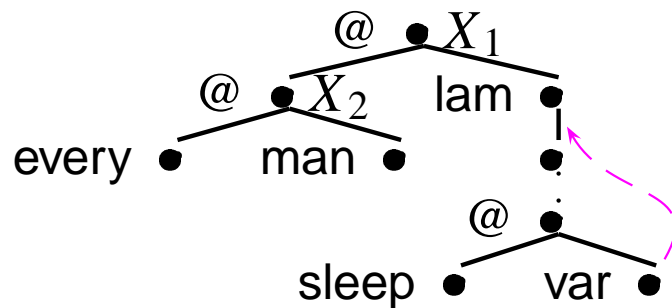
(35) Every man slept, and Mary did, too

- the meaning of the TS *so does Mary* is almost equal to the one of the SS *every man slept*
- the difference is that the semantic contribution of the source parallel element *every man* is replaced by the one of the target parallel element *Mary*

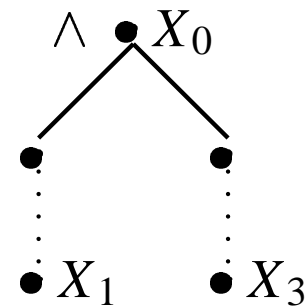


Parallelism in CLLS 3

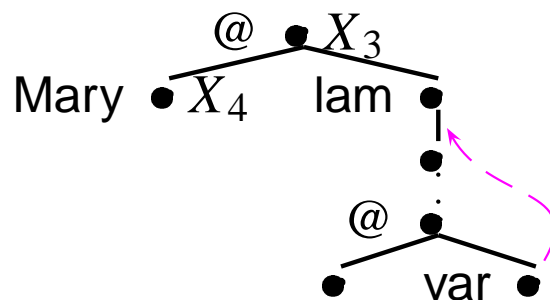
semantics of the SS:



semantics of the conjunction:



semantics of the TS:



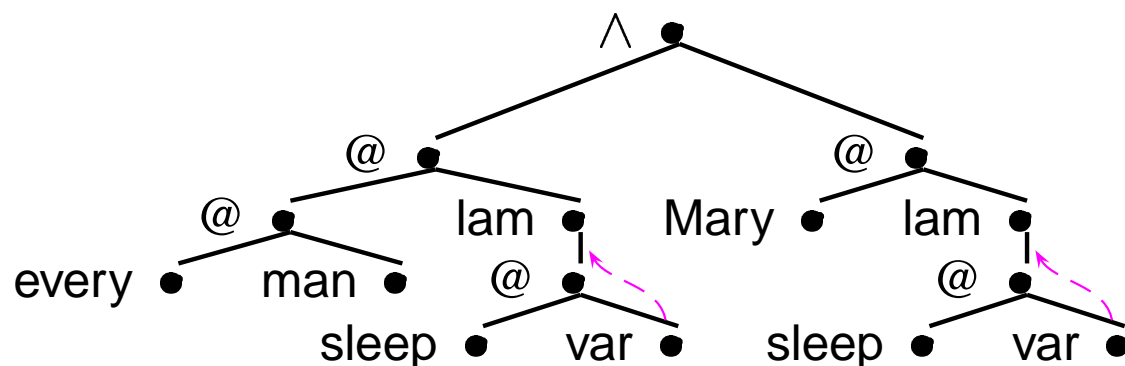
the parallelism literal:

$$X_1/X_2 \sim X_3/X_4$$

Parallelism in CLLS 4

- the parallelism literal states that the SS semantics tree has almost the same structure as the TS semantics tree
- the exceptions are the two subtrees for the semantics of the SS subject and of the TS subject
- the λ -structure (36) is the intended semantic representation of (35)

(36)



- it satisfies all four constraints
- for semantic construction of ellipsis examples, see Egg and Erk (2002)



Parallelism in CLLS 5

- CLLS can also handle the interaction of ellipsis and scope, as in (37)

(37) Every linguist attends a workshop. Every computer scientist does, too

- it also covers the interaction of ellipsis and anaphora that arise from ‘strict’ and ‘sloppy’ reconstructions of pronouns:

(38) Max_{*i*} loves his_{*i*} wife and Bill does, too

