Constraint Language for LRS

Gerald Penn

Goethe-Uni. Oberseminar
23rd January, 2023

Semantic Underspecification in HPSG

- Historically, HPSG's answer to semantics has been to encode the apparatus of well-formedness constraints alongside grammar:
  - Cooper storage: QSTORE [Pollard & Sag, 1994]
    - Massively overgenerated
    - Many other theoretical objections [Pollard & Yoo, 1998]
    - FS representation amounted to algorithmic storage devices, augmented with "principles"
    - Situation semantics in place of semantic typing, e.g. *e* -> *t*.

Semantic Underspecification in HPSG

- Historically, HPSG's answer to semantics has been to encode the apparatus of well-formedness constraints alongside grammar:
  - Cooper storage: QSTORE [Pollard & Sag, 1994]
  - Minimal Recursion Semantics: MRS [Egg, 1998; ERG]
    - First to separate underspecificed semantic object (logical term) from syntactic tree
    - Does not commit to the "real semantics" – more of a front end
    - Also first to think of this in terms of semantic *descriptions* subject to a fixed, extra-grammatical resolution procedure (acyclicity, free variables, etc.)
    - No true semantic typing.

Semantic Underspecification in HPSG

- Historically, HPSG's answer to semantics has been to encode the apparatus of well-formedness constraints alongside grammar:
  - Cooper storage: QSTORE [Pollard & Sag, 1994]
  - Minimal Recursion Semantics: MRS [Egg, 1998; ERG]
  - Underspecified Hole Semantics: Ty2U [Richter & Sailer, 1999]
    - First to attempt a "real semantics"
    - But used FSs to explicitly represent dominance [Muskens, 1995]
    - Semantic typing (hooray!) – but encoded in feature logic
    - But assumed description level and existence of extra-grammatical resolution.

Semantic Underspecification in HPSG

- Historically, HPSG's answer to semantics has been to encode the apparatus of well-formedness constraints alongside grammar:
    - Cooper storage: QSTORE [Pollard & Sag, 1994]
    - Minimal Recursion Semantics: MRS [Egg, 1998; ERG]
    - Underspecified Hole Semantics: Ty2U [Richter & Sailer, 1999]
    - Constraint Language for Lambda Structures: CLLS [Egg & Erk, 2001]
        - Has a "real semantics," but not your semantics
        - Traditional λ-calculus: beta-reduction instead of underspecification
        - No universal principles whatsoever – even so-called CLLS for HPSG [Egg &Erk, 2002] was just phrase-structure rules with FSs
        - No semantic typing, but somehow seemed to feel guilty about not having it.

Lexical Resource Semantics (LRS): why bother?

- All semantic idiosyncrasies are lexical, as a matter of design
- True semantic typing – no excuses
- In other respects, a different combination of choices on issues already under discussion:
  - (+CLLS, +Ty2U) real semantics
  - (+MRS, +Ty2U) descriptions, which can be underspecific; no beta-reduction; constraint-based
  - (+MRS) formerly tried using FSs, but not anymore
  - (-MRS) antecedents of constraints also have access to semantics; semantic description language itself is richer in other respects
- But this particular combination presents both challenges and opportunities:
  - Extra-grammatical resolution is a Good Thing: doesn't clutter up the grammar for linguists; computational linguists own the resolution procedure
  - Lots more potential there than just acyclicity – compatible with performing semantic inference:
    - E.g., $\forall x.[\mathrm{horse}(x) \rightarrow \mathrm{run}(x)],\ \mathrm{horse}(\textsc{SeaBiscuit}) \vdash \mathrm{run}(\textsc{SeaBiscuit})$
  - But even with just the basics, LRS's use of variables not even known to be PTIME
  - For the rest of this talk, "Resolution" = tractably eliminating superfluous underspecification
  - "Real semantics" means that underspecification is a matter of convenience

Assumptions

- We assume that semantic composition is built on a tripartite distinction among:
  - Internal content: nuclear contribution of semantic head
  - External content: contribution of semantic head's maximal syntactic projection
  - Semantic contribution: of pieces of Ty2 terms by component signs.
- LRS assumes more than this, and so the language I am describing (CLLRS) could encode a wider range of theories than LRS.

The CLLRS model

- We augment the models that come with standard feature logic:
  - @sem: $U \rightharpoonup \mathcal{P}(Ty2)$
  - {pivot}: $U \rightharpoonup \mathcal{P}(Ty2)$
  - ^root: $U \rightharpoonup \mathcal{P}(Ty2)$
  - /contributor: $Ty2 \rightharpoonup \mathcal{P}(U)$
- No extra expressive power here:
  - Logical-form semantics: we model Ty2 terms, not what Ty2 models
  - This could still be encoded in typed feature logic (and it is definitely not worth doing so).

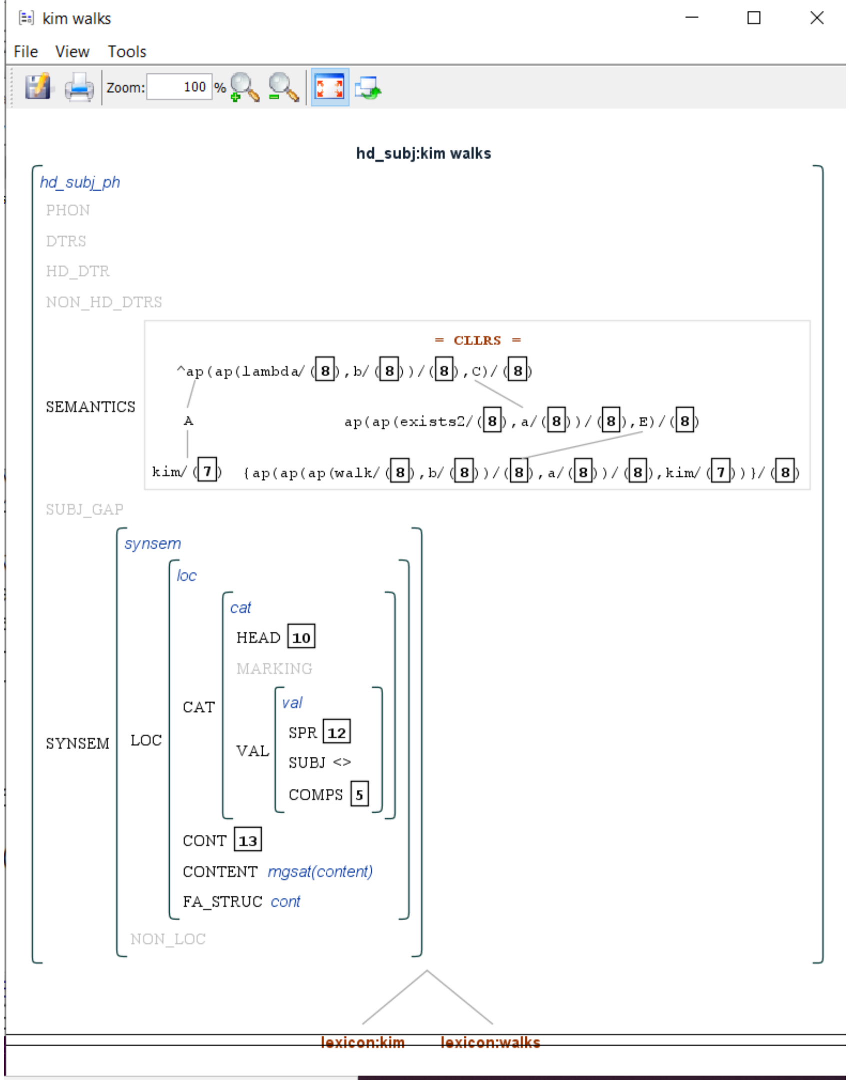
The CLLRS model

- We augment the standard type system for feature logic with functional types and polymorphic variables, but only for semantic terms:
  - ```
    semtype [t,f]: t.
    semtype [student,book]: (s->e->t).
    semtype read: (s->e->e->t).
    semtype [every,indefinite,exists]: (e->t->t->t).
    semtype w: var(s).
    semtype q: var(e->t->t->t).
    semtype [a,e,x,y,z]: var(e).
    semtype lambda: (var(A)->B->(A->B)).
    ```
- And, oh yeah, immediate/improper subterms.
    ```

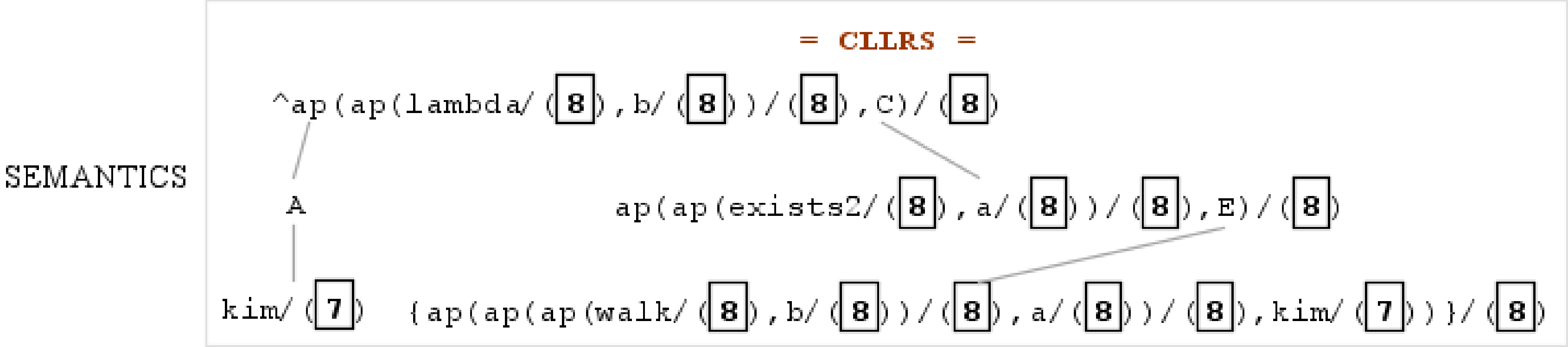
CLLRS constraint algebra

- literal/type consistency:
```
literal(N,Lit1,Type1) \ literal(N,Lit2,Type2)
                       <=> Lit1=Lit2, Type1=Type2.
```
- pivot and root are functions:
```
pivot(X,N) \ pivot(X,M) <=> N=M.
root(X,N) \ root(X,M) <=> N=M.
```
- immediate subterm irreflexivity:
```
ist(N,N,_) ==> fail.
```
- immediate subterm uniqueness:
```
ist(M1,N,A) \ ist(M2,N,A) <=> M1=M2.
```
- subterm anti-symmetry:
```
st(M,N), st(N,M) <=> M=N.
```
- Note the lack of arity consistency: applications are explicit parts of term trees.

This is what grammar gets you:

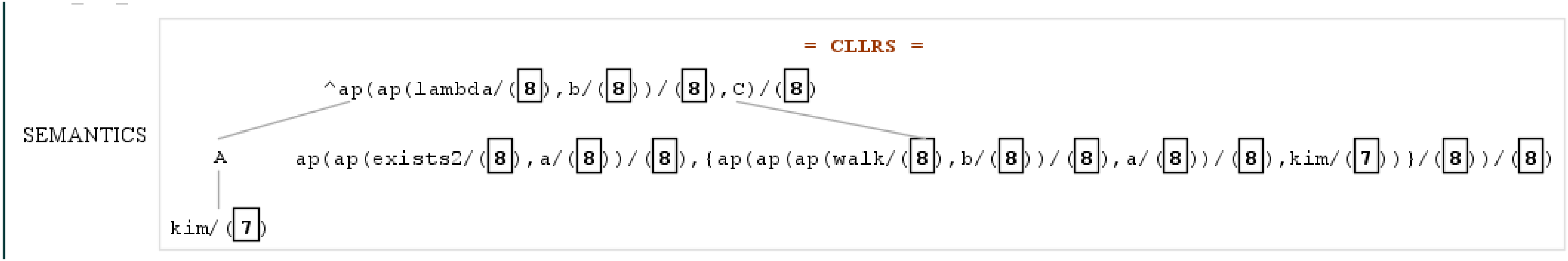


0) This is what we need to resolve:



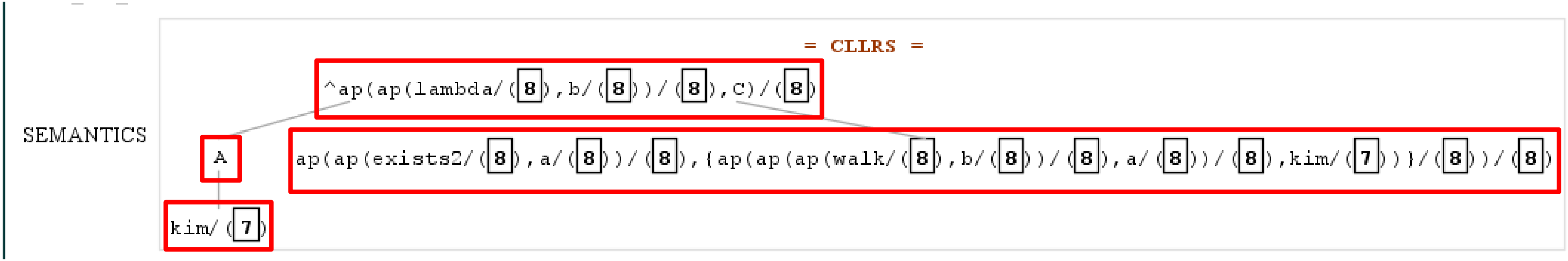
(Clearly, we also have a GUI problem)

1) Use the type system:



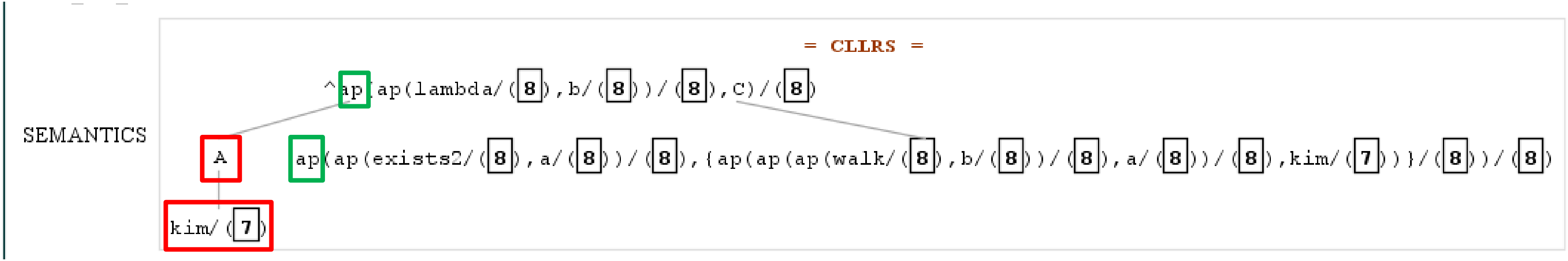
Out of 153 rejected candidate pairs for combination, 124 pairs (81%) were determined because of the type system; 26 (17%) because of cyclicity.

2) Identify “modules:”



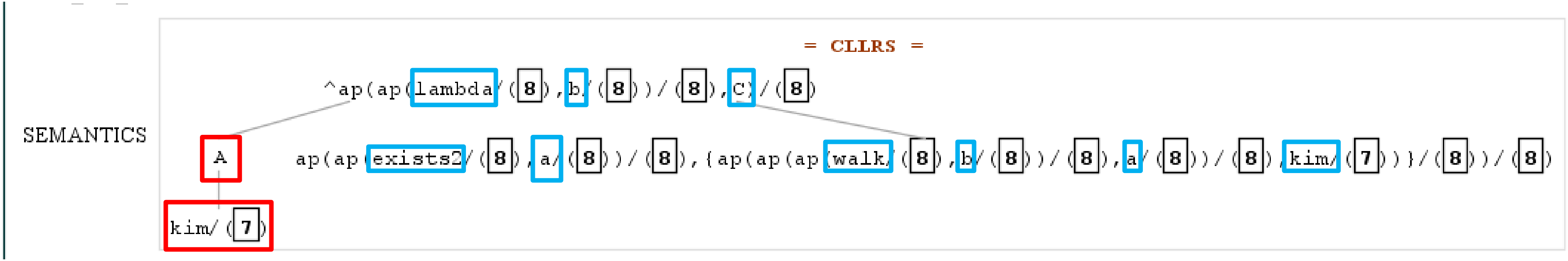
These are characterized by subgraphs connected by immediate subterm edges and no improper subterm edges.

3) Identify “tops” of modules:



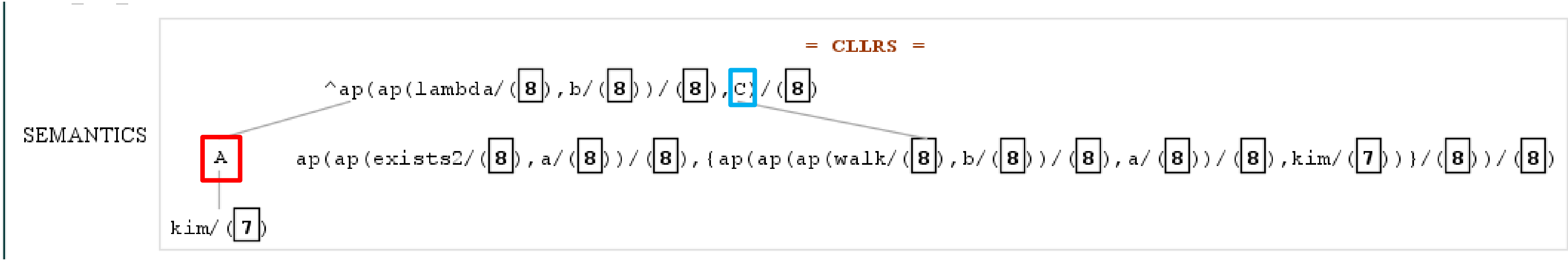
Tops are the term itself and the RHS of any improper subterm edge.

4) Identify "bottoms" of modules:



Bottoms are located by propagating downward from every top.

5) Non-determinately pick a top and bottom:



5) …and speculatively combine them:

SEMANTICS

= CLLRS =

^ap(ap(lambda/($\boxed{8}$),b/($\boxed{8}$))/($\boxed{8}$),$\boxed{\text{B}}$)/($\boxed{8}$)

ap(ap(exists2/($\boxed{8}$),a/($\boxed{8}$))/($\boxed{8}$),{ap(ap(ap(walk/($\boxed{8}$),b/($\boxed{8}$))/($\boxed{8}$),a/($\boxed{8}$))/($\boxed{8}$),kim/($\boxed{7}$))}/($\boxed{8}$))/($\boxed{8}$)

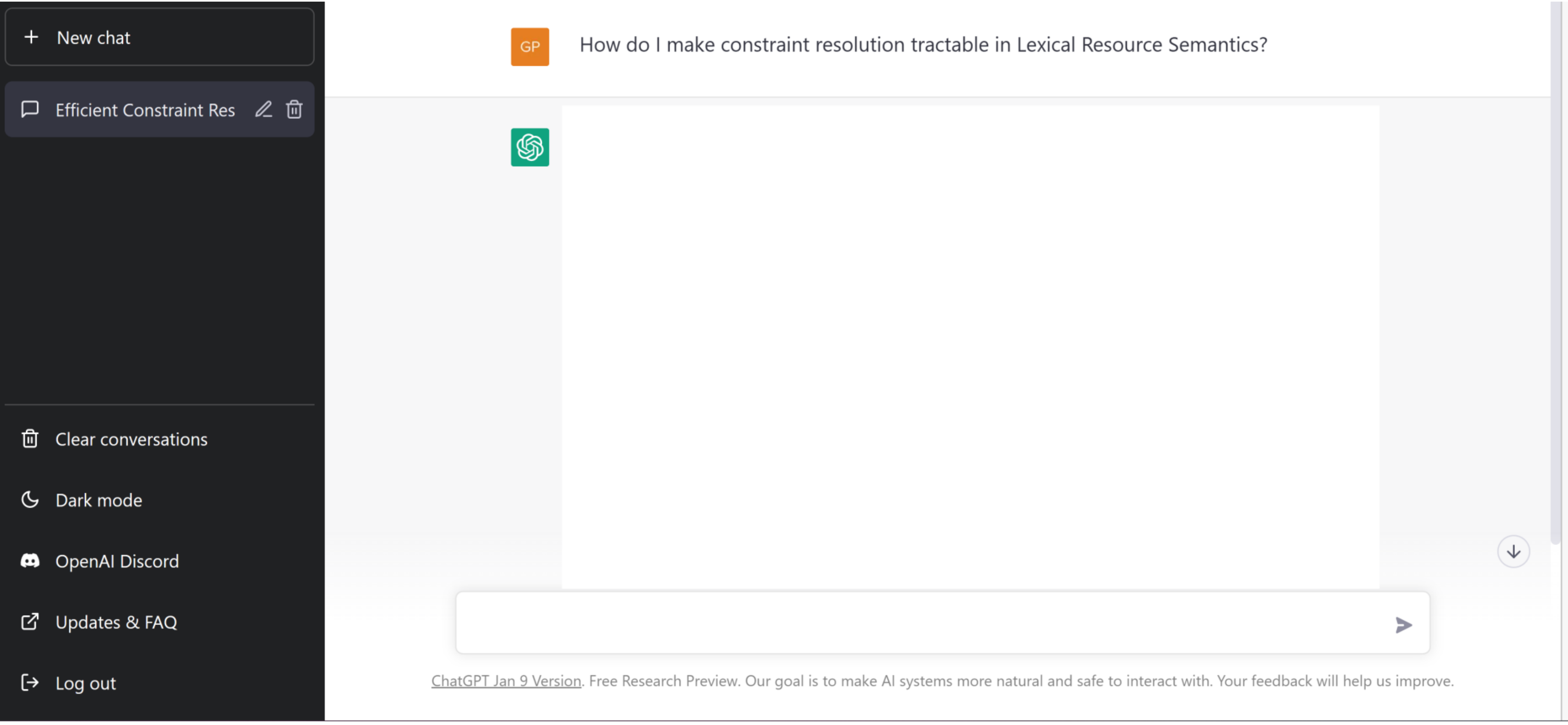
5) Keep doing this step:

SEMANTICS	= CLLRS = ^ap(ap(lambda/(8),b/(8))/(8),ap(ap(exists2/(8),a/(8))/(8),{ap(ap(ap(walk/(8),b/(8))/(8),a/(8))/(8),kim/(7))}/(8))/(8))/(8)

Intuition behind this strategy

- There are fewer modules than there are subterms
- With top-bottom combinations, the number of modules strictly decreases
- Improper subterm edges are never added by inference – these are primitive constraints added directly by grammarians, and so the grammarian has control
- In the background, we are still doing type-driven subterm combination, but only when the result is determinate
- There is a possibility of generating many answers through speculation, but we can count them before displaying them, and they can be undone
- There are nevertheless very simple grammars for which known determinate-only solving does not yield a completely resolved semantic term – unacceptable
- In the meantime, documenting which sorts of speculation prove to be fruitful (and determinate) is an important source of information for:
  - finding new determinate constraint-solving strategies, and
  - statistical inference that can guess what will work best (shortest path to resolution).

What should we do now?



What should we do now? ChatGPT knows…