

Constraint Language for LRS without Constraints or LRS

Gerald Penn

Goethe-Uni. Oberseminar
3rd June, 2024

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. $e \rightarrow t$.

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 underspecified 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 [Muskins, 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)

- 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., $\forall x.[\text{horse}(x) \rightarrow \text{run}(x)], \text{horse}(\text{SEABISCUIT}) \vdash \text{run}(\text{SEABISCUIT})$
 - 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

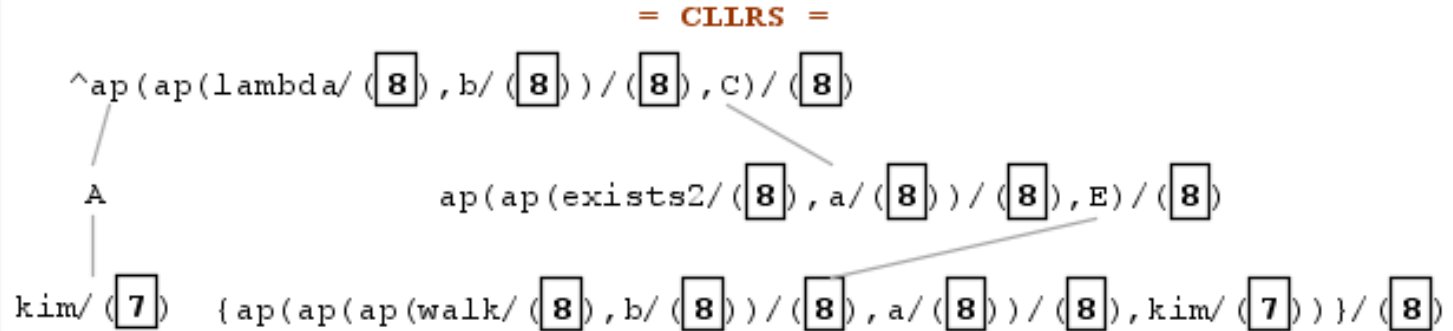
- We augment the models that come with standard feature logic:
 - @sem: $U \rightarrow P(Ty2)$
 - {pivot}: $U \rightarrow P(Ty2)$
 - ^root: $U \rightarrow P(Ty2)$
 - /contributor: $Ty2 \rightarrow P(U)$
- No extra expressive power here:
 - Logical-form semantics: we model Ty2 terms, not what Ty2 models
 - 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.

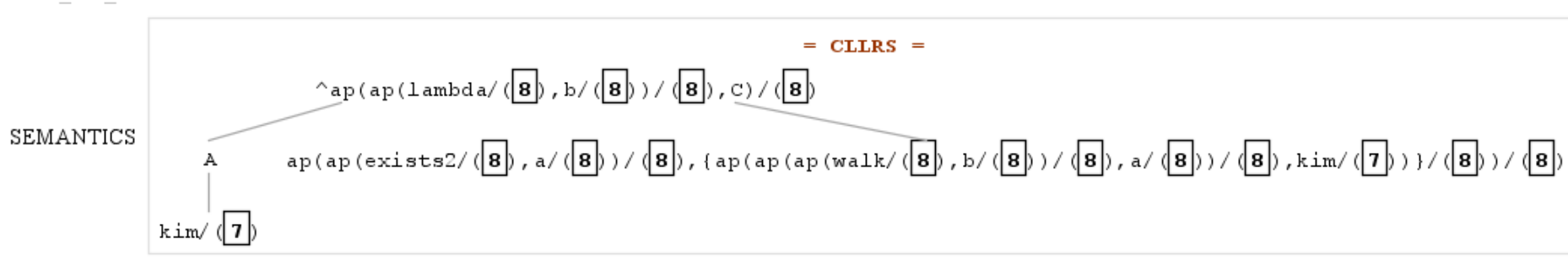
0) This is what we need to resolve:

SEMANTICS



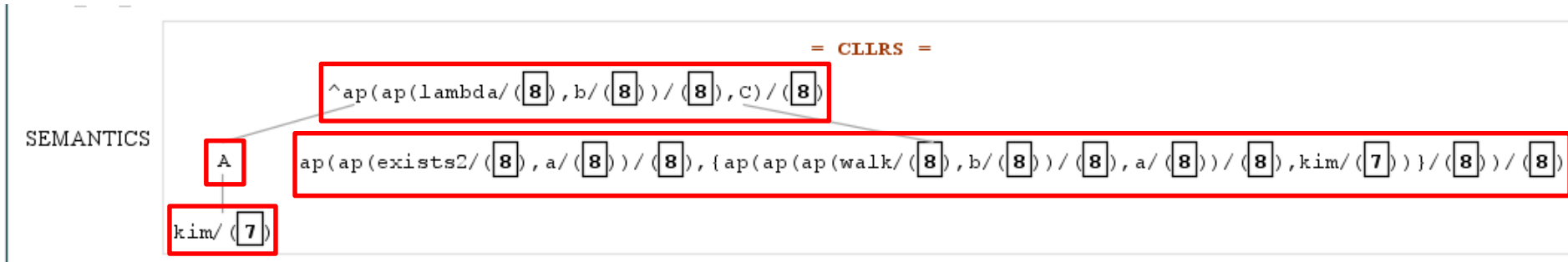
(Clearly, we also have a GUI problem)

1) Use the type system:



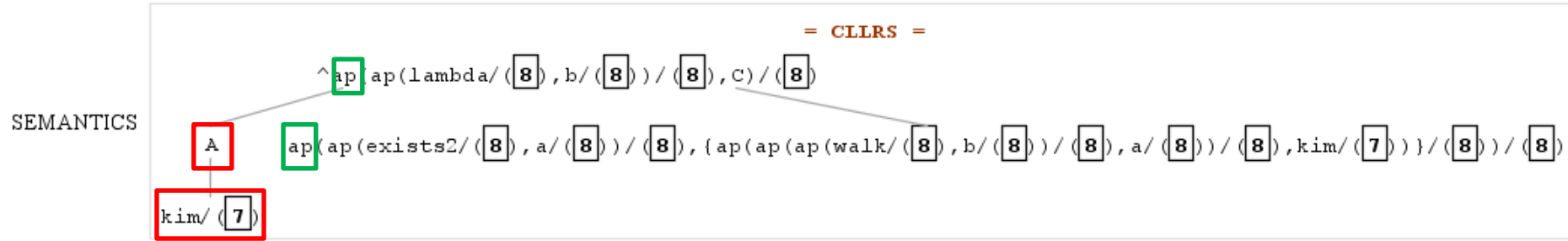
Out of 153 rejected candidate pairs for combination, 124 pairs (81%) were determined because of the type system; 26 (17%) because of cyclicity.

2) Identify “modules:”



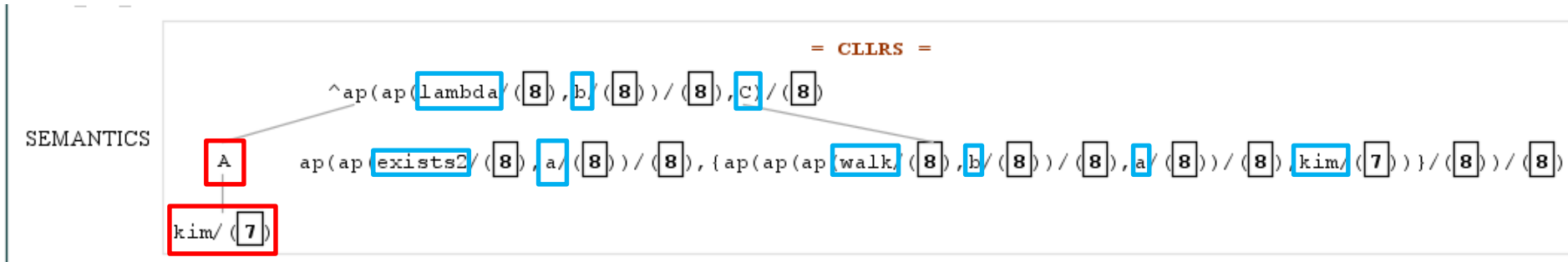
These are characterized by subgraphs connected by immediate subterm edges and no improper subterm edges.

3) Identify “tops” of modules:



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

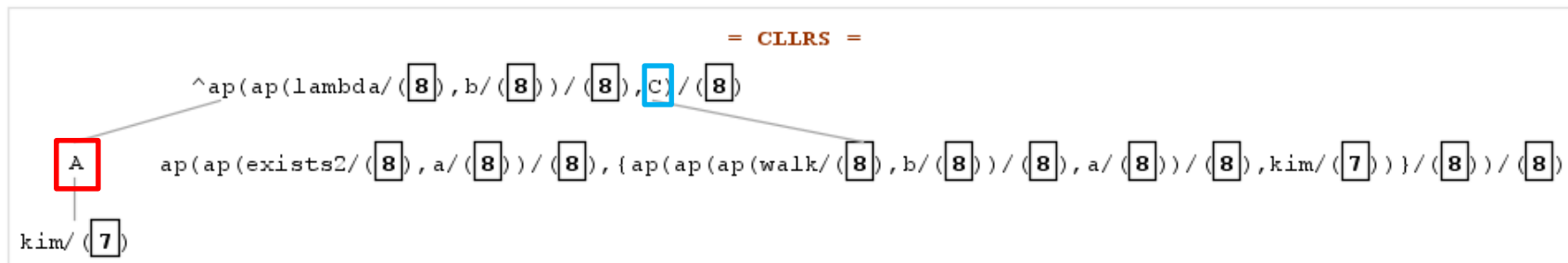
4) Identify “bottoms” of modules:



Bottoms are located by propagating downward from every top.

5) Non-determinately pick a top and bottom:

SEMANTICS



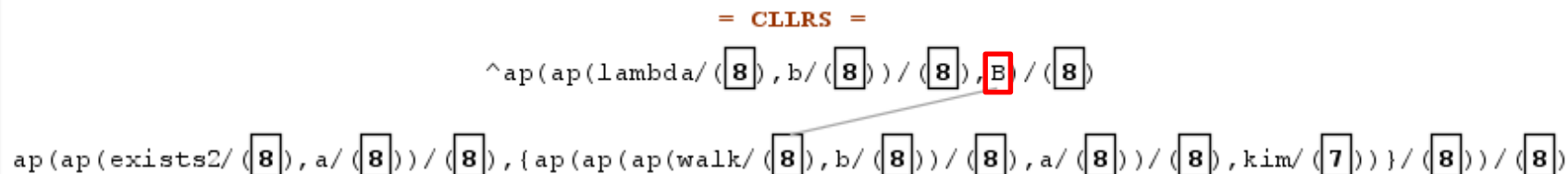
5) ...and speculatively combine them:

SEMANTICS

= CLLRS =

$\wedge \text{ap}(\text{ap}(\text{lambda}/\langle 8 \rangle, \text{b}/\langle 8 \rangle)/\langle 8 \rangle, \text{B}/\langle 8 \rangle)$

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



5) Keep doing this step:

SEMANTICS

= CLLRS =

$\wedge \text{ap}(\text{ap}(\text{lambda}/\langle 8 \rangle, \text{b}/\langle 8 \rangle)/\langle 8 \rangle, \text{ap}(\text{ap}(\text{exists2}/\langle 8 \rangle, \text{a}/\langle 8 \rangle)/\langle 8 \rangle, \{\text{ap}(\text{ap}(\text{ap}(\text{walk}/\langle 8 \rangle, \text{b}/\langle 8 \rangle)/\langle 8 \rangle, \text{a}/\langle 8 \rangle)/\langle 8 \rangle, \text{kim}/\langle 7 \rangle)\}/\langle 8 \rangle)/\langle 8 \rangle)/\langle 8 \rangle)$

Intuition behind this strategy

- There are fewer modules than there are subterms
 - If you stop and think about it, improper subterms could attach almost anywhere – perhaps these are best interpreted as restrictions on their symmetric duals?
- 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).

Intuition behind this strategy

- There are fewer modules than there are subterms
 - If you stop and think about it, improper subterms could attach almost anywhere – perhaps these are best interpreted as restrictions on their symmetric duals?
- 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).

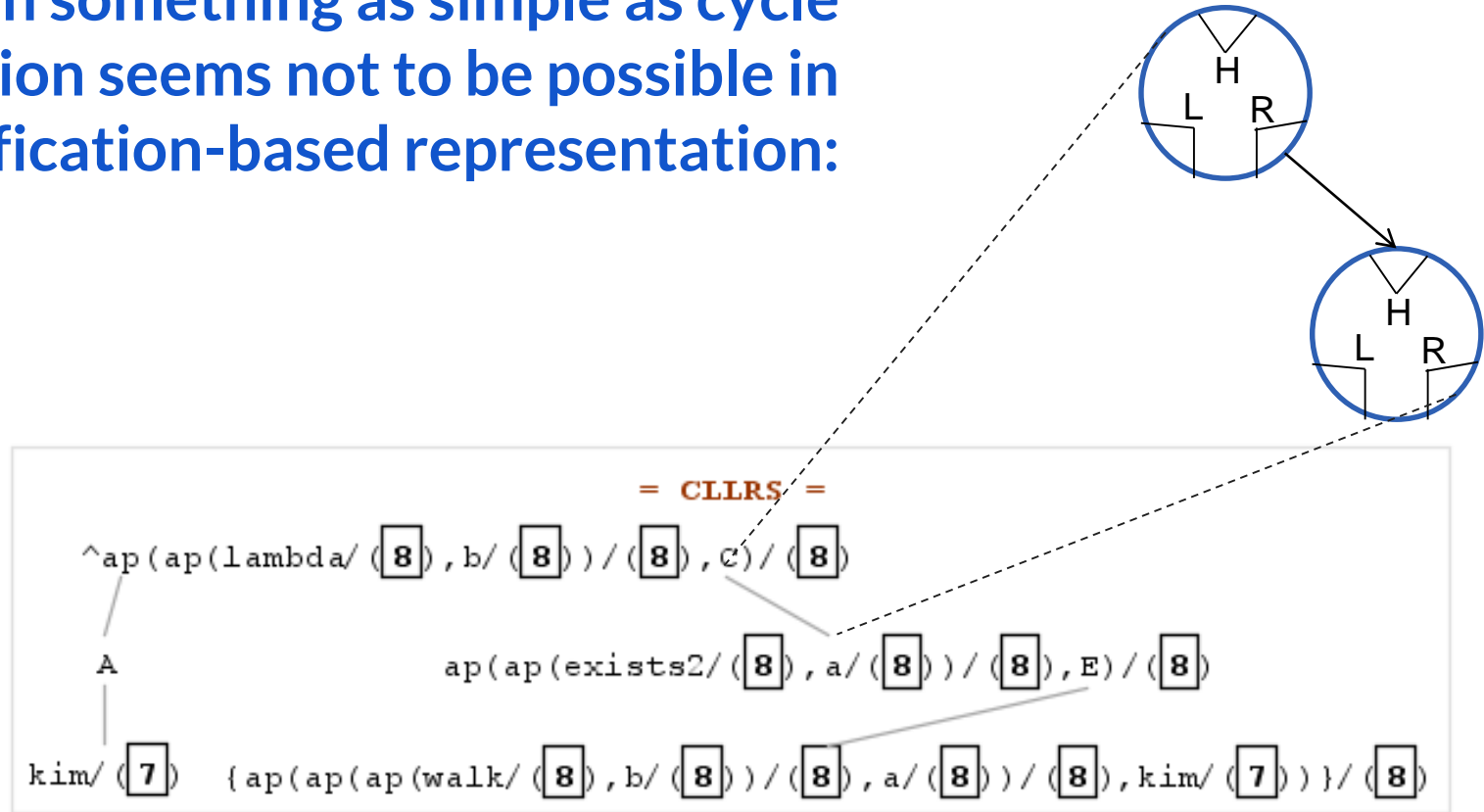
CLLRS constraint algebra in CHR

- literal/type consistency:
 $\text{literal}(N, \text{Lit1}, \text{Type1}) \setminus \text{literal}(N, \text{Lit2}, \text{Type2}) \iff \text{Lit1} = \text{Lit2}, \text{Type1} = \text{Type2}.$
- pivot and root are functions:
 $\text{pivot}(X, N) \setminus \text{pivot}(X, M) \iff N = M.$
 $\text{root}(X, N) \setminus \text{root}(X, M) \iff N = M.$
- immediate subterm irreflexivity:
 $\text{ist}(N, N, _) \implies \text{fail}.$
- immediate subterm uniqueness:
 $\text{ist}(M1, N, A) \setminus \text{ist}(M2, N, A) \iff M1 = M2.$
- subterm anti-symmetry:
 $\text{st}(M, N), \text{st}(N, M) \iff M = N.$
- Note the lack of arity consistency: applications are explicit parts of term trees.
- Note the last rule: cycles don't mean failure unless immediate subterms are involved.

Big Problems with CHR

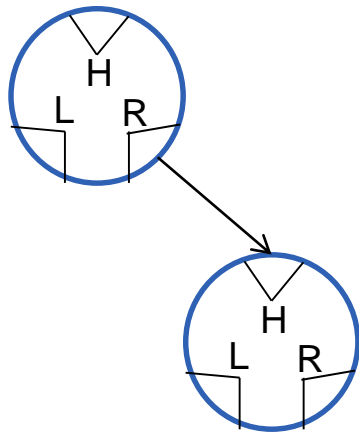
- Bugs – lots of them, and the remaining ones will not be fixed. We never did get CLLRS to work in SICStus Prolog 3.x because of these.
- There is additional overhead to using CHR on other platforms (SP4, SWI).
- Massive constraint-solving overhead (even in SP3): in addition to the rules on the previous slide, there are explicit rules for things like transitive closure of the subterm relation that add 1000s of other instances that must be managed alongside more substantive information.
- The underlying system is untyped, and so the savings accruable to strong typing are to some extent never realized.
- But what is the alternative?

Even something as simple as cycle detection seems not to be possible in a unification-based representation:



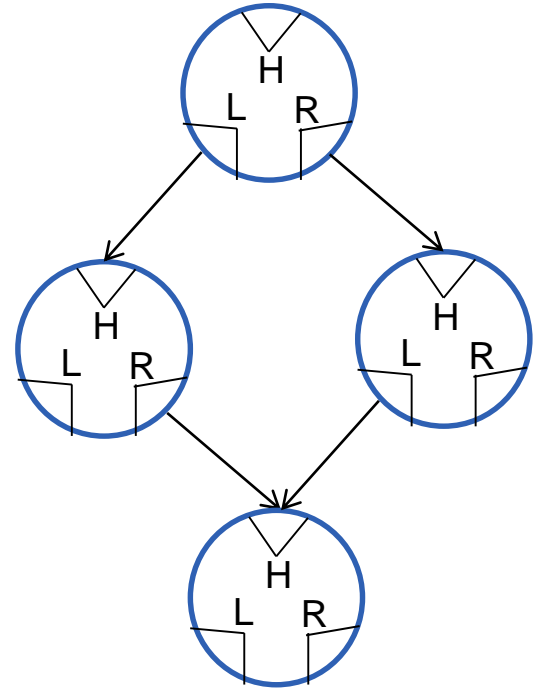
Even something as simple as cycle detection seems not to be possible in a unification-based representation:

- $Hx=Lx=Rx?$
- The simplest representation imaginable (node = meta-logical variable)
- But we have directed graphs that are cyclic in the category of undirected graphs.



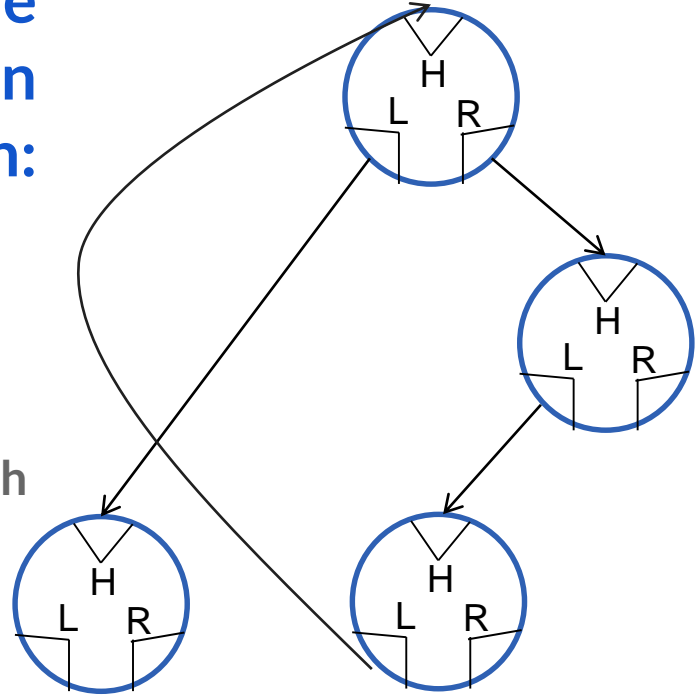
Even something as simple as cycle detection seems not to be possible in a unification-based representation:

- $Hx=Lx=Rx?$
- The simplest representation imaginable (node = meta-logical variable)
- But we have directed graphs that are cyclic in the category of undirected graphs.



Even something as simple as cycle detection seems not to be possible in a unification-based representation:

- ~~$Hx=Lx-Rx$~~
- Otherwise, there needs to be an explicit means of detecting cycles.
- Message-passing implementation through co-routining ($\text{when}/2$).
- Which direction the messages should travel in is an interesting question.
- Can speculate in parallel provided we carry a path around with us.



Where Things Stand Right Now

- Subterm algebra itself: nearly done with the TRALE implementation.
- Finite domains: designed (Colmerauer encodings).
- Contribution constraints: ?
- The good news is that search for paths is localized – whenever a new edge is added, we look at the part of the graph that may have developed a cycle as a result of adding that edge.
- How localized these graphs are depends on the grammarian.
- The bad news is that, as we begin to converge on a tailored algorithm for resolving our specific constraint problem, the age-old problems with tractability that are known to adhere to both LRS and RSRL (on which it is based) become exposed.
- “Improper subterm” is potentially very, very expensive.