Invited Talk at Paris VII

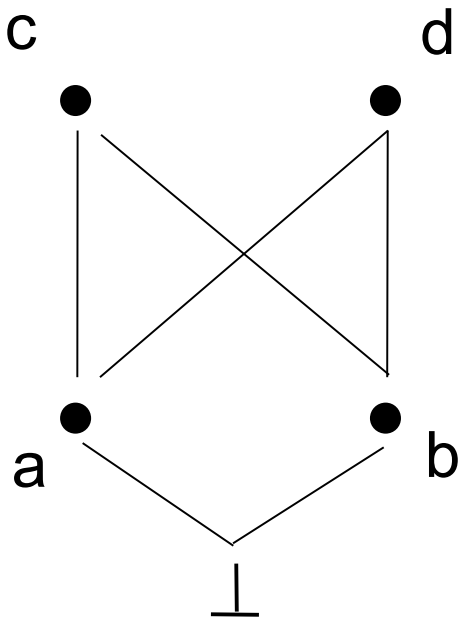
HPSG Design and Meet Semi-latticehood

Gerald Penn
Dept. of Computer Science
University of Toronto
(joint work with Rouzbeh Farahmand)

Meet Semi-latticehood

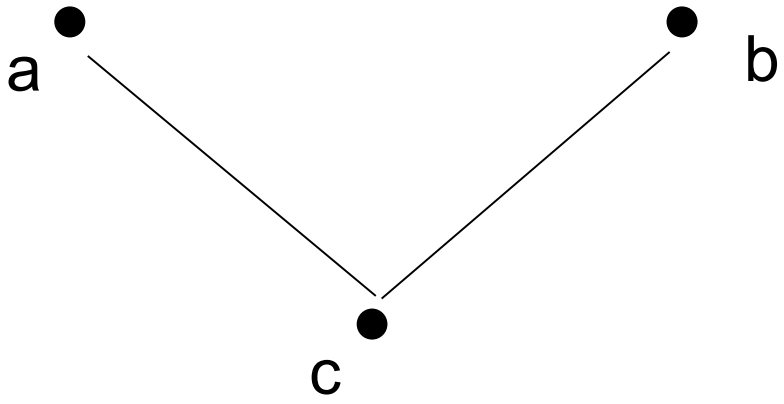
- A partially ordered set, (P,<), is a *meet semi-lattice* iff for every x,y in P, meet(x,y) is also in P.
- *Meet* is another word for greatest lower bound. It is therefore spatially defined.
  - Copestake (2002) draws her partial orders in the opposite orientation of mine (the type denoting everything is on the top), but she still refers to these structures as MSLs. They're not – for her, they should be *join semi-lattices*.
- A *join* is the dual of a meet – a least upper bound. In an MSL, some types have joins and some do not. The ones that do not are *incompatible* - they share no upper bounds at all. Sets that do have upper bounds (even if no joins) are *compatible*.

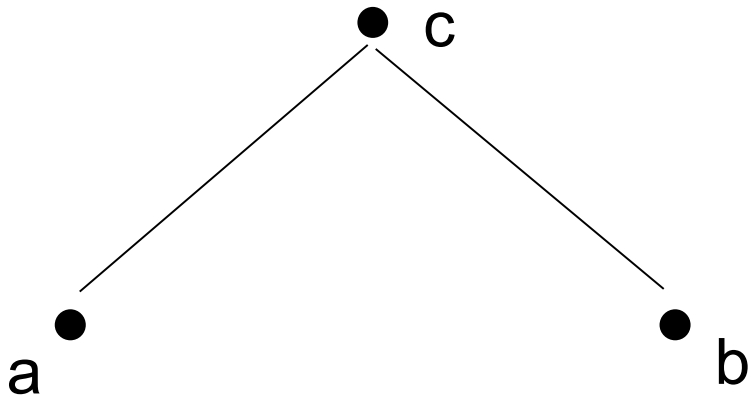
Example of a Non-meet-semi-lattice



Reducibility

- A type, c, is *meet-reducible* (resp. *join-reducible*) iff there exist types a and b such that a,b, and c are distinct and c=meet(a,b) (resp. join(a,b)).

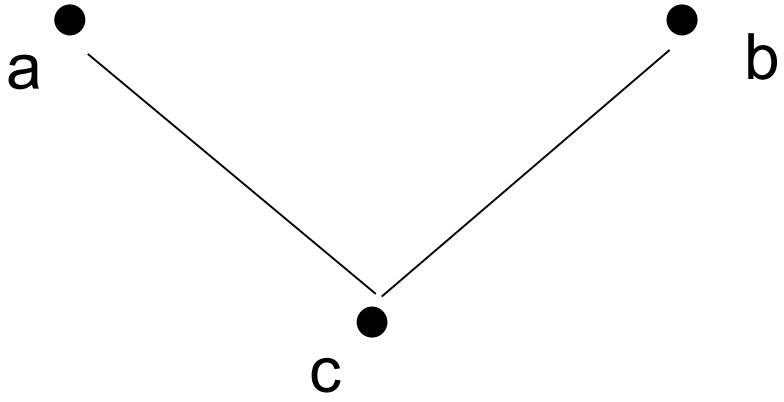


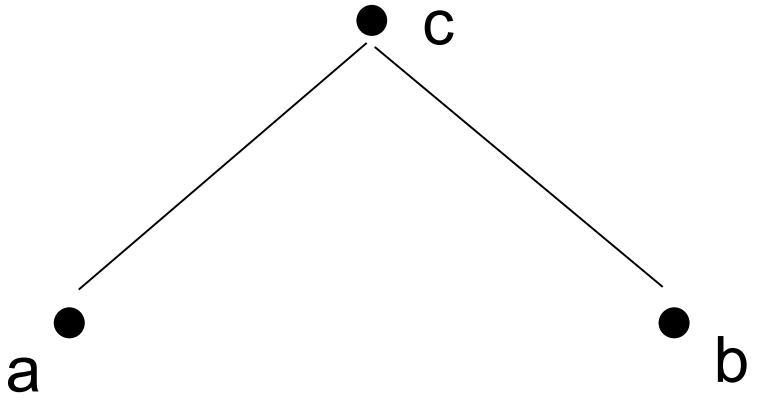


- **Proposition**: in a MSL, the meet irreducible types are the maximal types and the unary branching types.

Reducibility

- A type, c, is *meet-reducible* (resp. *join-reducible*) iff there exist types a and b such that a,b, and c are distinct and c=meet(a,b) (resp. join(a,b)).





- Meet/join-reducibility ratios are a good measure of how "interesting" the type signature is, and determines the size of their representations.

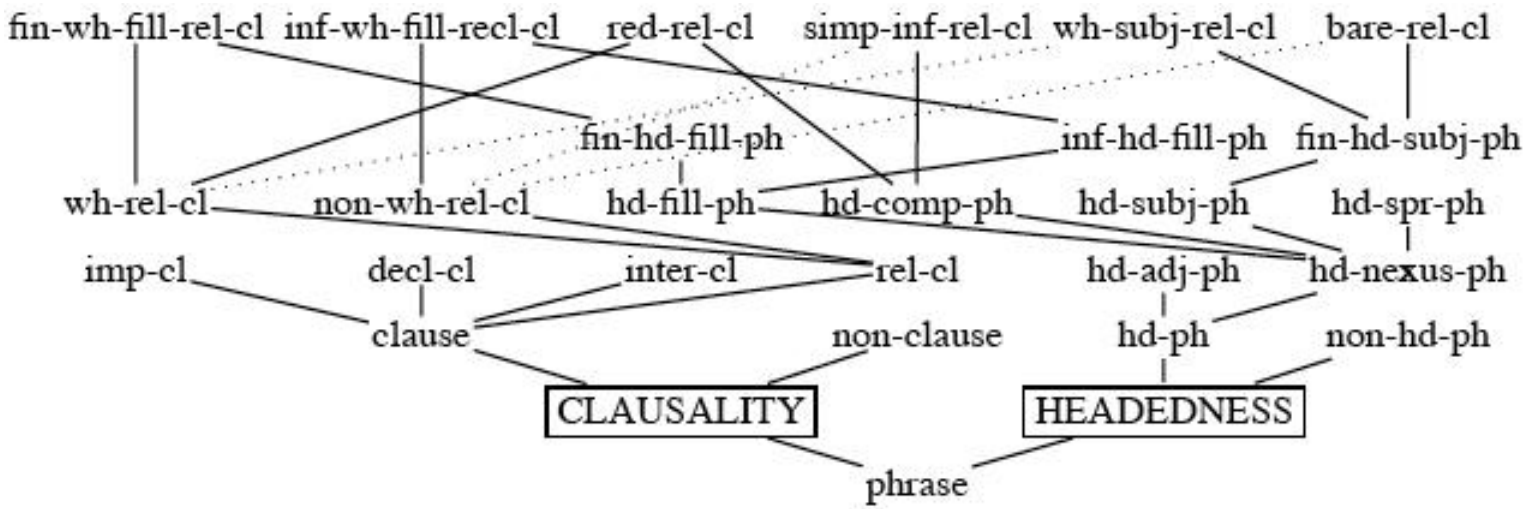
What's Wrong with Non-MSLs?

- In general: nothing.
- In HPSG: types play a very important role in introducing and constraining structure.
  - Many people prefer that every principle's antecedent be a single type.
  - Even people who do not so prefer will often introduce features at unique types, i.e., for every feature F there is a unique type t such that for all s, if s is appropriate to F, then s is a subtype of t.
- The result is that unifications (in the case of non-MSLs) or entire descriptions (in the case of non-unique feature introduction) must be either delayed or evaluated non-deterministically.

The ERG Type System is a Non-MSL

- No large-scale grammar has been more averse to disjunction than the English Resource Grammar (ERG; Flickinger et al. 1999).
- At the genesis of the ERG, disjunction was equated with non-deterministic search. This is not necessarily the case.
  - Independent sources of non-determinism in search can lead to intractability.
  - Delaying is less clear-cut.
- Yet the ERG's type system is not quite an MSL.
- It is very close, however...

Type Structure in the ERG



- There is a kind of structure here: multi-dimensional inheritance (Erbach, 1994).
- There's also a structure apparent in the choice of many type names, e.g., 1or3pl+2per+1per+non1sg
- The problem is that the structure isn't reified.

Do we really need Non-MSLs in HPSG?

- Under specific assumptions such as using multi-dimensional inheritance in the LKB (which provides no formal support for this choice), the answer seems to be 'yes'.
- On the other hand, there are also indeed grammars that bear little resemblance to the ERG but have non-MSL type systems.
- In my opinion, we have yet to see a satisfactory explanation or systematic investigation of when they're required or useful and why.
- This is especially interesting when considered against the backdrop of a community that does have some interest still in constrained formalisms.

Computational Approaches to Non-MSLs

- With not even a single exception that I am aware of, every HPSG parser has approached processing with non-MSLs by either prohibiting them altogether (e.g., most of the old ones, ALE until quite recently), or automatically converting them into an MSL.
- This is even more intriguing, because it implies that we need non-MSLs at the source-code level, in spite of a perceived difficulty in computing with them.
- ... and that perception is unsupported by experimental evidence (merely some very early, doomed, non-deterministic attempts).

Computational Approaches to Non-MSLs

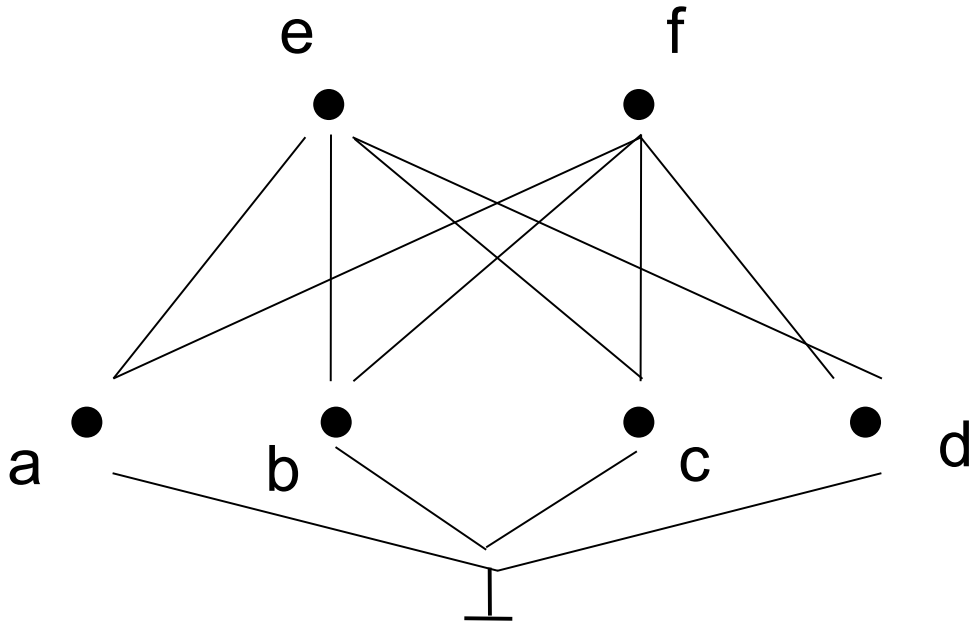
- Even so-called bit-vector-based approaches generally convert to an MSL (e.g., PET), even though the completion is in fact already latent in that choice of data structure (at least if unification is implemented by bitwise AND).
- The conversion can take place all at once during compilation (e.g., PET and now the LKB).
- ... or incrementally, as needed (e.g., the LKB's earlier method, although this apparently did not work correctly; such algorithms are known, however).

The Dedekind-MacNeille Completion

- Again, without exception, the conversion is implemented by computing the *Dedekind-MacNeille completion.*
- This is the smallest MSL that contains the original partially ordered set. If the original poset is not an MSL, it adds *completion types*.
- In the worst case, the DMC adds exponentially many completion types as a function of the size of the original poset.
- Presumably, everyone who performs a completion performs this one because of concern about size.

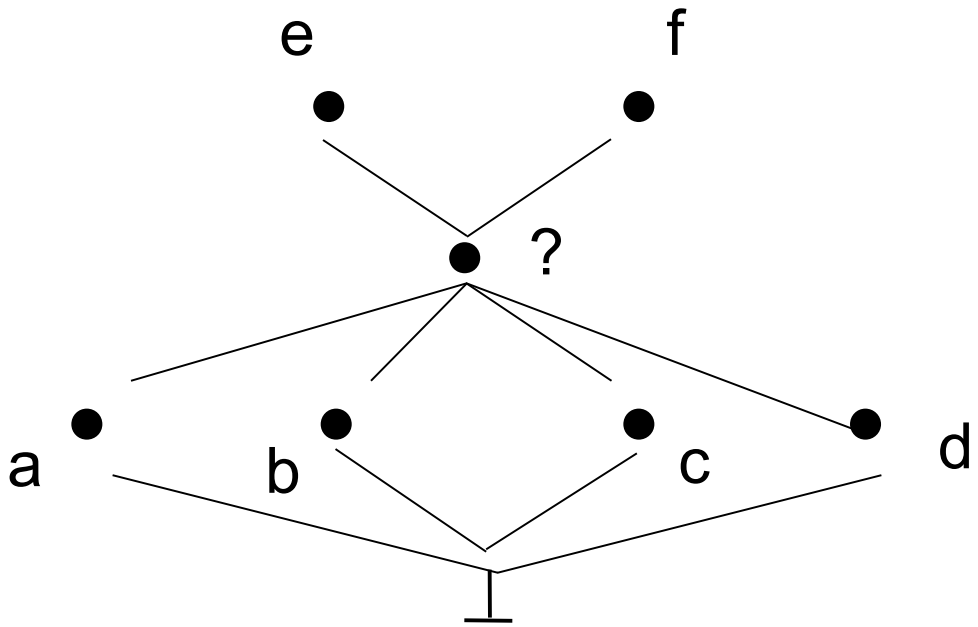
Example of a DMC

- Original:



Example of a DMC

- DMC:



Size of the DMC

- Size, however, is only a concern when enumerating or naming all of the completion types. Non-MSL users never do anyway (e.g., *glb127* in the LKB).
- It is generally not a concern with bit-vector-based representations. Size is determined by the choice and implementation of the supported structural operations.
  - For join-preserving bit-vector encodings, implemented by bitwise AND, the minimum attainable length of the vector (in bits) is the number of meet-irreducible elements in the poset. This is true of both the poset and its DMC!
- Non-bit-vector-based representations typically use string hashing, so almost no effect there either.

Time with the DMC

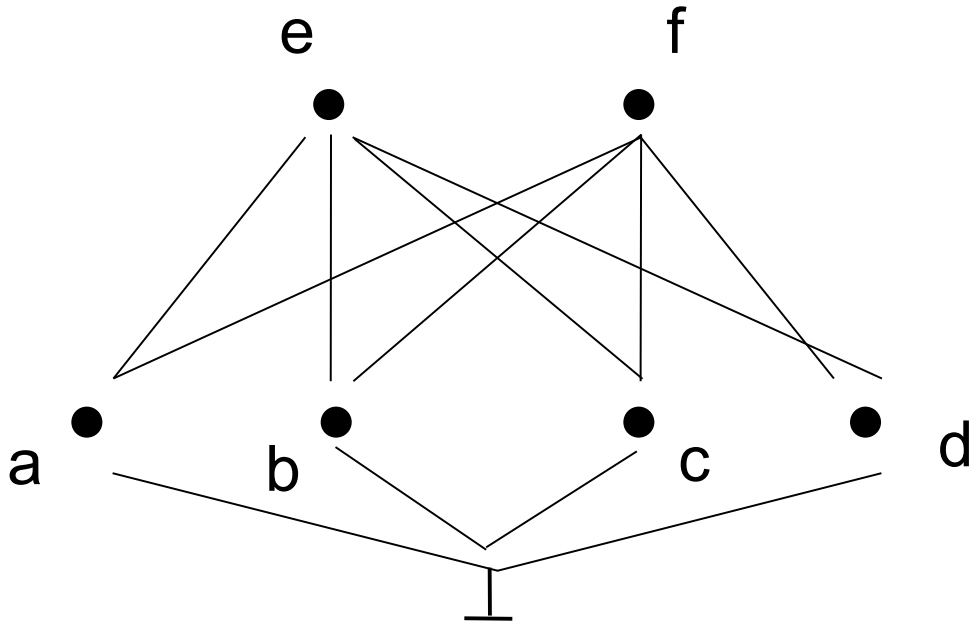
- When the DMC is compiled in advance, it adds essentially a constant factor to parsing time.
- In fact, one can regard completion types as an implementation of delaying (v. non-determinism) at the level of the type signature.
- We really don't know how much latency is added in practice by caching them on the fly – the only widely-used implementation of this had an error in it.
- We do know, however, that because the DMC is the *smallest* completion, it is the most *time-consuming* to cache-compute.

DMC Redux

- So there's a lot of room to expand into here – the DMC favours pre-computation or no computation and adds very little space.
- ...and remember that we've all decided to use non-MSLs at the source-code level because of their convenience.
- Is the DMC as convenient as it could be?
- Could completion types reify more than just an unnamed instance of delaying?
- The HPSG community has painted itself into a corner with a lot of naïve assumptions about computation with non-MSLs.

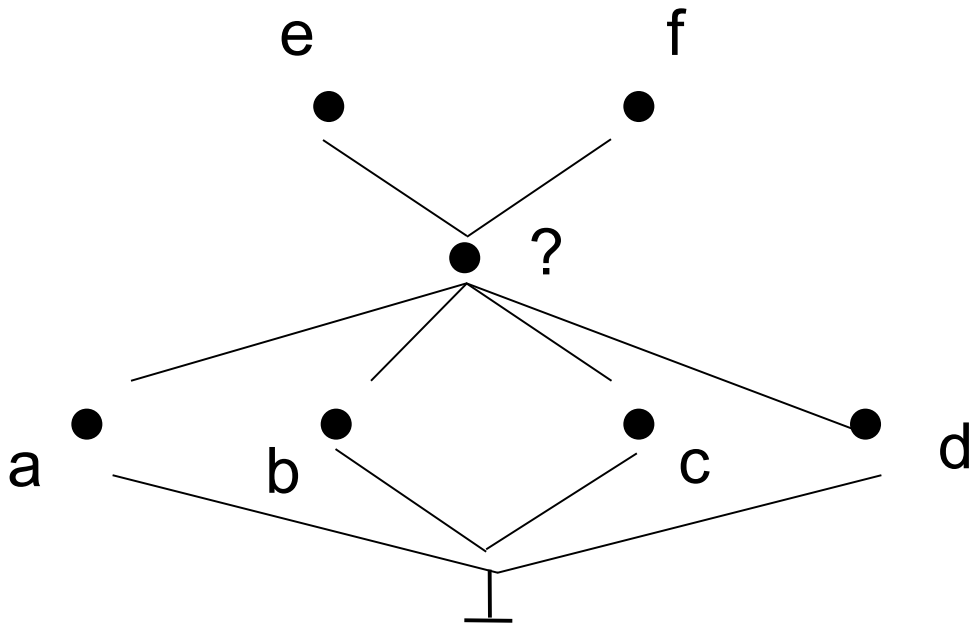
Example of a DMC

- Original:



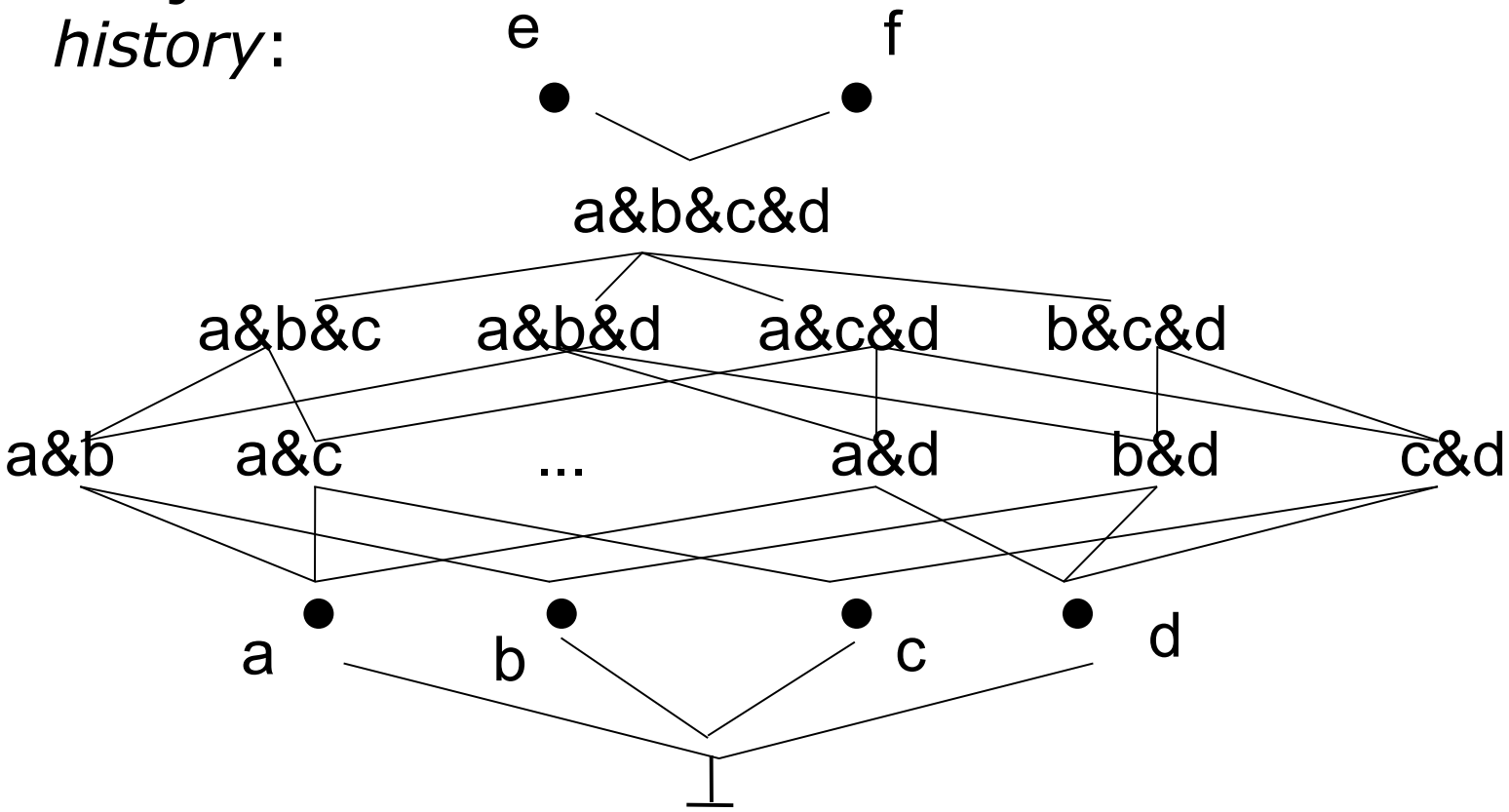
Example of a DMC

- DMC:



Example of (Part of) a Conjunctive Lattice

- Conjunctive Lattice tells us about unification *history*:

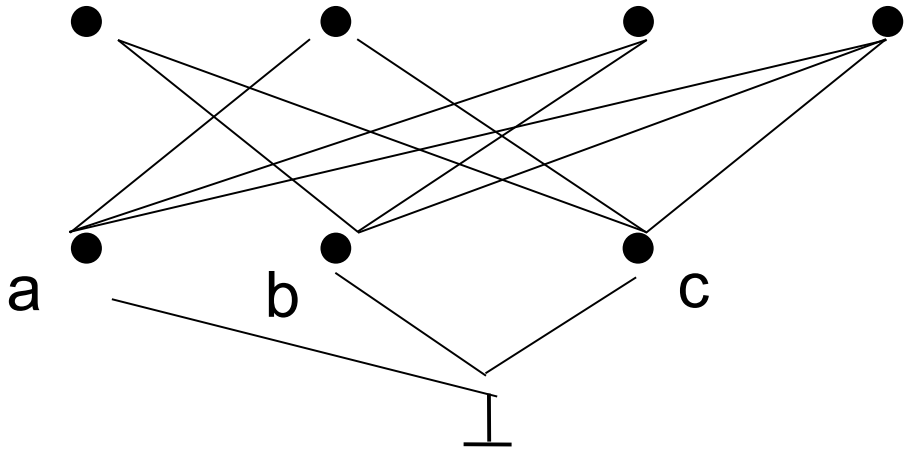


Direct Parsing with Non-MSLs

- Label representations of feature structures with conjoined sets of types.
- Whenever this set is a singleton, there may be some principles to enforce, features to add, etc.
- These labels can be shown to the user, and make sense.
- Naïve representation: sorted lists without joins (actually, sufficient to use maximally specific anti-chains)
- Naïve unification algorithm: set union, followed by attempting to "pinch" every pair to its join.

Pairwise Pinching will not Work

- No pair from {a,b,c} has a join, but {a,b,c} does!

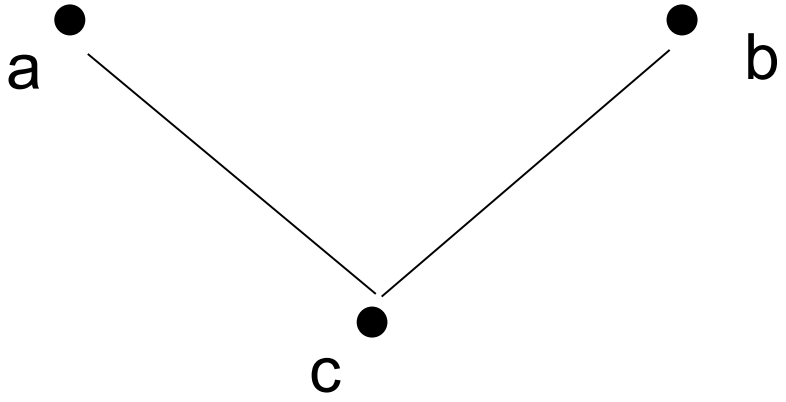


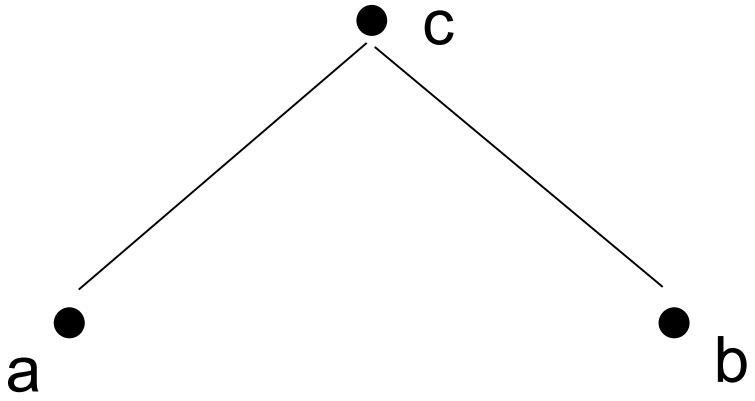
Prime Sets

- A *prime set* is an anti-chain, S, |S|>1, of which every non-empty subset, T, has a join iff |T|=|S| or |T|=1.
- The joins of prime sets of size > 2 cannot be computed by the pinching-pairs algorithm.
- **Proposition**: P is an MSL iff all of the prime sets are of size 2 or less.
- **Proposition**: The maximum size attainable by a prime subset of P is floor(½ (|P|-1)).
- In principle, there could be a lot of prime subsets.

Anti-Chain

- Given a poset P, and subset S, S is an *anti-chain* iff for all x,y in S, neither x<y nor y<x.





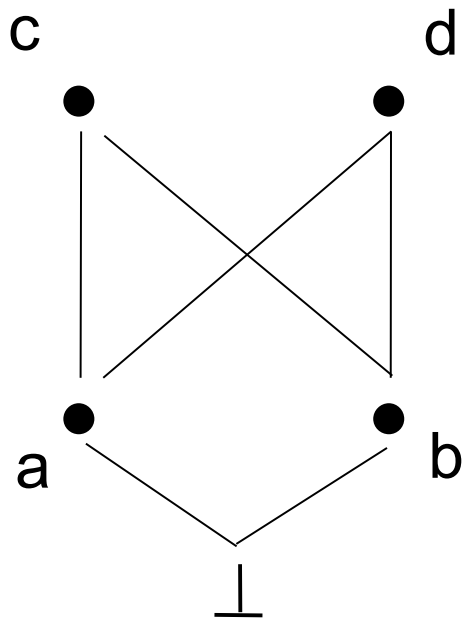
- {a,b} is an anti-chain. {a,c} and {a,b,c} are not.

Pseudoprime Sets

- Yet computing the prime sets of P doesn't have to be expensive, when there aren't many of them.
- A *pseudoprime* set is a compatible set, S, |S|>1, of which every non-empty subset, T, has a join iff |T|=1.
- **Proposition**: Every proper subset of a prime set is a pseudoprime or singleton set.
- **Proposition**: Every proper subset of a pseudoprime set is pseudoprime or a singleton set.
- These two propositions immediately give us a constructive recipe for discovering primes: don't try – find the pseudoprimes instead.

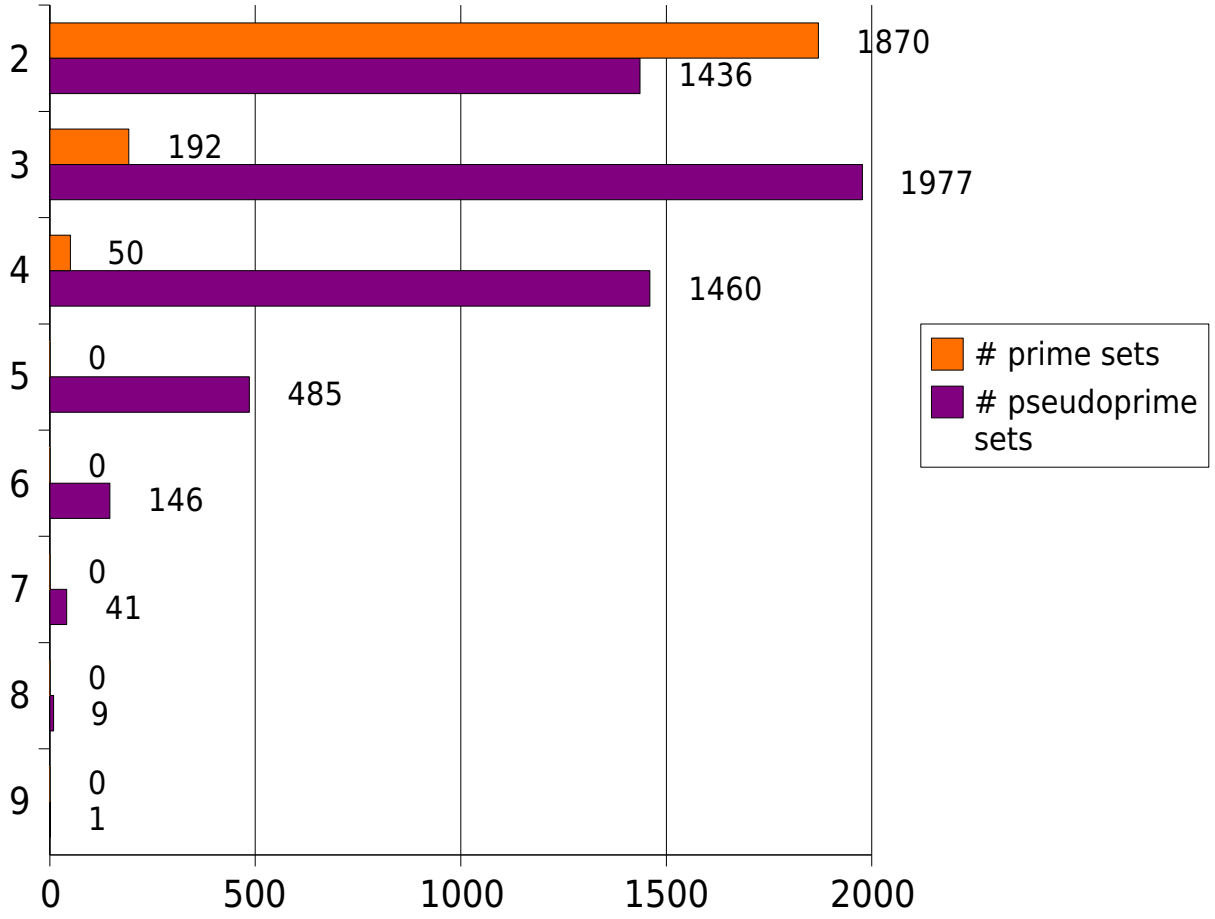
Seeding the Pseudoprime Algorithm

- Every such pair {a,b} is a pseudoprime set of rank 2.

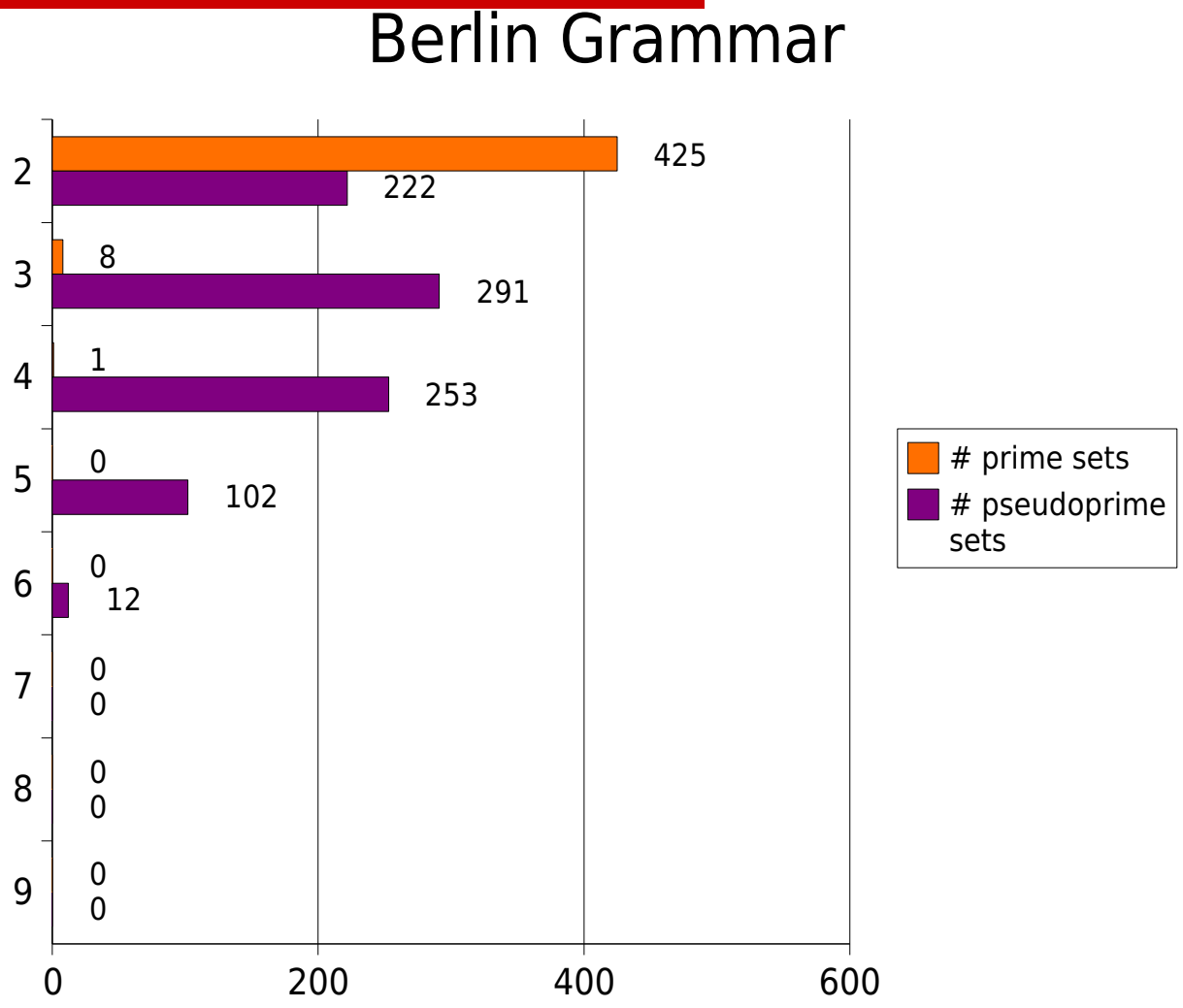


Primes and the ERG

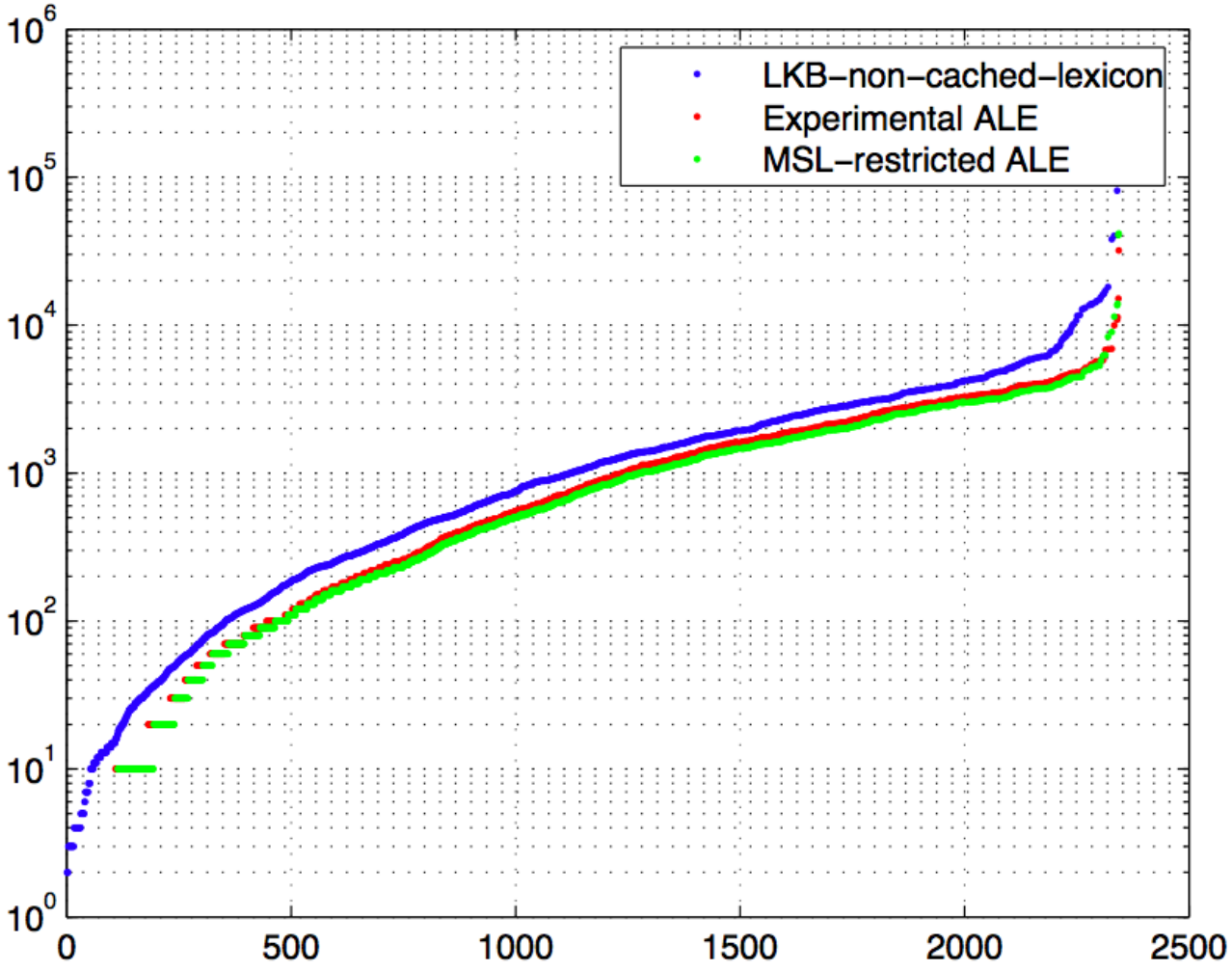
English Resource Grammar



Primes and the Berlin Grammar



How fast is Parsing without DMC?



Conclusion

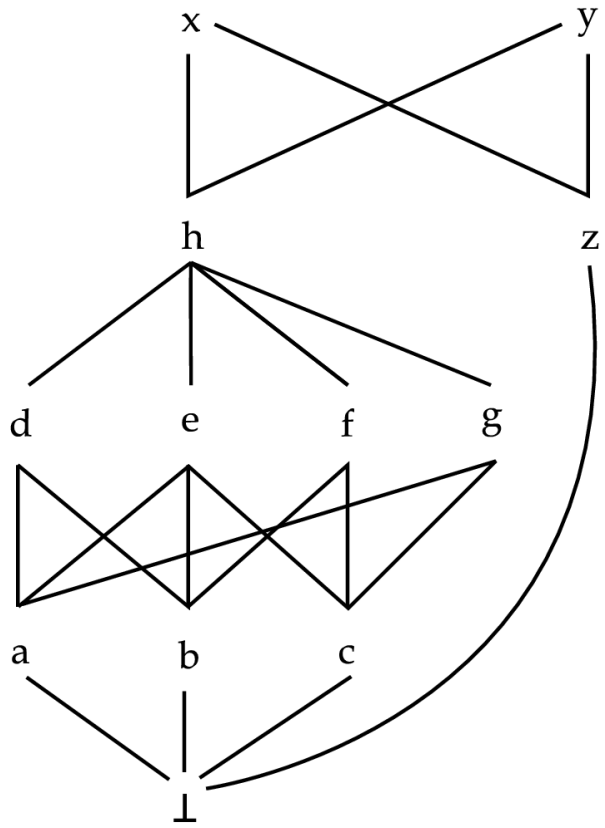
- There is a lot of very interesting work to be done on non-MSLs, both empirical (investigating conventions of use) and computational (better algorithms and representations).
- The Dedekind-MacNeille completion has been an article of faith in HPSG parsing that presents serious drawbacks to almost every aspect of using non-MSLs in HPSG design.
- Even if we do really need non-MSLs, we definitely do not need the DMC.

Conclusion

- Prime sets are a simple tool for spectrally analysing the structure of non-MSLs in a way that generalizes the usefulness of MSLs.
- Prime sets grow very slowly on multi-dimensional structures such as the ERG, in which intersections among dimensions are explicitly selected.
- This leads to a very natural and efficient procedure for type unification - even our first attempt at an implementation suggests an acceptable latency (6% at run-time) for this approach.
- They also allow us to name our types sensibly, using a conjunctive lattice.

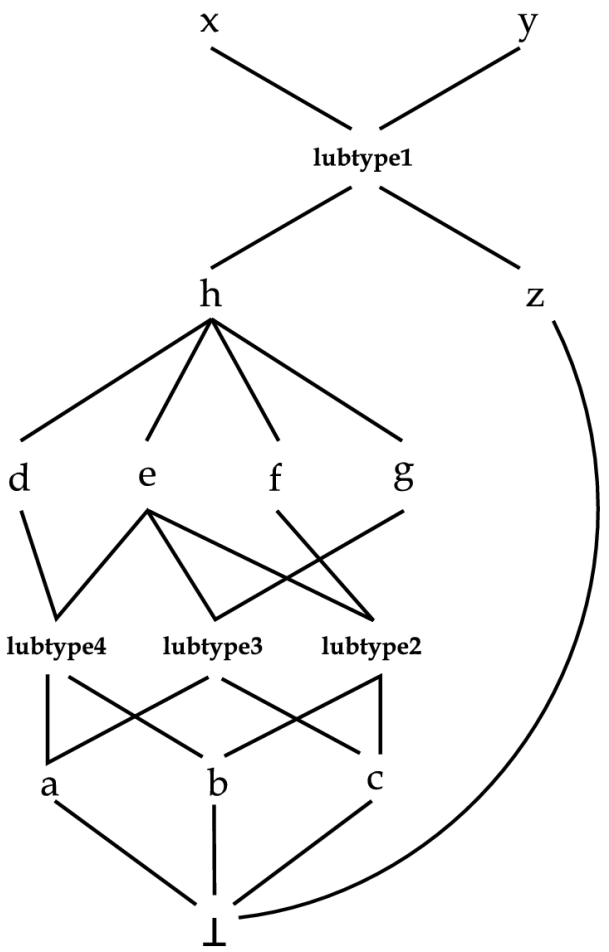
Automaton-based Indexing of Prime Sets by Example

- Original poset:



Automaton-based Indexing of Prime Sets by Example

- DMC (not computed):



Automaton-based Indexing of Prime Sets by Example

- Automaton: