

[acd]

a c d

[-cef]

a d e f

[-dgh]

a c f g h

[-dij]

[-fhl]

[-gmn]

[-hop]

a i j k l m n o p

[-irg]

a r j k l m n o p q

[-qbs]

a b j k l m n o p r s

[-rtv]

a b j k l m n o p s t v

[-suv]

a b j k l m n o p t u v w

[-wut]

no v-s r-q or we in -v clause

-vpn

-ukm

-tnm

-pon

-olj

-nml

-mkl

-ljk

-kab

or -kaj

or [-kbj]

[-jab]

[-ejrq]

[-irbs]

[-eiab]

[abv][ab-v]

[bj-l]

[ab-k][abk-v]

[ab-j][-vkhb]

[-ojb][-vjk o]

[-ojl][bj-l]

[bj-k][jk-l]

In order to stop a traversal, a clause $[c \dots]$ cannot share terms w/ the clauses w/

$[c x y]$

$[x z u]$

$[y u v]$

Otherwise $[c \dots]$ could pair w/ $[c x y]$ to make $[x y]$ and if \dots is z, u, v or y it could combine again w/o going over 4 terms

If $x \in [c \dots]$, $[x y z u]$ ~~is~~ ✓

If $z \in [c \dots] \rightarrow [z y u \dots]$

$u \in \dots \rightarrow [z y u \dots]$

If $y \in [c \dots] \rightarrow [x u v \dots]$

$u \rightarrow [x u v \dots]$

$v \rightarrow [x u v \dots]$

If y or $x \in [c \dots] \rightarrow [x y] \rightarrow [y z u]$ and $[x u v]$

To block traversal from $[a c d]$,

$[c u v]$ cannot have a or d and

$[d u x]$ a or c and

$[u \dots]$

Consider traversal from $[a c d]$ on the left page

$[a d e f]$ and $[a c g h]$

Can continue if $a, d, e \in [f \dots]$

$a, e, \text{ or } f \in [d \dots]$

$a, d, \text{ or } f \in [c \dots]$

$a, c, \text{ or } g \in [h \dots]$

$a, c, \text{ or } h \in [g \dots]$

$a, g, \text{ or } h \in [e \dots]$

a bei

-von

-kab

-swon

(ab-fl

ab-um

ab-ml

ab-ull

ab-ol

ab-lh

ab-l

ab-fl

ab-ij

ab-nm

ab-mk

ab-m-t

ab-tk

ab-tj

ab-t

ab-u-w

ab-ru

-irbs

-svut

-wtkm

-wnkm

-vpm

If we traverse the graph starting from a random clause, what will stop the traversal?

The cluster has no common terms w/ its neighbors

Which means...

The cluster we're working with is unlike the original large clause in which that clause popped a term.

I think what we're trying to do here is break up/reverse parts of the original clause to never deal w/ a clause larger than 3 terms.

How can we prove such a transformation is always possible?

Whenever you pop a term, you are left w/ a 2-, 3- or 4-t clause.

In this clause, there are 1-4 terms which are also popped

Looking at the clauses containing the negatives of those 1-4 terms, we may add that clause to the cluster if at least one term overlaps.

So what? There may be no overlap.

We know the (new) ab output must happen with

$[a \ b \ X]$

$[a \ b \ -X]$ where X is some OFT

Now we have two targets to hit. How can we guarantee we don't exceed 4-t?

How could the inputs to $[a \ b \ X]$ look?

$[a \ b \ Y] \ [-Y \ X \ a]$

$[a \ X \ Y] \ [-Y \ b \ a]$

$[a \ X \ Y] \ [-Y \ b \ X]$

$[a \ b \ X \ Y] \ [a \ b \ X \ -Y]$

$[a \ b \ X \ Y] \ [-Y \ X \ a]$

$[a \ b \ X \ Y] \ [-Y \ a \ b]$

The $-Y$ that pops always carries $(X, a, \text{ or } b)$

Always carries a term yet to be popped

$[ab]$
 $[ab-v] [abv]$
 $[ab-k] [abk-v]$
 $[bj-k] [ab-j] [ab-j] [bjk-v]$
 $[bj-o] [jko-v]$
 $[j-l-o] [bj-e] [j-k-n] [no-v]$
 $[bj-w] [jk-e] [k-e] [kl-n] [np-v] [no-p]$
 $[kl-m] [lm-n]$

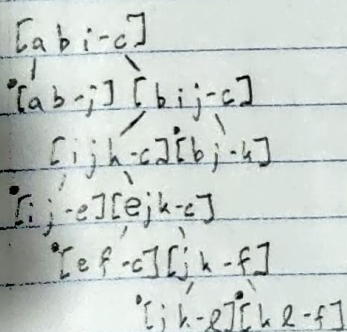
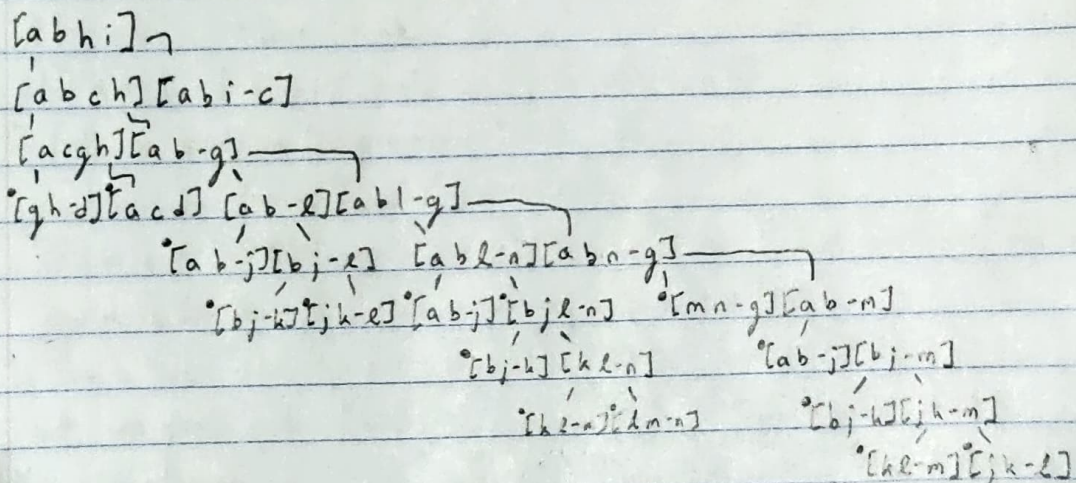
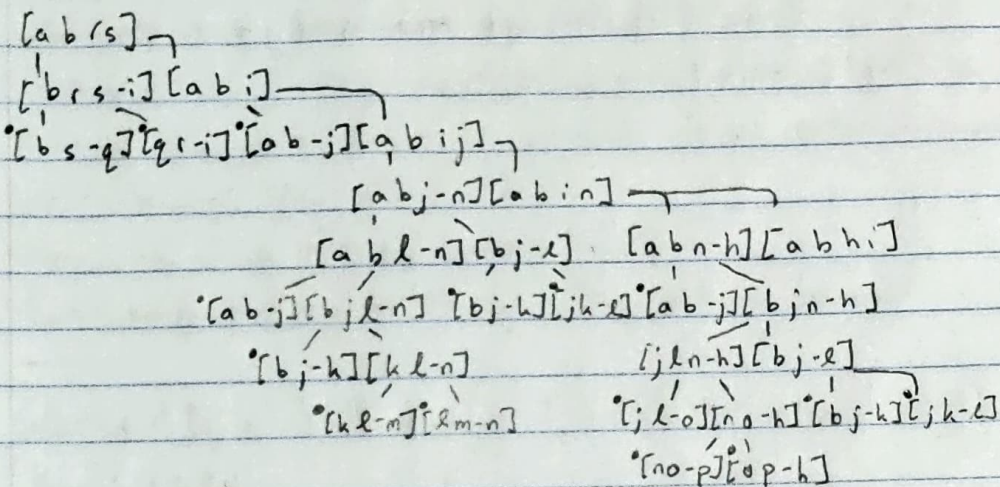
$[abv]$
 $[ab-r] [abrv]$
 $[ab-k] [abk-r]$
 $[bj-k] [ab-j] [ab-j] [bjk-r]$
 $[bjk-m] [jkm-r]$
 $[jk-e] [bjl-m] [jkm-t] [jmt-r]$
 $[kl-m] [bj-k] [mn-t] [jk-n] [km-v] [tu-r]$
 $[jk-e] [kl-n]$
 $[kl-m] [lm-n]$

$[abrv]$
 $[ab-sv] [abrs]$
 $[vw-s] [ab-w]$
 $[ab-h] [abk-w]$
 $[bj-k] [ab-j] [ab-j] [bjk-w]$
 $[bjk-m] [jkm-w]$
 $[jk-e] [bjl-m] [jkm-t] [jkt-w]$
 $[kl-m] [bj-k] [mn-t] [ju-n] [jk-e] [klt-w]$
 $[jk-e] [kl-n]$
 $[kl-m] [lm-n]$
 $[kl-m] [kmt-w]$
 $[km-v] [tu-w]$

Observations:

- For each clause, there were two inputs composed of an OFT + the output's terms
- Each clause's inputs have two rather separate branches
- Branches terminate when they reach a given 3-t clause
- ~~None of the following steps:~~

Oops, missed the abis branch



Observations:

- Every clause w/ ab has $[a b -j]$ somewhere in its grand*-children
- Consider an implication like $\neg [ABc] \rightarrow ([ABX][C A -x])$, it seems that along the X branch, you will never see a $\neg X$ and along the $\neg X$ branch, you will never see a X incorrect
- Every partner of $[a b -j]$ has a or b
- How can it be shown that $[ab]$ can be derived from the given clauses without processing a clause whose length exceeds 4 terms?
- Similarly to how the shrinking clauses contain the popped term + two terms yet to be popped, moving up a branch, clauses contain the popped term + two other terms in the cluster.
- Every derived clause represents the current cluster
- Sometimes you must derive multiple clusters before continuing traversal

Starting at $[-j a b]$ wts we can get $[-x a b] \forall -x$ in the shrinking clauses

1. $[k -]$ is either

1) $[-k a b] \rightarrow$ have $[k a b] \checkmark$

2) $[-k a j]$ or $[-k b j] \rightarrow$ imply w/ $[j a b] \rightarrow [-k a b]$

2. $[-l -]$ contains 2 terms from $\{a, b, j, k\}$

and we know $[k a b]$ and $[j a b]$ exist

if j or $k \in [-l x y]$, we imply $[-l a b y]$ where y is in $\{a, b, j, k\}$ and is not x
now we either have $[-l a b]$ or can imply it in one step

3. This is true for all shrinking clauses (TODO: paper proof w/ PMI)

Next, we come across either a clause like ① $[-x y z]$ where y and z are the initial placements of the positive form of the popped terms or ② $[-x a z]$

③ $[-x a b]$ ④ $[x a b]$

in case ①, we can derive $[-x a b]$ since $[-y a b]$ and $[-z a b]$ exist

②, we get $[-x a b]$ since $[-z a b]$ exists

③ we have $[-x a b]$

④ we know $[x]$ must be popped to find $[-x -]$ and it will fall into one of the 3 categories

Last part: We know the initial large clause is seeded w/ some positive terms and a , b , or both.

Therefore there must be a clause that pops these terms.

ie, there is a clause like $[-x y z]$ where x is one of the positive terms in the seed and y, z are other terms (could be a or b)

On the last page, it is shown we can acquire a clause like $[-x a b]$ for every term, x , that's popped

So we can pop the seeded terms and derive $[a b]$ w/o exceeding a clause of length 4.