

Improved Conflict-Clause Minimization Leads to Improved Propositional Proof Traces

Allen Van Gelder

Computer Science Department
University of California
Santa Cruz, CA, USA

`avg@cs.ucsc.edu`

`http://www.cse.ucsc.edu/~avg/Papers/`

`http://www.cse.ucsc.edu/~avg/ProofChecker/`

These slides are `ccmin-trans.pdf`

The Problem: Minimize a Resolution Derivation from a Conflict Graph

Conflict Graph: Basic Data Structure in Many Modern SAT Solvers

- Introduced in current form in **GRASP** by Marques-Silva and Sakallah [96,99]
- Adopted into **Chaff** by Moskewicz, Madigan, Zhao, L. Zhang, and Malik [01]
- Variations in **zChaff** by L. Zhang, Madigan, Moskewicz, and Malik [01]

If x is implied by unit-clause propagation, there is some clause $[x, \neg y_1, \neg y_2, \dots]$ such that y_1, y_2, \dots were assumed or implied earlier.

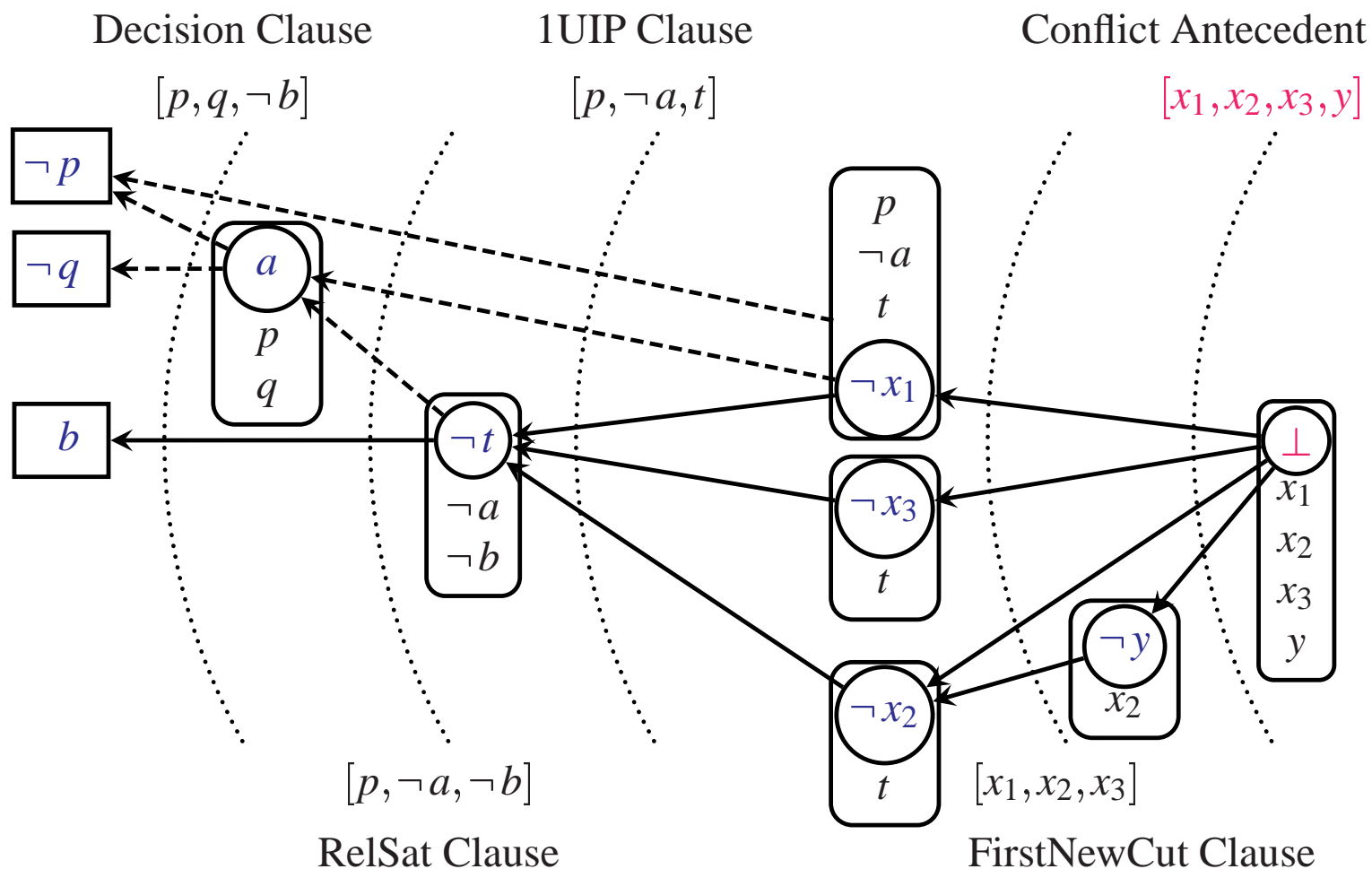
Put arrows from vertex x to vertices y_1, y_2, \dots

$[x, \neg y_1, \neg y_2, \dots]$ is called the *antecedent* of x .

x may be \perp , denoting *false*. Vertices reachable from *false* = conflict graph.

Observation: The set of clauses in a conflict graph is *renamable Horn*.

Various Cuts in Conflict Graph Yield Different Conflict Clauses



\perp denotes false.

Dashed lines go to vertices at lower (earlier) “decision levels”.

From Conflict Graph to Resolution Derivation of Conflict Clause

- L. Zhang and Malik pseudocode (paraphrased) to generate refutation [DATE 03]:

```
1.   cl = antecedent(false);
2.   while ( ! is_empty_clause(cl)) {
3.       lit = choose_literal(cl);
4.       ante_cl = antecedent(lit);
5.       cl = resolve(cl, ante_cl); }
```

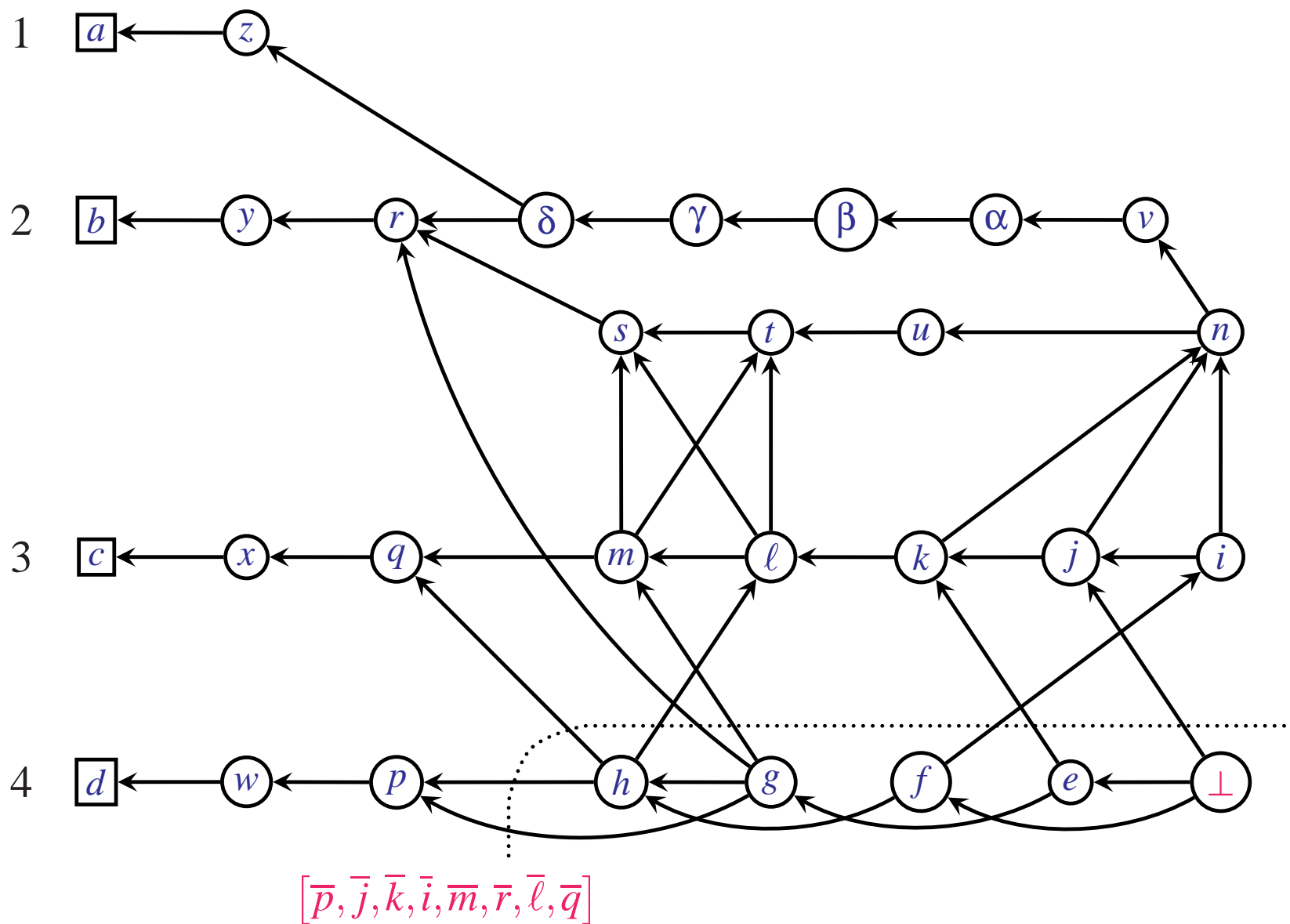
Literal order unspecified (see `choose_literal`).

- Theoretical analysis by Beame, Kautz, and Sabharwal [JAIR 2004]
Trivial resolution (TVR, defined there) does minimum number of resolutions.
- Exponential worst case for bad literal order shown by Van Gelder [SAT 07].
Reverse chronological order of implied literals produces a TVR.

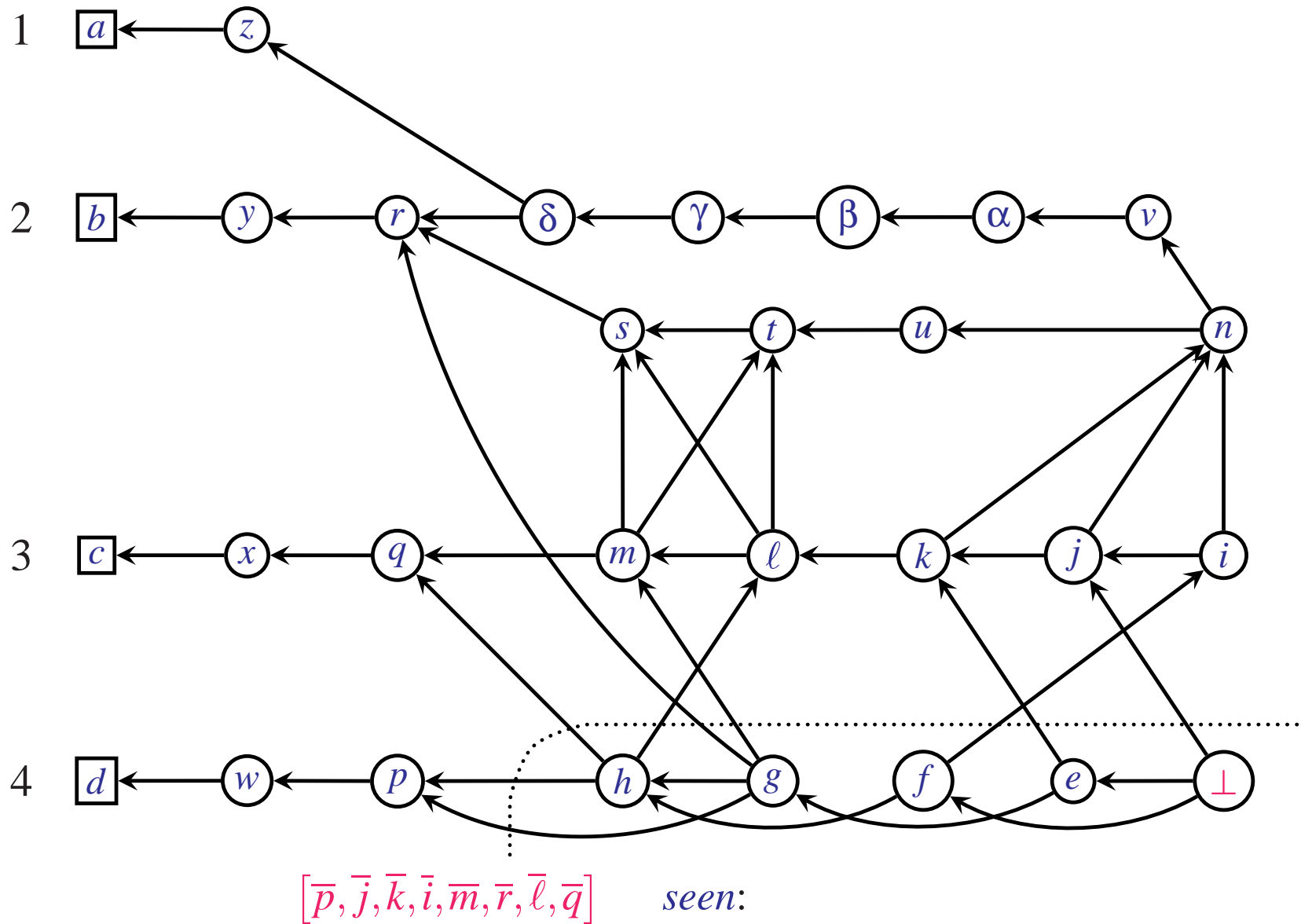
Relation to Clause Minimization:

- Linearly searching the **implied-literal trail** (in reverse) is practical for **1-UIP conflict clause** because only the *current* decision level gets processed.
- **Clause minimization** needs to process implied literals at *many* levels.

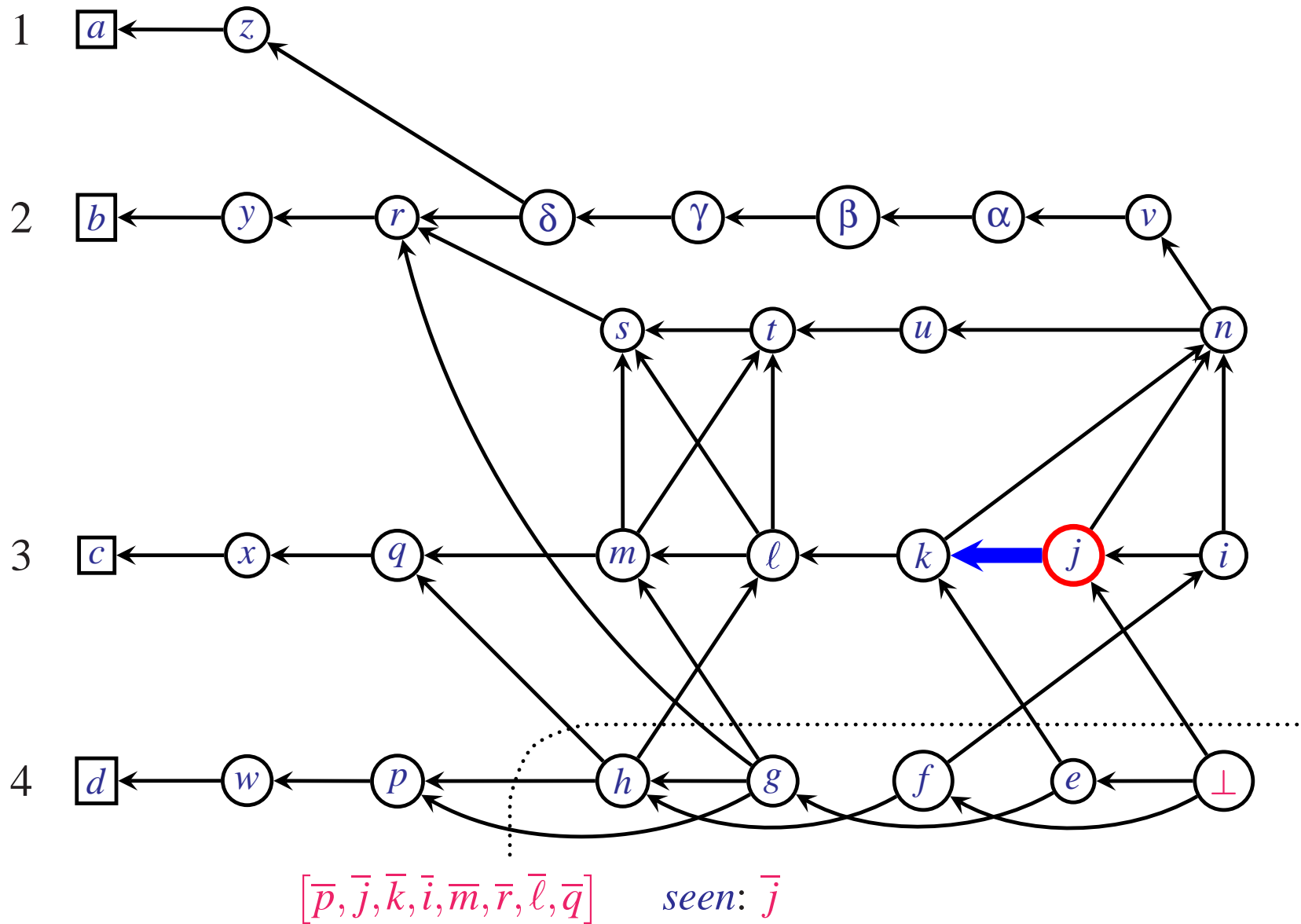
Conflict Graph with Minimization Opportunities



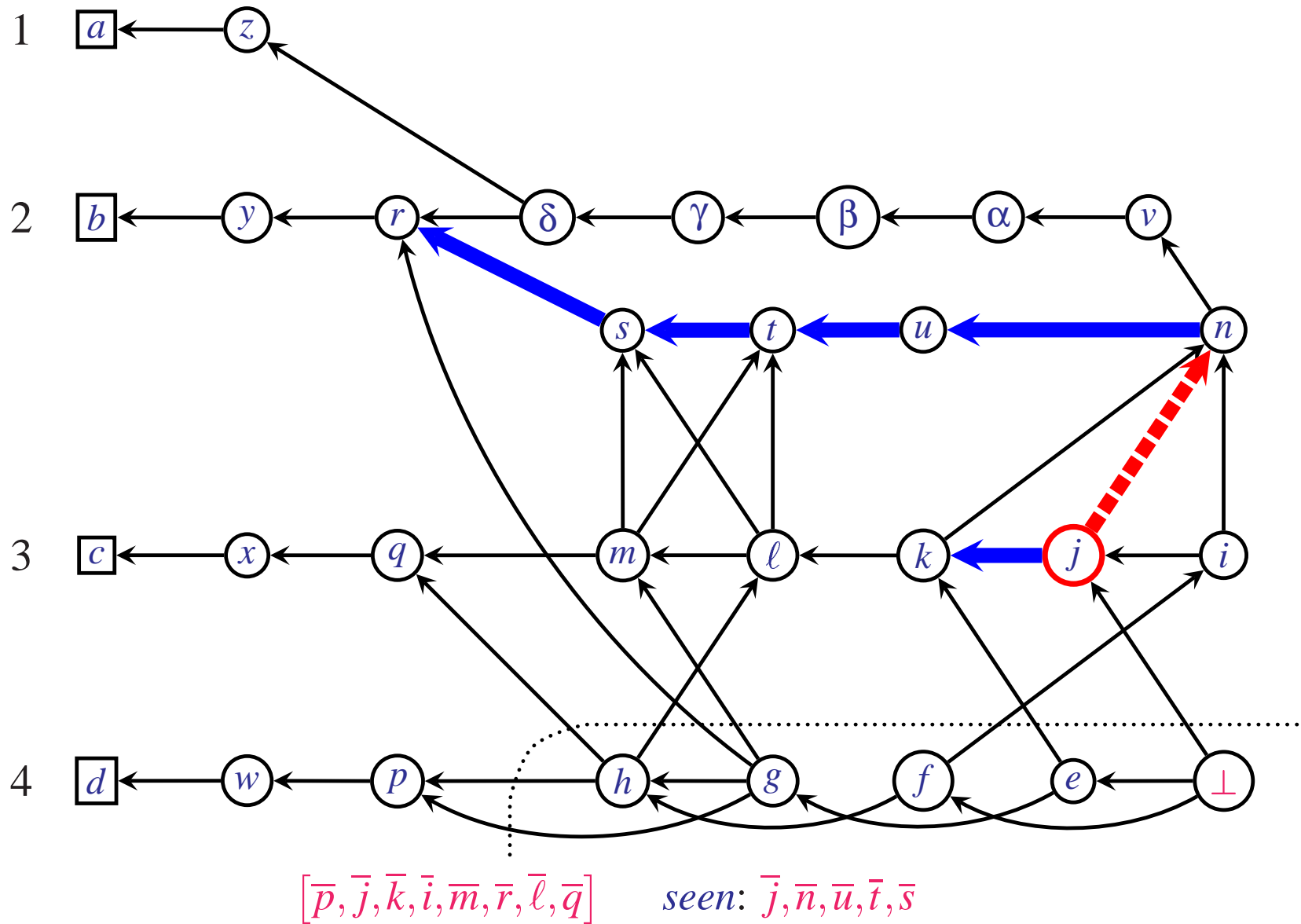
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



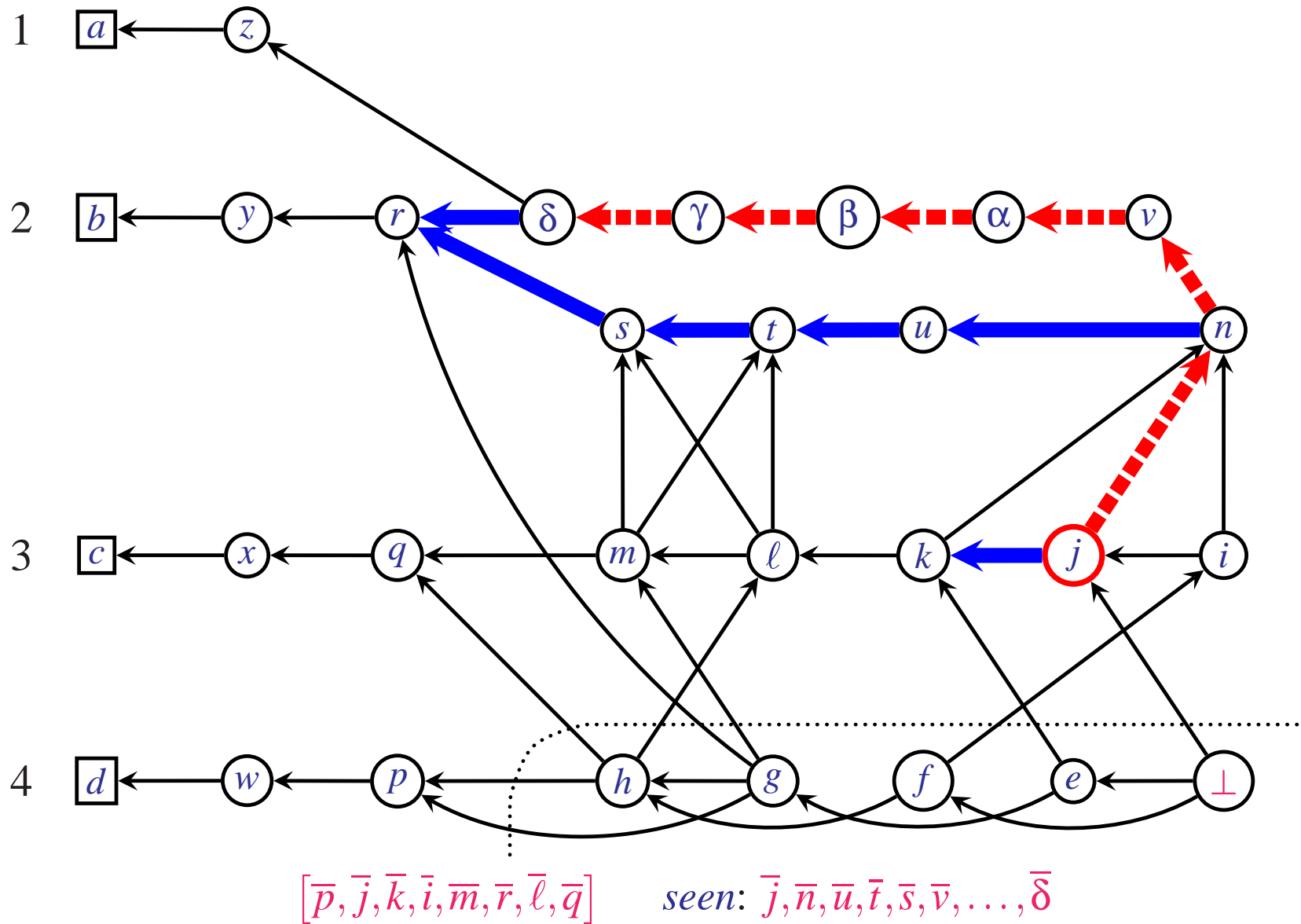
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



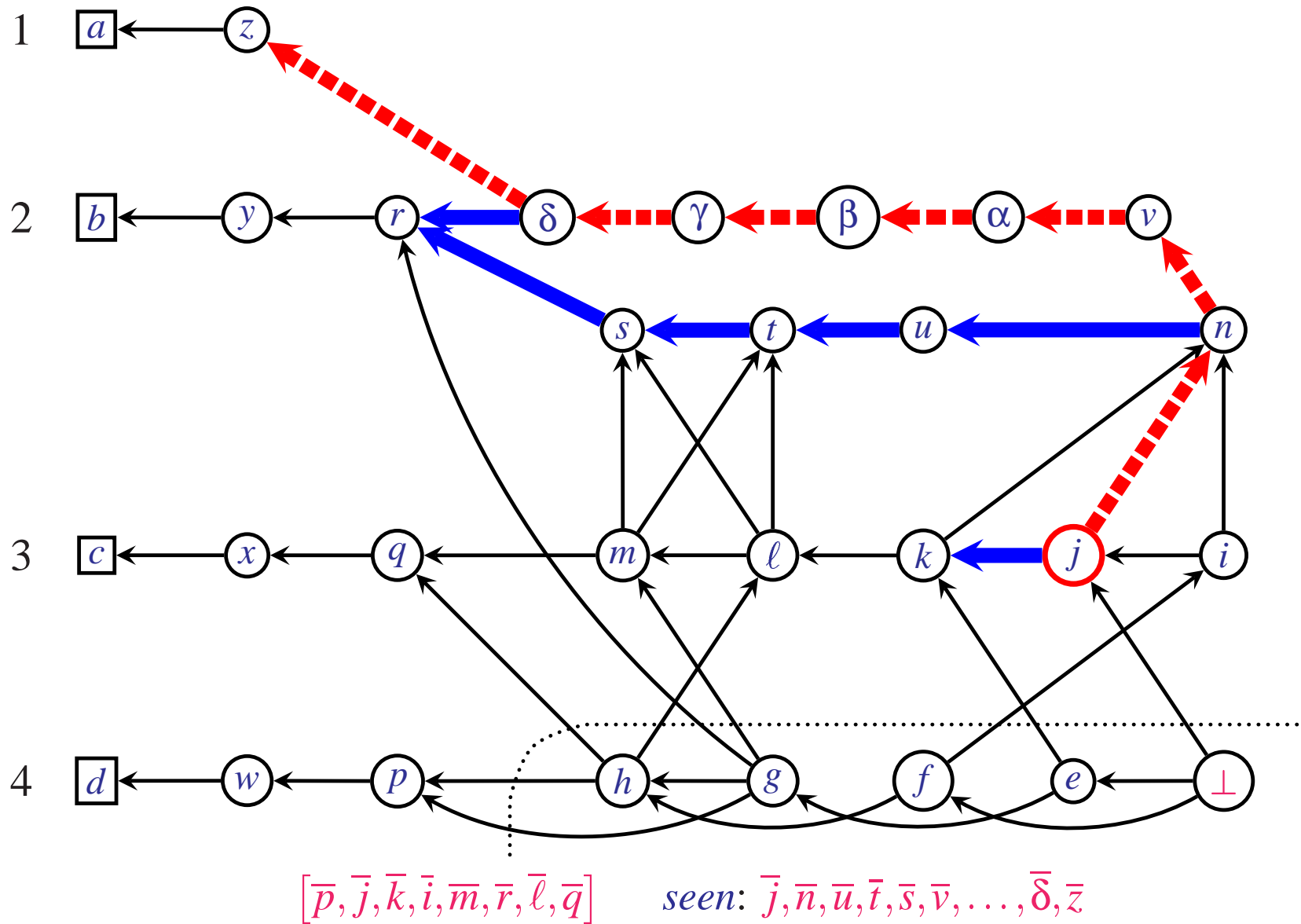
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



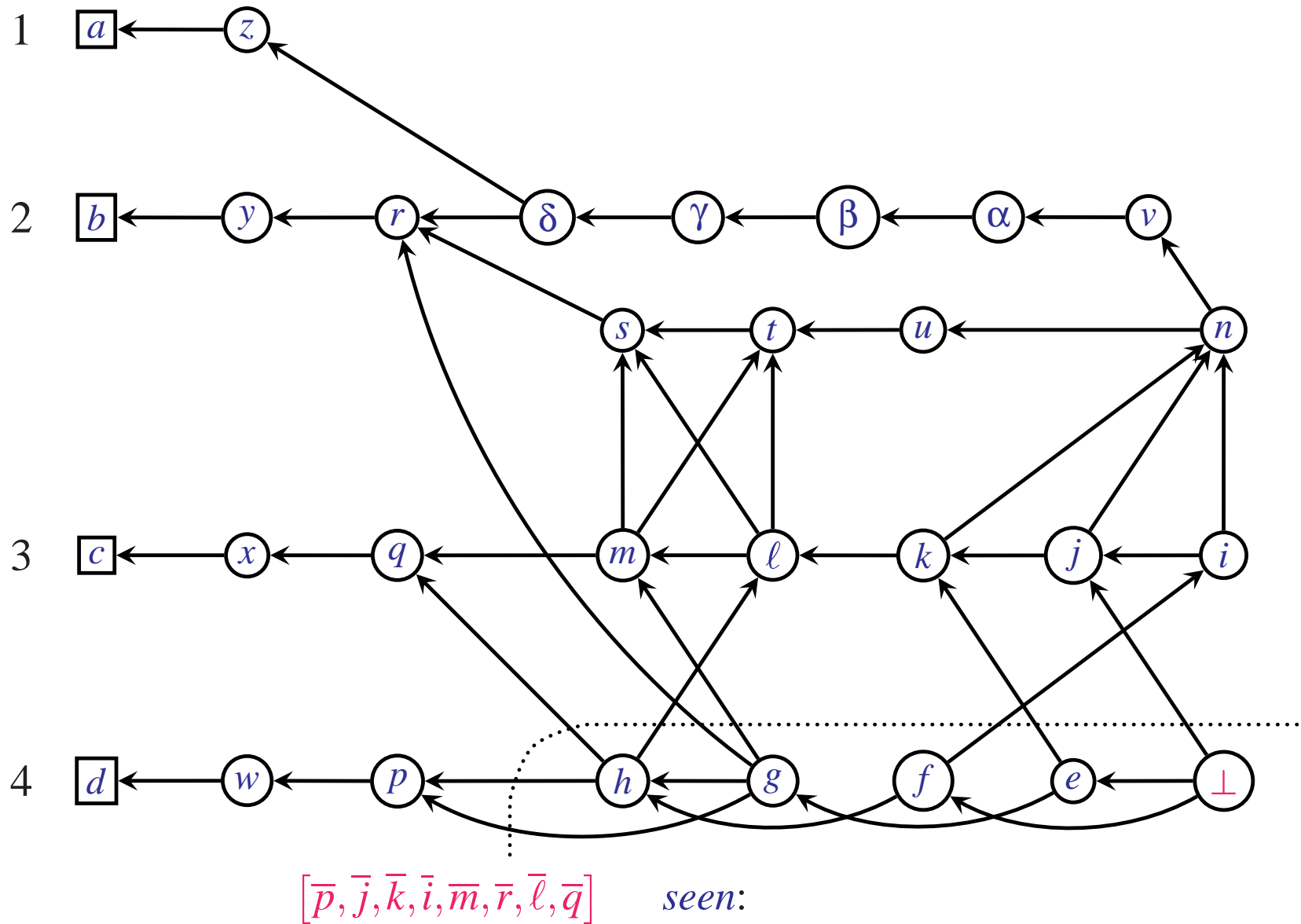
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



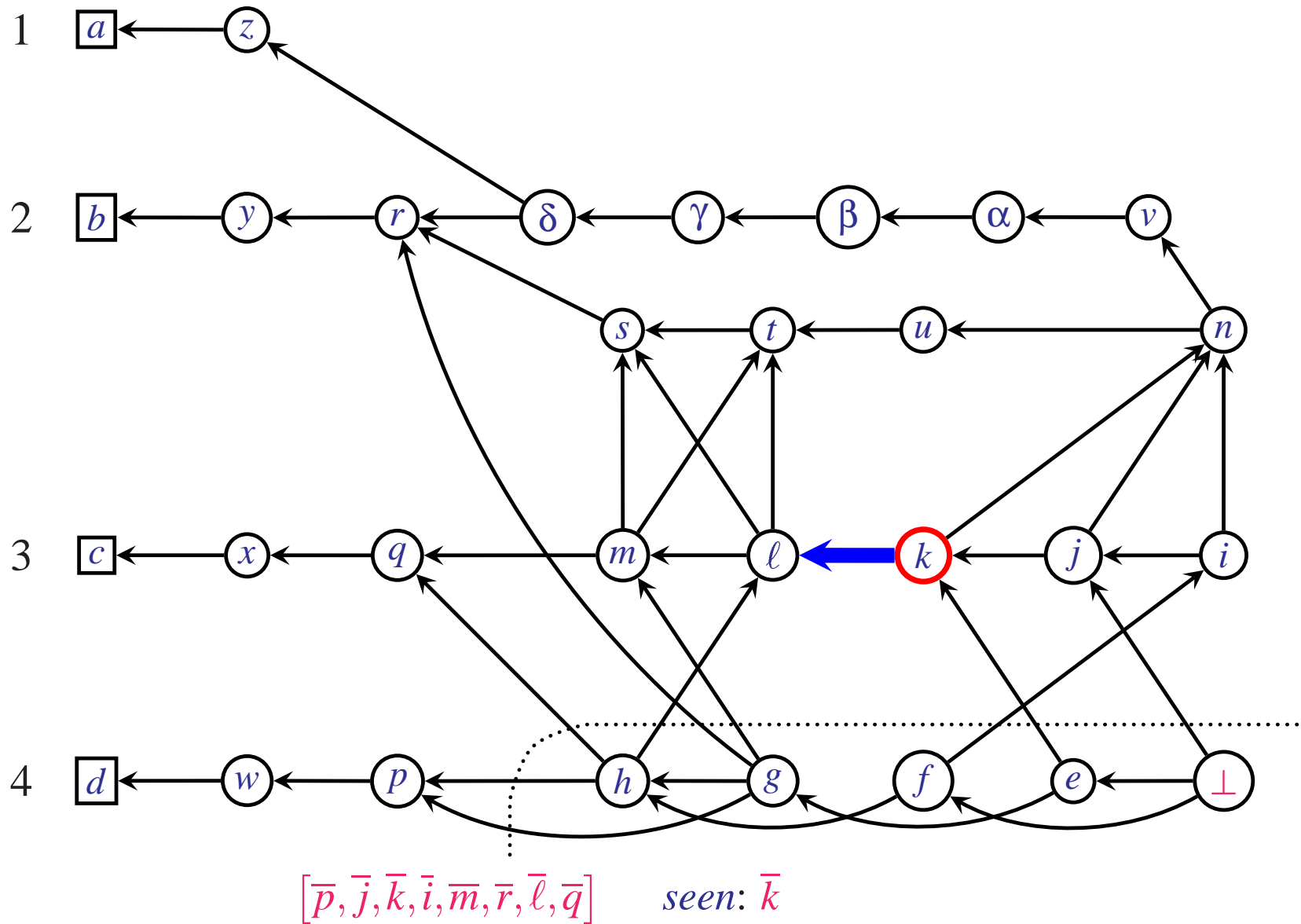
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



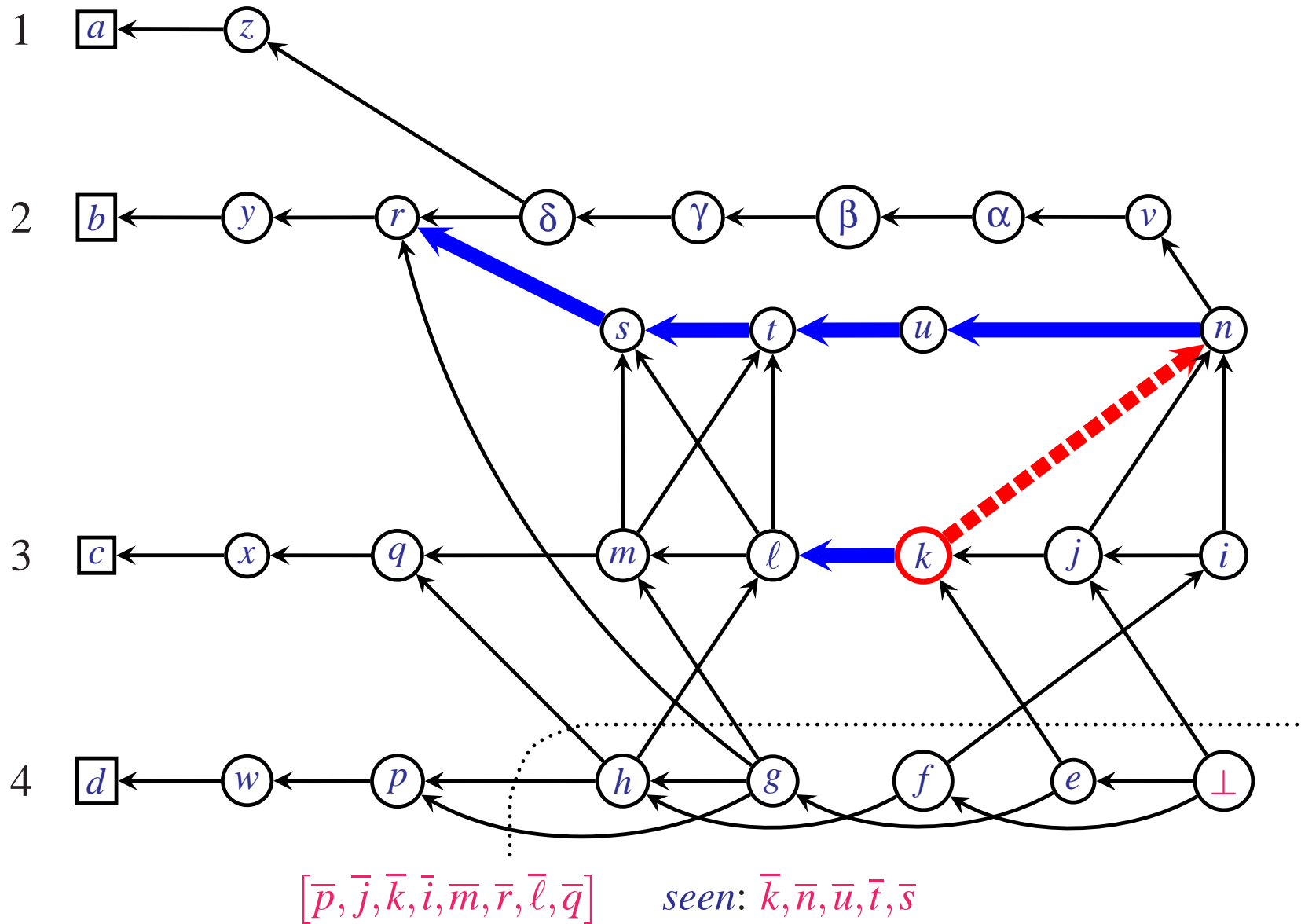
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



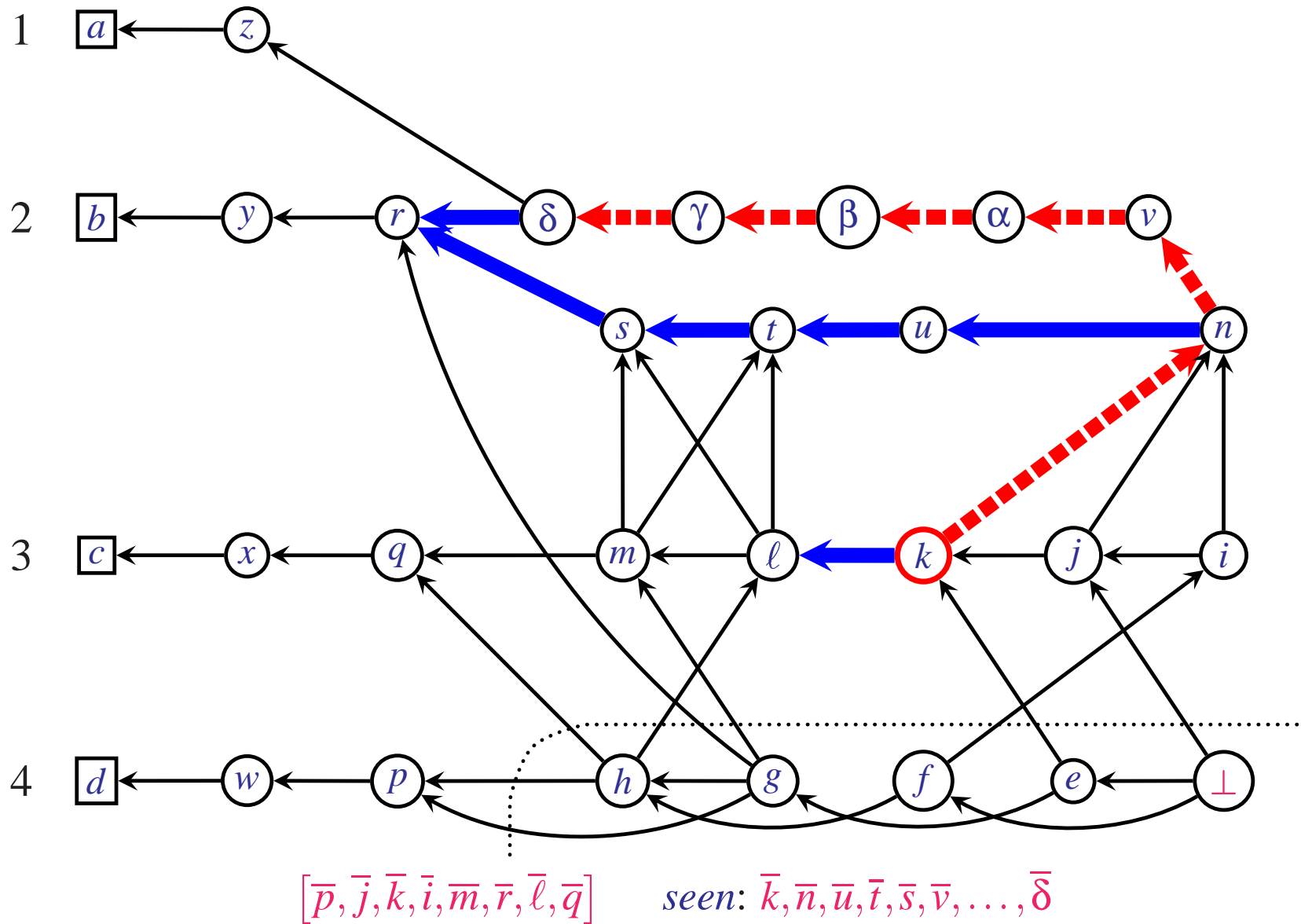
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



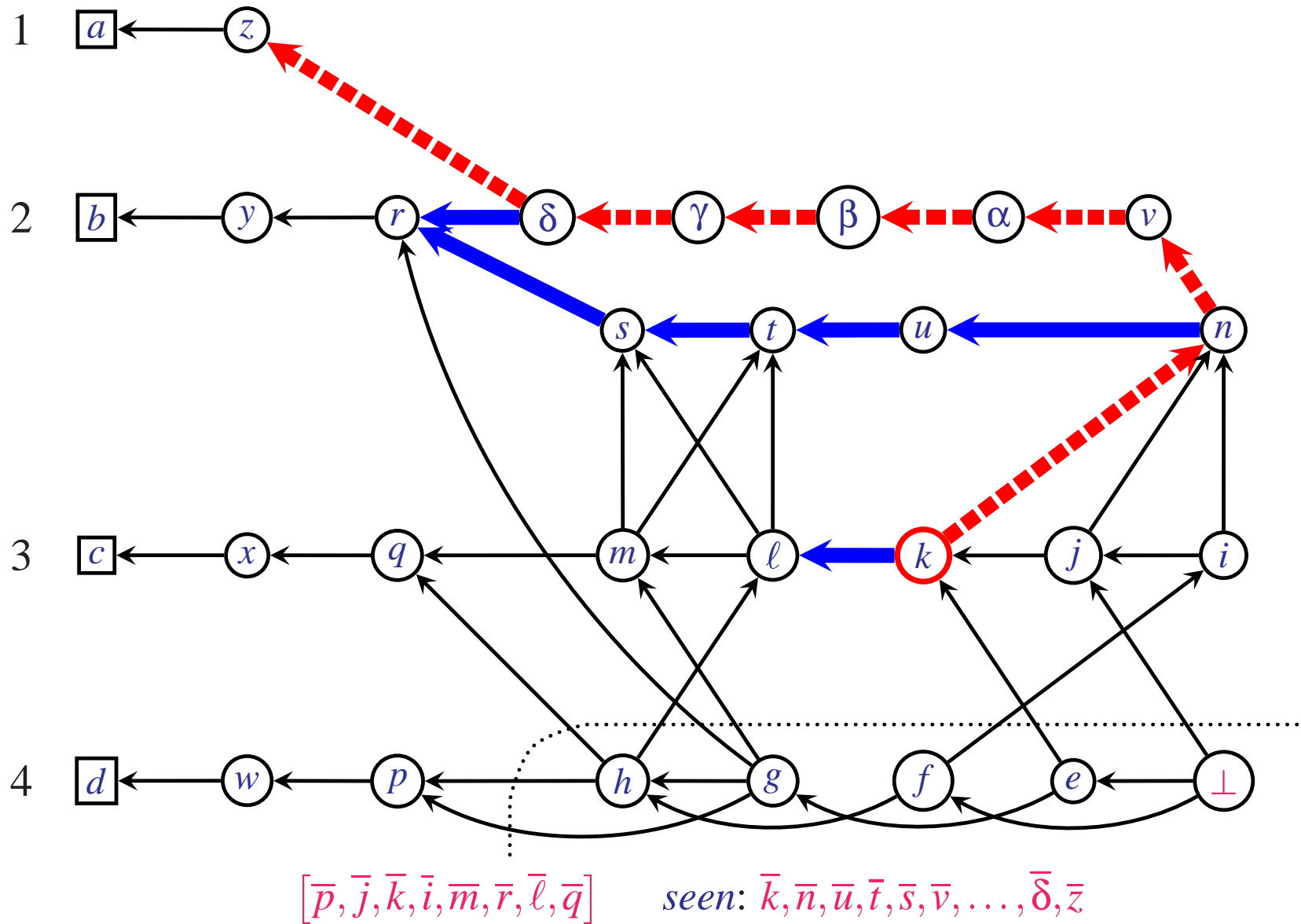
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



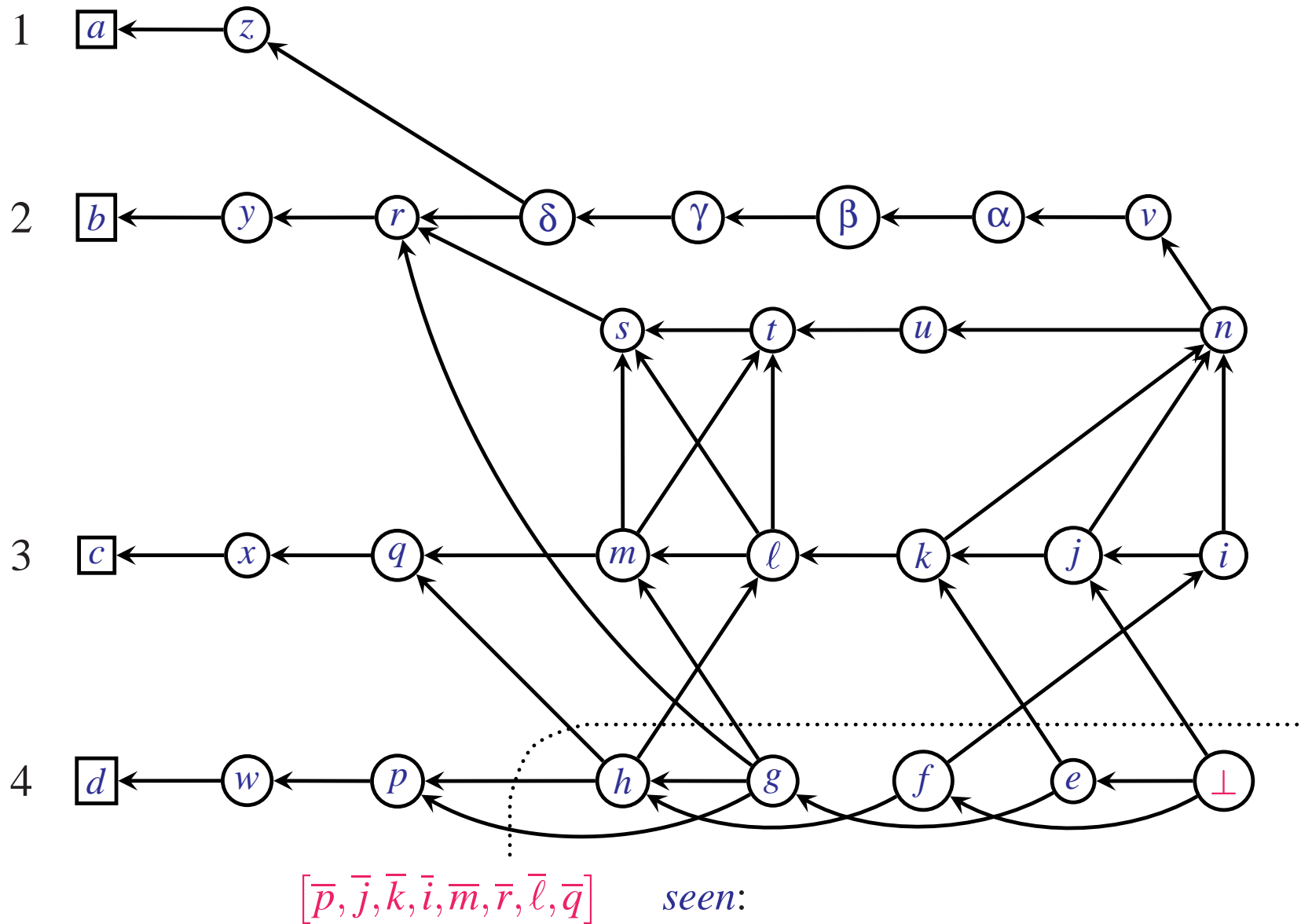
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



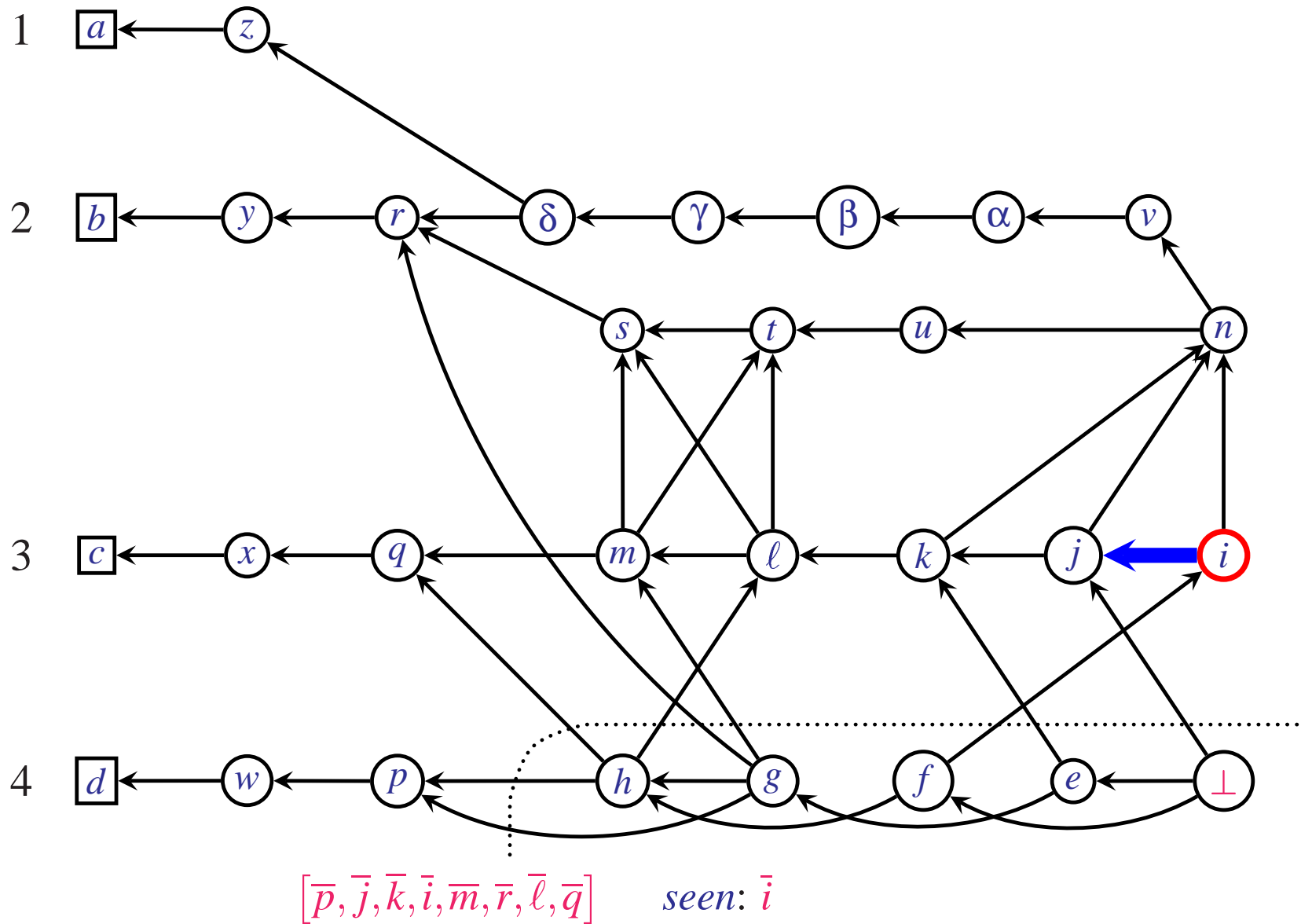
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



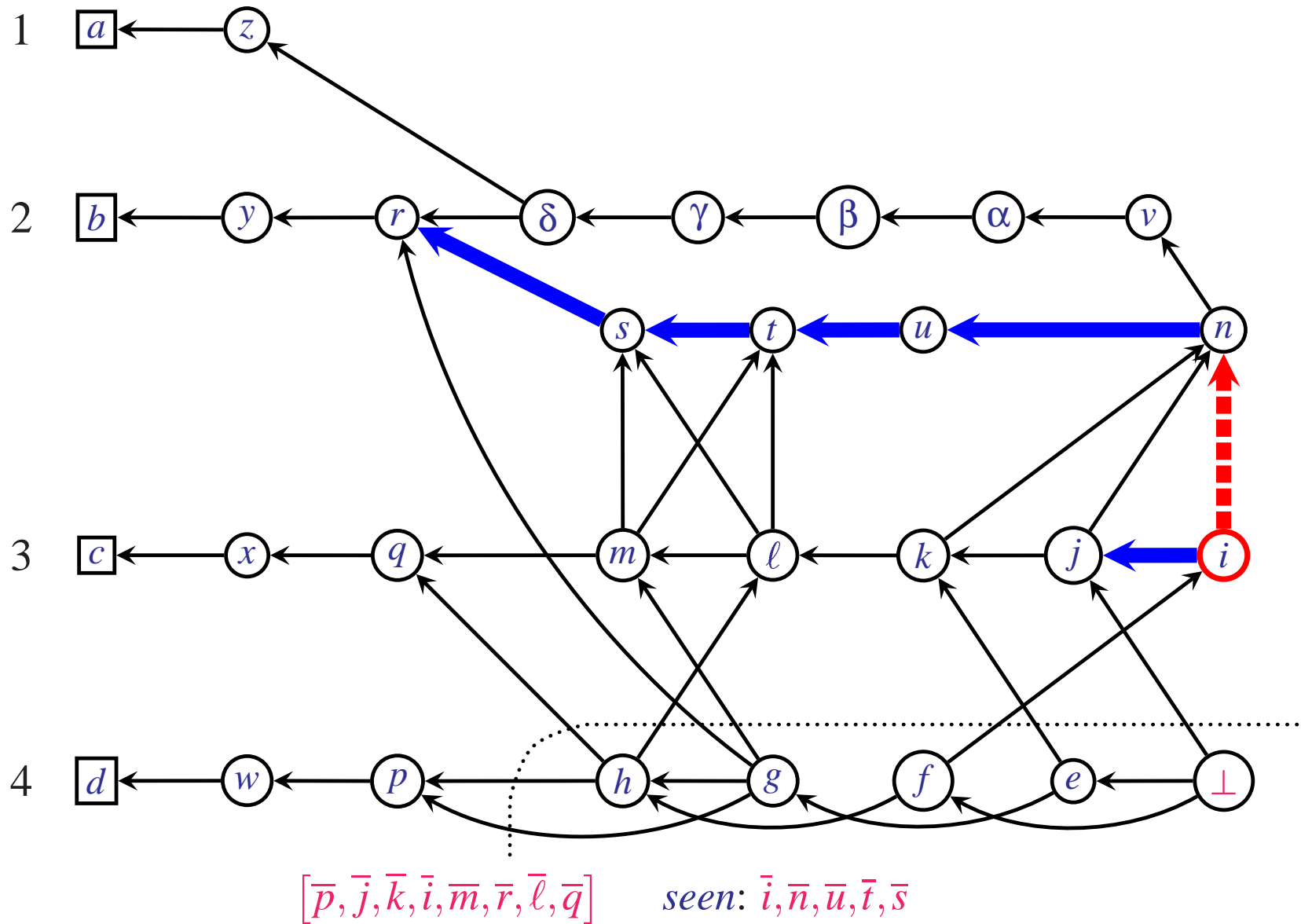
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



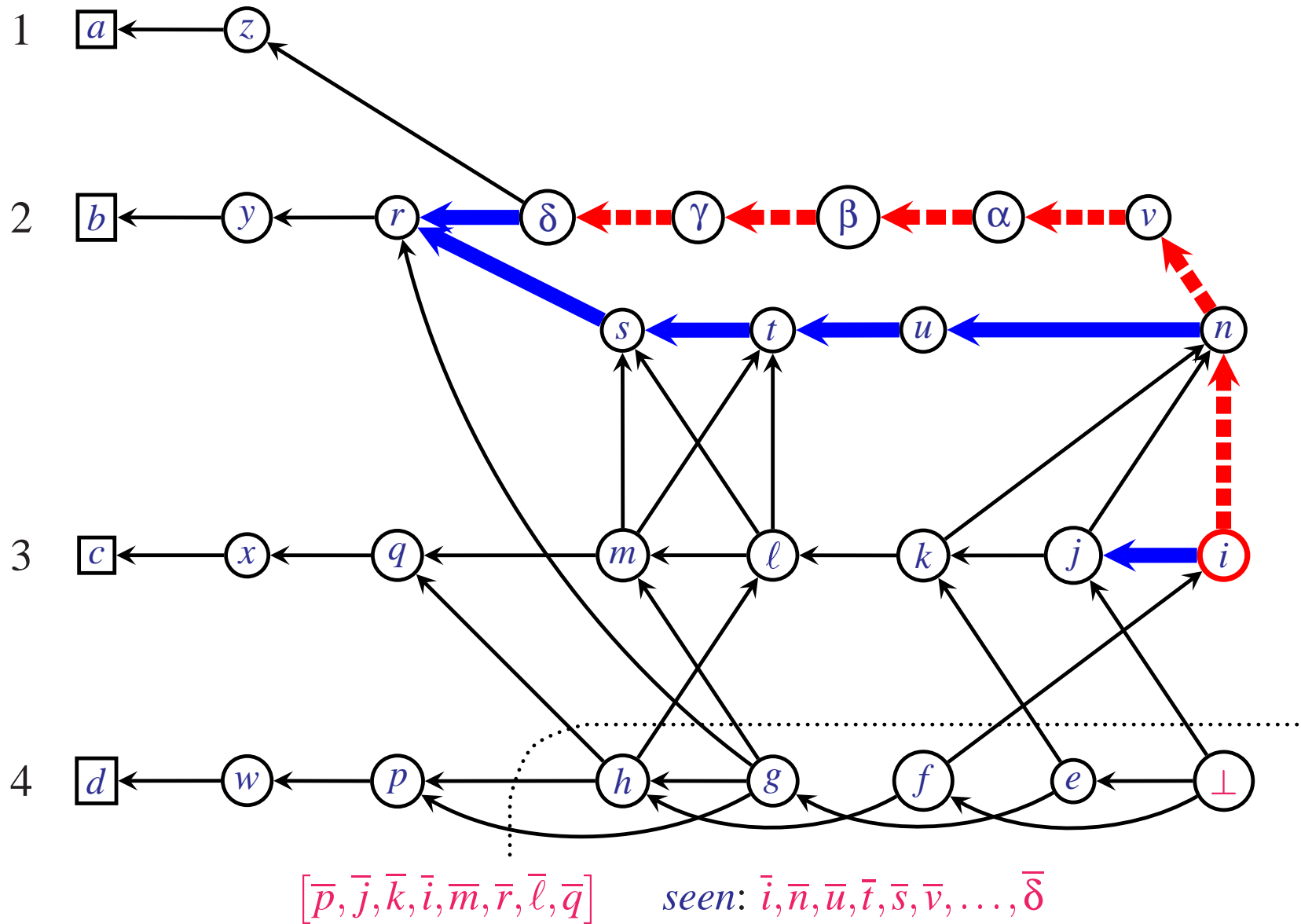
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



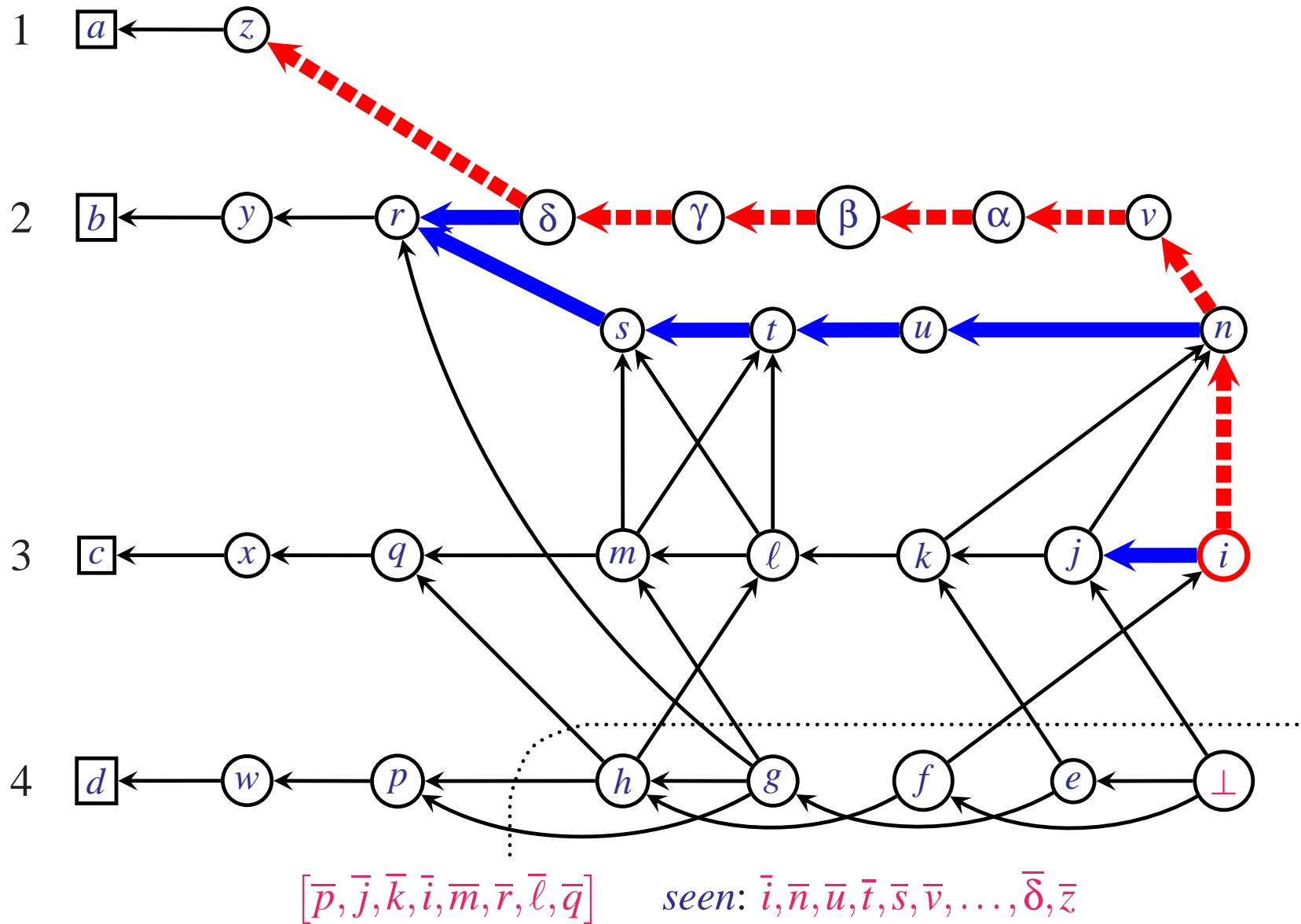
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



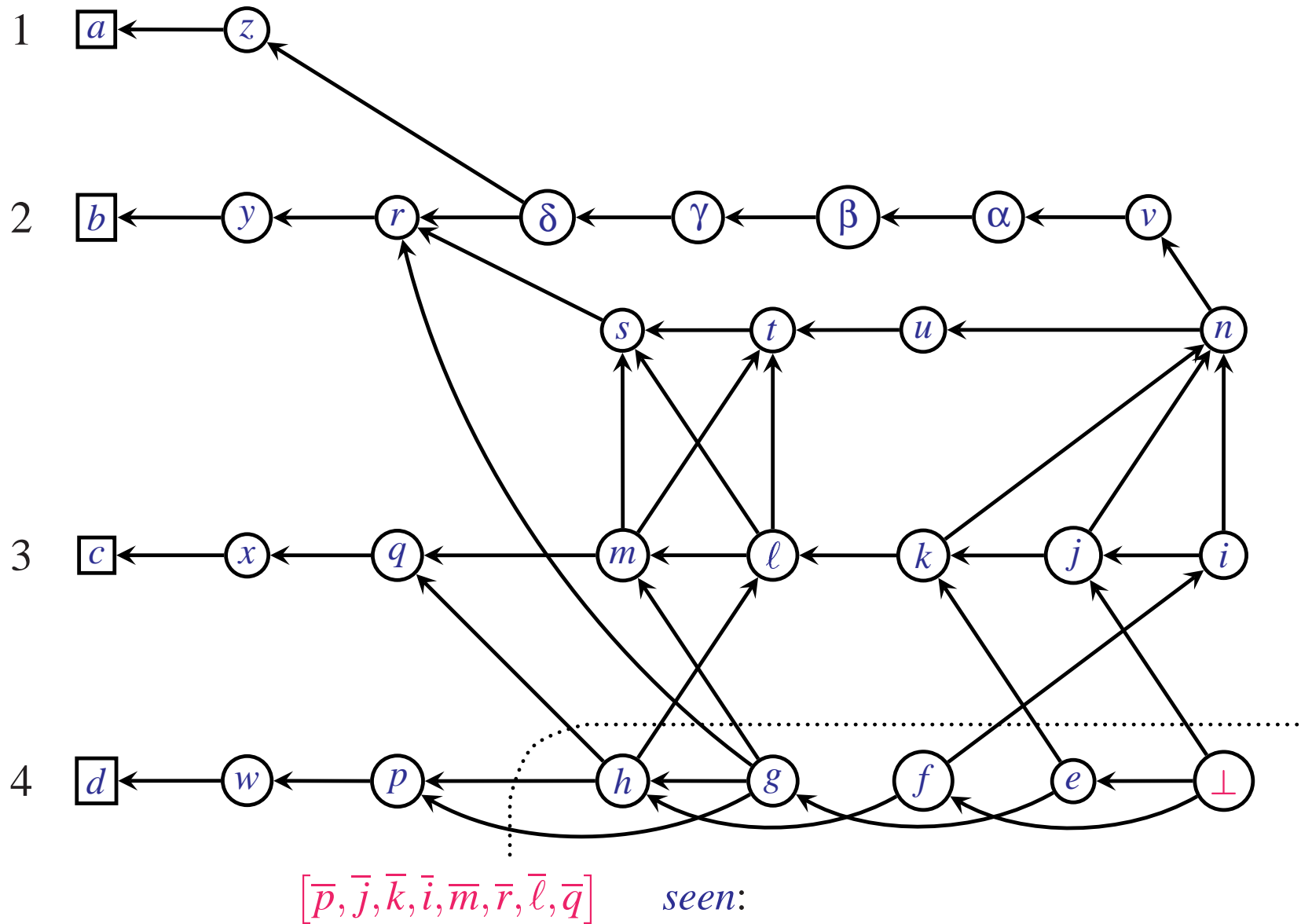
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



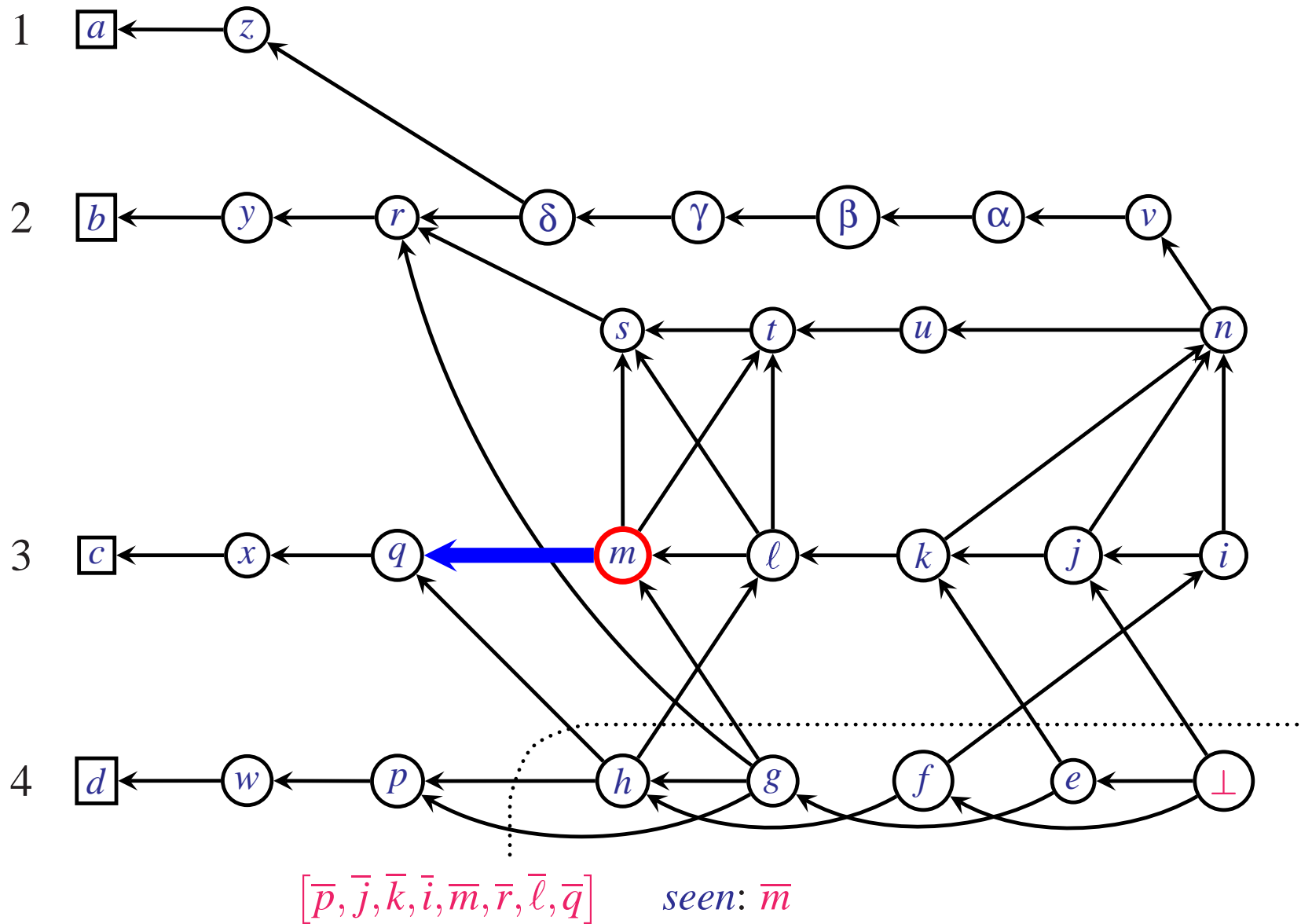
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



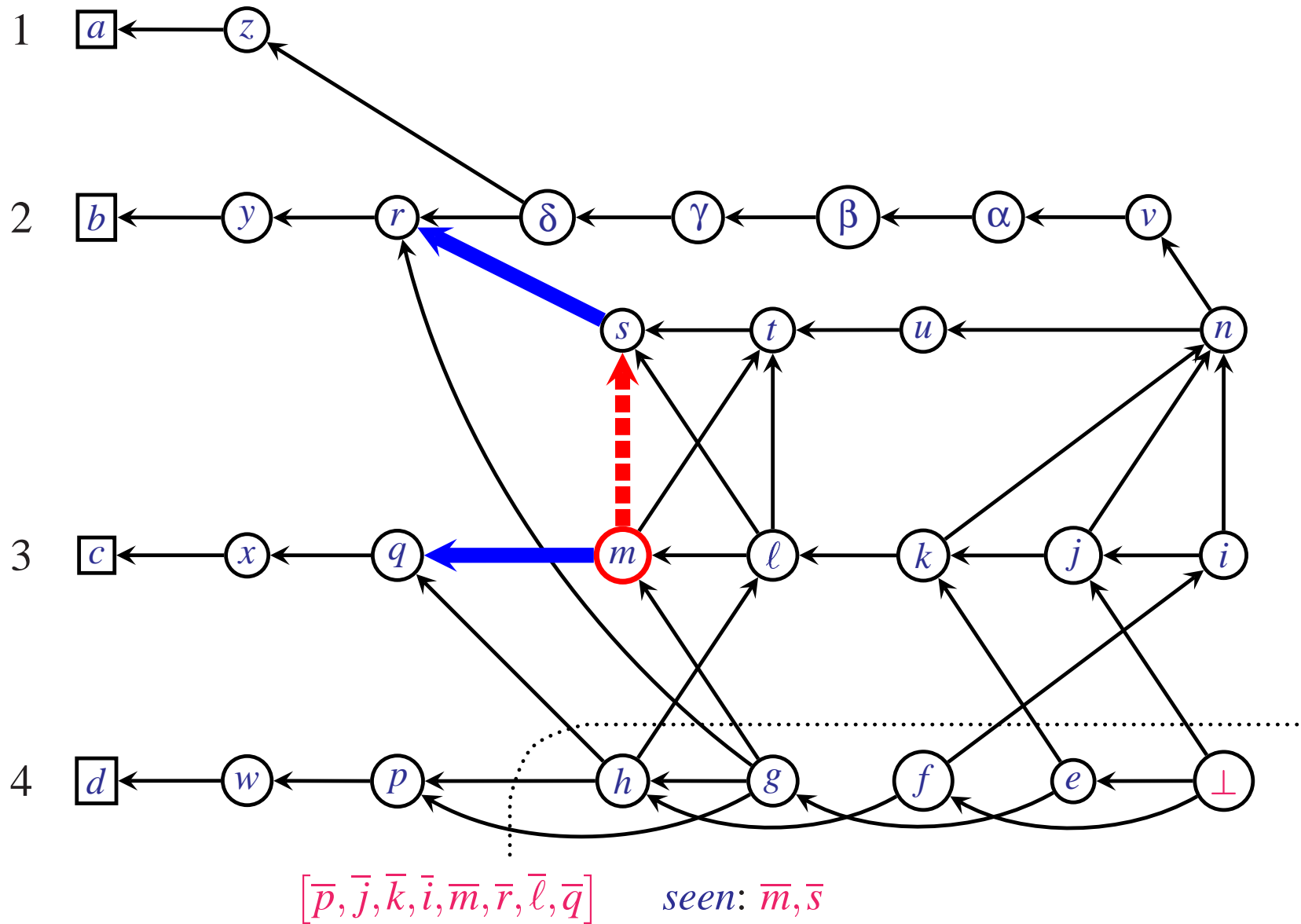
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



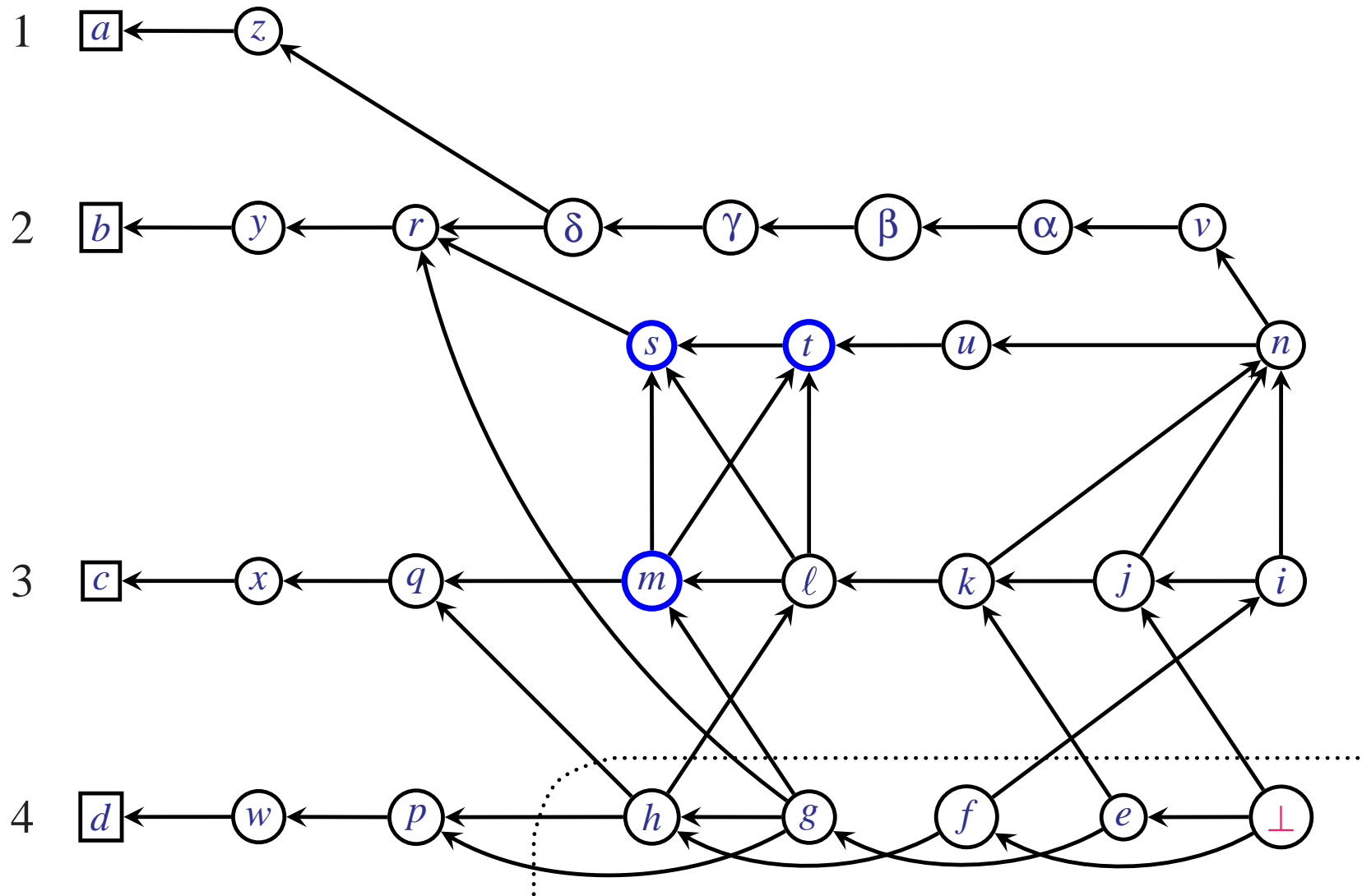
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization

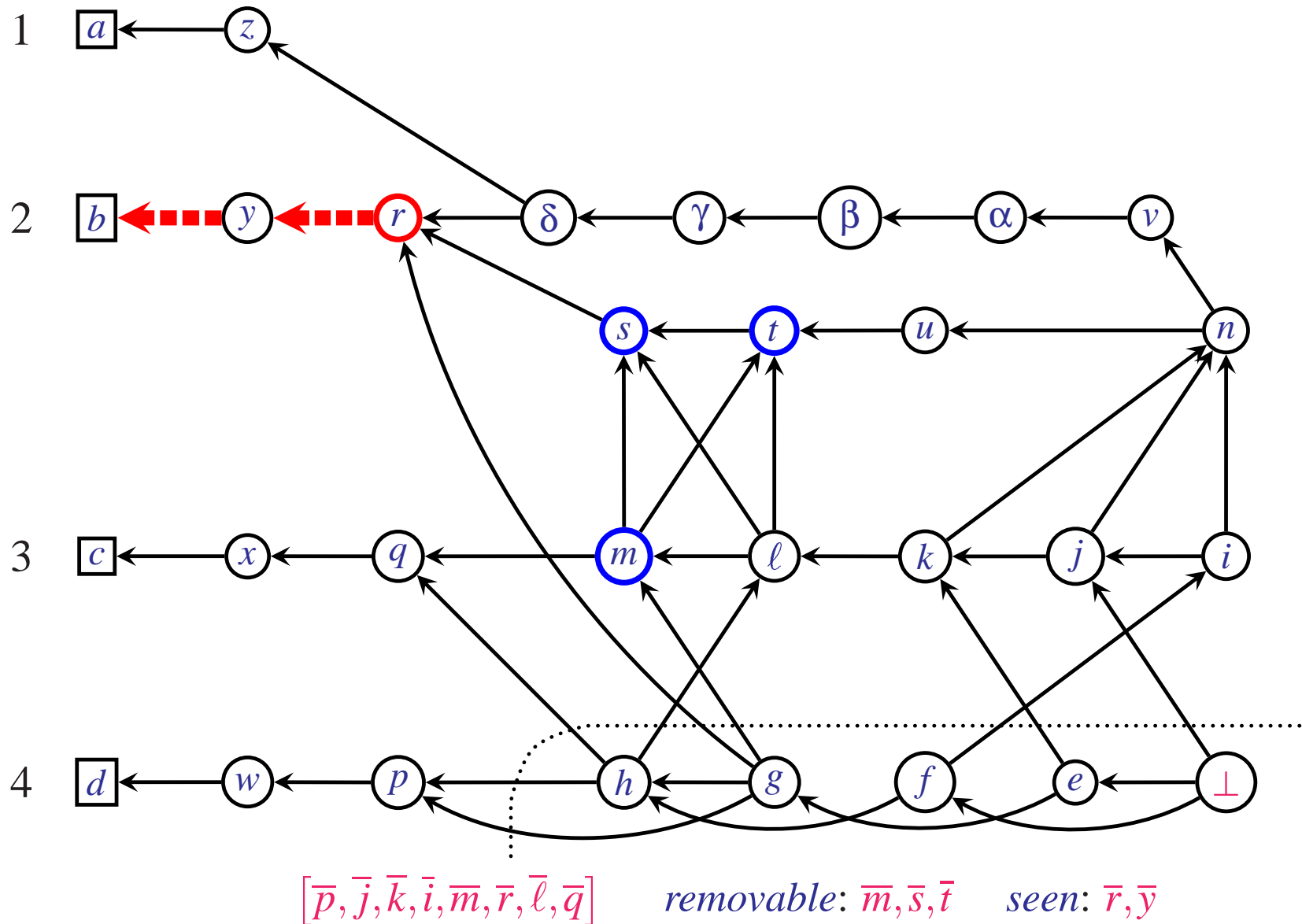


MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization

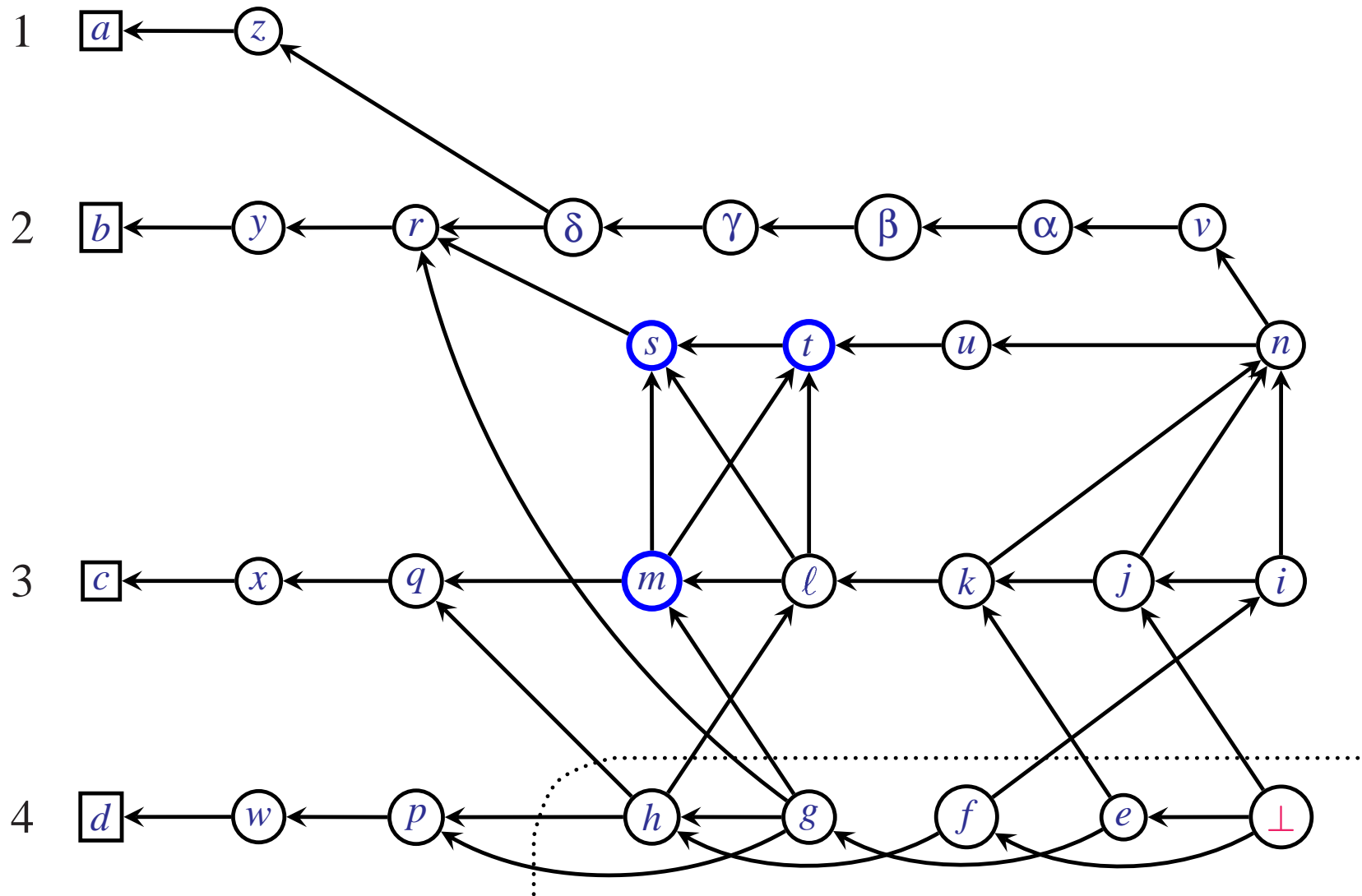


$[\bar{p}, \bar{j}, \bar{k}, \bar{i}, \bar{m}, \bar{r}, \bar{l}, \bar{q}]$ removable: $\bar{m}, \bar{s}, \bar{t}$ seen:

MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization

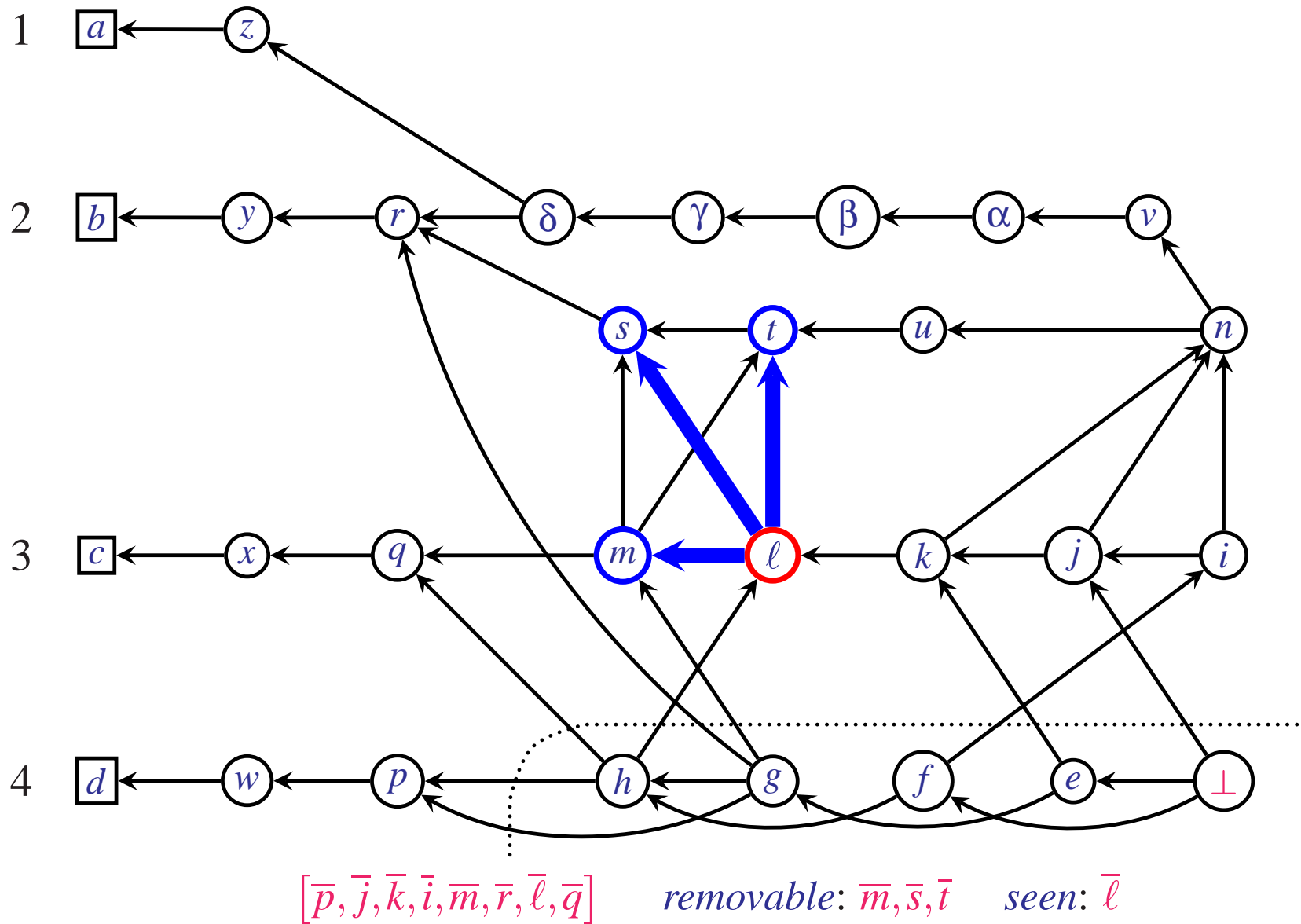


MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization

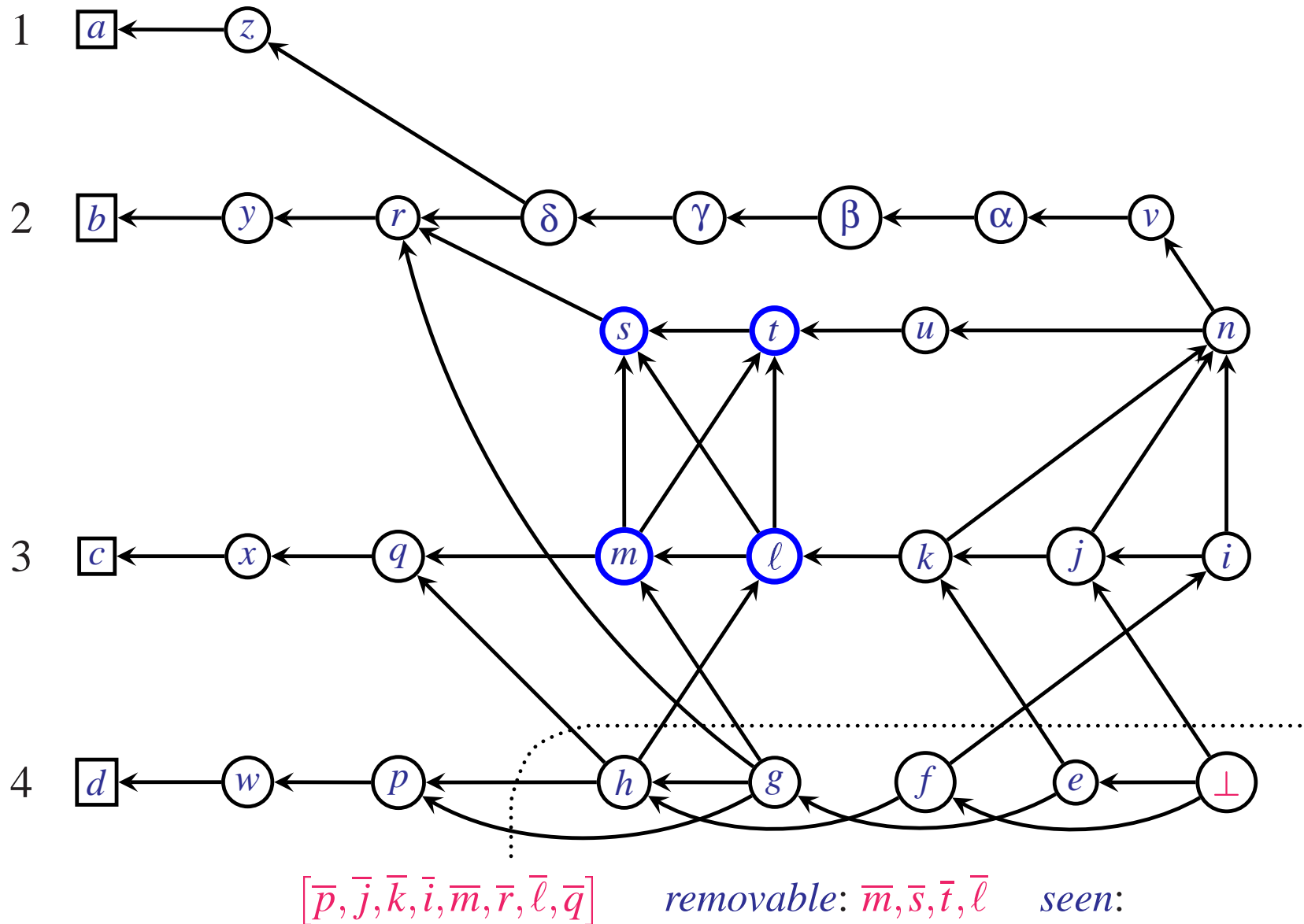


$[\bar{p}, \bar{j}, \bar{k}, \bar{i}, \bar{m}, \bar{r}, \bar{l}, \bar{q}]$ removable: $\bar{m}, \bar{s}, \bar{t}$ seen:

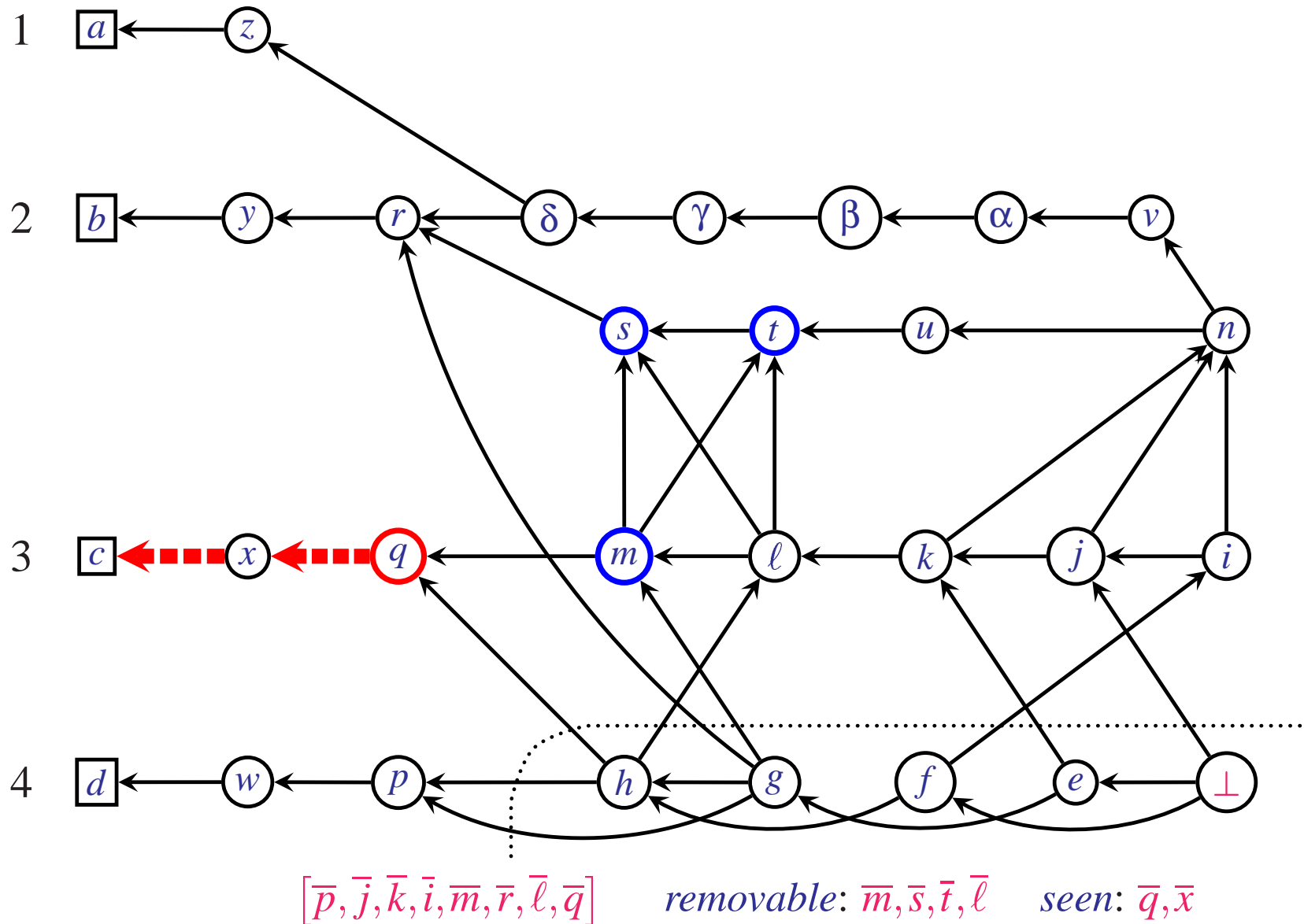
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



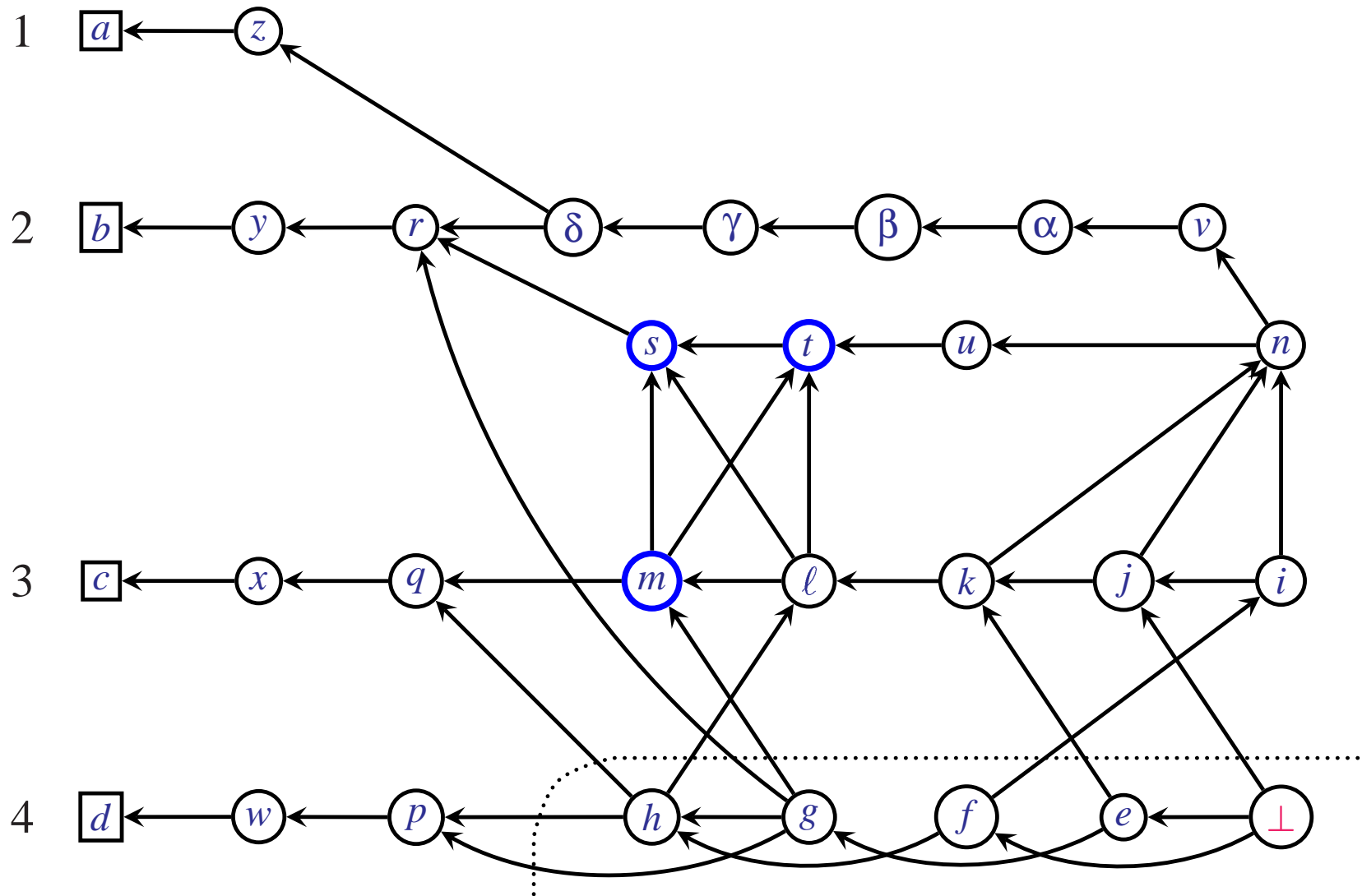
MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



MiniSat2.0 Recursive (Global, Expensive) Conflict Graph Minimization



$[\bar{p}, \bar{j}, \bar{k}, \bar{i}, \bar{m}, \bar{r}, \bar{l}, \bar{q}]$

removable: $\bar{m}, \bar{s}, \bar{t}, \bar{l}$ Order no good for TVR!

What Was the Fix? Recursive Depth-First Search

DFSearch(v)

Mark vertex v discovered

Pre-order process v

For each w adjacent to v

 If w is not seen

 DFSearch(w)

 useInfo(w)

Post-order process v

storeInfo(v)

Mark v finished

return

← If v is *not* a decision literal, v is *removable*, else ...
if v is *not* in the conflict clause, v is *poison*;
if v is in the conflict clause, v is *keep*;

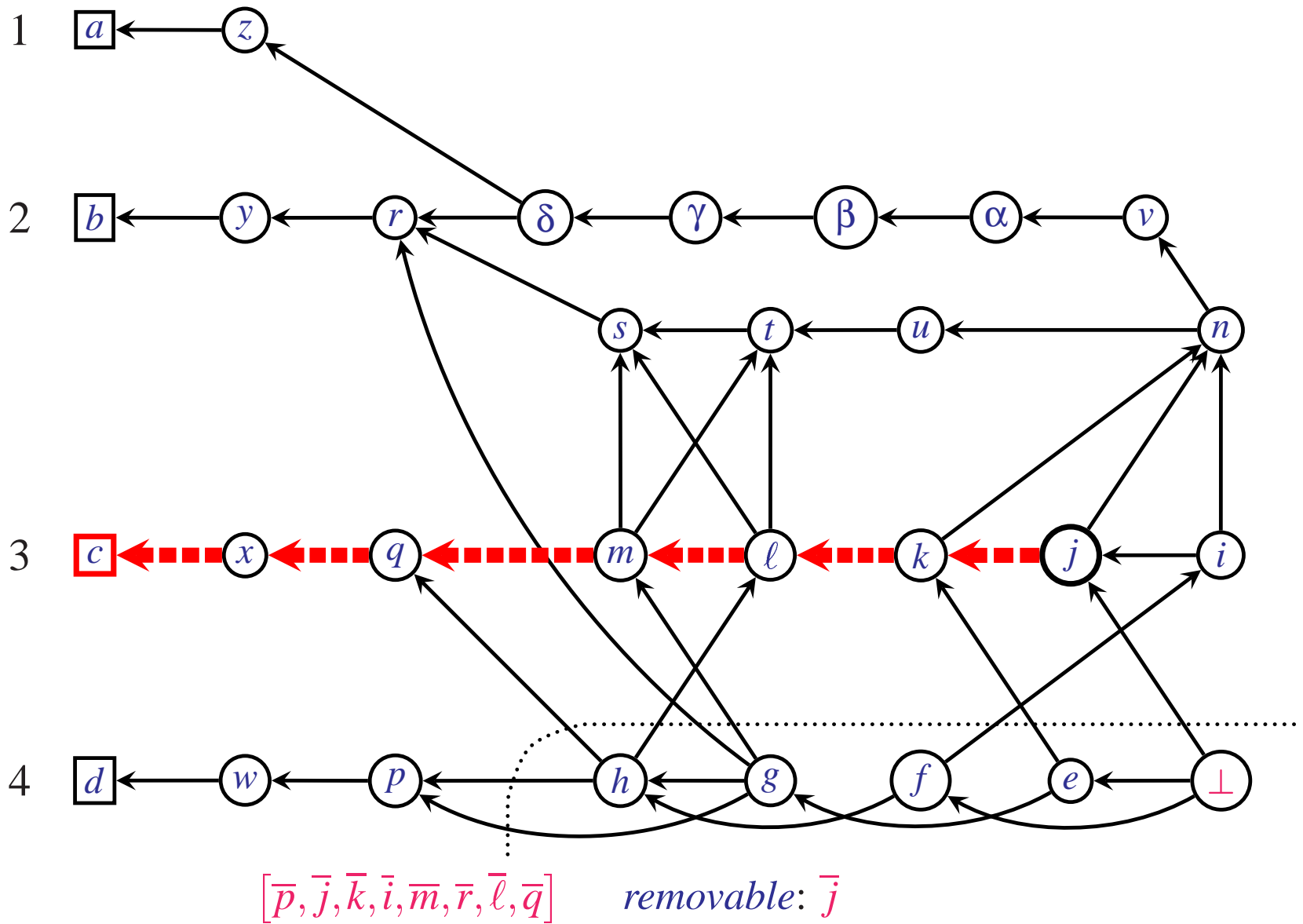
← If w is *poison* and v is *not* in the conflict clause,
 v is *poison*.

If w is *poison* and v is *in* the conflict clause,
 v is *keep*.

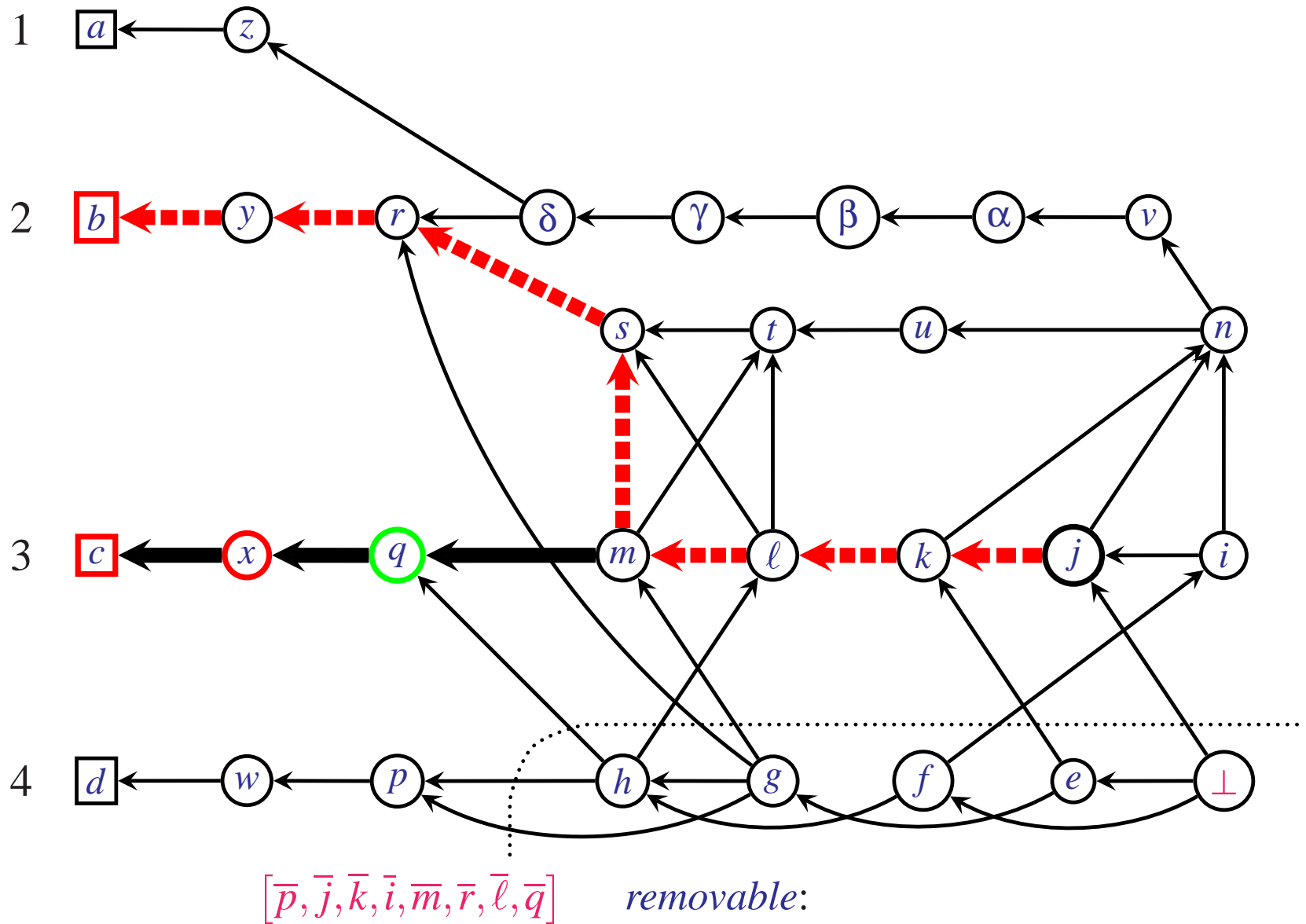
← If v is *removable*, push(v , *removableStack*)

← Record whether v is *removable*, *keep*, or *poison*

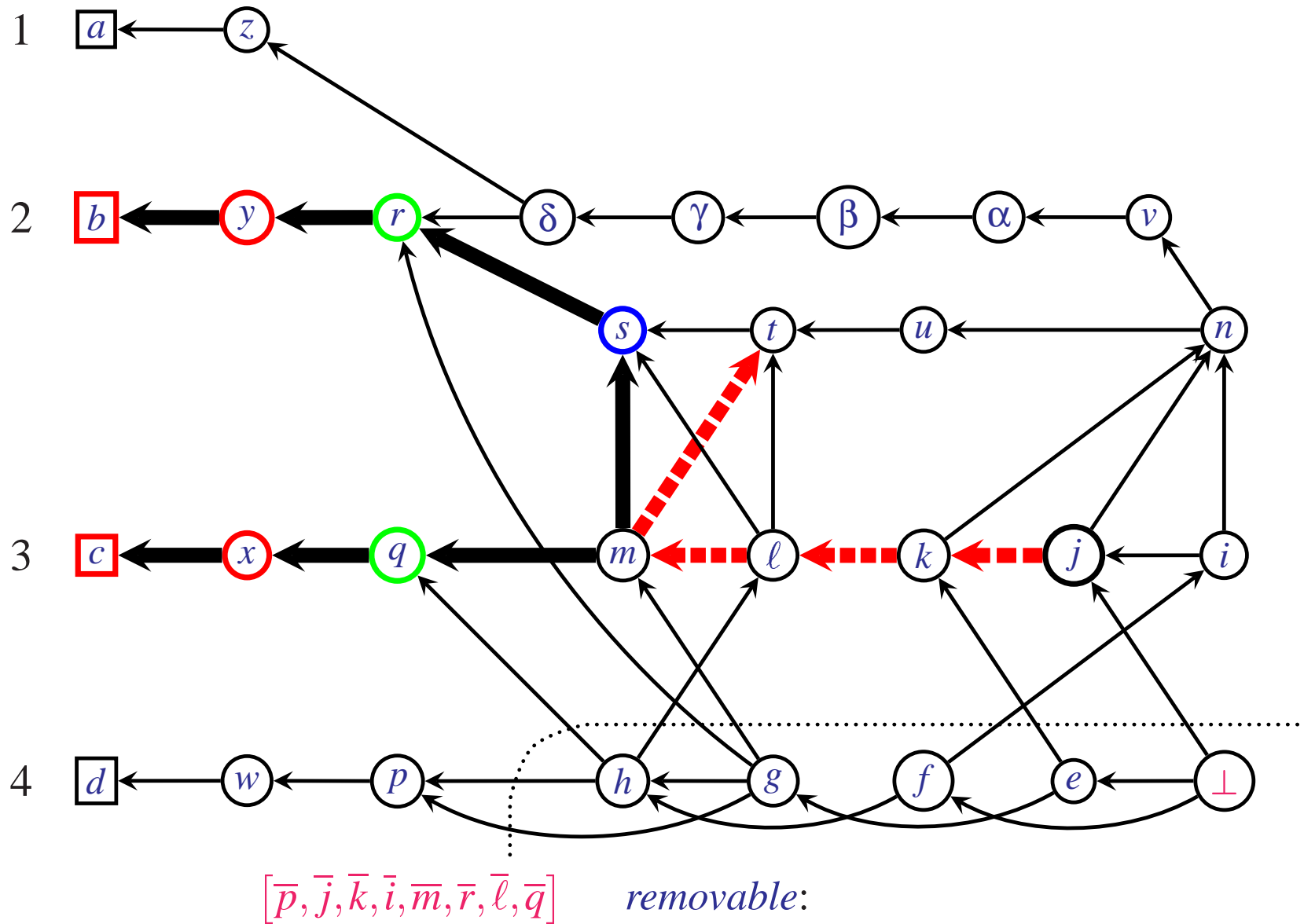
New Recursive (Global, Expensive) Conflict Graph Minimization



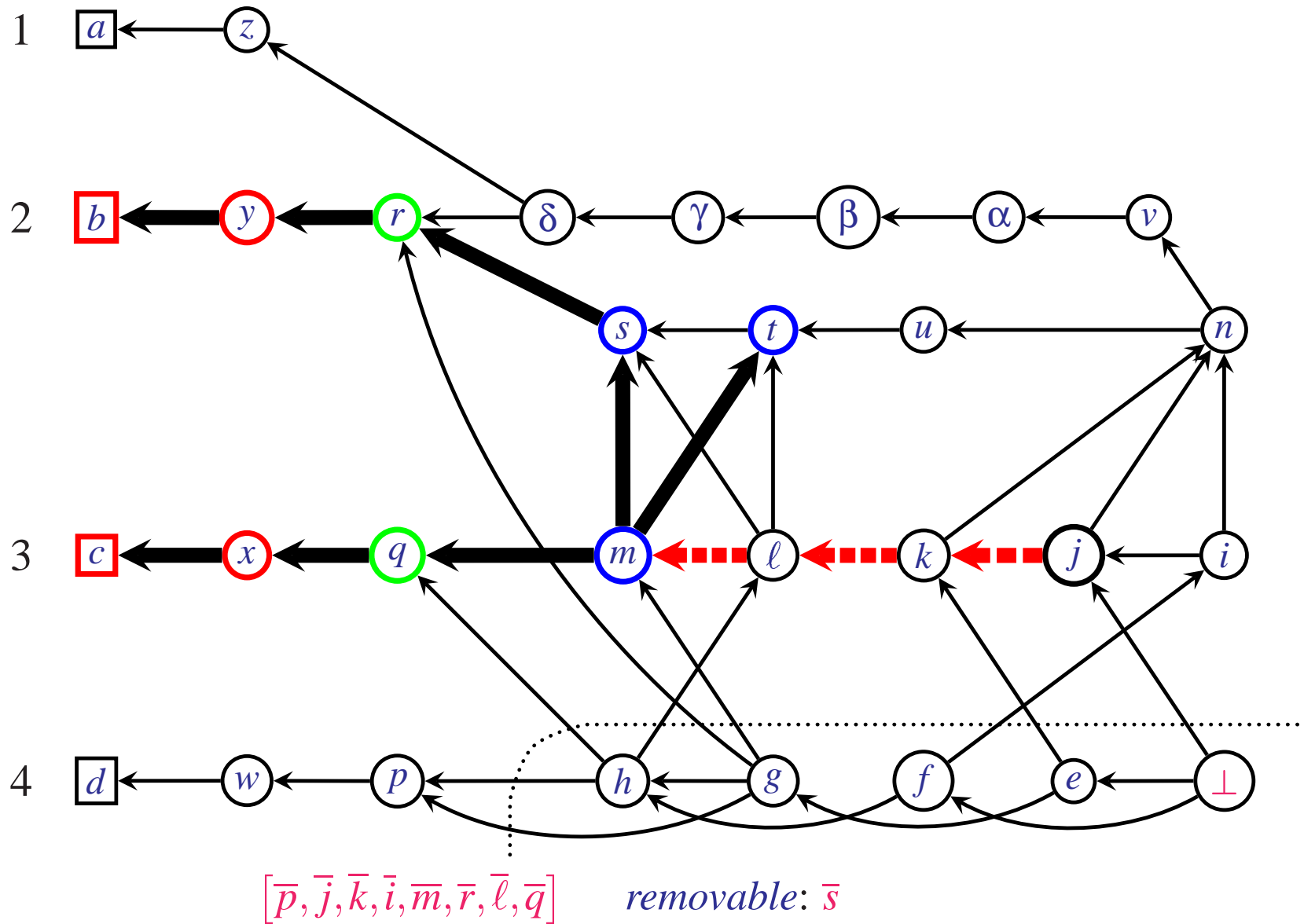
New Recursive (Global, Expensive) Conflict Graph Minimization



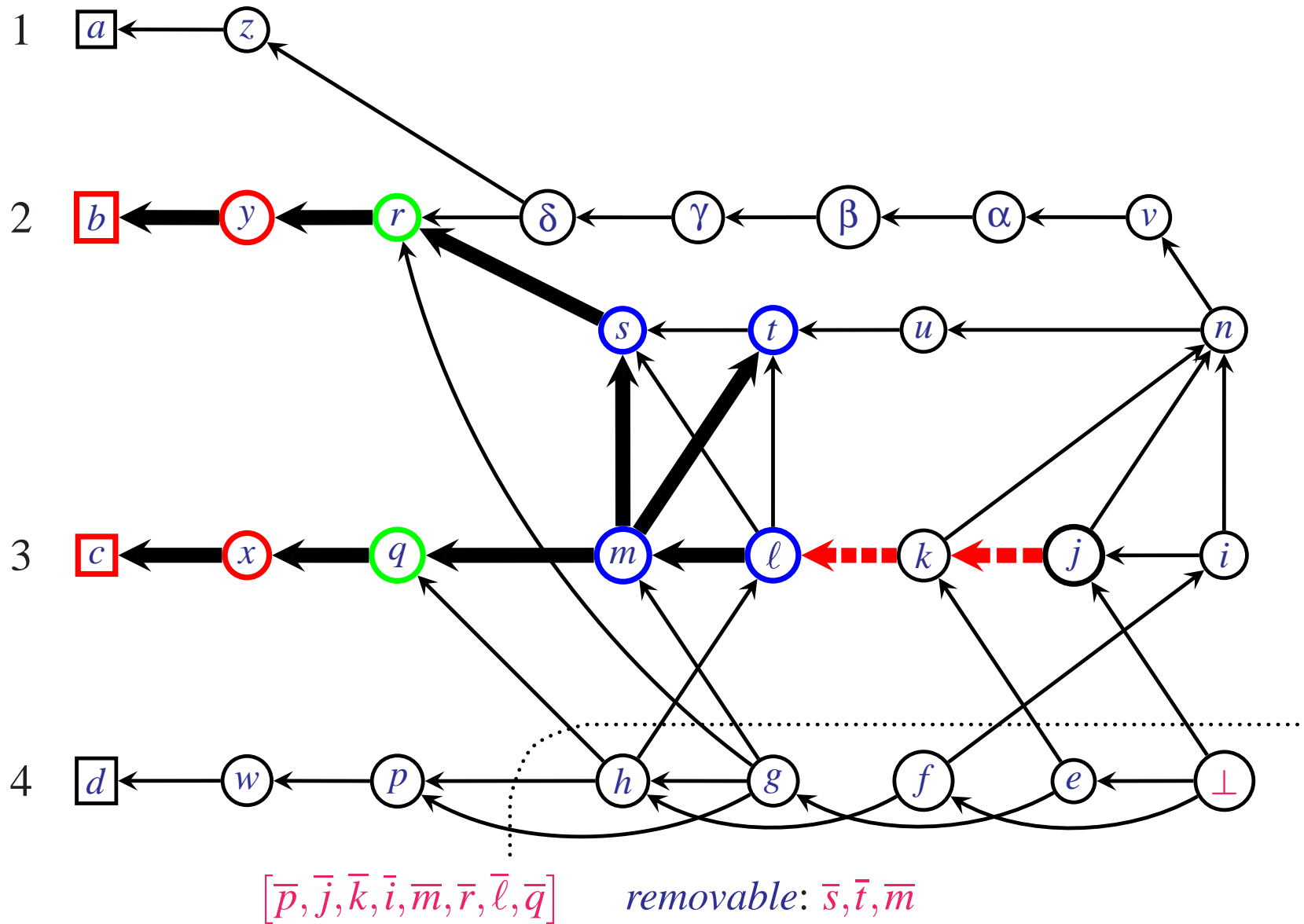
New Recursive (Global, Expensive) Conflict Graph Minimization



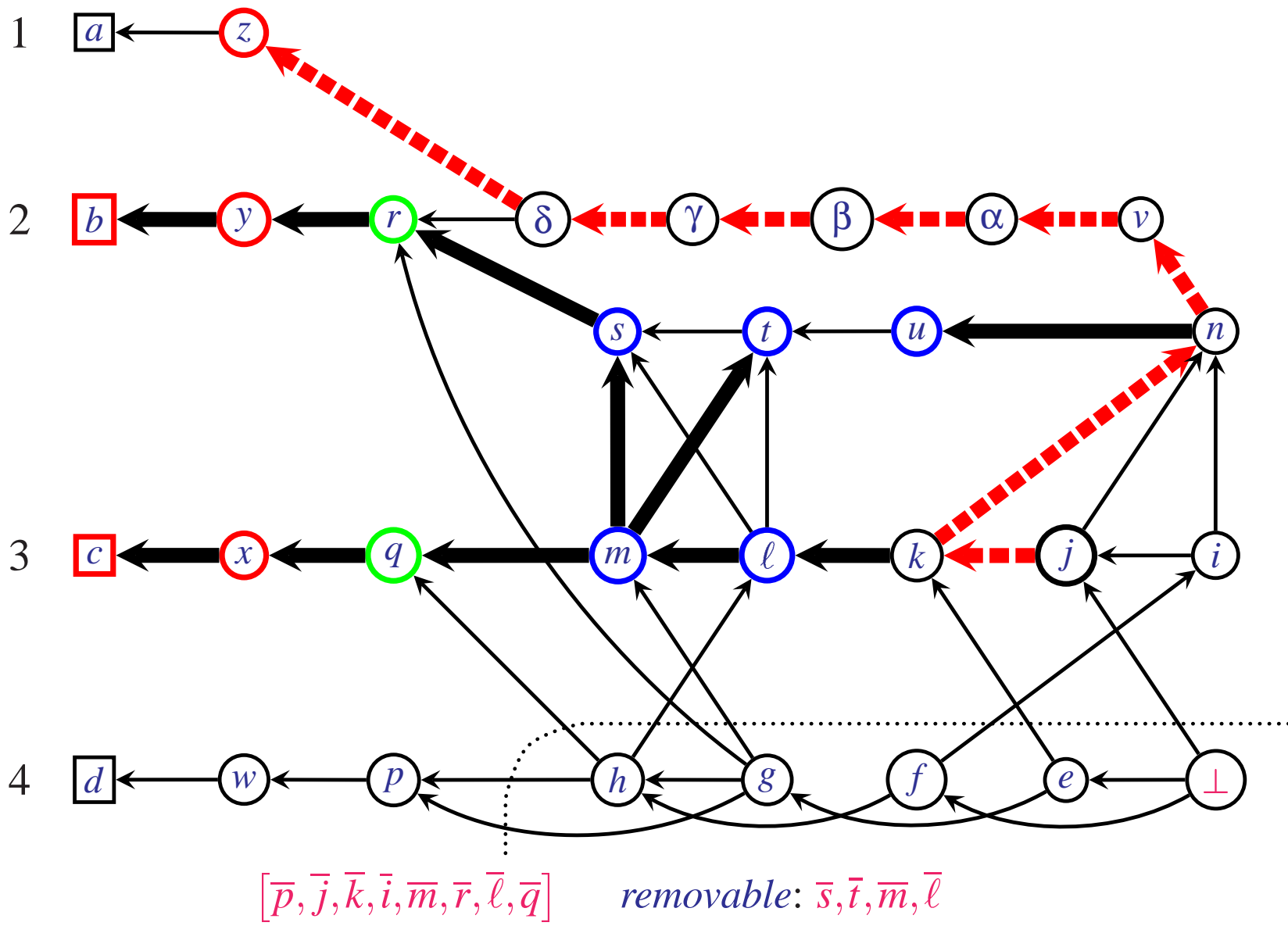
New Recursive (Global, Expensive) Conflict Graph Minimization



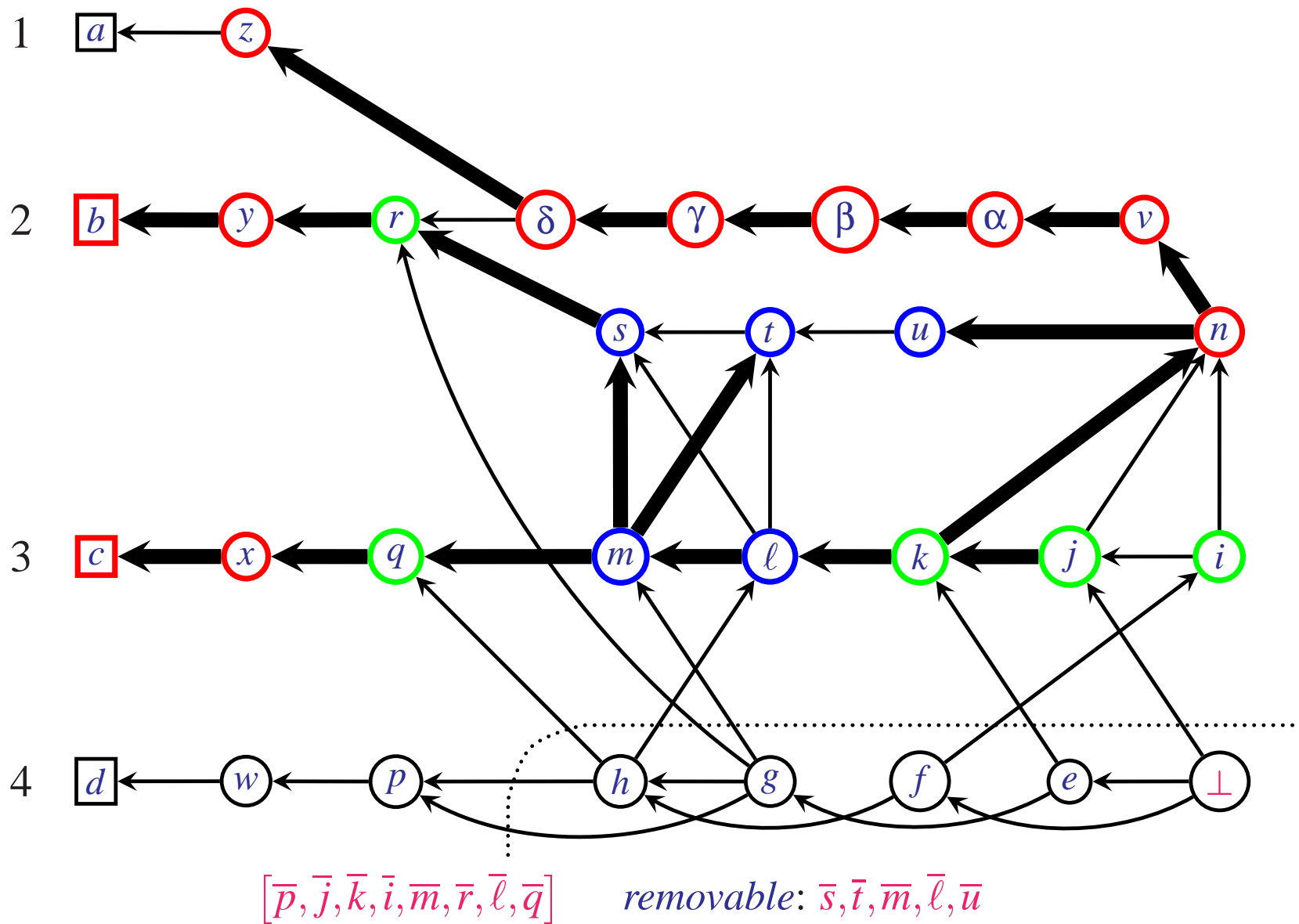
New Recursive (Global, Expensive) Conflict Graph Minimization



New Recursive (Global, Expensive) Conflict Graph Minimization



New Recursive (Global, Expensive) Conflict Graph Minimization



How Did It Come Out?

- A correct TVR order is produced by Conflict-Clause Minimization.
- Program ran from 3% slower to 14% faster.
- See results of the [MiniSat Hack Track](#) in this year's SAT competition.
- *Caution*: Literals in *removableStack* that are not in the conflict clause and are not reachable by a path of removable literals should be pruned (*u* in the example).

Conclusion

Study Recursive Depth-First Search.