Title: Intelligent Backtracking Algorithms:
A Theoretical Evaluation

Material: Kondrak & Van Beek, IJCAI'95

Foundations of Constraint Satisfaction
CSCE 990-005, Fall 2000

Berthe Y. Choueiry (Shu-we-ri)
Ferguson Hall, Room 104
choueiry@cse.unl.edu, Tel: (402)472-5444

Context

- Usually, backtracking algorithms evaluated by *empirical* testing
  Shortcoming: average case, representativeness of test problems
- Performance of backtracking depends heavily on problem instance
- Problem1: CSP is NP-complete: it is often possible to construct examples where is a *B* is worst than C-BT
- Problem2: comparison criteria? run time? cost constraint checks?
- This paper offers a *theoretical* approach
  $\rightarrow$ number of tree nodes visited
  1. number of constraint checks (does not account for cost of special data structures)
  2. number of nodes generated

Assumptions

- Constraints are binary
- Instantiation order fixed and static
- Seeking *all* solutions

Contributions

- Advantages:
  - Prove correctness of backtracking algorithms: BJ, CBJ
  - Determine a partial order (PO) between algorithms in terms of 2 performance criteria
    1. number of nodes visited
    2. number of consistency checks performed
  - PO explains/justifies experimental results
- Results:
  - Proves BJ & CBJ are correct (*i.e.*, soundness & completeness)
  - Proves FC never visits more nodes than BJ
  - Improves BMJ & BM-CBJ to perform less consistency checks
  - Provides framework for characterizing (future) backtracking algo.

Definitions

- Backtracking extends partial solutions
- A partial solution is consistent with a set of uninstantiated variables if it can consistently extended to these variables (if there are assignments to these variables such that the 'new' partial solution is consistent)
- Dead-end: when all values of current variable are rejected
- Lower levels: closer to the root (shallower)
- Higher levels: closer to the fringe (deeper)
- 2 backtracking algorithms are equivalent if on *every* CSP they generate the same tree and perform the same consistency checks

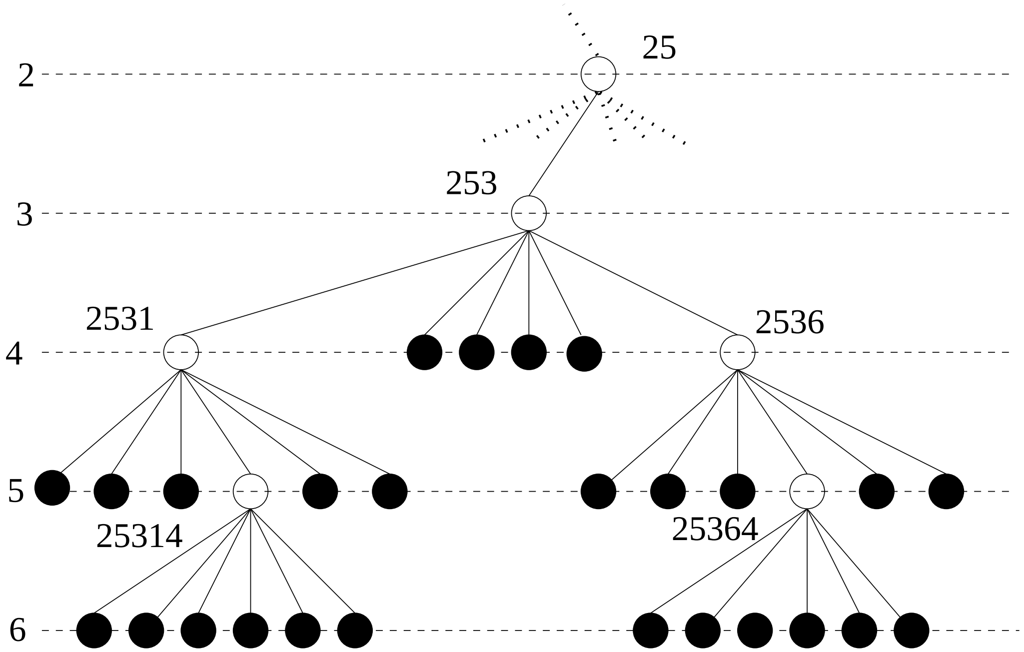
6-Queens

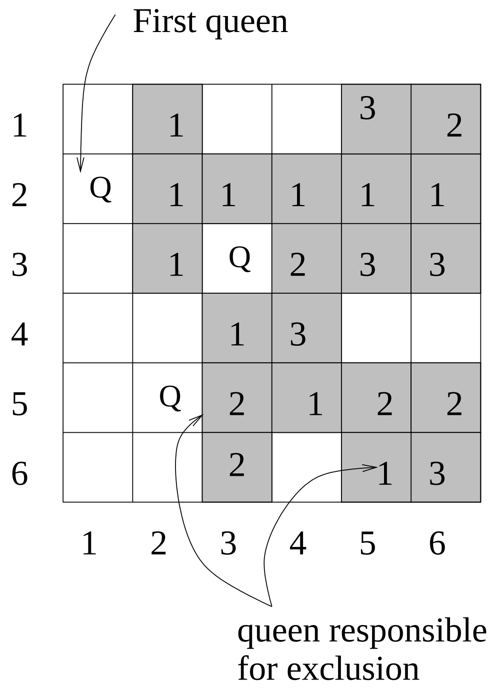
Representation

- Variables: board columns
- Domain values: board rows

1						
2	Q					
3						
4						
5		Q				
6						
	1	2	3	4	5	6

Chronological backtracking

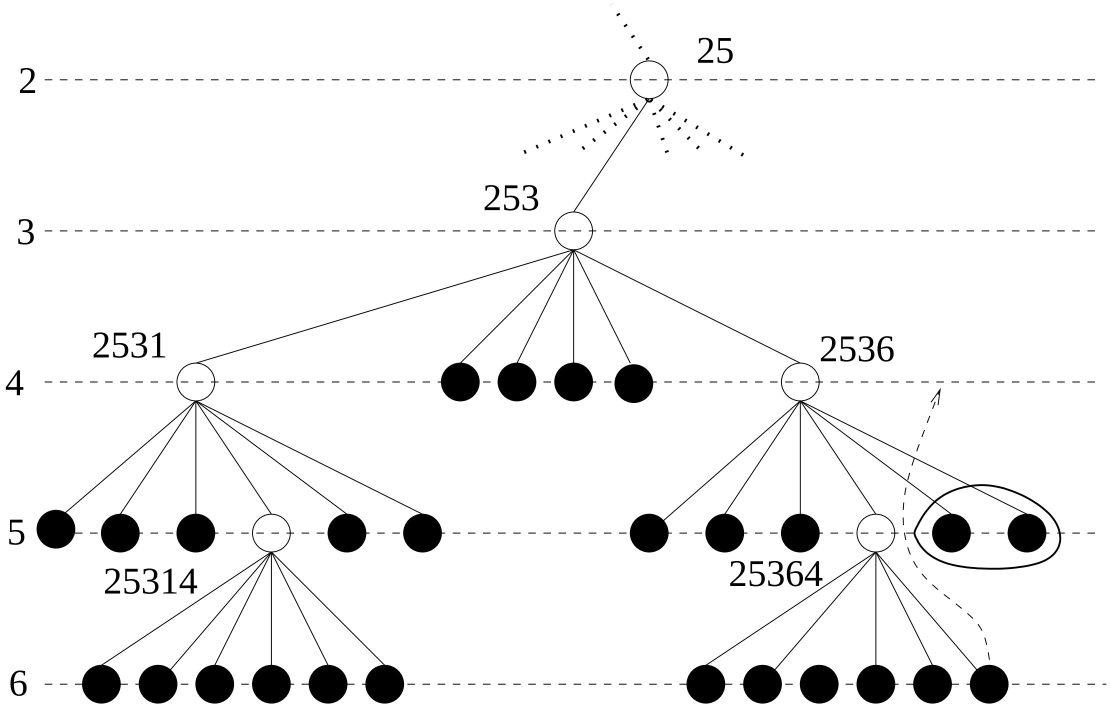




Backjumping

Reaches dead-end at Q6, when expanding 25364

BJumps to Q4: 25365 and 25366 are safely skipped



	1	2	3	4	5	6
1		1			3	2
2	Q	1	1	1	1	1
3		1	Q	2	3	3
4			1	3		
5		Q	2	1	2	2
6			2		1	3

Conflict-directed backjumping

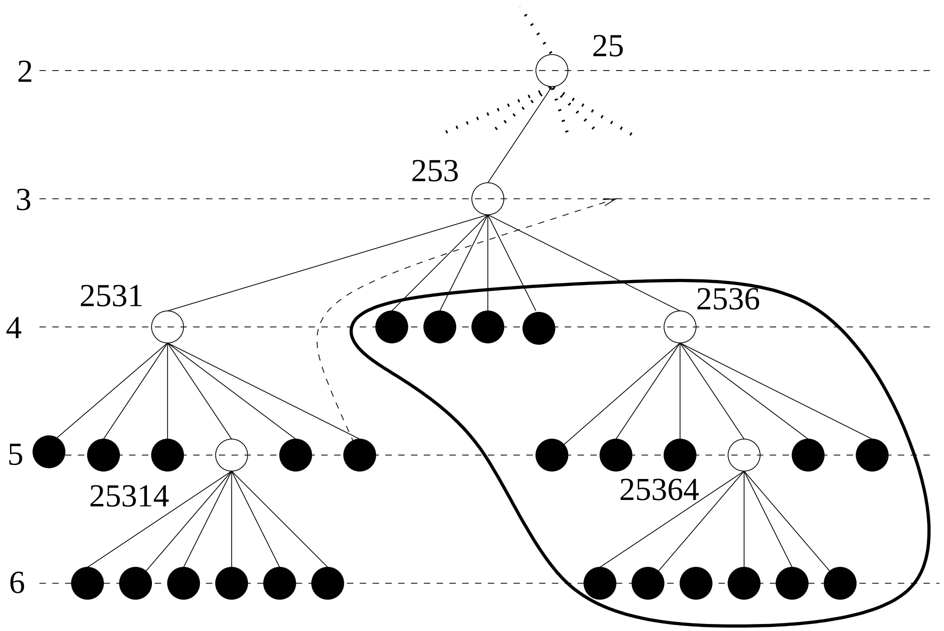
Reaches dead-end when expanding 25314

Conflict-set of Q6 is $\{1, 2, 3\}$

Tries Q5 = 5, 6; then BJumps to Q3

253 is inconsistent with {Q5, Q6}

although 253 is consistent with Q5 and with Q6 (separately)



	1	2	3	4	5	6
1		1			3	2
2	Q	1	1	1	1	1
3		1	Q	2	3	3
4			1	3		
5		Q	2	1	2	2
6			2		1	3

Forward checking

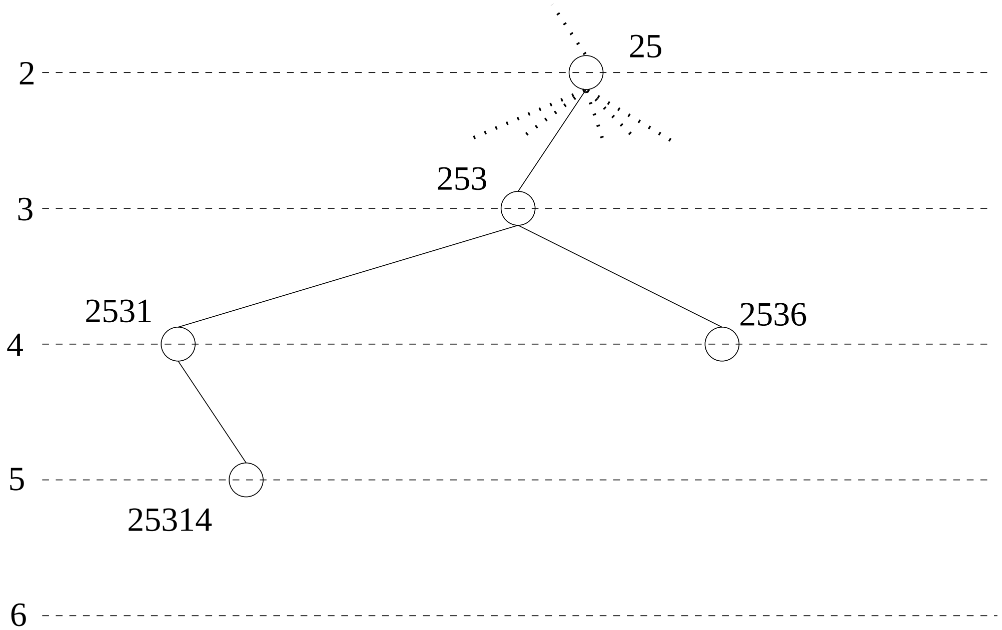
Visits only consistent nodes, but not 25364

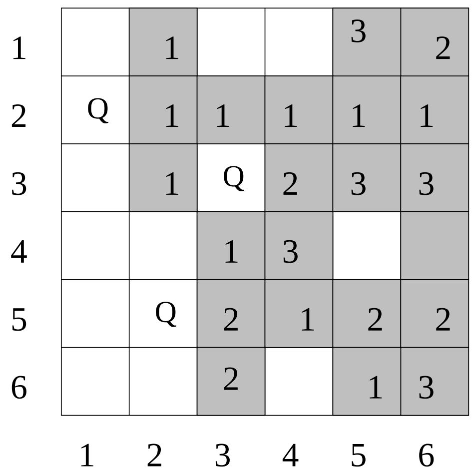
FC detects dead-end because Q4=6 and Q6 are inconsistent

FC detects inconsistency between current partial solution and a future variable *before* reaching it

FC cannot detect inconsistency between a *set* of variables

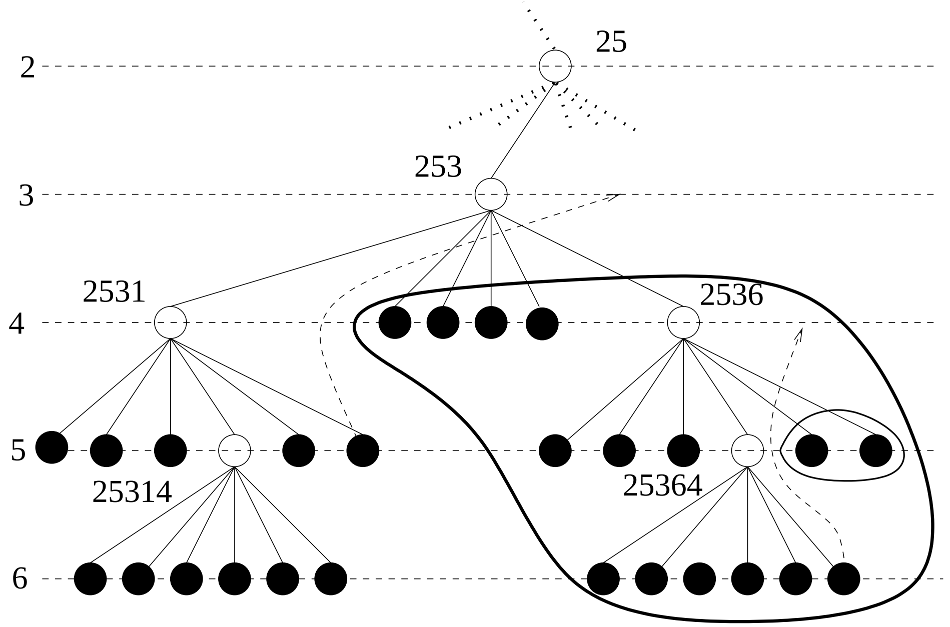
2536 is visited by FC, but skipped by CBJ





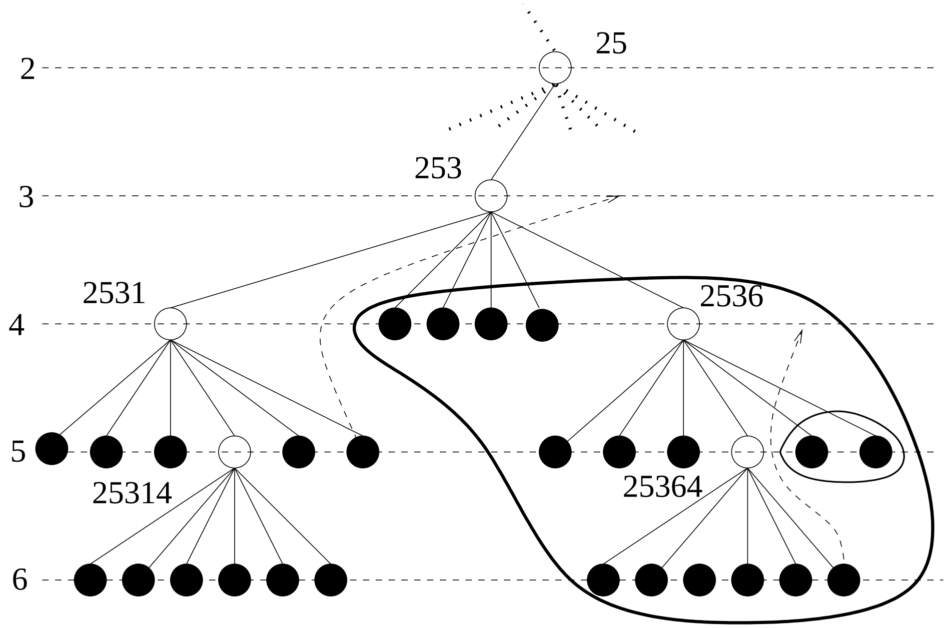
Theorem 1: sufficient conditions

- BT visits a node if its parent is consistent
- BJ visits a node if its parent is consistent with all variables
- CBJ visits a node if its parent is consistent with all sets of variables
- FC visits a node if it is consistent and its parent is consistent with all variables

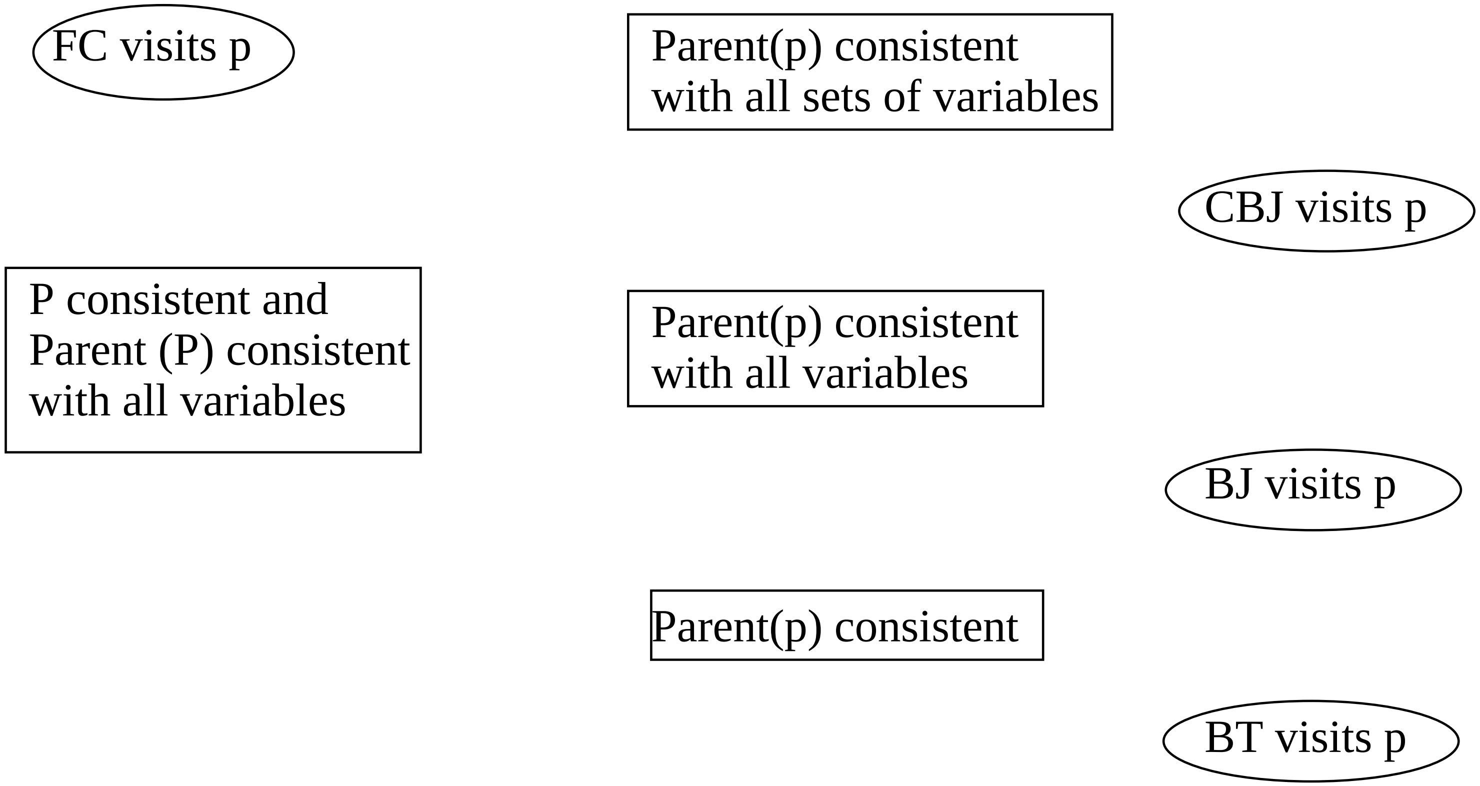


Theorem 2: necessary conditions

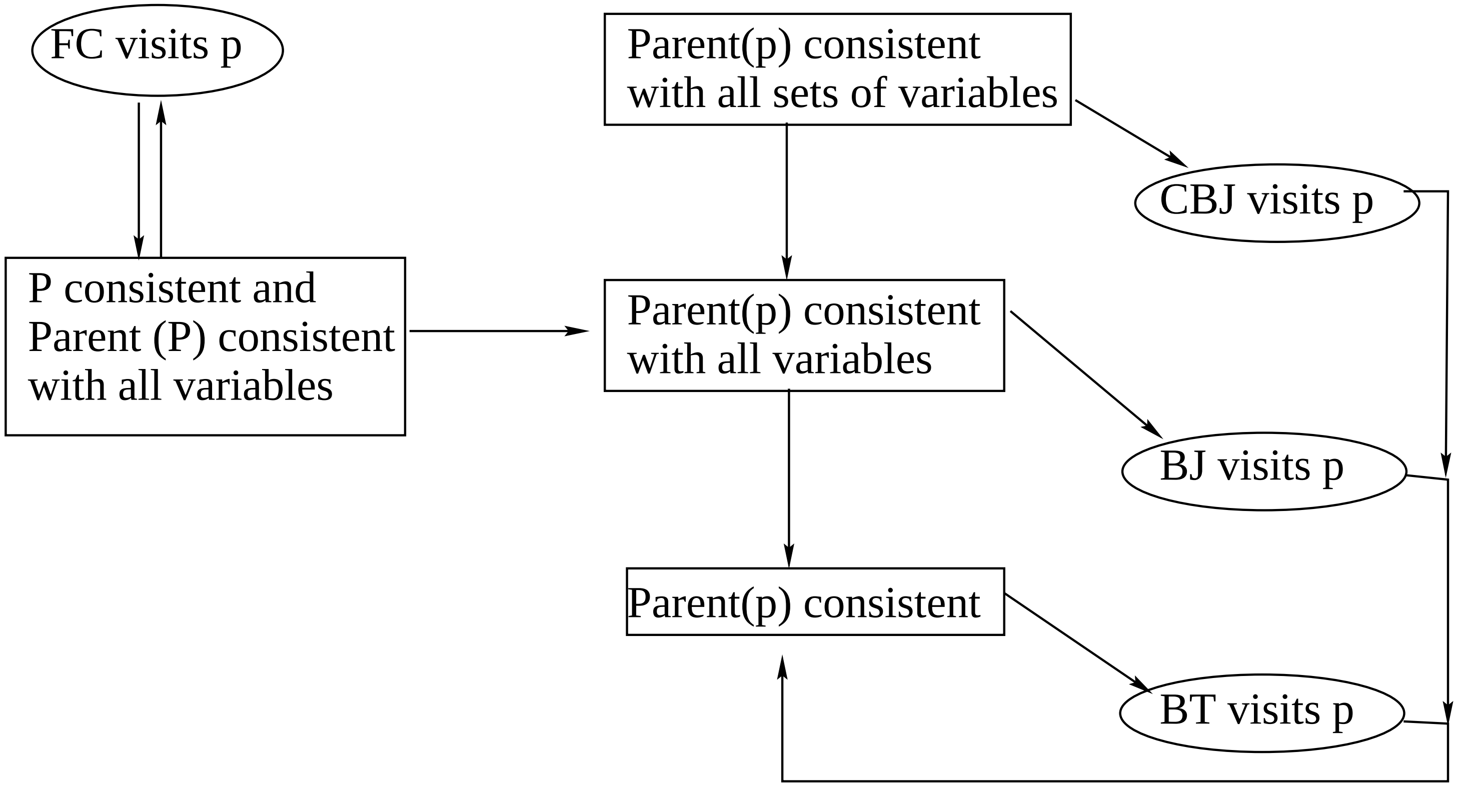
- BT visits a node *only* if its parent is consistent
- BJ visits a node *only* if its parent is consistent
- CBJ visits a node *only* if its parent is consistent
- FC visits a node *only* if it is consistent and its parent is consistent with all variables



Conditions graph

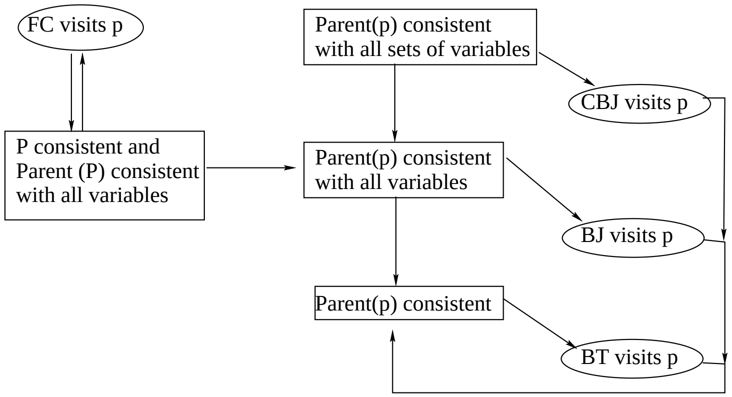


Conditions graph



Corollary 1:

- BT visits all nodes BJ visits
- BT visits all nodes CBJ visits
- BT visits all nodes FC visits
- BJ visits all nodes FC visits *New!*



Theorem 3: BJ visits all nodes CBJ visits

Note: Trace a partial/total order

FC vs. CBJ? *See 6-queens for counter-example*

Correctness:

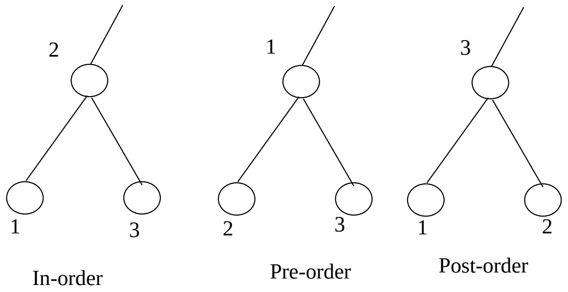
termination, soundness and completeness

Corollary 2:

- BT is correct
- BJ is correct
- CBJ is correct
- FC is correct

Extensions

- Approach can be extended to other algorithms
- Initial assumptions: seeking all solutions
- Theorems remain valid (for any number of solutions) if pre-order traversal is followed
  (Restriction to nodes that precede the last node visited)



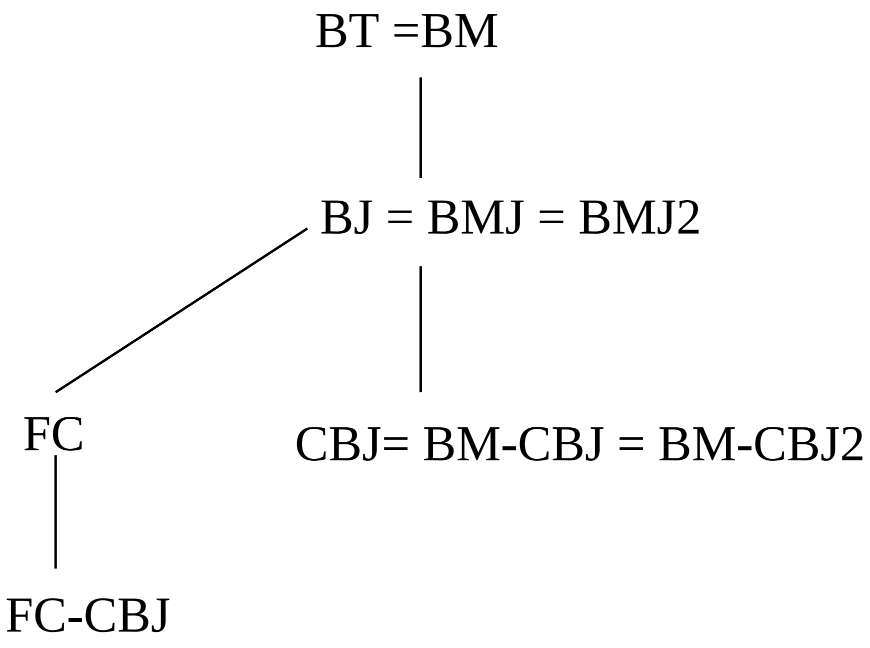
Theorems hold for "1 solution," proofs slightly different

BM + BJ

- BM uses mcl (n$\times$ m) and mbl (n)
- Prosser noted anomaly
- Kondrak & van Beek change mbl into 2-dim array (n$\times$ m)
- .. and propose BMJ2 and BM-CBJ2, which fix the anomaly

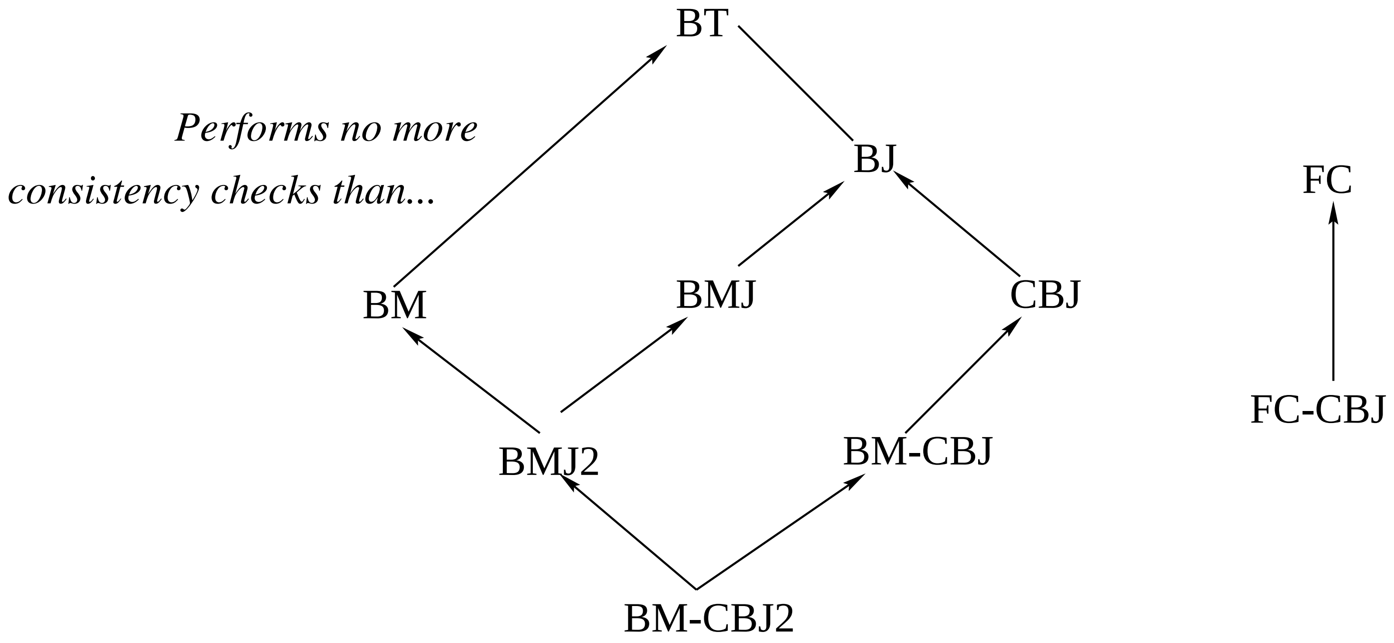
Hierarchy 1: number of nodes visited

- Backmarking does *not* affect the number of nodes visited
- All not shown relations can be disproved by counter-examples
  Surprise: FC-CBJ may visit more nodes than CBJ



Hierarchy 2: number of consistency checks performed

- BT, BJ, and CBJ perform the same amount of consistency checks at any given node ⟶ same order as in hierarchy 1
- BM reduces consistency checks
- All not shown relations can be disproved by counter-examples
  Surprise: FC-CBJ may perform more checks than BT!



Conclusions

- General theorems that (fully/partially) describe behavior of backtracking algorithms
- Theorems used to prove correctness of algorithms
- Theorems used to build hierarchy 1 and 2 (number of nodes visited, number of checks performed)
- Anomaly of BM (vs. BJ and CBJ) fixed
- Future: carry out same analysis for
  - Graph-based backjumping (Dechter, 1990)
  - Full look-ahead (Nadel, 1989)