Proof number search and QBF solving

Yifan He

Supervisor: Dr. Abdallah Saffidine

Motivation

- QBF is similar to 2-player strategy game in a theoretical level
- All search based QBF solvers use depth-first search
- DFS or IDS is not ideal for game solving
  - Shogi dfs cannot solve a game with more than 17 steps
  - Best first search can solve games with 100+ steps.
  - Huge searching depth
- QBF has a very deep searching space

Outline

- Background
- Expectation
- Methodology
- Result and Discussion
- Conclusion and Future work

Outline

- Background
- Expectation
- Methodology
- Implementation and Result
- Conclusion and Future work

QBF

- QBF expression: quantifier prefix + propositional formula (CNF)

  f: $Q_1 X_1 Q_2 X_2 \ldots Q_n X_n \Phi$ $\quad$ $(Q_i \in \{\exists, \forall\})$
- Semantic of QBF
  - If Φ contains a contradictory clause, false
  - If Φ has all clauses satisfied, true
  - If Q_1 is existential, f is true iff either $Q_2 X_2 \ldots Q_n X_n \Phi(X_1)$ **or** $Q_2 X_2 \ldots Q_n X_n \Phi(\neg X_1)$ is true.
  - If Q_1 is universal, f is true iff both $Q_2 X_2 \ldots Q_n X_n \ \Phi(X_1)$ **and** $Q_2 X_2 \ldots Q_n X_n \ \Phi(\neg X_1)$ is true.

- Significance of QBF: model checking, planning, games…
- QBF solvers: expansion-based (e.g. caqe), search-based (e.g. DepQBF)
- Notation: From now on, we would use x_i to represent existential variables, and y_i to represent universal variables.

Assertion clauses/cubes

- QBF under partial assignment μ
- μ-contradicted clause: a clause that has all existential literals falsified and no universal literal satisfied
  - μ=$[x_1, y_1, -x_2]$, C=$(-x_1 \vee -y_1 \vee x_2 \vee y_2)$
- μ-satisfied cube: a cube that has all universal literals satisfied and no existential literal falsified
  - μ=$[y_1, x_1, -y_2]$, T=$(y_1 \wedge x_1 \wedge -y_2 \wedge x_2)$
- μ-false clause: a clause that has all literals falsified
  - μ=$[x_1, y_1, -x_2, -y_2]$, C=$(-x_1 \vee -y_1 \vee x_2 \vee y_2)$
- μ-truth cube: a cube that has all literals satisfied
  - μ=$[y_1, x_1, -y_2, x_2]$, T=$(y_1 \wedge x_1 \wedge -y_2 \wedge x_2)$

Adapt from:
https://www.researchgate.net/publication/220543407_ClauseTerm_Resolution_and_Learning_in_the_Evaluation_of_Quantified_Boolean_Formulas

Search based QBF solving

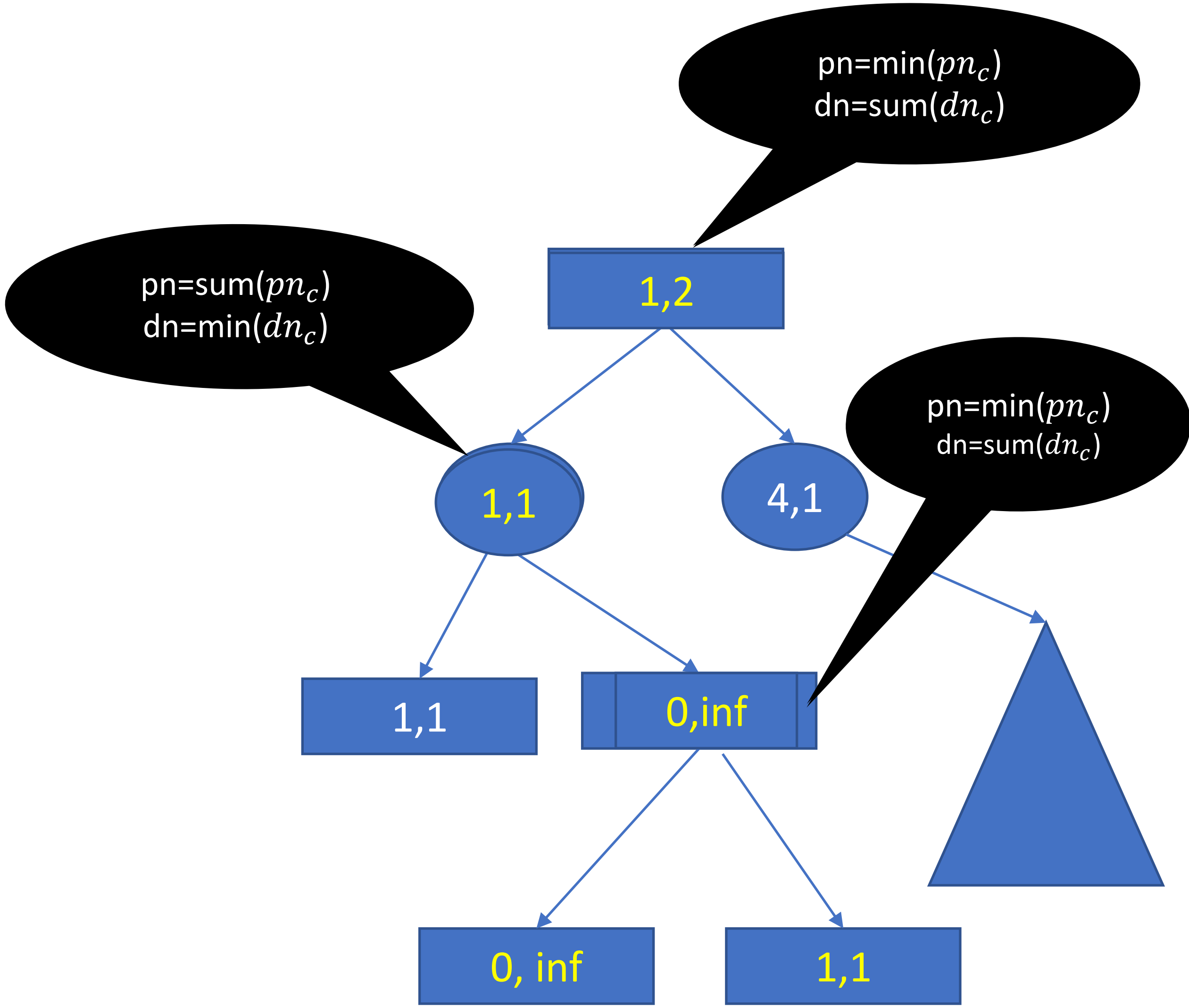
- 1998: QDPLL
  - QBF definition
  - Unit propagation
  - Pure literal elimination
- 2001: Backjumping (CBJ + SBJ)
  - Reason for conflict: associate each false state with a **μ-contradicted** clause
  - Reason for solution: associate each true state with a **μ-truth** cube
  - Q-resolution to calculate reasons for internal nodes
  - Pruning when reason for $\Phi_{\mu;l}$ does not contain $|l|$
- 2002: Conflict solution driven learning (QCDCL/CSDCL=CDCL + SDCL)
  - Backjumping improvement, reason not only affect the current path of the search tree
  - Associate each false state with a **μ-contradicted** clause, each true state with a **μ-satisfied** cube
  - Additional clauses and cubes are added conjunctively/disjunctively to the original formula
  - More unit propagation
  - Clauses and cubes are generated by Q-resolution
- **For both Backjumping and QCDCL, the QBF is false iff empty clause is derived, the QBF is true iff empty cube is derived**

QBF game representation

- QBF can be viewed as an and-or two player strategy game
  - existential quantifier is the or player (maximizer)
  - universal quantifier is the and player (minimizer)
  - existential quantifier tries to satisfy the formula
  - universal quantifier tries to falsify the formula

Proof number search

- Each node stores 2 information:
  - Proof number (pn)
  - Disproof number (dn)
- Iteration:
  - Selection (find MPN)
  - Expansion
  - Initialization
  - Backpropagation
- Advantage
  - Not get stuck at one side of the searching space
- Drawback
  - Memory issue, seesaw effect
- Variations
  - DeepPNS, df-pn, PN2, PN* etc.



Outline

- Background
- Expectation
- Methodology
- Implementation and Result
- Conclusion and Future work

Expectation

- Design a PNS-based QBF solver
- Combine existing search based QBF solving techniques with PNS
  - Backjumping
  - QCDCL

Expectation (cont.)

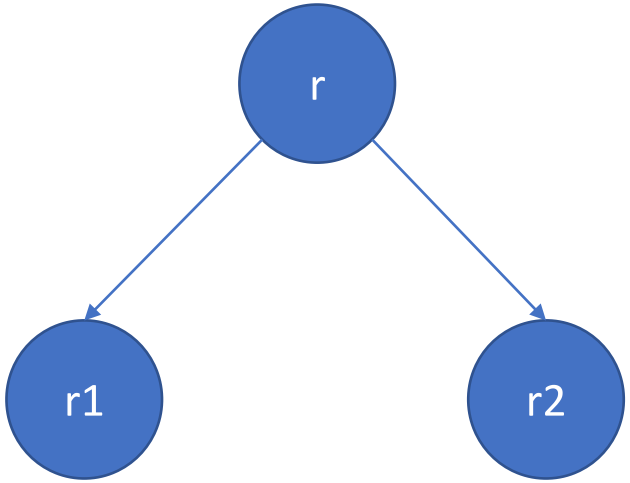
- Can we design and implement a PNS-based Backjumping solver?
- Can we design and implement a PNS-based QCDCL solver?
- Can PNS bring any performance benefits?

Outline

- Background
- Expectation
- Methodology
- Implementation and Result
- Conclusion and Future work

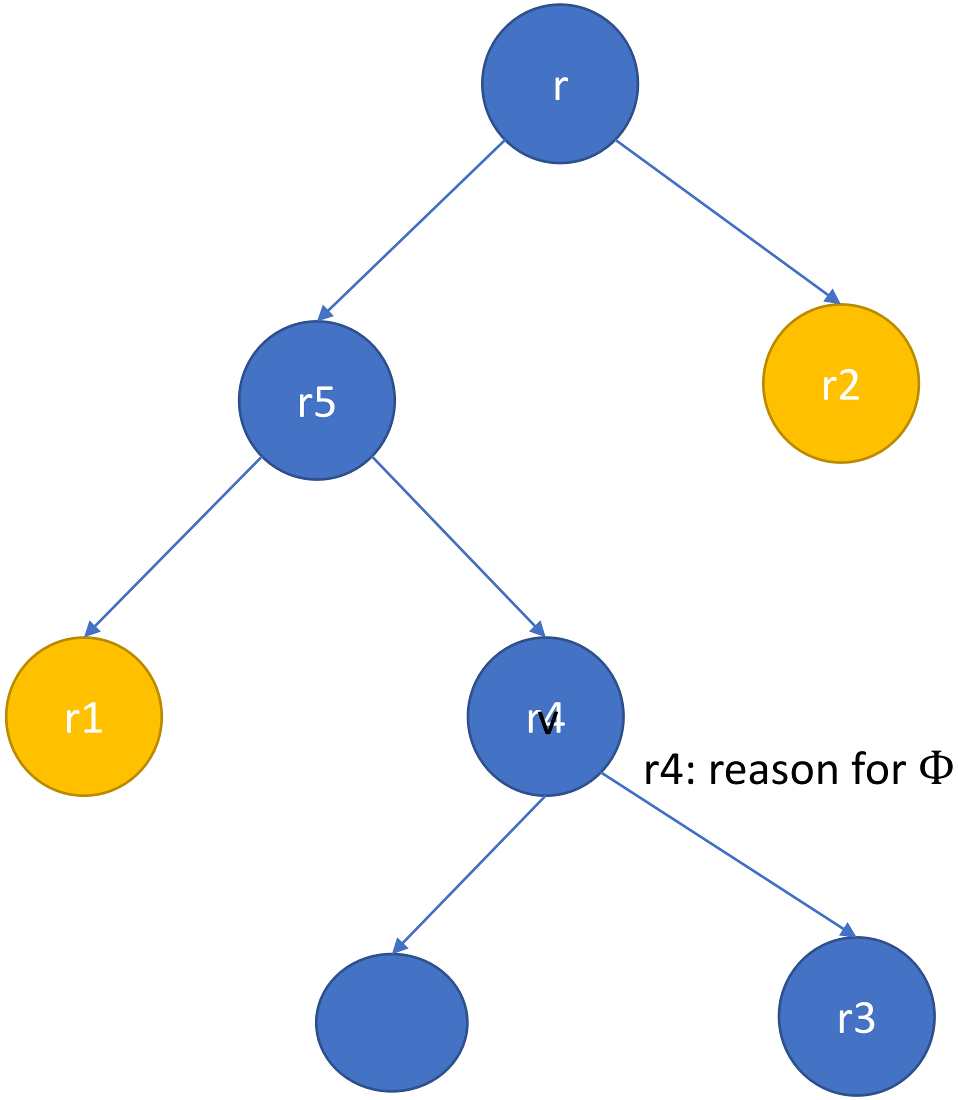
Proof number search and Backjumping

- Information in each node
  - Proof number (pn)
  - Disproof number (dn)
  - Branching variable (v)
  - Reason for (un)satisfiability (reason, default null)
  - Reason for UNSAT is a μ-contradicted clause
  - Reason for SAT is a μ-truth cube



During backpropagation, calculate reason r for the parent based on r1 and r2.

Proof number search and Backjumping

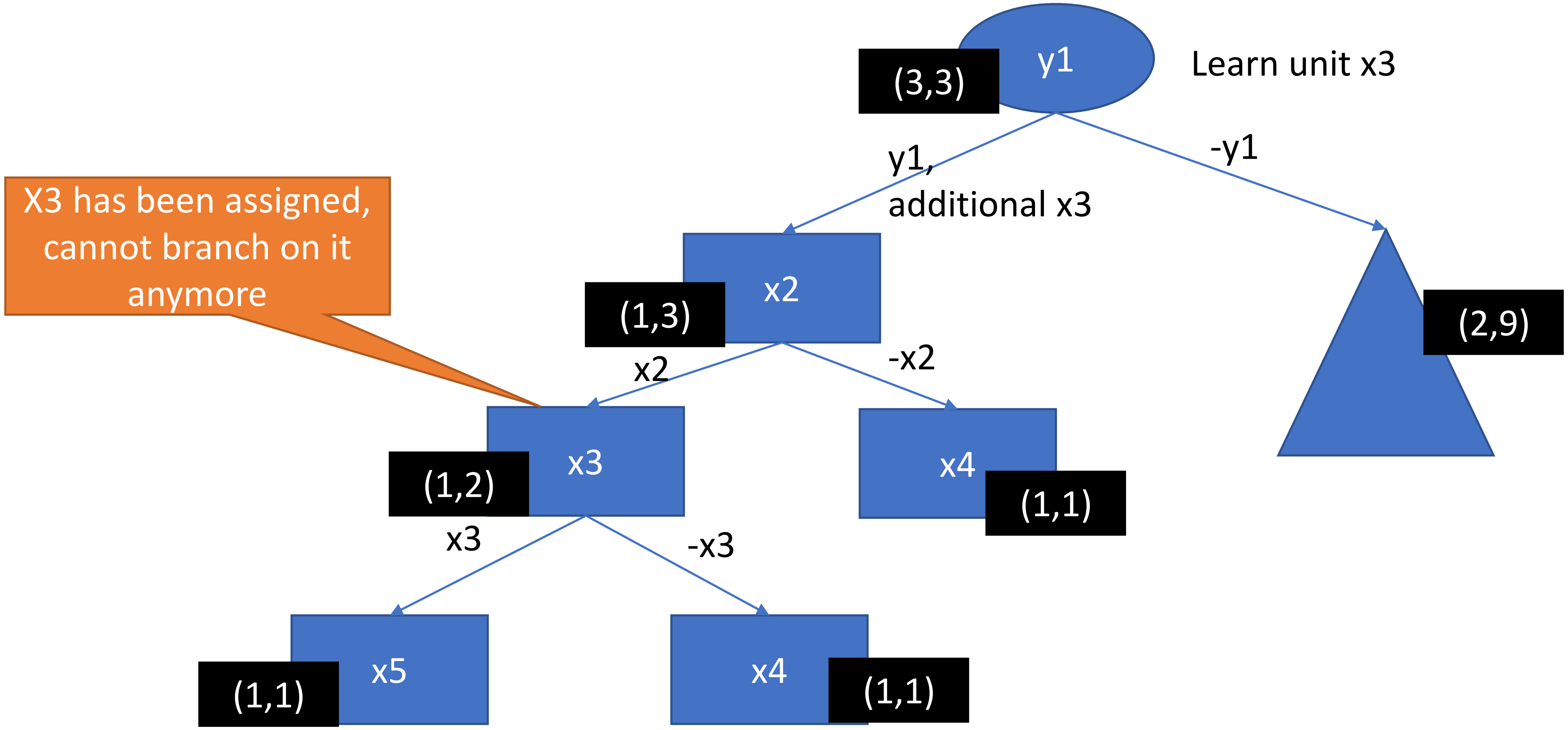


r4: reason for $\Phi_{\mu;-v}$ does not contain v or -v

Proof number search and QCDCL

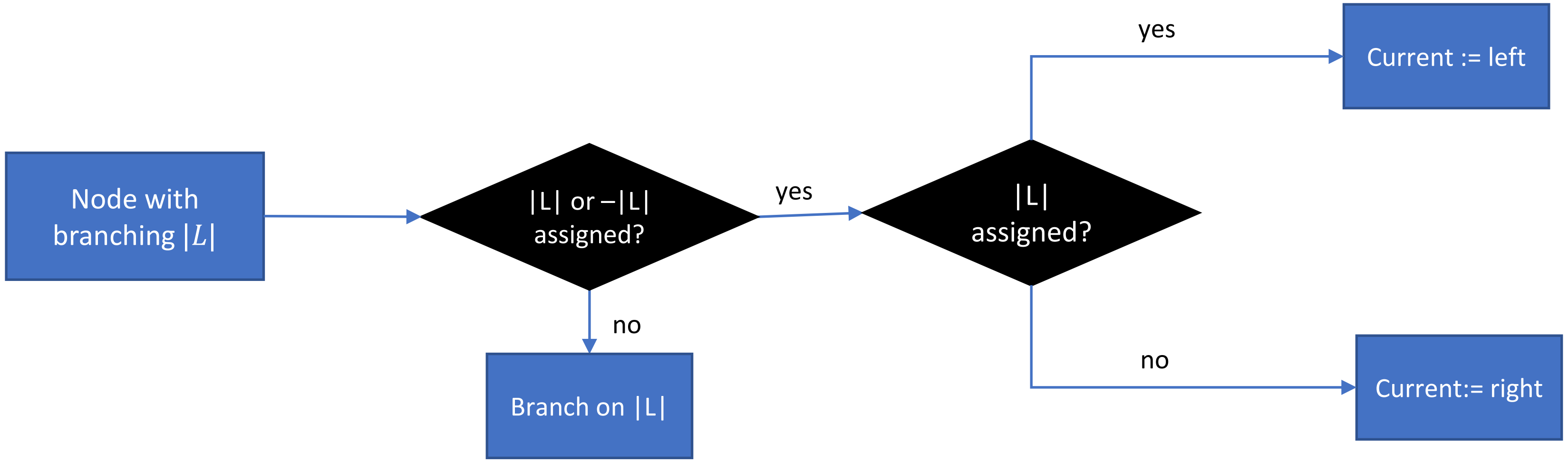
- Each node stores exactly the same information as backjumping
- Potential Benefit
  - Same as Backjumping vs QCDCL
  - Learned clause/cube can simplify the search on other parts of the tree
- Main obstacle
  - Learned constraints create more unit propagation, make the branching variables be assigned earlier

Proof number search and QCDCL (cont.)



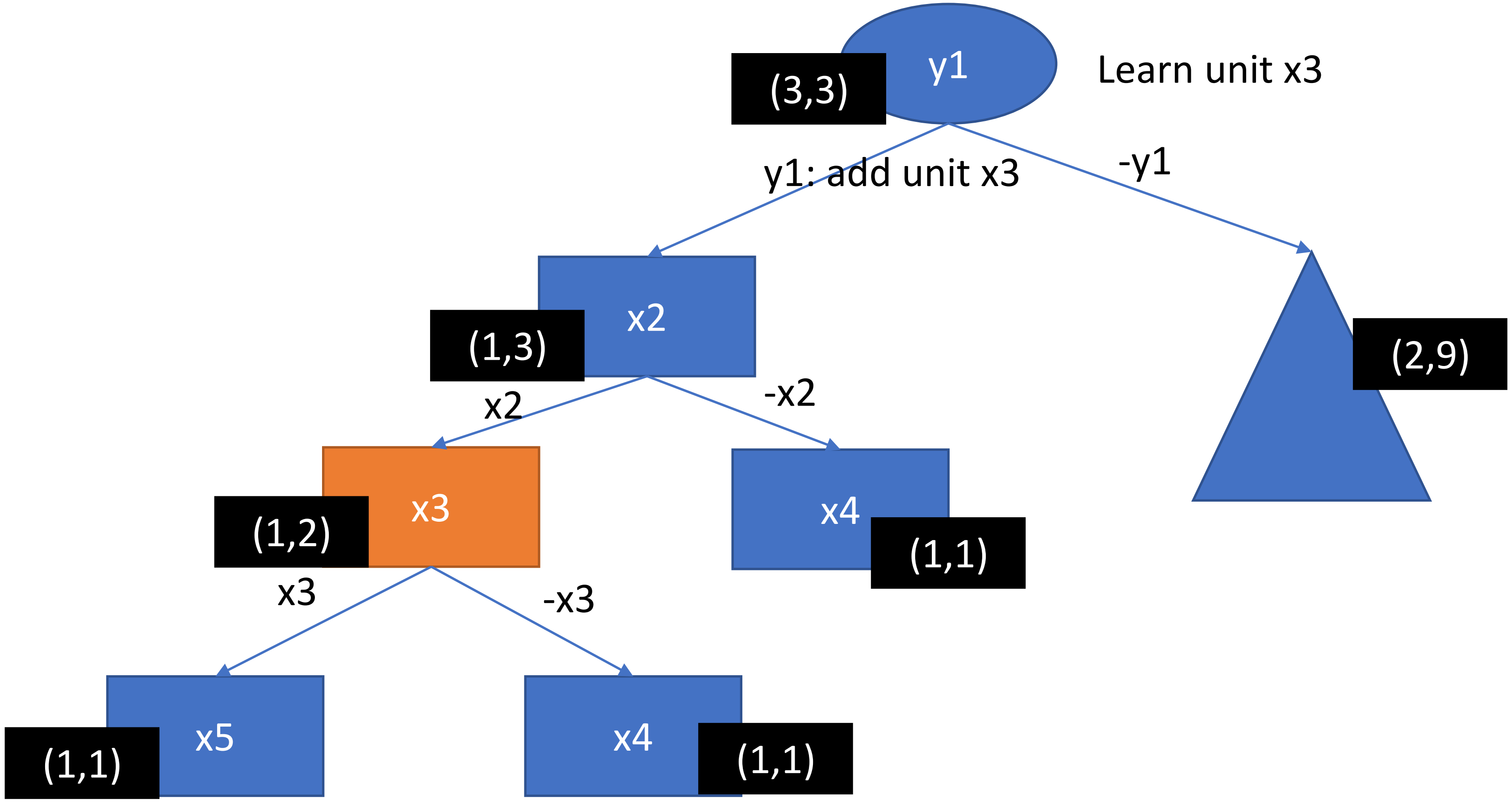
Proof number search and QCDCL (cont.)

- MCTS + CDCL based SAT solver, Schloeter (2017)
- Lazy fix the search tree

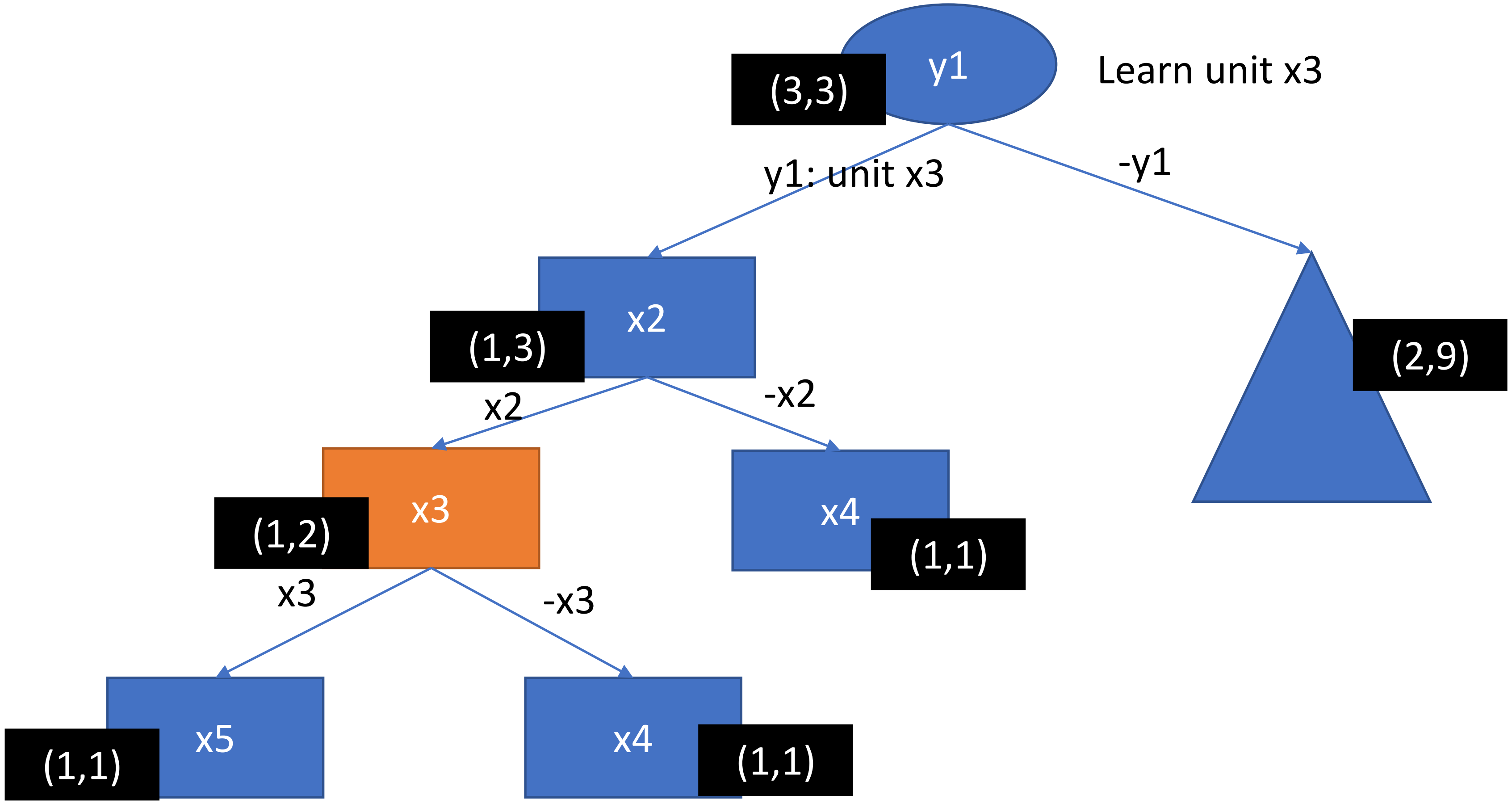


- All unit propagation must be done immediately
- Pure literal elimination is shut down

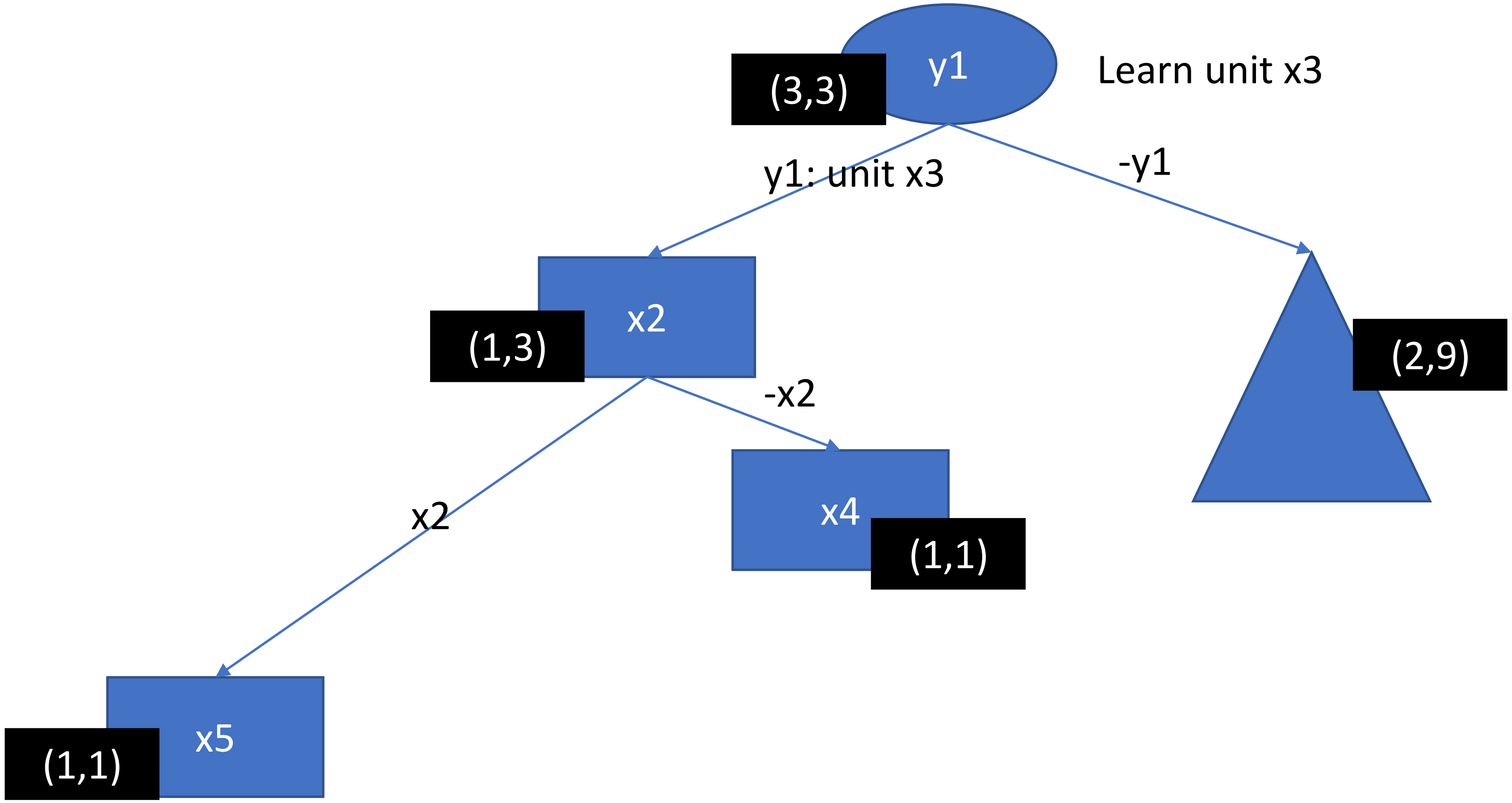
QCDCL and PNS example



QCDCL and PNS example



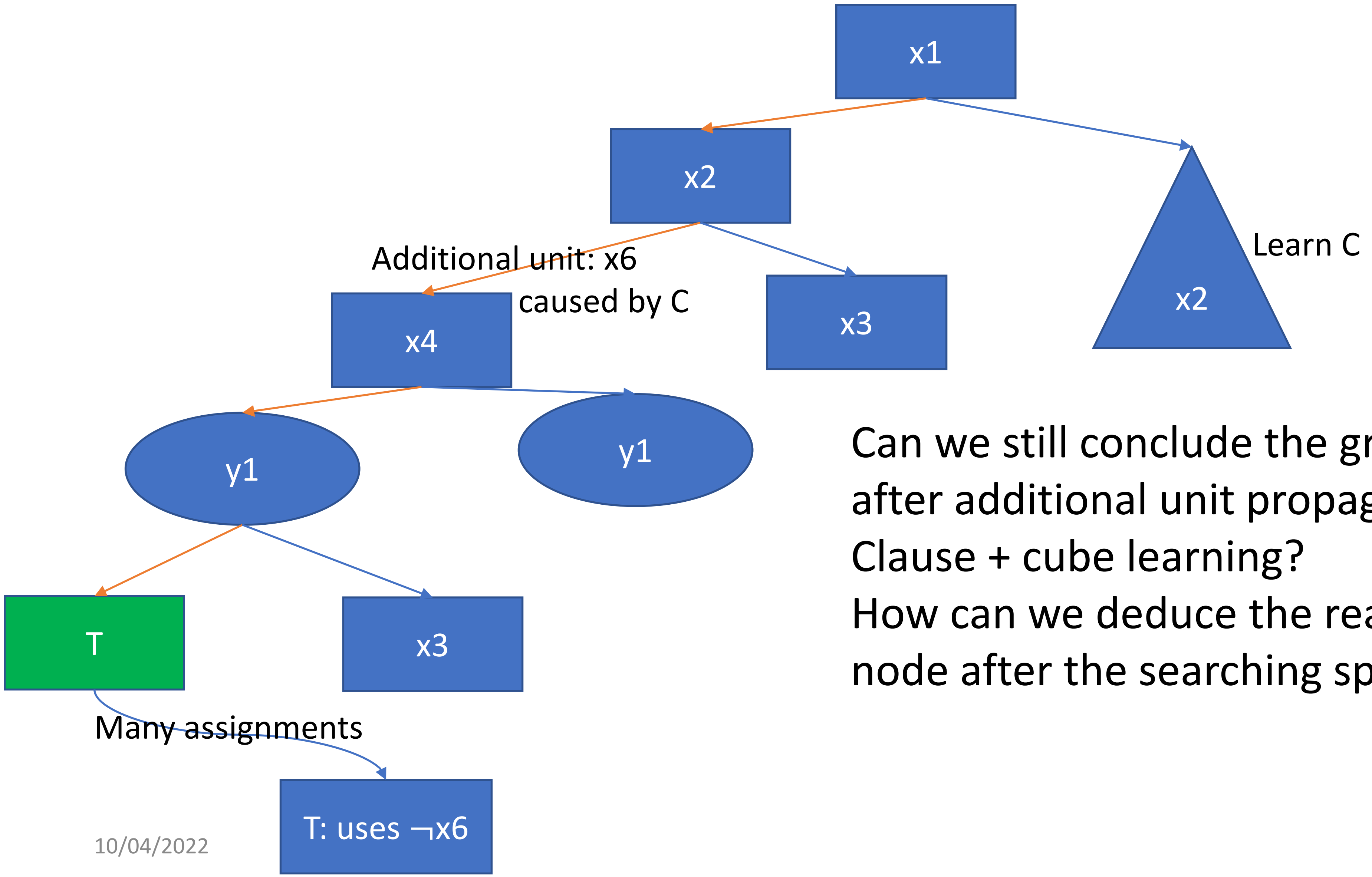
QCDCL and PNS example



Full story?

- We “can” design a PNS-based QBF solver with Schloeter’s method that features:
  - Unit propagation
  - Conflict driven clause learning
  - Solution driven cube learning
- Not even close!

Proof number search and QCDCL (cont.)



Can we still conclude the green node is SAT after additional unit propagation caused by Clause + cube learning?
How can we deduce the reason for the green node after the searching space has changed?

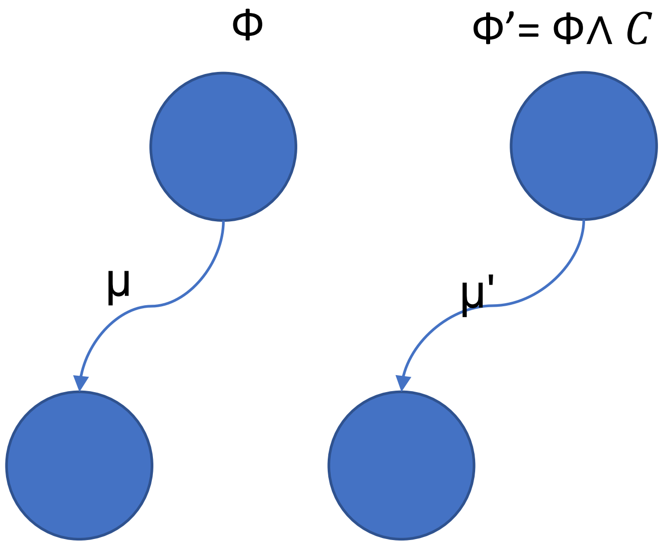
Important fact

- UP > PLE
- CDCL + SDCL (with special care) > CDCL + SBJ > Backjumping
- Most search based solvers has CDCL, some does not have SDCL (e.g. Quaffle-CDL)

Simplified problem

- Can we design a PNS-based QBF solver with Schloeter's method that features:
  - Unit propagation (UP)
  - Conflict driven clause learning (CDCL)
  - Solution driven Backjumping (SBJ)

Simplified problem (cont.)



- PNS + Schloeter + UP + CDCL + SBJ is sound
- Assume earlier assignment is μ and current assignment is μ’
- $\Phi'_{\mu'}$ is Unknown, we want to show $\Phi_{\mu} = \Phi'_{\mu'}$
- Sketch proof:
  - If a node is derived to be UNSAT, there’s a μ-contradicted clause associated with it
  - If a node is derived to be SAT, and there’s no SDCL, there’s a μ-truth cube associated with it
  - μ ⊆ μ’ and the set universal literals in μ is equal to the set universal literals in μ’
  - A μ-contradicted clause is a μ’-contradicted clause!
  - A μ-truth cube is a μ’-truth cube!
    - μ=$[x_1, y_1, -x_2]$, C=$(-x_1 \vee -y_1 \vee x_2 \vee y_2)$
    - μ=$[y_1, x_1, -y_2, x_2]$, T=$(y_1 \wedge x_1 \wedge -y_2 \wedge x_2)$
  - The assertion clause/cube is going to maintain the truth value of the node

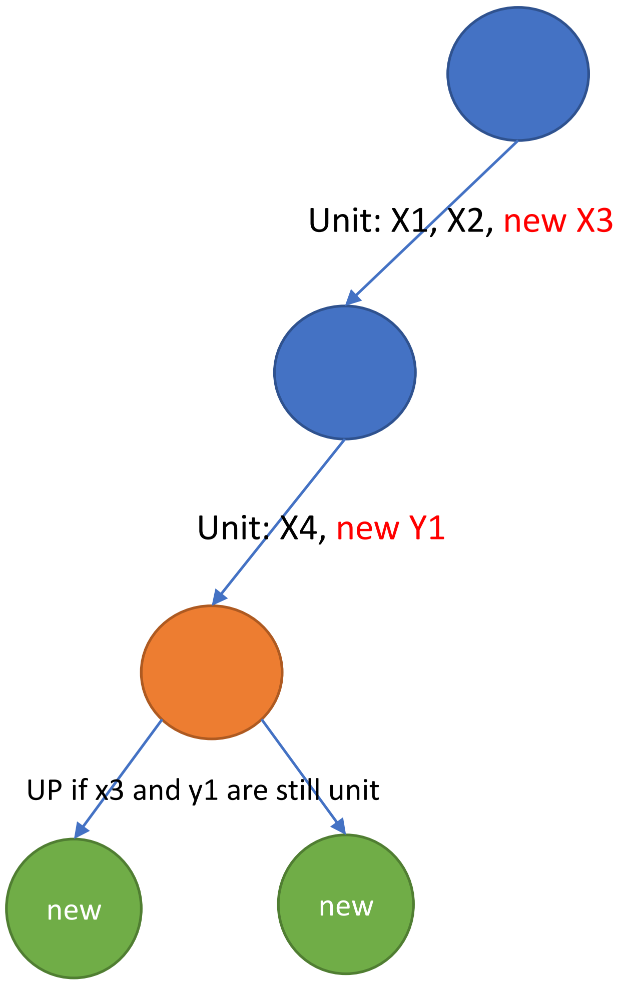
Can we go further?

- Can we design a PNS-based QBF solver with Scholter's method that features:
  - Unit propagation
  - Conflict driven clause learning
  - Solution driven cube learning

- SDCL could falsify μ ⊆ μ', because of universal unit propagation
  - Unit propagation order is unimportant without cubes, but is important with cubes
  - Example: $x \vee y$, $-y$ universal unit
- Universal (resp. existential) unit propagation could satisfy (resp. falsify) the μ-contradicted clause (resp. μ-satisfied cube) associated with a node
  - μ=[$x_1, y_1, -x_2$], C=($-x_1 \vee -y_1 \vee x_2 \vee y_2$)

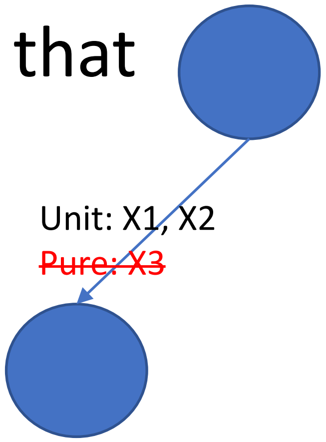
Can we go further?

- Can we design a PNS-based QBF solver with Schloeter's method that features:
  - Unit propagation
  - Conflict driven clause learning
  - Solution driven cube learning
- Potential solution:
  - Memorized the order of unit propagation, postpone additional unit propagation until creating new nodes
  - Postpone unit propagation is always undesired



Can we go further?

- Can we design a PNS-based QBF solver with Schloeter’s method that features:
  - Unit propagation
  - Pure literal elimination
  - Conflict driven clause learning
  - Solution driven backjumping



- Pure literal elimination can be blocked! $\mu \subseteq \mu'$ no longer holds!

Outline

- Background
- Expectation
- Methodology
- Implementation and Result
- Conclusion and Future work

Implementation details

- 2-watched literal data structure
  - Lazy data structure for state of art SAT solvers and many search based QBF solver
- Learning strategy
  - 1-UIP
  - Never delete learned constraints

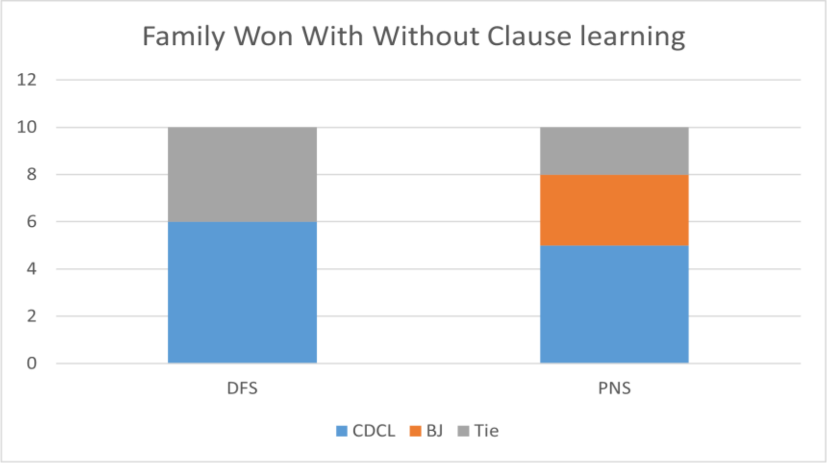
Result and discussion

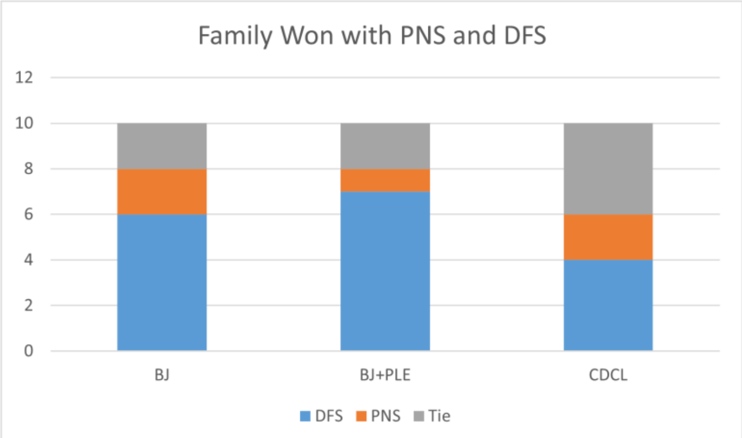
- The PNS based solver produced the correct result on all solved benchmarks
- Clause learning helps the PNS solver
- PNS solver is beneficial to some families of instances, although no benefits on most
- PNS cannot solve instances that DFS solvers cannot solve when advanced reasoning technique is required (e.g. Adder)
- PLE is very useful to PNS as well (gttt4x4)

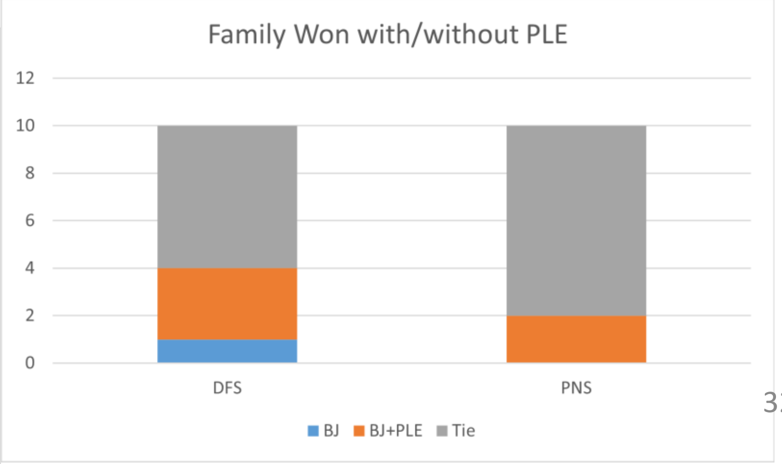
Family	Total	BJ	BJ + PLE	CDCL+SBJ	PBJ	PBJ + PLE	PCL+SBJ
Block (2004)	8	2	3	3	0	1	1
Chain (2004)	8	8	8	8	7	7	6
Counter (2004)	8	3	3	4	2	2	2
K_dum_p (2004)	8	3	3	7	1	1	7
K_lin_p (2004)	8	2	3	8	1	1	3
Logn	2	2	1	2	0	0	2
Toilet	8	6	6	6	7	7	7
Tree (2004)	8	6	6	8	6	6	8
	58	32	33	46	24	25	36
Real world instances							
Adder	32	4	4	4	4	4	4
Gttt4x4	95	13	40	15	18	38	17

Number of solved instances by different methods

- Time limit per instance: 900s
- PLE pure literal elimination, BJ: DFS based Backjumping solver; CDCL+SBJ: DFS based CDCL solver with SBJ
- PBJ: PNS based Backjumping solver,PCL: PNS based CDCL solver; PCL+SBJ: PNS based CDCL solver with SBJ





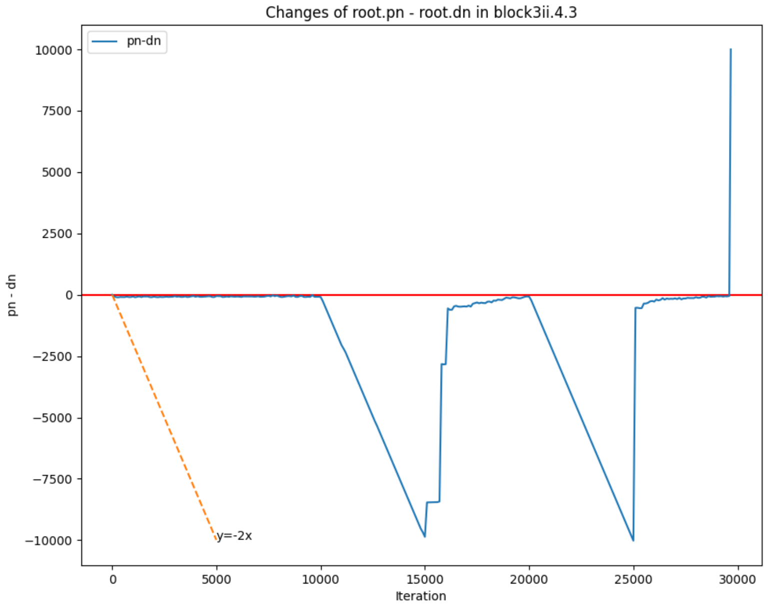


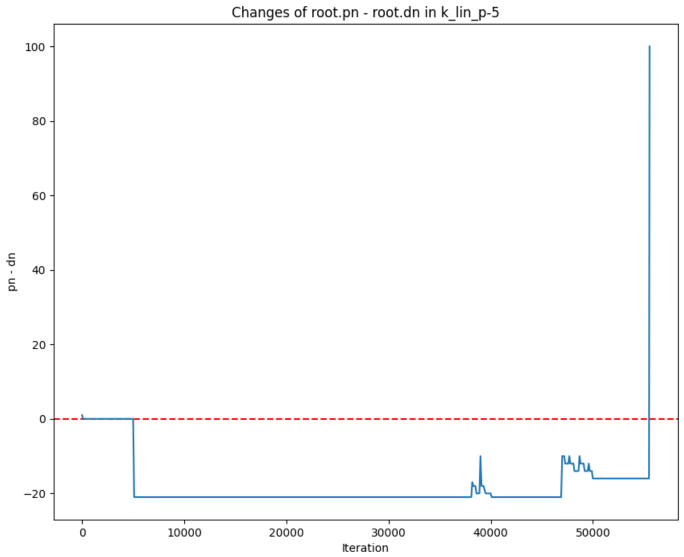
Result and discussion (cont.)

Family	#Both CDCL-SBJ and PCL-SBJ	Average visit ratio in PCL-SBJ	#Solved PCL-SBJ not CDCL-SBJ
Block (2004)	1	69	0
Chain (2004)	6	37	0
Counter (2004)	2	12	0
K_dum_p (2004)	7	11	1
K_lin_p (2004)	3	14	0
Logn	2	98	0
Toilet	5	39	2
Tree (2004)	8	21	0

Result and discussion (cont.)

- Proof and disproof number is not very informative in the current setting





PNS is unhelpful: Both instances are UNSAT, pn-dn becomes positive only before they are solved

Outline

- Background
- Expectation
- Methodology
- Implementation and Result
- Conclusion and Future work

Outcome

- Can we design and implement a PNS-based Backjumping solver?
  - Definitely
- Can we design and implement a PNS-based QCDCL solver?
  - Yes, if we ignore cube learning and pure literal elimination
  - Doable with full QCDCL if postpone new unit propagation
  - There are significant obstacles when we activate pure literal elimination
- Can PNS bring any performance benefits?
  - Open question, need to design initialization heuristics for QBF

Related Work

<table>
<tr><th>Related work</th><th>My work</th></tr>
<tr><td>Proof number search (Allis, 1994)</td><td rowspan="4">PNS + Backjumping based QBF solver</td></tr>
<tr><td>QDLL algorithm (Cadoli, 1998)</td></tr>
<tr><td>2 watched literal data structure (Zhang, 2001)</td></tr>
<tr><td>Backjumping in QBF (Giunchiglia, 2001)</td></tr>
<tr><td>QCDCL (Zhang, Letz, Giunchiglia, 2002)</td><td rowspan="2">PNS + CDCL + SBJ based QBF solver
Show the difficulty of combining PNS + SDCL or activate pure literal elimination</td></tr>
<tr><td>MCTS + CDCL based SAT solver (Schloeter, 2017)</td></tr>
</table>

Future work

- Resolve the issue with PLE
- PNS and SDCL
- Parameter tuning
  - Initialization heuristics

Thanks