Cube and Conquer

Guiding CDCL SAT Solvers by Lookaheads

Marijn Heule[1,2], Oliver Kullmann[3], Siert Wieringa[4] and Armin Biere[1]

[1]Johannes Kepler University, Linz, Austria

[2]Delft University of Technology, Delft, The Netherlands

[3]Swansea University, United Kingdom
http://cs.swan.ac.uk/~csoliver

[4]Aalto University, Helsinki, Finland

Belgrade, February 3, 2012,

Fifth Workshop on Formal and Automated Theorem Proving and Applications

Best Paper Award at HVC 2011 [Heule et al., 2012]

Motivation

Context:

- Huge performance boost of CDCL solvers in the last decade
- CDCL solvers have become a crucial tool, e.g. in Formal Verification

Challenges:

- CDCL is not strong on small hard combinatorial problems
- CDCL is hard to parallelise effectively

CDCL: Conflict-Driven Clause Learning

Satisfiability problem

Satisfiability (SAT) problem:

- Given a formula in Conjunctive Normal Form, is there a truth assignment to the Boolean variables satisfying all clauses?
- clause: $(a \vee b \vee c)$ ("CNF-clause")
- cube: $(d \wedge e \wedge f)$ (alternatively, think of it as a partial assignment).

Major complete SAT solver architectures:

- Conflict-Driven Clause Learning
- Lookahead.

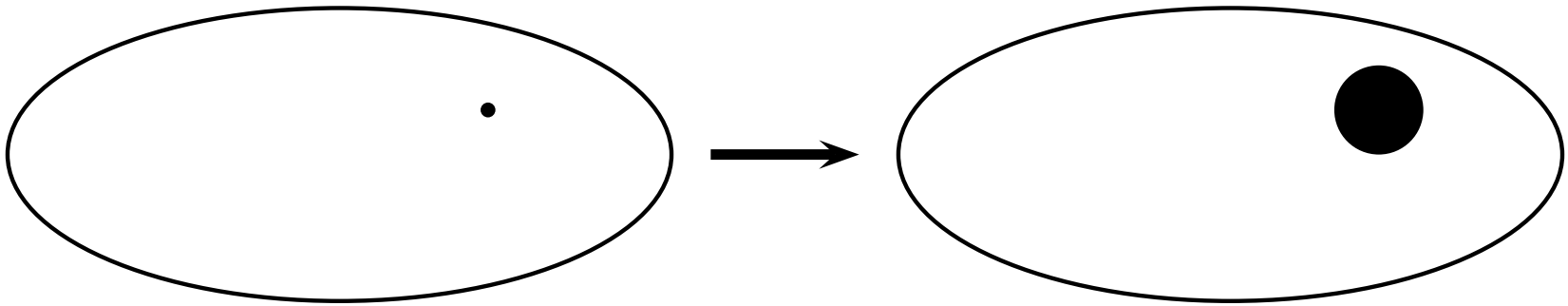
Conflict-Driven Clause Learning solvers

Highlights:

- goal: find small effective conflict clauses
- decisions: assign variables that occur in recent conflicts
- strength: powerful on "easy" problems

Ideal CDCL situation:

- hit a conflict that can be generalised / analysed to a small clause



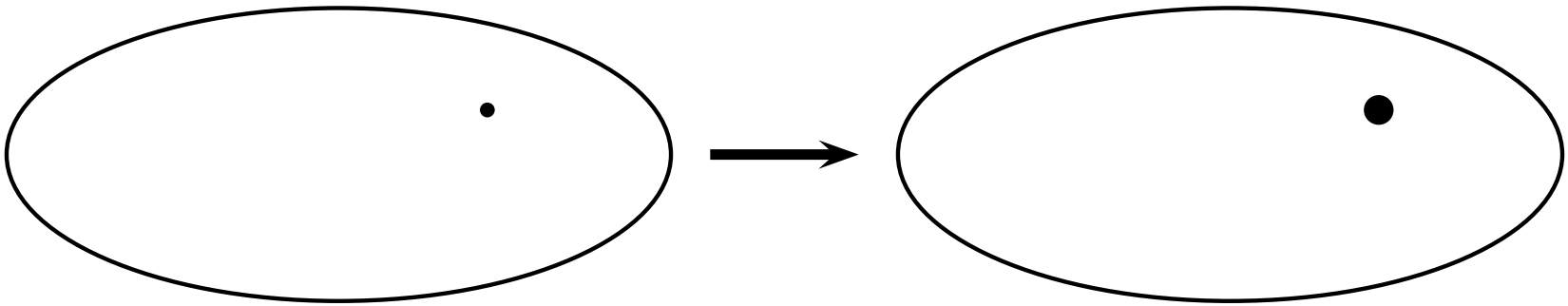
Conflict-Driven Clause Learning solvers

Highlights:

- goal: find small effective conflict clauses
- decisions: assign variables that occur in recent conflicts
- strength: powerful on "easy" problems

General CDCL situation:

- hit a conflict that can be generalised / analysed to a large clause



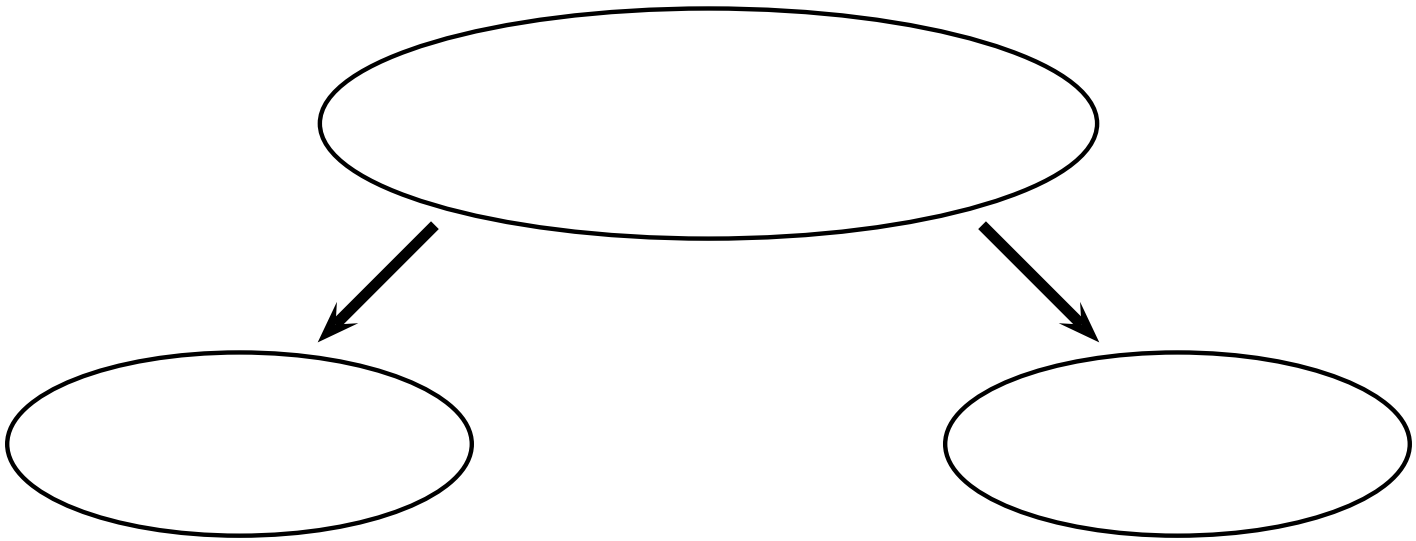
Lookahead solvers

Highlights:

- goal: construct a small binary search tree
- decisions: assign variables that cause a large reduction
- strength: powerful on small hard problems

Ideal lookahead situation:

- split the search space into two equally large but smaller parts



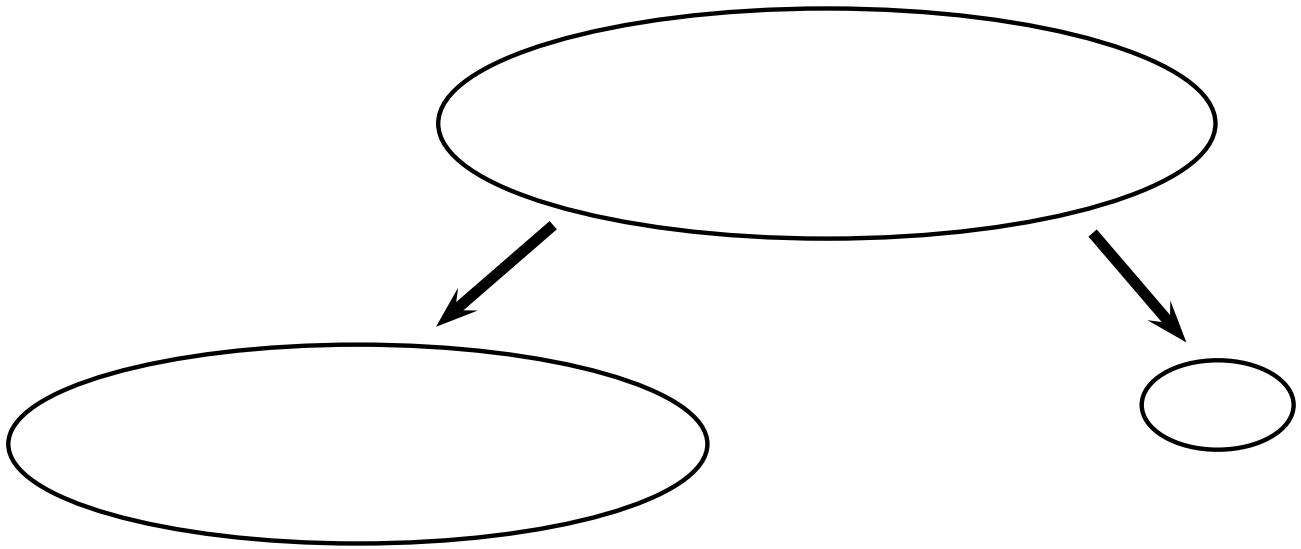
Lookahead solvers

Highlights:

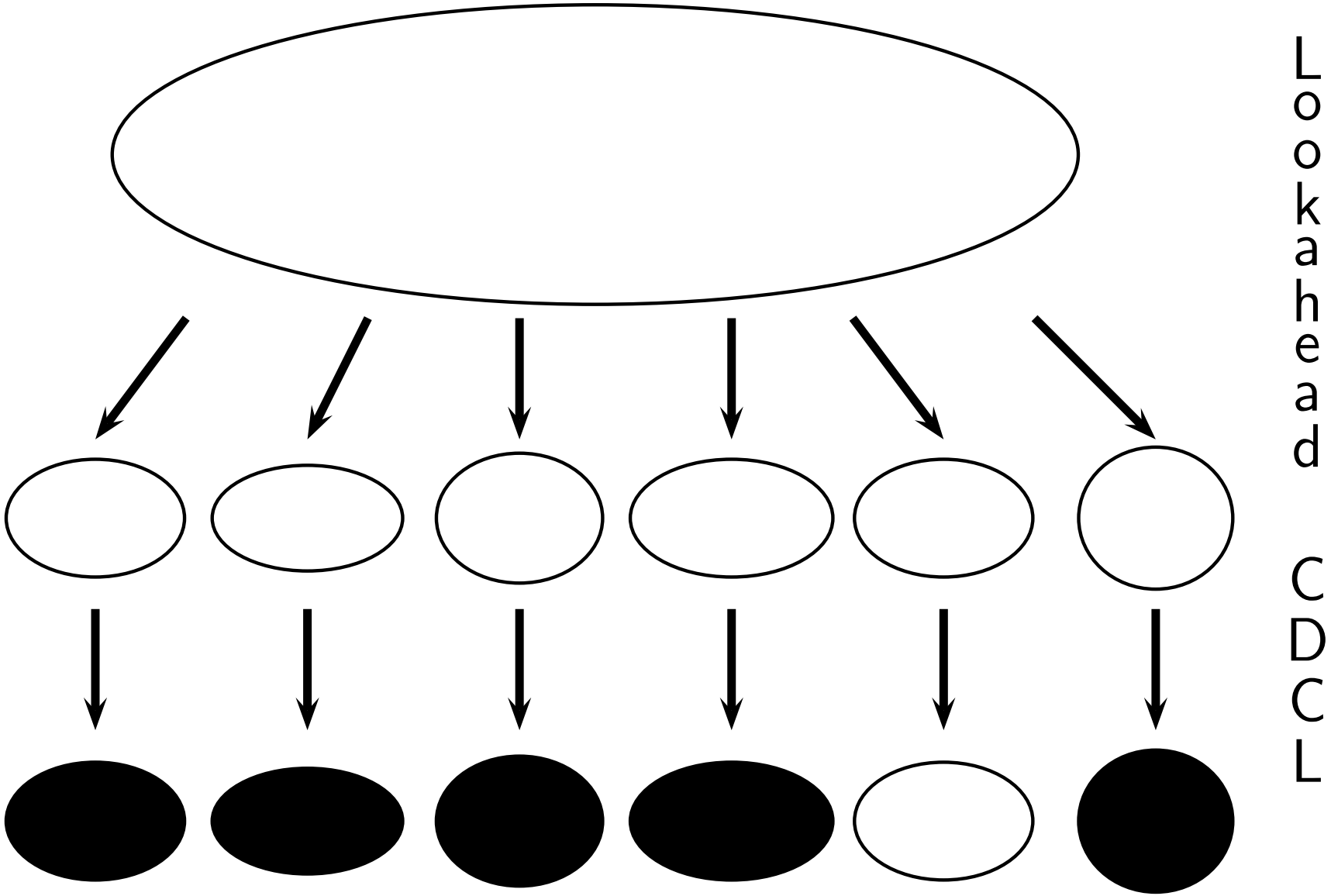
- goal: construct a small binary search tree
- decisions: assign variables that cause a large reduction
- strength: powerful on small hard problems

General lookahead situation:

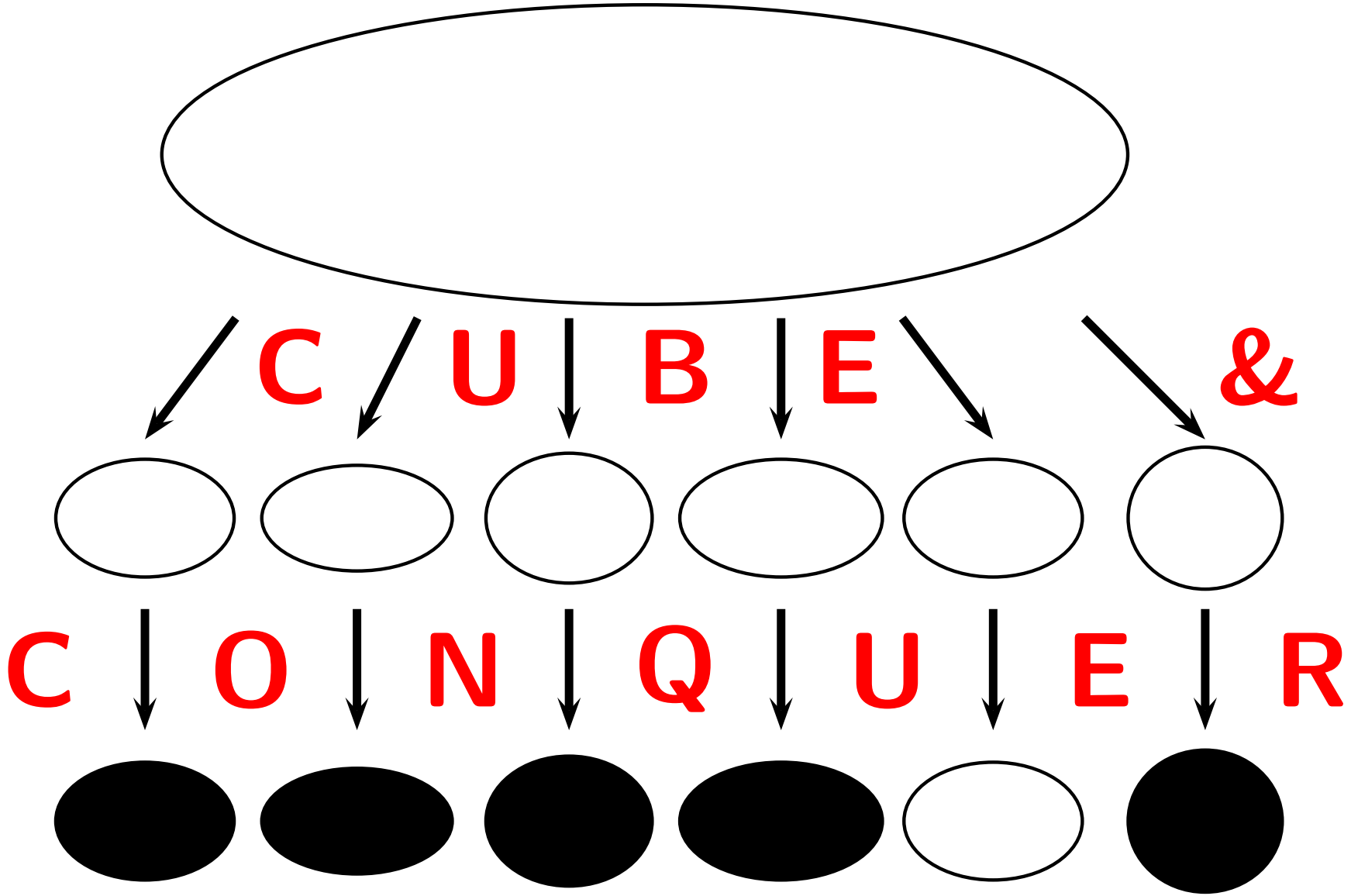
- the search space is split into a large and a small part



Best of both worlds: Combining Lookahead and CDCL



Best of both worlds: Cube and Conquer



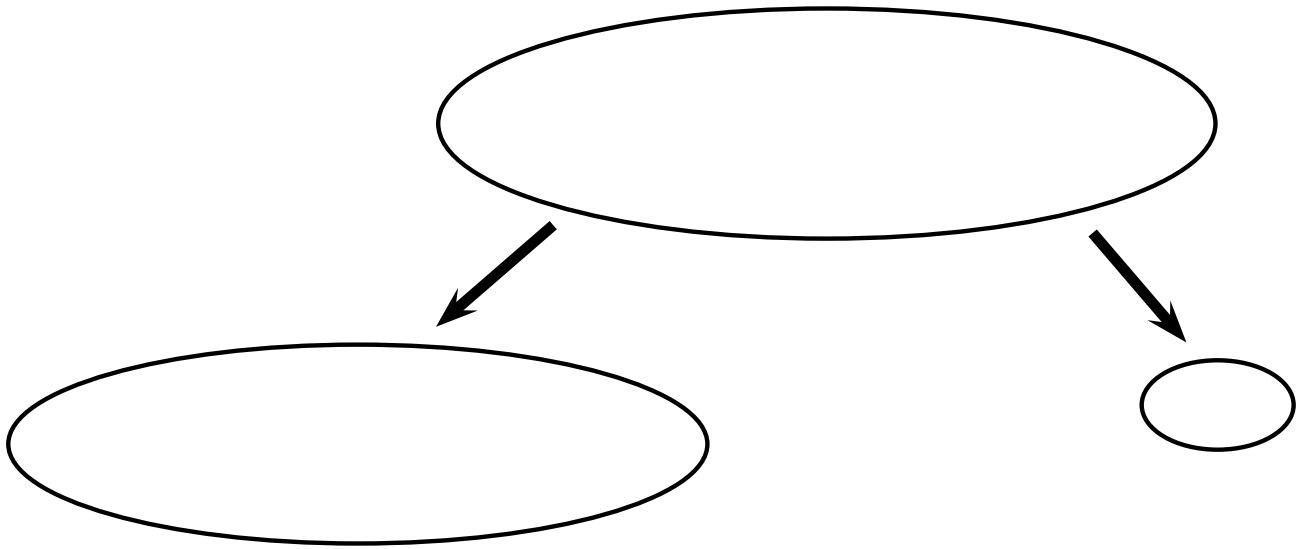
Cube: key observation / contribution

Split until the (sub-)problems become easy:

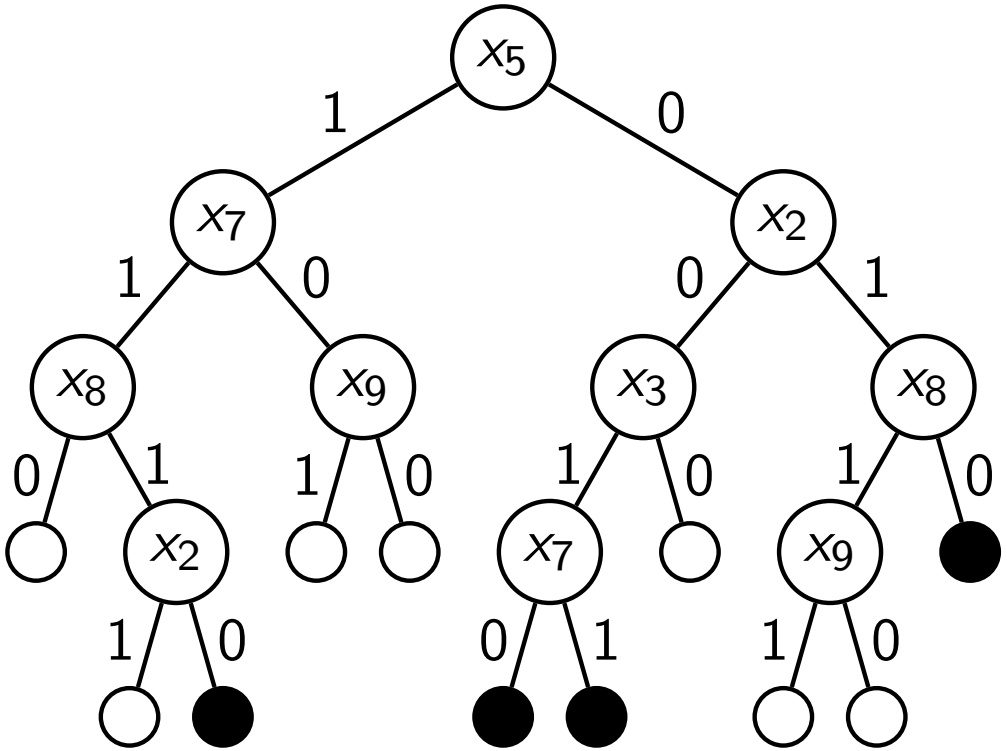
- do not have a fixed cut off depth
- determine hardness by (total) number of assigned variables
- create many thousands or even millions of cubes.

General lookahead situation:

- the search space is split into a large and a small part



Cube: example



cutoff leaf

refuted leaf

$$F_1 := F \wedge (x_5 \wedge x_7 \wedge \neg x_8)$$
$$F_2 := F \wedge (x_5 \wedge x_7 \wedge x_8 \wedge x_2)$$
$$F_3 := F \wedge (x_5 \wedge \neg x_7 \wedge x_9)$$
$$F_4 := F \wedge (x_5 \wedge \neg x_7 \wedge \neg x_9)$$

$$F_5 := F \wedge (\neg x_5 \wedge \neg x_2 \wedge \neg x_3)$$
$$F_6 := F \wedge (\neg x_5 \wedge x_2 \wedge x_8 \wedge x_9)$$
$$F_7 := F \wedge (\neg x_5 \wedge x_2 \wedge x_8 \wedge \neg x_9)$$

Cube: pseudo-code (1)

Cube(CNF F, DNF $\mathcal{A}$, CNF $\mathcal{C}$, dec. lits. φ_{dec}, imp. lits. φ_{imp})
returns a pair $(\mathcal{A}, \mathcal{C})$, the list of cubes and the list of learned clauses

1 $\theta := 1.05 \cdot \theta$
2 $(F, \varphi_{\text{imp}}) :=$ lookahead_simplify_and_learn(F, φ_{dec}, φ_{imp})
3 **if** $\varphi_{\text{dec}} \cup \varphi_{\text{imp}}$ falsify a clause in F **or** $|\varphi_{\text{dec}}| > 20$ **then** $\theta := 0.7 \cdot \theta$
4 **if** $\varphi_{\text{dec}} \cup \varphi_{\text{imp}}$ falsify a clause in F **then return** $(\mathcal{A}, \mathcal{C} \cup \{\neg\varphi_{\text{dec}}\})$
5 **if** cutoff heuristic is triggered **then return** $(\mathcal{A} \cup \{\varphi_{\text{dec}}\}, \mathcal{C})$
6 $l_{\text{dec}} :=$ lookahead_decide(F, φ_{dec}, φ_{imp})
7 $(\mathcal{A}, \mathcal{C}) :=$ Cube$(F, \mathcal{A}, \mathcal{C}, \varphi_{\text{dec}} \cup \{l_{\text{dec}}\}, \varphi_{\text{imp}})$
8 **return** Cube$(F, \mathcal{A}, \mathcal{C}, \varphi_{\text{dec}} \cup \{\neg l_{\text{dec}}\}, \varphi_{\text{imp}})$

Cube: pseudo-code (2)

Cube (CNF F, DNF $\mathcal{A}$, CNF $\mathcal{C}$, dec. lits. φ_{dec}, imp. lits. φ_{imp})

1 $\theta := 1.05 \cdot \theta$

2 $\langle F, \varphi_{\text{imp}} \rangle :=$ *lookahead_simplify_and_learn* $(F, \varphi_{\text{dec}}, \varphi_{\text{imp}})$

3 **if** $\varphi_{\text{dec}} \cup \varphi_{\text{imp}}$ falsify a clause in F **or** $|\varphi_{\text{dec}}| > 20$ **then** $\theta := 0.7 \cdot \theta$

4 **if** $\varphi_{\text{dec}} \cup \varphi_{\text{imp}}$ falsify a clause in F **then return** $\langle \mathcal{A}, \mathcal{C} \cup \{\neg\varphi_{\text{dec}}\} \rangle$

5 **if** $|\varphi_{\text{dec}}| \cdot |\varphi_{\text{dec}} \cup \varphi_{\text{imp}}| > \theta \cdot |\text{vars}(F)|$ **then return** $\langle \mathcal{A} \cup \{\varphi_{\text{dec}}\}, \mathcal{C} \rangle$

6 $l_{\text{dec}} :=$ *lookahead_decide* $(F, \varphi_{\text{dec}}, \varphi_{\text{imp}})$

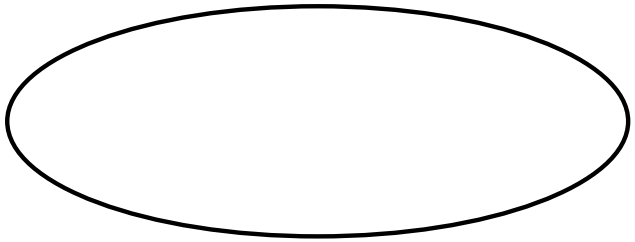
7 $\langle \mathcal{A}, \mathcal{C} \rangle :=$ *Cube* $(F, \mathcal{A}, \mathcal{C}, \varphi_{\text{dec}} \cup \{l_{\text{dec}}\}, \varphi_{\text{imp}})$

8 **return** *Cube* $(F, \mathcal{A}, \mathcal{C}, \varphi_{\text{dec}} \cup \{\neg l_{\text{dec}}\}, \varphi_{\text{imp}})$

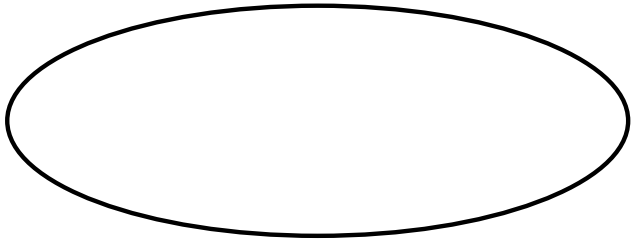
Conquer: describing cubes

How much information to send to the CDCL solver?

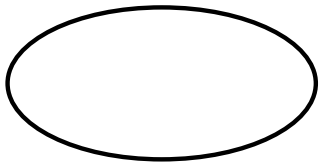
- Only the decisions


$\wedge \ \varphi_{\text{dec}}$

- The full assignment (including failed literals)


$\wedge \ \varphi_{\text{dec}} \ \wedge \ \varphi_{\text{imp}}$

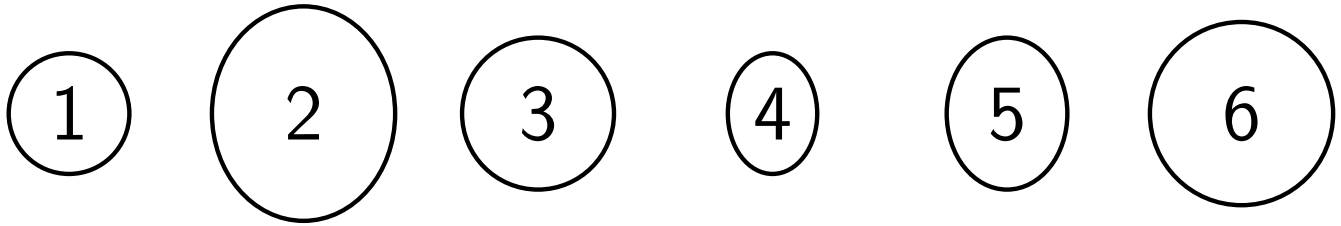
- The simplified formula (including local learnt clauses)



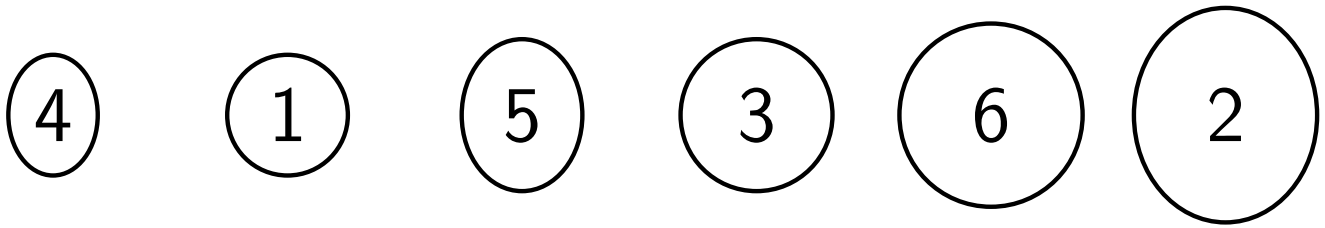
Conquer: ordering cubes

What is the optimal order to solve the cubes?

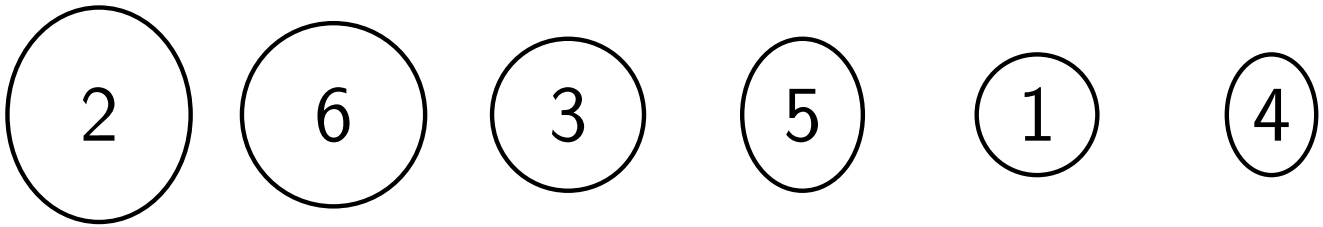
- Depth-first search (in lookahead order)



- Solves cubes with increasing (approximated) search space



- Solves cubes with decreasing (approximated) search space



Conquer: pseudo-code

Conquer (CDCL solver S, CNF formula F, DNF of assumptions $\mathcal{A}$)

```
1  S.Load (F)
2  while A is not empty do
3     get a cube c from A and remove c from A
4     if S.SolveWithAssumptions (c) = satisfiable then
5        return satisfiable
6     S.AnalyzeFinal ()
7     S.ResetClauseDeletionPolicy ()
8  return unsatisfiable
```

Conquer: parallel solving

Strategies to solve cubes in parallel:

1. cores solve different cubes in parallel
2. cores solve the same cube in parallel
3. start with (1) till no new cubes are available, continue with (2)

What to share between cores?

- nothing, so hardly communication required (only ask / receive cubes)
- sharing the *AnalyzeFinal* clauses (maybe only to master)
- sharing the short conflict clauses, units (maybe also binaries)

Conquer: parallel solving

Strategies to solve cubes in parallel:

1. cores solve different cubes in parallel
2. cores solve the same cube in parallel
3. start with (1) till no new cubes are available, continue with (2)

What to share between cores?

- nothing, so hardly communication required (only ask / receive cubes)
- sharing the *AnalyzeFinal* clauses (maybe only to master)
- sharing the short conflict clauses, units (maybe also binaries)

Results: two experiments

1^{st} experiment: single core on Van der Waerden numbers

- hard combinatorial problem in Ramsey Theory
- comparison with the best solver for each instance
- cube solver: `OKsolver`
- conquer solver: `minisat-2.2.0`
- describing the cubes (just) by the naked simplified formula (applying the partial assignments; without any local learning).

2^{nd} experiment: multi core on challenging applications

- unsolved application instances from the SAT09 benchmarks
- comparison with the best parallel solvers
- cube solver: `march`
- conquer solver: `lingeling`.

Results: palindromic Van der Waerden numbers

- k_1 : arithmetic progression of first set
- k_2 : arithmetic progression of second set
- n : number of variables
- best solver : time of fastest sequential solver
- D : cut off depth.

k_1	k_2	n	#cls	?	best solver	D	#cubes	C&C
3	25	294	45779	U	$\sim$ 13 days	45	9120	6.5 hours
3	25	304	49427	U	$\sim$ 13 days	45	13462	2 days
4	12	194	15544	U	$>$ 14 days	30	132131	2 days
4	12	198	15889	U	$>$ 14 days	34	147237	8 hours
5	8	157	9121	U	3.5 days	20	2248	5 hours
5	8	162	9973	U	53 days	20	87667	40 hours

See [Ahmed et al., 2011, Kullmann, 2012].

Results: parallel SAT solving

Portfolio solvers:

- run multiple versions of the same solver (different seeds)
- share short conflict clauses such as units
- solver `pLingeling` (`pLing`), on a 12-core machine

Grid based SAT solving approach:

- run solvers with different cubes on a grid
- grid constraints: limited communication, possible delay and timeout
- solver `PartitionTree` (`PTree`) on a grid, up to 60 jobs in parallel

Results: hard application benchmarks

Benchmark	?	#cubes	I total	II total	II 12-core	pLing 12-core	PTree 60-core
9dlx_vliw_at_b_iq8	U	121	150	—	—	**3256**	—
9dlx_vliw_at_b_iq9	U	100	179	—	—	**5164**	—
AProVE07-25	U	84247	89	100340	**8690**	—	9967
dated-5-19-u	U	57716	418	3214	**1451**	4465	2522
eq.atree.braun.12	U	86541	85	3261	**273**	—	4691
eq.atree.braun.13	U	81313	77	18165	**1517**	—	9972
gss-24-s100	S	18237	48	4975	**415**	2930	3492
gss-26-s100	S	19455	57	37259	**3108**	18173	10347
gus-md5-14	U	60102	961	—	—	—	**13890**
ndhf_xits_09_UNS	U	37358	82	71096	12041	—	**9583**
rbcl_xits_09_UNK	U	54669	132	94911	11542	—	**9819**
rpoc_xits_09_UNS	U	30681	114	48028	**8366**	—	8635
sortnet-8-ipc5-h19	S	724	153	48668	4067	**2700**	4304
total-10-17-u	U	9192	288	5638	4517	**3672**	4447
total-5-15-u	U	14914	215	—	—	—	**18670**

Results: hard application benchmarks

Benchmark	?	#cubes	I total	II total	II 12-core	pLing 12-core	PTree 60-core
9dlx_vliw_at_b_iq8	U	121	150	—	—	**3256**	—
9dlx_vliw_at_b_iq9	U	100	179	—	—	**5164**	—
AProVE07-25	U	84247	89	100340	**8690**	—	9967
dated-5-19-u	U	57716	418	3214	**1451**	4465	2522
eq.atree.braun.12	U	86541	85	3261	**273**	—	4691
eq.atree.braun.13	U	81313	77	18165	**1517**	—	9972
gss-24-s100	S	18237	48	4975	**415**	2930	3492
gss-26-s100	S	19455	57	37259	**3108**	18173	10347
gus-md5-14	U	60102	961	—	—	—	**13890**
ndhf_xits_09_UNS	U	37358	82	71096	12041	—	**9583**
rbcl_xits_09_UNK	U	54669	132	94911	11542	—	**9819**
rpoc_xits_09_UNS	U	30681	114	48028	**8366**	—	8635
sortnet-8-ipc5-h19	S	724	153	48668	4067	**2700**	4304
total-10-17-u	U	9192	288	5638	4517	**3672**	4447
total-5-15-u	U	14914	215	—	—	—	**18670**

Results: hard application benchmarks

Benchmark	?	#cubes	I total	II total	II 12-core	pLing 12-core	PTree 60-core
9dlx_vliw_at_b_iq8	U	121	150	—	—	**3256**	—
9dlx_vliw_at_b_iq9	U	100	179	—	—	**5164**	—
AProVE07-25	U	84247	89	100340	**8690**	—	9967
dated-5-19-u	U	57716	418	3214	**1451**	4465	2522
eq.atree.braun.12	U	86541	85	3261	**273**	—	4691
eq.atree.braun.13	U	81313	77	18165	**1517**	—	9972
gss-24-s100	S	18237	48	4975	**415**	2930	3492
gss-26-s100	S	19455	57	37259	**3108**	18173	10347
gus-md5-14	U	60102	961	—	—	—	**13890**
ndhf_xits_09_UNS	U	37358	82	71096	12041	—	**9583**
rbcl_xits_09_UNK	U	54669	132	94911	11542	—	**9819**
rpoc_xits_09_UNS	U	30681	114	48028	**8366**	—	8635
sortnet-8-ipc5-h19	S	724	153	48668	4067	**2700**	4304
total-10-17-u	U	9192	288	5638	4517	**3672**	4447
total-5-15-u	U	14914	215	—	—	—	**18670**

Conclusions

Cube and Conquer:

- effectively combining lookahead and CDCL
- many thousands or even million of cubes
- natural to parallelise

Future work, online scheduling:

- adjust heuristics based on *AnalyzeFinal*
- communication between solvers
- all-in CDCL

Future work, theoretical foundations:

- create a proof-theoretic framework for understanding "tree-like versus dag-like resolution" and their interaction
- better understanding of cdcl-proof-systems in this context.

Bibliography I


Ahmed, T., Kullmann, O., and Snevily, H. (2011).
On the van der Waerden numbers $w(2;3,t)$.
Technical Report arXiv:1102.5433v2 [math.CO], arXiv.


Heule, M. J., Kullmann, O., Wieringa, S., and Biere, A. (2012).
Cube and conquer: Guiding CDCL SAT solvers by lookaheads.
To appear in LNCS, HVC 2011; available at
`http://cs.swan.ac.uk/~csoliver/papers.html`.


Kleine Büning, H. and Kullmann, O. (2009).
Minimal unsatisfiability and autarkies.
In Biere, A., Heule, M. J., van Maaren, H., and Walsh, T., editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 11, pages 339–401. IOS Press.

Bibliography II


Kullmann, O. (2012).
Computing ordinary and palindromic van der Waerden numbers via collaboration between look-ahead and conflict-driven SAT solvers.
In preparation.

End