DESIGN, AUTOMATION & TEST IN EUROPE
09 – 13 March 2020 · ALPEXPO · Grenoble · France
The European Event for Electronic System Design & Test

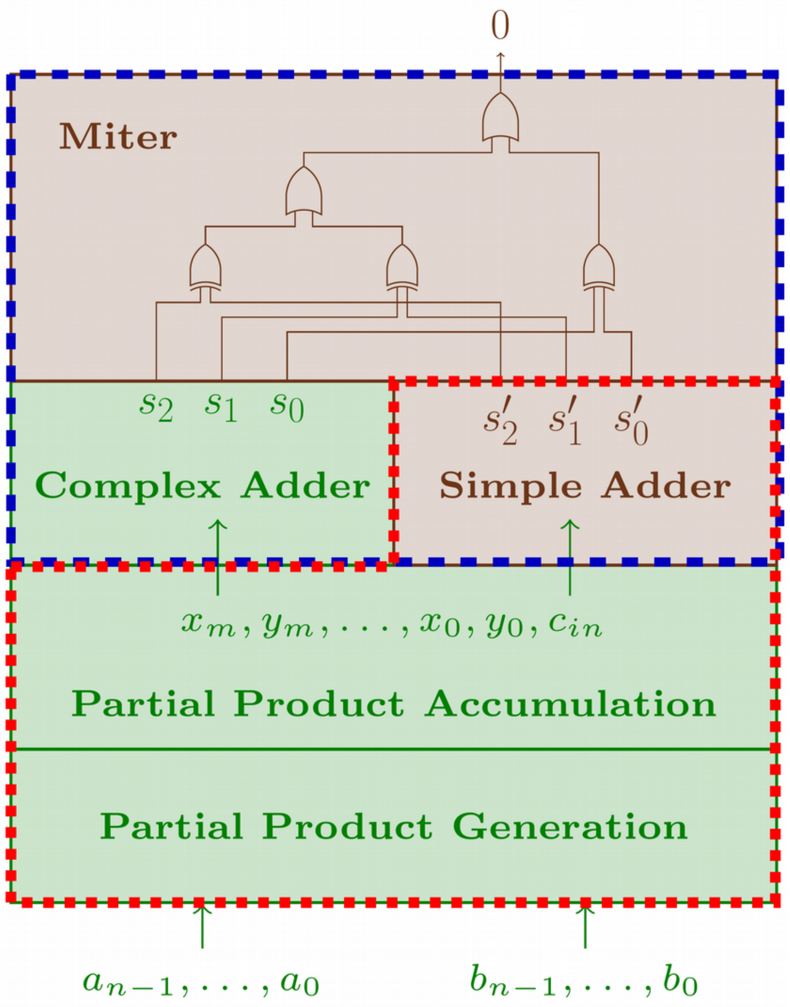
From DRUP to PAC and Back

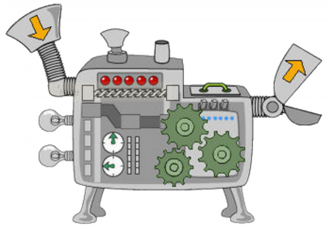
Daniela Kaufmann Armin Biere Manuel Kauers

Der Wissenschaftsfonds.
JOHANNES KEPLER UNIVERSITÄT LINZ

DATE'20

Motivation





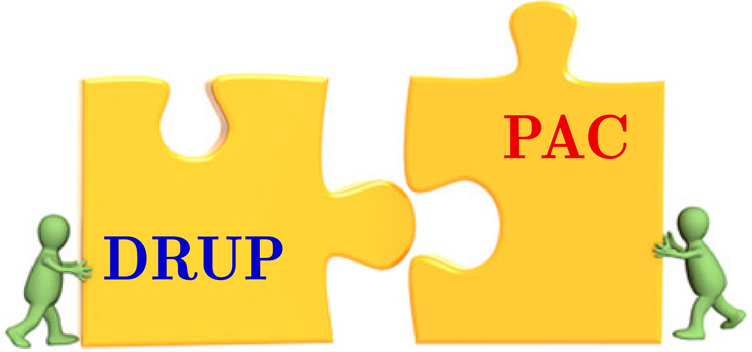
SAT

Computer Algebra

DRUP

Practical Algebraic Calculus

?



Computer Algebra & PAC

- Let $f \in \mathbb{Z}[X]$ and $P \subseteq \mathbb{Z}[X]$. We are interested whether the polynomial equation $f = 0$ is implied by the equations $p = 0$ with $p \in P$, i.e., decide $f \in \langle P \rangle$.
- All variables $x \in X$ represent logic gates and thus take only values in $\{0, 1\}$. This is enforced by *Boolean value constraints*. Let $B(X) = \{x(1-x) \mid x \in X\} \subseteq \mathbb{Z}[X]$ be the set of Boolean value constraints for X.
- PAC proofs are sequences of proof rules, which model the ideal properties:

$+ : p_i,\ p_j,\ p_i + p_j;$	p_i, p_j appearing earlier in the proof or are contained in constraint set P and $p_i + p_j$ being reduced by $B(X)$
$* : p_i,\ q,\ qp_i;$	p_i appearing earlier in proof or in P and $q \in \mathbb{Z}[X]$ being arbitrary and qp_i being reduced by $B(X)$

Example: Let $P = \{-x + 3z, 2xz\} \subseteq \mathbb{Z}[x,y,z]$ and let $f = -2x \in \mathbb{Z}[x,y,z]$. The proof shows $f \in \langle P \cup B(X) \rangle$:

```
* :   -x+3z,   2x, -2x+6xz;
* :     2xz,   -3, -6xz;
+ : -2x+6xz, -6xz, -2x;
```

SAT & DRUP

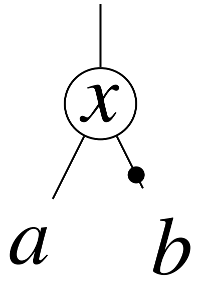
- The SAT problem seeks for an assignment such that a formula F in conjunctive normal form evaluates to **true**. If no satisfying assignment can be found it is *unsatisfiable*.
- The most basic clausal proof format is *reverse unit propagation* (RUP).
  Let $\overline{C}$ denote the negation of a clause C. We say C is a *RUP clause* if $F \wedge \overline{C}$ evaluates to **false**. A RUP proof is a sequence of RUP clauses containing the empty clause. DRUP extends RUP by adding deletion information.

Example: This is an unsatisfiable CNF in DIMACS format (left) with DRUP (middle) and TraceCheck (right) proofs.

```
p cnf 3 5         -2 0         1  1 -2 -3 0 0
 1 -2 -3  0     d  3 0         2  1  2  0 0
 1  2      0     d  1 -2 -3 0  3 -1 -2  0 0
-1 -2      0     d -1 -2 0     4 -1  2  0 0
-1  2      0         0         5  3  0  0
       3  0                    6 -2  0 3 1 5 0
                               7  0  4 2 6 0
```

From DRUP to PAC

1. Polynomial encoding of CNF



	Propostional Formula	Polynomial Relation
	$(x \leftrightarrow a \wedge \overline{b}) = \top$	$-x + a(1-b) = 0$
CNF	$(x \vee \overline{a} \vee b) = \top$	$(1-x)a(1-b) = 0$
	$(\overline{x} \vee a) = \top$	$x(1-a) = 0$
	$(\overline{x} \vee \overline{b}) = \top$	$xb = 0$

Using the fact that $x^2 - x = 0,\ b^2 - b = 0$ and $a^2 - a = 0$ we multiply the polynomial equation $-x + a(1-b)$ by different factors to derive the desired polynomials.

$$\begin{aligned}
0 = &\quad (-x + a(1-b))(-ba + a) &&= (1-x)a(1-b) \\
0 = &\quad (-x + a(1-b))(b-1) &&= x(1-a) \\
0 = &\quad (-x + a(1-b))(-a) &&= xb
\end{aligned}$$

From DRUP to PAC

2. Encoding of resolution steps

- Add bit-flipping polynomials (similar to “Polynomial Calculus with Resolution”):
  Clause $x \vee y \vee z$ can be translated to $(1-x)(1-y)(1-z) = 0$, which generates $2^3 - 1$ monomials.
  If we introduce $-f_x + 1 - x = 0$, $-f_y + 1 - y = 0$, $-f_z + 1 - z = 0$, the same equation can be depicted as $f_x f_y f_z = 0$.

- Encode resolution using the traces provided in TraceCheck:

```
1  1 -2 -3 0 0
2  1  2  0 0
3 -1 -2  0 0
4 -1  2  0 0
5  3  0  0
6 -2  0 3 1 5 0
7  0  4 2 6 0
```

Let $a = $ `1`, $b = $ `2` and $c = $ `3`.
We encode the first resolution step of rule 6 (resolving clause 3 and 1).
Thus from $a \vee \overline{b} \vee \overline{c}$ and $\overline{a} \vee \overline{b}$ we resolve the clause $\overline{b} \vee \overline{c}$.

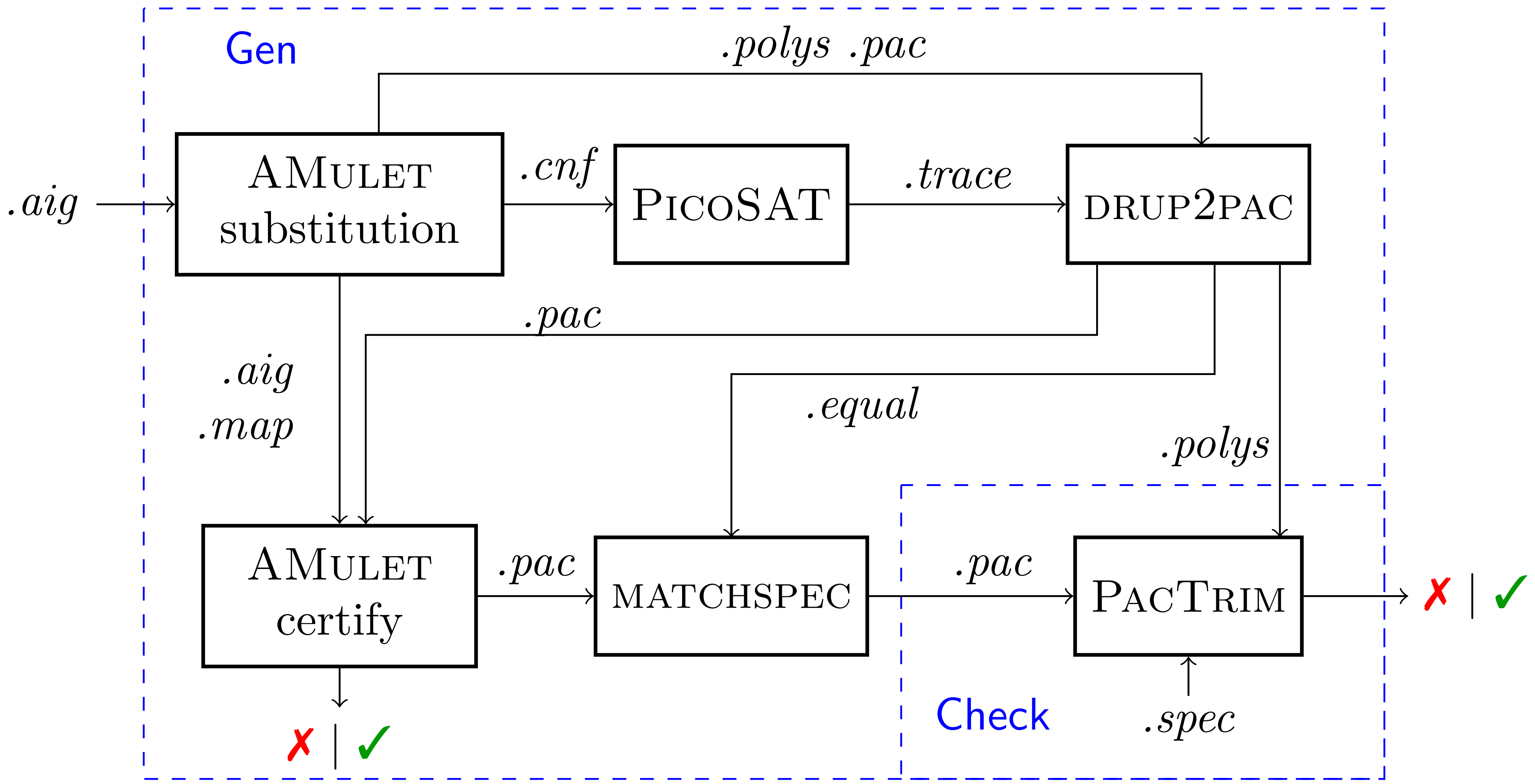
The corresponding PAC encoding is:

```
* :       b*a,     c,  c*b*a;
+ : -c*b*a+c*b, c*b*a,    c*b;
```

3. Merge PAC proofs

PAC proofs can be merged by combining constraint sets and proof rules.

From DRUP to PAC



From PAC to DRUP

1. **SMT encoding**

- The polynomial proof is translated into a bit-vector proof.
- To this end we encode the PAC proof as an SMT problem over the theory of quantifier free fixed size bit vectors.
- Each PAC rule is individually translated to SMT.
- Each variable in the PAC proof represents the input or output of a gate As a consequence we encode each variable in the PAC proof as a single bit and the coefficients are encoded as bit vectors.
- **Gap:** It is not checked that the specification is derived at the end.

From PAC to DRUP

1. Encode as SMT formula

Consider the following PAC rule

$$+ : 3x - z,\ 2y - 3x,\ 2y - z;$$

Checking the correctness of this rule can be encoded as:

```
(set-logic QF_BV)
(declare-fun x () (_ BitVec 1))
(declare-fun y () (_ BitVec 1))
(declare-fun z () (_ BitVec 1))
(assert
(let (($v0 (bvadd (bvand #b011 ((_ sign_extend 2) x))
      (bvand #b111 ((_ sign_extend 2) z)))))
(let (($w0 (bvadd (bvand #b010 ((_ sign_extend 2) y))
      (bvand #b101 ((_ sign_extend 2) x)))))
(let (($p0 (bvadd (bvand #b010 ((_ sign_extend 2) y))
      (bvand #b111 ((_ sign_extend 2) z)))))
(let (($e0 (= (bvadd $v0 $w0)  $p0)))

(not $e0))))))
(check-sat)
```

From PAC to DRUP

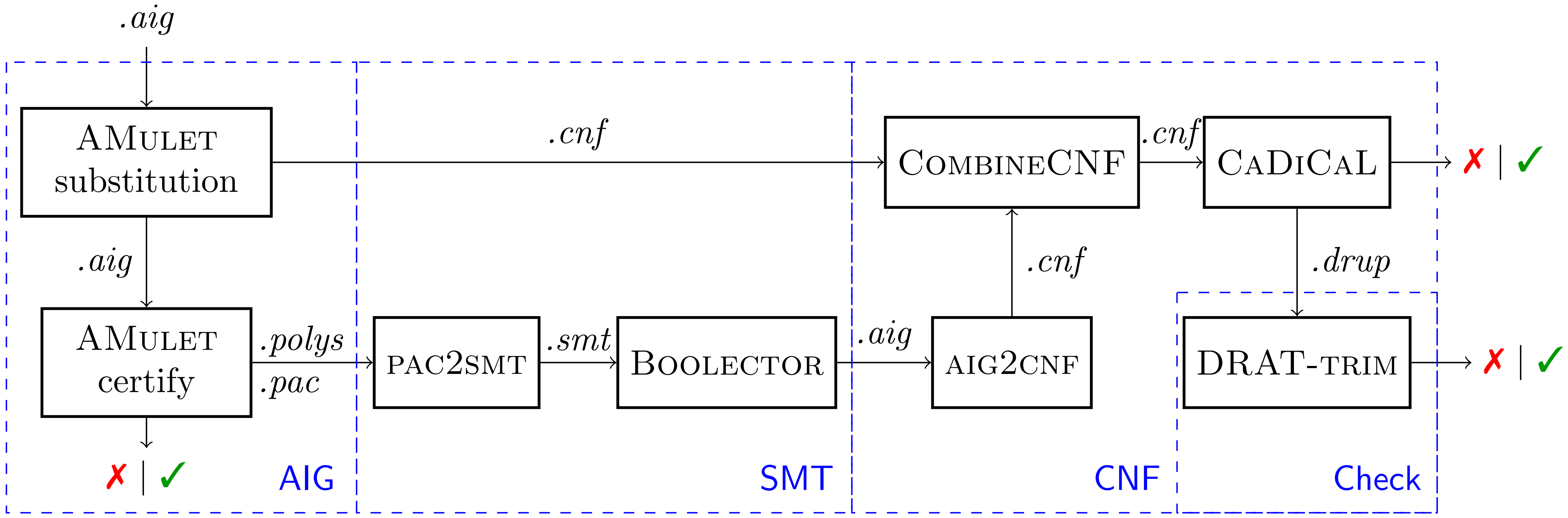
2. SMT to CNF

- SMT encoding is given to SMT solver Boolector.
- **Gap:** Internal rewriting steps are not covered.
- Boolector generates an And-Inverter-Graph that is translated into CNF.

3. Merge proofs

- Collect all clauses of both CNFs, except for output assumptions.
- The two output clauses $C_0 = l_0, C_1 = l_1$ are merged into the clause $l_0 \vee l_1$.
- Merged CNF is solved using SAT solver and DRUP proof is generated.

From PAC to DRUP



Experimental Results

architecture	n	Separate Proofs							PAC				DRUP					
		DRUP			PAC			total										
		gen	check	size	gen	check	size		gen	check	total	size	aig	smt	cnf	check	total	size
sp-ar-cl	16	0	0	1 299	0	0	7 962	0	2	2	3	185 588	0	7	300	264	570	19 317 884
sp-dt-lf	16	0	0	1 167	0	0	7 787	0	1	1	2	136 349	0	6	279	277	562	18 153 668
bp-ct-bk	16	0	0	1 029	0	0	7 205	0	1	1	2	128 720	0	7	TO	-	-	-
bp-wt-cl	16	0	0	2 902	0	0	7 946	0	30	11	41	614 742	0	7	TO	-	-	-
sp-ar-cl	32	0	0	14 927	0	1	33 834	1	133	31	164	1 597 897	0	56	TO	-	-	-
sp-dt-lf	32	0	0	3 138	0	1	33 451	1	2	3	5	321 720	0	52	TO	-	-	-
bp-ct-bk	32	0	0	2 276	0	1	27 312	1	1	2	3	217 128	0	49	TO	-	-	-
bp-wt-cl	32	1	1	46 502	0	1	30 561	2	3 133	242	3 375	5 536 176	0	55	TO	-	-	-

PPG: simple (sp), Booth (bp)

PPA: array (ar), Dadda tree (dt), compressor tree (ct), Wallace tree (wt)

FSA: carry look-ahead (cl), Ladner-Fischer (lf), Brent-Kung (bk)

TO = 3600 sec

Conclusion

From DRUP to PAC:

- requires algebraic reasoning
- include bit-flipping techniques to reduce size
- use TraceCheck format

From PAC to DRUP:

- encode PAC proof as an SMT problem
- translated into CNF using bit-blasting
- leaves gaps in the proof

Single DRUP proofs are three orders of magnitude larger than PAC proofs and contain gaps.

From DRUP to PAC and Back

Daniela Kaufmann Armin Biere Manuel Kauers

FWF
Der Wissenschaftsfonds.

JKU
JOHANNES KEPLER
UNIVERSITÄT LINZ

RiSE

DATE’20