Decision Procedures in Hardware Design

Armin Biere

Institute for Formal Models and Verification
Johannes Kepler University
Linz, Austria

Dagstuhl Seminar 10161

Decision Procedures in Software, Hardware and Bioware

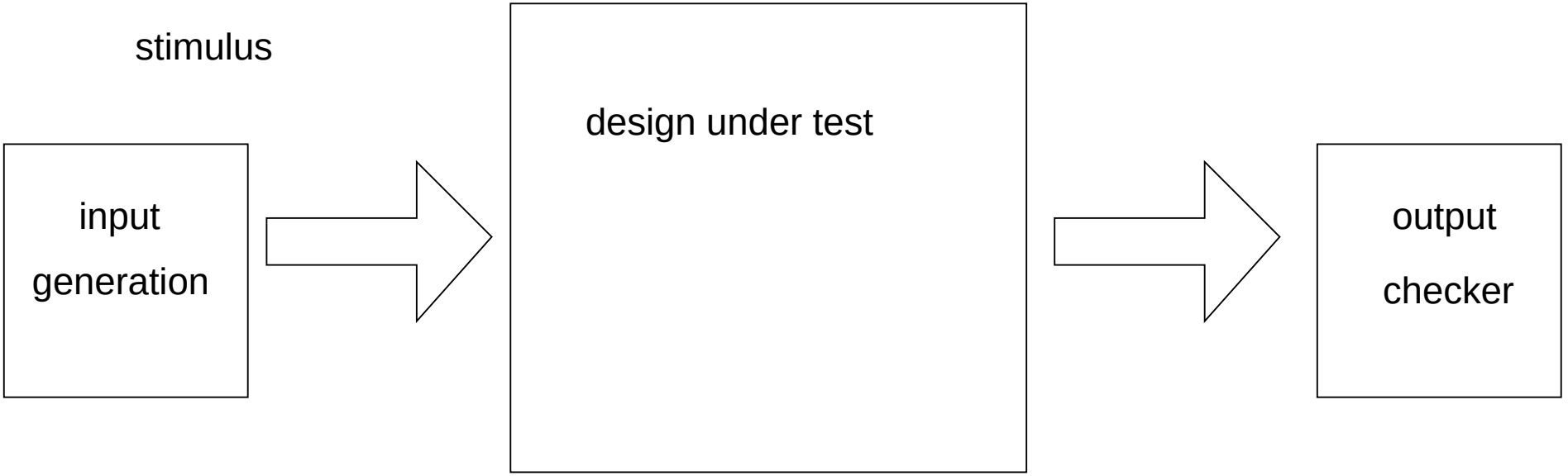
Schloss Dagstuhl, Germany

April 2010

Overview

- Decision Procedures for Hardware Design and Verification
  - constrained random simulation[1]
  - semi-formal[2] and bounded model checking[3]
  - equivalence checking[4] and model checking[5]
  - inductive techniques[6] and theorem proving[7]
  - combinational[8] and sequential[9] synthesis
- New Results in SAT Preprocessing
  - BCE = blocked clause elimination
  - witnesses for mixing BCE with equivalence reasoning

Constrained Random Simulation[1]



procedural

(non-synthesizable) Verilog

e.g. compute checksum of input packet

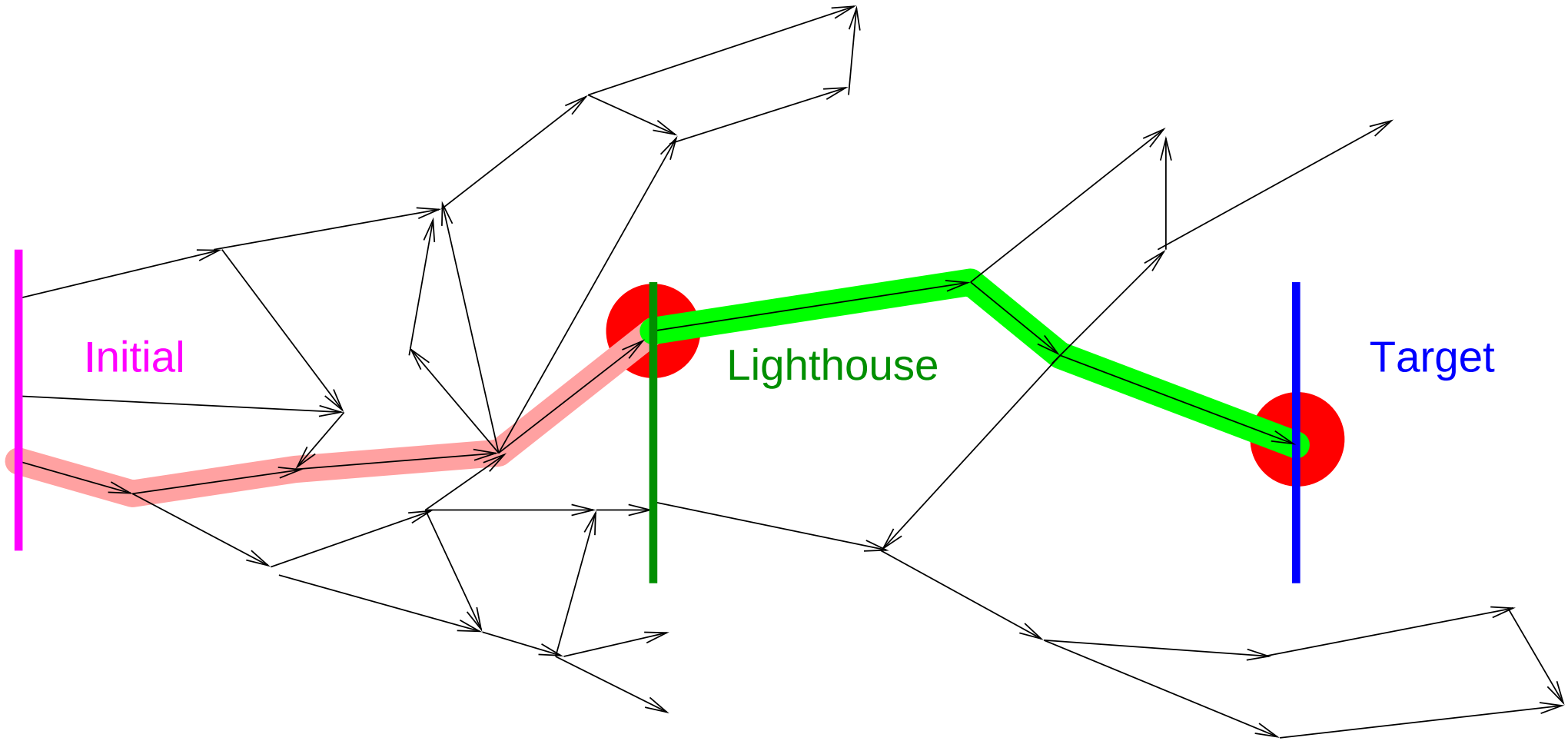
use random number generators

declarative

constraints

e.g. assert checksum field to be correct

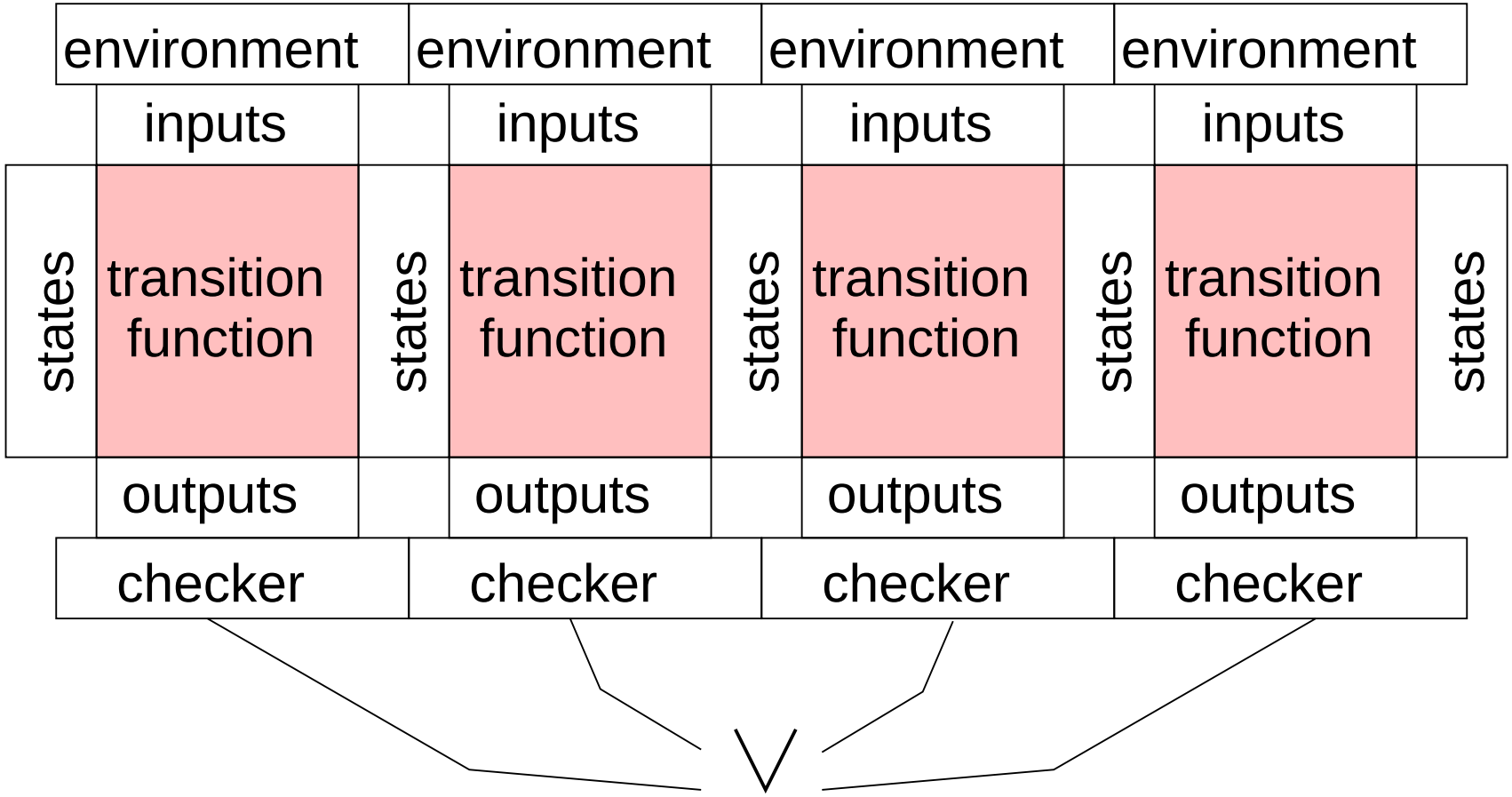
specify distribution of input values



hit functional coverage target, specify light house as intermediate step

focus on coverage not on formal property verification

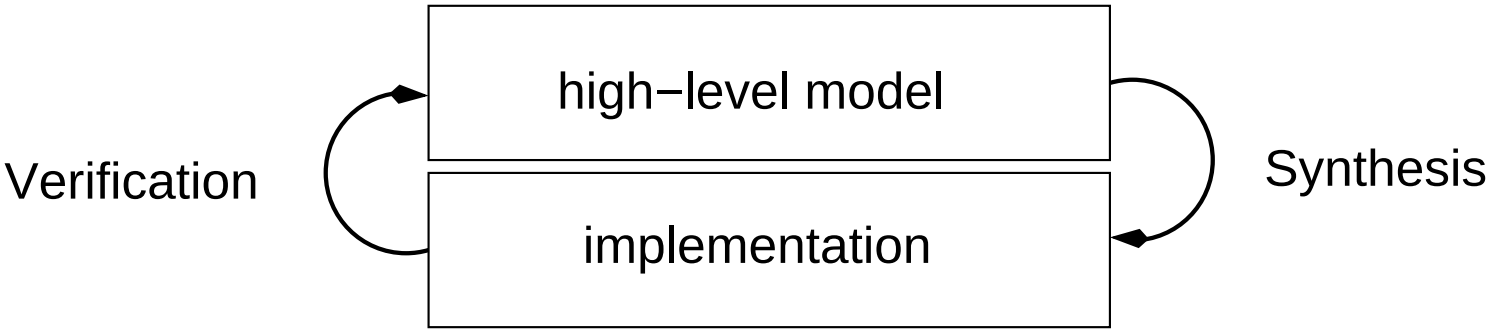
Bounded Model Checking[3]



check temporal properties, focus on falsification, use SAT solvers

used in many companies effectively

Equivalence Checking[4]



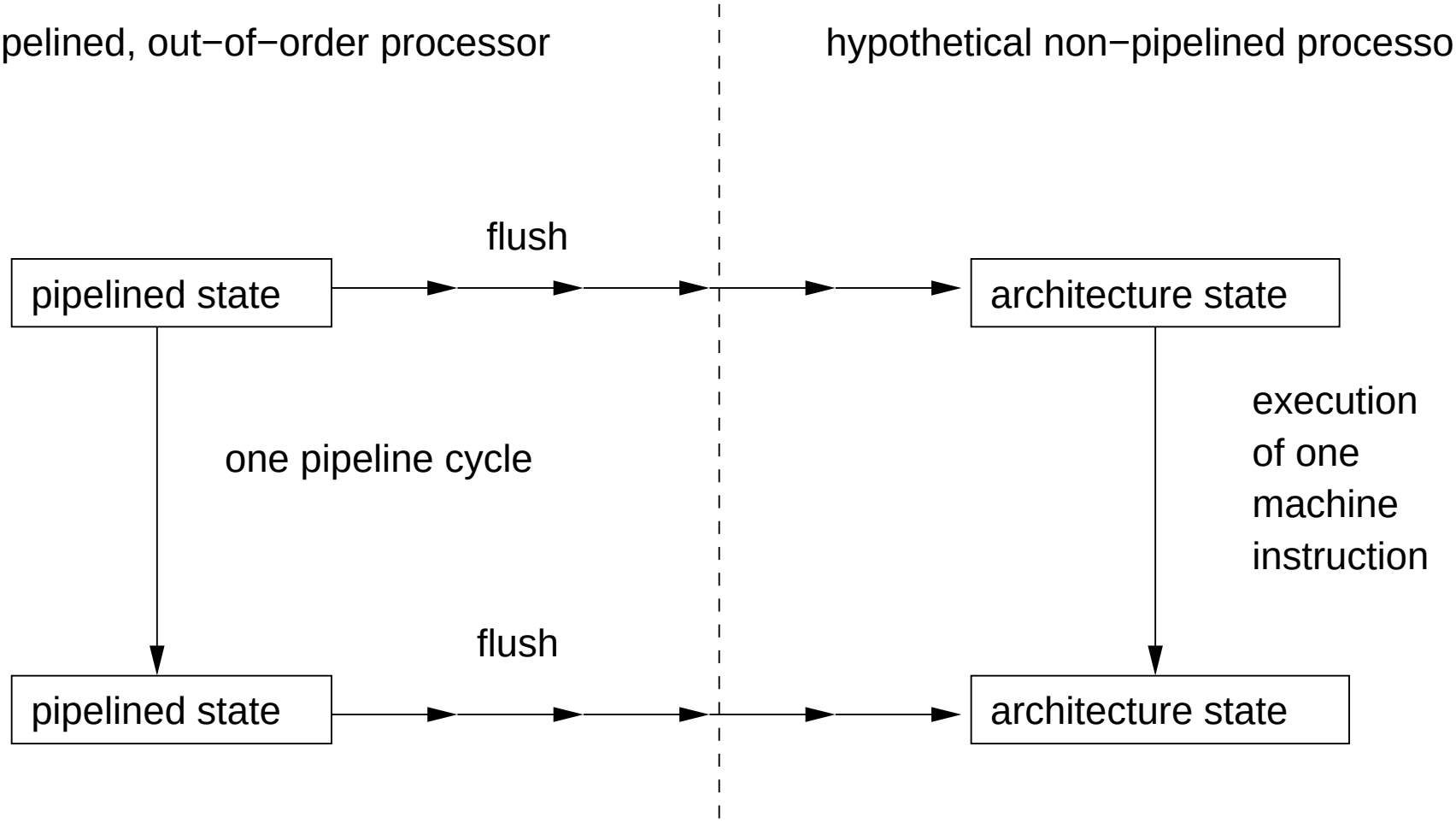
high-level model	implementation
RTL: Verilog, VHDL	gate- or switch-level
C models	Verilog

RTL vs gate in widespread use

Model Checking[5]

- check temporal properties of RTL models
  - *verification* instead of *falsification* as focus
  - needs quantifier elimination, BDDs, k-induction or interpolation
  - which in turn scales not as good $\Rightarrow$ capacity issues
- **commercially far less successful** than equivalence checking
  - specifications in temporal logic are hard to obtain
  - modeling the environment is even harder
- capacity is increasing, but not much research in academia
  - 3rd Hardware Model Checking Competition this year at FLOC
- *explicit model checking for protocols is working*

how to prove pipelined processor to implement the architecture



needs additional state invariants, completion-function as inductive invariant for flush

- reachable states as strongest inductive invariant
  - hard to compute; symbolic reachability is PSPACE complete $\Rightarrow$ QBF
  - shows that inductive invariants can always be used (in theory)

- k-inductive invariants do not have to be inductive (= 1-inductive)

  let $\mathbf{G}^k p \equiv \bigwedge_{i=0}^{k-1} \mathbf{X}^i p$, p is k-inductive iff $M \models \mathbf{G}^k p \rightarrow \mathbf{X}\mathbf{G}^k p$ iff $M \models \mathbf{G}^k p \rightarrow \mathbf{X}^k p$

  - strengthening of properties (with other invariants) still useful

- symbolic trajectory evaluation (STE) and also OneSpin's approach
  - user specifies property automaton: assume / guarantees on transitions
  - over design variables, inductively show guarantees in parallel product

Theorem Proving[7]

- (academic) examples of full blown processor verification
  - using ACL2, PVS, Isabelle for some processors
  - incorporation of SMT solvers is ongoing research
- special purpose applications
  - mostly for ALU but also more recently for instruction decoding
  - theorem prover connects lemmas which are proven with decision procedures
  - large theorems could involve reals (IEEE floating point standard conformance)
- specialized activity without many users

Combinational Synthesis[8]

- generic problem: $\exists p[\forall i[g(p,i) = s(i)]]$ inputs i, parameter p
- for instance fitting / merging logic to a small part (rectification) ...
- ... or synthesizing clock gating control
- ... or functional substitution (can actually be done with interpolants)
- ... or fixing circuits automatically

Sequential Synthesis[9]

- game theory: the spec s is a temporal property
- not only in theory much more complex, needs application specific restrictions

Summary

- decision procedures applied in HW vary on the amount of “formal”
  - constrained random simulation[1]
  - semi-formal[2] and bounded model checking[3]
  - equivalence checking[4] and model checking[5]
  - inductive techniques[6] and theorem proving[7]
- combinational[8] and sequential[9] synthesis interesting topics
- if your problem is hard and does not scale $\Rightarrow$ change the problem

Definition

A literal l in a clause C of a CNF F blocks C w.r.t. F if for every clause $C' \in F$ with $\bar{l} \in C'$, the resolvent $(C \setminus \{l\}) \cup (C' \setminus \{\bar{l}\})$ obtained from resolving C and C' on l is a tautology.

Definition [Blocked Clause] A clause is blocked if has a literal that blocks it.

Definition [Blocked Literal] A literal is blocked if it blocks a clause.

Example $(a \vee b) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee c)$

Only first clause is not blocked.

Second clause contains two blocked literals: a and $\bar{c}$.

Literal c in the last clause is blocked.

After removing either $(a \vee \bar{b} \vee \bar{c})$ or $(\bar{a} \vee c)$, the clause $(a \vee b)$ becomes blocked.

All clauses can be removed.

[JärvisaloBiereHeule-TACAS10]

COI Cone-of-Influence reduction

MIR Monontone-Input-Reduction

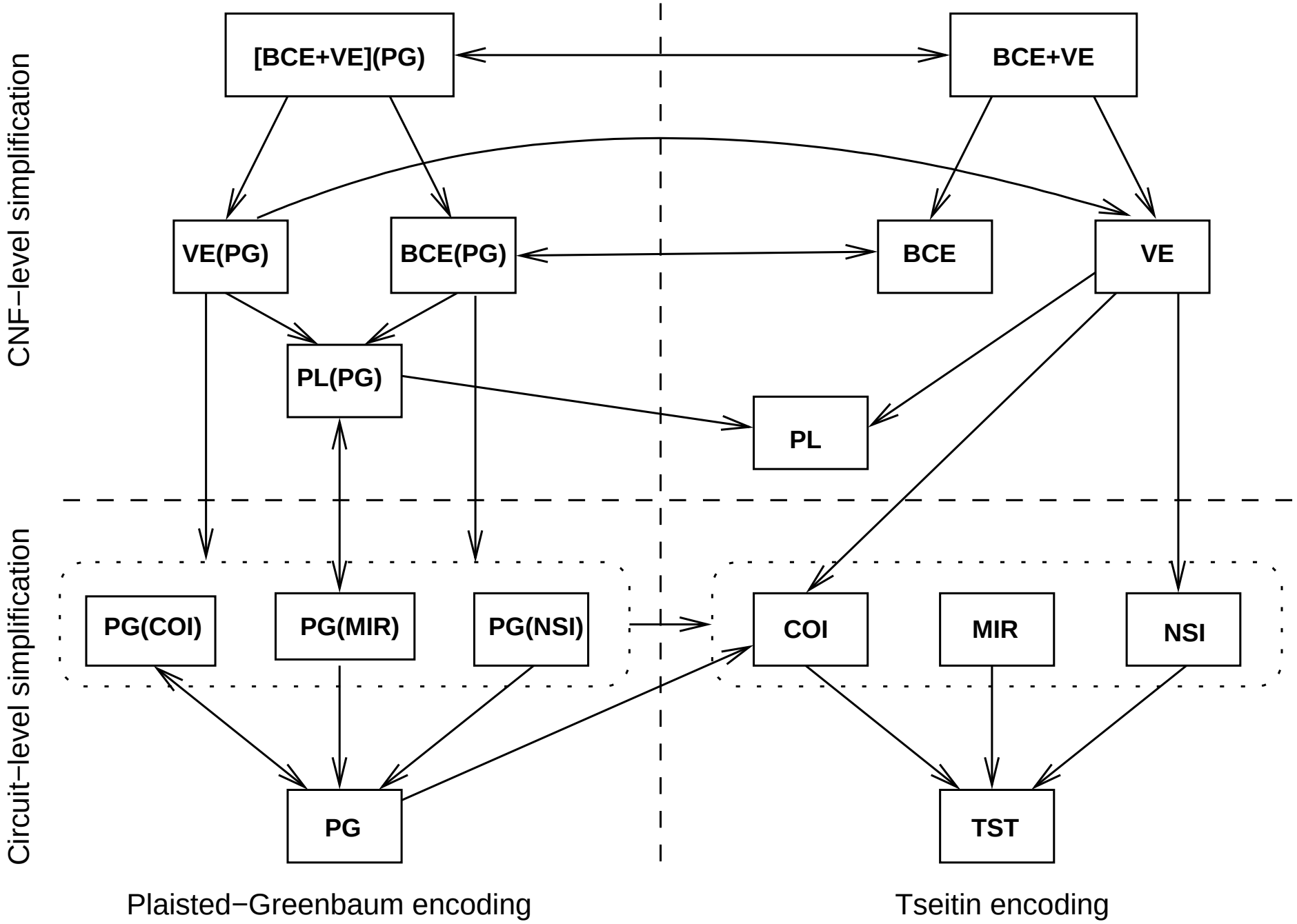
NSI Non-Shared Inputs reduction

PG Plaisted-Greenbaum polarity based encoding

TST standard Tseitin encoding

VE Variable-Elimination as in DP / Quantor / SATeLite

BCE Blocked-Clause-Elimination


CNF−level simplification
Circuit−level simplification
[BCE+VE](PG)
BCE+VE
VE(PG)
BCE(PG)
BCE
VE
PL(PG)
PL
PG(COI)
PG(MIR)
PG(NSI)
COI
MIR
NSI
PG
TST
Plaisted−Greenbaum encoding
Tseitin encoding

	encoding			b			be			beb			bebe			e		
	T	V	C	T	V	C	T	V	C	T	V	C	T	V	C	T	V	C
SU	0	46	256	2303	29	178	1042	11	145	1188	11	145	569	11	144	2064	11	153
AT	12	9	27	116	7	18	1735	1	8	1835	1	6	34	1	6	244	1	9
AP	10	9	20	94	7	18	1900	1	6	36	1	6	34	1	6	1912	1	6
AM	190	1	8	42	1	7	178	1	7	675	1	7	68	1	7	48	1	8
AN	9	3	10	50	3	10	1855	1	6	36	1	6	34	1	6	1859	1	6
HT	147	121	347	1648	117	277	2641	18	118	567	18	118	594	18	116	3240	23	140
HP	130	121	286	1398	117	277	2630	18	118	567	18	118	595	18	116	2835	19	119
HM	6961	16	91	473	16	84	621	12	78	374	12	77	403	12	76	553	15	90
HN	134	34	124	573	34	122	1185	17	102	504	17	101	525	17	100	1246	17	103
BT	577	442	1253	5799	420	1119	7023	57	321	1410	56	310	1505	**52**	**294**	8076	64	363
BP	542	442	1153	5461	420	1119	7041	57	321	1413	56	310	1506	52	294	7642	57	322
BM	10024	59	311	1252	58	303	1351	53	287	1135	53	286	1211	52	280	1435	55	303
BN	13148	196	643	2902	193	635	4845	108	508	2444	107	504	2250	105	500	5076	114	518

S = Sat competition
A = AIG competition
H = HW model checking competition
B = bit-vector SMT competition

T = plain Tseitin encoding
P = Plaisted Greenbaum
M = MiniCirc encoding
N = NiceDAGs

- equivalence reasoning
  - if $l = k$ is derived, actually clauses $(\bar{l} \vee k)(l \vee \bar{k})$, then replace l by k
  - in practice use Tarjan's union-find algorithm, where
  - each literal has a pointer to the representative of its equivalence class
  - for substituted variables get their value from the representative
- blocked clause elimination
  - save the removed blocked clauses on a stack
  - traverse stack in reverse order, values of blocked literals may need to be flipped
- how to obtain a witness if both techniques are used?

No Need to Flip Values of Substituted Literals

[JärvisaloBiere-SAT10]

Theorem

Let $C = (l \vee l_1 \vee \ldots \vee l_n)$ be a clause *blocked* on l wrt. F and

further assume $F \backslash C \models l = k$ then

for any assignment σ with $\sigma(F \backslash C) = \top$

we also have $\sigma(C) = \sigma(F) = \top$

Proof Sketch

$\sigma(l) = \bot \qquad \sigma(k) = \top$

given resolution deriviation of $\bar{l} \vee k$ from clauses $F \backslash C$, w.l.o.g. tree like

definition: l forcing for a clause $l \vee k_1 \vee \ldots \vee k_m$ iff $\sigma(k_1) = \ldots = \sigma(k_m) = \bot$ (and $\sigma(l) = \top$)

show per induction: there is a path in the tree on which all clauses are l forcing

leaf on path is an input clause $B \in F \backslash C$, with $l \in B$ but also $\bar{k}_j \in B$ for one j since C is blocked

since B is l forcing $\sigma(\bar{k}_j) = \bot$ and thus $\sigma(k_j) = \sigma(C) = \top$ q.e.d.