WAPS: Weighted and Projected Sampling

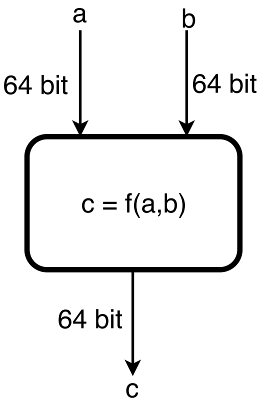
Rahul Gupta[1], Shubham Sharma[1],
Subhajit Roy[1], and Kuldeep S. Meel[2]

[1]Indian Institute of Technology Kanpur, India
[2]National University of Singapore, Singapore

TACAS 2019

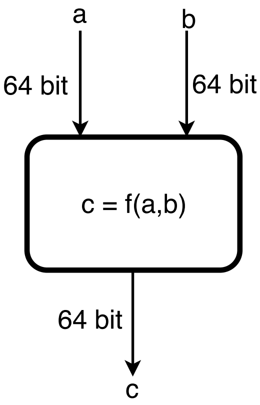
The tool is available online https://github.com/meelgroup/WAPS

Hardware Validation



- Design is simulated with test vector (values of a and b)
- Results from simulation compared to intended results
- Challenge: How do we generate test vectors?
  - 2^{128} combinations for a toy circuit
- Use constraints to represent interesting verification scenarios

Constrained Simulation



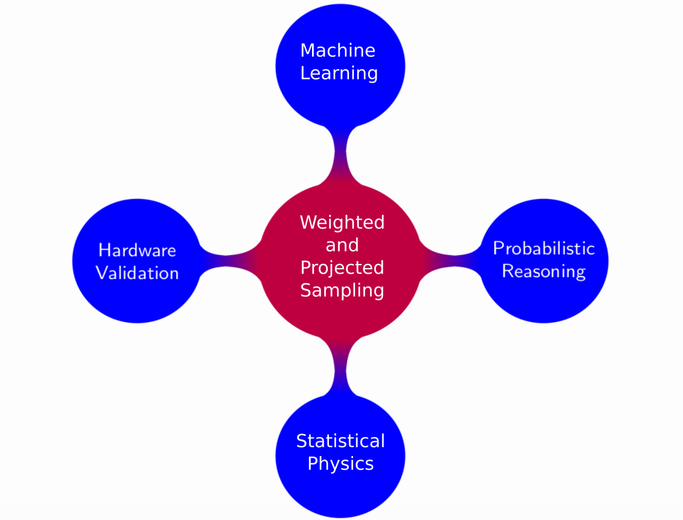
- **Constraints**:
  - Designers:
    - $a +_{64} 11 *_{32} b = 12$
    - $a <_{64} (b >> 4)$
  - Past Experience:
    - $40 <_{64} 34 +_{64} a <_{64} 5050$
    - $120 <_{64} b <_{64} 230$
  - Users:
    - $232 *_{32} a +_{64} b! = 1100$
    - $1020 <_{64} (b /_{64} 2) +_{64} a <_{64} 2200$
- **Weight distribution**: $W(\cdot)$ on solutions of constraints to achieve coverage goals
- **Test vectors**: sampled solutions of constraints conditioned on $W(\cdot)$

Weighted and Projected Sampling

- Given:
  - CNF formula F over a set of variables X
  - Set of projecting variables $P \subseteq X$
  - Weight function $W(\cdot)$ over literals
    - The weight of assignment is the product of weights of its literals
- Weighted and Projected Sampler:

$$\forall y \in R_{F\downarrow P}, \Pr[y \text{ is output }] = \frac{W(y)}{W(R_{F\downarrow P})}$$

Applications of Weighted and Projected Sampling



Wish List

- Theoretical guarantee that samples projected over a sampling set satisfy a certain weight distribution.
- Scalability
- Anytime Generation:
  - Testing typically involves multiple iterations of sampling and verification.

Prior Work

Strong guarantees but poor scalability

- Hashing based techniques - WeightGen (Chakraborty et al. 2014)
- BDD based techniques (Yuan et al. 1999, Yuan et al. 2004, Kukula and Shiple 2000)

Weak guarantees but impressive scalability

- MCMC, Metropolis-Hasting (Jerrum et al. 1996, Mardras et al. '02)

How to bridge this gap between theory and practice?

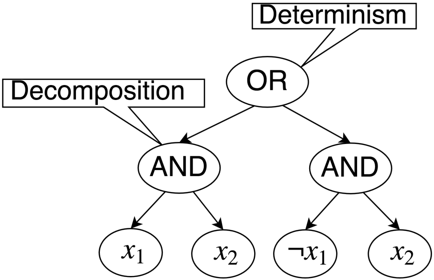
Close Cousins: Counting and Sampling

- Approximate counting and almost-uniform sampling are inter-reducible (Jerrum et al. 1986)
- Can we design exact samplers that require only one call to exact counter?
- All exact counting techniques generate d-DNNF (Huang and Darwiche 2008)
- Uniform sampling can be performed by making constant number of passes over compiled d-DNNF (Sharma et al. 2018)
- Can knowledge compilation technique be used to perform weighted and projected sampling?

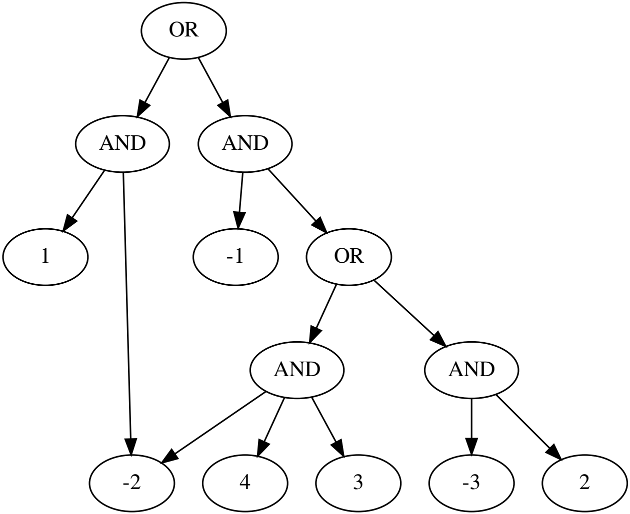
d-DNNF

The Deterministic Decomposable Negation Normal Form (**d-DNNF**) is a strict subset of **NNF** that further imposes that the representation is:

- **Deterministic**: the operands of $\vee$ in all well-formed Boolean formula in the NNF are mutually inconsistent.
- **Decomposable**: the operands of $\wedge$ in all well-formed Boolean formula in the NNF are expressed in a mutually disjoint set of variables.



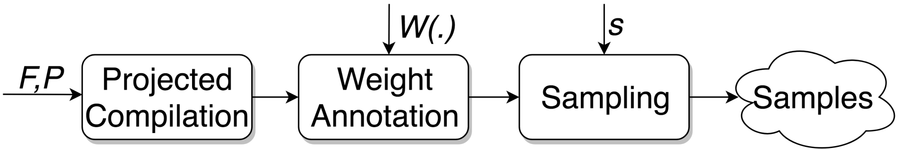
An example of d-DNNF



The power of d-DNNF

- Counting is linear time in the size of d-DNNF.
  - **OR Node**: Sum the solutions of children
  - **AND Node**: Multiply the solutions of children

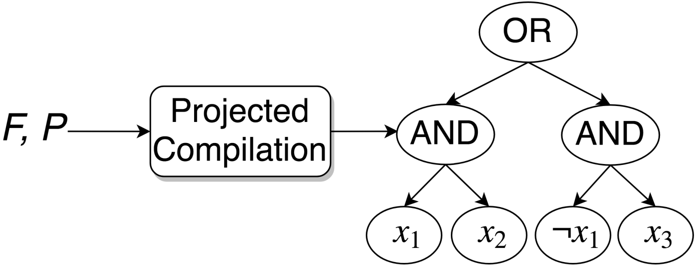
WAPS



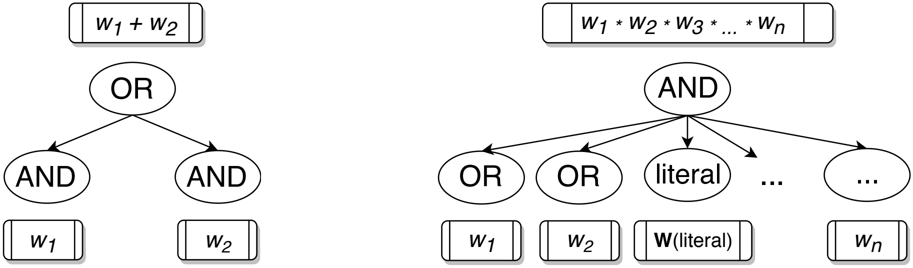
F,P
Projected Compilation
$W(.)$
Weight Annotation
s
Sampling
Samples

Projected Compilation

$$F = (x_6 \lor x_5 \lor \neg x_1 \lor x_3) \land (x_3 \lor x_6 \lor \neg x_5 \lor \neg x_1) \land (\neg x_2 \lor x_4 \lor \neg x_1) \land$$
$$(x_1 \lor x_2) \land (x_3 \lor \neg x_6 \lor \neg x_1) \land (\neg x_3 \lor \neg x_5 \lor x_6)$$
$$P = \{x_1, x_2, x_3\}$$

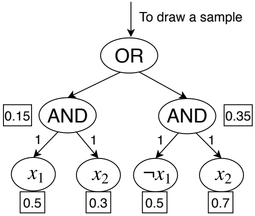


Weight Annotation

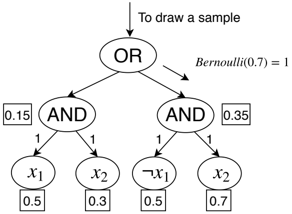


Getting one sample

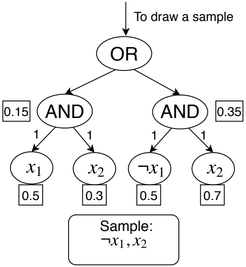
- Since, solutions are disjoint at different children of an OR node, to draw a sample, we can simply perform a Bernoulli experiment with probabilities proportional to the weights of children at OR node.
- At AND node, we simply stitch the samples from its children.



DAG with annotated weights



Deciding the path by bernoulli trial



Stitching sample from all children of AND node

Getting multiple samples

To draw multiple samples

- Use Binomial Distribution at the OR nodes to find number of samples to be drawn from each child
- Randomly shuffle the samples before stitching them at the AND node.

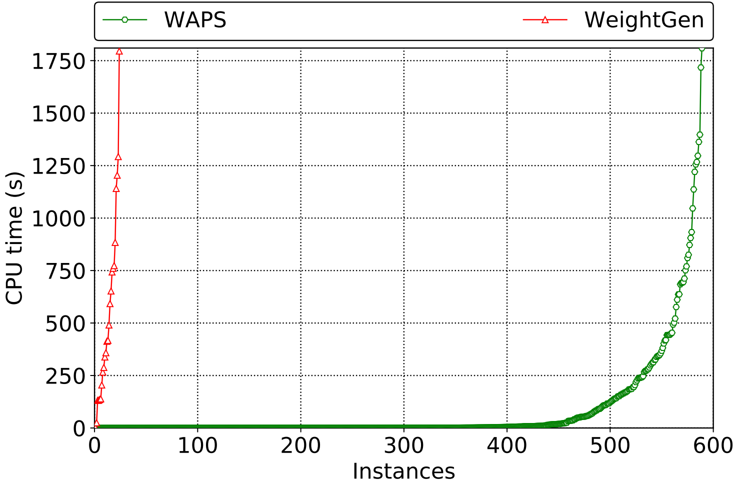
Theoretical Guarantees

$$\forall y \in R_{F\downarrow P}, \Pr\left[y \text{ is output }\right] = \frac{W(y)}{W(R_{F\downarrow P})}$$

Experimental Evaluation

- 773 benchmarks arising from ISCAS89 circuits, DQMR networks, bit-blasted versions of SMT-LIB (SMT)
- Compared with WeightGen: state-of-the-art weighted and projected sampler
- Objectives:
  - Runtime performance
  - Anytime Generation
  - Distribution comparison
  - Effect of Weight Distribution

Runtime Performance-I



- WeightGen solved 24 benchmarks
- WAPS solved 588 benchmarks

Runtime Performance-II

Benchmark	Vars	Clauses	$\|P\|$	WeightGen	WAPS			WeightGen / WAPS
					Compile	A+S	Total	
s526_15_7	452	1303	22	652.48	91.66	31.15	122.81	5.31
s526a_3_2	366	944	24	490.34	15.37	1.96	17.33	28.29
LoginService	11511	41411	36	1203.93	15.02	0.75	15.77	76.34
blockmap_5_2	1738	3452	1738	1140.87	0.04	5.30	5.34	213.65
s526_3_2	365	943	24	417.24	0.06	0.67	0.73	571.56
or-50-5-9-UC-40	100	250	100	743.1	0.01	0.41	0.42	1769.29
or-100-5-4-UC	200	500	200	1795.52	0.01	0.74	0.75	2426.38
or-50-5-10-UC	100	250	100	1292.67	0.01	0.36	0.37	3590.75
blasted_case35	400	1414	46	TO	0.57	1.46	2.03	-
or-100-20-4-UC	200	500	200	TO	0.19	2.48	2.67	-

Table: Run time (in seconds) for 1000 samples

- WAPS outperformed WeightGen with a geometric speedup of 296$\times$

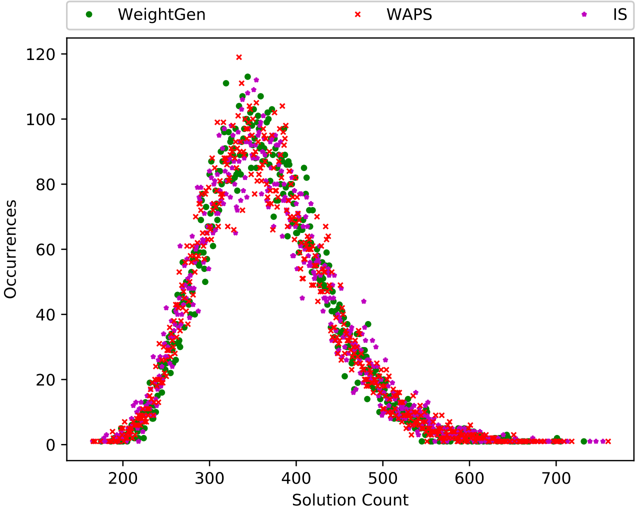
Anytime Generation

Benchmark	Vars	Clauses	$\|P\|$	WAPS		Speedup
				1000	10,000	
case110	287	1263	287	1.14	9.28	1.26
or-70-10-10-UC-20	140	350	140	2.75	9.02	6.56
s526_7_4	383	1019	24	60.38	143.16	13.20
or-60-5-2-UC-10	120	300	120	12.10	20.35	16.50
s35932_15_7	17918	44709	1763	69.01	106.65	20.73
case121	291	975	48	35.85	51.41	20.73
s641_15_7	576	1399	54	729.38	916.83	35.01
squaring7	1628	5837	72	321.95	365.13	67.10
LoginService	11511	41411	36	15.89	18.12	64.13
ProjectService	3175	11019	55	184.51	195.25	154.61

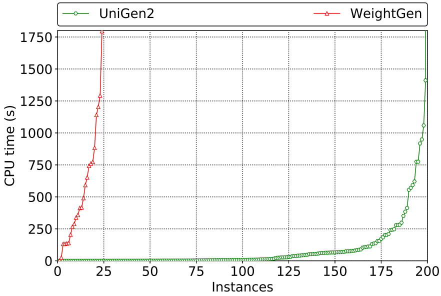
Table: Runtimes (in sec.) of WAPS for incremental sampling

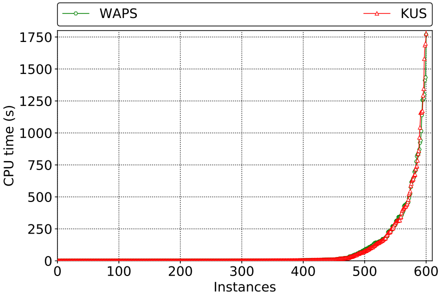
- WAPS achieves a geometric speedup of 3.69$\times$

Distribution Comparison



Effect of Weight Distribution





- The performance of hashing-based techniques is limited in its ability to handle literal-weighted sampling.
- The performance of knowledge compilation based sampling technique is oblivious to the weight distribution.

Conclusion

- Weighted and projected sampling is a fundamental problem with a wide variety of applications
- Deep connection between sampling and counting offers opportunities for design of sampling algorithms
- The trace of exact counting algorithms generates compiled knowledge forms. (d-DNNF)
- Knowledge representations can be used to generate samples
- Outperforms existing state of the art techniques
- Where do we go from here? Knowledge compilation for sampling?

Conclusion

- Exact counting is #P-complete
- Exact counting can be done in linear time in the size of d-DNNF
- Relaxations of d-DNNF that allow sampling in polynomial time but not exact counting?

Conclusion

- The tool is available online https://github.com/meelgroup/WAPS