Experiments from “Checking δ-Satisfiability of Reals with Integrals”

Cody Rivera, Bishnu Bhusal, Rohit Chadha, A. Prasad Sistla, and Mahesh Viswanathan

1 Introduction

The benchmarks we evaluate with $\int$dReal include formulas involving single, double, and triple integrals, as well as some $\exists - \forall$ queries. These come from domains including algorithmic fairness, differential privacy, and some synthetic examples. We present our benchmarks by classifying the queries used.

2 Algorithmic Fairness Experiments

2.1 Income Fairness by Gender

One example comes from algorithmic fairness: checking that an algorithm for determining the income of an employee is not biased by gender. This example is derived from the work on FairSquare by Albarghouthi et. al. [1], and more specifically their decision tree hiring procedure DT_4 on an independent population model. The formula used here is

$$\phi(\epsilon, \mu, \sigma) ::= \frac{2.02389 \left(3307 \text{ erf}\left(\frac{14147-2\mu}{2\sqrt{2}\sqrt{\sigma}}\right) + 16693\right)}{6693 \text{ erf}\left(\frac{12625081-2000\mu}{400\sqrt{50\sigma+2253955380}}\right) + 13307} > 1 - \epsilon,$$

where

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt.$$

Two queries in the benchmark suite are based on this formula. Here, we try to prove that a negated query is UNSAT, showing fairness of the algorithm.

1. (high_inc_gd_00): $\epsilon = 0.15$, $\mu = 568.4105$, $\sigma = 24248365.5428$. $\neg\phi(\epsilon, \mu, \sigma)$ given to solver.
2. (high_inc_gd_01): $\epsilon = 0.15$, The following formula is given to the solver:

$$\exists \mu \in [548.4105, 588.4105], \sigma \in [548.4105, 588.4105], \neg\phi(\epsilon, \mu, \sigma)$$

We also test a version of this formula that reflects an unfair version of the algorithm.

$$\phi ::= \frac{2.02389 \left(3307 \text{ erf}\left(\frac{14147-2\mu}{2\sqrt{2}\sqrt{\sigma}}\right) + 16693\right)}{6693 \text{ erf}\left(\frac{12625081-2000\mu}{400\sqrt{50\sigma+2253955380}}\right) + 53228} > 1 - \epsilon.$$

Two queries in the benchmark suite are based on this formula. Again, we try to prove that a negated query is UNSAT, trying but failing to show fairness of the algorithm.

1. (high_inc_gd_unfair_00): $\epsilon = 0.15$, $\mu = 568.4105$, $\sigma = 24248365.5428$. $\neg\phi(\epsilon, \mu, \sigma)$ given to solver.
2. (high_inc_gd_unfair_01): $\epsilon = 0.15$, The following formula is given to the solver:

$$\exists \mu \in [548.4105, 588.4105], \sigma \in [548.4105, 588.4105], \neg\phi(\epsilon, \mu, \sigma)$$

2.2 Fair Hiring with Respect to College Rank and Ethnicity

This example also comes from algorithmic fairness and is also derived from the work on FairSquare by Albarghouthi et. al. [1]. This time, the example is based on the illustrative example from the paper: how to make a hiring algorithm which depends on years of experience and college rank fair according to ethnicity, when college rank is influenced by ethnicity. The program that is used is the one found in Figure 1 of [1]. We evaluate the fair version of the algorithm with the statement "`expRank` $\leftarrow$ 5 $\times$ `yExp` + `colRank`", and the unfair version of the algorithm which puts too much weight on college rank, with the statement "`expRank` $\leftarrow$ `yExp` + `colRank`". The formula used here is

$$\phi ::= \mathrm{LHS}(\epsilon, \mu, \sigma) \geq (1 - \epsilon)\ \mathrm{RHS}(\epsilon, \mu, \sigma),$$

where

$$\mathrm{LHS}(\epsilon, \mu, \sigma) ::= \int_{-4\sigma}^{t-5} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2} dx + \int_{t-5}^{4\sigma} \int_{\frac{x}{5}}^{4} \sigma_x \frac{1}{5\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{z-10}{5})^2} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2} dz dx,$$

$$\mathrm{RHS}(\epsilon, \mu, \sigma) ::= \int_{-4\sigma}^{t} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2} dx + \int_{t}^{4\sigma} \int_{\frac{x-5}{5}}^{4\sigma} \frac{1}{5\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{z-10}{5})^2} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2} dz dx + \frac{14}{100000}.$$

Two queries in the benchmark suite are based on this formula. Here, we try to prove that a negated query is UNSAT, showing fairness of the algorithm.

1. (eth_colrank_fair_00): $\epsilon = 0.1$, $\mu = 25$, $\sigma = 10$. $\neg\phi$ given to solver.
2. (eth_colrank_fair_01): $\epsilon = 0.1$, $\mu = [20, 30]$, $\sigma = [10, 20]$ (except at integral endpoints, where we take $\sigma = 10$). $\neg\phi$ given to solver.

We also test a version of this formula that reflects an unfair version of this algorithm. Here, we set

$$\mathrm{LHS}(\epsilon, \mu, \sigma) ::= \int_{-4\sigma}^{t-5} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2} dx + \int_{t-5}^{4\sigma} \int_{x}^{4} \sigma_x \frac{1}{5\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{z-10}{5})^2} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2} dz dx + \frac{14}{100000},$$

$$\mathrm{RHS}(\epsilon, \mu, \sigma) ::= \int_{-4\sigma}^{t} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2} dx + \int_{t}^{4\sigma} \int_{x-5}^{4\sigma} \frac{1}{5\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{z-10}{5})^2} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2} dz dx,$$

and we solve two queries in the benchmark set with those formulas. Here, we try to prove that the original query is UNSAT, showing that the algorithm is unfair. Note that for eth_colrank_unfair_01, that we show that the algorithm is unfair for all (rather than some) values of μ and σ in the given interval.

1. (eth_colrank_unfair_00): $\epsilon = 0.1$, $\mu = 25$, $\sigma = 10$. ϕ given to solver.
2. (eth_colrank_unfair_01): $\epsilon = 0.1$, $\mu = [20, 30]$, $\sigma = [10, 20]$ (except at integral endpoints, where we take $\sigma = 10$). ϕ given to solver.

3 Differential Privacy

3.1 Synthesis of Parameters for Accuracy

Here is an example of an $\exists - \forall$ query that appears in our benchmarks. The query will synthesize a value of a parameter a in a one-element version of the Sparse Vector Technique (SVT) [2, 4] algorithm such that the algorithm is accurate for many values of ϵ. The accuracy formula used here is

$$\phi(a, \epsilon, \delta, k) ::= \frac{a(1-a)\epsilon^2}{2\pi} \int_{-k}^{k} \int_{x}^{k} e^{-\frac{a^2\epsilon^2x^2}{2}} e^{-\frac{(1-a)^2\epsilon^2(y-1)^2}{2}} dydx - 0.5 - \delta > 0,$$

where $\delta = 0.001$ and $k = 20$, representing the δ to which we set the $\int$dReal tool. The query corresponds to gauss_forall_00, and is structured as follows:

$$\exists a \in [0.25, 0.75], \forall \epsilon \in [0.5, 0.9], \phi(a, \epsilon, \delta, k).$$

The query is run in $\int$dReal, and outputs δ-SAT with an interval enclosing the optimal value of a, 0.5. We can verify that this is correct by constructing a verification query $\exists \epsilon \in [0.5, 0.9], \neg\phi$, with $a = 0.5$, and using $\int$dReal to check whether it is UNSAT.

3.2 Accuracy of the Laplace Mechanism

The next examples concern the accuracy of the Laplace mechanism in differential privacy [3]. Consider a Laplace mechanism over the identity function f over 2 inputs: $f(u, v) = (u, v)$, with sensitivity $\Delta f = 1$. The resulting accuracy query will be:

$$\phi ::= \frac{\epsilon^2}{4} \int_{-\delta'}^{\delta'} e^{-\epsilon|x|} dx \int_{-\delta'}^{\delta'} e^{-\epsilon|y|} dy > 1 - \delta,$$

where $\delta' = \frac{1}{\epsilon}\ln(\frac{2}{\delta})$. We have the following two queries from this example, for which we try to prove UNSAT:

1. (lap_acc_00_alt1): $\epsilon \in [0.1, 1]$ and $\delta \in (0.001, 1]$, $\neg\phi$ given to solver.
2. (lap_acc_00_alt2): $\epsilon \in [0.1, 1]$ and $\delta \in (0.001, 1]$, $\neg\phi$ given to solver. The two integrals, which are multiplied together in ϕ, are instead squared.

We can also generalize these queries to the identity function on k-tuples where $k > 2$. The generalized form of the formula is

$$\phi ::= \frac{\epsilon^k}{2^k} \left(\int_{-\delta'(k)}^{\delta'(k)} e^{-\epsilon|x|} dx \right)^k > 1 - \delta,$$

where $\delta'(k) = \frac{1}{\epsilon}\ln(\frac{k}{\delta})$. We obtain four more queries from variants of this formula, which we try to prove UNSAT.

1. (lap_acc_01): $k = 3$, $\epsilon \in [0.1, 1]$ and $\delta \in (0.001, 1]$, $\neg\phi$ given to solver.
2. (lap_acc_01_alt1): $k = 3$, $\epsilon \in [0.1, 1]$ and $\delta \in (0.001, 1]$, $\neg\phi$ given to solver. Instead of exponentiating, the integral is written as the product of k integrals.
3. (lap_acc_02): $k = 4$, $\epsilon \in [0.1, 1]$ and $\delta \in (0.001, 1]$, $\neg\phi$ given to solver.
4. (lap_acc_02_alt1): $k = 4$, $\epsilon \in [0.1, 1]$ and $\delta \in (0.001, 1]$, $\neg\phi$ given to solver. Instead of exponentiating, the integral is written as the product of k integrals.

3.3 Privacy of the Laplace Mechanism

The next examples concern the privacy of the Laplace mechanism in differential privacy. The formula pertaining to the first (private) mechanism can be seen here:

$$\phi_1 ::= \int_a^b e^{-\epsilon|x|}dx \le e^{\epsilon} \int_a^b e^{-\epsilon|x+1|}dx.$$

Throughout the queries, $a = 1$ and $b = 2$. We try to prove the privacy of the mechanism for all ϵ in a given range by negating ϕ_1 and checking for UNSAT, as well as try to disprove the privacy for all ϵ in the same range by checking ϕ_1 for UNSAT. The queries are below:

1. (lap_mech_00): $\epsilon \in [0.1, 1]$. $\neg\phi_1$ given to solver.
2. (lap_mech_00_not_pri): $\epsilon \in [0.1, 1]$. ϕ_1 given to solver.

The formula pertaining to the second (non-private) mechanism can be seen here:

$$\phi_2 ::= \int_a^b e^{-\epsilon|x|}dx \le e^{\frac{\epsilon}{2}} \int_a^b e^{-\epsilon|x+1|}dx.$$

Throughout the queries, $a = 1$ and $b = 2$. We try to prove the privacy of the mechanism for all ϵ in a given range by negating ϕ_2 and checking for UNSAT, as well as try to disprove the privacy for all ϵ in the same range by checking ϕ_2 for UNSAT. The queries are below:

1. (lap_mech_sat_01): $\epsilon \in [0.1, 1]$. $\neg\phi_2$ given to solver.
2. (lap_mech_sat_01_not_pri): $\epsilon \in [0.1, 1]$. ϕ_2 given to solver.

3.4 Privacy of the Gaussian Mechanism

The next examples concern the privacy of the Gaussian mechanism in differential privacy. Here specifically, we take the mechanism which maps (u, v) to $(u + \mathcal{N}(0, \sigma^2), v + \mathcal{N}(0, \sigma^2))$, where $\mathcal{N}$ is the normal distribution. We consider two points (u_1, v_1) and (u_2, v_2) neighbors by Euclidean distance, i.e., $0 \le \sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2} \le 1$. The formula pertaining to the private mechanism can be seen below:

$$\phi ::= P_{\text{gauss}}(u_1, v_1, a, b, c, d) \le e^{\epsilon} P_{\text{gauss}}(u_2, v_2, a, b, c, d) + \delta,$$

where

$$P_{\text{gauss}}(u, v, a, b, c, d) = \frac{1}{2\pi\sigma^2} \int_a^b \int_c^d e^{-\frac{(x-u)^2}{2\sigma^2}} e^{-\frac{(y-v)^2}{2\sigma^2}} dxdy,$$

$\delta = \frac{1}{8}$, $\sigma = \frac{2.2}{\epsilon}$ and $a = b = c = d = 10$. The formula here comprises the following query:

1. (gauss_mech_00): $\epsilon \in (0.1, 1)$, $\neg\phi$ sent to solver.

There is additionally a version of the query with the double integral converted to a product of integrals, where P_{gauss} is defined as follows:

$$P_{\text{gauss}}(u, v, a, b, c, d) = \frac{1}{2\pi\sigma^2} \left(\int_a^b e^{-\frac{(x-u)^2}{2\sigma^2}} dx \right) \left(\int_c^d e^{-\frac{(y-v)^2}{2\sigma^2}} dy \right).$$

The query is as follows:

1. (gauss_mech_00_alt1): $\epsilon \in (0.1, 1)$, $\neg\phi$ sent to solver.

3.5 Privacy of the SVT algorithm with the Gaussian Mechanism

We once again turn to the Sparse Vector Technique (SVT), again using the Gaussian distribution, but with a vector size of $N = 2$ this time. The formula we check is:

$$\phi ::= P^{\text{SVT}}_{\mathcal{N}(u_1,v_1),LHS} \leq e^{\epsilon} P^{\text{SVT}}_{\mathcal{N}(u_2,v_2),RHS} + \delta,$$

where

$$P^{\text{SVT}}_{\mathcal{N}(u,v),LHS} = \frac{1}{2\pi\sigma^3\sqrt{2\pi}} \int_{-k}^{+k} \int_{v-k}^{z} \int_{u-k}^{z} e^{-\frac{(x-u)^2}{2\sigma^2}} e^{-\frac{(y-v)^2}{2\sigma^2}} e^{-\frac{z^2}{2\sigma^2}} dxdydz + \frac{3}{100}$$
$$P^{\text{SVT}}_{\mathcal{N}(u,v),RHS} = \frac{1}{2\pi\sigma^3\sqrt{2\pi}} \int_{-k}^{k} \int_{v-k}^{z} \int_{u-k}^{z} e^{-\frac{(x-u)^2}{2\sigma^2}} e^{-\frac{(y-v)^2}{2\sigma^2}} e^{-\frac{z^2}{2\sigma^2}} dxdydz$$

There are two different queries run here, with eight different variations in total. Note that points (u_1, v_1) and (u_2, v_2) here are adjacent in that they have a Manhattan distance, or $|u_2 - u_1| + |v_2 - v_1|$, of 1 from each other.

1. (svt_gauss_00): $\delta = \frac{1}{8}$, $(u_1, v_1) = (1, 0)$, $(u_2, v_2) = (0, 0)$, $k = \{4, 8\}$, $\epsilon \in \{[0.1, 0.5], [0.1, 0.9]\}$. $\neg\phi$ sent to solver.
2. (svt_gauss_sat_00): $\delta = 0$, $(u_1, v_1) = (1, 0)$, $(u_2, v_2) = (0, 0)$, $k = \{4, 8\}$, $\epsilon \in \{[0.1, 0.5], [0.1, 0.9]\}$. $\neg\phi$ sent to solver.

4 Synthetic Benchmarks

For the following benchmarks, we define the following list of functions:

1. $f_1(x, y, \epsilon) = \sin(x) + \epsilon y$
2. $f_2(x, y, \epsilon) = \sin(x + \epsilon y)$
3. $f_3(x, y, \epsilon) = \sin(x) + \sin(\epsilon y)$
4. $f_4(x, y, \epsilon) = \sin(x)\sin(\epsilon y)$
5. $f_5(x, y, \epsilon) = \sin(x)e^{\epsilon y}$
6. $f_6(x, y, \epsilon) = \sin(x)^2 + e^{3\epsilon y}$.

4.1 Volume Comparison

We define regions as follows:

$$R_i^1(\epsilon) = \int_0^1 \int_0^y f_i(x, y, \epsilon) dx dy,$$
$$R_i^2(\epsilon) = \int_0^1 \int_0^{y^2} f_i(x, y, \epsilon) dx dy.$$

The two types of queries we have are as follows:

$$\phi_1 ::= R_i^1(\epsilon) > R_i^2(\epsilon) + \frac{1}{100}$$
$$\phi_2 ::= R_i^1(\epsilon) < R_i^2(\epsilon) + \frac{1}{100}$$

We instantiate into twelve queries as follows:

1. (vol_cmp_00): $\epsilon \in [0.1, 0.2]$, $i = 1$, $\neg\phi_1$ sent to solver.
2. (vol_cmp_01): $\epsilon \in [0.1, 0.2]$, $i = 1$, $\neg\phi_2$ sent to solver.
3. (vol_cmp_02): $\epsilon \in [0.1, 0.2]$, $i = 2$, $\neg\phi_1$ sent to solver.
4. (vol_cmp_03): $\epsilon \in [0.1, 0.2]$, $i = 2$, $\neg\phi_2$ sent to solver.
5. (vol_cmp_04): $\epsilon \in [0.1, 0.2]$, $i = 3$, $\neg\phi_1$ sent to solver.
6. (vol_cmp_05): $\epsilon \in [0.1, 0.2]$, $i = 3$, $\neg\phi_2$ sent to solver.
7. (vol_cmp_06): $\epsilon \in [0.1, 0.2]$, $i = 4$, $\neg\phi_1$ sent to solver.
8. (vol_cmp_07): $\epsilon \in [0.1, 0.2]$, $i = 4$, $\neg\phi_2$ sent to solver.
9. (vol_cmp_08): $\epsilon \in [0.1, 0.2]$, $i = 5$, $\neg\phi_1$ sent to solver.
10. (vol_cmp_09): $\epsilon \in [0.1, 0.2]$, $i = 5$, $\neg\phi_2$ sent to solver.
11. (vol_cmp_10): $\epsilon \in [0.1, 0.2]$, $i = 6$, $\neg\phi_1$ sent to solver.
12. (vol_cmp_11): $\epsilon \in [0.1, 0.2]$, $i = 6$, $\neg\phi_2$ sent to solver.

4.2 Quantifier Alternation

We propose the following examples of quantifier alternation. To do this, we define the following functions: $g_i(x, y, a, \epsilon) = f_i(\frac{\pi x}{2}, a\epsilon y, \epsilon)$, and

$$S_i^1(a, \epsilon) = \int_0^1 \int_0^y g_i(x, y, a, \epsilon) dx dy,$$
$$S_i^3(a, \epsilon) = \int_0^1 \int_y^1 g_i(x, y, a, \epsilon) dx dy.$$

The query we have is:

$$\phi ::= \exists a \in [1, 2], \forall \epsilon \in [0.1, 0.9], S_i^1(a, \epsilon) - S_i^3(a, \epsilon) - \frac{1}{100} - \delta \geq 0.$$

Note that throughout, $\delta = 0.001$. We instantiate into six queries:

1. (vol_cmp_00_forall): $i = 1$, ϕ sent to solver.
2. (vol_cmp_02_forall): $i = 2$, ϕ sent to solver.
3. (vol_cmp_04_forall): $i = 3$, ϕ sent to solver.
4. (vol_cmp_06_forall): $i = 4$, ϕ sent to solver.
5. (vol_cmp_08_forall): $i = 5$, ϕ sent to solver.
6. (vol_cmp_10_forall): $i = 6$, ϕ sent to solver.

References

1. Albarghouthi, A., D'Antoni, L., Drews, S., Nori, A.V.: Fairsquare: Probabilistic verification of program fairness. Proc. ACM Program. Lang. **1**(OOPSLA) (oct 2017). https://doi.org/10.1145/3133904, https://doi.org/10.1145/3133904
2. Dwork, C., Naor, M., Reingold, O., Rothblum, G.N., Vadhan, S.: On the complexity of differentially private data release: efficient algorithms and hardness results. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. p. 381–390. STOC '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1536414.1536467, https://doi.org/10.1145/1536414.1536467
3. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. **9**(3–4), 211–407 (aug 2014). https://doi.org/10.1561/0400000042, https://doi.org/10.1561/0400000042
4. Lyu, M., Su, D., Li, N.: Understanding the sparse vector technique for differential privacy. Proc. VLDB Endow. **10**(6), 637–648 (feb 2017). https://doi.org/10.14778/3055330.3055331, https://doi.org/10.14778/3055330.3055331