Available online at [www.sciencedirect.com](http://www.sciencedirect.com/)

[Electronic Notes in Theoretical Computer Science 325 (2016) 277–298](http://dx.doi.org/10.1016/j.entcs.2016.09.043)

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Approximate Relational Hoare Logic for Continuous Random Samplings

Tetsuya Sato[1](#_bookmark0)

*Research Institute for Mathematical Sciences, Kyoto University, Kyoto, 606-8502, Japan*

**Abstract**

Approximate relational Hoare logic (apRHL) is a logic for formal verification of the differential privacy of databases written in the programming language pWHILE. Strictly speaking, however, this logic deals only with discrete random samplings. In this paper, we define the graded relational lifting of the subprobabilistic variant of Giry monad, which described differential privacy. We extend the logic apRHL with this graded lifting to deal with continuous random samplings. We give a generic method to give proof rules of apRHL for continuous random samplings.

*Keywords:* Differential privacy, Denotational semantics, Giry monad, Graded monad, Relational lifting

# Introduction

Differential privacy is a *deﬁnition* of privacy of *randomised* databases proposed by Dwork, McSherry, Nissim and Smith [[7](#_bookmark25)]. A randomised database satisfies *ε*- differential privacy (written *ε*-differentially private) if for any two adjacent data, the difference of their output probability distributions is bounded by the privacy strength *ε*. Differential privacy guarantees high secrecy against database attacks regardless of the attackers’ background knowledge, and it has the composition laws, with which we can calculate the privacy strength of a composite database from the privacy strengths of its components.

*Approximate relational Hoare logic* (apRHL) [[2](#_bookmark19),[17](#_bookmark35)] is a probabilistic variant of the *relational Hoare logic* [[4](#_bookmark22)] for formal verification of the differential privacy of databases written in the programming language pWHILE. In the logic apRHL, a parametric relational lifting, which relate probability distributions, play a central role to describe differential privacy in the framework of verification. This para- metric lifting is an extension of the relational lifting [[10](#_bookmark28), Section 3] that captures

1 Email:[satoutet@kurims.kyoto-u.ac.jp](mailto:satoutet@kurims.kyoto-u.ac.jp)

<http://dx.doi.org/10.1016/j.entcs.2016.09.043>

1571-0661/© 2016 The Author(s). Published by Elsevier B.V.

This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

probabilistic bisimilarity of Markov chains [[13](#_bookmark30)] (see also [[6](#_bookmark24), lemma 4]). The concept of differential privacy is described in the category of binary relation and mappings between them, and verified by the logic apRHL.

Strictly speaking, however, apRHL deals only with random samplings of *discrete* distributions, while the algorithms in many actual studies for differential privacy are modelled with *continuous* distributions, such as, the Laplacian distributions over real line. Therefore apRHL is desired to be extended to deal with random continuous samplings.

* 1. *Contributions*

Main contributions of this paper are the following two points:

* We define the graded relational lifting of sub-Giry monad describing differential privacy for continuous random samplings.
* We extend the logic apRHL [[2](#_bookmark19),[17](#_bookmark35)] for continuous random samplings (we name

*continuous apRHL*) .

This graded relational lifting is developed without witness distributions of proba- bilistic coupling, and hence is constructed in a different way from the coupling-based parametric lifting of relations given in the studies of apRHL [[1](#_bookmark20),[2](#_bookmark19),[17](#_bookmark35)].

In the continuous apRHL, we mainly extend the proof rules for relation com- positions and the frame rule. We also develop a generic method to construct proof rules for random samplings. By importing the new rules added to apRHL+ in [[1](#_bookmark20)], we give a formal proof of the differential privacy of the *above-threshold algorithm* for real-valued queries [[8](#_bookmark26), Section 3.6].

* 1. *Preliminaries*

We denote by **Set**, *ω***CPO***⊥*, and **Meas** the categories of all sets and functions, all *ω*-complete partial orders with the least element and continuous functions be- tween them, and all measurable spaces and measurable functions respectively. The category **Meas** is complete, cocomplete, and distributive. The forgetful functor *U* : **Meas** *→* **Set** preserves limits and colimits. For each measurable space *X*, we write Σ*X* for the *σ*-algebra of *X*. For any *A ∈* Σ*X* , the *indicator function χA* : *X →* [0*,* 1] of *A* is defined by *χA*(*x*)=1 if *X ∈ A* and *χA*(*x*) = 0 otherwise.

## The Category of Relations between Measurable Spaces

We introduce the category **BRel**(**Meas**) of binary relations between *measurable spaces* as follows:

* An object is a triple (*X, Y,* Φ) consisting of measurable spaces *X* and *Y* and a relation Φ between *X* and *Y* (i.e. Φ *⊆ UX × UY* ). We remark that Φ does not necessary to be a measurable subset of the product space *X × Y* .
* An arrow (*f, g*): (*X, Y,* Φ) *→* (*Xj,Y j,* Φ*j*) is a pair of measurable functions *f* : *X →*

*Xj* and *g* : *Y → Y j* such that (*Uf × Ug*)(Φ) *⊆* Φ*j*.

When we write an object (*X, Y,* Φ) in **BRel**(**Meas**), we omit to write the underlying spaces *X* and *Y* if they are obvious from the context. We write *p* for the forgetful functor **BRel**(**Meas**) *→* **Meas**2 extracting underlying spaces: (*X, Y,* Φ) *'→* (*X, Y* ). The category **BRel**(**Meas**) is complete and cocomplete, and the forgetful func-

tor *p* preserves limits and colimits. We write *×*˙ and +˙ for operators of the binary

products and coproducts in **BRel**(**Meas**) respectively:

(*X, Y,* Φ)*×*˙ (*Z, W,* Ψ)

= (*X × Z, Y × W, {* ((*x, z*)*,* (*y, w*)) *|* (*x, y*) *∈* Φ*,* (*y, z*) *∈* Ψ *}*) (*X, Y,* Φ)+˙ (*Z, W,* Ψ)

= (*X* + *Z, Y* + *W, {* (*ι*1(*x*)*, ι*1(*y*)) *|* (*x, y*) *∈* Φ *}∪{* (*ι*2(*z*)*, ι*2(*w*)) *|* (*x, y*) *∈* Ψ *}*)*.*

## The Sub-Giry Monad

The Giry monad on **Meas** is introduced in [[9](#_bookmark27)] to give a categorical approach to probability theory; each arrow *X → Y* in the Kleisli category of the Giry monad bijectively corresponds to a probabilistic transition from *X* to *Y* , and the Chapman- Kolmogorov equation corresponds to the associativity law of the Giry monad.

We recall the sub-probabilistic variant of the Giry monad, which we call the

*sub-Giry monad* (see also [[18](#_bookmark36), Section 4]):

* For any measurable space (*X,* Σ*X* ), the measurable space *GX* is defined as follows: the underlying set *UGX* of *GX* is the set of subprobability measures over *X*, and the *σ*-algebra Σ*GX* of *GX* is the coarsest *σ*-algebra over *UGX* that makes the evaluation function ev*A* : *GX →* [0*,* 1] (*ν '→ ν*(*A*)) measurable for any *A ∈* Σ*X* .
* For each *f* : *X → Y* in **Meas**, *Gf* : *GX → GY* is defined by (*Gf* )(*ν*)= *ν*(*f−*1(*−*)).
* The unit *η* is defined by *ηX* (*x*)= *δx*, where *δx* is the *Dirac measure* centred on *x*. The multiplication *μ* is defined by *μ* (Ξ)(*A*)= ev *d*(Ξ). The Kleisli lifting of *f* : *X → GY* is given by *f*(*ν*)(*A*)= *X f* (*−*)(*A*) *dν* (*ν ∈ GX*).
* *X* ∫*GX A*

∫

The monad *G* is strong and commutative with respect to the cartesian product in **Meas**. The strength st*−,*= : (*−*)*×G*(=) *⇒ G*(*−×*=) is given by the product measure st*X,Y* (*x, ν*)= *δx⊗ν*. The commutativity of *G* is shown from the Fubini theorem. The double strength dst*−,*= : *G*(*−*)*×G*(=) *⇒ G*(*−×*=) is given by dst*X,Y* (*ν*1*, ν*2)= *ν*1*⊗ν*2. The Kleisli category **Meas***G* is often called the category **SRel** of *stochastic rela- tions* [[18](#_bookmark36), Section 3]. The category **SRel** is *ω***CPO***⊥*-enriched (with respect to the

cartesian monoidal structure) with the following pointwise order:

*f ± g ⇐⇒ ∀x ∈ X, B ∈* Σ*Y .f* (*x*)(*B*) *≤ g*(*x*)(*B*) (*f, g* : *X → Y* in **SRel**)*.*

The *least upper bound* sup*n∈*N *fn* of any *ω*-chain *f*0 *± f*1 *± · · · ± fn ± · · ·* is given by (sup*n fn*)(*x*)(*B*)= sup*n*(*fn*(*x*)(*B*)). The *least function* of each **SRel**(*X, Y* ) (written *⊥X,Y* ) is the constant function of the null-measure over *Y* . The *continuity* of composition is obtained from the following two facts:

* From the definition of Lebesgue integral, for any *ω*-chain *{νn}* of subprobability

measures over *X*, *f d*(sup *νn*)= sup *f dνn* holds.

∫*X n n* ∫*X*

∫ ∫

* From the monotone convergence theorem, we have *X* sup*n fn dν* = sup*n X fn dν*.

This enrichment is equivalent to the partially additive structure on **SRel** [[18](#_bookmark36), Section 5]: For any *ω*-chain *{fn}n∈*N of *fn* : *X → Y* in **SRel**, we have the summable

sequence *{gn}n* where *g*0 = *f*0 and *gn*+1 = *fn*+1 *−fn*. Conversely, for any summable

sequence *{gn}n∈*N, the functions *fn* = Σ*n gn* form an *ω*-chain.

*k*=0

## Differential privacy

Throughout this paper, we define the differential privacy as follows:

**Definition 1.1** ([[8](#_bookmark26), Definition 2.4], Modified) A measurable function (a query) *c* : R*m → G*(R*n*) is (*ε, δ*)-differentially private if *c*(*x*)(*A*) *≤ eεc*(*y*)(*A*)+ *δ* holds whenever *||x − y||*1 *≤* 1 and *A ∈* ΣR*n* , where *||· ||*1 is 1-norm of the space R*m*.

What we modify from the original definition [[8](#_bookmark26), Definition 2.4] is the domain and codomain of *c*; we replace the domain from N to R, and replace the codomain from a discrete probability space to *G*(R*n*). We apply this definition to the interpretation of pWHILE programs. The input and output spaces can be other spaces: in section [5](#_bookmark15) we consider the *above-threshold algorithm* Above whose output space is Z. The above modification is essential in describing and verifying the differential privacy of this algorithm because it takes a sample from Laplace distribution over *real line*.

# A Graded Monad for Differential Privacy

Barthe, K¨opf, Olmedo, and Zanella-B´eguelin constructed a *parametric relational lifting* describing differential privacy, and developed a framework for compositional verification of differential privacy [[2](#_bookmark19)]. The multiplication law of the lifting [[2](#_bookmark19), Lemma 6] plays crucial role to in the formal verification of the differential privacy of queries.

Following this relational approach, we construct the parametric relational lifting of Giry monad to describe differential privacy for *continuous random samplings*. This lifting forms a graded monad on the category **BRel**(**Meas**) in the sense of [[11](#_bookmark29)]. The axioms of graded monad correspond to the (sequential) composition law of differential privacy.

* 1. *Graded Monads*

**Definition 2.1** ([[11](#_bookmark29), Definition 2.2-bis]) Let C be a category, and (*M, ·,* 1*, ≤*) be a

*preordered* monoid. An *M* -graded (or *M* -parametric effect) monad on C consists of

* a collection *{Te}e∈M* of endofunctors on C,
* a natural transformation *η* : Id *⇒ T*1,
* a collection *{μe*1*,e*2 *}e ,e ∈M* of natural transformations *μe*1*,e*2 : *Te Te ⇒ Te e* ,

1

2

1

2

1 2

* a collection *{±e*1*,e*2 *}e*1*≤e*2 of natural transformations *±e*1*,e*2 : *Te*1 *⇒ Te*2

satisfying

* + *μe,*1 *◦ Teη* = *μ*1*,e ◦ ηT*

*e*

= Id*Te*

for any *e ∈ M* ,

* + *μ*(*e*1*e*2)*,e*3 *◦ μe*1*,e*2 *Te*

3

= *μe*1*,*(*e*2*,e*3) *◦ Te μe*2*,e*3 for all *e*1*, e*2*, e*3 *∈ M* ,

* + *±e,e* = Id*T* for any *e* and *±e*2*,e*3 *◦ ±e*1*,e*2 = *±e*1*,e*3 whenever *e*1 *≤ e*2 *≤ e*3,

1

*e*

* + *±*(*e*1*e*2)*,*(*e*3*e*4) *◦μe*1*,e*2 = *μe*3*,e*4 *◦* (*±e*1*,e*3 *∗±e*2*,e*4 ) whenever *e*1 *≤ e*3 and *e*2 *≤ e*4.

We call an *M* -graded monad (*{Te}e∈M , η, μe*1*,e*2 *, ±e*1*,e*2 ) on C an *M* -graded lifting of a monad (*T, ηT , μT* ) on D along *U* : C *→* D if *U Te* = *TU* , *U* (*η*)= *ηT U* , *U* (*μe*1*,e*2 )= *μT U* (*e*1*, e*2 *∈ M* ), and *U* (*±e*1*,e*2 )= id*T* (*e*1 *≤ e*2).

Let *T* be a monad on **Meas**. We call an *M* -graded lifting of the product monad

*T × T* of along the forgetful functor *p* an *M-graded relational lifting* of *T* .

* 1. *A Graded Relational Lifting of Giry Monad for Differential Privacy*

Let *M* be the cartesian product of the monoids ([0*, ∞*)*,* +*,* 0) and ([0*, ∞*)*,* +*,* 0) equipped with the product order of numerical orders. The monoid *M* is the set of parameters (*ε, δ*) of differential privacy. For each (*ε, δ*) *∈ M* , we define the following mapping of **BRel**(**Meas**)-objects by

(*ε,δ*)

*G*

(*X, Y,* Φ)

= (*GX, GY, {* (*ν*1*, ν*2) *| ∀A ∈* Σ*X,B ∈* Σ*Y .*Φ(*A*) *⊆ B* =*⇒ ν*1(*A*) *≤ eεν*2(*B*)+ *δ }*)*.*

**Theorem 2.2** *{G*(*ε,δ*)*}*(*ε,δ*)*∈M forms an M-graded relational lifting of G.*

**Proof.** Since the functor *p* is faithful, it suffices to show:

1. (*Gf, Gg*) is an arrow *G*(*ε,δ*)(*Z, W,* Ψ) *→ G*(*ε,δ*)(*X, Y,* Φ) in **BRel**(**Meas**) for any

arrow (*f, g*): (*Z, W,* Ψ) *→* (*X, Y,* Φ) in **BRel**(**Meas**) and (*ε, δ*) *∈ M* .

*′ ′*

1. (id*GX,* id*GY* ) is an arrow *G*(*ε,δ*)(*X, Y,* Φ) *→ G*(*ε ,δ* )(*X, Y,* Φ) in **BRel**(**Meas**) for

any (*X, Y,* Φ) and (*ε, δ*)*,* (*εj, δj*) *∈ M* that satisfy *ε ≤ εj* and *δ ≤ δj*.

1. (*ηX, ηY* ) is an arrow (*X, Y,* Φ) *→ G*(0*,*0)(*X, Y,* Φ) in **BRel**(**Meas**).

*′ ′ ′*

1. (*μX, μY* ) is an arrow *G*(*ε,δ*)*G*(*ε ,δ* )(*X, Y,* Φ) *→ G*(*ε*+*ε ,δ*+*δ*)(*X, Y,* Φ) in **BRel**(**Meas**)

for any (*X, Y,* Φ) and (*ε, δ*)*,* (*εj, δj*) *∈ M* .

We prove these statements:

1. Let (*ν*1*, ν*2) *∈ G*(*ε,δ*)Ψ. We have Ψ(*f—*1(*A*)) *⊆ g—*1(*B*) for any *A ∈* Σ*X* and *B ∈* Σ*Y* such that Φ(*A*) *⊆ B*. This implies ((*Gf* )(*ν*1)*,* (*Gg*)(*ν*2)) *∈ G*(*ε,δ*)Φ.
2. We have the obvious inclusion *G*(*ε,δ*)Φ *⊆ G*(*ε′,δ′*)Φ.
3. Let (*x, y*) *∈* Φ. We have (*ηX* (*x*)(*A*)*, ηY* (*y*)(*B*)) = (0*,* 0)*,* (0*,* 1)*,* (1*,* 1) for any

*A ∈* Σ*X* and *B ∈* Σ*Y* such that Φ(*A*) *⊆ B*. This implies (*ηX* (*x*)*, ηY* (*y*)) *∈ G*(0*,*0)Φ.

1. We first prove the following equalities:

(*f, g*): Φ

(*ε,δ*)

*G*

*S*(*ε* + *ε ,δ* + *δ*) (*f, g*): Φ *→ S*(*ε ,δ* ) in **BRel**(**Meas**)

(*†*) (*f ×*

*—*1

*g* )

*S*(*ε, δ*)

*.*

*→* (*G*1*, G*1*, ≤*) in **BRel**(**Meas**) ,

(*:*)

Φ =

=

(*f*

*× g* )

*—*1 *j*

*j j* ,

where, *S*(*ε, δ*) = (*G*1*, G*1*, {* (*α*1*, α*2) *| α*1 *≤ eεα*2 + *δ }*) and (*G*1*, G*1*, ≤*) = *S*(0*,* 0). We remark *G*1 *⊆* [0*,* 1].

**(*†*)** We prove in the similar way as [[12](#_bookmark31), Theorem 12]: (*⊇*) Suppose (*ν*1*, ν*2) *∈*

(*f ×*

*g* )

*S*(*ε, δ*) for any (*f, g*): Φ

*—*1

*→≤*, and suppose that *A ∈* Σ

and *B ∈* Σ

satisfy Φ(*A*) *⊆ B*. Since (*χA, χB*): Φ *→ ≤*, we obtain *ν*1(*A*) *≤ eεν*2(*B*)+*δ*. This implies (*ν*1*, ν*2) *∈ G*(*ε,δ*)Φ. (*⊆*) Let (*ν*1*, ν*2) *∈ G*(*ε,δ*)Φ and (*f, g*): Φ *→≤*. Since

*X*

*Y*

Φ(*f—*1[*α,* 1]) *⊆ g—*1[*α,* 1] for any *α ∈* [0*,* 1], we obtain (*f*(*ν*1)*, g*(*ν*2)) *∈ S*(*ε, δ*) from

∫ *fdν*1 = sup Σ*n*

*i*=0

*X*

*αiν*1(*f—*1[Σ*i*

*αi,* 1]) *∀i.*0 *< αi,* Σ*n αi ≤* 1 ,

*≤* sup Σ*n αi*(*eεν*2(*g—*1[Σ*i*

*k*=0

*i*=0

*i*=0

*k*=0

*i*=0

*αi,* 1]) + *δ*) *∀i.*0 *< αi,* Σ*n*

*αi ≤* 1 ,

*≤ eε* ∫ *gdν*2 + *δ.*

*Y*

**(*‡*)** (*⊇*) Obvious. (*⊆*) Suppose that (*kν*1*, lν*2) *∈ S*(*ε, δ*) holds for any (*k, l*): Φ *→≤*. Let (*f, g*): Φ *→ S*(*εj, δj*). The pair (max(*f − δj,* 0)*,* min(*eε g,* 1)) forms an ar- row Φ *→ ≤* in **BRel**(**Meas**) because *f − δj ≤* 1 and 0 *≤ eε g*. We have (*f*(*ν*1)*, g*(*ν*2)) *∈ S*(*ε* + *εj,δ* + *δj*) from

∫ *fdν*1 *− δj ≤* ∫ max(*f − δj,* 0)*dν*1

*′*

*′*

*X*

*X*

*≤ eε* ∫ min(*eε′ g,* 1)*dν* + *δ*

2

*Y*

*≤ e*(*ε*+*ε′*) ∫

*gdν*2

+ *δ.*

Now, we prove the inclusion (*μX*

*Y*

*× μY*

)(*G*(*ε,δ*)*G*(*ε′,δ′*)Φ) *⊆ G*(*ε*+*ε′,δ*+*δ*)Φ.

Let (Ξ *,* Ξ ) *∈ G*(*ε,δ*)*G*(*ε′,δ′*)Φ and (*f, g*): Φ *→ S*(*εjj, δjj*). From the equalities (*†*)

1

2

and (*‡*), we have (*f, g*): *G*(*ε′,δ′*)Φ *→ S*(*εj* + *εjj, δj* + *δjj*). We therefore obtain

(*f*(*μX* (Ξ1))*, g*(*μY* (Ξ2))) = ((*f*)(Ξ1)*,* (*g*)(Ξ2)) *∈ S*(*ε* + *εj* + *εjj,δ* + *δj* + *δjj*)*.*

Since (*f, g*): Φ *→ S*(*εjj, δjj*) is arbitrary, (*μX* (Ξ1)*, μY* (Ξ2)) *∈ G*(*ε*+*ε ,δ*+*δ*)Φ holds.

*′*

*2*

Now we characterise the differential privacy with the lifting *{G*(*ε,δ*)*}*(*ε,δ*)*∈M* .

**Theorem 2.3** *A measurable function c* : R*m → G*(R*n*) *is* (*ε, δ*)*-differentially pri-* *vate* if and only if (*c, c*) *is an arrow {* (*x, y*) *| ||x − y||*1 *≤* 1 *} → G*(*ε,δ*)EqR*n in* **BRel**(**Meas**)*.*

The sequential and parallel composability (see also [[8](#_bookmark26),[14](#_bookmark32)]) of differential privacy are obtained from the following property of the *M* -graded lifting *{G*(*ε,δ*)*}*(*ε,δ*)*∈M* :

## Proposition 2.4 (Composabilities)

1. *For any* (*f ,g* ): Φ

1

1

*→ G*(*ε,δ*)Ψ

*and* (*f ,g* ): Φ

*→ G*(*ε′,δ′*)Ψ

*in* **BRel**(**Meas**)*,*

(dst *◦* (*f × f* )*,* dst *◦* (*g*

1

1

2

2

2

2

1

2

1

2

2

*× g* )) *is an arrow* Φ *×*˙ Φ

*→ G*(*ε*+*ε′,δ*+*δ′*)(Ψ *×*˙ Ψ ) *in*

**BRel**(**Meas**)*.*

1

1

2

1. *For any* (*f ,g* ): Φ *→ G*(*ε,δ*)Ψ *and* (*f ,g* ): Φ

1

1

1

2

2

*→ G*(*ε′,δ′*)Ψ *in* **BRel**(**Meas**)*,*

([*f ,f* ]*,* [*g ,g* ]) *is an arrow* Φ +˙ Φ

2

*→ G*(max(*ε,ε′*)*,*max(*δ,δ*))Ψ *in* **BRel**(**Meas**)*.*

1 2 1 2 1 2

## Proof.

1. It suffices to show that (st*,* st) is an arrow : Φ1*×*˙ *G*(*ε,δ*)Φ2 *→ G*(*ε,δ*)(Φ1*×*˙ Φ2) for any objects Φ1 and Φ2 in **BRel**(**Meas**). We let Φ*i* = (*Xi, Yi,* Φ*i*) (*i* = 1*,* 2).

Suppose ((*x, ν*1)*,* (*y, ν*2)) *∈* Φ1*×*˙ *G*(*ε,δ*)Φ2, and assume that *A ∈* Σ*X ×X* and

1

2

*B ∈* Σ*Y*1*×Y*2 satisfy (Φ1*×*˙ Φ2)(*A*) *⊆ B*. We obtain st(*x, ν*1)(*A*) = *ν*1(*Ax*) and

st(*y, ν*2)(*B*)= *ν*1(*By*). Here, *Ax* = *{ w ∈ X*1 *|* (*x, w*) *∈ A }* and *By* is given in the same way. We have *Ax ∈* Σ*X*1 and *By ∈* Σ*X*2 from the construction of prod- uct spaces. We obtain Φ2(*Ax*) *⊆ By* by *z ∈* Φ2(*Ax*) =*⇒ ∃w ∈ AX.*(*y, z*) *∈* (Φ1*×*˙ Φ2)(*x, w*) This implies st(*x, ν*1)(*A*) *≤ eε*st(*y, ν*2)(*B*)+ *δ*.

1. It suffices to prove *G*(*ε,δ*)Φ *∩G*(*ε′,δ′*)Φ *⊆ G*(max(*ε,ε′*)*,*max(*δ,δ′*))Φ for any object Φ in **BRel**(**Meas**). This is proved from the equality (*†*) in the proof of Theorem [2.2](#_bookmark2) and the inclusion *S*(*ε, δ*) *∩ S*(*εj, δj*) *⊆ S*(max(*ε, εj*)*,* max(*δ, δj*)).

*2*

In fact, we have *S*(*ε, δ*) *∩ S*(*εj, δj*) *⊆ S*(max(log( 1*—δ′′* )+ *ε,* log( 1*—δ′′* )+ *εj*)*, δjj*)

1*—δ* 1*—δ′*

where *δjj* = max(*δ, δj*). Hence, the parallel composability ([ii](#_bookmark9)) can be improved.

## The Symmetrised Lifting of *G*(*ε,δ*)

We recall that the relations *{* (*x, y*) *| ||x − y||*1 *≤* 1 *}* and EqR*n* in Theorem [2.3](#_bookmark7) are symmetric. We hence observe that the *M* -graded lifting *{G*(*ε,δ*)*}*(*ε,δ*)*∈M* describes only one side of inequalities in the definition of differential privacy. By symmetrising this lifting, We obtain an *M* -graded lifting *{G*(*ε,δ*)*}*(*ε,δ*)*∈M* exactly describing the differential privacy for continuous probabilities:

) ) *.*

*G*(*ε,δ*) = *G*(*ε,δ*)(*−*) *∩* (*G*(*ε,δ*)(*−* op op

* 1. *Parametric Lifting in the Original(discrete) apRHL*

In the original works [[2](#_bookmark19),[3](#_bookmark21)] of apRHL, the following parametric relational lifting (*−*)(*ε,δ*) of the (sub)distribution monad *D* on **Set** is introduced to describe dif- ferential privacy. This lifting relates two distributions if there are intermediate distributions *d*1 and *dR*, called *witnesses*, whose skew distance, defined by

Δ*X* (*dL, dR*)= sup max(*dL*(*C*) *− eεdR*(*C*)*, dR*(*C*) *− eεdL*(*C*)*,* 0)*.*

*ε*

*C⊆X*

**Definition 2.5** ([[3](#_bookmark21), Definition 4], [[17](#_bookmark35), Definition 4.3] and [[1](#_bookmark20), Definition 8]) Let Ψ be a relation between sets *X* and *Y* . We define the relation Ψ(*ε,δ*) *⊆ DX × DY* as follows: *d*1 *∈ DX* and *d*2 *∈ DY* satisfy (*d*1*, d*2) *∈* Ψ(*ε,δ*) if and only if there are two (sub)probability distributions *dL, dR ∈ D*(*X × Y* ), called *witnesses*, such that

*Dπ*1(*dL*)= *d*1*, Dπ*2(*dR*)= *d*2*,* supp(*dL*) *⊆* Ψ*,* supp(*dR*) *⊆* Ψ*,* Δ*X×Y* (*dL, dR*) *≤ δ.*

*ε*

**Proposition 2.6** *For any countable discrete spaces X and Y , and relation* Ψ *⊆*

*X × Y , we have* Ψ(*ε,δ*) *⊆ G*(*ε,δ*)Ψ*.*

**Proof.** Suppose (*d*1*, d*2) *∈* Ψ(*ε,δ*) with witnesses *dL* and *dR*. For any *A ⊆ X*, since supp(*dL*) *⊆* Ψ and (*A × Y* ) *∩* Ψ *⊆ X ×* Ψ(*A*), we obtain:

*d*1(*A*)= *Dπ*1(*dL*)(*A*)= *dL*(*A × Y* )= *dL*((*A × Y* ) *∩* Ψ) *≤ dL*(*X ×* Ψ(*A*))

*ε ε*

*≤ εdR*(*X ×* Ψ(*A*)) + *δ* = *e Dπ*2(*dR*)(Ψ(*A*)) + *δ* = *e d*2(Ψ(*A*)) + *δ.*

This implies (*d*1*, d*2) *∈ G*(*ε,δ*)Ψ. Since the construction of (*−*)(*ε,δ*) is symmetric, we conclude (*d*1*, d*2) *∈ G*(*ε,δ*)Ψ. *2*

We remark that we may regard *GX* = *DX* for countable discrete space *X*. When *X* is not countable, we have the same results by embedding each *d ∈ DX* in the set *DXj* of subprobability distributions over the countable *subspace Xj* = *X ∩* supp(*d*).

**Corollary 2.7** *We have* Eq(*ε,δ*) = *G*(*ε,δ*)Eq

*X*

*X*

*for any countable discrete space X.*

**Proof.** (*⊆*) This inclusion is given from Proposition [2.6](#_bookmark10). (*⊇*) Suppose (*d*1*, d*2) *∈*

*G*(*ε,δ*)Eq

. This is equivalent to Δ*X* (*d*1*, d*2) *≤ δ*. Hence (*d*1*, d*2) *∈* Eq(*ε,δ*) is proved

by the witnesses given by *dL* = Σ*x∈X d*1(*x*) *· δ*(*x,x*) and *dR* = Σ*x∈X d*2(*x*) *· δ*(*x,x*). *2*

*X*

*ε*

*X*

When Ψ = *∅* and *δ >* 0, the inclusion of Proposition [2.6](#_bookmark10) is proper, because Ψ(*ε,δ*) is the singleton *{*(0*,* 0)*}*, but *G*(*ε,δ*)Ψ contains at least all pairs (*d*1*, d*2) such that *d*1(*X*)*, d*2(*Y* ) *< δ*. Thus, the lifting *G*(*ε,δ*) is strictly larger than the lifting (*−*)(*ε,δ*) even in the countable discrete cases. This implies that, roughly speaking, we can reuse formal proofs in the original apRHL to the continuous apRHL.

When *ε* = *δ* = 0, the lifting (*−*)(*ε,δ*) describes coalgebraic bisimulations between Markov chains, that is, *D*-coalgebras [[13](#_bookmark30)] (see also [[6](#_bookmark24),[10](#_bookmark28)]), and the lifting *G*(*ε,δ*) corresponds to the relational lifting (codensity lifting) of the sub-Giry monad *G* describing simulations between Markov processes [[12](#_bookmark31), Theorem 12].

# The Continuous apRHL

We introduce a variant of the approximate probabilistic relational Hoare logic (apRHL) to deal with continuous random samplings. We name it the *continuous apRHL*.

* 1. *The Language pWHILE*

We recall and reformulate categorically the language pWHILE [[2](#_bookmark19)]. The language pWHILE is constructed in the standard way, hence we sometimes omit the details of its construction. In this paper, we mainly refer to the categorical semantics of a probabilistic language given in [[5](#_bookmark23), Section 2].

* + 1. *Syntax*

We introduce the syntax of pWHILE by the following BNF:

*τ* ::= bool *|* int *|* real *| ... e* ::= *x | p*(*e*1*,..., em*)

*ν* ::= *d*(*e*1*,..., em*)

*i* ::= *x → e | x →−*$ *ν |* if *e* then *c*1 else *c*2 *|* while *e* do *c*

*c* ::= skip *|* null *| I*; *C*

Here, *τ* is a *value type*; *x* is a *variable*; *p* is an *operation*; *d* is a *probabilistic operation*; *e* is an *expression*; *ν* is a *probabilistic expression*; *i* is an *imperative*; *c* is a *command* (or program). We remark constants are 0-ary operations.

We introduce the following syntax sugars for simplicity:

if *b* then *c* = if *b* then *c* else skip

[while *b* do *c*]*n*

= if *b* then null else skip*,* if *n* =0

if *b* then *c*; [while *b* do *c*]*k,* if *n* = *k* +1

* + 1. *Typing Rules*

We introduce a typing rule on the language pWHILE. A typing context is a finite set Γ = *{x*1 : *τ*1*, x*2 : *τ*2*,..., xn* : *τn}* of pairs of a variable and a value type such that each variable occurs only once in the context.

We give typing rules of pWHILE as follows:

Γ *▶t e*1 : *τ*1 *...* Γ *▶t en* : *τn p* : (*τ*1*,..., τn*) *→ τ*

Γ *▶t p*(*e*1*,..., en*): *τ*

Γ*,x* : *τ t e* : *τ*

Γ*,x* : *τ ▶ x → e* Γ *▶* skip

*▶*

*x* : *τ ∈* Γ Γ *▶t e*1 : *τ*1 *...* Γ *▶t en* : *τn d* : (*τ*1*,..., τn*) *→ τ*

Γ *▶* null

Γ *▶ x →−*$ *d*(*e ,...,e* ): *τ*

1

*n*

Γ *▶ i* Γ *▶ c*

Γ *▶ i*; *c*

Γ *▶t b* : bool Γ *▶ c*1 Γ *▶ c*2

Γ *▶* if *b* then *c*1 else *c*2

Γ *▶t b* : bool Γ *▶ c*

Γ *▶* while *b* do *c*

Here, the type (*τ*1*,..., τn*) *→ τ* of each operation *p* and each probabilistic operation

*d* are assumed to be given in advance.

We easily define inductively the set of free variables of commands, expressions, and probabilistic expressions (denoted by *FV* (*c*), *FV* (*e*), and *FV* (*ν*)).

* + 1. *Denotational Semantics*

We introduce a denotational semantics of pWHILE in **Meas**. We give the interpre- tations **[***τ* **]** of the value types *τ* :

* + - * [[bool]] = B =1 + 1 = *{*true*,* false*}* (discrete space)
      * [[int]] = Z (discrete space)
      * [[real]] = R (Lebesgue measurable space)

We interpret a typing context Γ = *{x*1 : *τ*1*, x*2 : *τ*2*,..., xn* : *τn}* as the product space [[*τ*1]] *×* [[*τ*2]] *×· · · ×* [[*τn*]]. We interpret each operation *p* : (*τ*1*,... τm*) *→ τ* as a mea- surable function **[***p*]] : **[***τ*1]] *× ··· ×* [[*τm*]] *→* [[*τ* ]], and each probabilistic operation *d* : (*τ*1*,... τm*) *→ τ* as **[***d*]] : **[***τ*1]] *× · · · ×* [[*τm*]] *→ G*[[*τ* ]]. Typed termsΓ *▶t e* : *τ* and commands Γ *▶ c* are interpreted to measurable functions of the forms [[Γ]] *→* [[*τ* ]] and [[Γ]] *→ G*[[Γ]] respectively.

The interpretation of expressions are defined inductively by:

[[Γ *▶t x* : *τ* ]] = *πx*: *τ* [[Γ *▶t p*(*e*1*,...,* e*m*)]]= [[*p*]]( **[**Γ *▶t e*1]]*,...* [[Γ *▶t em*]]) The interpretation of commands are defined inductively by:

[[Γ *▶* skip]] = *η*[[Γ]] [[Γ *▶* null]] = *⊥*[[Γ]]*,*[[Γ]] [[Γ *▶ i*; *c*]]= ( **[**Γ *▶ c*]]) *◦* [[Γ *▶ i*]]

[[Γ *▶ x →−*$ *d*(*e*1*,...,* e*m*)]]

= *G*(*ρ*(*x* : *τ,*Γ)) *◦* st[[*τ* ]]*,*[[Γ]] *◦ ⟨*[[*d*]]( **[**Γ *▶t e*1]]*,...* [[Γ *▶t em*]])*,* id[[Γ]]*⟩*

[[Γ*,x* : *τ ▶ x → e*]] = *η*[[Γ*,x* : *τ* ]] *◦ ρ*(*x* : *τ,*Γ) *◦ ⟨*[[Γ*,x* : *τ ▶ e*]]*,* id[[Γ*,x* : *τ* ]]*⟩*

[[Γ *▶* if *b* then *c*1 else *c*2]]= [ **[**Γ *▶ c*1]]*,* [[Γ *▶ c*2]]] *◦ ∼*=[[Γ]] *◦⟨*[[Γ *▶ b*]]*,* id[[Γ]]*⟩*

[[Γ while *b* do *c*]] = sup [[Γ [while *e* do *c*]*n*]]

*▶ ▶*

*n∈*N

Here,

* *ρ*(*xk* : *τk,*Γ) = *⟨fl⟩l∈{*1*,*2*,...,n}* : [[*τk*]]*×* **[**Γ]] *→* [[Γ]], where Γ = *{x*1 : *τ*1*, x*2 : *τ*2*,..., xn* : *τn}*, *fk* = *π*2, and *fl* = *πl ◦ π*2 (*l /*= *k*).
* *∼*=*X* : 2 *× X → X* + *X* is the inverse of [*⟨ι*1*◦*!*X, id⟩, ⟨ι*2*◦*!*X, id⟩*]: *X* + *X →* 2 *× X*, which is obtained from the distributivity of the category **Meas**.

We remark that, from the commutativity of the monad *G*, if Γ *▶ x* : *τ* and *x ∈/ FV* (*c*) then [[Γ *▶ c*]] *∼*= dst[[Γ*′*]]*,*[[*τ* ]]([[Γ*j ▶ c*]] *× η*[[*τ*]]) where Γ*j* =Γ *\ {x* : *τ}*.

* 1. *Judgements*

A judgement of apRHL is

*c*1 *∼ε,δ c*2 : Ψ *⇒* Φ*,*

where *c*1 and *c*1 are commands, and Ψ and Φ are objects in **BRel**(**Meas**). We call the relations Ψ and Φ the *precondition* and *postcondition* of the judgement respectively. Inspired from the validity of asymmetric apRHL [[2](#_bookmark19)], we introduce the validity of the judgement of apRHL.

**Definition 3.1** Let Ψ and Φ be relations on the space [[Γ]]. A judgement *c*1 *∼ε,δ c*2 : Ψ *⇒* Φ is valid (written *|*= *c*1 *∼*(*ε,δ*) *c*2 : Ψ *⇒* Φ) when ([[Γ *▶ c*1]]*,* [[Γ *▶ c*2]]) is an arrow Ψ *→ G*(*ε,δ*)Φ in **BRel**(**Meas**).

We often write preconditions and postconditions in the following manner: Let Γ = *{x*1 : *τ*1*, x*2 : *τ*2*,..., xn* : *τn}*. Assume Γ *▶ e*1 : *τ* and Γ *▶ e*2 : *τ* , and let *R* be a relation on **[***τ* ]] (e.g. =, *≤*,... ). We define the relation *e*1*⟨*1*⟩Re*2*⟨*2*⟩* on **[**Γ]] by

(*e*1*⟨*1*⟩Re*2*⟨*2*⟩*)= *{* (*m*1*, m*2) *∈* **[**Γ]] *|* [[Γ *▶ e*1]](*m*1)*R*[[Γ *▶ e*2]](*m*2) *} .*

To prove (*ε, δ*)-differential privacy of a program Γ *▶ c* in (continuous) apRHL, we show the validity of judgement of the form *c ∼*(*ε,δ*) *c* : *||x⟨*1*⟩− x⟨*2*⟩||*1 *≤* 1 *⇒ y⟨*1*⟩* = *y⟨*2*⟩*, where *x* and *y* are variables for inputs and outputs respectively.

* 1. *Proof Rules*

We mainly refer the proof rules of apRHL from [[2](#_bookmark19),[17](#_bookmark35)], but we modify the [comp] and [frame] rules to verify differential privacy for continuous random samplings.

*x*1 : *τ*1*, x*2 : *τ*2 *∈* Γ Γ *▶t e*1 : *τ*1 Γ *▶t e*2 : *τ*2

(*ρ*(*x*1 : *τ*1*,*Γ) *◦ ⟨*[[*e*1]]*,* id*⟩, ρ*(*x*2 : *τ*2*,*Γ) *◦ ⟨*[[*e*2]]*,* id*⟩*): Ψ *→* Φ

*|*= *x*1 *→ e*1 *∼*(0*,*0) *x*2 *→ e*2 : Ψ *⇒* Φ

[assn]

Γ *▶t e*1 : *τ*1 *...* Γ *▶t e*1 : *τm* Γ *▶t e*2 : *τ*1 *...* Γ *▶t e*2 : *τm x*1 : *τ, x*2 : *τ ∈* Γ

1

*m*

1

*m*

(*⟨*[[*e*1]]*,...,* [[*e*1 ]]*⟩, ⟨*[[*e*2]]*,...,* [[*e*2 ]]*⟩*): Ψ*j* *→* Ψ in **BRel**(**Meas**)

1

*m*

1

*m*

*d* : (*τ*1*,..., τm*) *→ τ* ([[*d*]]*,* [[*d*]]) : Ψ *→ G*(*ε,δ*)(Eq[[*τ* ]]) in **BRel**(**Meas**)

[rand]

$

$

*|*= *x*1 *→− d*(*e*1*,..., e*1 ) *∼ x*2 *→− d*(*e*2*,..., e*2 ): Ψ*j ⇒* (*x*1*⟨*1*⟩* = *x*2*⟨*1*⟩*)

1

*m*

(*ε,δ*)

1

*m*

*|*= *c*1 *∼*(*ε,δ*) *c*2 : Ψ *⇒* Φ*j*

*|*= *cj*1 *∼*(*ε′,δ′*) *cj*2 : Φ*j ⇒* Φ

*|*= skip *∼*(0*,*0) skip: Φ *⇒* Φ

[skip]

*j j* [seq]

*|*= *c*1; *c*1 *∼*(*ε*+*ε′,δ*+*δ′*) *c*2; *c*2 : Ψ *⇒* Φ

Γ *▶t b* : bool Γ *▶t b* : bool Ψ *⇒ b⟨*1*⟩* = *bj⟨*2*⟩*

*|*= *c*1 *∼*(*ε,δ*) *cj*1 : Ψ *∧ b⟨*1*⟩⇒* Φ *|*= *c*2 *∼*(*ε,δ*) *cj*2 : Ψ *∧ ¬b⟨*1*⟩⇒* Φ

*|*= if *b* then *c*1

else *c*2

*∼*(*ε,δ*)

if *bj* then *cj*1

else *cj*2 : Ψ *⇒* Φ

[cond]

Γ *▶t e* : int *ε* = Σ*n—*1 *εk δ* = Σ*n—*1 *δk*

*k*=0

*k*=0

Θ *⇒ b*1*⟨*1*⟩* = *b*2*⟨*2*⟩* Θ *∧ e⟨*1*⟩≥ n ⇒ ¬b*1*⟨*1*⟩*

*∀k* : int*. |*= *c*1 *∼*(*εk,δk* ) *c*2 : Θ *∧ e⟨*1*⟩* = *k ∧ e⟨*1*⟩≤ n* =*⇒* Θ *∧ e⟨*1*⟩ > k*

[while]

*|*= while *b* do *c*1 *∼*(*ε,δ*) while *bj* do *c*2 : Θ *∧ b*1*⟨*1*⟩∧ e⟨*1*⟩≥* 0 *⇒* Θ *∧ ¬b*1*⟨*1*⟩*

*|*= *c*1 *∼*(*ε,δ*) *c*2 : Ψ *∧* Θ *⇒* Φ *|*= *c*1 *∼*(*ε,δ*) *c*2 : Ψ *∧ ¬*Θ *⇒* Φ

*|*= *c*1 *∼*(*ε,δ*) *c*2 : Ψ *⇒* Φ

*|*= *c*1 *∼*(*ε,δ*) *c*2 : Ψ *⇒* Φ Ψ*j ⇒* Ψ Φ *⇒* Φ*j* [weak]

*|*= *c*1 *∼*(*ε,δ*) *c*2 : Ψ*j ⇒* Φ*j*

[case]

*|*= *c*1 *∼*(*ε,δ*) *c*2 : Ψ *⇒* Φ [op]

*|*= *c*2 *∼*(*ε,δ*) *c*1 : Ψop *⇒* Φop

The relational lifting *G*(*ε,δ*) does not preserve every relation composition. However, it preserve the composition of relations if the relations are *measurable*, that is, the images and inverse images along them of measurable sets are also measurable (see

also [[12](#_bookmark31), Section 3.3]). Generally speaking, it is difficult to check measurability of re- lations, hence the continuous apRHL is weak for dealing with relation compositions. However, we have the following two special cases:

* The *equality/diagonal* relation *on* any space is a measurable relation.
* Any relation between *discrete* spaces is automatically a measurable relation. Hence, the following [comp] rule is an extension of the original [comp] rule in [[2](#_bookmark19)]:

Φ and Φ*j*are measurable relations

*|*= *c*1 *∼*(*ε,δ*) *c*2 : Ψ *⇒* Φ *|*= *c*2 *∼*(*ε′,δ′*) *c*3 : Ψ*j ⇒* Φ*j*

*|*= *c*1

*∼*(*ε*+*ε′,*min(*δ*+*eεδ′,δ′*+*eε′ δ*))

*c*3 : Ψ *◦* Ψ*j ⇒* Φ *◦* Φ*j*

[comp]

To define the [frame] rule in continuous apRHL, for any relation Θ on [[Γ]], we define the following relation Range(Θ):

Range(Θ)

= (*ν*1*, ν*2) *∃A, B ∈* Σ[[Γ]]*.*(*A × B ⊆* Θ *∧ ν*1(*A*)= *ν*1([[Γ]]) *∧ ν*2(*B*)= *ν*2([[Γ]])) } *.*

We define the [frame] rule with the construction Range(*−*):

*|*= *c*1 *∼*(*ε,δ*) *c*2 : Ψ *⇒* Φ ( **[***c*1]]*,* [[*c*2]]): Θ *→* Range(Θ)

*|*= *c*1 *∼*(*ε,δ*) *c*2 : Ψ *∧* Θ *⇒* Φ *∧* Θ

[frame]

If [[Γ]] is countable discrete then the condition (*ν*1*, ν*2) *∈* Range(Θ) is equivalent to supp(*ν*1) *×* supp(*ν*2) *⊆* Θ, and hence the above [frame] rule is an extension of the original [frame] rule in [[2](#_bookmark19)].

Note that if the *σ*-algebra of the space **[***τ* **]** contains all singleton subsets, and Θ does not restrict any variables in *FV* (*c*1)*∪FV* (*c*2) then ( **[***c*1]]*,* [[*c*2]]) : Θ *→* Range(Θ).

* 1. *Soundness*

The soundness of the rules [assn] and [case] are given from the composition of arrows in **BRel**(**Meas**). The rules [skip] and [seq] are sound because *G*(*ε,δ*) is an *M* -graded relational lifting of *G*. The rules [weak] and [op] are sound because *G*(*ε,δ*) is monotone with respect to the inclusion order of relations, and preserves opposites of relations. The soundness of [comp] is given from the measurability of the postconditions.

**Lemma 3.2** *The rule [rand] is sound.*

**Proof.** We assume that *x*1 and *x*2 are different variables, since the soundness is obvious if *x*1 and *x*2 are the same variables. We have Γ = Γ*j, x*1 : *τ, x*2 : *τ* . Hence, we let [[Γ]] = [[Γ*j*]] *×* [[*τ* ]] *×* [[*τ* ]]. From the symmetry of discussion, it suffices to show,

*m*

1

*m*

([[Γ *▶ x*1 *→−*$

1

*d*(*e*1*,..., e*1 )]]*,* [[Γ *▶ x*2 *→−*$

*d*(*e*2*,..., e*2 )]]) : Ψ *→ G*(*ε,δ*)(Φ)

holds in **BRel**(**Meas**), where

Φ= (*x*1*⟨*1*⟩* = *x*2*⟨*2*⟩*)= ( **[**Γ **]***,* **[**Γ]]*, {* (*m*1*, m*2) *| πx*1 (*m*1)= *πx*2 (*m*2) *}*)*.*

Let (*m*1*, m*2) *∈* Ψ and *A ∈* Σ[[Γ]]. We have Φ(*A*) = [[Γ*j*]] *×* [[*τ* ]] *× Ax*1 , where *Ax*1 =

*{ π*3(*m*) *| m ∈ A }*. Note that *Ax*1 *∈* Σ[[*τ* ]], and hence Φ(*A*) *∈* Σ[[Γ]]. We write

*νi* = [[*d*]]( **[**Γ *▶t ei* ]](*mi*)*,...,* [[Γ *▶t ei* ]](*mi*)) (*i* = 1*,* 2), and we define *f, g* : [[*τ* ]] *→* [0*,* 1]

1

*m*

by *f* = *χ*(*ρ*(*x* : *τ,*Γ)(*—,m*1))*−*1(*A*) and *g* = *χAx*1 . We then obtain from Fubini theorem:

1

[[Γ *▶ x*1 *→−*$ *d*(*e*1*,..., e*1 )]](*m*1)(*A*)

1

*m*

= *G*(*ρ*(*x* : *τ,*Γ)) *◦* st[[*τ* ]]*,*[[Γ]] *◦ ⟨ν*1*, m*1*⟩*(*A*)= st[*G*[*τ* ]]*,*[[Γ]]

1

= (*ν*1 *⊗ δm* )(*ρ*(*x* : *τ,*Γ)*—* (*A*)) = ∫

(*ν*1*, m*1)(*ρ*(*x* : *τ,*Γ)*—*1(*A*))

1

1 1

[[*τ* ]]*×*[[Γ]]

(*x*1 : *τ,*Γ)

∫

∫

*χρ*(*x* : *τ, −*1(*A*) *d*(*ν*1 *⊗ δm*1 )

=

1 Γ)

*a∈*[[*τ* ]]

*χρ*

[[Γ]]

*−*1(*A*)(*a, −*) *d*(*δm*1 ) *dν*1 = ∫

[[*τ* ]]

*f dν*1

[[Γ *▶ x*2 *→−*$ *d*(*e*2*,..., e*2 )]](*m*2)(Φ(*A*))

1

*m*

= [[Γ *▶ x*2 *→−*$ *d*(*e*2*,..., e*2 )]](*m*2)([[Γ*j*]] *×* [[*τ* ]] *× Ax* )

1

*m*

1

= (*ν*2 *⊗ δm* )(*ρ*(*x* : *τ,*Γ)*—*1([[Γ*j*]] *×* [[*τ* ]] *× Ax* )) = (*ν*2 *⊗ δm* )(*Ax* )= ∫

2

2

1

2

1

*g dν*2*,*

[[*τ* ]]

Since the pair (*f, g*) is an arrow Eq[[*τ* ]] *→≤* in **BRel**(**Meas**), we obtain the followings:

[[Γ *▶ x*1 *→−*$

*d*(*e*1*,..., e*1 )]](*m*1)(*A*) *≤ eε*[[Γ *▶ x*2 *→−*$

*d*(*e*2*,..., e*2 )]](*m*1)(*A*)+ *δ.*

Since *A* is arbitrary, we conclude

1

*m*

1

*m*

([[Γ *▶ x*1 *→−*$

*d*(*e*1*,..., e*1 )]](*m*1)*,* [[*x*2 *→−*$

*d*(*e*2*,..., e*2 )]](*m*2)) *∈ G*(*ε,δ*)(Φ)*.*

*2*

1

*m*

1

*m*

**Lemma 3.3** *The [cond] rule is sound.*

**Proof.** Let (*m*1*, m*2) *∈* Ψ. We have [[Γ *▶ b*]](*m*1)= [[Γ *▶ bj*]](*m*2) from the precondi- tions of the [cond] rule. Since

[[Γ *▶* if *b* then *c*1 else *c*2]]= [ **[**Γ *▶ c*1]]*,* [[Γ *▶ c*2]]] *◦ ∼*=[[Γ]] *◦⟨*[[Γ *▶ b*]]*,* id[[Γ]]*⟩,*

we have the following two cases:

1. If [[Γ *▶ b*]](*m*1)= *ι*1(*∗*) then we have

[[Γ *▶* if *b* then *c*1 else *c*2]](*m*1)= [[Γ *▶ c*1]](*m*1)*,* [[Γ *▶* if *bj* then *cj*1 else *cj*2]](*m*2)= [[Γ *▶ cj*1]](*m*2)

Hence we have the following membership:

([[Γ *▶* if *b* then *c*1 else *c*2]](*m*1)*,* [[Γ *▶* if *bj* then *cj*1 else *cj*2]](*m*2)) *∈ G*(*ε,δ*)Φ*.*

1. If [[Γ *▶ b*]](*m*1)= *ι*2(*∗*) then the same membership holds as in the case ([i](#_bookmark11)).

*2*

**Lemma 3.4** *The [while] rule is sound.*

**Proof.** We write *ci*(*n*)= [while *bi* do *ci*]*n* (*i* = 1*,* 2). We prove by induction on *n*:

*|*= *c*1(*n*) *∼*(Σ*n−*1 *εk,*Σ*n−*1 *δk* ) *c*2(*n*): Θ *∧ b*1*⟨*1*⟩∧ e⟨*1*⟩≥ k ⇒* Θ *∧ e⟨*1*⟩≥ n* + *k.* (1)

*k*=0

*k*=0

**case:** *n* =0 We obtain *|*= null *∼*(0*,*0) null: Θ *∧ b*1*⟨*1*⟩ ∧ e⟨*1*⟩ ≥ k ⇒ ∅* since [[Γ *▶* null **]** is the null measure over [[Γ]]. We recall that the following equality:

*ci*(0) = [while *bi* do *ci*]0 = if *bi* then null else skip*,*

We obtain from the equality ([1](#_bookmark12)) by applying [skip], [cond], and [weak].

**case:** *n* = *m* + 1 From the precondition of [while] and the soundness of [case],

*|*= *c*1 *∼*(*εm,δm*) *c*2 : Θ *∧* (*e⟨*1*⟩* = *k*) =*⇒* (*e⟨*1*⟩ > k*)*.*

By the induction hypothesis,

*|*= *c*1(*m*) *∼*(Σ*m−*1 *εk,*Σ*m−*1 *δk* ) *c*2(*m*): Θ *∧ b*1*⟨*1*⟩∧ e⟨*1*⟩≥ k ⇒* Θ *∧ e⟨*1*⟩≥ m* + *k.*

*k*=0

*k*=0

From the soundness of the [seq] rule, we obtain

*|*= *c*1; *c*1(*m*) *∼*(Σ*m*

*k*=0

*εk,*Σ*m*

*δk* ) *c*2; *c*2(*m*): Θ*∧b*1*⟨*1*⟩∧e⟨*1*⟩≥ k ⇒* Θ*∧e⟨*1*⟩≥ m*+1+*k.*

From the soundness of [weak], [cond], and [skip] we conclude ([1](#_bookmark12)).

*k*=0

Next, it is obvious that Θ *⇒ b*1*⟨*1*⟩* = *b*2*⟨*2*⟩* implies

*|*= while *b*1 do *c*1 *∼*(0*,*0) while *b*2 do *c*2 : Θ *∧ ¬b*1*⟨*1*⟩⇒* Θ *∧ ¬b*1*⟨*1*⟩.* (2)

We write *ε* = Σ*m εk* and *δ* = Σ*m δk*. From ([1](#_bookmark12)) and ([2](#_bookmark13)), we obtain by applying

*k*=0

*k*=0

[cond] and [seq],

*|*= *c*1(*n*); while *b*1 do *c*1 *∼*(*ε,δ*) *c*2(*n*); while *b*2 do *c*2 : Θ*∧b*1*⟨*1*⟩∧e⟨*1*⟩≥* 0 *⇒* Θ*∧¬b*1*⟨*1*⟩.*

We obtain [[Γ *▶ ci*(*n*); while *bi* do *ci*]] = [[Γ *▶* while *bi* do *ci*]] (*i* = 1*,* 2) because the interpretations [[Γ *▶* while *bi* do *ci* **]** is the least upper bound of *{*[[Γ *▶ ci*(*n*)]]*}n* with respect to the *ω***CPO***⊥* structure (see section [1.2](#_bookmark1)). Hence we conclude,

*|*= while *b*1 do *c*1 *∼*(*ε,δ*) while *b*2 do *c*2 : Θ *∧ b*1*⟨*1*⟩∧ e⟨*1*⟩≥* 0 *⇒* Θ *∧ ¬b*1*⟨*1*⟩.*

*2*

**Lemma 3.5** *The rule [frame] is sound.*

**Proof.** Let (*m*1*, m*2) *∈* Ψ *∧* Θ, *ν*1 = [[Γ *▶ c*1]](*m*1), and *ν*2 = [[Γ *▶ c*2]](*m*2). Since (*ν*1*, ν*2) *∈* Range(Θ), there exist *Aj, Bj ∈* Σ[[Γ]] such that *Aj × Bj ⊆* Θ, and *ν*1(*C*)= *ν*1(*C ∧ Aj*) and *ν*2(*D*)= *ν*2(*D ∧ Bj*) for all *C, D ∈* Σ[[Γ]]. Suppose that *A, B ∈* Σ[[Γ]] satisfy (Φ *∧* Θ)(*A*) *⊆ B*. Since *Aj × Bj ⊆* Θ, we have (Φ *∧* (*Aj × Bj*))(*A*) *⊆ B*. This implies Φ(*A ∧ Aj*) *∧ Bj ⊆ B*. Thus, Φ(*A ∧ Aj*) *⊆ B* + ( **[**Γ]] *\* (*B ∨ Bj*)). Therefore

*ν*1(*A*)= *ν*1(*A ∧ Aj*) *≤ eεν*2(*B* + ( **[**Γ]] *\* (*B ∨ Bj*)) + *δ*

= *eεν*2((*B* + ( **[**Γ]] *\* (*B ∨ Bj*)) *∧ Bj*)+ *δ ≤ eεν*2(*B ∧ Bj*)+ *δ ≤ eεν*2(*B*)+ *δ.*

)

.*2*

Hence, (*ν ,ν* ) *∈ G*(*ε,δ*)(Θ *∧* Φ). Similarly, we obtain (*ν ,ν* ) *∈* (*G*(*ε,δ*)(Θ *∧*

1

2

1

2

Φ)

op op

# Differentially Private Mechanisms

In this section, we give a generic method to construct the rules for random sam- plings, and by instantiating the method we show the soundness of the proof rules in prior researches: [Lap] for Laplacian mechanism [[7](#_bookmark25)], [Exp] for Exponential mech- anism [[15](#_bookmark33)], [Gauss] for Gaussian mechanism [[8](#_bookmark26), Theorem 3.22, Theorem A.1], and [Cauchy] for the mechanism by Cauchy distributions [[16](#_bookmark34)].

Let *f* : *X × Y →* R be a positive measurable function, and *ν* be a measure over

*Y* . We define the following function *fa* : Σ*Y* *→* [0*,* 1] by the following normalisation:

*f* (*a,* ) *dν*

*B −*

*f* (*B*)= ∫ *.*

*Y*

∫

*a*

*f* (*a, −*) *dν*

If the function is not ‘almost everywhere zero’ and Lebesgue integrable, that is, 0 *< Y f* (*a, −*) *dν < ∞* then the above *fa*(*−*) is a *probability measure*.

**Proposition 4.1** *Let f* : *X × Y →* R *be a positive measurable function, and ν be a measure over Y . For all a, aj X,* 0 *ε, εj,* 0 *δ, and Z* Σ*Y (window set), if*

*∈ ≤ ≤ ∈*

(*ε*+*ε′,δ*)

*the following three conditions hold then* (*fa, fa′* ) *∈G* (*Y, Y,* Eq*Y* )*:*

*′* ∫

1. 0 *<* 1

*eε*

*Y f* (*aj, −*) *dν ≤* ∫*Y*

*f* (*a, −*) *dν < ∞,*

1. *∀b ∈ Z.f* (*a, b*) *≤ eεf* (*aj, b*)*, and*
2. *fa*(*Y \ Z*) *≤ δ.*

**Proof.** From the three conditions of this proposition, for each *B ∈* Σ*Y* , we obtain,

∫*B f* (*a, −*) *dν*

∫

*eε* ∫*B∩Z f* (*aj, −*) *dν*

(*ε*+*ε′*)

*fa*(*B*)=

*Y*

*f* (*a, −*) *dν ≤*

1

*Y*

*eε′* ∫

*f* (*aj, −*) *dν* + *δ ≤ e*

*fa′* (*B*)+ *δ*

*2*

This proposition is an extension of [[2](#_bookmark19), Lemma 7], and plays the central role in the construction of *sound* proof rules of (continuous) apRHL on random samplings.

## Laplacian mechanism [[7](#_bookmark25)].

We give the function *f* : R *×* R *→* R by *f* (*a, b*) = 2 exp( *—|b—a|* ), where *σ >* 0

*σ*

*σ*

is the variance of Laplacian mechanism. We introduce the probabilistic operation

Lap*σ* : real *→* real with **[**Lap*σ* ]] = *f*(*—*), whose measurability is shown from the continuity of the mapping *a '→ f* (*a, x*)*dx* (*α, β ∈* R).

*α*

∫ *β*

We show (*f*

(*—*)

*, f*(*—*)

): *{* (*a, aj*) *| |a − aj| < r }→ G*( *r ,*0)Eq

by instantiating Propo-

sition [4.1](#_bookmark14) as follows: If *|a − aj| < r* then *ε* = *r/σ*, *εj* = 0, *δ* = 0, the given function

*σ*

R

*f* , the Lebesgue measure *ν* over R, and *Z* = R satisfy the conditions (i)–(iii):

1. Since t∫he function *f* (*a*∫*, −*) is the density function of Laplacian distribution, and

hence

R *f* (*a, −*)*dν* =

R *f* (*aj, −*)*dν* = 1.

1. From the triangle inequality *|b − aj|≤ |a − aj|* + *|b − a|*, we have

= exp *≤* exp *≤* exp *.*

*f* (*a, b*) *|b − aj|− |b − a| |a − aj| r*

*f* (*aj, b*)

*σ*

*σ*

*σ*

This implies *f* (*a, b*) *≤ eεf* (*aj, b*).

1. It is obvious since R *\ Z* = *∅*.

Hence, (*f*

(*—*)

*, f*(*—*)

): *{* (*a, aj*) *| |a − aj| < r }→ G*( *r ,*0)Eq

since *{* (*a, aj*) *| |a − aj| < r }*

and EqR are symmetric. From the [rand] rule, the following proof rule is sound:

*σ*

R

Γ *▶t e*1 : real Γ *▶t e*2 : real *m*1Ψ*m*2 *⇒ |*[[*e*1]]*m*1 *−* [[*e*2]]*m*2*| < r*

[Lap]*.*

$ $

*|*= *x →−* Lap*σ* (*e*1) *∼*( *r ,*0) *y →−* Lap*σ* (*e*2): Ψ *⇒ x⟨*1*⟩* = *y⟨*2*⟩*

*σ*

## Exponential mechanism [[15](#_bookmark33), Modified].

Let *D* be the discrete Euclidean space Z*n*, and (*R, ν*) be a (positive) measure space. Let *q* : *D × R →* R be a measurable function su∫ch that sup*b∈R |q*(*a, b*) *−*

*q*(*aj, b*)*| ≤ c · ||a − aj||*1 for some *c >* 0. Suppose 0 *<*

*R* exp(*γq*(*a, −*)) *dν < ∞*

for any *a ∈ D*. We give the function *f* : *D × R →* R by *f* (*a, b*) = exp(*γq*(*a, b*)), where *γ >* 0 is a constant. We add the value types D and R with **[**D]] = *D* and [[R]] = *R* to pWHILE, and introduce the probabilistic operation Exp*⟨q,ν,ε⟩* : D *→* R with **[**Exp*⟨q,ν,ε⟩*]] = *f*(*—*).

We show (*f*(*—*)*, f*(*—*)): *{* (*a, aj*) *| ||a − aj| |*1 *< r }→ G*(2*γrc,*0)Eq*R* by instantiating Proposition [4.1](#_bookmark14) as follows: Suppose *||a − aj||*1 *< r*. Then *ε* = *εj* = *γcr*, *δ* = 0, the given function *f* , the given measure *ν*, and *Z* = *R* satisfy the conditions (i)–(iii):

1. whenever *||a − aj||*1 *< r*, we obtain,

*f* (*a, b*)

*f* (*aj, b*)

*≤* exp *γ|q*(*a, b*) *− q*(*aj, b*)*|* *≤* exp *γc||a − aj||*1 *≤* exp (*γcr*) *.*

This implies R *f* (*a, −*)*dν ≤ eε* R *f* (*aj, −*)*dν*.

∫ ∫

1. In the same way as (i), we have *f* (*aj, b*) *≤ eε f* (*a, b*).

*′*

1. Obvious.

From the [rand] rule, the following proof rule is sound:

Γ *▶t e*1 : D Γ *▶t e*2 : D *m*1Ψ*m*2 *⇒ ||*[[*e*1]]*m*1 *−* [[*e*2]]*m*2*||*1 *< r*

[Exp]*.*

$ $

*|*= *x →−* Exp*⟨q,ν,ε⟩*(*e*1) *∼*(2*γcr,*0) *y →−* Exp*⟨q,ν,ε⟩*(*e*2): Ψ *⇒ x⟨*1*⟩* = *y⟨*2*⟩*

## Gaussian mechanism [[8](#_bookmark26), Theorem 3.22, Theorem A.1].

We give the function *f* : R*×*R *→* R by *f* (*a, b*)= *√* 1 exp(*−* (*b—a*)2 ), where *σ >* 0

2*πσ*2

2*σ*2

is the variance of Gaussian mechanism. We introduce the probabilistic operation

Gauss*σ* : real *→* real with **[**Gauss*σ*]] = *f*(*—*), whose continuity is easily proved.

We obtain (*f*(*—*)*, f*(*—*)): *{* (*a, aj*) *| |a − aj| < r }→ G*(*ε,δ*)EqR by instantiating Propo- sition [4.1](#_bookmark14) as follows: If *|a−aj| < r*,0 *< ε <* 1, and *εj* = 0 hold, and there is (3*/*2) *< c* such that 2 log(1*.*25*/δ*) *≤ c*2 and (*cr/ε*) *≤ σ*, then the parameters *ε*, *εj*, and *δ*, the given function *f* , and the Lebesgue measure *ν* over R satisfy the conditions (i)–(iii) when *Z* = *b |b −* (*a* + *aj*)*/*2*|≤* (*σ*2*ε/r*) (see [[8](#_bookmark26), Theorem A.1]). From the [rand] rule, the following proof rule is sound:

*∃c >* 3 *.* (2 log( 1*.*25 ) *< c*2 *∧ cr ≤ σ*) 0 *< ε <* 1

}

2

*δ*

*ε*

Γ *▶t e*1 : real Γ *▶t e*2 : real *m*1Ψ*m*2 *⇒ |*[[*e*1]]*m*1 *−* [[*e*2]]*m*2*| < r*

*.*

[Gauss]

$ $

*|*= *x →−* Gauss*σ*(*e*1) *∼*(*ε,δ*) *y →−* Gauss*σ*(*e*2): Ψ *⇒ x⟨*1*⟩* = *y⟨*2*⟩*

We can relax the above conditions for *c* to ((1 + *√*3)*/*2) *< c* and 2 log(0*.*66*/δ*) *< c*2

by changing the window set *Z*.

**Lemm***√***a 4.2 ([**[**8**](#_bookmark26)**, Theorem A.1], Relaxed)** *Suppose |a − aj| < r. Assume that*

((1 + 3)*/*2) *< c,* 0 *< ε <* 1*, and* 0 *< δ <* 1 *satisfy* 2 log(0*.*66*/δ*) *< c*2 *and* (*cr/ε*) *≤*

*σ. Then ε, εj* = 0*, and δ, the function f, and the Lebesgue measure ν over* R *satisfy*

}

*the conditions (i)–(* *iii)* *of Proposition* [*4.1*](#_bookmark14) *when Z*} = *b* *b ≤* (*a* + *aj*)*/*2+ (*σ*2*ε/r*)

*if a ≤ aj and Z* = *b* *b ≥* (*a* + *aj*)*/*2 *−* (*σ*2*ε/r*) *if aj ≤ a.*

**Proof.** We assume *aj ≤ a*. In the case of *aj > a*, we prove in the similar way.

(i) For each *a ∈* R the∫function *f* (*a, −*∫) is the density function of Gaussian distri-

bution, and hence

R *f* (*a, −*)*dν* =

R *f* (*aj, −*)*dν* = 1.

(ii) We have *Z* = *b b ≤* (*a* + *aj*)*/*2+ (*σ*2*ε/r*) . Take an arbitrary *b ∈ Z*. We then calculate as follows:

*f* (*a, b*)

}

(*b − aj*)2 *−* (*b − a*)2

*ε*

2*σ*2

1

*j a* + *aj*

*≤* exp

= exp

*f* (*aj, b*)

*r* (*b σ*2

*a* + *aj*

)

2

*≤* exp

*r σ*2*ε*

*σ*2 *r*

= exp

*σ*2 (*a − a* )(*b −*

*≤ e .*

This implies *∀b ∈ Z.f* (*a, b*) *≤ eεf* (*aj, b*).

*−*

)

2

1. Let *H* = *a′—a* + *σε*. Since ((1+*√*3)*/*2) *< c*, *−r < aj−a*, and *cr ≤ σ*, we have1 *<*

*√*

*δ*

) *< c*2 *−* 1. We obtain 2 log(

*δ*

1

2*π*

) *< c*2 *−* 1 *< H*2 from

*c−*  1

2*c*

*—* 2*c < H*. Thus 0 *<* log(*H*). We have 2 log( *δ*

2*π* )

*c*2. Thus 2 log(

2*σ r*

*< c ε*

1

2*π*

*√*

1 √ *e ε ≤* 2 log(0*.*66*/δ*) *<*

*√*

* 1. *< c −*  *ε*

*c −*

2*c*

2*c*

*< H*. Therefore, we conclude log(

*δ*

1 ) *<* log(*H*)+ *H*2*/*2.

2*π*

Let *Hj* = *a*+*a′* + *σ*2*ε* . We have *f* (R *\ Z*) *≤ δ* from the following calculation:

* 1. *r*

∫ 1

*a*

(*x − a*)2

R*\Z*

*σ√*2*π* exp *−*

2*σ*2 *dν*

1 ∫ *∞*

*√*

=

2*π*

exp

*db*

(*x − a*)2 1 ∫ *∞*

*b*2

1 ∫ *∞*  *b b*2 1

*√*

*σ*

*H′*

exp

*—*

2*σ*2

*dx* =

2*π*

*H*

*—* 2

*H*2

*≤ √*2*π H H* 2

*—*

exp

*db ≤ √*2*πH* exp *−* 2

*≤ δ.*

*2*

## Mechanism of Cauchy distributions [[16](#_bookmark34)]

We give the function *f* : R R R by *f* (*a, b*)= *ρ* . We introduce the

*× →*

*π*((*a—b*)2+*ρ*2)

probabilistic operation Cauchy*ρ* : real *→* real with **[**Cauchy*ρ*(*e*)]]Γ*m* = *f*(*—*), whose

continuity is easily proved.

Let *ε* = log 1+ *r*2+*r√r*2+4*ρ*2 . We obtain (*f*

2*ρ*2

(*—*)

*, f*(*—*)

): *{* (*a, aj*) *| |a − aj| < r }→*

*G*(*ε,*0)EqR by instantiating Proposition [4.1](#_bookmark14) as follows: If *|a − aj| < r* then the pa- rameters satisfy the conditions (i)–(iii): the given *ε*, *εj* = 0, *δ* = 0, the Lebesgue measure *ν* over R, and *Z* = R.

From the [rand] rule, we obtain the following rule:

Γ *▶t e* : real *m*1Ψ*m*2 *⇒ |*[[*e*1]]*m*1 *−* [[*e*2]]*m*2*| < r*

*|*= *x →−*

$

Cauchy (*e*1) *∼*

*y →−*$

Cauchy (*e*1): Ψ *⇒* (*πx × πy*)*—*1(Eq )

[Cauchy]

# Example: The Above Threshold Algorithm

*ρ*

(*ε,*0)

*ρ*

R

Barthe, Gaboardi, Gr´egoire, Hsu, and Strub extended the logic apRHL to the logic apRHL+ with new proof rules to describe the *sparse vector technique* (see also [[8](#_bookmark26), Section 3.6]). They gave a formal proof of the differential privacy of *above threshold algorithm* in [[1](#_bookmark20)].

In this section, we demonstrate that the above threshold algorithm with *real- valued queries* is proved with *almost the same proof* as in [[1](#_bookmark20)]. The new proof rules of apRHL+ are still sound in the framework of the continuous apRHL.

We consider the following algorithm AboveT:

We recall the setting of this algorithm. This algorithm has two fixed parameters: the threshold *t* : real and the set *Q* : queries of queries where *|Q|* : int is the number of *Q*. The input variable is *d* : int, and the output variable is *r* : int. We prepare the new value types queries and data with **[**data]] = R*N* and queries =

**Algorithm 1** The Above Threshold Algorithm ([[1](#_bookmark20)], Modified)

1: AboveT(*T* : real, *Q* : queries, *d* : data)

2: *j →* 1; *r → |Q|* + 1; *T →−* Lap*ε/*2(*t*);

$

3: while *j < |Q|* do

4: *S →−* Lap*ε/*4(eval(*Q, i, d*));

$

5: if *T ≤ S ∧ r* = *|Q|* +1 then

6: *r → j*;

7: *j → j* +1

int (alias), and the typings *j* : int, *T* : real, and *S* : real. We assume that an operation eval: (queries*,* int*,* data) *→* real is given for evaluating *i*-th query in *Q* for the input *d*. We require **[**eval]] to be 1*-sensitivity* for the data *d*, that is,

*||d − dj||*1 *≤* 1 *⇒ |*[[eval]](*Q, i, d*) *−* [[eval]](*Q, i, dj*)*|≤* 1.

The differential privacy of Above is characterised as follows:

*|*= AboveT *∼*exp(*ε*)*,*0 AboveT: *||d⟨*1*⟩− d⟨*2*⟩||*1 *≤* 1 *⇒ r⟨*1*⟩* = *r⟨*2*⟩.*

The following rules in apRHL+ are sound in the framework of continuous apRHL:

Σ

*∀i* : int*. |*= *c*1 *∼*(*ε,δi*) *c*2 : Ψ *⇒* (*x⟨*1*⟩* = *i ⇒ x⟨*2*⟩* = *i*) *i* : int [[*δi*]] = *δ* [Forall-Eq]

*|*= *c*1 *∼*(*ε,δ*) *c*2 : Ψ *⇒ x⟨*1*⟩* = *x⟨*2*⟩*

Γ *▶t e*1 : real Γ *▶t e*2 : real *m*1Ψ*m*2 *⇒ |*[[*e*1]]*m*1 + *rj −* [[*e*2]]*m*2*| < r*

*|*= *x →−*

$

Lap*σ*

(*e*1) *∼*( *r ,*0)

*y →−*$

Lap*σ*

(*e*2): Ψ *⇒ x⟨*1*⟩* + *rj* = *y⟨*2*⟩*

[LapGen]

Γ *▶t e*1 : real Γ *▶t e*2 : real *x ∈/ FV* (*e*1) *y ∈/ FV* (*e*2)

*σ*

[LapNull]

$ $

*|*= *x →−* Lap*σ* (*e*1) *∼*(0*,*0) *y →−* Lap*σ* (*e*2): Ψ *⇒ x⟨*1*⟩− y⟨*2*⟩* = *e*1*⟨*1*⟩− e*2*⟨*2*⟩*

Hence we extend the continuous apRHL by adding these rules, and therefore we construct a formal proof almost the same proof as in [[1](#_bookmark20)] in the extended continuous apRHL.

The soundness of the rule [Forall-Eq] is proved from the following lemma:

**Lemma 5.1 ([**[**1**](#_bookmark20)**, Proposition 6], Modified)** *If x* : *τ ∈* Γ *and the space* [[*τ* ]] *is countable and discrete then*

*i∈* [[*τ* ]]

*G*(*ε,δi*)(*x⟨*1*⟩* = *i ⇒ x⟨*2*⟩* = *i*) *⊆ G*(*ε,*Σ*i∈*[[*τ* ]] *δi*)(*x⟨*1*⟩* = *x⟨*2*⟩*)*.*

**Proof.** Let [[Γ*,x* : *τ* ]] = [[*τ* ]] *×* [[Γ]]. Suppose (*ν*1*, ν*2) *∈ i∈*[[*τ*]] *G*(*γ,δi*)(*x⟨*1*⟩* = *i ⇒ x⟨*2*⟩* = *i*). Take an arbitrary *A ∈* Σ[[Γ*,x* : *τ* ]]. Since **[***τ* **]** is countable and discrete, we decompose *A* = *i∈*[[*τ* ]](*{i}× Ai*). We may assume *Ai /*= *∅* because *{i}×∅* = *∅*. Since (*x⟨*1*⟩* = *i ⇒ x⟨*2*⟩* = *i*)(*{i} × Ai*) = *{i} ×* [[Γ]], we obtain *ν*1(*{i} × Ai*) *≤ eεν*2(*{i}×* [[Γ]]) + *δi* for each *i ∈* [[*τ* ]]. By summing them up, we obtain *ν*1(*A*) *≤ e ν*2((*x⟨*1*⟩* = *x⟨*2*⟩*)(*A*)) + *i∈*[[*τ* ]] *δi*. *2*

Σ

*ε* Σ

The soundness of the rule [LapGen] is proved from the rules [Lap] and [assn]

and the semantically equivalence **[***x →−*$ Lap (*e* + *rj*); *x → x − rj*]] = [[*x →−*$ Lap (*e*)]].

*σ σ*

The soundness of [LapNull] is proved by using the [LapGen] and [Frame] rules.

## Formal Proof

We now demonstrate that the (*ε,* 0)-differential privacy of algorithm AboveT is proved with almost the same proof as in [[1](#_bookmark20)].

From the [Forall-Eq] rule with variable *r*, it suffices to prove for all integer *i*,

*|*= AboveT *∼*(*ε,*0) AboveT: *||d⟨*1*⟩− d⟨*2*⟩||*1 *≤* 1 *⇒* (*r⟨*1*⟩* = *i ⇒ r⟨*2*⟩* = *i*)*.*

We denote by *c*0 the sub-command consisting of the initialisation line 2 of AboveT. From the rules [assn], [LapGen] rule with *r* = *rj* = 1, and *σ* = 2*/ε*, [seq], and [frame] we obtain

where

*|*= *c*0 *∼*(*ε/*2*,*0) *c*0 : *||d⟨*1*⟩− d⟨*2*⟩||*1 *≤* 1 *⇒ ||d⟨*1*⟩− d⟨*2*⟩||*1 *≤* 1 *∧* Ψ*.*

Ψ= *T⟨*1*⟩* +1 = *T⟨*2*⟩∧ j⟨*1*⟩* = *j⟨*2*⟩∧ j⟨*1*⟩* =1 *∧ r⟨*1*⟩* = *r⟨*2*⟩∧ r⟨*1*⟩* = *|Q|* + 1*.*

We denote by *c*1 and *c*2 the main loop and the body of the main loop respectively (i.e. *c*1 = while (*j < |Q|*) do *c*2). We aim to prove the following judgement by using the [while] rule:

*|*= *c*1 *∼*(*ε/*2*,*0) *c*1 : (*||d⟨*1*⟩− d⟨*2*⟩||*1 *≤* 1 *∧* Ψ) *⇒* (*r⟨*1*⟩* = *i ⇒ r⟨*2*⟩* = *i*)*.*

To prove this, it suffices to show the following cases for the loop body *c*2:

1. If *k < i* then *|*= *c*2 *∼*(0*,*0) *c*2 : (Θ *∧ j⟨*1*⟩* = *k*) *⇒* (Θ *∧ j⟨*1*⟩ > k*)
2. If *k* = *i* then *|*= *c*2 *∼*(*ε/*2*,*0) *c*2 : (Θ *∧ j⟨*1*⟩* = *k*) *⇒* (Θ *∧ j⟨*1*⟩ > k*)
3. If *k > i* then *|*= *c*2 *∼*(0*,*0) *c*2 : (Θ *∧ j⟨*1*⟩* = *k*) *⇒* (Θ *∧ j⟨*1*⟩ > k*) Here, we provide the following *loop invariant* as follows:

Θ =(*j⟨*1*⟩ < i ⇒* ((*r⟨*1*⟩* = *|Q|* +1 *⇒ r⟨*2*⟩* = *|Q|* + 1) *∧* (*r⟨*1*⟩* = *|Q|* +1 *∨ r⟨*1*⟩ < i*)))

*∧* (*j⟨*1*⟩≥ i ⇒* (*r⟨*1*⟩* = *i ⇒ r⟨*2*⟩* = *i*))

*∧ ||d⟨*1*⟩− d⟨*2*⟩||*1 *≤* 1 *∧ T⟨*1*⟩* +1 = *T⟨*2*⟩∧ j⟨*1*⟩* = *j⟨*2*⟩*

The judgement in the case ([i](#_bookmark16)) is proved from the rules [seq], [assn], [cond], and [frame] and the following fact obtained from the [LapNull] rule:

$ $

*|*=*S →−* Lap*ε/*4(eval(*Q, i, d*)) *∼*(0*,*0) *S →−* Lap*ε/*4(eval(*Q, i, d*)):

(*||d⟨*1*⟩− d⟨*2*⟩||*1 *≤* 1) *∧* (*T⟨*1*⟩* +1 = *T⟨*2*⟩*) *⇒* ((*S⟨*1*⟩ < T⟨*1*⟩*) *⇒* (*S⟨*2*⟩ < T⟨*2*⟩*))*.*

The case ([ii](#_bookmark17)) is proved from the rules [seq], [assn], [cond], and [frame] and the following fact obtained from the [LapGen] rule:

$ $

*|*=*S →−* Lap*ε/*4(eval(*Q, i, d*)) *∼*(*ε/*2*,*0) *S →−* Lap*ε/*4(eval(*Q, i, d*)):

(*||d⟨*1*⟩− d⟨*2*⟩||*1 *≤* 1 *∧ T⟨*1*⟩* +1 = *T⟨*2*⟩*) *⇒* (*S⟨*1*⟩* +1 = *S⟨*2*⟩∧ T⟨*1*⟩* +1 = *T⟨*2*⟩*)*.*

The case ([iii](#_bookmark18)) is proved in the similar way as ([i](#_bookmark16)).

# Acknowledgement

The author thanks Shin-ya Katsumata for many valuable comments and stimulating discussions, Marco Gaboardi for helpful suggestions and the introduction of his preprint of [[1](#_bookmark20)] in arXiv, Gilles Barthe, Masahito Hasegawa, Naohiko Hoshino, Takeo Uramoto and anonymous reviewers of MFPS for advices that contributed to improve the writing of this paper.

# References

1. Gilles Barthe, Marco Gaboardi, Benjamin Gr´egoire, Jastin Hsu, and Pierre-Yves Strub. Proving Differential Privacy via Probabilistic Couplings. In *Proceedings of Thirty-First Annual ACM/IEEE* *Symposium on LOGIC IN COMPUTER SCIENCE (LICS)*, to appear.
2. Gilles Barthe, Boris Ko¨pf, Federico Olmedo, and Santiago Zanella-B´eguelin. Probabilistic relational reasoning for differential privacy. In *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL ’12, pages 97–110, New York, NY, USA, 2012. ACM.
3. Gilles Barthe and Federico Olmedo. Beyond differential privacy: Composition theorems and relational logic for f-divergences between probabilistic programs. In F. Fomin, R. Freivalds, M. Kwiatkowska, and

D. Peleg, editors, *Automata, Languages, and Programming*, volume 7966 of *Lecture Notes in Computer Science*, pages 49–60. Springer Berlin Heidelberg, 2013.

1. Nick Benton. Simple relational correctness proofs for static analyses and program transformations. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL ’04)*, number MSR-TR-2005-26, page 43. ACM, January 2004.
2. Daniel Brown and Riccardo Pucella. Categories of timed stochastic relations. *Electronic Notes in Theoretical Computer Science*, 249:193 – 217, 2009. Proceedings of the 25th Conference on Mathematical Foundations of Programming Semantics (MFPS 2009).
3. E.P de Vink and J.J.M.M Rutten. Bisimulation for probabilistic transition systems: a coalgebraic approach. *Theoretical Computer Science*, 221(1 - 2):271 – 293, 1999.
4. Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer Berlin Heidelberg, 2006.
5. Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and* *Trends⃝*R *in Theoretical Computer Science*, 9(3-4):211–407, 2013.
6. Mich`ele Giry. A categorical approach to probability theory. In B. Banaschewski, editor, *Categorical Aspects of Topology and Analysis*, volume 915 of *Lecture Notes in Mathematics*, pages 68–85. Springer Berlin Heidelberg, 1982.
7. Bart Jacobs and Jesse Hughes. Simulations in coalgebra. *Electronic Notes in Theoretical Computer Science*, 82(1):128–149, 2003. CMCS’03, Coalgebraic Methods in Computer Science (Satellite Event for ETAPS 2003).
8. Shin-ya Katsumata. Parametric effect monads and semantics of effect systems. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL ’14, pages 633–645, New York, NY, USA, 2014. ACM.
9. Shin-ya Katsumata and Tetsuya Sato. Codensity Liftings of Monads. In Lawrence S. Moss and Pawel Sobocinski, editors, *6th Conference on Algebra and Coalgebra in Computer Science (CALCO 2015)*, volume 35 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 156–170, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
10. Kim Guldstrand Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and* *Computation*, 94(1):1–28, 1991.
11. Frank McSherry. Privacy integrated queries. Association for Computing Machinery, Inc., June 2009.
12. Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS ’07, pages 94–103, Washington, DC, USA, 2007. IEEE Computer Society.
13. Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC ’07, pages 75–84, New York, NY, USA, 2007. ACM.
14. Federico Olmedo. *Approximate Relational Reasoning for Probabilistic Programs*. PhD thesis, Technical University of Madrid, 2014.
15. Prakash Panangaden. The category of markov kernels. *Electronic Notes in Theoretical Computer Science*, 22:171 – 187, 1999. PROBMIV’98, First International Workshop on Probabilistic Methods in Verification.