# Formalizing the Transformation of Product Haar Measures under Component-wise Continuous Multiplicative Equivalences in Lean 4: A Proof of map_haar_pi

This report provides a comprehensive guide to completing the formal proof of the map_haar_pi lemma in Lean 4, utilizing the Mathlib library. The lemma describes the behavior of a product of Haar measures under a transformation defined by a family of continuous multiplicative equivalences. Due to the inaccessibility of the specific GitHub context referenced in the user query [1], this report will proceed by making reasonable assumptions based on standard Mathlib definitions and conventions for any potentially locally defined terms. The aim is to furnish a rigorous and detailed pathway to the proof, suitable for researchers and advanced students engaged in formal mathematics.

## Section 1: Introduction and Lemma Statement

The formalization of mathematical theories in proof assistants like Lean 4 necessitates precise statements and rigorous proofs for even foundational results. The map_haar_pi lemma is one such result, pertinent to the study of measures on product spaces, particularly in the context of topological groups endowed with Haar measures.

### 1.1. The map_haar_pi Lemma

The lemma is stated in Lean 4 as follows:

Lean

```
lemma map_haar_pi [Fintype ι] (ψ : ∀ i, (H i) ≃□* (H i)) :
  Measure.map (ContinuousMulEquiv.piCongrRight ψ)
    (Measure.pi fun i ↦ haar) =
  (Π i, mulEquivHaarChar (ψ i)) •
    Measure.pi fun i ↦ haar
```

Mathematically, this lemma asserts how a product of Haar measures, defined on a product of topological groups Hi, transforms under a specific type of mapping. The mapping, ContinuousMulEquiv.piCongrRight ψ, is constructed from a family of continuous multiplicative equivalences ψi, each acting on a component group Hi. The lemma states that the pushforward of the product Haar measure under this composite map is equivalent to the original product Haar measure scaled by a constant. This constant is the product of individual scaling factors, mulEquivHaarChar (ψ i), where each factor quantifies how the respective equivalence ψi scales the Haar measure on its corresponding group Hi.

This type of result is fundamentally a change of variables theorem for measures, specialized to the algebraic and topological context of Haar measures on product groups. The product ΠimulEquivHaarChar(ψi) plays a role analogous to the Jacobian determinant in multivariate calculus, capturing the overall scaling effect of the transformation F=ContinuousMulEquiv.piCongrRight ψ. The structure Measure.mapFμ=C⋅μ is characteristic of how a measure μ transforms under a map F that scales the measure by a constant C, rather than strictly preserving it. The product structure of both the space and the map F suggests that this overall scaling factor C arises from the multiplicative combination of scaling effects on each component.

## 1.2. Contextual Assumptions and Prerequisites

To proceed with the proof, certain assumptions about the components of the lemma are necessary, especially given the inaccessible source file. These assumptions are grounded in standard Mathlib practices.

- **Assumption on** Hi: Each Hi is assumed to be a topological group. For the Haar measure haar to be well-defined and essentially unique (up to a positive scalar multiple), each Hi must be locally compact and Hausdorff.[2] This report will operate under this standard assumption. The groups are equipped with the Borel σ-algebra, which is standard for topological groups in Mathlib's measure theory framework.[2]
- **Assumption on haar**: The term haar is assumed to refer to a left-invariant regular Haar measure on Hi. In Mathlib, this could be an instance of MeasureTheory.Measure.haarMeasure (H i) or any measure satisfying the IsHaarMeasure typeclass.[2] The specific normalization constant chosen for each haar (H i) is not critical for the validity of the lemma's structure, as the mulEquivHaarChar (ψ i) term is defined relative to the *chosen* Haar measure on Hi and will correctly capture its scaling.
- **Assumption on Fintype ι**: The index set ι is a finite type (Fintype ι). This is a crucial hypothesis. Mathlib's definition of the product measure Measure.pi is typically given for finite index sets.[4] Furthermore, the product ΠimulEquivHaarChar(ψi) is a finite product, ensuring it is well-defined and typically results in a finite, non-negative (often positive) real number, assuming

each mulEquivHaarChar (ψ i) is itself such a number. Infinite products of measures or constants would require additional considerations of convergence not present in this lemma's statement.

- **Measurability**: All functions and sets involved are assumed to be measurable with respect to the appropriate Borel σ-algebras. Continuous functions, such as those forming ContinuousMulEquiv, are Borel measurable. The map ContinuousMulEquiv.piCongrRight ψ will also be shown to be measurable.

The inability to access the original Lean file [1] means that any local definitions or specific properties of Hi, haar, or particularly mulEquivHaarChar are unknown. The proof strategy outlined here will rely on interpretations consistent with Mathlib's general framework. The validity of the resulting proof in the user's specific context will depend on how closely these assumptions align with their actual, inaccessible definitions. Adhering to standard Mathlib interpretations provides the most broadly applicable guidance.

## Section 2: Deconstructing the Lemma: Core Mathlib Definitions

Understanding the map_haar_pi lemma requires familiarity with several key definitions from the Mathlib library, spanning measure theory, algebra, and topology.

- 2.1. MeasureTheory.Measure α (Measure Space)
  A measure μ on a measurable space α (a type equipped with a σ-algebra of measurable sets) is a function that assigns a value in ENNReal (the extended non-negative real numbers, [0,∞]) to each measurable set. It must satisfy μ(∅)=0 and countable additivity: for any countable sequence of pairwise disjoint measurable sets Sj, μ(⋃jSj)=Σjμ(Sj).5 In Mathlib, measures are technically defined as outer measures that are countably additive on measurable sets, and the outer measure is the canonical extension of the restricted measure.
- 2.2. Measure.map (f : α → β) (μ : Measure α)
  The pushforward measure, denoted Measure.map f μ, describes how a measure μ on a space α is transported to a space β via a measurable function f:α⟶β. For any measurable set S⊆β, the measure of S under the pushforward measure is defined as (Measure.mapfμ)(S)=μ(f−1(S)).5 This is a fundamental construction for understanding how measures transform under mappings.
- 2.3. Measure.pi (μ : ∀ i, Measure (α i)) (Product Measure)
  Given a family of measure spaces (αi,Mi,μi) indexed by i∈ι, where ι is a Fintype, Measure.pi μ (where μ is the function i↦μi) defines the product measure on the product space ∀i,αi (the space of functions from ι to ⋃iαi such that x(i)∈αi for each i). This product measure is characterized by its action on "measurable

rectangles": if $S=\text{Set.pi univ } s$ (where s is a function $i\mapsto s_i$, and each $s_i\subseteq\alpha_i$ is measurable), then $(\text{Measure.pi } \mu)(S)=\prod_i\mu_i(s_i)$.[4] The construction often involves ensuring σ-finiteness of the component measures.

- 2.4. haar (Haar Measure)
  A Haar measure on a locally compact topological group G is a non-trivial (not identically zero) regular measure that is left-invariant. Left-invariance means that for any $g\in G$ and any measurable set $S\subseteq G$, $\mu(gS)=\mu(S)$, where $gS=\{gx\mid x\in S\}$. Equivalently, Measure.map (fun x ↦ g * x) μ = μ. Haar measures are unique up to multiplication by a positive scalar constant.[2] In the lemma, haar appears as fun i ↦ haar, suggesting it's a dependent function providing the Haar measure for each group $H_i$. This is likely MeasureTheory.Measure.haarMeasure (H i) or a similar instance from the IsHaarMeasure class.
- 2.5. ContinuousMulEquiv α β (Notation: α ≃□* β)
  This represents a continuous multiplicative equivalence between two topological groups (or monoids) α and β. It bundles a MulEquiv α β (a group isomorphism) with proofs that both the forward map and its inverse (the symm map of the MulEquiv) are continuous functions.[7] This structure ensures that the algebraic isomorphism also respects the topological structures, making it a homeomorphism that is also a group isomorphism.
- 2.6. ContinuousMulEquiv.piCongrRight (ψ : ∀ i, H i ≃□* K i)
  This function constructs a ContinuousMulEquiv between product types. Given a family of continuous multiplicative equivalences $\psi_i:H_i\simeq_t*K_i$ indexed by $i\in\iota$, ContinuousMulEquiv.piCongrRight ψ defines an equivalence from $(\forall i,H_i)$ to $(\forall i,K_i)$. It acts component-wise: if $x=(x_i)_{i\in\iota}$ is an element of $\forall i,H_i$, then $(\text{ContinuousMulEquiv.piCongrRight } \psi)(x)=(\psi_i(x_i))_{i\in\iota}$. The continuity of this product map follows from the continuity of its components when using the product topology. The name is analogous to structures like Equiv.piCongrRight [9] and LinearMap.pi.[10] In the context of the lemma, $K_i=H_i$ for all i.
- 2.7. mulEquivHaarChar ($\psi_i$ : H i ≃□* H i)
  This term, mulEquivHaarChar $\psi_i$, represents the "Haar characteristic" or scaling factor associated with the continuous multiplicative equivalence $\psi_i$ acting on the group $H_i$ with its Haar measure haar (H i).
  Inferred Definition: mulEquivHaarChar $\psi_i$ is the unique positive constant $c_i\in\mathbb{R}_{\geq 0}$ (typically $c_i\in\mathbb{R}_{>0}$ if $H_i$ is non-trivial) such that
  $\text{Measure.map } \psi_i(\text{haar } (H_i))=c_i\cdot(\text{haar } (H_i))$
  This concept is standard in Haar measure theory. An automorphism ϕ of a locally compact group G transforms a Haar measure μ into another Haar measure Measure.map ϕμ. By the uniqueness of Haar measure, $\text{Measure.map } \phi\mu=c\cdot\mu$ for some positive constant c, often called the modulus of the automorphism ϕ.

mulEquivHaarChar $\psi_i$ captures this modulus for the equivalence ψi. While specific snippets 7 and 18 do not define this, 13 (showing LinearIsometryEquiv can be measure preserving) implies that more general equivalences might scale measures.

- 2.8. Π i,... (Finset.prod)
  This denotes the product of a family of terms indexed by the Fintype ι. In the lemma, ΠimulEquivHaarChar(ψi) is the finite product of the scaling factors ci obtained from each component equivalence ψi.11 This product results in an ENNReal value.
- 2.9. c • μ (Scalar Multiplication of a Measure)
  For a scalar c∈ENNReal (typically $c \in R \geq 0$) and a measure μ, c⋅μ (or c • μ in Lean) is a new measure defined by (c⋅μ)(S)=c⋅μ(S) for all measurable sets S. This is a standard operation in Mathlib.

The interplay of these definitions is central to the lemma. It combines measure theory (Measure.map, Measure.pi, haar), algebra (MulEquiv, Π), and topology (Continuous), all within the framework of ENNReal-valued measures. The proof must navigate these structures carefully. For instance, the map ContinuousMulEquiv.piCongrRight ψ is built from components ψi which are ContinuousMulEquiv. This ensures each ψi is continuous and thus Borel measurable. The product map itself, under the product topology, will also be continuous because its component functions are continuous. Continuous maps between topological spaces (equipped with Borel σ-algebras) are measurable. This measurability is a prerequisite for Measure.map (ContinuousMulEquiv.piCongrRight ψ)... to be well-defined.

The following table summarizes these key components:

**Table 1: Key Mathlib Definitions and Their Roles**

| **Mathlib Term** | **Brief Description / Role in Lemma** | **Relevant Snippet(s) / Basis for Inference** |
|---|---|---|
| Measure.map f μ | Pushforward measure of μ under f. | [5] |
| Measure.pi (fun i ↦ $\mu_i$) | Product measure on ∀ i, H i. | [4] |
| haar (H i) | Haar measure on the group H | [2] |

| | | |
|---|---|---|
| | i. | |
| ContinuousMulEquiv (≃□*) | Continuous multiplicative isomorphism. | [7] (general) |
| ContinuousMulEquiv.piCongrRight ψ | Component-wise application of ψ i on ∀ i, H i. | Inferred; cf. [9] |
| mulEquivHaarChar (ψ i) | Scaling factor ci such that map (ψ i) haar = c_i • haar. | Inferred; standard concept, cf. [13] |
| Π i, $c_i$ (Finset.prod) | Product of scaling factors ci. | [11] |
| c • μ | Scalar multiplication of measure μ by c. | Standard Mathlib definition |

This table serves as a quick reference, mapping the lemma's notation to its conceptual meaning within Mathlib. It is particularly useful for terms like mulEquivHaarChar where the definition is inferred based on common patterns in Haar measure theory.

## Section 3: Proof Strategy and Key Mathlib Theorems

The proof of the map_haar_pi lemma will proceed by demonstrating the equality of two measures. A standard technique in measure theory to show that two measures μ1 and μ2 on a space X are equal is to show that they assign the same value to all integrals of non-negative measurable functions, i.e., ∫Xgdμ1=∫Xgdμ2 for all suitable test functions g. In Mathlib, this typically involves using the Lebesgue-Stieltjes integral lintegral (denoted ∫⁻) for functions mapping to ENNReal.

### 3.1. Overall Approach: Equality of Measures via lintegral

Let F:=ContinuousMulEquiv.piCongrRight ψ.
Let μpi:=Measure.pi (fun i ↦ haar (H i)).
Let C:=ΠimulEquivHaarChar(ψi).
The lemma to be proven is Measure.map Fμpi=C·μpi.
To establish this equality, it is sufficient to show that for any non-negative, measurable function g:(∀i,Hi)→R≥0∞ (i.e., g:(∀i,Hi)→ENNReal), the following holds:
$$ \int⁻ g , \partial(\text{Measure.map } F \mu_{\text{pi}}) = \int⁻ g , \partial(C \cdot
\mu_{\text{pi}}) $$

### 3.2. Transforming the Left-Hand Side (LHS)

The left-hand side involves an integral with respect to a pushforward measure. The change of

variables formula for lintegral (often found in Mathlib as MeasureTheory.lintegral_map or a similar theorem) states that for a measurable function F and a non-negative measurable function g:
$$ \int⁻ g , \partial(\text{Measure.map } F \mu_{\text{pi}}) = \int⁻ (g \circ F) , \partial\mu_{\text{pi}} $$
This transformation relies on the measurability of F (which, as discussed, holds because F is continuous) and g. The general concept of change of variables in integration is fundamental.14

## 3.3. Transforming the Right-Hand Side (RHS)

The right-hand side involves an integral with respect to a scalar multiple of a measure. The property MeasureTheory.lintegral_smul_measure (or an equivalent) states that for a non-negative scalar $C \in \text{ENNReal}$:
$$\int^- g \partial (C \cdot \mu\text{pi}) = C \cdot \int^- g \partial \mu\text{pi}$$

## 3.4. The Core Task: Relating Integrals

Combining these transformations, the proof reduces to showing:
$$ \int⁻ (g \circ F) , \partial\mu_{\text{pi}} = \left( \prod_i \text{mulEquivHaarChar}(\psi_i) \right) \cdot \int⁻ g , \partial\mu_{\text{pi}} $$
This equation is itself a specialized change of variables formula for the specific map F and the product measure μpi. The entire proof strategy hinges on establishing this equality, leveraging the definition of mulEquivHaarChar and the properties of product measures.

## 3.5. Key Theorem: Fubini-Tonelli for Measure.pi (using lintegral)

The integral $\int^- (g \circ F) \partial \mu\text{pi}$ is an integral over a product measure. The Fubini-Tonelli theorem allows such an integral to be computed as an iterated integral. Mathlib provides versions of this theorem, such as MeasureTheory.lintegral_prod for the product of two measures [16], which can be generalized or applied iteratively for finite products like Measure.pi over a Fintype ι. A theorem like MeasureTheory.lintegral_pi (or an equivalent constructed via MeasurableEquiv as suggested by the Measure.pi construction notes [4]) would be used. This theorem often requires component measures to be σ-finite. Haar measures on locally compact Hausdorff groups are indeed σ-finite (they are Radon measures, which are σ-finite on σ-compact spaces; locally compact groups are σ-compact if they are, for example, second-countable, but Haar measures are generally σ-finite even without this).

Using such a theorem, $\int^- (x \mapsto (g \circ F)(x)) \partial(\text{Measure.pi (fun } i \mapsto \text{haar } (H\ i)))$ can be expressed as an iterated integral. If ι is ordered $i1,\ldots,in$:
$$ \int⁻_{H_{i_1}} \dots \int⁻_{H_{i_n}} (g \circ F)(x_{i_1}, \dots, x_{i_n}) , \partial(\text{haar}(H_{i_n})) \dots \partial(\text{haar}(H_{i_1})) $$
The Fubini-Tonelli theorem is the bridge that allows the product structure of the measure and the map to be analyzed component by component.

## 3.6. Applying mulEquivHaarChar Property Component-wise

Recall that (g∘F)(x)=g((ψi(xi))i∈ι). When the iterated integral is evaluated, say for the innermost integral with respect to xik over haar(Hik), the integrand will depend on ψik(xik). The defining property of mulEquivHaarChar(ψi) is that Measure.map ψi(haar (Hi))=mulEquivHaarChar(ψi)⋅(haar (Hi)). This translates to an lintegral property: for any non-negative measurable function f′:Hi⟶ENNReal,
$$ \int⁻ (f' \circ \psi_i) , \partial(\text{haar } (H_i)) = \text{mulEquivHaarChar}(\psi_i) \cdot \int⁻ f' , \partial(\text{haar } (H_i)) $$
This is precisely how a "local Jacobian" factor emerges. This property will be applied iteratively to each component integral in the Fubini-Tonelli expansion. Each application will extract a mulEquivHaarChar(ψik) factor, which is constant with respect to the remaining outer integrals.

### 3.7. Reconstructing the Product and Final Integral

After iterating through all components i∈ι, the product of these scaling factors, ∏imulEquivHaarChar(ψi), will be factored out of the entire iterated integral. The remaining iterated integral will be of the form:
$$ \int⁻_{H_{i_1}} \dots \int⁻_{H_{i_n}} g(y_{i_1}, \dots, y_{i_n}) , \partial(\text{haar}(H_{i_n})) \dots \partial(\text{haar}(H_{i_1})) $$
(where yik are now the integration variables after the change of variables yik=ψik(xik) has been accounted for by the mulEquivHaarChar factor). This remaining iterated integral is, by applying the Fubini-Tonelli theorem in reverse, simply ∫−g∂μpi.
Thus, the left-hand side ∫−(g∘F)∂μpi becomes (∏imulEquivHaarChar(ψi))⋅∫−g∂μpi, which matches the target expression derived from the right-hand side of the original goal.

## Section 4: Step-by-Step Proof Derivation (Formal Outline)

The formal proof in Lean 4 will involve a sequence of rewrites and applications of theorems corresponding to the strategy outlined above.

- 4.1. Setup and Goal Statement
  The proof typically starts by introducing all hypotheses and the arbitrary non-negative measurable test function g.
  Lean
  ```
  proof
    -- Assume ι, H, ψ are given as per the lemma statement.
    -- Let F := ContinuousMulEquiv.piCongrRight ψ
    -- Let μ_pi := Measure.pi (fun i ↦ haar (H i))
    -- Let C := ∏ i, mulEquivHaarChar (ψ i)
    -- The goal is: Measure.map F μ_pi = C • μ_pi

    -- To prove equality of measures, show equality of lintegrals for any
  non-negative measurable g.
    apply Measure.ext_lintegral_measurable_on -- or a similar extensionality
  principle
  ```

```
 intro g hg_nn hg_meas -- g is non-negative and measurable

 -- LHS: ∫⁻ g ∂(Measure.map F μ_pi)
 -- RHS: ∫⁻ g ∂(C • μ_pi)

 -- Apply change of variables to LHS
 rw MeasureTheory.lintegral_map hg_meas F.measurable_toFun
 -- Goal: ∫⁻ (g ∘ F) ∂μ_pi = ∫⁻ g ∂(C • μ_pi)
 -- F.measurable_toFun follows from F being a ContinuousMulEquiv, hence
continuous, hence Borel measurable.

 -- Apply scalar multiplication property to RHS
 rw MeasureTheory.lintegral_smul_measure _ C -- C is an ENNReal constant
 -- Goal: ∫⁻ (g ∘ F) ∂μ_pi = C * ∫⁻ g ∂μ_pi
```

The term F.measurable_toFun asserts that F (the function part of ContinuousMulEquiv.piCongrRight ψ) is measurable. Since F is constructed from continuous equivalences ψi, F itself is continuous with respect to the product topology. Continuous functions between topological spaces equipped with Borel σ-algebras are Borel measurable. This is a standard result in Mathlib.

- 4.2. Applying Fubini-Tonelli to ∫⁻ (g ∘ F) ∂μ_pi
  The next step is to decompose the integral ∫−(g∘F)∂μpi using a Fubini-Tonelli type theorem for Measure.pi. Since ι is a Fintype, an equivalence e : ι ≃ Fin (Fintype.card ι) can be used to establish an order for iteration. Mathlib's Measure.pi construction itself might rely on such an ordering.4 A theorem like MeasureTheory.lintegral_pi_equiv (this name is illustrative; the actual theorem might be lintegral_pi or an iterated application of lintegral_prod) would be applied.

Lean
```
 -- Let Φ := g ∘ F
 -- Current goal: ∫⁻ Φ ∂μ_pi = C * ∫⁻ g ∂μ_pi
 -- Apply Fubini-Tonelli to the LHS integral: ∫⁻ Φ ∂(Measure.pi (fun i ↦ haar (H i)))
 -- This might involve `rw MeasureTheory.lintegral_pi Φ` or similar.
 -- Assuming Fintype.card ι = n, and using an equiv e : ι ≃ Fin n
 -- The LHS becomes an iterated integral:
 -- ∫⁻ (x₀ : H (e.symm 0)) ∂(haar (H (e.symm 0))),...
 --   ∫⁻ (x_{n-1} : H (e.symm (Fin.last (n-1)))) ∂(haar (H (e.symm (Fin.last (n-1))))),
 --     Φ (fun i ↦ (match e i with | 0 => x₀ |... | n-1 => x_{n-1}))
 -- (The exact syntax for reconstructing the pi-type element will depend on
Mathlib's API)
```

The handling of Fintype ι and iteration is a common pattern. Mathlib often uses Fintype.equivFin to obtain an explicit indexing by Fin k, which is linearly ordered, facilitating iterated integration. The choice of this equivalence does not affect the outcome due to the symmetry of product measures and finite products over commutative operations.

- 4.3. Iteratively Applying the mulEquivHaarChar Property
  The core of the proof lies in transforming each layer of the iterated integral. Consider the innermost integral, say over $x_k$ with respect to $\mathrm{haar}(H_{i_k})$ (where $i_k = e.\mathrm{symm}\ k$). The function $\Phi$ is $g \circ F$, so $\Phi(x_0, \dots, x_{n-1}) = g((\psi_{i_0}(x_0)), \dots, (\psi_{i_{n-1}}(x_{n-1})))$.
  For the innermost integral with respect to $x_{n-1}$ over $\mathrm{haar}(H_{i_{n-1}})$:
  Let $G(\dots, y_{n-1}) =$ integrand involving $g(\dots, y_{n-1}, \dots)$.
  The integral is $\int^{-}(x_{n-1} \mapsto G(\dots, \psi_{i_{n-1}}(x_{n-1}), \dots))\partial(\mathrm{haar}(H_{i_{n-1}}))$.
  Using the lintegral version of the mulEquivHaarChar definition:
  $\int^{-}(f' \circ \psi_j)\partial(\mathrm{haar}(H_j)) = \mathrm{mulEquivHaarChar}(\psi_j) \cdot \int^{-} f' \partial(\mathrm{haar}(H_j))$.
  This allows rewriting the innermost integral as:
  $\mathrm{mulEquivHaarChar}(\psi_{i_{n-1}}) \cdot \int^{-}(y_{n-1} \mapsto G(\dots, y_{n-1}, \dots))\partial(\mathrm{haar}(H_{i_{n-1}}))$.
  This process is repeated for each integral from innermost to outermost. Each step j factors out $\mathrm{mulEquivHaarChar}(\psi_{i_j})$.
  Lean
  ```
   -- Schematically, after applying Fubini-Tonelli:
   -- LHS = ∫⁻ dx_0... ∫⁻ dx_{n-1}, (g ◦ F) (x_0,..., x_{n-1})
   -- Consider the innermost integral: ∫⁻ dx_{n-1}, (g' ◦ ψ_{i_{n-1}}) (x_{n-1})
   --   (where g' depends on x_0,..., x_{n-2} and the other ψ_j(x_j))
   -- This becomes: (mulEquivHaarChar (ψ_{i_{n-1}})) * ∫⁻ dx_{n-1}, g'(x_{n-1})
   -- Pull this constant factor out. Repeat for x_{n-2},..., x_0.
   -- This will require lemmas that allow pulling constants out of lintegral,
   -- and the change of variables for each ψ_i.
   -- E.g., using `lintegral_map_equiv (ψ i_k)` effectively.
  ```

  The order of applying rewrite rules (rw) in Lean is crucial. Unfolding definitions or applying theorems appropriately will manage the goal state. If mulEquivHaarChar and related functions have @[simp] lemmas, some steps might be automated by the simp tactic.[9]
- 4.4. Accumulating Finset.prod and Finalizing
  After iterating through all n components, all the $\mathrm{mulEquivHaarChar}(\psi_{i_j})$ factors will have been extracted. Their product is precisely $C = \prod_i \mathrm{mulEquivHaarChar}(\psi_i)$.
  The remaining iterated integral will be:
  $$ \int⁻ (x_0 : H_{i_0}) \dots \int⁻ (x_{n-1} : H_{i_{n-1}}), g(x_0, \dots, x_{n-1}) ,

\partial(\text{haar}(H_{i_{n-1}})) \dots \partial(\text{haar}(H_{i_0})) $$
This is, by the Fubini-Tonelli theorem (e.g., MeasureTheory.lintegral_pi or equivalent) applied in reverse, equal to ∫−g∂μpi.
So, the LHS has been transformed into C·∫−g∂μpi.
Lean
```
 -- After all n steps, the LHS becomes:
 -- (Π j in Finset.range n, mulEquivHaarChar (ψ (e.symm j))) *
 --   (∫⁻ dx_0... ∫⁻ dx_{n-1}, g(x_0,..., x_{n-1}))
 -- The product term is C, by Finset.prod_equiv e.symm or similar.
 -- The iterated integral is ∫⁻ g ∂μ_pi, by Fubini-Tonelli in reverse.
 -- So, LHS = C * ∫⁻ g ∂μ_pi.
 -- This matches the RHS.
 rw Finset.prod_univ_equiv e (fun j => mulEquivHaarChar (ψ (e.symm j))) -- or similar for product rearrangement
 -- Apply Fubini-Tonelli in reverse (e.g., `← MeasureTheory.lintegral_pi g`)
 -- The goal becomes C * ∫⁻ g ∂μ_pi = C * ∫⁻ g ∂μ_pi, which is true by `rfl`.
qed
```

- **4.5. Key Mathlib Theorems to Invoke (or find equivalents)**
  - MeasureTheory.Measure.ext_lintegral_measurable_on: For proving measure equality via integrals.
  - MeasureTheory.lintegral_map: The change of variables formula for lintegral and Measure.map.
  - MeasureTheory.lintegral_smul_measure: Property of lintegral with respect to scalar multiplication of measures.
  - A Fubini-Tonelli theorem for Measure.pi and lintegral, such as an iterated version of MeasureTheory.lintegral_prod [16] or a direct MeasureTheory.lintegral_pi. This is the most crucial and potentially complex part to locate or adapt.
  - The definitional property of mulEquivHaarChar (ψ i): This might be a lemma like MeasureTheory.lintegral_map_equiv_smul_of_haarChar (hypothetical name) derived from the definition Measure.map (ψ i) haar = (mulEquivHaarChar (ψ i)) • haar. If mulEquivHaarChar is custom to the user's library, its definition *is* effectively this property for Measure.map, which then implies the lintegral version.
  - ContinuousMulEquiv.measurable_toFun (or proving its components: continuity implies Borel measurability).
  - Properties of Finset.prod for re-indexing or distributing, if needed (e.g. Finset.prod_equiv).[11]

- Measurability theorems for compositions of functions and functions on product spaces.

## Section 5: Addressing Inaccessible Context and Conclusion

The proof strategy detailed above provides a robust framework for formally verifying the map_haar_pi lemma in Lean 4. However, its direct applicability depends on certain assumptions made due to the inaccessibility of the lemma's original context.

### 5.1. Summary of Assumptions Made

The core assumptions underpinning this guide are:

1. Each space Hi is a locally compact Hausdorff topological group, allowing for a well-defined Haar measure.
2. haar refers to a standard left-invariant Haar measure on Hi, as provided by Mathlib (e.g., MeasureTheory.Measure.haarMeasure).
3. mulEquivHaarChar (ψ i) is defined as the scalar ci such that Measure.map (ψi)(haar (Hi))=ci⋅(haar (Hi)). This is the standard definition of the modulus of a group automorphism/equivalence with respect to Haar measure.
4. Standard Mathlib theorems for lintegral, Measure.map, Measure.pi, and Fubini-Tonelli are available and applicable.

### 5.2. Recap of the Proof Strategy

The proof proceeds by establishing the equality of the measures Measure.map (ContinuousMulEquiv.piCongrRight ψ)(Measure.pi fun i ↦ haar) and (ΠimulEquivHaarChar(ψi))⋅(Measure.pi fun i ↦ haar) by showing that they yield the same lintegral for any arbitrary non-negative measurable function g. This involves:

1. Applying the lintegral_map (change of variables) theorem to the left-hand side.
2. Applying the lintegral_smul_measure theorem to the right-hand side.
3. Using a Fubini-Tonelli type theorem (e.g., lintegral_pi) to decompose the integral over the product measure into iterated integrals.
4. Iteratively applying the defining property of mulEquivHaarChar (ψ i) to each component integral, factoring out the respective characteristic constant.
5. Recombining the product of these constants and the remaining iterated integral (which simplifies back to ∫−g∂(Measure.pi fun i ↦ haar)) to match the transformed right-hand side.

This strategy is robust because it relies on fundamental principles of measure theory that are well-established in Mathlib. Even if the exact names of theorems differ slightly from those hypothesized (e.g., lintegral_pi_equiv), equivalent functionalities should exist.

### 5.3. Potential Challenges and Verifications

The user undertaking this proof should:

1. **Verify Definitions:** Crucially, confirm that their local definition of mulEquivHaarChar aligns with the assumed property. If it's defined differently, the proof steps involving its application will need adjustment.
2. **Ensure Measurability:** While ContinuousMulEquiv implies continuity and thus Borel measurability for the main map F and its components ψi, all intermediate functions constructed during the Fubini-Tonelli expansion must also be measurable. This is usually handled by Mathlib's measurability calculus.
3. **Locate Fubini-Tonelli for Measure.pi:** The precise form of the Fubini-Tonelli theorem for Measure.pi and lintegral over a Fintype index set needs to be identified in Mathlib. Documentation for MeasureTheory.Measure.pi [4] and MeasureTheory.Integral.Prod [16] are good starting points. It might be MeasureTheory.lintegral_pi or require iterative use of MeasureTheory.lintegral_prod.
4. **Handle Fintype Iteration:** The iteration order for Fubini-Tonelli over Fintype ι needs to be correctly managed, possibly using Fintype.equivFin and properties of Finset.prod under equivalences.

### 5.4. Broader Implications and Connections

The map_haar_pi lemma is more than a technical exercise; it embodies a fundamental property of how measures behave under structured transformations.

- **Modulus of Automorphisms:** It generalizes the concept of the modulus of a single group automorphism to product groups and component-wise equivalences. The term mulEquivHaarChar is precisely this modulus for each component.
- **Harmonic Analysis:** Such transformation rules are foundational in harmonic analysis on groups, particularly when dealing with function spaces and integral transforms on product domains.
- **Probability Theory:** In contexts where Haar measures are normalized to be probability measures (e.g., on compact groups), this lemma describes how product probability distributions transform.
- **Geometric Measure Theory:** Analogues of this result appear in geometric measure theory when considering how volumes change under diffeomorphisms, where the Jacobian determinant plays a similar role to ΠimulEquivHaarChar(ψi).

The term mulEquivHaarChar (ψ i) is indispensable because ContinuousMulEquiv does not generally preserve measure. Haar measure is unique only up to a scalar multiple; mulEquivHaarChar quantifies this unique scalar for the transformation ψi. Without this factor, the lemma would be incorrect. For example, if Hi=R (additive group) and ψi(x)=ax for a偃=0,±1, then haar(R) (Lebesgue measure) is scaled by ∣a∣. `mulEquivHaarChar}(\psi_i)$ would be $|a|$.

## 5.5. Concluding Remarks

Successfully formalizing the map_haar_pi lemma in Lean 4 represents a significant step in developing a rigorous library for measure theory on topological groups. The process involves a careful interplay of algebraic, topological, and measure-theoretic concepts, all precisely expressed within Lean's dependent type theory. This endeavor not only validates the mathematical statement but also enhances the toolkit available for more advanced formalizations. Proving this lemma is an excellent exercise in mastering these advanced areas of Mathlib. Should specific tactic assistance or theorem identification become challenging, the Mathlib community, particularly on Zulip [17], is a valuable resource.

## Works cited

1. accessed December 31, 1969, https://github.com/AbdulShabazz/FLT/blob/main/FLT/HaarMeasure/HaarChar/AddEquiv.lean
2. Mathlib.MeasureTheory.Measure.Haar.Basic - Lean community, accessed June 11, 2025, https://leanprover-community.github.io/mathlib4_docs/Mathlib/MeasureTheory/Measure/Haar/Basic.html
3. Mathlib.MeasureTheory.Measure.Haar.OfBasis - Floris van Doorn, accessed June 11, 2025, https://florisvandoorn.com/carleson/docs/Mathlib/MeasureTheory/Measure/Haar/OfBasis.html
4. Mathlib.MeasureTheory.Constructions.Pi, accessed June 11, 2025, https://leanprover-community.github.io/mathlib4_docs/Mathlib/MeasureTheory/Constructions/Pi.html
5. Mathlib.MeasureTheory.Measure.MeasureSpaceDef - Lean community, accessed June 11, 2025, https://leanprover-community.github.io/mathlib4_docs/Mathlib/MeasureTheory/Measure/MeasureSpaceDef.html
6. Mathlib.Probability.Density - Lean community, accessed June 11, 2025, https://leanprover-community.github.io/mathlib4_docs/Mathlib/Probability/Density.html
7. How to set the version of both lean-toolchain and mathlib of a Lean 4 project to a

stable Lean version? - Proof Assistants Stack Exchange, accessed June 11, 2025, https://proofassistants.stackexchange.com/questions/4701/how-to-set-the-version-of-both-lean-toolchain-and-mathlib-of-a-lean-4-project-to
8. Splines/lean-continuous: Continuous functions formalized in Lean4. A students project accompanied by a YouTube video. - GitHub, accessed June 11, 2025, https://github.com/Splines/lean-continuous
9. Mathlib.Tactic.Simps.Basic - Lean community, accessed June 11, 2025, https://leanprover-community.github.io/mathlib4_docs/Mathlib/Tactic/Simps/Basic.html
10. Mathlib.LinearAlgebra.Pi - Lean community, accessed June 11, 2025, https://leanprover-community.github.io/mathlib4_docs/Mathlib/LinearAlgebra/Pi.html
11. Mathlib.Data.Finset.Card - Lean community, accessed June 11, 2025, https://leanprover-community.github.io/mathlib4_docs/Mathlib/Data/Finset/Card.html
12. Mathlib.Data.Finset.Functor - Lean community, accessed June 11, 2025, https://leanprover-community.github.io/mathlib4_docs/Mathlib/Data/Finset/Functor.html
13. Mathlib.MeasureTheory.Measure.Haar.InnerProductSpace - Lean community, accessed June 11, 2025, https://leanprover-community.github.io/mathlib4_docs/Mathlib/MeasureTheory/Measure/Haar/InnerProductSpace.html
14. [2207.12742] A formalization of the change of variables formula for integrals in mathlib, accessed June 11, 2025, https://arxiv.org/abs/2207.12742
15. A Formalization of the Change of Variables Formula for Integrals in mathlib - ResearchGate, accessed June 11, 2025, https://www.researchgate.net/publication/367588832_A_Formalization_of_the_Change_of_Variables_Formula_for_Integrals_in_mathlib
16. Mathlib.MeasureTheory.Integral.Prod, accessed June 11, 2025, https://leanprover-community.github.io/mathlib4_docs/Mathlib/MeasureTheory/Integral/Prod.html
17. leanprover-community/mathlib4: The math library of Lean 4 - GitHub, accessed June 11, 2025, https://github.com/leanprover-community/mathlib4
18. Lean 4 math proofs - need some help, can't get mathlib working. : r/haskell - Reddit, accessed June 11, 2025, https://www.reddit.com/r/haskell/comments/1k0wy4w/lean_4_math_proofs_need_some_help_cant_get/