COMPUTABLY T.D.L.C. GROUPS

NOTES FOR TWO TALKS, NEWCASTLE, 2022

ANDRE NIES

Abstract. These notes form a short version of A. Melnikov and A. Nies. *Computably totally disconnected locally compact groups* (2022), preprint, available at

https://arxiv.org/pdf/2204.09878.pdf.

In the first talk we will introduce two notions of computable presentation of a t.d.l.c. group, and show their equivalence. The first notion relies on standard notions of computability in the uncountable setting. The second notion restricts computation to a countable structure of approximations of the elements, the "meet groupoid" of compact open cosets. Based on this, I obtain various examples of computably t.d.l.c. groups, such as $Aut(T_d)$ and some algebraic groups over the field of p-adic numbers. The first talk also outlines the computability theoretic notions that are needed.

In the second talk we show that given a computable presentation of a t.d.l.c. group, the modular function and the Cayley-Abels graphs (in the compactly generated case) are computable. We discuss the open question whether the scale function can be non-computable. We will give a criterion based on meet groupoids when the computable presentation is unique up to computable isomorphism. We explain why the class of computably t.d.l.c. groups is closed under most of the constructions studied by Wesolek [24, Thm. 1.3].

We thank Stephan Tornier for helpful conversations on t.d.l.c. groups, and for providing references.

Contents

1. Introduction

The talks are centred on the following questions.

(a) How can one define a computable presentation of a t.d.l.c. group? Which t.d.l.c. groups have such a presentation?
(b) Given a computable presentation of a t.d.l.c. group, are objects such as the rational valued Haar measures, the modular function, or the scale function computable?
(c) Do constructions that lead from t.d.l.c. groups to new t.d.l.c. groups have algorithmic versions?
(d) When is a computable presentation of a t.d.l.c. group unique up to computable isomorphism?

1.1. **Background on t.d.l.c. groups.** Van Dantzig [23] showed that each t.d.l.c. group has a neighbourhood basis of the identity consisting of compact open subgroups. With Question (a) in mind, we discuss six well-known examples of t.d.l.c. groups, and indicate a compact open subgroup when it is not obvious. We will return to them repeatedly during the course of the paper.

(i) All countable discrete groups are t.d.l.c.
(ii) All profinite groups are t.d.l.c.
(iii) $(\mathbb{Q}_p, +)$, the additive group of p-adic numbers for a prime p is an example of a t.d.l.c. group that is in neither of the two classes above. The additive group $\mathbb{Z}_p$ of p-adic integers forms a compact open subgroup.
(iv) The semidirect product $\mathbb{Z} \ltimes \mathbb{Q}_p$ corresponding to the automorphism $x \mapsto px$ on $\mathbb{Q}_p$, and $\mathbb{Z}_p$ is a compact open subgroup.
(v) Algebraic groups over local fields, such as $\mathrm{SL}_n(\mathbb{Q}_p)$ for $n \geq 2$, are t.d.l.c. Here $\mathrm{SL}_n(\mathbb{Z}_p)$ is a compact open subgroup.
(vi) Given a connected countable undirected graph such that each vertex has finite degree, its automorphism group is t.d.l.c. The stabiliser of any vertex forms a compact open subgroup.

By convention, all t.d.l.c. groups will be infinite.

1.2. **Computable structures: the countable case.** Towards defining computable presentations, we first recall the definition of a computable function on $\mathbb{N}$, slightly adapted to our purposes in that we allow the domain to be any computable set.

Definition 1.1. Given a set $S \subseteq \mathbb{N}^k$, where $k \geq 1$, a function $f\colon S \to \mathbb{N}$ is called computable if there is a Turing machine that on inputs $n_1, \ldots, n_k$ decides whether the tuple of inputs $(n_1, \ldots, n_k)$ is in S, and if so outputs $f(n_1, \ldots, n_k)$.

One version of the Church-Turing thesis states that computability in this sense is the same as being computable by some algorithm.

A structure in the model theoretic sense consists of a nonempty set D, called the domain, with relations and functions defined on it. The

following definition was first formulated in the 1960s by Mal'cev and Rabin independently.

Definition 1.2. A *computable structure* is a structure such that the domain is a computable set $D \subseteq \mathbb{N}$, and the functions and relations of the structure are computable. A countable structure S is called *computably presentable* if some computable structure W is isomorphic to it. In this context we call W a *computable copy* of S.

Example 1.3. For each $k \geq 1$, the group $GL_k(\mathbb{Q})$ is computably presentable. To obtain a computable copy, one fixes an algorithmic encoding of the rational $k \times k$ matrices by natural numbers, and lets the domain D be the computable set of numbers that encode a matrix with nonzero determinant. Since the encoding is algorithmic, the domain and the matrix operations are computable.

1.3. **Computable structures: the uncountable case.** In the field of computable analysis (for detail see e.g. Pauly [18] or Schröder [21]), to define computability for an uncountable structure, one begins by representing all the elements by "names", which are infinite objects simple enough to be accessible to computation of oracle Turing machines. Names usually are elements of the set $[T]$ of paths on some computable subtree T of $\mathbb{N}^*$ (the tree of strings with natural number entries). For instance, a standard name of a real number r is a path coding a sequence of rationals $\langle q_n \rangle_{n \in \mathbb{N}}$ such that $|q_n - q_{n+1}| \leq 2^{-n}$ and $\lim_n q_n = r$.

Via Turing machines with tapes that hold the input, one can define computability of functions and relations on $[T]$. One requires that the functions and relations of the uncountable structure are computable on the names. This defines computability on spaces relevant to computable analysis; for instance, one can define that a function on $\mathbb{R}$ is computable. Since each totally disconnected Polish space is homeomorphic to $[T]$ for some subtree T of $\mathbb{N}^*$, there is no need to distinguish between names and objects in our setting. An *ad hoc* way to define computability often works for particular classes of uncountable structures: impose algorithmic constraints on the definition of the class.

An example is the definition of when a profinite group G is computable due to Smith [22] and la Roche [10]: $G = \varprojlim_i (A_i, \psi_i)$ for a computable diagram $(A_i, \psi_i)_{i \in \mathbb{N}}$ of finite groups and epimorphisms $\psi_i \colon A_i \to A_{i-1}$ $(i > 0)$.

We now discuss the questions posed at the beginning in more detail.

1.4. **Computable presentations of t.d.l.c. groups.** We aim at a robust definition of the class of t.d.l.c. groups with a computable presentation. We want this class to have good algorithmic closure properties, and also ask that our definition extend the existing definitions for discrete, and for profinite groups. We provide two types of computable

presentations, which will turn out to be equivalent: a t.d.l.c. group has a computable presentation of one type iff it has one of the other type.

Computable Baire presentations. One asks that the domain of G is what we call an effectively σ-compact subtree of $\mathbb{N}^*$ (the tree of strings with natural number entries), and the operations are computable in the sense of oracle Turing machines. Baire presentations appear to be the simplest and most elegant notion of computable presentation for general totally disconnected Polish groups. However, computable Baire presentations are hard to study because the domain is usually uncountable.

Computable presentations via a meet groupoid. We introduce an algebraic structure $\mathcal{W}(G)$ on the countable set of compact open cosets in G, together with $\emptyset$. This structure is a partially ordered groupoid, with the usual set inclusion, and multiplication of a left coset of a subgroup U with a right coset of U (which is a coset). The intersection of two compact open cosets is such a coset itself, unless it is empty, so we have a meet semilattice. A computable presentation of G via meet groupoids is a computable copy of the meet groupoid of G such that the index function on compact open subgroups, namely $U, V \mapsto |U : U \cap V|$, is also computable.

1.5. **Which t.d.l.c. groups G have computable presentations?** Discrete groups, as well as profinite groups, have a computable presentation as t.d.l.c. groups if and only if they have one in the previously established sense from the 1960s and 1980s, reviewed in Section 1.2 above. We provide numerous examples of computable presentations for t.d.l.c. groups outside these two classes. For $(\mathbb{Q}_p, +)$ we use meet groupoid presentations. For $Aut(T_d)$ and $\mathrm{SL}_n(\mathbb{Q}_p)$ we use Baire presentations.

It can be difficult to determine whether a particular t.d.l.c. group has a computable presentation. Nonetheless, our thesis is that *all* "natural" groups that are considered in the field of t.d.l.c. groups have computable presentations. An interesting testing ground for this thesis is given by Neretin's groups $\mathcal{N}_d$ of almost automorphisms of T_d, for $d \geq 3$; see for instance [6].

1.6. **Associated computable objects.** Recall that to a t.d.l.c. group G we associate its meet groupoid $\mathcal{W}(G)$, an algebraic structure on its compact open cosets. If G is given by a computable Baire presentation, then we construct a copy $\mathcal{W}$ of the meet groupoid $\mathcal{W}(G)$ that is computable in a strong sense, essentially including the condition that some (and hence any) rational valued Haar measure on G is computable when restricted to a function $\mathcal{W} \to \mathbb{R}$. We will show in Corollary 5.9 that the left, and hence also the right, action of G on $\mathcal{W}$ is computable. We conclude that the modular function on G is computable. If G is compactly generated, for each Cayley-Abels graph one can determine

a computable copy, and any two copies of this type are computably quasi-isometric (Theorem 6.2). Intuitively, this means that the large-scale structure of G is a computable invariant.

Assertions that the scale function is computable have been made for particular t.d.l.c. groups in works such as Glöckner [4] and Willis [27, Section 6]; see the survey Willis [28]. In these particular cases, it was generally clear what it means that one can compute the scale $s(g)$: provide an algorithm that shows it. One has to declare what kind input the algorithm takes; necessarily it has to be some approximation to g, as g ranges over a potentially uncountable domain. Our new framework allows us to give a precise meaning to the question whether the scale function is computable for a particular computable presentation of a t.d.l.c. group, thus also allowing for a precise negative answer. This appears reminiscent of the answer to Hilbert's 10th problem, which asked for an algorithm that decides whether a multivariate polynomial over $\mathbb{Z}$ has a zero. Only after a precise notion of computable function was introduced in the 1930s, it became possible to assert rigorously that no such algorithm exists; the final negative answer was given in 1970 by Y. Matyasevich [12] (also see [13]). We leave the following open; see further discussion below.

Question 1.4. *Given a computable presentation of a t.d.l.c. group G, is the scale function computable for this presentation?*

If the answer is in the negative, one can further ask whether for some computably presented G, the scale is non-computable for *each* of its computable presentations.

1.7. **Algorithmic versions of constructions that lead from t.d.l.c. groups to new t.d.l.c. groups.** Section 7 shows that the class of computably t.d.l.c. groups is closed under suitable algorithmic versions of many constructions that have been studied in the theory of t.d.l.c. groups. In particular, the constructions (1), (2), (3) and (6) described in Wesolek [24, Thm. 1.3] can be phrased algorithmically in such a way that they stay within the class of computably t.d.l.c. groups; this provides further evidence that our class is robust. These constructions are suitable versions, in our algorithmic topological setting, of

- passing to closed subgroups,
- taking group extensions via continuous actions,
- forming "local" direct products, and
- taking quotients by closed normal subgroups

(see [24, Section 2] for detail on these constructions). The algorithmic version of taking quotients (Theorem 7.5) is the most demanding; it uses extra insights from the proofs that the various forms of computable presentation are equivalent.

1.8. **When is a computable presentation unique?** Viewing a computable Baire presentation as a description, we are interested in the question whether such a description is unique, in the sense that between any two of them there is a computable isomorphism. Adapting terminology for countable structures going back to Mal'cev, we will call such a group *autostable*. If a t.d.l.c. group is autostable, then computation in the group can be seen as independent of its particular description. Criterion 8.2 reduces the problem of whether a t.d.l.c. group is autostable to the countable setting of meet groupoids.

2. Computability on paths of rooted trees

2.1. Computably σ-compact subtrees of $\mathbb{N}^*$.

Notation 2.1. Let $\mathbb{N}^*$ denote the set of strings with natural numbers as entries. We use letters σ, τ, ρ etc. for elements of $\mathbb{N}^*$. The set $\mathbb{N}^*$ can be seen as a directed tree: the empty string is the root, and the successor relation is given by appending a number at the end of a string. We write $\sigma \preceq \tau$ to denote that σ is an initial segment of τ, and $\sigma \prec \tau$ to denote that σ is a proper initial segment. We can also identify finite strings of length $n+1$ with partial functions $\mathbb{N} \to \mathbb{N}$ having finite support $\{0, \ldots, n\}$. We then write τ_i instead of $\tau(i)$. By $\max(\tau)$ we denote $\max\{\tau_i \colon i \leq n\}$. Let $h \colon \mathbb{N}^* \to \mathbb{N}$ be the canonical encoding given by $h(w) = \prod_{i<|w|} p_i^{w_i+1}$, where p_i is the i-th prime number.

Definition 2.2 (Strong indices for finite sets of strings)**.** For a finite set $u \subseteq \mathbb{N}^*$ let $n_u = \sum_{\eta \in u} 2^{h(\eta)}$; one says that n_u is the *strong index* for u.

We will usually identify a finite subset of $\mathbb{N}^*$ with its strong index. Unless otherwise mentioned, by a (directed) tree we mean a nonempty subset T of $\mathbb{N}^*$ such that $\sigma \in T$ and $\rho \prec \sigma$ implies $\rho \in T$. By $[T]$ one denotes the set of paths of a tree T. Our trees usually have no leaves, so $[T]$ is a closed set in Baire space $\mathbb{N}^{\mathbb{N}}$ equipped with the usual product topology. Note that $[T]$ is compact if and only if each level of T is finite, in other words T is finitely branching. For $\sigma \in T$ let

$$[\sigma]_T = \{X \in [T] \colon \sigma \prec X\}.$$

That is, $[\sigma]_T$ is the cone of paths on T that extend σ.

Definition 2.3 (computably σ-compact trees)**.** Let T be a computable subtree of $\mathbb{N}^*$ without leaves such that only the root can have infinitely many successors. We say that T is *computably σ-compact*, or c.s.c. for short, if there is a computable binary function H such that, if $\rho \in T$ is a nonempty string, then $\rho(i) \leq H(\rho(0), i)$ for each $i < |\rho|$.

Given such a tree T, the compact open subsets of $[T]$ can be algorithmically encoded by natural numbers. The notation below will be used throughout.

Definition 2.4 (Code numbers for compact open sets)**.** Let T be a c.s.c. tree. For a finite set $u \subseteq T - \{\emptyset\}$, let

$$\mathcal{K}_u = \bigcup_{\eta \in u} [\eta]_T,$$

(note that this set is compact). By a *code number* for a compact open set $\mathcal{K} \subseteq [T]$ we mean the strong index for a finite set u of strings such that $\mathcal{K} = \mathcal{K}_u$.

Such a code number is not unique (unless $\mathcal{K}$ is empty). So we will need to distinguish between the actual compact open set, and any of its code numbers. So one can decide, given $u \in \mathbb{N}$ as an input, whether u is a code number. Clearly, each compact open subset of $[T]$ is of the form $\mathcal{K}_u$ for some u.

The following lemma shows that the basic set-theoretic relations and operations are decidable for sets of the form $\mathcal{K}_u$, similar to the case of finite subsets of $\mathbb{N}$.

Lemma 2.5. Let T be a c.s.c. tree. Given code numbers u, w,

(i) one can compute code numbers for $\mathcal{K}_u \cup \mathcal{K}_w$ and $\mathcal{K}_u \cap \mathcal{K}_w$;
(ii) one can decide whether $\mathcal{K}_u \subseteq \mathcal{K}_w$. In particular, one can, given a code number $u \in \mathbb{N}$, compute the minimal code number $u^* \in \mathbb{N}$ such that $\mathcal{K}_{u^*} = \mathcal{K}_u$.

Proof. (i) The case of union is trivial. For the intersection operation, it suffices to consider the case that u and w are singletons. For strings $\alpha, \beta \in T$, one has $[\alpha]_T \cap [\beta]_T = \emptyset$ if α, β are incompatbile, and otherwise $[\alpha]_T \cap [\beta]_T = [\gamma]_T$ where γ is the longest common initial segment of α, β.
(ii) Let H be a computable binary function as in Definition 2.3. It suffices to consider the case that u is a singleton. Suppose that $\alpha \in T - \{\emptyset\}$. The algorithm to decide whether $[\alpha]_T \subseteq \mathcal{K}_w$ is as follows. Let N be the maximum length of a string in w. Answer "yes" if for each $\beta \succeq \alpha$ of length N such that $\beta(k) \leq H(\alpha, k)$ for each $k < N$, there is $\gamma \in w$ such that $\gamma \preceq \beta$. Otherwise, answer "no". □

Definition 2.6. Given a c.s.c. tree T, let E_T denote the set of *minimal* code numbers for compact open subsets of $[T]$. By the foregoing lemma, E_T is decidable.

2.2. **Computable functions on the set of paths of computable trees.** Most of the content of this subsection can either be seen as a special case of known results in abstract computable topology, or can be derived from such results.

Let T be a computable subtree of $\mathbb{N}^*$ without leaves. To define that a function which takes arguments from the potentially uncountable domain $[T]$ is computable, one descends to the countable domain of strings on T, where the usual computability notions work. The first definition, Def. 2.7 below, will apply when we show in Corollary 6.1

that the modular function on a computable presentation of a t.d.l.c. group is computable.

Definition 2.7. (1) A function $\Phi : [T] \times \mathbb{N} \to \mathbb{N}$ is computable if there is an oracle Turing machine as follows. Given $f \in [T]$ and $w \in \mathbb{N}$, when it has the list of the values $f(0), f(1), f(2), \ldots$ written on the oracle tape, with sufficiently many queries of the type "what is $f(q)$?" it can determine the value $\Phi(f, w)$.

(2) A function $\Phi\colon [T] \to [\mathbb{N}^*]$ is computable if and only if the function $\widetilde{\Phi} : [T] \times \mathbb{N} \to \mathbb{N}$ given by $\widetilde{\Phi}(g, n) = \Phi(g)(n)$ is computable.

(3) Similarly, one defines that $\Phi\colon [T] \times [S] \to [\mathbb{N}^*]$ is computable, using a TM with two oracle tapes.

Example 2.8. Let $T = \mathbb{N}^*$. The function $\Phi(f, n) = \sum_{i=0}^{n} f(i)$ is computable. The oracle TM with f written on the oracle tape queries the values of $f(i)$ for $i = 0, \ldots, n$ one by one and adds them.

Note that in general our functions will only be defined on $[T]$, not on all of $[\mathbb{N}^*]$. Thus, the oracle TM will only return an answer if the oracle f is in $[T]$.

For proofs of the following see [15, Section 6].

Lemma 2.9. Suppose that K and S are computable trees without leaves. Suppose further that there is a computable function H such that $\sigma(i) < H(i)$ for each $\sigma \in K$ and $i < |\sigma|$. Let $\Phi\colon [K] \to [S]$ be computable via an oracle TM M.

(i) There is a computable function γ as follows: in a computation of $\Phi(f, n)$, M only needs queries up to $\gamma(n)$.

(ii) If Φ is a bijection then Φ^{-1} is computable, via a partial computable function that is obtained uniformly in K, H, S and P_Φ.

Intuitively, the function g in (i) computes the "δ" in the definition of uniform continuity from the "ε": if $\delta = 1/n$ we have $\varepsilon = 1/g(n)$.

Lemma 2.10. Let T, S be c.s.c. trees. Suppose a function $\Phi\colon [T] \to [S]$ is computable via a partial computable function P_Φ. Given code numbers u, w, one can decide whether $\Phi(\mathcal{K}^T_u) \subseteq \mathcal{K}^S_w$.

To prove Example 3.2 below, we will need a criterion on whether, given a computable subtree S of a c.s.c. tree T (where S potentially has leaves), the maximally pruned subtree of S with the same set of paths is computable.

Proposition 2.11. *Let T be a c.s.c. tree. Let S be a computable subtree of T, and suppose that there is a uniformly computable dense sequence $(f_i)_{i\in\mathbb{N}}$ in $[S]$. Then the tree $\widetilde{S} = \{\sigma\colon [\sigma]_S \neq \emptyset\}$ is decidable. (It follows that $\widetilde{S}$ is c.s.c. Of course, $[\widetilde{S}] = [S]$.)*

Proof. Given a string $\sigma \in T$, if $\sigma = \emptyset$ then $\sigma \in \widetilde{S}$. Assuming $\sigma \neq \emptyset$, we can compute the least $t \in \mathbb{N}$ such that $\sigma \prec f_t$, or $\rho \notin S$ for each $\rho \in T$ of length t such that $\rho \succeq \sigma$; the latter condition can be decided

by the hypothesis that T is c.s.c. Clearly $\sigma \in \widetilde{S}$ iff the former condition holds. □

3. Defining computably t.d.l.c. groups via Baire presentations

Each totally disconnected Polish space X is homeomorphic to $[T]$ for some tree $T \subseteq \mathbb{N}^*$; see [7, I.7.8]. Clearly X is locally compact iff for each $f \in [T]$ there is an n such that the tree above $f\restriction_n$ is finitely branching; we can then assume that only the root can be infinitely branching. This suggests to work, in the algorithmic setting, with a domain of the presentation that has the form $[T]$ for a computably σ-compact tree T, and require that the group operations on $[T]$ be computable according to Definition 2.7. The same approach would work for other types of algebraic structure defined on $[T]$ for a computably σ-compact tree T, e.g. computably t.d.l.c. rings.

Definition 3.1. A *computable Baire presentation* is a topological group of the form $H = ([T], \mathit{Mult}, \mathit{Inv})$ such that

(1) T is computably σ-compact as defined in 2.3;
(2) Mult: $[T] \times [T] \to [T]$ and Inv: $[T] \to [T]$ are computable.

We say that a t.d.l.c. group G is *computably t.d.l.c.* (via a Baire presentation) if $G \cong H$ for such a group H.

Example 3.2. Let p be a prime, and let $n \geq 2$. Let $\mathbb{Q}_p$ denote the rings of p-adic numbers. (i) The t.d.l.c. ring $\mathbb{Q}_p$ has a computable Baire presentation. (ii) The t.d.l.c. group $\mathrm{SL}_n(\mathbb{Q}_p)$ has a computable Baire presentation.

Proof. (i) Let Q be the tree of strings $\sigma \in \mathbb{N}^*$ such that all entries, except possibly the first, are among $\{0, \ldots, p-1\}$, and $r0 \not\preceq \sigma$ for each $r > 0$. We think of a string $r\sigma \in Q$ as denoting the rational $p^{-r}n_\sigma \in \mathbb{Z}[1/p]$, where n_σ is the number which has σ as a p-ary expansion, written in reverse order:

$$n_\sigma = \textstyle\sum_{i<|\sigma|} p^i \sigma(i).$$

We allow the case that σ ends in 0. The condition that $r0 \not\preceq \sigma$ for each $r > 0$ says that p does not divide n_σ.

For instance, let $p = 3$; then

$$(3, 1, 0, 2) \text{ denotes the rational } 3^{-3} \cdot (1 + 2 \cdot 9) = 19/27.$$

For the addition operation, consider an oracle Turing machine with two oracle tapes starting with notations $r\sigma$ and $s\tau$ of numbers $p^{-r}m$ and $p^{-s}n$. Say $r \leq s$. Then $p^{-r}m + p^{-s}n = p^{-s}(p^{s-r}m + n)$. Clearly the machine can output a string denoting $p^{-r}m + p^{-s}n$. To continue the example above, if the machine sees tapes starting with $(3, 1, 0, 2)$ and $(4, 1, 2, 0, 0)$, it will internally replace the first string by $(4, 0, 1, 0, 2)$, and then keep the leading 4 and carry out the addition modulo 3^4 of

the numbers 57 and 10 with base 3 expansions $(0,1,0,2)$ and $(1,2,0,0)$ respectively, resulting in $(4,1,0,1,2)$. (This corresponds to $19/27 + 10/81 = 67/81$.)

A similar argument works for multiplication. It is important that we allow improper expansions i.e. strings ending in zeros as in the example above, so that the operation of the machines is monotonic.

(ii) We now provide a computable Baire presentation $([T], \mathit{Mult}, \mathit{Inv})$ of $\mathrm{SL}_n(\mathbb{Q}_p)$. Let T be the computable tree that is an n^2-fold "power" of Q. More precisely, $T = \{\sigma \colon \forall i < n^2\, [\sigma^i \in Q]\}$, where σ^i is the string of entries of σ in positions of the form $kn^2 + i$ for some $k, i \in \mathbb{N}$. Note that T itself is not c.s.c. as nodes up to level $n^2 - 1$ are infinitely branching. However, we can assume it is by skipping the levels $1, \ldots, n^2 - 1$. Clearly, $[T]$ can be naturally identified with the matrix algebra $M_n(\mathbb{Q}_p)$. By the computability of the ring operations on $\mathbb{Q}_p$ as verified above, the matrix product is computable as a function $[T] \times [T] \to [T]$, and the function $\det \colon [T] \to [Q]$ is computable.

Basic computability theory shows that for any c.s.c. trees T and R, any computable path f of R, and any computable function $\Phi \colon [T] \to [R]$, there is a computable subtree S of T such that $[S]$ equals the pre-image $\Phi^{-1}(f)$. Applying this to the function $\det \colon [T] \to [Q]$ and the path $f = 01000\ldots$ that denotes $1 \in \mathbb{Q}_p$, we obtain a computable subtree S of T such that $[S]$ can be identified with $SL_n(\mathbb{Q}_p)$. Note that S could have dead ends. We fix this next:

It is well-known that $SL_n(\mathbb{Z}[1/p])$ is dense in $SL_n(\mathbb{Q}_p)$. This is a special case of strong approximation for algebraic groups (see [19, Ch. 7]), but can also be seen in an elementary way using Gaussian elimination. The paths on S corresponding to matrices in $SL_n(\mathbb{Z}[1/p])$ are precisely the ones that are 0 from some point on. Clearly there is a computable listing (f_i) of these paths. So by Proposition 2.11 we can replace S by a c.s.c. tree $\widetilde{S}$ such that $[\widetilde{S}] = [S]$.

To obtain a computable Baire presentation based on $\widetilde{S}$, note that matrix multiplication on $[\widetilde{S}]$ is computable as the restriction of matrix multiplication on $[T]$. To define the matrix inversion operation Inv, we use the fact that the inverse of a matrix with determinant 1 equals its adjugate matrix; the latter can be obtained by computing determinants on minors. □

4. Defining computably t.d.l.c. groups via meet groupoids

This section provides the detail for the second type (Type M) of computable presentations of t.d.l.c. groups described in Section 1.4.

4.1. **The meet groupoid of a t.d.l.c. group.**

Intuitively, the notion of a groupoid generalizes the notion of a group by allowing that the binary operation is partial. A groupoid is given by a domain $\mathcal{W}$ on

which a unary operation $(.)^{-1}$ and a partial binary operation, denoted by "$\cdot$", are defined. These operations satisfy the following conditions:

(a) associativity in the sense that $(A \cdot B) \cdot C = A \cdot (B \cdot C)$, with either both sides or no side defined (and so the parentheses can be omitted);
(b) $A \cdot A^{-1}$ and $A^{-1} \cdot A$ are always defined;
(c) if $A \cdot B$ is defined then $A \cdot B \cdot B^{-1} = A$ and $A^{-1} \cdot A \cdot B = B$.

It follows from (c) that a groupoid satisfies the left and right cancellation laws. One says that an element $U \in \mathcal{W}$ is *idempotent* if $U \cdot U = U$. Clearly this implies that $U = U \cdot U^{-1} = U^{-1} \cdot U$ and so $U = U^{-1}$ by cancellation. Conversely, by (c) every element of the form $A \cdot A^{-1}$ or $A^{-1} \cdot A$ is idempotent.

Definition 4.1. A *meet groupoid* is a groupoid $(\mathcal{W}, \cdot, (.)^{-1})$ that is also a meet semilattice $(\mathcal{W}, \cap, \emptyset)$ of which $\emptyset$ is the least element. Writing $A \subseteq B \Leftrightarrow A \cap B = A$ and letting the operation $\cdot$ have preference over $\cap$, it satisfies the conditions

(d) $\emptyset^{-1} = \emptyset = \emptyset \cdot \emptyset$, and $\emptyset \cdot A$ and $A \cdot \emptyset$ are undefined for each $A \neq \emptyset$,
(e) if U, V are idempotents such that $U, V \neq \emptyset$, then $U \cap V \neq \emptyset$,
(f) $A \subseteq B \Leftrightarrow A^{-1} \subseteq B^{-1}$, and
(g) if $A_i \cdot B_i$ are defined $(i = 0, 1)$ and $A_0 \cap A_1 \neq \emptyset \neq B_0 \cap B_1$, then

$$(A_0 \cap A_1) \cdot (B_0 \cap B_1) = A_0 \cdot B_0 \cap A_1 \cdot B_1.$$

From (g) it follows and that the groupoid operations are monotonic: if $A_i \cdot B_i$ are defined $(i = 0, 1)$ and $A_0 \subseteq A_1, B_0 \subseteq B_1$, then $A_0 \cdot B_0 \subseteq A_1 \cdot B_1$. Also, if U and V are idempotent, then so is $U \cap V$ (this can also be verified on the basis of (a)-(f) alone).

For meet groupoids $\mathcal{W}_0, \mathcal{W}_1$, a bijection $h \colon \mathcal{W}_0 \to \mathcal{W}_1$ is an *isomorphism* if it preserves the three operations. Given a meet groupoid $\mathcal{W}$, the letters A, B, C will range over general elements of $\mathcal{W}$, and the letters U, V, W will range over idempotents of $\mathcal{W}$.

We use set theoretic notation for the meet semilattice because for the motivating examples of meet groupoids the intersection symbol means the usual. Note that the intersection of two cosets is empty, or again a coset.

Definition 4.2. Let G be a t.d.l.c. group. We define a meet groupoid $\mathcal{W}(G)$. Its domain consists of the compact open cosets in G (i.e., cosets of compact open subgroups of G), as well as the empty set. We define $A \cdot B$ to be the usual product AB in case that $A = B = \emptyset$, or A is a left coset of a subgroup V and B is a right coset of V; otherwise $A \cdot B$ is undefined.

Fact 4.3. *$\mathcal{W}(G)$ is a meet groupoid with the groupoid operations $\cdot$ and $A \to A^{-1}$, and the usual intersection operation $\cap$.*

We will use the usual group theoretic terminology for elements of an abstract meet groupoid $\mathcal{W}$. If U is an idempotent of $\mathcal{W}$ we call U a

subgroup, if $AU = A$ we call A a *left coset* of U, and if $UB = B$ we call B a *right coset* of U. Based on the axioms, one can verify that if $U \subseteq V$ for subgroups U, V, then the map $A \mapsto A^{-1}$ induces a bijection between the left cosets and the right cosets of U contained in V.

We note that $\mathcal{W}(G)$ satisfies the axioms of inductive groupoids defined in Lawson [11, page 109]. See [2, Section 4] for more on an axiomatic approach to meet groupoids.

Remark 4.4. It is well-known [5] that one can view groupoids as small categories in which every morphism has an inverse. The elements of the groupoid are the morphisms of the category. The idempotent morphisms correspond to the objects of the category. One has $A\colon U \to V$ where $U = A \cdot A^{-1}$ and $V = A^{-1} \cdot A$. Thus, in $\mathcal{W}(G)$, $A\colon U \to V$ means that A is a right coset of U and a left coset of V.

The idea to study appropriate Polish groups via an algebraic structure on their open cosets is due to Katrin Tent, and first appeared in [8]. This idea was further elaborated in a paper by Nies, Schlicht and Tent on the complexity of the isomorphism problem for oligomorphic groups [17]. There, approximation structures are used that are given by the ternary relation "$AB \subseteq C$", where A, B, C are certain open cosets. They are called "coarse groups". In the present work, it will be important that we have explicit access to the combination of the groupoid and the meet semilattice structures (which coarse groups don't provide). Coarse groups are too "coarse" an algebraic structure to analyse algorithmic aspects of t.d.l.c. groups.

4.2. **Second definition of computably t.d.l.c. groups.**

Definition 4.5. A meet groupoid $\mathcal{W}$ is called *Haar computable* if

(a) its domain is a computable subset D of $\mathbb{N}$;
(b) the groupoid and meet operations are computable in the sense of Definition 1.1; in particular, the relation $\{\langle x, y\rangle\colon\ x, y \in S \wedge x \cdot y \text{ is defined}\}$ is computable;
(c) the partial function with domain contained in $D \times D$ sending a pair of subgroups $U, V \in \mathcal{W}$ to $|U : U \cap V|$ is computable.

Here $|U\colon U \cap V|$ is defined abstractly as the number of left, or equivalently right, cosets of the nonzero idempotent $U \cap V$ contained in U; we require implicitly that this number is always finite. Note that by (b), the partial order induced by the meet semilattice structure of $\mathcal{W}$ is computable. Also, (b) implies that being a subgroup is decidable when viewed as a property of elements of the domain S; this is used in (c). The condition (c) corresponds to the computable bound H required in Definition 2.3. For ease of reading we will say that $n \in D$ *denotes* a coset A, rather than saying that n "is" a coset.

Definition 4.6 (Computably t.d.l.c. groups via meet groupoids)**.** Let G be a t.d.l.c. group. We say that G is *computably t.d.l.c.* via a meet groupoid if $\mathcal{W}(G)$ has a Haar computable copy $\mathcal{W}$. In this context, we call $\mathcal{W}$ a computable presentation of G (in the sense of meet groupoids).

Remark 4.7. In this setting, Condition (c) of Definition 4.5 is equivalent to saying that every Haar measure μ on G that assigns a rational number to some compact open subgroup (and hence is rational-valued) is computable on $\mathcal{W}$, in the sense that the function assigning to a compact open coset A the rational $\mu(A)$ is computable. Consider left Haar measures, say. First suppose that (c) holds. Given A, compute the subgroup V such that $A = A \cdot V$, i.e., A is a left coset of V. Compute $W = U \cap V$. We have $\mu(A) = \mu(V) = \mu(U) \cdot |V : W|/|U : W|$.

Conversely, if the Haar measure is computable on $\mathcal{W}$, then (c) holds because $|U\colon V| = \mu(U)/\mu(V)$.

For discrete groups, the condition (c) can be dropped, as the proof of the following shows.

Example 4.8. A discrete group G is computably t.d.l.c. via a meet groupoid $\Leftrightarrow$ G has a computable copy in the usual sense of Definition 1.2.

Proof. For the implication $\Leftarrow$, we may assume that G itself is computable; in particular, we may assume that its domain is a computable subset of $\mathbb{N}$. Each compact coset in G is finite, and hence can be represented by a strong index for a finite set of natural numbers. Since the group operations are computable on the domain, this implies that the meet groupoid of G has a computable copy. It is then trivially Haar computable.

For the implication $\Rightarrow$, let $\mathcal{W}$ be a Haar computable copy of $\mathcal{W}(G)$. Since G is discrete, $\mathcal{W}$ contains a least subgroup U. The set of left cosets of U is computable, and forms a group with the groupoid and inverse operations. This yields the required computable copy of G. □

By $\mathbb{Q}_p$ we denote the additive group of the p-adics. By the usual definition of semidirect products ([20, p. 27]), $\mathbb{Z} \ltimes \mathbb{Q}_p$ is the group defined on the Cartesian product $\mathbb{Z} \times \mathbb{Q}_p$ via the binary operation $\langle z_1, \alpha_1\rangle \cdot \langle z_2, \alpha_2\rangle = \langle z_1 + z_2, p^{z_2}\alpha_1 + \alpha_2\rangle$. This turns $\mathbb{Z} \ltimes \mathbb{Q}_p$ into a topological group with the product topology.

Example 4.9. For any prime p, the additive group $\mathbb{Q}_p$ and the group $\mathbb{Z} \ltimes \mathbb{Q}_p$ are computably t.d.l.c. via a meet groupoid.

Proof. We begin with the additive group $\mathbb{Q}_p$. Note that its open proper subgroups are of the form $U_r := p^r\mathbb{Z}_p$ for some $r \in \mathbb{Z}$. Let C_{p^∞} denote the Prüfer group $\mathbb{Z}[1/p]/\mathbb{Z}$, where $\mathbb{Z}[1/p] = \{zp^{-k}\colon\ z \in \mathbb{Z} \wedge k \in \mathbb{N}\}$. For each r there is a canonical epimorphism $\pi_r\colon \mathbb{Q}_p \to C_{p^\infty}$ with kernel U_r: if $\alpha = \sum_{i=-n}^{\infty} s_i p^i$ where $0 \leq s_i < p$, $n \in \mathbb{N}$, we have

$$\pi_r(\alpha) = \mathbb{Z} + \textstyle\sum_{i=-n}^{r-1} s_i p^{i-r};$$

here an empty sum is interpreted as 0. (Informally, $\pi_r(\alpha)$ is obtained by taking the "tail" of α from the position $r-1$ onwards to the last position, and shifting it in order to represent an element of C_{p^∞}.) So

each compact open coset in $\mathbb{Q}_p$ can be uniquely written in the form $D_{r,a} = \pi_r^{-1}(a)$ for some $r \in \mathbb{Z}$ and $a \in C_{p^\infty}$. The domain $S \subseteq \mathbb{N}$ of the Haar computable copy $\mathcal{W}$ of $\mathcal{W}(\mathbb{Q}_p)$ consists of natural numbers canonically encoding such pairs $\langle r, a\rangle$. They will be identified with the cosets they denote.

The groupoid operations are computable because we have $D_{r,a}^{-1} = D_{r,-a}$, and $D_{r,a} \cdot D_{s,b} = D_{r,a+b}$ if $r = s$, and undefined otherwise. It is easy to check that $D_{r,a} \subseteq D_{s,b}$ iff $r \geq s$ and $p^{r-s}a = b$. So the inclusion relation is decidable. We have $D_{r,a} \cap D_{s,b} = \emptyset$ unless one of the sets is contained in the other, so the meet operation is computable. Finally, for $r \leq s$, we have $|U_r : U_s| = p^{s-r}$ which is computable.

Next, let $G = \mathbb{Z} \ltimes \mathbb{Q}_p$; we build a Haar computable copy $\mathcal{V}$ of $\mathcal{W}(G)$. We will extend the listing $(D_{r,a})_{r\in\mathbb{Z},a\in C_{p^\infty}}$ of compact open cosets in $\mathbb{Q}_p$ given above. For each compact open subgroup of G, the projection onto $\mathbb{Z}$ is compact open, and hence the trivial group. So the only compact open subgroups of G are of the form U_r. Let $g \in G$ be the generator of $\mathbb{Z}$ such that $g^{-1}\alpha g = p\alpha$ for each $\alpha \in \mathbb{Q}_p$ (where $\mathbb{Z}$ and $\mathbb{Q}_p$ are thought of as canonically embedded into G). Each compact open coset of G has a unique form $g^z D_{r,a}$ for some $z \in \mathbb{Z}$. Formally speaking, the domain of the computable copy of $\mathcal{W}(G)$ consists of natural numbers encoding the triples $\langle z, r, a\rangle$ corresponding to such cosets; as before they will be identified with the cosets they denote.

To show that the groupoid and meet operations are computable, note that we have $gD_{r,a} = D_{r-1,a}g$ for each $r \in \mathbb{Z}, a \in C_{p^\infty}$, and hence $g^z D_{r,a} = D_{r-z,a}g^z$ for each $z \in \mathbb{Z}$. Given two cosets $g^v D_{r,a}$ and $g^w D_{s,b} = D_{s-w,b}g^w$, their composition is defined iff $r = s - w$, in which case the result is $g^{v+w}D_{s,a+b}$. The inverse of $g^z D_{r,a}$ is $D_{r,-a}g^{-z} = g^{-z}D_{r-z,-a}$.

To decide the inclusion relation, note that we have $g^z D_{r,a} \subseteq g^w D_{s,b}$ iff $z = w$ and $D_{r,a} \subseteq D_{s,b}$, and otherwise, they are disjoint. Using this one can show that the meet operation is computable (by an argument that works in any computable meet groupoid $\mathcal{V}$): if $A_0, A_1 \in \mathcal{V}$, $A_i \colon U_i \to V_i$, and A_0, A_1 are not disjoint, then $A_0 \cap A_1$ is the unique $C \in \mathcal{V}$ such that $C \colon U_0 \cap U_1 \to V_0 \cap V_1$ and $C \subseteq A_0, A_1$. Since $\mathcal{W}$ satisfies Condition (c) in Definition 4.5, and $\mathcal{V}$ has no subgroups beyond the ones present in W, we conclude that $\mathcal{V}$ is Haar computable. □

5. Equivalence of the two types of computable presentations

We show that a t.d.l.c. group G has a computable presentation in the sense of Def. 3.1 iff G has a computable presentation in the sense of Def. 4.6.

We need some preliminaries. For strings $\sigma_0, \sigma_1 \in \mathbb{N}^*$ of the same length n, let $\sigma_0 \otimes \sigma_1$ denote the string τ of that length such that $\tau(k) = \langle \sigma_0(k), \sigma_1(k)\rangle$ for each $k < n$ (where $\langle ., .\rangle$ is a computable pairing function, such as Cantor's).

Lemma 5.1. Let G be computably t.d.l.c. via a computable Baire presentation $([T], \mathit{Mult}, \mathit{Inv})$. Recall the set E_T of minimal code numbers for compact open sets from Definition 2.6.
(i) There is a computable function $I\colon E_T \to E_T$ such that for each $u \in E_T$, one has $\mathcal{K}_{I(u)} = (\mathcal{K}_u)^{-1}$.
(ii) For $u, v, w \in E_T$ one can decide whether $\mathcal{K}_u\mathcal{K}_v \subseteq \mathcal{K}_w$.

Proof. (i) By Lemma 2.10 one can decide whether $\mathcal{K}_u \subseteq (\mathcal{K}_w)^{-1}$. The equality $\mathcal{K}_u = (\mathcal{K}_w)^{-1}$ is equivalent to $\mathcal{K}_u \subseteq (\mathcal{K}_w)^{-1} \wedge \mathcal{K}_w \subseteq (\mathcal{K}_u)^{-1}$. So one lets $I(u)$ be the least index v such that this equality holds.
(ii) Let $\widetilde{T}$ be the tree of initial segments of strings of the form $\sigma_0 \otimes \sigma_1$, where $\sigma_0, \sigma_1 \in T$ have the same length. Then $\widetilde{T}$ is a c.s.c. tree, $[\widetilde{T}]$ is naturally homeomorphic to $[T] \times [T]$, and Mult can be seen as a computable function $[\widetilde{T}] \to [T]$. Now one applies Lemma 2.10. □

We will also need a computable presentation of the topological group of permutations of $\mathbb{N}$ based on a subtree of $\mathbb{N}^*$. Define a computable tree without leaves by

$$\mathit{Tree}(S_\infty) = \{\sigma \otimes \tau\colon \sigma, \tau \in \mathbb{N}^* \wedge \\ \sigma, \tau \text{ are 1-1 } \wedge\ \sigma(\tau(k)) = k\ \wedge\ \tau(\sigma(i)) = i \text{ whenever defined}\}.$$

A string $\sigma \otimes \tau \in \mathit{Tree}(S_\infty)$ gives rise to a finite injection $\alpha_{\sigma\otimes\tau}$ on $\mathbb{N}$, defined by

$$\alpha_{\sigma\otimes\tau}(r) = s \text{ iff } \sigma(r) = s\ \vee\ \tau(s) = r. \tag{1}$$

The paths of $\mathrm{Tree}(S_\infty)$ can be viewed as the permutations of $\mathbb{N}$, paired with their inverses:

$$[\mathit{Tree}(S_\infty)] = \{f \otimes f^{-1}\colon\ f \text{ is permutation of } \mathbb{N}\}.$$

The group operations on $\mathit{Tree}(S_\infty)$ are computable: we have

$$\begin{aligned} (f_0 \otimes f_1)^{-1} &= f_1 \otimes f_0 \\ (f_0 \otimes f_1) \cdot (g_0 \otimes g_1) &= (f_0 \circ g_0) \otimes (g_1 \circ f_1). \end{aligned}$$

For a closed subgroup $\widetilde{G}$ of S_∞, we write

$$\mathit{Tree}(\widetilde{G}) = \{\sigma \in \mathit{Tree}(S_\infty)\colon\ [\sigma]_{\mathit{Tree}(S_\infty)} \cap \widetilde{G} \neq \emptyset\}.$$

Note that this is a subtree of $\mathit{Tree}(S_\infty)$ without leaves. We say that $\widetilde{G}$ is computable if $\mathit{Tree}(\widetilde{G})$ is computable.

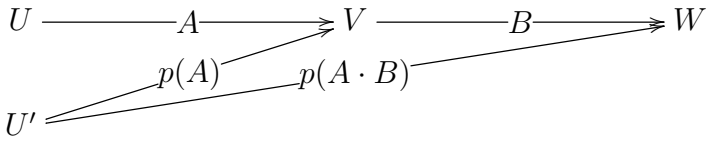
Theorem 5.2.
A group G is computably t.d.l.c. via a Baire presentation (Def. 3.1) $\Leftrightarrow$ G is computably t.d.l.c. via a meet groupoid (Def. 4.6).
From a presentation of G of one type, one can uniformly obtain a presentation of G of the other type.

Proof. $\Leftarrow$: (This is the harder implication - if you don't want to read it skip to Page 18.)

We begin by defining an operator that, for Haar computable meet groupoids, is dual to the operation of sending G to a computable copy of $\mathcal{W}(G)$ obtained above.

Definition 5.3. Given a meet groupoid $\mathcal{W}$ with domain $\mathbb{N}$, let $\widetilde{G} = \mathcal{G}_{\mathrm{comp}}(\mathcal{W})$ be the closed subgroup of S_∞ consisting of elements p that preserve the meet operation of $\mathcal{W}$, and satisfy $p(A) \cdot B = p(A \cdot B)$ whenever $A \cdot B$ is defined.

Recall that the elements of S_∞ are not actually permutations, but paths on $\mathit{Tree}(S_\infty)$ encoding pairs consisting of a permutation and its inverse. However, if $p \in \mathit{Tree}(\widetilde{G})$, and $A \in \mathcal{W}$ is denoted by i, we will suggestively write $p(A)$ for the element of $\mathcal{W}$ denoted by the first component of the pair of natural numbers encoded by $p(i)$. We note that for each subgroup $U \in \mathcal{W}(G)$, the set $B = p(U)$ satisfies $B \cdot U = p(U) \cdot U = p(U \cdot U) = B$, and hence is a left coset of U. The following diagram displays the condition in the definition above in category terms as a commutative diagram.



Now suppose that $\mathcal{W}$ is as in Definition 4.6. Recall the convention that all t.d.l.c. groups are infinite. So the domain of $\mathcal{W}$ equals $\mathbb{N}$, and there is an isomorphism of meet groupoids $\mathcal{W} \to \mathcal{W}(G)$, which below we will use to identify $\mathcal{W}$ and $\mathcal{W}(G)$. Define a group homomorphism $\Phi\colon G \to \widetilde{G}$ by letting $\Phi(g)$ be the element of S_∞ corresponding to the left action of g, i.e. $A \mapsto gA$ where $A \in \mathcal{W}(G)$. Note that Φ is injective because the compact open subgroups form a neighbourhood basis of 1: if $g \neq 1$ then $g \notin U$ for some compact open subgroup U, so that $\Phi(g)(U) \neq U$.

Claim 5.4. $\Phi\colon G \cong \widetilde{G}$.

To show that Φ is onto, let $p \in \widetilde{G}$. Since

$$\{p(U)\colon U \in \mathcal{W}(G) \text{ is a subgroup}\}$$

is a filter on $\mathcal{W}(G)$ containing a compact set, there is an element g in its intersection. Then $\Phi(g) = p$: recall that for each subgroup $U \in \mathcal{W}(G)$, the set $B = p(U)$ is a left coset of U, and hence equals gU. So, if A is a right coset of U, then $p(A) = p(U \cdot A) = B \cdot A = gA$.

To show that Φ is continuous at 1 (and hence continuous), note that a basis of neighbourhoods of the identity in $\widetilde{G}$ is given by the open sets

$$\{p \in \widetilde{G}\colon \forall i \leq n\, [p(A_i) = A_i]\},$$

where $A_1, \dots, A_n \in \mathcal{W}(G)$. Given such a set, suppose A_i is a right coset of U_i, and let $U = \bigcap U_i$. If $g \in U$ then $gA_i = A_i$ for each i.

The open mapping theorem for Hausdorff groups says that every surjective continuous homomorphism from a σ-compact group (such as a t.d.l.c. group with a countable basis of the topology) onto a Baire group is open. So Φ is open. This verifies the claim.

Using the assumption that $\mathcal{W}$ is Haar computable, we now show that $\mathit{Tree}(\widetilde{G})$ is c.s.c. as in Definition 2.3. The following claim will be used to show that $\mathit{Tree}(\widetilde{G})$ is computable.

Claim 5.5. *A finite injection α on $\mathbb{N}$ can be extended to some $p \in \widetilde{G}$ $\Leftrightarrow$ $B \cdot A^{-1}$ is defined whenever $\alpha(A) = B$, and*

$$\bigcap \{B \cdot A^{-1} \colon \alpha(A) = B\} \neq \emptyset.$$

For right to left, let g be an element of the intersection. Then $gA = B \cdot A^{-1} \cdot A = B = \alpha(A)$ for each $A \in \mathrm{dom}(\alpha)$.

For left to right, suppose $p \in \widetilde{G}$ extends α. By Claim 5.4, there is $g \in G$ such that $p = \Phi(g)$. Then $gA = p(A) = B$ for each A, B such that $\alpha(A) = B$. Such A, B are right cosets of the same subgroup. Hence $B \cdot A^{-1}$ is defined, and clearly g is in the intersection. This establishes the claim.

By (1),

$$S = \{\sigma \otimes \tau \colon \alpha_{\sigma\otimes\tau} \text{ can be extended to some } p \in \widetilde{G}\}$$

is a computable subtree of $\mathit{Tree}(S_\infty)$ without leaves, and $\widetilde{G} = [S]$. Hence $S = \mathit{Tree}(\widetilde{G})$.

Claim 5.7 below will verify that $S = \mathit{Tree}(\widetilde{G})$ is a c.s.c. tree as defined in 2.3. The following lemma does the main work. Informally it says that given some subgroup $U \in \mathcal{W}$, if one declares that $p \in \widetilde{G}$ has a value $L \in \mathcal{W}$ at U, then one can compute for any $F \in \mathcal{W}$ the finite set of possible values of p at F.

Lemma 5.6 (Effectively finite suborbits)**.** Suppose that $U \in \mathcal{W}$ is a subgroup and L is a left coset of U. Let $F \in \mathcal{W}$. One can uniformly in U, L and F compute a strong index for the finite set $\mathcal{L} = \{p(F) \colon p \in [S] \wedge p(U) = L\}$.

To see this, first one computes $V = F^{-1} \cdot F$, so that F is a right coset of the subgroup V. Next one computes $k = |U \colon U \cap V|$, the number of left cosets of $U \cap V$ in U. Note that

$$\mathcal{L}_0 = \{p(U \cap V) \colon p \in [S] \wedge p(U) = L\}$$

is the set of left cosets of $U \cap V$ contained in L. Clearly this set has size k. By searching $\mathcal{W}$ until all of its elements have appeared, one can compute a strong index for this set. Next one computes a strong index

for the set $\mathcal{L}_1$ of left cosets D of V such that $C \subseteq D$ for some $C \in \mathcal{L}_0$ (this uses that given C one can compute D). Finally one outputs a strong index for the set $\{D \cdot F\colon\ D \in \mathcal{L}_1\}$, which equals $\mathcal{L}$. This shows the lemma.

We make the assumption that 0 denotes a subgroup U in $\mathcal{W}$. This does not affect the uniformity statement of the theorem: otherwise we can search $\mathcal{W}$ for the least n such that n is a subgroup, and then work with a new copy of $\mathcal{W}(G)$ where 0 and n are swapped.

Claim 5.7. *There is a computable binary function H such that, if $\rho \in S$, then $\rho(i) \leq H(\rho(0), i)$ for each $i < |\rho|$.*

Let F be the coset denoted by k. Let $\rho(0) = \langle a_0, a_1\rangle$ and let L_r be the coset denoted by a_r, $r = 0, 1$. Applying Lemma 5.6 to U, L_r, F, one can compute $H(\sigma, i)$ as the greatest pair $\langle b_0, b_1\rangle$ such that b_r denotes an element of $\{p(F)\colon\ p \in [S] \wedge p(U) = L_r\}$ for $r = 0, 1$.

$\Rightarrow$: We build a Haar computable copy $\mathcal{W}$ of the meet groupoid $\mathcal{W}(G)$ as in Definition 4.6. By Lemma 5.1, one can decide whether $u \in E_T$ is the code number of a subgroup (Definition 2.4). Furthermore, one can decide whether $B = \mathcal{K}_v$ is a left coset of a subgroup $U = \mathcal{K}_u$: this holds iff $BU \subseteq B$ and $BB^{-1} \subseteq U$, and the latter two conditions are decidable by Lemma 5.1. Similarly, one can decide whether B is a right coset of U.

It follows that the set $\{u \in E_T\colon\ \mathcal{K}_u \text{ is a coset}\}$ can be obtained via an existential quantification over a computable binary relation (in other words, V is recursively enumerable). Hence, by a basic fact of computability theory, there is computable 1-1 function θ defined on an initial segment of $\mathbb{N} - \{0\}$ such that the range of θ equals this set. Write $A_n = \mathcal{K}_{\theta(n)}$ for $n > 0$, and $A_0 = \emptyset$.

The domain of $\mathcal{W}$ is all of $\mathbb{N}$. By Lemma 2.5 the intersection operation on $\mathcal{W}$ is computable, i.e., there is a computable binary function c on $\mathbb{N}$ such that $A_{c(n,k)} = A_n \cap A_k$. Next, given $n, k \in \mathbb{N} - \{0\}$ one can decide whether A_n is a right coset of the same subgroup that A_k is a left coset of. In that case, one can compute the number r such that $A_r = A_n \cdot A_k$: one uses that A_r is the unique coset C such that

(a) $A_nA_k \subseteq C$, and
(b) C is a right coset of the same subgroup that A_k is a right coset of.

For subgroups U, V, one can compute $|U\colon U \cap V|$ by finding in $\mathcal{W}$ further and further distinct left cosets of $U \cap V$ contained in U, until their union reaches U. The latter condition is decidable. □

Definition 5.8. Given a computable Baire presentation G, by $\mathcal{W}_{\mathrm{comp}}(G)$ we denote the computable copy of $\mathcal{W}(G)$ with domain $\mathbb{N}$ obtained in the proof above.

Corollary 5.9. *In this context, the left and right actions* $[T] \times \mathbb{N} \to \mathbb{N}$, *given by* $(g, A) \mapsto gA$ *and* $(g, A) \mapsto Ag$, *are computable.*

Proof. For the left action, we use an oracle Turing machine that has as an oracle a path g on $[T]$, and as an input an $A \in \mathcal{W}$. If A is a left coset of a subgroup V, it outputs the left coset B of V such that it can find a string $\sigma \prec g$ with $[\sigma]_T A \subseteq B$.

For the right action use that $Ag = (g^{-1}A^{-1})^{-1}$ and inversion is computable both in G and in $\mathcal{W}_{\mathrm{comp}}(G)$. □

Recall from the introduction that $\mathrm{Aut}(T_d)$ is the group of automorphism of the undirected tree T_d where each vertex has degree d.

Example 5.10. Let $d \geq 3$. The t.d.l.c. group $G = \mathrm{Aut}(T_d)$ has a computable Baire presentation.

Proof. Via an effective encoding of the vertices of T_d by the natural numbers, we can view G itself as a closed subgroup of S_∞. A finite injection α on T_d can be extended to an automorphism of T_d iff it preserves distances, which is a decidable condition. Each $\eta \in \mathit{Tree}(S_\infty)$ corresponds to an injection on T_d via (1). So we can decide whether $[\eta]_{\mathit{Tree}(G)} = [\eta]_{\mathit{Tree}(S_\infty)} \cap G \neq \emptyset$. Clearly $[\eta]_{\mathit{Tree}(G)}$ is compact for *every* such nonempty string η.

To see that $\mathit{Tree}(G)$ is c.s.c., note that if $\sigma \in \mathit{Tree}(G)$ maps $x \in T_d$ to $y \in T_d$, then every extension $\eta \in \mathit{Tree}(G)$ of σ maps elements in T_d at distance n from x to elements in T_d at distance n from y, and conversely. This yields a computable bound $H(\sigma, i)$ as required in (3) of Def. 2.3. □

6. Algorithmic properties of objects associated with a t.d.l.c. group

6.1. **The modular function is computable.** In Subsection 1.6 we discussed the modular function $\Delta\colon G \to \mathbb{R}^+$. As an application of Corollary 5.9, we show that for any computable presentation, the modular function is computable.

Corollary 6.1. *Let G be computably t.d.l.c. via a Baire presentation $([T], \mathit{Mult}, \mathit{Inv})$. Then the modular function $\Delta\colon [T] \to \mathbb{Q}^+$ is computable.*

Proof. Let $V \in \mathcal{W}$ be any subgroup. Given $g \in [T]$, using Corollary 5.9 compute $A = gV$. Compute $U \in \mathcal{W}$ such that A is a right coset of U, and hence $A = Ug$. For any left Haar measure μ on G, we have

$$\Delta(g) = \mu(A)/\mu(U) = \mu(V)/\mu(U).$$

By Remark 4.7 we can choose μ computable; so this suffices to determine $\Delta(g)$. □

6.2. **Cayley-Abels graphs are computable.** Let G be a t.d.l.c. group that is compactly generated, i.e., algebraically generated by a compact subset. Then there is a compact open subgroup U, and a set $S = \{s_1, \dots, s_k\} \subseteq G$ such that $S = S^{-1}$ and $U \cup S$ algebraically generates G. The *Cayley-Abels graph*

$$\Gamma_{S,U} = (V_{S,U}, E_{S,U})$$

of G is given as follows. The vertex set $V_{S,U}$ is the set $L(U)$ of left cosets of U, and the edge relation is

$$E_{S,U} = \{\langle gU, gsU\rangle \colon\ g \in G, s \in S\}.$$

Some background and original references are given in Section 5 of [28]. For more detailed background see Part 4 of [25], or [9, Section 2]. If G is discrete (and hence finitely generated), then $\Gamma_{S,\{1\}}$ is the usual Cayley graph for the generating set S. Any two Cayley-Abels graphs of G is are quasi-isometric. See [9, Def. 3] or [25] for the formal definition.

Theorem 6.2. *Suppose that G is computably t.d.l.c. and compactly generated.*

(i) Each Cayley-Abels graph $\Gamma_{S,U}$ of G has a computable copy $\mathcal{L}$.
(ii) If $\Gamma_{T,V}$ is another Cayley-Abels graph obtained as above, then $\Gamma_{S,U}$ and $\Gamma_{T,V}$ are computably quasi-isometric.

Proof. (i) For the domain of the computable copy $\mathcal{L}$, we take the computable set of left cosets of U. We show that the edge relation is first-order definable from the parameters in such a way that it can be verified to be computable as well.

Let $V_i = C_i \cdot C_i^{-1}$ so that C_i is a right coset of V_i. Let $V = U \cap \bigcap_{1\le i\le k} V_i$. To first-order define E_Γ in $\mathcal{W}$ with the given parameters, the idea is to replace the elements g in the definition of E_Γ by left cosets P of V, since they are sufficiently accurate approximations to g. It is easy to verify that $\langle A, B\rangle \in E_\Gamma \Leftrightarrow$

$$\exists i \le k \exists P \in L(V) \exists Q \in L(V_i)\, [P \subseteq A \wedge P \subseteq Q \wedge B = Q \cdot C_i],$$

where $L(U)$ denotes the set of left cosets of a subgroup U: For the implication "$\Leftarrow$", let $g \in P$; then we have $A = gU$ and $B = gs_iU$. For the implication "$\Rightarrow$", given $A = gU$ and $B = gs_iU$, let $P \in L(V)$ such that $g \in P$.

We verify that the edge relation E_Γ is computable. Since $\mathcal{W}$ is Haar computable, by the usual enumeration argument we can obtain a strong index for the set of left cosets of V contained in A. Given P in this set and $i \le k$, the left coset $Q = Q_{P,i}$ of V_i in the expression above is unique and can be determined effectively. So we can test whether $\langle A, B\rangle \in E_\Gamma$ by trying all P and all $i \le k$ and checking whether $B = Q_{P,i} \cdot C_i$.

(ii) (Sketch) First suppose that $V \subseteq U$. There is a computable map $\psi \colon L(U) \to L(V)$ such that $\psi(A) \subseteq A$. The proof of [9, Thm. 2^+] shows that $\psi \colon \Gamma_{S,U} \to \Gamma_{T,V}$ is a quasi-isometry. In the general case, let

$R \subseteq G$ be a finite symmetric set such that $(U \cap V) \cup R$ algebraically generates G. There are computable quasi-isometries $\phi\colon \Gamma_{S,U} \to \Gamma_{R,U\cap V}$ and $\psi\colon \Gamma_{T,V} \to \Gamma_{R,U\cap V}$ as above. There is a computable quasi-isometry $\theta : \Gamma_{R,U\cap V} \to \Gamma_{T,V}$: given a vertex $y \in L(U \cap V)$, let $x = \theta(y)$ be a vertex in $L(V)$ such that $\psi(x)$ is at distance at most c from y, where c is a constant for ψ as above. Then $\theta \circ \phi$ is a quasi-isometry as required. □

6.3. **Algorithmic properties of the scale function.** The scale function $s\colon G \to \mathbb{N}^+$ for a t.d.l.c. group G was introduced by Willis [26]. Recall that for a compact open subgroup V of G and an element $g \in G$ one defines $m(g, V) = |V^g\colon V \cap V^g|$, and

$$s(g) = \min\{m(g, V)\colon V \text{ is a compact open subgroup}\}.$$

Willis proved that the scale function is continuous, where $\mathbb{N}^+$ carries the discrete topology. He introduced the relation that a compact open subgroup V is *tidy for* g, and showed that this condition is equivalent to being minimizing for g in the sense that $s(g) = m(g, V)$. Möller [16] used graph theoretic methods to show that V is minimizing for g if and only if $m(g^k, V) = m(g, V)^k$ for each $k \in \mathbb{N}$. He also derived the "spectral radius formula": for any compact open subgroup U, one has $s(g) = \lim_k m(g^k, U)^{1/k}$.

The following example is well-known ([28, Example 2]); we include it to show that our framework is adequate as a general background for case-based approaches to computability for t.d.l.c. groups used in earlier works.

Example 6.3 (with Stephan Tornier)**.** For $d \geq 3$, the scale function on $Aut(T_d)$ in the computable presentation of Example 5.10 is computable.

Proof. An automorphism g of T_d has exactly one of three types (see [3]):

(1) g fixes a vertex v: then $s(g) = 1$ because g preserves the stabilizer of v, which is a compact open subgroup.
(2) g inverts an edge: then $s(g) = 1$ because g preserves the set-wise stabilizer of the set of endpoints of this edge.
(3) g translates along a geodesic (a subset of T_d that is a homogeneous tree of degree 2): then $s(g) = (d-1)^\ell$ where ℓ is the length. To see this, for $\ell = 1$ one uses as a minimizing subgroup the compact open subgroup of automorphisms that fix two given adjacent vertices on the axis. For $\ell > 1$ one uses that $s(r^k) = s(r)^k$ for each k and $r \in Aut(T_d)$; see again [26].

The oracle Turing machine, with a path corresponding to $g \in Aut(T_d)$ as an oracle, searches in parallel for a witness to (1), a witness to (2), and a sufficiently long piece of the axis in (3) so that the shift becomes "visible". It then outputs the corresponding value of the scale. □

For the rest of this section, fix a computable Baire presentation $([T], Mult, Inv)$ of a t.d.l.c. group G as in Def. 3.1. Let $\mathcal{W} = \mathcal{W}_{\mathrm{comp}}(G)$ be the Haar computable copy of $\mathcal{W}(G)$ given by Definition 5.8. Recall that the domain of $\mathcal{W}$ is $\mathbb{N}$. Via $\mathcal{W}$ we can identify compact open cosets of G with natural numbers. The following is immediate from Corollary 5.9.

Fact 6.4. *The function $m\colon [T] \times \mathbb{N} \to \mathbb{N}$ (defined to be 0 if the second argument is not a subgroup) is computable.*

It is of interest to study whether the scale function, seen as a function $s\colon [T] \to \mathbb{N}$, is computable in the sense of Definition 2.7. We note that neither Möller's spectral radius formula, nor the tidying procedure of Willis (see again [28]) allow to compute the scale in our sense.

The scale is computable if and only if one can algorithmically decide whether a subgroup is mimimizing:

Fact 6.5. *The scale function on $[T]$ is computable $\Leftrightarrow$ the following function Φ is computable in the sense of Def. 2.7: if $g \in [T]$ and V is a compact open subgroup of G, then $\Phi(g, V) = 1$ if V is minimizing for g; otherwise $\Phi(g, V) = 0$.*

Proof. $\Rightarrow$: An oracle Turing machine with oracle g searches for the first V that is minimizing for g, and outputs $m(g, V)$.
$\Leftarrow$: For oracle g, given input V check whether $m(g, V) = s(g)$. If so output 1, otherwise 0. □

We next provide a fact restricting the complexity of the scale function. We say that a function $\Psi : [T] \to \mathbb{N}$ is *computably approximable from above* if there is a computable function $\Theta : [T] \times \mathbb{N} \to \mathbb{N}$ such that $\Theta(f, r) \geq \Theta(f, r + 1)$ for each $f \in [T], r \in \mathbb{N}$, and

$$\Psi(f) = k \text{ iff } \lim_r \Theta(f, r) = k.$$

Fact 6.6. *The scale function is computably approximable from above.*

Proof. Let $\Theta(f, r)$ be the minimum value of $m(f, s)$ over all $s \leq r$. □

7. Closure properties of the class of computably t.d.l.c. groups

All computable presentations in this section will be Baire presentations (see Definition 3.1), and we will usually view a t.d.l.c. group G concretely as a computable Baire presentation. Extending the previous notation in the setting of closed subgroups of S_∞, by $Tree(G)$ we denote the c.s.c. tree underlying this computable Baire presentation. The following is immediate.

Fact 7.1 (Computable closed subgroups)**.** *Let G be a computably t.d.l.c. group. Let H be a closed subgroup of G (so that $Tree(H)$ is a subtree*

of Tree(G)). Then H is computably t.d.l.c. via the Baire presentation based on the Tree(H) (which is c.s.c.), with the operations of G restricted to H.

For instance, consider the closed subgroups $U(F)$ of $Aut(T_d)$, where $d \geq 3$ and F is a subgroup of S_d, introduced by Burger and Mozes [1]. By Example 5.10 together with the preceding fact, each group $U(F)$ is computably t.d.l.c.

For another example, consider the computable Baire presentation of $\mathrm{SL}_2(\mathbb{Q}_p)$ given by Example 3.2. Let S be the c.s.c. subtree of T whose paths describe matrices of the form $\begin{pmatrix} r & 0 \\ 0 & s \end{pmatrix}$ (so that $s = r^{-1}$). This yields a computable Baire presentation of the group $(\mathbb{Q}_p^*, \cdot)$.

Example 7.2. For each prime p and $n \geq 2$, the group $\mathrm{GL}_n(\mathbb{Q}_p)$ is computably t.d.l.c.

Proof. We employ the embedding $F\colon GL_n(\mathbb{Q}_p) \to SL_{n+1}(\mathbb{Q}_p)$ which extends a matrix A to the matrix B where the new row and new column vanish except for the diagonal element (which necessarily equals $(\det A)^{-1}$). Clearly there is a c.s.c. subtree S of the c.s.c. subtree of T in Example 3.2 for $n+1$ such that $[S] = \mathrm{range}(F)$. Now we apply Fact 7.1. □

A further construction staying within the class of t.d.l.c. groups is the semidirect product based on a continuous action. In the effective setting, we use actions that are computable in the sense of Section 2.2. For computable actions in the more general context of Polish groups see [14].

Proposition 7.3 (Closure under computable semidirect products)**.** *Let G, H be computably t.d.l.c. groups. Suppose $\Phi\colon G \times H \to H$ is a computable function that specifies an action of G on H via topological automorphisms. Then the topological semidirect product $L = G \ltimes_\Phi H$ is computably t.d.l.c.*

Proof. Let T be the tree obtained by pairing corresponding components of strings of the same length from the trees of G and H, i.e.

$$T = \{\sigma \otimes \tau\colon\ \sigma \in Tree(G) \wedge \tau \in Tree(H)\}.$$

It is clear that T is a c.s.c. tree. Via the natural bijection

$$[T] \to [Tree(G)] \times [Tree(H)],$$

one can write elements of L in the form $\langle g, h\rangle$ where $g \in [Tree(G)]$ and $h \in [Tree(H)]$.

By the standard definition of semidirect product ([20, p. 27]), writing the operations for G and H in the usual group theoretic way, we have

$$\begin{aligned} Mult(\langle g_1, h_1\rangle, \langle g_2, h_2\rangle) &= \langle g_1 g_2, \Phi(g_2, h_1) h_2\rangle \\ Inv(\langle g, h\rangle) &= \langle g^{-1}, (\Phi(g^{-1}, h))^{-1}\rangle. \end{aligned}$$

This shows that *Mult* and *Inv* are computable, and hence yields a computable Baire presentation $([T], \mathit{Mult}, \mathit{Inv})$ for L. □

The next two closure properties are proved in the underlying paper [15, Section 11]. For local direct products see Wesolek [24, Def. 2.3].

Proposition 7.4 (Prop 11.5 in [15]). *Let $(G_i)_{i\in\mathbb{N}^+}$ be computably t.d.l.c. groups uniformly in i, and for each i let U_i be a compact open subgroup of G_i, uniformly in i. Then $G = \bigoplus_{i\in\mathbb{N}^+}(G_i, U_i)$ is computably t.d.l.c.*

The hardest one is the closure under quotients by computable closed normal subgroups.

Theorem 7.5 (Thm. 11.11 in [15]). *Let G be computably t.d.l.c. Let N be a closed normal subgroup of G such that $\mathit{Tree}(N)$ is a computable subtree of $\mathit{Tree}(G)$. Then G/N is computably t.d.l.c.*

Example 7.6. For each prime p and each $n \geq 2$, the group $\mathrm{PGL}_n(\mathbb{Q}_p)$ is computably t.d.l.c.

Proof. In Example 7.2 we obtained a computable Baire presentation $(T, \mathit{Mult}, \mathit{Inv})$ of $\mathrm{GL}_n(\mathbb{Q}_p)$. In this presentation, the centre N of $\mathrm{GL}_n(\mathbb{Q}_p)$ is given by the diagonal $(n+1)\times(n+1)$ matrices such that the first n entries of the diagonal agree. So clearly $\mathit{Tree}(N)$ is a computable subtree of the tree S in Example 7.2. Hence we can apply Theorem 7.5. □

8. Uniqueness of computable presentation

As discussed in Subsection 1.8, a countable structure is called autostable if it has a computable copy, and all its computable copies are computably isomorphic. We adapt this notion to the present setting.

Definition 8.1. A computably t.d.l.c. group G is called *autostable* if for any two computable Baire presentations of G, based on trees $T, S \subseteq \mathbb{N}^*$, there is a computable group homeomorphism $\Psi\colon [T] \to [S]$. Note that Ψ^{-1} is also computable by [15, Cor 9.3].

We now provide a criterion for autostability, and show its usefulness through various examples.

Criterion 8.2. *A computably t.d.l.c. group G is autostable $\Leftrightarrow$ any two Haar computable copies of its meet groupoid $\mathcal{W}(G)$ are computably isomorphic.*

We will only apply the implication "$\Leftarrow$". However, the converse implication is interesting on its own right because it shows that our notion of autostability is independent of whether we use computable Baire presentation, or computable presentations based on meet groupoids.

Proof. See [15], proof of Criterion 12.2. □

Theorem 8.3. *The computably t.d.l.c. groups $\mathbb{Q}_p$ and $\mathbb{Z} \ltimes \mathbb{Q}_p$ are autostable.*

Proof. In Example 4.9 we obtained a Haar computable copy $\mathcal{W}$ of the meet groupoid $\mathcal{W}(\mathbb{Q}_p)$. Recall that the elements of $\mathcal{W}$ are given as cosets $D_{r,a} = \pi_r^{-1}(a)$ where $r \in \mathbb{Z}$, $\pi_r \colon \mathbb{Z}_p \to C_{p^\infty}$ is the canonical projection with kernel $U_r = p^r\mathbb{Z}_p$, and $a \in C_{p^\infty}$.

By the criterion above, it suffices to show that any Haar computable copy $\widetilde{\mathcal{W}}$ of $\mathcal{W}(\mathbb{Q}_p)$ is computably isomorphic to $\mathcal{W}$. By hypothesis on $\widetilde{\mathcal{W}}$ there is an isomorphism $\Gamma \colon \mathcal{W} \to \widetilde{\mathcal{W}}$. Let $\widetilde{U}_r = \Gamma(U_r)$ for $r \in \mathbb{Z}$. We will construct a *computable* isomorphism $\Delta \colon \mathcal{W} \to \widetilde{\mathcal{W}}$ which agrees with Γ on the set $\{U_r \colon r \in \mathbb{Z}\}$. First we show that from r one can compute the subgroup $\widetilde{U}_r \in \widetilde{\mathcal{W}}$.

(a) If $\widetilde{U}_r$ has been determined, $r \geq 0$, compute $\widetilde{U}_{r+1}$ by searching for the unique subgroup in $\widetilde{\mathcal{W}}$ that has index p in $\widetilde{U}_r$.
(b) If $\widetilde{U}_r$ has been determined, $r \leq 0$, compute $\widetilde{U}_{r-1}$ by searching for the unique subgroup in $\widetilde{\mathcal{W}}$ such that $\widetilde{U}_r$ has index p in it.

The shift homeomorphism $S \colon \mathbb{Q}_p \to \mathbb{Q}_p$ is defined by $S(x) = px$. Note that $B \to S(B)$ is an automorphism of the meet groupoid $\mathcal{W}$. Using the notation of Example 4.9 (recalled above), for each $\alpha \in \mathbb{Q}_p, r \in \mathbb{Z}$, one has $\pi_{r+1}(S(\alpha)) = \pi_r(\alpha)$, and hence for each $a \in C_{p^\infty}$,

$$S(D_{r,a}) = D_{r+1,a}. \tag{2}$$

We show that S is definable within $\mathcal{W}$ by an existential formula using subgroups U_r as parameters. Recall that given a meet groupoid $\mathcal{W}$, by $L(U)$ we denote the set of left cosets of a subgroup U. For $D \in L(U_r)$ we write D^k for $D \cdot \ldots \cdot D$ (with k factors), noting that this is defined, and in $L(U_r)$.

Claim 8.4. *Let $B \in L(U_r)$ and $C \in L(U_{r+1})$. Then*

$$C = S(B) \Leftrightarrow \exists D \in L(U_{r+1})\, [D \subseteq B \wedge D^p = C].$$

$\Leftarrow$: If $x \in C$ then $x = py$ for some $y \in B$, so $x \in S(B)$. So $C \subseteq S(B)$ and hence $C = S(B)$ given that $S(B) \in L(U_{r+1})$.
$\Rightarrow$: Let $x \in C$, so $x = S(y)$ for some $y \in B$. Let $y \in D$ where $D \in L(U_{r+1})$. Then $D \subseteq B$. Since $D^p \cap C \neq \emptyset$, these two (left) cosets of U_{r+1} coincide. This shows the claim.

We use this to show that the function $\widetilde{S} = \Gamma \circ S \circ \Gamma^{-1}$ defined on $\widetilde{\mathcal{W}}$ is computable. Since $\Gamma(U_r) = \widetilde{U}_r$, $(r \in \mathbb{Z})$, $\widetilde{S}$ satisfies the claim when replacing the U_r by the $\widetilde{U}_r$. Since the meet groupoid $\widetilde{\mathcal{W}}$ is computable, given $B \in \widetilde{\mathcal{W}}$, one can search $\widetilde{\mathcal{W}}$ for a witness $D \in L(\widetilde{U}_{r+1})$ as on the right hand side, and then output $C = \widetilde{S}(B)$. So the function $\widetilde{S}$ is computable.

We build the computable isomorphism $\Delta \colon \mathcal{W} \to \widetilde{\mathcal{W}}$ in four phases. The first three phases build a computable isomorphism $L(U_0) \to L(\widetilde{U}_0)$, where $L(\widetilde{U}_0) \subseteq \widetilde{\mathcal{W}}$ denotes the group of left cosets of $\widetilde{U}_0$. (This group is

isomorphic to C_{p^∞}, so this amounts to defining a computable isomorphism between two computable copies of C_{p^∞}.) The last phase extends this isomorphism to all of $\mathcal{W}$, using that $\widetilde{S}$ is an automorphism of $\widetilde{\mathcal{W}}$.

For $q \in \mathbb{Z}[1/p]$ we write $[q] = \mathbb{Z}+q \in C_{p^\infty}$. We define $\widetilde{D}_{r,[q]} = \Delta(D_{r,a})$ for $r \in \mathbb{Z}, q \in \mathbb{Z}[1/p]$

(a) Let $\widetilde{D}_{0,[p^{-1}]}$ be an element of order p in $L(\widetilde{U}_0)$.
(b) Recursively, for $m > 0$ let $\widetilde{D}_{0,[p^{-m}]}$ be an element of order p^m in $L(\widetilde{U}_0)$ such that $(\widetilde{D}_{0,[p^{-m}]})^p = \widetilde{D}_{0,[p^{-m+1}]}$.
(c) For $a = [kp^{-m}]$ where $0 \leq k < p^m$ and p does not divide k, let $\widetilde{D}_{0,a} = (\widetilde{D}_{0,[p^{-m}]})^k$.
(d) For $r \in \mathbb{Z} - \{0\}$ let $\widetilde{D}_{r,a} = \widetilde{S}^r(\widetilde{D}_{0,a})$.

One can easily verify that $\Delta\colon \mathcal{W} \to \widetilde{\mathcal{W}}$ is computable and preserves the meet groupoid operations. To verify that Δ is onto, let $B \in \widetilde{\mathcal{W}}$. We have $B \in L(\widetilde{U}_r)$ for some r. There is a least m such that $B = (\widetilde{D}_{r,[p^{-m}]})^k$ for some $k < p^m$. Then p does not divide k, so $B = \widetilde{D}_{r,[kp^{-m}]}$.

We next treat the case of $G = \mathbb{Z} \ltimes \mathbb{Q}_p$. Let $\mathcal{V}$ be the Haar computable copy of $\mathcal{W}(G)$ obtained in Example 4.9, and let $\widetilde{\mathcal{V}}$ be a further Haar computable copy of $\mathcal{W}(G)$. Using the notation of Example 4.9, let

$$E_{z,r,a} = g^z D_{r,a} \text{ for each } z, r \in \mathbb{Z}, a \in C_{p^\infty}.$$

We list some properties of these elements of $\mathcal{V}$ that will be needed shortly. Note that we can view $\mathcal{W}$ as embedded into $\mathcal{V}$ by identifying $\langle r, a\rangle$ with $\langle 0, r, a\rangle$. Also note that $E_{z,r,a}\colon U_{r-z} \to U_r$ (using the category notation discussed after Fact 4.3). Since $D_{r+1,a} \subseteq D_{r,pa}$, we have

$$E_{z,r+1,a} \subseteq E_{z,r,pa}. \tag{3}$$

Furthermore,

$$E_{z,r,0} = g^z U_r = U_{r+z} g^z = (g^{-z} U_{r-z})^{-1} = (E_{-z,r-z,0})^{-1}. \tag{4}$$

By hypothesis on $\widetilde{\mathcal{V}}$, there is a meet groupoid isomorphism $\overline{\Gamma}\colon \mathcal{V} \to \widetilde{\mathcal{V}}$. Since G has no compact open subgroups besides the ones present in $\mathcal{W}(\mathbb{Q}_p)$, the family $(\widetilde{U}_r)_{r\in\mathbb{Z}}$, where $\widetilde{U}_r = \overline{\Gamma}(U_r)$, is computable in $\widetilde{\mathcal{V}}$ by the same argument as before. The set of elements A of $\widetilde{\mathcal{V}}$ that are a left and a right coset of the same subgroup is computable by checking whether $A^{-1} \cdot A = A \cdot A^{-1}$. The operations of $\widetilde{\mathcal{V}}$ induce a Haar computable meet groupoid $\widetilde{\mathcal{W}}$ on this set. Clearly the restricted map $\Gamma = \overline{\Gamma} \mid \mathcal{W}$ is an isomorphism $\mathcal{W} \to \widetilde{\mathcal{W}}$. So by the case of $\mathbb{Q}_p$, there is a computable isomorphism $\Delta\colon \mathcal{W} \to \widetilde{\mathcal{W}}$.

We will extend Δ to a computable isomorphism $\overline{\Delta}\colon \mathcal{V} \to \widetilde{\mathcal{V}}$. The following summarizes the setting:

$$\begin{array}{ccc} \mathcal{V} & \xrightarrow{\overline{\Gamma},\overline{\Delta}} & \widetilde{\mathcal{V}} \\ \subseteq\uparrow & & \uparrow\subseteq \\ \mathcal{W} & \xrightarrow[\Gamma,\Delta]{} & \widetilde{\mathcal{W}} \end{array}$$

In five phases we define a computable family $\widetilde{E}_{z,r,a}$ $(z, r \in \mathbb{Z}, a \in C_{p^\infty})$, and then let $\overline{\Delta}(E_{z,r,a}) = \widetilde{E}_{z,r,a}$. As before write $\widetilde{D}_{r,a} = \Delta(D_{r,a})$.

(a) Let $\widetilde{E}_{0,r,a} = \widetilde{D}_{r,a}$. Choose $F_0 := \widetilde{E}_{-1,0,0}\colon \widetilde{U}_1 \to \widetilde{U}_0$

(b) compute $F_r := \widetilde{E}_{-1,r,0}\colon U_{r+1} \to U_r$ by recursion on $|r|$, where $r \in \mathbb{Z}$, in such a way that $\widetilde{F}_{r+1} \subseteq \widetilde{F}_r$ for each $r \in \mathbb{Z}$; this is possible by (3) and since $\mathcal{V} \cong \widetilde{\mathcal{V}}$ via $\overline{\Gamma}$.

(c) For $z < -1$, compute $\widetilde{E}_{z,r,0}\colon U_{r-z} \to U_r$ as follows:

$$\widetilde{E}_{z,r,0} = F_{r-z-1} \cdot F_{r-z-2} \cdot \ldots \cdot F_r.$$

(d) For $z > 0$ let $E_{z,r,0} = (\widetilde{E}_{-z,r-z,0})^{-1}$; this is enforced by (4).

(e) Let $\widetilde{E}_{z,r,a} = \widetilde{E}_{z,r,0} \cdot \widetilde{D}_{r,a}$.

One verifies that $\overline{\Delta}$ preserves the meet groupoid operations (we omit the formal detail). To show that $\overline{\Delta}$ is onto, suppose that $\widetilde{E} \in \mathcal{V}$ is given. Then $\widetilde{E} = \Gamma(E_{z,r,a})$ for some z, r, a. By (3) we may assume that $z < 0$. Then $E_{z,r,0} = \prod_{i=1}^{-z} E_{-1,r-z-i,0}$ as above. So, writing F_s for $\widetilde{E}_{-1,s,0}$, we have $\Gamma(E_{z,r,0}) = \prod_{i=1}^{-z} F_{r-z-i}\widetilde{D}_{r-z-i,a_i}$ for some $a_i \in C_{p^\infty}$.

Note that $\widetilde{S}(D) = F \cdot D \cdot F^{-1}$ for each $D \in L(\widetilde{U}_r) \cap \widetilde{\mathcal{W}}$ and $F\colon \widetilde{U}_{r+1} \to \widetilde{U}_r$. For, the analogous statement clearly holds in $\mathcal{V}$; then one uses that $\widetilde{S} = \Gamma \circ S \circ \Gamma^{-1}$, and that $\overline{\Gamma}\colon \mathcal{V} \to \widetilde{\mathcal{V}}$ is an isomorphism. Since $\widetilde{D}_{r+1,a} = \widetilde{S}(\widetilde{D}_{r,a})$, we may conclude that $\widetilde{D}_{r+1,a} \cdot F = F \cdot D_{r,a}$ for each such F. We can use these "quasi-commutation relations" to simplify the expression $\prod_{i=1}^{-z} F_{r-z-i}\widetilde{D}_{r-z-i,a_i}$ to $\widetilde{E}_{z,r,0}\widetilde{D}_{r,b}$ for some $b \in C_{p^\infty}$. Hence $\widetilde{E} = \widetilde{E}_{z,r,0}\widetilde{D}_{r,b}\widetilde{D}_{r,a}$. This shows that $\widetilde{E}$ is in the range of $\overline{\Delta}$, as required. □

References

[1] M. Burger and S. Mozes. Groups acting on trees: from local to global structure. *Publications Mathématiques de l'IHÉS*, 92:113–150, 2000.

[2] A. Nies (editor). Logic Blog 2020. Available at https://arxiv.org/pdf/2101.09508.pdf, 2020.

[3] A. Figà-Talamanca and C. Nebbia. *Harmonic Analysis and Representation Theory for Groups Acting on Homogenous Trees*, volume 162. Cambridge University Press, 1991.

[4] H. Glöckner. Scale functions on p-adic lie groups. *manuscripta mathematica*, 97(2):205–215, 1998.

[5] P. Higgins. *Categories and groupoids*. Van Nostrand Reinhold, 1971.

[6] C. Kapoudjian. Simplicity of Neretin's group of spheromorphisms. In *Annales de l'institut Fourier*, volume 49, pages 1225–1240, 1999.
[7] A. S. Kechris. *Classical descriptive set theory*, volume 156. Springer-Verlag New York, 1995.
[8] A. S. Kechris, A. Nies, and K. Tent. The complexity of topological group isomorphism. *The Journal of Symbolic Logic*, 83(3):1190–1203, 2018.
[9] B. Krön and R. Möller. Analogues of Cayley graphs for topological groups. *Mathematische Zeitschrift*, 258(3):637–675, 2008.
[10] P. La Roche. Effective Galois theory. *The Journal of Symbolic Logic*, 46(2):385–392, 1981.
[11] M. Lawson. *Inverse semigroups: the theory of partial symmetries*. World Scientific, 1998.
[12] Y. Matijasevic. Enumerable sets are diophantine (Russian). *Dokl. Akad. Nauk SSSR*, 191:279–282, 1970. Translation in Soviet Math Doklady, Vol 11, 1970.
[13] Y. Matijasevic. *Hilbert's Tenth Problem*. MIT press, Cambridge, 1993.
[14] A. Melnikov and A. Montalbán. Computable Polish group actions. *J. Symb. Log.*, 83(2):443–460, 2018.
[15] A. Melnikov and A. Nies. Computably totally disconnected locally compact groups. Available at arxiv.org/pdf/2204.09878.pdf., 2022.
[16] R. Möller. Structure theory of totally disconnected locally compact groups via graphs and permutations. *Canadian Journal of Mathematics*, 54(4):795–827, 2002.
[17] A. Nies, P. Schlicht, and K. Tent. Coarse groups, and the isomorphism problem for oligomorphic groups. *Journal of Mathematical Logic*, page 2150029, 2021.
[18] A. Pauly. On the topological aspects of the theory of represented spaces. *Computability*, 5(2):159–180, 2016.
[19] V. Platonov and A. Rapinchuk. *Algebraic groups and number theory*. Academic press, 1993.
[20] D. Robinson. *A course in the theory of groups*. Springer–Verlag, 1988.
[21] M. Schröder. Admissibly represented spaces and qcb-spaces. In *Handbook of Computability and Complexity in Analysis*, pages 305–346. Springer, 2021.
[22] R. Smith. Effective aspects of profinite groups. *The Journal of Symbolic Logic*, 46(04):851–863, 1981.
[23] D. Van Dantzig. Zur topologischen Algebra. iii. Brouwersche und Cantorsche Gruppen. *Compositio Mathematica*, 3:408–426, 1936.
[24] P. Wesolek. Elementary totally disconnected locally compact groups. *Proceedings of the London Mathematical Society*, 110(6):1387–1434, 2015.
[25] P. Wesolek. An introduction to totally disconnected locally compact groups. *Preprint of a book, people.math.binghamton.edu/wesolek/mathdocs/TDLC_Groups.pdf*, 2018.
[26] G. Willis. The structure of totally disconnected, locally compact groups. *Mathematische Annalen*, 300(1):341–363, 1994.
[27] G. Willis. Further properties of the scale function on a totally disconnected group. *Journal of Algebra*, 237(1):142–164, 2001.
[28] G. Willis. Computing the scale of an endomorphism of a totally disconnected locally compact group. *Axioms*, 6(4):27, 2017.

School of Computer Science, The University of Auckland, New Zealand
E-mail address: andre@cs.auckland.ac.nz