COMPUTING IN UNIPOTENT AND REDUCTIVE ALGEBRAIC GROUPS

ARJEH M. COHEN, SERGEI HALLER, AND SCOTT H. MURRAY

ABSTRACT. The unipotent groups are an important class of algebraic groups. We show that techniques used to compute with finitely generated nilpotent groups carry over to unipotent groups. We concentrate particularly on the maximal unipotent subgroup of a split reductive group and show how this improves computation in the reductive group itself.

1. INTRODUCTION

A linear algebraic group is *unipotent* if its elements are unipotent (ie, the elements only have eigenvalue one in every representation). Unipotent groups play a prominent role in the theory of algebraic groups. In this paper, we present algorithms for efficient element operations in unipotent groups. Since unipotent groups are nilpotent, we adapt methods used for computing in finitely generated nilpotent groups. A *PC group* provides a unique computer representation for the elements of nilpotent groups (see, for example, [HEO05]). *Collection* gives efficient algorithms for multiplication and inversion of group elements. We modify these concepts to work with a large class of unipotent groups defined over a field. This class contains all unipotent groups if the field has characteristic zero. It also contains the full unipotent subgroup of every split reductive group.

Algorithms for element operations in split reductive algebraic groups are given in [CMT04]. Computations in the unipotent subgroup make the largest single contribution to the time taken by these algorithms. This was the main impetus for the current paper, in which we prove:

Theorem 1.1. *Let $\mathbb{F}$ be a field with effective algorithms for the basic element operations. Let G be a split reductive algebraic group over $\mathbb{F}$ with rank n. Then there is a normal form for elements of $G(\mathbb{F})$. The word problem for elements in normal form requires $O(n^2)$ field operations, and multiplying or inverting them requires $O(n^3)$ field operations.*

This theorem is a great improvement over the analysis of [CMT04], where we proved that the operations are polynomial time, but did not compute the exponent. This result is optimal in the sense that the timings are asymptotically the same as the straightforward methods for matrices.

In Section 2, we construct *FC group schemes*, which give normal forms for elements of unipotent groups. We adapt two basic collection strategies in Section 3.

Date: November 1, 2006.

This paper was written during a stay of the second author at the Magma group at University of Sydney. The authors would like to thank Björn Assmann, John Cannon, Michael Harrison, Anthony Henderson, Bill Kantor, Allan Steel, and Donald E. Taylor for useful suggestions and discussions.

We have developed two new algorithms for the full unipotent subgroup of a split reductive algebraic group: The first is a new collection strategy called *collection from the outside* (Section 4). The second is a direct method for computing products and inverses in classical groups using standard representations (Section 5). Section 6 gives the asymptotic analysis and the proof of Theorem 1.1. Section 7 compares practical timings for the various methods considered and describes the default method used in Magma [BCP97] from version 2.13.

2. Unipotent groups and presentations

Throughout this paper, $\mathbb{F}$ is a field and $\mathbb{E}$ is a commutative unital algebra over $\mathbb{F}$. We assume that we have effective algorithms for the basic element operations in $\mathbb{F}$ and $\mathbb{E}$. We find it convenient to use the scheme-theoretic definition of algebraic groups. So an algebraic group defined over $\mathbb{F}$ is a functor from the category of all commutative unital algebras over $\mathbb{F}$ to the category of groups, satisfying the appropriate additional conditions [DG70, Wat79]. If G is an algebraic group, then $G(\mathbb{E})$ is an abstract group, called the *rational point group* of G over $\mathbb{E}$. For example, the additive group $\mathbb{G}_\mathrm{a}$ has rational point group $\mathbb{G}_\mathrm{a}(\mathbb{E}) = \mathbb{E}^+$.

We define an $\mathbb{F}$-*unipotent group* to be an algebraic group defined over $\mathbb{F}$ with a normal series in which every quotient is isomorphic to $\mathbb{G}_\mathrm{a}$. This is analogous to the definition of an $\mathbb{F}$-soluble algebraic group in [DG70].

Proposition 2.1. *Over a perfect field $\mathbb{F}$, every connected unipotent group is $\mathbb{F}$-unipotent. Over a field $\mathbb{F}$ of characteristic zero, every unipotent group is $\mathbb{F}$-unipotent.*

Proof. The first statement follows from [Ros57, Proposition 5, Corollary 2]. The second follows from the first using the standard fact that all unipotent groups are connected in characteristic zero. □

The group $\boldsymbol{\alpha}_p$, defined in [Wat79] over fields of characteristic $p > 0$, is unipotent but not $\mathbb{F}$-unipotent.

Let U be an $\mathbb{F}$-unipotent group. Fix a central series

$$U = U_1 > U_2 > \cdots > U_{N+1} = 1,$$

such that each U_r/U_{r+1} is $\mathbb{F}$-isomorphic to $\mathbb{G}_\mathrm{a}$. The projection $U_r \to U_r/U_{r+1} \cong \mathbb{G}_\mathrm{a}$ splits as an $\mathbb{F}$-morphism of schemes by [Spr98, Theorem 16.2.6]. Fix splitting maps $x_r : \mathbb{G}_\mathrm{a} \to U_r$. Clearly U is parametrised by N-dimensional affine space:

$$\mathbb{A}^N \to U, \quad (a_1, \dots, a_N) \mapsto x_1(a_1) \cdots x_N(a_N).$$

Multiplication and inversion in $U(\mathbb{E})$ are given by polynomials. To be precise

$$\prod_{r=1}^{N} x_r(a_r) \prod_{r=1}^{N} x_r(b_r) = \prod_{r=1}^{N} x_r(F_r(a_1, \dots, a_N, b_1, \dots, b_N)), \tag{1}$$

$$\left(\prod_{r=1}^{N} x_r(a_r)\right)^{-1} = \prod_{r=1}^{N} x_r(G_r(a_1, \dots, a_N)), \tag{2}$$

where all products are written in ascending order, each F_r is a polynomial in $2N$ indeterminates, and each G_r is a polynomial in N indeterminates. The F_r and G_r are called *Hall polynomials* [Hal69]. These polynomials have coefficients in $\mathbb{F}$, but tend to be very large and unwieldy. In order to do practical computations in U, we need a more concise description.

We now construct a presentation for the rational point group $U(\mathbb{E})$. Since the series $U(\mathbb{E}) = U_1(\mathbb{E}) > U_2(\mathbb{E}) > \cdots > U_N(\mathbb{E}) > U_{N+1}(\mathbb{E}) = 1$ is central, we have

$$\begin{array}{rcl} x_r(a)x_r(b) & \in & x_r(a+b)U_{r+1}(\mathbb{E}), \\ x_r(a)^{-1} & \in & x_r(-a)U_{r+1}(\mathbb{E}), \\ x_s(b)x_r(a) & \in & x_r(a)x_s(b)U_{s+1}(\mathbb{E}), \end{array}$$

for $a, b \in \mathbb{E}$ and $1 \leq r < s \leq N$. Hence we have relations

$$x_r(a)x_r(b) = x_r(a+b) \prod_{t=r+1}^{N} x_t(f_{rt}(a,b)), \tag{3}$$

$$x_r(a)^{-1} = x_r(-a) \prod_{t=r+1}^{N} x_t(g_{rt}(a)), \tag{4}$$

$$x_s(b)x_r(a) = x_r(a)x_s(b) \prod_{t=s+1}^{N} x_t(h_{rst}(a,b)), \tag{5}$$

where f_{rt}, g_{rt}, and h_{rst} are polynomials defined over $\mathbb{F}$. We note that many of these relations are redundant (including (4) for all r), but the extra relations are useful for computation.

Theorem 2.2. *Let U be an $\mathbb{F}$-unipotent group and let $\mathbb{E}$ be a commutative $\mathbb{F}$-algebra. Let $\tilde{U}(\mathbb{E})$ be the group with generators $x_r(a)$, for $a \in \mathbb{E}$, $r = 1, \ldots, N$, and relations (3), (4), and (5), for $a, b \in \mathbb{E}$, $1 \leq r < s \leq N$. Then the natural map $\tilde{U}(\mathbb{E}) \to U(\mathbb{E})$ is an isomorphism of abstract groups.*

Proof. Since the given relations hold in $U(\mathbb{E})$, the map is well defined. The map is onto because $U(\mathbb{E})$ is generated by the images of the generators of $\tilde{U}(\mathbb{E})$. Every element of $\tilde{U}(\mathbb{E})$ is a word with terms of the form $x_r(a)$ or $x_r(a)^{-1}$. This word can be *collected* into a product $\prod_{r=1}^{N} x_r(a_r)$. This is achieved by first eliminating all inverses using (4), then putting the terms in order by the subscripts using (5) and removing multiple terms with the same subscript using (3). If the words $\prod_{r=1}^{N} x_r(a_r)$ and $\prod_{r=1}^{N} x_r(b_r)$ are equal in $U(\mathbb{E})$, then $a_r = b_r$ for all r, and so these words are also equal in $\tilde{U}(\mathbb{E})$. Hence the map is injective and we are done. □

We say that the group scheme U is *presented by* $\tilde{U}$.

Now suppose we are given an arbitrary system of $\mathbb{F}$-polynomials $f_{rt}(a,b)$ and $g_{rt}(a,b)$, for $1 \leq r < t \leq N$; and $h_{rst}(a)$, for $1 \leq r < s < t \leq N$. Define the group functor $\tilde{U}$ by taking $\tilde{U}(\mathbb{E})$ to be the abstract group given by generators $x_r(a)$, for $a \in \mathbb{E}$, $r = 1, \ldots, N$, and relations (3), (4), and (5). We call $\tilde{U}$ an *FC group functor over* $\mathbb{F}$. FC stands for field-commutator, since the relations involve field operations and commutators, just as the PC presentation of a nilpotent group involves powers and commutators. We call $\tilde{U}$ *consistent* if the map

$$\mathbb{E}^N \to \tilde{U}(\mathbb{E}), \quad (a_1, \ldots, a_N) \mapsto x_1(a_1) \cdots x_N(a_N)$$

is injective for every commutative $\mathbb{F}$-algebra $\mathbb{E}$. Theorem 2.2 implies that every $\mathbb{F}$-unipotent group is presented by a consistent FC group functor. We now prove the converse:

Theorem 2.3. *Every consistent FC group functor defined over $\mathbb{F}$ is an $\mathbb{F}$-unipotent group.*

Proof. Let $\tilde{U}$ be the consistent FC group functor. Ignoring the multiplication, we can consider $\tilde{U}$ to be the N-dimensional affine scheme. Using collection, as in the proof of Theorem 2.2, we can find polynomials F_r and G_r such that equations (1) and (2) are satisfied in $\tilde{U}(\mathbb{E})$. So $\tilde{U}$ is an $\mathbb{F}$-algebraic group scheme, since F_r and G_r are clearly defined over $\mathbb{F}$. Finally define algebraic subgroups $U_r = \prod_{k=r}^{N} \mathrm{im}(x_r)$. These give a normal series for U in which every quotient is isomorphic to $\mathbb{G}_\mathrm{a}$, and so $\tilde{U}$ is an $\mathbb{F}$-unipotent group. □

Let U be an $\mathbb{F}$-unipotent group. Suppose the projection $U_r \to \mathbb{G}_\mathrm{a}$ splits as a homomorphism of $\mathbb{F}$-group schemes, not just as a morphism of $\mathbb{F}$-schemes. Then we can take $x_r : \mathbb{G}_\mathrm{a} \to U_r$ to be a homomorphism, and so replace (3) and (4) by

$$x_r(a)x_r(b) = x_r(a+b). \tag{6}$$

It follows immediately that

$$x_r(a)^{-1} = x_r(-a), \tag{7}$$

and so all the polynomials f_{rt} and g_{rt} are zero. If x_r is a homomorphism for $r = 1, \dots, N$, we call the corresponding FC group functor *split*.

Theorem 2.4. *If $\mathbb{F}$ is a field of characteristic zero, then every unipotent group defined over $\mathbb{F}$ is presented by a split FC group functor over $\mathbb{F}$.*

Proof. Since $\mathbb{F}$ has characteristic zero, every unipotent group U defined over $\mathbb{F}$ is $\mathbb{F}$-unipotent by Proposition 2.1. By induction, we can assume that U/U_N is presented by a split FC group functor. Fix maps $y_r : \mathbb{G}_\mathrm{a} \to U/U_N$, for $r = 1, \dots, N-1$, defining this functor. By [Ser88, Proposition VII.8], $\mathrm{Ext}(\mathbb{G}_\mathrm{a}, \mathbb{G}_\mathrm{a}) = 0$. Hence $\mathrm{Ext}(U/U_N, \mathbb{G}_\mathrm{a}) = 0$, by repeated application of the long exact sequence for $\mathrm{Hom}(\circ, \mathbb{G}_\mathrm{a})$. So there exists a homomorphism $y : U/U_N \to U$ splitting the projection $U \to U/U_N$. Define $x_r = y \circ y_r$ for $r = 1, \dots, N-1$. Take x_N to be the $\mathbb{F}$-injection $\mathbb{G}_\mathrm{a} \cong U_N \to U$. These maps clearly define a split FC group functor presenting U. □

If $\mathbb{F}$ has positive characteristic, then the Witt-vector groups [Ser88] provide examples of $\mathbb{F}$-unipotent groups which cannot be presented by split FC group functors. The full unipotent subgroup of a split reductive group is always presented by a split FC group functor, as we show in Proposition 4.1 below.

3. Collection and symbolic collection

We now extend some of the standard collection strategies for PC groups to FC group functors. The precise order in which the relations are applied has a huge impact on the speed of collection. Many strategies have been suggested, and we have not attempted to extend them all to FC group functors. We have implemented two fundamental techniques: *collection from the left* [LGS90, VL90]; and a slightly improved version of *collection to the left* [HJ76] (we collect the rightmost rather than the leftmost occurrence of the least uncollected letter).

Let U be an FC group functor over $\mathbb{F}$, and let $\mathbb{E}$ be a commutative $\mathbb{F}$-algebra. The algorithms in this section operate on a word $w \in U(\mathbb{E})$. This word is always equal to $\prod_{i=1}^{M} x_{r_i}(a_i)^{\varepsilon_i}$, ie, the parameters $M \in \mathbb{N}$, $a_i \in \mathbb{E}$, $\varepsilon_i = \pm 1$, and $r_i \in \{1, \dots, N\}$ are automatically modified when w is. Algorithm 1 describes the basic step of collection. When we say "*apply* a certain relation to a subword", we mean match the subword with the left hand side of the relation, and replace it by the right hand side. CollectSubword looks at the term at position j in the word w, and either removes an inverse (if $\varepsilon_j = -1$) or ensures that $r_{j-1} < r_j$. In addition to the modified word w, it returns indices j_1 and j_2. Collection to the left (Algorithm 2) works by collecting all terms $x_1(a)$, followed by all terms $x_2(a)$, and so on. This uses the index j_1, which gives the new largest j such that $r_j = r$. Collection from the left (Algorithm 3) goes through the word from left to right, correcting each term that is out of position. This uses the index j_2, which gives the next term which is potentially out of position.

```
CollectSubword := function($U, w = \prod_{i=1}^{M} x_{r_i}(a_i)^{\varepsilon_i}, j$)
  if $\varepsilon_j = -1$ then
      apply (4) to the subword $x_{r_j}(a_j)^{-1}$
      $j_1 := j, \quad j_2 := j$
  else if $j > 1$ and $r_{j-1} = r_j$ then
      apply (3) to the subword $x_{r_j}(a_{j-1})x_{r_j}(a_j)$
      $j_1 := j - 1, \quad j_2 := j_1 + \#\{t : f_{r_j t}(a_{j-1}, a_j) \neq 0\}$
  else if $j > 1$ and $r_{j-1} > r_j$ then
      apply (5) to the subword $x_{r_{j-1}}(a_{j-1})x_{r_j}(a_j)$
      $j_1 := j - 1$
      if $j_1 > 1$ and $r_{j_1-1} < r_{j_1}$ then
          $j_2 := j_1$
      else
          $j_2 := j_1 + 1 + \#\{t : h_{r_j r_{j-1} t}(a_{j-1}, a_j) \neq 0\}$
      end if
  else
      $j_1 := j, \quad j_2 := j_1 + 1$
  end if
  return $w, j_1, j_2$
```

Algorithm 1: Collect subword

```
Input: An FC group functor $U$ and a word $w = \prod_{i=1}^{M} x_{r_i}(a_i)^{\varepsilon_i}$.
Output: A product $\prod_{r=1}^{N} x_r(b_r)$ that is equal to $w$ as an element of $U(\mathbb{E})$.
  for $r := 1$ to $N$ do
      let $j$ be the largest $i$ such that $r_i = r$
      while $j \geq r$ do
          $w, j_1, j_2 :=$ CollectSubword($U, w, j$), $\quad j := j_1$
      end while
  end for
  return $w$
```

Algorithm 2: Collection to the left

Symbolic collection is a standard method for improving the efficiency of element multiplication in a PC group. This depends on the observation that we can collect a generic product, and then substitute into polynomials for subsequent collections. This is particularly easy in the case of FC group functors: simply take

```
Input: An FC group functor $U$ and a word $w = \prod_{i=1}^{M} x_{r_i}(a_i)^{\varepsilon_i}$.
Output: A product $\prod_{r=1}^{N} x_r(b_r)$ that is equal to $w$ as an element of $U(\mathbb{E})$.
  $j := 1$
  while $j \leq M$ do
      $w, j_1, j_2 :=$ CollectSubword$(U, w, j)$,    $j := j_2$
  end while
  return $w$
```

Algorithm 3: Collection from the left

$\mathbb{E} = \mathbb{F}[a_1, a_2, \ldots, a_N, b]$ and do N collections in $U(\mathbb{E})$ to get relations

$$\left(\prod_{s=r+1}^{N} x_s(a_s)\right) x_r(b) = x_r(b)\left(\prod_{s=r+1}^{N} x_s(c_{rs})\right) \tag{8}$$

where c_{rs} is a polynomial in b and $a_{r+1}, \ldots, a_N$. Now, taking arbitrary $\mathbb{E}$ again, we can multiply two collected words $\prod_{i=1}^{N} x_i(a_i)$ and $\prod_{i=1}^{N} x_i(b_i)$ by substituting values from $\mathbb{E}$ into the $(N-1)N/2$ polynomials c_{rs} in the obvious manner. A similar method can be used to compute inverses.

The advantage of symbolic collection is that each operation is faster. The disadvantage is that more preprocessing time and memory are required. In order to save memory, we represent our polynomials as straight-line programs (see [**?**] for a description of straight-line programs for group elements; the implementation for polynomials is due to Allan Steel). This means that the polynomials are basically just the collection preserved in amber, so the collection method used is still important. We note that there is another common symbolic collection algorithm, called Deep Thought [LGS98, Mer97], but we have not implemented it for unipotent groups.

4. Collection in the full unipotent subgroup

We now describe a new collection strategy for the full unipotent subgroup of a reductive group. Let G be an $\mathbb{F}$-split reductive algebraic group [Spr98]. Fix a split maximal torus T in G, and a Borel subgroup B containing T. Let U be the unipotent radical of B. Since U is unique up to G-conjugacy, we refer to U as the *full unipotent subgroup* of G. Let Φ be the root system of G with respect to T, and let $\Phi^+ \subseteq \Phi$ be the positive roots with respect to B.

Write $\Phi^+ = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$ with the roots in an *order compatible with height*, ie, $\mathrm{ht}(\alpha_r) < \mathrm{ht}(\alpha_s)$ implies $r < s$. For each $\alpha \in \Phi^+$, there is a subgroup X_α of U isomorphic to $\mathbb{G}_\mathrm{a}$. Write x_r for the isomorphism $\mathbb{G}_\mathrm{a} \to X_{\alpha_r}$. Then $U_r = \prod_{s=r}^{N} X_{\alpha_s}$, for $r = 1, \ldots, N+1$, defines a central series for U. The corresponding FC group functor has relations (6) and

$$x_s(b)x_r(a) = x_r(a)x_s(b) \prod_{\substack{\alpha_t = i\alpha_r + j\alpha_s, \\ i,j>0}} x_t(C_{ij\alpha_r\alpha_s} a^i b^j), \tag{9}$$

where the constants $C_{ij\alpha_r\alpha_s}$ are defined as in [Car72]. Recall that these constants depend on the combinatorics of Φ, and on the choice of a sign for each nonsimple positive root. In [CMT04], a method for computing these constants is given which is efficient for small rank groups. For large ranks, we outline a new method in Section 5. We now have:

Proposition 4.1. *The full unipotent subgroup of a split reductive group is presented by a split FC group functor.*

Note that most of the polynomials h_{rst} from (5) of Section 2 are zero in this case. This gives us much greater flexibility in how we collect words. The ordering of the roots is of vital importance in this section. We specify an ordering in terms of subscripts: $\alpha_1, \ldots, \alpha_N$. The subscripts on the injections $x_r : \mathbb{G}_{\mathrm{a}} \to U$ are always kept in agreement with the root ordering under discussion.

Words in U need not be collected into an order compatible with height. In fact, the algorithms of the previous section work for all orderings $\alpha_1, \alpha_2, \ldots, \alpha_N$ of Φ^+ with the property that $\alpha_r + \alpha_s = \alpha_t$ implies $t > r$ and $t > s$. We call such an ordering *left-additive*.

An ordering $\alpha_1, \alpha_2, \ldots, \alpha_N$ of Φ^+ is called *additive* if $\alpha_r + \alpha_s = \alpha_t$ implies t lies between r and s (ie, $r < t < s$ or $s < t < r$). Additive orderings reflect more of the combinatorial structure of the root system than left-additive orderings do. The existence and construction of additive orderings is considered in [Pap94] (see Section 6 below for more details).

In order to collect a word into the additive ordering, we replace relation (9) with

$$x_s(b)x_r(a) = x_r(a)\Bigg(\prod_{\substack{\alpha_t = i\alpha_r + j\alpha_s,\\ i,j>0}} x_t(C_{ji\alpha_s\alpha_r} a^i(-b)^j)\Bigg) x_s(b), \tag{10}$$

which can be proved by applying (9) to $x_r(a)x_s(-b)$. We then use *collection from the outside* (Algorithm 4), which is a modified version of collection from the left. The basic idea is to run collection from both sides simultaneously, until we meet in the middle. The two collections could easily be run in parallel, but we alternate between them. The subroutine CollectSubword from Algorithm 3 is slightly modified for both collections (CollectSubwordL and CollectSubwordR). The returned value L is the increase in the word length and k is the index of the next term potentially out of position.

Finally we note that symbolic collection works with collection from the outside, with the obvious minor modifications.

5. The full unipotent subgroup of a classical group

In this section, we present an alternative to collection which is much more efficient when G has large semisimple rank. We derive formulas for the defining polynomials of the full unipotent subgroup U of a split classical group G. This allows us to write code that implicitly applies the defining polynomials of U without having to store them explicitly in memory. We do not consider exceptional groups here, because their semisimple rank is at most 8.

The rough outline of our method is as follows: We index the roots by pairs of integers. We then construct a minimal degree matrix representation of G. We take the basis for the representation consisting of weight vectors, ordered according to the dominance ordering on the corresponding weights [**?**]. We then order the roots by going down each column of this matrix representation, and seeing where the parameters corresponding to each root appear. We call this the *representation*

```
CollectSubwordL := function($U, w = \prod_{i=1}^{M} x_{r_i}(a_i)^{\varepsilon_i}, j$)
  if $\varepsilon_j = -1$ then
      apply (7) to the subword $x_{r_j}(a_j)^{-1}$
      $k := j, \quad L := 0$
  else if $j > 0$ and $r_{j-1} = r_j$ then
      apply (6) to the subword $x_{r_j}(a_{j-1})x_{r_j}(a_j)$
      $k := j - 1, \quad L := -1$
  else if $j > 0$ and $r_{j-1} > r_j$ then
      apply (10) to the subword $x_{r_{j-1}}(a_{j-1})x_{r_j}(a_j)$
      $k := j - 1, \quad L := \#\{t : \alpha_t = k\alpha_{r_j} + l\alpha_{r_{j-1}} \text{ for } k, l > 0\}$
  else
      $k := j + 1, \quad L := 0$
  end if
  return $w$, $k$, $L$
```

```
CollectSubwordR := function($U, w = \prod_{i=1}^{M} x_{r_i}(a_i)^{\varepsilon_i}, j$)
  if $\varepsilon_j = -1$ then
      apply (7) to the subword $x_{r_j}(a_j)^{-1}$
      $k := j, \quad L := 0$
  else if $j < M$ and $r_j = r_{j+1}$ then
      apply (6) to the subword $x_{r_j}(a_j)x_{r_j}(a_{j+1})$
      $k := j, \quad L := -1$
  else if $j < M$ and $r_j > r_{j+1}$ then
      apply (10) to the subword $x_{r_j}(a_j)x_{r_{j+1}}(a_{j+1})$
      $L := \#\{t : \alpha_t = k\alpha_{r_{j+1}} + l\alpha_{r_j} \text{ for } k, l > 0\}, \quad k := j + 1 + L$
  else
      $k := j - 1, \quad L := 0$
  end if
  return $w$, $k$, $L$
```

Input: An FC group functor U and a word $w = \prod_{i=1}^{M} x_{r_i}(a_i)^{\varepsilon_i}$.
Output: A product $\prod_{r=1}^{m} x_r(b_r)$ that is equal to the input as an element of $U(\mathbb{E})$.

```
  $i := 1, \quad j := M$
  while $i < j$ do
      $w, k, L :=$ CollectSubwordL($U, w, i$), $\quad i := k,\ M := M + L,\ j := j + L$
      if $i < j$ then
          $w, k, L :=$ CollectSubwordR($U, w, j$), $\quad j := k,\ M := M + L$
      end if
  end while
  return $w$
```

Algorithm 4: Collection from the outside

ordering. Note that the representation ordering is also an additive ordering, except in Cartan type C_ℓ, where it gives an additive ordering on the coroots.

The representation of roots by pairs of integers can also be used to compute the constants $C_{ij\alpha_r\beta_r}$. For large classical groups, this is much more efficient than the method given in [CMT04]. We omit the details since this is a technical but straightforward application of formulas in [Car72].

We consider each classical type in Subsections 5.1 to 5.4. In each subsection, we fix a particular isogeny type and particular extraspecial signs [Spr98]. The choice of isogeny class is irrelevant, since the structure of the unipotent subgroup is independent of isogeny. We can easily transform between different choices of extraspecial signs using Theorem 29 of [Ste68].

5.1. Cartan type A_ℓ: Linear of degree $\ell + 1$.

Let $G = \mathrm{SL}_{l+1}$ and let U be the algebraic subgroup of all lower unitriangular matrices. The root system of G has Cartan type A_ℓ. Let $V = \mathbb{R}^{\ell+1}$ with basis $e_1, \dots, e_{\ell+1}$. Then the roots are

$$\alpha_{ij} = e_i - e_j$$

for $i, j = 1, \dots, \ell+1$ with $i \neq j$. The simple roots are $\alpha_{i,i+1}$ for $i = 1, \dots, \ell$. A root α_{ij} is positive if, and only if, $i < j$. Roots add by the formulas $\alpha_{ij} + \alpha_{jm} = \alpha_{im}$, and $\alpha_{ij} + \alpha_{km} = 0$ unless $j = k$ or $i = m$.

Define the map $x_{ij} : \mathbb{G}_\mathrm{a} \to G$ by $x_{ij}(a) = I + aE_{ji}$. The representation order on the roots is simply the lexicographic order on the corresponding pairs of integers. We label the coordinates of $\mathbb{A}^{(\ell+1)\ell/2}$ by the root pairs in this order, ie, $\boldsymbol{a} \in \mathbb{A}^{(\ell+1)\ell/2}(\mathbb{E})$ has the form

$$\boldsymbol{a} = (a_{12}, a_{13}, \dots, a_{1,\ell+1},\ a_{23}, \dots, a_{2,\ell+1},\ \dots,\ a_{\ell,\ell+1}).$$

We can now define a parametrisation $\varphi : \mathbb{A}^{(\ell+1)\ell/2} \to U$ by

$$\varphi(\boldsymbol{a}) := \prod_{i=1}^{\ell+1} \prod_{j=i+1}^{\ell+1} x_{ij}(a_{ij}) = \begin{pmatrix} 1 & & & & \\ a_{12} & 1 & & & \\ a_{13} & a_{23} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ a_{1,\ell+1} & a_{2,\ell+1} & \dots & a_{\ell,\ell+1} & 1 \end{pmatrix}.$$

We now get $\varphi(\boldsymbol{a})\varphi(\boldsymbol{b}) = \varphi(\boldsymbol{c})$ where

$$c_{ij} = a_{ij} + \sum_{i<k<j} b_{ik}a_{kj} + b_{ij}.$$

Also $\varphi(\boldsymbol{a})^{-1} = \varphi(\boldsymbol{d})$ where

$$d_{ij} = -a_{ij} - \sum_{i<k<j} d_{ik}a_{kj}$$

The formulas for inversion are defined recursively and are computed in reverse representation order. We note that it is easy to derive a direct formula for d_{ij}, but the recursive version can be evaluated with fewer operations.

5.2. **Cartan type B_ℓ: Orthogonal of degree $2\ell+1$.** Let F_m be the $m \times m$ matrix over $\mathbb{F}$ of the form

$$\begin{pmatrix} 0 & \dots & 0 & 1 \\ 0 & \dots & 1 & 0 \\ \vdots & ⋰ & \vdots & \vdots \\ 1 & \dots & 0 & 0 \end{pmatrix}.$$

We assume, for this subsection only, that $\mathbb{F}$ has odd characteristic. Since the group of type B_ℓ is isomorphic to the group of type C_ℓ in characteristic 2, this restriction is not critical. Let $G = \mathrm{SO}_{2\ell+1}$ be the special orthogonal group of the orthogonal form with matrix

$$\begin{pmatrix} 0 & 0 & F_\ell \\ 0 & 2 & 0 \\ F_\ell & 0 & 0 \end{pmatrix}.$$

Let U be the group of all lower unitriangular matrices in G. The root system of G has Cartan type B_ℓ. Let $V = \mathbb{R}^\ell$ with basis $e_1, \dots, e_\ell$. The roots are

$$\alpha_{si,tj} = se_i - te_j \quad \text{and} \quad \alpha_{si,0} = se_i,$$

for $i, j = 1, \dots, \ell$ with $i \neq j$ and $s, t = \pm 1$. For the sake of readability, we write $\bar{\imath}$ instead of $-i$ in subscripts, eg, $\alpha_{2,-3}$ is denoted $\alpha_{2\bar{3}}$. Note that $\alpha_{ij} = \alpha_{\bar{\jmath}\bar{\imath}}$ for all $i, j = \pm 1, \dots, \pm \ell$. The simple roots are $\alpha_{i,i+1}$, for $i = 1, \dots, \ell-1$, and α_{n0}. A root $\alpha_{i,tj}$ for $i, j > 0$ is positive if, and only if, $i < j$; a root $\alpha_{si,0}$ is positive if, and only if, $s = +1$.

Define the root maps by

$$\begin{aligned} x_{ij}(a) &= I + a(E_{ji} - E_{2\ell-i+2,2\ell-j+2}), \\ x_{i\bar{\jmath}}(a) &= I + a(E_{2\ell-j+2,i} - E_{2\ell-i+2,j}), \\ x_{i0}(a) &= I + a(2E_{\ell+1,i} - E_{i,\ell+1}) - a^2 E_{2\ell-i+2,i}, \quad \text{and} \\ x_{\bar{\imath}0}(a) &= I + a(E_{2\ell-i+2,\ell+1} - 2E_{\ell+1,2\ell-i+2}) - a^2 E_{i,2\ell-i+2}. \end{aligned}$$

Let J_i be the sequence of integers $[i+1, i+2, \dots, \ell, 0, -\ell, \dots, -(i+2), -(i+1)]$. Let $J_i' := J_i \setminus \{0\}$. The representation order on the positive roots is the lexicographic order on pairs, with the integers ordered as in J_0. Label the coordinates of $\mathbb{A}^{\ell^2}$ by the root pairs in this order. Now define a parametrisation $\varphi : \mathbb{A}^{\ell^2} \to U$ by

$$\begin{aligned} \varphi(\boldsymbol{a}) &:= \prod_{i=1}^{\ell} \prod_{j \in J_i} x_{ij}(a_{ij}) \\ &= \begin{pmatrix} 1 &&&&&&&& \\ a_{12} & 1 &&&&&&& \\ \vdots & \ddots & \ddots &&&&&& \\ a_{1\ell} & \dots & a_{\ell-1,\ell} & 1 &&&&& \\ 2a_{10} & \dots & 2a_{\ell-1,0} & 2a_{\ell 0} & 1 &&&& \\ a_{1\bar{\ell}} & \dots & a_{\ell-1,\bar{\ell}} & a_\ell'' & a_{\ell 0}' & 1 &&& \\ \vdots & ⋰ & ⋰ & a_{\ell-1,\bar{\ell}}' & a_{\ell-1,0}' & a_{\ell-1,\ell} & \ddots && \\ a_{1\bar{2}} & a_2'' & ⋰ & \vdots & \vdots & \vdots & \ddots & 1 & \\ a_1'' & a_{1\bar{2}}' & \dots & a_{1\bar{\ell}}' & a_{10}' & a_{1\ell}' & \dots & a_{12}' & 1 \end{pmatrix}. \end{aligned}$$

where

$$\begin{aligned} a''_i &= -2{a_{i0}}^2 - \sum_{i<k\leq\ell} a_{ik}a_{i\bar{k}}, \qquad a'_{i0} = -a_{i0} - \sum_{i<k\leq\ell} a_{ik}a'_{k0}, \\ a'_{ij} &= -a_{ij} - \sum_{i<k<j} a_{ik}a'_{kj}, \\ a'_{i\bar{j}} &= -a_{i\bar{j}} - \sum_{i<k<j} a_{ik}a'_{k\bar{j}} - a_{ij}a''_j - 2a_{i0}a_{j0} - \sum_{k\in J'_j} a_{ik}a_{j\bar{k}}. \end{aligned}$$

Now $\varphi(\boldsymbol{a})\varphi(\boldsymbol{b}) = \varphi(\boldsymbol{c})$ where

$$\begin{aligned} c_{i0} &= a_{i0} + b_{i0} + \sum_{i<k\leq\ell} b_{ik}a_{k0}, \qquad c_{ij} = a_{ij} + b_{ij} + \sum_{i<k<j} b_{ik}a_{kj}, \\ c_{i\bar{j}} &= a_{i\bar{j}} + b_{i\bar{j}} + \sum_{i<k<j} b_{ik}a_{k\bar{j}} + a''_j b_{ij} + 2b_{i0}a'_{j0} + \sum_{k\in J'_j} b_{ik}a'_{j\bar{k}}. \end{aligned}$$

And $\varphi(\boldsymbol{a})^{-1} = \varphi(\boldsymbol{d})$ where

$$\begin{aligned} d_{i0} &= -a_{i0} - \sum_{i<k\leq\ell} d_{ik}a_{k0}, \qquad d_{ij} = -a_{ij} - \sum_{i<k<j} d_{ik}a_{kj}, \\ d_{i\bar{j}} &= -a_{i\bar{j}} - \sum_{i<k<j} d_{ik}a_{k\bar{j}} - a''_j d_{ij} - 2d_{i0}a'_{j0} - \sum_{k\in J'_j} d_{ik}a'_{j\bar{k}}. \end{aligned}$$

All of these equations are recursive and are computed in reverse representation order.

5.3. **Cartan type C: Symplectic.** Let $G = \mathrm{Sp}_{2\ell}$ be the symplectic group of the symplectic form with matrix

$$\begin{pmatrix} 0 & F_\ell \\ -F_\ell & 0 \end{pmatrix}$$

Let U be the group of all lower unitriangular matrices in G. The root system of G has Cartan type C_ℓ. Let $V = \mathbb{R}^\ell$ with basis $e_1, \ldots, e_\ell$. The roots are

$$\alpha_{si,tj} = se_i - te_j,$$

for $i, j = 1, \ldots, \ell$ with $i \neq j$, and $s, t = \pm 1$. Once again $\alpha_{ij} = \alpha_{\bar{j}\bar{\imath}}$ for all $i, j = 0, \pm 1, \ldots, \pm\ell$. The simple roots are $\alpha_{i,i+1}$ for $i = 1, \ldots, \ell - 1$, and $\alpha_{\ell\bar{\ell}}$. A root $\alpha_{i,tj}$ is positive if, and only if, $i < j$.

Define the root maps by

$$\begin{aligned} x_{ij}(a) &= I + a(E_{ji} - E_{2\ell-i+1,2\ell-j+1}), \\ x_{i\bar{j}}(a) &= I + a(E_{2\ell-j+1,i} + E_{2\ell-i+1,j}), \quad \text{and} \\ x_{i\bar{\imath}}(a) &= I + aE_{2\ell-i+1,i}. \end{aligned}$$

The representation order on the positive roots is the lexicographic order on pairs, with the integers ordered as in J'_0. Label the coordinates of $\mathbb{A}^{\ell^2}$ by the root pairs in this order. Now define a parametrisation $\varphi : \mathbb{A}^{\ell^2} \to U$ by

$$\varphi(\boldsymbol{a}) := \prod_{i=1}^{\ell+1} \prod_{j \in J_i'} x_{ij}(a_{ij})$$

$$= \begin{pmatrix} 1 &&&&&&& \\ a_{12} & 1 &&&&&& \\ \vdots & \ddots & \ddots &&&&& \\ a_{1\ell} & \dots & a_{\ell-1,\ell} & 1 &&&& \\ a_{1\bar{\ell}} & \dots & a_{\ell-1,\bar{\ell}} & a''_\ell & 1 &&& \\ \vdots & \iddots & \iddots & a'_{\ell-1,\bar{\ell}} & a'_{\ell-1,\ell} & \ddots && \\ a_{1\bar{2}} & a''_2 & \iddots & \vdots & \vdots & \ddots & 1 & \\ a''_1 & a'_{1\bar{2}} & \dots & a'_{1\bar{\ell}} & a'_{1\ell} & \dots & a'_{12} & 1 \end{pmatrix},$$

where

$$a''_i = a_{i\bar{\imath}} - \sum_{i<j\le\ell} a_{ij}a_{i\bar{\jmath}}, \qquad a'_{ij} = a_{ij} - \sum_{i<k<j} a_{ik}a'_{kj},$$

$$a'_{i\bar{\jmath}} = a_{i\bar{\jmath}} - \sum_{i<k<j} a_{ik}a'_{k\bar{\jmath}} - a_{ij}a''_j - \sum_{k\in J'_j} \operatorname{sign}(k)a_{ik}a_{j\bar{k}}.$$

Now $\varphi(\boldsymbol{a})\varphi(\boldsymbol{b}) = \varphi(\boldsymbol{c})$ where

$$c_{ij} = a_{ij} + b_{ij} + \sum_{i<k<j} b_{ik}a_{kj}, \qquad c_{i\bar{\imath}} = a''_i + b''_i + \sum_{i<k\le\ell} c_{ik}c_{i\bar{k}} + \sum_{k\in J'_i} b_{ik}a'_{i\bar{k}},$$

$$c_{i\bar{\jmath}} = a_{i\bar{\jmath}} + b_{i\bar{\jmath}} + \sum_{i<k<j} b_{ik}a_{k\bar{\jmath}} + a''_j b_{ij} + \sum_{k\in J'_j} b_{ik}a'_{j\bar{k}}.$$

And $\varphi(\boldsymbol{a})^{-1} = \varphi(\boldsymbol{d})$ where

$$d_{ij} = -a_{ij} - \sum_{i<k<j} d_{ik}a_{kj}, \qquad d_{i\bar{\imath}} = -a''_i + \sum_{i<k\le\ell} d_{ik}d_{i\bar{k}} - \sum_{k\in J'_i} d_{ik}a'_{i\bar{k}},$$

$$d_{i\bar{\jmath}} = -a_{i\bar{\jmath}} - \sum_{i<k<j} d_{ik}a_{k\bar{\jmath}} - a''_j d_{ij} - \sum_{k\in J'_j} d_{ik}a'_{j\bar{k}}.$$

5.4. **Cartan type D: Even-degree orthogonal.** Let $G = \mathrm{SO}_{2\ell}$ be the orthogonal group of the orthogonal form with matrix $F_{2\ell}$. Let U be the group of all lower unitriangular matrices in G. The root system of G has Cartan type D_ℓ. Let $V = \mathbb{R}^\ell$ with basis $e_1, \dots, e_\ell$. The roots are

$$\alpha_{si,tj} = se_i - te_j,$$

for $i, j = 1, \dots, \ell$ with $i \neq j$ and $s, t = \pm 1$. The simple roots are $\alpha_{i,i+1}$, for $i = 1, \dots, \ell-1$, and $\alpha_{n-1,\bar{\ell}}$. A root $\alpha_{i,tj}$ for $i, j > 0$ is positive if, and only if, $i < j$.

Define the root maps by

$$x_{ij}(a) = I + a(E_{ji} - E_{2\ell-i+1,2\ell-j+1}) \quad \text{and}$$
$$x_{i\bar{\jmath}}(a) = I + a(E_{2\ell-j+1,i} - E_{2\ell-i+1,j}).$$

The representation order on the positive roots is the lexicographic order on pairs, with the integers ordered as in J_0'. Label the coordinates of $\mathbb{A}^{\ell(\ell-1)/2}$ by the root pairs in this order. Now define a parametrisation $\varphi : \mathbb{A}^{\ell(\ell-1)/2} \to U$ by

$$\begin{aligned}
\varphi(\boldsymbol{a}) &:= \prod_{i=1}^{\ell} \prod_{j \in J_i''} x_{ij}(a_{ij}) \\
&= \begin{pmatrix}
1 & & & & & & & \\
a_{12} & 1 & & & & & & \\
\vdots & \ddots & \ddots & & & & & \\
a_{1\ell} & \dots & a_{\ell-1,\ell} & 1 & & & & \\
a_{1\bar{\ell}} & \dots & a_{\ell-1,\bar{\ell}} & a_\ell'' & 1 & & & \\
\vdots & \iddots & \iddots & a_{\ell-1,\bar{\ell}}' & a_{\ell-1,\ell}' & \ddots & & \\
a_{1\bar{2}} & a_2'' & \iddots & \vdots & \vdots & \ddots & 1 & \\
a_1'' & a_{1\bar{2}}' & \dots & a_{1\bar{\ell}}' & a_{1\ell}' & \dots & a_{12}' & 1
\end{pmatrix}.
\end{aligned}$$

where

$$\begin{gathered}
a_i'' = \sum_{i<j\leq\ell} a_{ij}a_{i\bar{j}}, \qquad a_{ij}' = -a_{ij} - \sum_{i<k<j} a_{ik}a_{kj}', \\
a_{i\bar{j}}' = -a_{i\bar{j}} - \sum_{i<k<j} a_{ik}a_{k\bar{j}}' - a_j''a_{ij} - \sum_{k\in J_j'} a_{ik}a_{j\bar{k}}.
\end{gathered}$$

Now $\varphi(\boldsymbol{a})\varphi(\boldsymbol{b}) = \varphi(\boldsymbol{c})$ where

$$\begin{aligned}
c_{ij} &= a_{ij} + b_{ij} + \sum_{i<k<j} b_{ik}a_{kj}, \\
c_{i\bar{j}} &= a_{i\bar{j}} + b_{i\bar{j}} + \sum_{i<k<j} b_{ik}a_{k\bar{j}} + a_j''b_{ij} + \sum_{k\in J_j'} b_{ik}a_{j\bar{k}}'.
\end{aligned}$$

And $\varphi(\boldsymbol{a})^{-1} = \varphi(\boldsymbol{d})$ where

$$\begin{aligned}
d_{ij} &= -a_{ij} - \sum_{i<k<j} d_{ik}a_{kj}, \\
d_{i\bar{j}} &= -a_{i\bar{j}} - \sum_{i<k<j} d_{ik}a_{k\bar{j}} - a_j''d_{ij} - \sum_{k\in J_j'} d_{ik}a_{j\bar{k}}'.
\end{aligned}$$

6. Analysis and reductive groups

We can now give precise asymptotic timings for operations in reductive groups and their full unipotent subgroups. We give our analysis in terms of the number of basic operations in the algebra $\mathbb{E}$: addition, negation, multiplication, and testing equality. Once again let G be an $\mathbb{F}$-split reductive algebraic group, with split maximal torus T, and Borel subgroup B containing T. Let U be the unipotent radical of B. Let $W = N_G(T)/T$ be the Weyl group and let Φ be the root system. The reflection in W corresponding to the root α is denoted s_α. Let Φ^+ be the positive roots with respect to B.

First we give an analysis for element operations in $U(\mathbb{E})$.

Theorem 6.1. *Let $\mathbb{F}$ be a field and let $\mathbb{E}$ be a commutative unital $\mathbb{F}$-algebra. Let U be the full unipotent subgroup of a split reductive linear algebraic group G over $\mathbb{F}$. Let ℓ be the semisimple rank of G. Then there is a normal form for elements of $U(\mathbb{E})$. The word problem for elements in normal form requires $O(\ell^2)$ algebra operations, and multiplying or inverting them requires $O(\ell^3)$ algebra operations.*

Proof. The normal form is a collected word, so the timing for the word problem follows from the fact that $N = |\Phi^+|$ is $O(\ell^2)$. We can assume that G is simple, since U is a direct sum of the full unipotent subgroups of the simple components of G.

If G is classical, the formulas of Section 5 require $O(\ell^3)$ field operations.

If G is exceptional, then ℓ is bounded. In this case we use symbolic collection. The Hall polynomials of the full unipotent subgroup of a split reductive group are independent of the algebra $\mathbb{E}$, since split reductive groups can be constructed as $\mathbb{Z}$-schemes. So the number of algebra operations required for inversion or multiplication is independent of the choice of $\mathbb{E}$. □

In the rest of this section, we take $\mathbb{E}$ to be an extension field of $\mathbb{F}$ and we add inversion to the list of basic operations in $\mathbb{E}$. We are primarily interested in computing in $U(\mathbb{E})$ because it allows us to compute in $G(\mathbb{E})$ with the algorithms of [CMT04, Section 5]. Recall that $G(\mathbb{E})$ has a Steinberg presentation with generators $x_\alpha(a)$, for $\alpha \in \Phi$ and $a \in \mathbb{E}$; n_α, for $\alpha \in \Phi$; and $t \in T(\mathbb{E})$. Note that the generator $x_r(a)$ of Section 4 can be identified with the generator $x_{\alpha_r}(a)$ of the Steinberg presentation. Every element $g \in G(\mathbb{E})$ can be written uniquely in Bruhat form:

$$g = ut\dot{w}u',$$

for

- $u \in U(\mathbb{E})$ stored as a collected word;
- $t \in T(\mathbb{E})$ stored as in [CMT04];
- $\dot{w} = n_{\alpha_1} \cdots n_{\alpha_m}$, where $s_{\alpha_1} \cdots s_{\alpha_m}$ is a reduced expression for $w \in W$; and
- $u' \in U_w(\mathbb{E})$ as a collected word, where U_w is the subgroup of U generated by the terms $x_\alpha(a)$, for α in $\Phi_w := \{\alpha \in \Phi^+ \mid \alpha w^{-1} \notin \Phi^+\}$.

Given two elements in Bruhat form, we need to find the Bruhat form of their product. The usual element operations in $U(\mathbb{E})$ are not sufficient for this purpose. There are two difficult steps, each of which requires a new operation in U. We now describe these operations and show how to carry them out with the methods of the previous sections.

6.1. **Single-term separation.** One difficult step is multiplying $g = ut\dot{w}u'$ by n_α for some $\alpha \in \Phi$. This is achieved with Algorithm 3 of [CMT04], which uses the following operation: write $u' = \prod_{\beta\in\Phi_w} x_\beta(a_\beta)$ in the form $x_\alpha(a_\alpha)v$ where $v = \prod_{\beta\in\Phi_w\setminus\{\alpha\}} x_\beta(b_\beta)$. We call this operation *single-term separation.*

This is easily done by collection: simply collect the term $x_\alpha(a_\alpha)$ to the front of the product as in collection to the left, then put v in the required form with collection from the outside. No extra terms of the form $x_\alpha(b)$ can appear in v because only terms corresponding to roots higher than α are created. We can also do single term separation symbolically as in Section 3.

Alternatively, for classical groups, we can compute v as the product $x_\alpha(-a_\alpha)u'$ using the formulas of Section 5. If $\alpha = \alpha_{ij}$, then the only possible nonzero constants in $\varphi(\boldsymbol{a})$ are a_{ij}, a'_{ij}, and a''_i. Hence at most $O(\ell)$ of the formulas for c_{ij} are nontrivial. Each such formula has at most a constant number of nonzero terms. We now have:

Proposition 6.2. *Single-term separation in $U(\mathbb{E})$ requires $O(\ell)$ field operations.*

Proof. Use formulas for classical components and symbolic collection for exceptional components. □

Note that, when both of the elements being multiplied are in Bruhat form, Algorithm 3 of [CMT04] only uses single-term separation for α simple. We have considered nonsimple roots as well, because they will be useful in the next subsection.

6.2. **Weyl separation.** The other difficult step for multiplication in G is computing the product of $g \in G$ and $v \in U$. Write g in Bruhat form $ut\dot{w}u'$. Then multiply u' and v, and decompose the product into the form $v''v'$ where

$$v'' = \prod_{\alpha \in \Phi^+ \setminus \Phi_w} x_\alpha(b_\alpha) \quad \text{and} \quad v' = \prod_{\alpha \in \Phi_w} x_\alpha(b_\alpha).$$

We call this operation *Weyl separation.* We now get the Bruhat form

$$gv = [u(v'')^{\dot{w}^{-1}t^{-1}}]t\dot{w}v'$$

where $(v'')^{\dot{w}^{-1}t^{-1}}$ is in U since $\alpha \in \Phi^+ \setminus \Phi_w$ implies αw^{-1} is positive.

If we take the elements of $\Phi^+ \setminus \Phi_w$ in an order compatible with height, followed by the elements of Φ_w in an order compatible with height, we get a left-additive ordering on Φ^+. So the algorithms of Section 3 can also be used for separation. But note that $(v'')^{\dot{w}^{-1}t^{-1}}$ will need to be collected again, since the image of the left additive ordering on $\Phi^+ \setminus \Phi_w$ under w^{-1} need not be left additive.

We can also use collection from the outside for Weyl separation. We need the following classification of additive orderings from [Pap94]:

Theorem 6.3. *Let w be an element of the Weyl group W. Let $s_{\beta_1} \cdots s_{\beta_m}$ be a reduced expression for w. Then*

$$\beta_1 s_{\beta_2} \cdots s_{\beta_N}, \ldots, \ \beta_{N-2} s_{\beta_{N-1}} s_{\beta_N}, \ \beta_{N-1} s_{\beta_N}, \ \beta_N$$

is an additive ordering on Φ_w. All additive orderings on Φ_w arise from reduced expressions in this manner.

Now let w_0 be the longest word in W and fix a reduced expression $s_{\alpha_1} \cdots s_{\alpha_N}$ for w_0 (in practice, we use the lexicographically least reduced expression, but this is not necessary). We use the additive ordering on Φ^+ corresponding to this reduced expression as the fixed order for collection. Now let w be a Weyl group element. If we restrict the fixed ordering to Φ_w we get an additive ordering, with corresponding reduced expression $s_{\beta_1} \ldots s_{\beta_m} = w$. Similarly we restrict to get an ordering on $\Phi_{w_0 w^{-1}} = (\Phi^+ \setminus \Phi_w)w^{-1}$ and a corresponding reduced expression $s_{\gamma_1} \ldots s_{\gamma_{N-m}} = w_0 w^{-1}$. Now $w_0 = s_{\gamma_1} \ldots s_{\gamma_{N-m}} s_{\beta_1} \ldots s_{\beta_m}$ is reduced. The corresponding ordering is: our fixed ordering restricted to $\Phi^+ \setminus \Phi_w$ and transformed by w, followed by our fixed ordering restricted to Φ_w. This is precisely the ordering we need for separation.

Finally we analyse Weyl separation:

Proposition 6.4. *Weyl separation in U requires $O(\ell^3)$ field operations.*

Proof. For classical components, we apply single-term separation for each root in Φ_w. By Proposition 6.2, this takes $O(N\ell) = O(\ell^3)$ operations. For exceptional components use symbolic collection. □

In the exceptional case, this proposition assumes we have a system of symbolic-collection polynomials for every Weyl element. Although this is polynomial time, the memory required to store all these polynomials is prohibitive. In practice, it is much faster to use collection from the outside for Weyl separation in exceptional groups.

6.3. **Operations in reductive groups.** We now prove the following result on computation in G:

Theorem 6.5. *Let $\mathbb{F}$ be a field and let $\mathbb{E}$ be an extension of $\mathbb{F}$. Let G be a split reductive linear algebraic group over the field $\mathbb{F}$. Let ℓ be the semisimple rank of G and let n be the reductive rank. Then there is a normal form for elements of $G(\mathbb{E})$. The word problem for elements in normal form requires $O(n + \ell^2)$ field operations, and multiplying or inverting them requires $O(n\ell^2)$ field operations.*

Proof. We use the Bruhat decomposition to store $g \in G$ in the normal form $g = uh\dot{w}u'$. Here u, u', and $\dot{w}$ are words of length at most N, while h has length n. Once again the timing for the word problem is clear. Now G is a central product of simple algebraic groups and a central torus of dimension at most n. Multiplying a toral element by an element in a simple component is done as in Subsection 5.5 of [CMT04], and takes time $O(n\ell^2)$. So it suffices to show that multiplication and inversion in a simple group G requires $O(\ell^3)$ operations. The algorithms for multiplication and inversion given in [CMT04] require a constant number of multiplications or inversions in $U(\mathbb{E})$, together with a constant number of Weyl separations and at most $O(\ell^2)$ single-term separations. The theorem now follows from Theorem 6.1, Proposition 6.2, and Proposition 6.4. □

Theorem 1.1 is an immediate consequence of this result and the fact that $\ell \leq n$.

7. Implementation and timings

A number of heuristic improvements are built into our implementations of the algorithms described. Most of them are either obvious or were suggested by our profiling of the code. We restrict ourselves here to a brief description of the basic data types used. Representations of elements of the field $\mathbb{F}$ or algebra $\mathbb{E}$ are taken care of by the Magma computer algebra system [BCP97]. Most of our code is written in traditional C [KR88] and incorporated into the Magma core. Less time-critical code is written in the Magma language itself.

A collected product $\prod_{r=1}^{N} x_r(a_r)$ is stored as a sequence $[a_1, \dots, a_N]$. While doing the collection, we represent a term $x_r(a)$ as a pair (r, a) of an integer and an element of $\mathbb{E}$. Note that pairs $(r, 0)$ are trivial – they are always eliminated as soon as they occur. A word $\prod_{i=1}^{M} x_{r_i}(a_i)^{\varepsilon_i}$ is represented as a doubly linked chain. That is, every root element in the chain contains a reference to its predecessor and successor, which is a null-reference if the element is the first (resp. last) in the chain:

$$\leftarrow \circ \leftrightarrows \circ \leftrightarrows \cdots \leftrightarrows \circ \rightarrow$$

We use this data structure because inserting and deleting terms in the word when applying relations (3)–(5) can be done in constant time. For sequences, insertion and deletion would be more expensive, since the tail of the sequence has to be copied in memory. The chain is doubly linked, since we need both the predecessor and the successor of a term in the word for the CollectSubword functions.

In our tables we use the following abbreviations for collection algorithms:

CTL:	Collection to the left, Section 3.
CFL:	Collection from the left, Section 3.
CFO:	Collection from the outside, Section 4.
SCFL:	Symbolic collection from the left, Section 3.
SCFO:	Symbolic collection from the outside, Section 4.

We have two different implementations of the method of Section 5:

D: Modified matrix multiplication. For given $\boldsymbol{a}, \boldsymbol{b}$, we use formulas of Section 5 to compute $\varphi(\boldsymbol{a})$ and the significant part of $\varphi(\boldsymbol{b})$ (we do not use b'_{ij} and, except in types B (even characteristic) and C, we do not use b''_i). Then the product of the two matrices is computed using algorithms implemented in the Magma computer algebra system [BCP97]. The resulting matrix agrees with $\varphi(\boldsymbol{c})$ in the entries c_{ij} and in the entries c''_i (where they are needed). Thus we can recover the product $\boldsymbol{c} = \boldsymbol{ab}$ from the resulting matrix by formulas of Section 5.

SD: Compute the polynomials of (8) in Section 3, using the formulas instead of collection.

Method D outperforms SD in most cases, since asymptotically fast algorithms are used for matrix multiplication. But SD is faster for fields with very rapid blow-up of terms, such as multivariate rational function fields. All timings were run on an AMD Opteron 150 Processor with 2393 MHz.

Table 1 gives times and memory consumption for creating the reductive groups and precomputing all constants. For symbolic algorithms, this also includes time taken to compute the polynomials. Note that all constants and polynomials are independent of the field, and are computed on a per-root-datum basis. This means that preprocessing time is nearly zero if a group with the same root datum has already been created in the same Magma session. We used a workspace of 4 gigabytes – when this is insufficient we do not give a time and write $> 4GB$ in the memory column. In columns D and SD, the constants are computed as they are needed and not stored in memory.

Table 2 gives average times for multiplying and inverting random elements of full unipotent groups over the field with 17 elements. The average is taken over 100 multiplications. The same random elements are used for different algorithms. If a single multiplication required more than 2 gigabytes of memory, we write $> 2GB$ instead of a time. We did not attempt those cases where the preprocessing took more than 4 gigabytes of memory.

Table 3 gives similar times for multiplying and inverting random elements of the reductive group itself. Each such operation involves a number of collections. Computing random elements in a reductive group can be time consuming, but this is not included in our timings.

Table 4 gives average times for multiplying and inverting random elements of full unipotent groups of reductive groups over different fields. Over the field of rational numbers, the random field elements are chosen by taking a random numerator and a random denominator of size up to the given number of bits and a random sign.

	Time							Memory (MB)						
	CTL	CFL	CFO	SCFL	SCFO	D	SD	CTL	CFL	CFO	SCFL	SCFO	D	SD
$A_{10}(17)$	0.790	0.790	0.780	0.780	0.820	0.180	0.200	4.469	4.469	4.469	4.674	4.659	3.726	3.800
$A_{20}(17)$	8.950	8.870	8.990	9.000	9.310	0.180	0.190	12.853	12.853	12.853	15.488	13.821	3.708	4.227
$A_{30}(17)$	44.110	44.330	44.150	46.670	45.580	0.200	0.240	53.915	53.915	53.915	79.217	58.180	3.781	5.416
$A_{100}(17)$	–	–	–	–	–	0.640	2.250	$> 4GB$	$> 4GB$	$> 4GB$	$> 4GB$	$> 4GB$	8.139	52.477
$B_{10}(17)$	4.490	4.320	4.480	4.390	4.580	0.180	0.180	4.731	4.731	4.731	6.219	6.377	3.693	3.992
$B_{20}(17)$	68.700	68.870	68.570	72.270	69.750	0.190	0.250	39.072	39.072	39.072	83.714	45.900	3.710	7.951
$B_{30}(17)$	357.170	357.060	355.760	437.320	365.790	0.200	0.430	177.452	177.452	177.452	613.679	426.778	3.790	14.250
$B_{100}(17)$	–	–	–	–	–	0.620	14.380	$> 4GB$	$> 4GB$	$> 4GB$	$> 4GB$	$> 4GB$	8.160	445.203
$C_{10}(17)$	4.460	4.400	4.560	4.380	4.530	0.180	0.200	4.730	4.730	4.730	6.218	5.549	3.693	3.992
$C_{20}(17)$	68.730	68.680	68.800	72.720	69.870	0.190	0.250	39.078	39.078	39.078	83.721	65.053	3.710	7.951
$C_{30}(17)$	354.660	357.230	354.970	436.860	363.670	0.190	0.440	177.449	177.449	177.449	613.676	285.094	3.790	14.250
$C_{100}(17)$	–	–	–	–	–	0.620	16.490	$> 4GB$	$> 4GB$	$> 4GB$	$> 4GB$	$> 4GB$	8.160	445.205
$D_{10}(17)$	1.820	1.820	1.710	1.820	1.870	0.190	0.160	4.344	4.344	4.344	5.198	4.895	3.692	3.986
$D_{20}(17)$	28.540	28.400	28.400	31.640	29.910	0.190	0.270	34.589	34.589	34.589	79.227	54.305	3.708	7.911
$D_{30}(17)$	153.440	153.080	152.900	220.920	159.850	0.200	0.400	167.857	167.857	167.857	604.077	249.470	3.785	14.242
$D_{100}(17)$	–	–	–	–	–	0.640	14.180	$> 4GB$	$> 4GB$	$> 4GB$	$> 4GB$	$> 4GB$	8.158	445.176
$G_2(17)$	0.210	0.190	0.190	0.190	0.200	–	–	3.580	3.580	3.580	3.580	3.580	–	–
$F_4(17)$	0.410	0.440	0.400	0.400	0.430	–	–	3.764	3.764	3.764	3.859	3.764	–	–
$E_6(17)$	0.440	0.450	0.430	0.440	0.460	–	–	4.532	4.532	4.532	4.532	4.532	–	–
$E_7(17)$	0.990	1.000	0.980	1.010	0.980	–	–	4.726	4.726	4.726	4.305	3.870	–	–
$E_8(17)$	3.130	3.110	3.100	3.180	3.230	–	–	6.088	6.088	6.088	14.175	7.376	–	–

TABLE 1. Time and memory consumption for preprocessing

Group	Multiply							Invert						
	CTL	CFL	CFO	SCFL	SCFO	D	SD	CTL	CFL	CFO	SCFL	SCFO	D	SD
$A_{10}(17)$	0.009	0.006	0.006	0.007	0.006	0.006	0.006	0.003	0.002	0.002	0.003	0.002	0.001	0.002
$A_{20}(17)$	1.511	0.058	0.030	0.174	0.031	0.020	0.032	1.051	0.042	0.012	0.157	0.019	0.003	0.017
$A_{30}(17)$	114.543	0.768	0.111	3.069	0.334	0.046	0.182	84.299	0.734	0.063	3.052	0.326	0.007	0.148
$A_{100}(17)$	–	–	–	–	–	0.854	51.744	–	–	–	–	–	0.071	52.425
$B_{10}(17)$	0.101	0.016	0.013	0.045	0.038	0.012	0.014	0.069	0.009	0.005	0.039	0.032	0.004	0.007
$B_{20}(17)$	280.288	1.144	0.180	5.134	0.614	0.050	0.321	209.010	1.105	0.136	5.474	0.622	0.017	0.305
$B_{30}(17)$	$> 2GB$	25.472	1.412	107.734	49.190	0.122	2.015	$> 2GB$	25.419	1.240	115.646	49.338	0.045	2.024
$B_{100}(17)$	–	–	–	–	–	2.728	1025.957	–	–	–	–	–	1.115	1048.822
$C_{10}(17)$	0.093	0.016	0.013	0.044	0.016	0.013	0.014	0.065	0.009	0.005	0.038	0.009	0.004	0.007
$C_{20}(17)$	266.161	1.133	0.178	4.999	2.306	0.050	0.319	194.365	1.106	0.119	5.411	2.316	0.017	0.300
$C_{30}(17)$	$> 2GB$	27.562	1.443	113.089	21.446	0.125	2.099	$> 2GB$	27.656	1.097	123.097	21.609	0.047	2.098
$C_{100}(17)$	–	–	–	–	–	2.760	1046.944	–	–	–	–	–	1.161	1054.659
$D_{10}(17)$	0.062	0.014	0.011	0.020	0.012	0.011	0.012	0.040	0.007	0.004	0.014	0.006	0.004	0.006
$D_{20}(17)$	195.012	0.887	0.144	4.470	1.621	0.046	0.293	141.648	0.874	0.096	4.713	1.619	0.016	0.275
$D_{30}(17)$	$> 2GB$	22.057	1.181	102.313	15.470	0.121	1.963	$> 2GB$	22.229	0.931	108.976	15.613	0.045	1.969
$D_{100}(17)$	–	–	–	–	–	2.631	1038.942	–	–	–	–	–	1.093	1044.249
$G_2(17)$	0.001	0.001	0.001	0.001	0.001	–	–	0.000	0.000	0.000	0.000	0.000	–	–
$F_4(17)$	0.004	0.003	0.003	0.003	0.003	–	–	0.001	0.001	0.001	0.001	0.001	–	–
$E_6(17)$	0.006	0.004	0.004	0.005	0.004	–	–	0.002	0.001	0.001	0.002	0.001	–	–
$E_7(17)$	0.042	0.009	0.007	0.014	0.008	–	–	0.029	0.004	0.002	0.009	0.004	–	–
$E_8(17)$	4.924	0.053	0.016	0.309	0.032	–	–	3.966	0.044	0.008	0.292	0.027	–	–

TABLE 2. Average time to multiply random elements of the full unipotent group

Group	Multiply							Invert						
	CTL	CFL	CFO	SCFL	SCFO	D	SD	CTL	CFL	CFO	SCFL	SCFO	D	SD
$A_{10}(17)$	0.123	0.119	0.118	0.121	0.117	0.288	0.295	0.163	0.160	0.162	0.163	0.160	0.222	0.220
$A_{20}(17)$	1.359	0.729	0.662	1.293	0.695	2.063	2.129	2.268	1.147	1.080	1.546	1.112	1.982	2.031
$A_{30}(17)$	120.524	10.798	7.887	21.833	8.936	29.141	29.752	104.102	13.639	12.655	21.838	13.484	21.959	22.627
$A_{100}(17)$	–	–	–	–	–			–	–	–	–	–		
$B_{10}(17)$	0.423	0.325	0.304	0.453	0.414	1.353	1.359	0.561	0.495	0.486	0.592	0.570	0.860	0.867
$B_{20}(17)$	342.015	9.964	5.184	28.542	7.234	27.707	29.383	237.331	10.345	8.721	24.834	10.331	18.471	19.728
$B_{30}(17)$	$> 2GB$	170.359	34.042	560.664	228.163	190.445	198.912	$> 2GB$	92.252	51.170	397.654	197.548	140.195	146.980
$B_{100}(17)$	–	–	–	–	–			–	–	–	–	–		
$C_{10}(17)$	0.430	0.329	0.314	0.455	0.334	1.420	1.426	0.556	0.488	0.483	0.588	0.502	0.893	0.905
$C_{20}(17)$	347.298	10.341	5.380	28.422	14.166	27.950	29.116	225.542	10.033	8.517	24.409	15.189	18.399	19.387
$C_{30}(17)$	$> 2GB$	174.191	33.764	580.168	116.204	181.660	190.782	$> 2GB$	99.784	54.166	414.583	115.765	138.295	145.607
$C_{100}(17)$	–	–	–	–	–			–	–	–	–	–		
$D_{10}(17)$	0.344	0.280	0.268	0.317	0.278	1.170	1.184	0.454	0.412	0.409	0.441	0.416	0.727	0.734
$D_{20}(17)$	225.229	8.183	4.606	24.562	10.825	23.712	24.692	166.790	8.538	7.340	21.308	12.015	15.674	16.571
$D_{30}(17)$	$> 2GB$	147.254	31.131	521.051	90.824	174.995	185.490	$> 2GB$	82.122	47.233	373.211	94.738	135.237	142.438
$D_{100}(17)$	–	–	–	–	–			–	–	–	–	–		
$G_2(17)$	0.008	0.008	0.006	0.008	0.008	–	–	0.006	0.006	0.006	0.006	0.006	–	–
$F_4(17)$	0.080	0.076	0.075	0.077	0.074	–	–	0.060	0.060	0.059	0.059	0.060	–	–
$E_6(17)$	0.172	0.166	0.158	0.165	0.158	–	–	0.128	0.129	0.124	0.127	0.126	–	–
$E_7(17)$	0.774	0.508	0.451	0.518	0.455	–	–	0.419	0.369	0.363	0.381	0.366	–	–
$E_8(17)$	52.481	3.098	1.566	3.815	1.624	–	–	8.781	1.396	1.288	1.974	1.353	–	–

Table 3. Average time to multiply random elements of the reductive group

Group	Multiply				Invert			
	CFO	SCFO	D	SD	CFO	SCFO	D	SD
$A_{20}(2)$	0.021	0.030	0.020	0.031	0.005	0.017	0.003	0.015
$A_{20}(17)$	0.029	0.031	0.020	0.032	0.012	0.018	0.003	0.017
$A_{20}(\mathbb{Q})$, 32 bits	0.045	0.038	0.024	0.038	0.051	0.109	0.213	0.135
$A_{20}(\mathbb{Q})$, 64 bits	0.049	0.041	0.028	0.041	0.086	0.195	0.736	0.265
$A_{20}(\mathbb{Q})$, 128 bits	0.056	0.047	0.039	0.047	0.177	0.394	2.877	0.611
$A_{20}(\mathbb{Q}(i))$, 32 bits	0.074	0.052	0.047	0.049	0.080	0.143	0.048	0.117
$A_{20}(\mathbb{Q}(p))$, 32 bits	0.108	0.069	0.062	0.062	0.122	0.265	0.068	0.214
$A_{20}(R)$	0.049	0.038	0.025	0.038	6.868	2.562	0.594	1.609
$B_{20}(2)$	0.056	0.589	0.047	0.303	0.023	0.590	0.015	0.282
$B_{20}(17)$	0.174	0.592	0.049	0.309	0.125	0.596	0.017	0.289
$B_{20}(\mathbb{Q})$, 32 bits	0.420	0.811	0.254	1.901	0.986	2.630	2.625	5.169
$B_{20}(\mathbb{Q})$, 64 bits	0.534	0.960	0.630	3.594	2.205	5.111	11.937	12.567
$B_{20}(\mathbb{Q})$, 128 bits	0.798	1.269	1.865	8.058	5.575	11.475	50.818	34.318
$B_{20}(\mathbb{Q}(i))$, 32 bits	0.680	0.972	1.078	2.124	1.270	2.654	1.344	3.874
$B_{20}(\mathbb{Q}(p))$, 32 bits	1.092	1.362	1.410	3.864	2.276	4.542	2.053	7.212
$B_{20}(R)$	0.589	0.777	$> 2GB$	31.884	$> 2GB$	$> 2GB$	$> 2GB$	$> 2GB$
$E_8(2)$	0.012	0.028	–	–	0.003	0.022	–	–
$E_8(17)$	0.016	0.029	–	–	0.007	0.023	–	–
$E_8(\mathbb{Q})$, 32 bits	0.047	0.125	–	–	0.061	0.283	–	–
$E_8(\mathbb{Q})$, 64 bits	0.075	0.206	–	–	0.118	0.526	–	–
$E_8(\mathbb{Q})$, 128 bits	0.141	0.390	–	–	0.262	1.099	–	–
$E_8(\mathbb{Q}(i))$, 32 bits	0.077	0.162	–	–	0.094	0.310	–	–
$E_8(\mathbb{Q}(p))$, 32 bits	0.113	0.291	–	–	0.146	0.621	–	–
$E_8(R)$	0.315	0.349	–	–	0.904	1.166	–	–

Table 4. Operations for random elements of the full unipotent group over different fields

	CFL		CFO	
	max	avg	max	lim avg
A_ℓ	ℓ	$(\ell+2)/3$	2	2
B_ℓ	$2\ell-1$	$(2\ell+\frac{3}{2}-\frac{1}{2\ell})/3$	4	4
C_ℓ	$2\ell-1$	$(2\ell+\frac{3}{2}-\frac{1}{2\ell})/3$	3	3
D_ℓ	$2\ell-3$	$(2\ell-1)/3$	3	3

Table 5. Total degrees of Hall polynomials

Similar random elements were used for the Gaussian integers $\mathbb{Q}(i)$ and for $\mathbb{Q}(p)$, which is the splitting field of a random irreducible polynomial of degree 6 with integral coefficients in the range 1 to 10. The field R is the multivariate rational function field over $\mathbb{Q}$ with 10 variables. Random field elements over R were taken to be random invariates. In $B_{20}(R)$ the coefficient blowup is so large that over 2 gigabytes of memory was needed in some cases (see entries in the table).

Finally, in Table 5, we compare the total degrees of the polynomials used for different kinds of symbolic collection. The last column contains $\lim_{\ell\to\infty} \operatorname{avg}\{\deg(p) : p \in \mathcal{P}\}$, where $\mathcal{P}$ is the set of polynomials used for symbolic collection. This goes a long way towards explaining why collection from the outside works so well. Since the polynomials are multivariate in $2N = 2|\Phi^+|$ variables, we still can have large polynomials. To have a very rough idea of the size, we printed the largest of the polynomials in type B_{15} as a string and measured its size in bytes. Using collection from the outside, the size is 9226 bytes; using collection from the left, the size is about 297 megabytes.

References

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (1997), no. 3-4, 235–265, Computational algebra and number theory (London, 1993). MR MR1484478

[Car72] Roger W. Carter, *Simple groups of Lie type*, John Wiley & Sons, London-New York-Sydney, 1972, Pure and Applied Mathematics, Vol. 28.

[CMT04] Arjeh M. Cohen, Scott H. Murray, and D. E. Taylor, *Computing in groups of Lie type*, Math. Comp. **73** (2004), 1477–1498.

[DG70] Michel Demazure and Pierre Gabriel, *Groupes algébriques. Tome I: Géométrie algébrique, généralités, groupes commutatifs*, Masson & Cie, Éditeur, Paris, 1970, Avec un appendice *Corps de classes local* par Michiel Hazewinkel.

[Hal69] Philip Hall, *The Edmonton notes on nilpotent groups*, Queen Mary College Mathematics Notes, Mathematics Department, Queen Mary College, London, 1969. MR MR0283083 (44 #316)

[HEO05] Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien, *Handbook of computational group theory*, Discrete Mathematics and its Applications (Boca Raton), Chapman & Hall/CRC, Boca Raton, FL, 2005. MR MR2129747

[HJ76] Marshall Hall Jr., *The theory of groups*, Chelsea Publishing Co., New York, 1976, Reprinting of the 1968 edition.

[KR88] Brian W. Kernighan and Dennis M. Ritchie, *The C programming language*, 2nd ed., Prentice-Hall Internat. Ser. Comput. Sci., Englewood Cliffs, NJ, 1988.

[LGS90] C. R. Leedham-Green and L. H. Soicher, *Collection from the left and other strategies*, J. Symbolic Comput. **9** (1990), no. 5-6, 665–675, Computational group theory, Part 1. MR 92b:20021

[LGS98] C. R. Leedham-Green and Leonard H. Soicher, *Symbolic collection using Deep Thought*, LMS J. Comput. Math. **1** (1998), 9–24 (electronic). MR 99f:20002

[Mer97] Wolfgang Wilhelm Merkwitz, *Symbolische multiplikation in nilpotenten gruppen mit deep thought*, Master's thesis, Rheinisch-Westfälischen Technischen Hochschule Aachen, 1997.

[Pap94] Paolo Papi, *A characterization of a special ordering in a root system*, Proc. Amer. Math. Soc. **120** (1994), no. 3, 661–665. MR 94e:20056

[Ros57] Maxwell Rosenlicht, *Some rationality questions on algebraic groups*, Ann. Mat. Pura Appl. (4) **43** (1957), 25–50. MR MR0090101 (19,767h)

[Ser88] Jean-Pierre Serre, *Algebraic groups and class fields*, Graduate Texts in Mathematics, vol. 117, Springer-Verlag, New York, 1988, Translated from the French. MR MR918564 (88i:14041)

[Spr98] T. A. Springer, *Linear algebraic groups*, second ed., Birkhäuser Boston Inc., Boston, MA, 1998.

[Ste68] R. Steinberg, *Lectures on Chevalley groups*, Tech. report, Yale University, 1968.

[VL90] M. R. Vaughan-Lee, *Collection from the left*, J. Symbolic Comput. **9** (1990), no. 5-6, 725–733, Computational group theory, Part 1. MR 92c:20065

[Wat79] William C. Waterhouse, *Introduction to affine group schemes*, Graduate Texts in Mathematics, vol. 66, Springer-Verlag, New York, 1979. MR 82e:14003

Department of Mathematics and Computer Science, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, Netherlands

E-mail address: A.M.Cohen@tue.nl

School of Mathematics and Statistics F07, Faculty of Science, University of Sydney, NSW 2006, Australia

E-mail address: sergei@sergei-haller.de, murray@maths.usyd.edu.au