

Implementing Constructive Recognition Algorithms for Finite Classical Groups

Peter Brooksbank

Department of Mathematics

Bucknell University

`pbrooksb@bucknell.edu`

Magma Workshop on Group Theory and Algebraic Geometry
University of Warwick, August 22–26, 2005

Constructive Recognition

\mathcal{C} = family of finite quasisimple groups

G = group, given by generators, isomorphic to known $T \in \mathcal{C}$

An **effective isomorphism** $\Psi: T \rightarrow G$ consists of two “efficient” procedures for computing:

- $\Psi(t) \in G$ for any given $t \in T$
- $\Psi^{-1}(g) \in T$ for any given $g \in G$

A **constructive recognition algorithm** for \mathcal{C} is an algorithm that produces an effective isomorphism $\Psi: T \rightarrow G$

Here \mathcal{C} will always be a family of classical groups

Straight-line Programs

In the usual strategy we aim to achieve rather more.

Let $G = \langle X \rangle$ be the given group, isomorphic to T (quasisimple).

(1) Choose a set, \mathcal{T} , of “standard generators” for T from which one can write a short **straight-line program (SLP)** to any given $t \in T$.

(2) Construct (from X) the image, X^* , of \mathcal{T} under some isomorphism $\Psi: T \rightarrow G$.

(3) Devise an algorithm that writes an SLP from X^* to any given $g \in G$. (Note: In so doing, we also obtain an SLP from the original generating set X , thus solving the “word problem” for G .)

Having done all this, effecting the isomorphism Ψ is easy. For example, given $t \in T$, write an SLP from \mathcal{T} to t , and then evaluate it from X^* to obtain $\Psi(t) \in G$.

Representations

There are various representations in which $G \cong T$ may be given. It is convenient to think broadly about two cases:

- Natural: e.g. $G \cong SL_2(q)$ given as 2×2 matrices over $GF(q)$.
- Unnatural: we often consider two subcases:
 - Natural (defining) characteristic: for T defined in char p , G consists of matrices also in char p
 - Black-box: we assume only “oracles” for operations in G (we use no representation specific information)

We often refer to algorithms that handle these three cases, as well as to the input groups themselves, respectively as

white box

grey box

black box

black box classical groups

Despite the lack of information one has to work with, there are constructive recognition algorithms for black box classical groups

- $GL_d(2)$ (Cooperman, Finkelstein & Linton)
- All classical groups (Kantor & Seress)

Purely black-box groups are in some sense “small” in relation to their input length (Landazuri & Seitz)

Consequence: The Kantor-Seress algorithm works fairly well in the black box setting

Discrete Logarithms

Given $G \cong T$, a classical group over $GF(q = p^k)$, a fundamental problem is to construct certain elements of G of order p

Random search is not fast enough: for example, roughly $1/q$ elements of $SL_2(q)$ have order divisible by p

[G&CIII, 1999] Conder & Leedham-Green announced an ingenious new way to handle white box $SL_2(q)$ using discrete logarithms in $GF(q)^*$

This led to white box “polynomial time” algorithms for all classical groups

$SL_2(q)$

Conder, Leedham-Green and O'Brien [TAMS, 2005] take any representation of $SL_2(q)$ in defining characteristic and recover the natural representation, thus providing a grey box $SL_2(q)$ algorithm using discrete logarithms

THESIS: Assuming an “ SL_2 -oracle” there exist polynomial time black box recognition algorithms for all classical groups

[B & Kantor, 2001] $SL_d(q)$ and $Sp_d(q)$

[B, 2003] $SU_d(q)$

[B & Kantor, 2005] $\Omega_d^\epsilon(q)$

Implementation

white box

- $SL_d(q)$, $Sp_d(q)$ (GAP)
- $Sp_d(q)$, $SU_d(q)$, $\Omega_d^\epsilon(q)$ (partial) (GAP)
- $SL_d(q)$ (Magma)

grey box

- $SL_2(q)$ (Magma)

black box

- $SL_d(q)$, $Sp_d(q)$ (q odd) (GAP & translated into Magma)

black box with oracle

- $SU_d(q)$ (partial) (GAP)

Current Goal

Provide complete, general purpose GAP implementations to constructively recognise $Sp_d(q)$ for q even or odd

Recent progress:

- An $SL_2(q)$ -oracle for GAP that handles inputs either as `white box` or `black box`
- Implementations for $Sp_4(q)$, all fields, all representations:
 - `white box` q odd
 - `white box` q even
 - `black box` q odd
 - `black box` q even