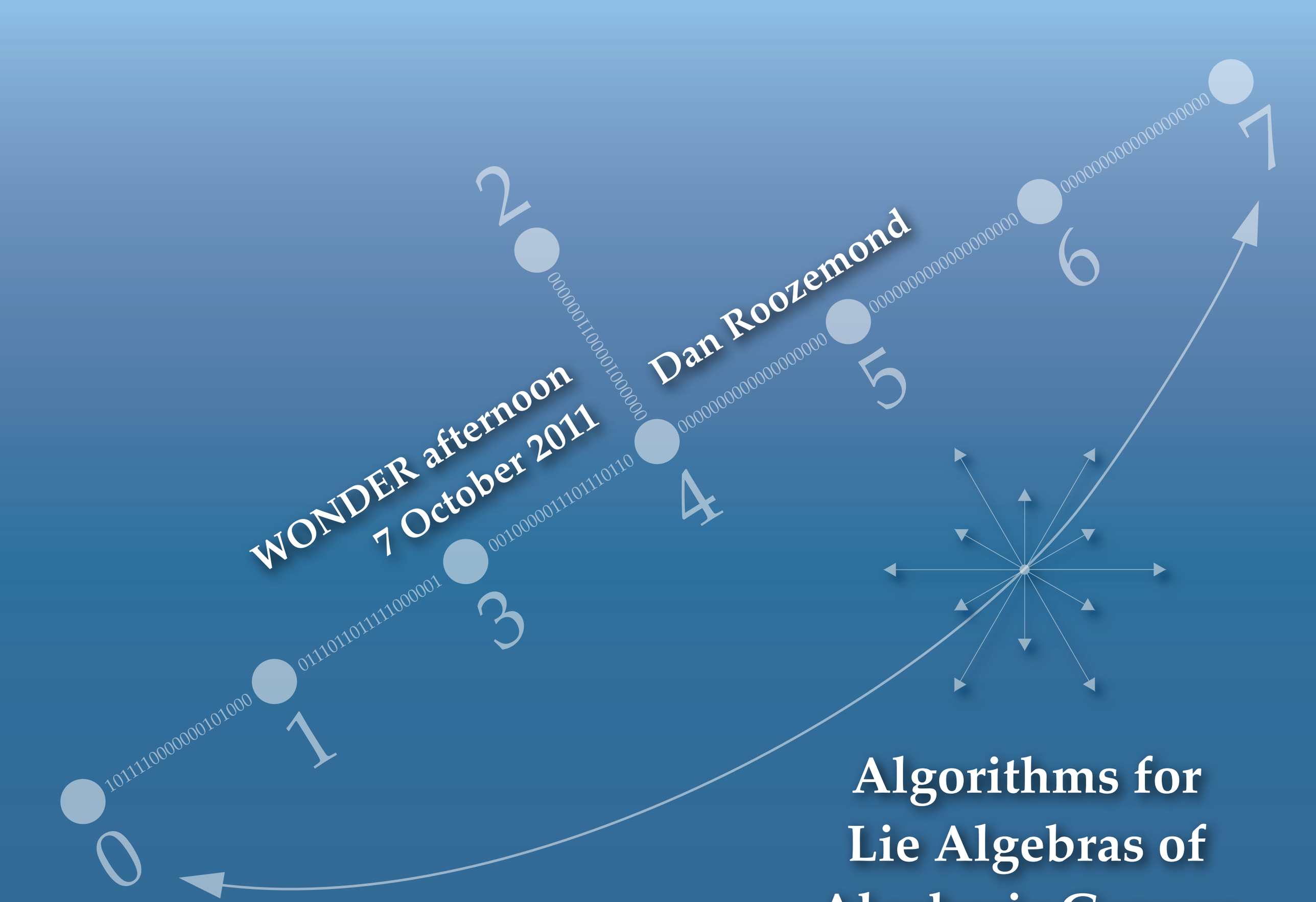


Algorithms for Lie Algebras of Algebraic Groups

Outline

- What is a Lie algebra?
- "Special" cases
- Recognising Lie algebras
- Applying new algorithms

What is a Lie Algebra?

Definition (*Lie algebra L*)

- Vector space $\mathbb{F}^n$
- With a multiplication $[\cdot,\cdot] : L \times L \to L$ that is
  - Bilinear,
  - Anti-symmetric, and
  - Satisfies the *Jacobi* identity

$$[x,[y,z]] + [y,[z,x]] + [z,[x,y]] = 0$$

What is a Lie Algebra?

Definition (*Lie algebra L*)

- Vector space $\mathbb{F}^n$
- With a multiplication $[\cdot,\cdot] : L \times L \to L$ that is
  - Bilinear,
  - Anti-symmetric, and
  - Satisfies the *Jacobi* identity

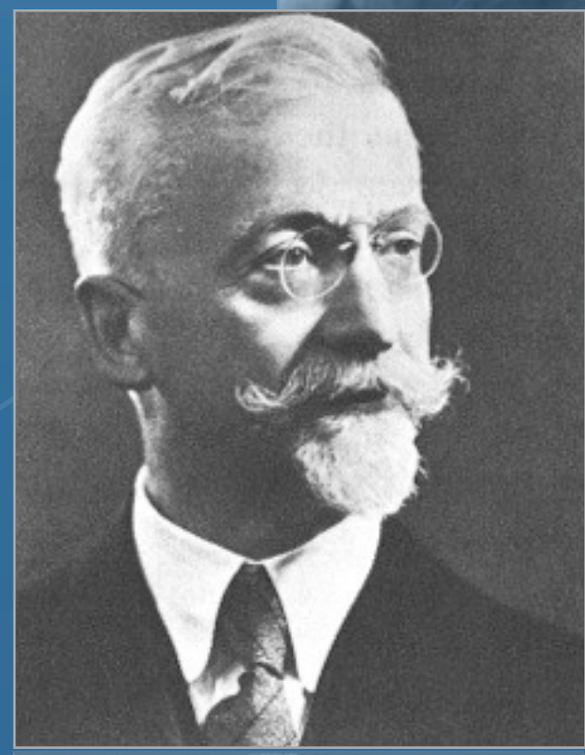
$$[x,[y,z]] + [y,[z,x]] + [z,[x,y]] = 0$$

Simple Lie algebras

Classification (Killing, Cartan)

For $\mathbb{F}$ algebraically closed and $\mathrm{char}(\mathbb{F}) = 0$ the only simple Lie algebras are:

A_n	$(n \geq 1)$	E_6, E_7, E_8
B_n	$(n \geq 2)$	F_4
C_n	$(n \geq 3)$	G_2
D_n	$(n \geq 4)$	



Why study Lie algebras?

- Study *groups* by their Lie algebras:
  - Simple algebraic group $G \leftrightarrow$ unique Lie algebra L
  - Many properties carry over to L
  - Easier to calculate in L
  - $G \leq \mathrm{Aut}(L)$, often even $G = \mathrm{Aut}(L)$
- Opportunities for:
  - Recognition,
  - Solving conjugation problems,
  - ...
- Because there are problems to be solved!
  - ... and there was a thesis to be written...

Example

- *Matrix* Lie algebra: elements are 2x2 matrices of trace 0, called $\mathfrak{sl}_2$
- Basis: $H = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, E = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, F = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$
- Multiplication: $[x, y] = xy - yx$

Definition (*Lie algebra L)*

- ✔ Vector space $\mathbb{F}^n$
- ✔ With a multiplication $[\cdot, \cdot] : L \times L \to L$ that is
  - ✔ Bilinear,
  - ✔ Anti-symmetric, and
  - ✔ Satisfies the *Jacobi* identity

$$[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0$$

Example

- *Matrix* Lie algebra: elements are 2x2 matrices of trace 0, called $\mathfrak{sl}_2$
- Basis: $H = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, E = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, F = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$
- Multiplication: $[x, y] = xy - yx$

The same Lie algebra

$[\cdot,\cdot]$	E	F	H
E	0	H	$-2E$
F	$-H$	0	$2F$
H	$2E$	$-2F$	0

Outline

- What is a Lie algebra?
- “Special” cases
- Recognising Lie algebras
- Applying new algorithms

The multiplication and the field

The multiplication and the field

The multiplication and the field

The multiplication and the field

Example, revisited

$\mathfrak{sl}_2(\mathbb{Q}) = \mathrm{A}_1^{\mathrm{SC}}(\mathbb{Q})$

$[\cdot,\cdot]$	E	F	H
E	0	H	$-2E$
F	$-H$	0	$2F$
H	$2E$	$-2F$	0

$\rightarrow$

$\mathrm{A}_1^{\mathrm{SC}}(\mathbb{F}_2)$

$[\cdot,\cdot]$	E	F	H
E	0	H	0
F	$-H$	0	0
H	0	0	0

$\cong$

$\mathrm{A}_1^{\mathrm{Ad}}(\mathbb{Q})$

$[\cdot,\cdot]$	E	F	H
E	0	$2H$	$-E$
F	$-2H$	0	F
H	E	$-F$	0

$\rightarrow$

$\mathrm{A}_1^{\mathrm{Ad}}(\mathbb{F}_2)$

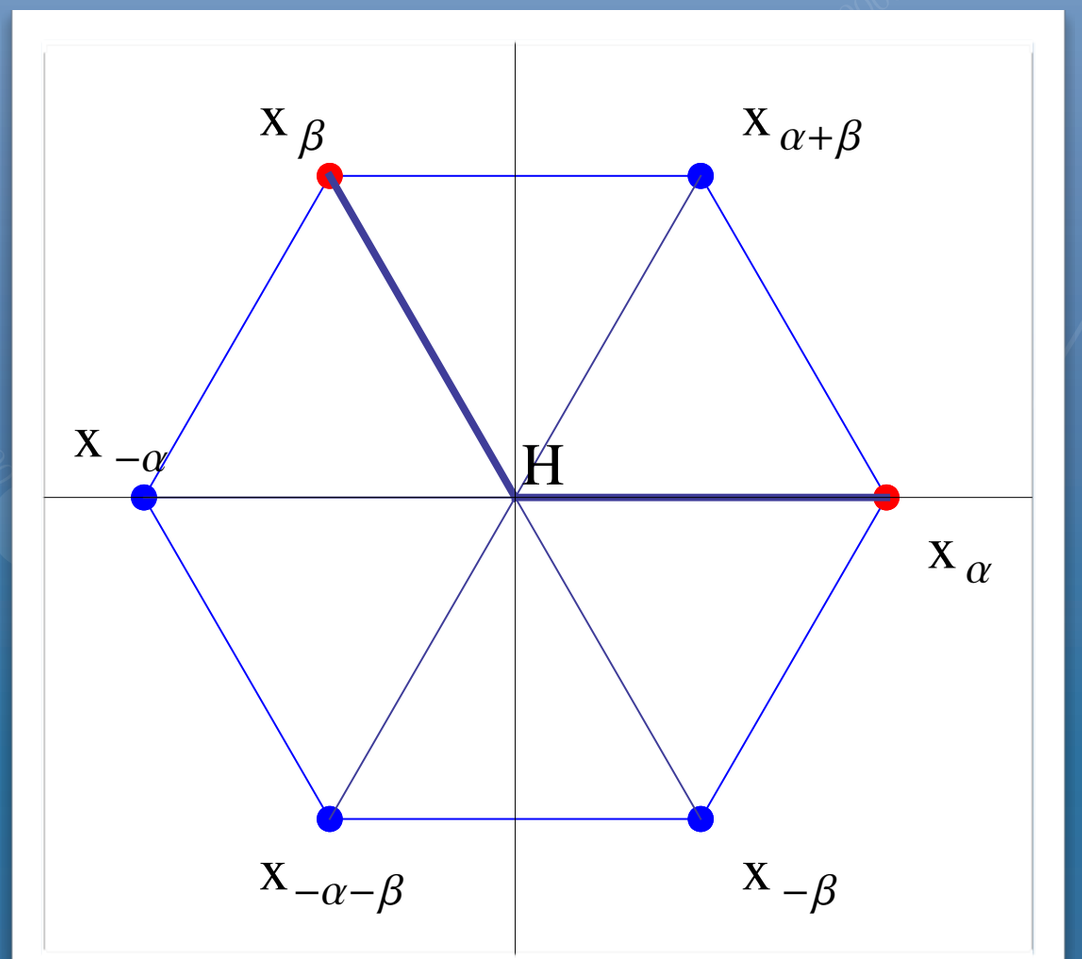
$[\cdot,\cdot]$	E	F	H
E	0	0	$-E$
F	0	0	F
H	E	$-F$	0

$\not\cong$

Outline

- What is a Lie algebra?
- "Special" cases
- Recognising Lie algebras
- Applying new algorithms

Chevalley Bases

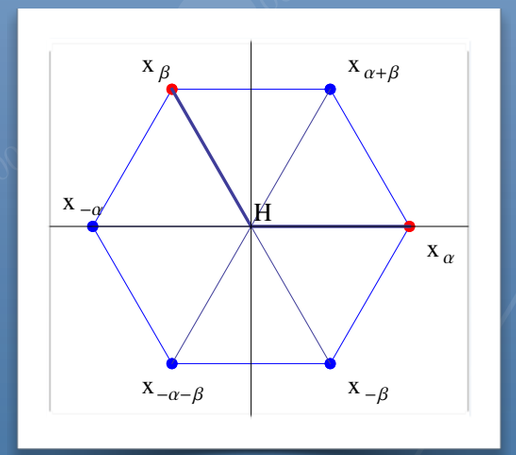


Many Lie algebras have a *Chevalley basis!*

Why Chevalley bases?

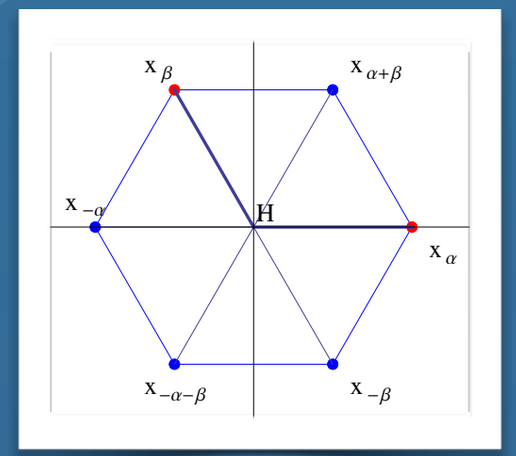
- A Chevalley basis for L certifiably states whether L is A_n, B_n, C_n, D_n, E_6, E_7, E_8, F_4, G_2 (or a combination thereof).
- And we can test isomorphism between two Lie algebras (and find isomorphisms!) by computing Chevalley bases.

Why Chevalley bases?

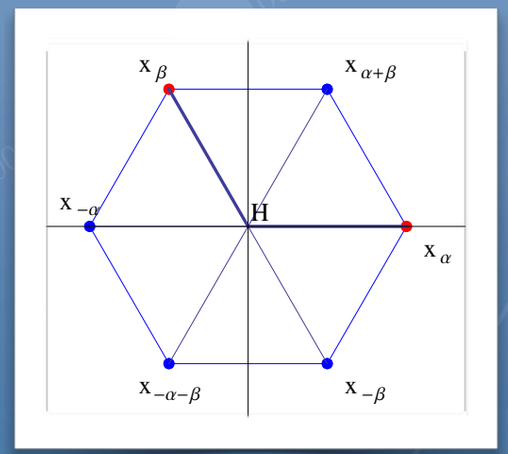


isomorphic!

equal

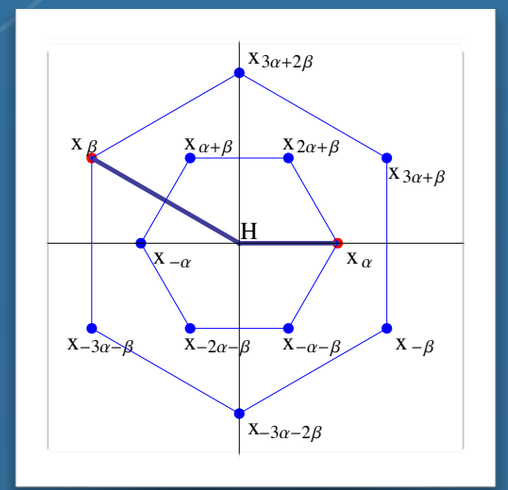


Why Chevalley bases?



not equal

non-isomorphic!



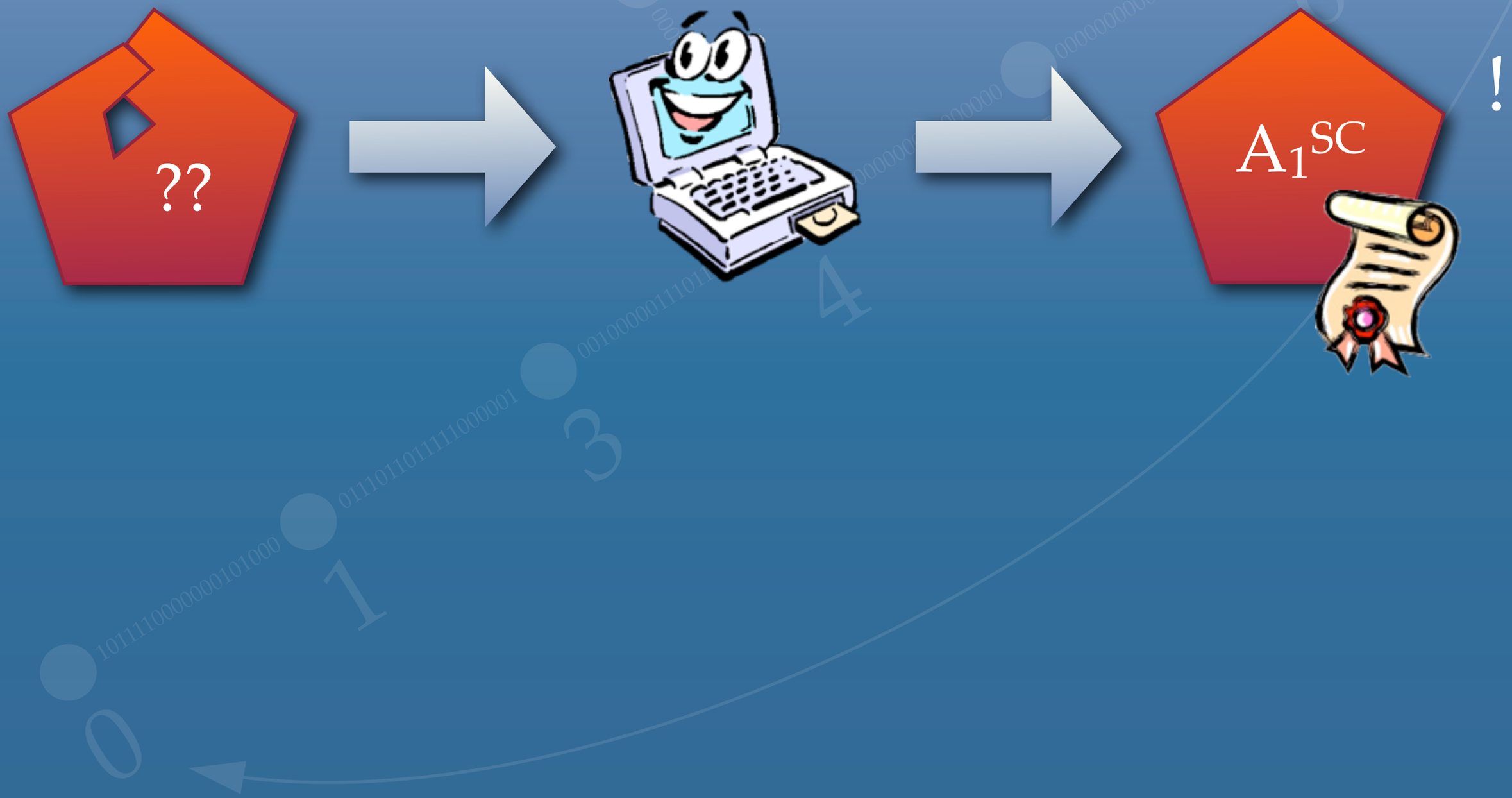
Computing Chevalley bases

Existing algorithms: Often (especially over $\mathbb{Q}$)

But sometimes (especially over $\mathbb{F}_2$)

New algorithms

In *almost all* cases, including characteristic 2:



Algorithms

- Carefully study the “special cases”,
- Use their quirks to handle them,
- Several algorithms per type,
- Quite “high-level”.

Algorithms

x findframe_A3_char2.m

```
112   L := CBD`L;
113
114   //Clean the returned vectors; make them into vectorspaces...
115   zr := [ i : i in [1..#ev] | IsZero(ev[i]) ];
116   es := [ es[i] : i in [1..#es] | i notin zr ];
117   ev := [ ev[i] : i in [1..#ev] | i notin zr ];
118   dubs := { { i : i in [1..#ev] | ev[i] eq ev[j] } : j in [1..#ev] };
119   dubs := [ SetToSequence(d) : d in dubs ];
120   esvs := [ VectorSpaceWithBasis([ Vector(e) : e in es[d] ]) : d in dubs] ;
121
122   //Check
123   if { Dimension(V) : V in esvs } ne {2} then
124      error "A_3(2): Unexpected form of diagonal action in Der(A_3(2)) (1)";
125   end if;
126
127   //Pull back these spaces to A3, and insert them into the CBD
128   //Note that on the pullback, we may inadvertedly add a bit of
129   //  centre into the eigenspaces.
130   z := L!(Basis(Centre(L))[1]);
131   BasisH := [ h : h in CBD`SCSAVectors ];
132
133   pullback := function(espc)
134      local y, vswb, LHS, Ms, es, ev, dims, p;
135
136      vswb := VectorSpaceWithBasis(
137         [ Vector(backtoL(e)) : e in Basis(espc) ]
138         cat [ Vector(z) ]);
139      Ms := [ Matrix([ Coordinates(vswb, Vector(L!b*L!h)) : b in Basis(vswb) ])
140         : h in BasisH ];
141      es,ev := simdiag(Ms);
142
143      dims := [ Dimension(e) : e in es ];
144      if Seqset(dims) ne {1,2} then
145         error "Could not pullback from Der(A3) to A3.";
146      end if;
147      p := Position(dims,2);
148
149      return [ Vector(b*Matrix(Basis(vswb))) : b in Basis(es[p]) ];
150   end function;
151
152   newspcs := [ pullback(e) : e in esvs ];
153   insert_found_frame(~CBD, &cat(newspcs), false);
154 end procedure;
155
156 findframe_A3_nonAd_char2_usingDerL := procedure(~CBD)
157   local F,L, H, es, ev, backtoL, ok;
158
159   L := CBD`L;
160   H := CBD`H;
161   F := CBD`F;
162
```

Magma 2010

```
 movel (A1)+,D7 |andl #0x8000FFFF,D2
 tstl D3 |bpl 5f |moveb #"-",(A0)+ |moveb #" ",(A0)+
5: cmpw #2,D5 |beq 6f
 subw #0x2001,D2 |addw #0x3FFF,D2
 swap D2 |clrw D2 |tstl D3 |bpl 3f |bset #31,D2
 negl D7 |negxl D4 |negxl D3
3: lsll #1,D7 |roxll #1,D4 |roxll #1,D3
6: cmpw #1,D5 |bpl 4f |moveq #3,D5 |tstl D7
 bpl 1f |addql #1,D4 |bcc 1f |addql #1,D3 |bcc 1f
 roxrl #1,D3 |roxrl #1,D4 |bra 1f
4: moveq #4,D5
1: movel D2,D0 |movel D3,D2 |movel D4,D3
 movel D7,D4 |moveq #8,D6 |moveb #"0",(A0)+
 moveb #"x",(A0)+
2: roll #4,D0 |moveq #0xF,D1 |andl D0,D1
 moveb (A2,D1),(A0)+ |subql #1,D6 |bgt 2b
 moveb #",",(A0)+ |subql #1,D5 |bgt 1b
 clrb -(A0) |movel A0,D0 |rts
;
hexus: .ASCII "0123456789ABCDEF"
;
_number: movel A1,nraddr0 |lea araddr0,A2       ;number. Convert nr(A1) to
 movel A0,(A2)+ |swap D6 |movew D6,D5   ;string starting in (A0). Address
 cmpw #3,D5 |bpl 8f |bmi octpr |moveq #5,D5     ;terminating 0 on exit in D0.
8: cmpw #26,D5 |bmi 9f |moveq #26,D5    ;D6 contains prec, raton prec.
9: movew D5,nprec0+2 |clrl rr10 |clrl rr20      ;Default prec=5, maximum 26.
 clrw nprec0 |clrl rr30 |swap D6 |movel A0,(A2) ;arradr+4=start number,
 tstw D6 |ble 2f |bsr raton |tstl D7 |beq 2f    ;not including sign.
 movel araddr0,A0 |lea iup0+4,A1 |movel (A1),D3
 movel -(A1),D2 |bpl 0f |negl D3                ;Supply -, but not +.
 negxl D2 |moveb #"-",(A0)+                     ;-
 moveb #" ",(A0)+ |movel A0,araddr0+4
0: moveq #0,D1 |bsr transip     ;D0=last A0
 lea ido0+4,A1 |moveq #1,D6     ;do not use D0 here (for evt exit)
 movel (A1),D3 |movel -(A1),D2 |bne 1f  ;no / needed if ido0=1
 subl D3,D6 |beq nrexitp
1: moveb #"/",(A0)+            ;/
 clrl D1 |bsr transip |bra nrexitp
2: movel nraddr0,A3 |jsr _Fget
 movel araddr0,A0 |tstl D1 |bpl 3f
 negl D3 |negxl D2 |negxl D1
 moveb #"-",(A0)+ |moveb #" ",(A0)+            ;-
 movel A0,araddr0 |movel A0,araddr0+4
3: bclr #15,D0 |beq 10f |moveb #"1",(A0)+ |bra nrexit
10: lea rr10,A3 |jsr _Fput                    ;pos. nr to rr10
 movew nprec0+2,D7 |moveq #53,D6 |mulu D6,D7 |asrl #4,D7        ;*53/16=
 addw #0x2001,D7 |subw D0,D7 |bmi 4f    ;conversion digits-bits prec.
 bsr integq |tstl D7 |beq 4f
 lea rr20,A0 |jsr _Fcfi
 movel araddr0,A0 |movel D7,D3 |movel D6,D2
 movel D5,D1 |bsr transi |bra nrexitp
4: movew rr10+2,D0 |subw #0x2001,D0
 movew #0x4D10,D1 |muls D0,D1 |swap D1  ;4D10=log2. Estimate log
 extl D1 |addql #1,D1 |movew D1,iex0
 bpl 5f |negl D1
5: lea iup0,A2 |movel D1,(A2) |lea rr20,A0 |lea Ten,A1 |jsr _Fexp
 movel A0,A2 |lea rr10,A0 |movel A0,A1
 tstw iex0 |bmi 6f |jsr _Fdiv |bra 7f
6: jsr _Fmul
7: lea rr10,A3 |jsr _Fget
 cmpw #0x1FFD,D0 |bgt 9f |bne 8f |movel #0x66666666,D4
 cmpl D4,D1 |bgt 9f |bne 8f |tstl D2 |bmi 9f    ;compare with 0.1
;
```

Schoonschip 1960's

Outline

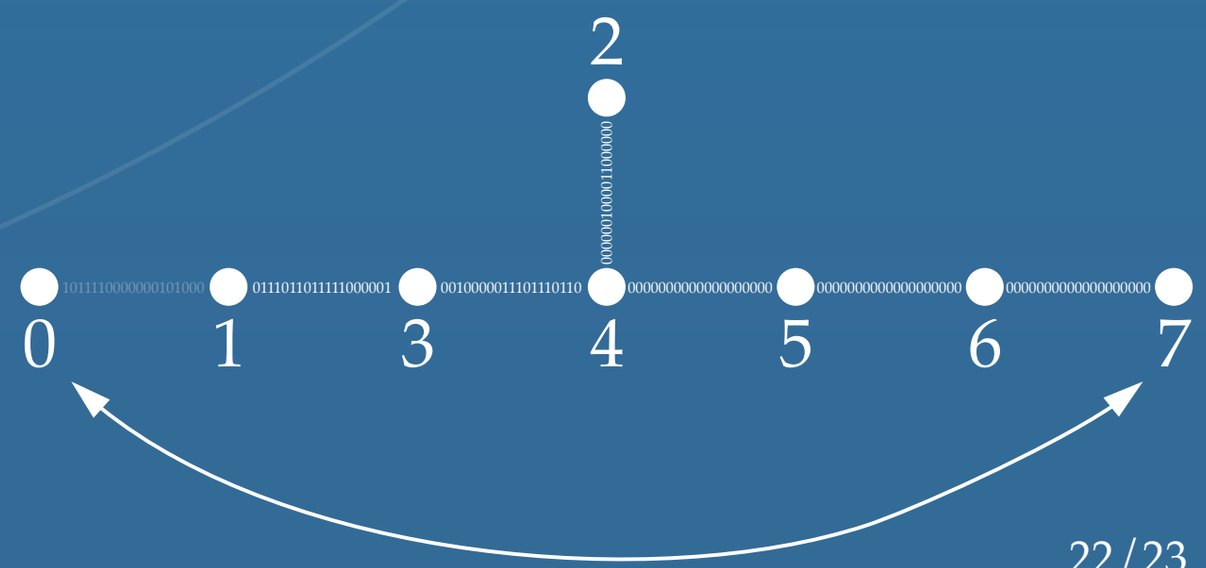
- What is a Lie algebra?
- "Special" cases
- Recognising Lie algebras
- Applying new algorithms

Straightforward applications

- Recognition of Lie algebras,
- Isomorphism testing and construction for Lie algebras,
- Indirectly: matrix group recognition.

Distance-Transitive Graphs

- Long-term effort (Cohen, Liebeck, Saxl) to "classify the graphs on which an almost simple group of exceptional Lie type acts distance-transitively".
- One of open cases: $H = {}^2\mathrm{A}_7(2^2)$ in $G = \mathrm{E}_7(2)$.
- Use the new algorithms to construct in Magma,
- and search for different H-orbits on Lie(G),
- proving non-existence.



Outline

- What is a Lie algebra?
- “Special” cases
- Recognising Lie algebras
- Applying new algorithms
- Thank you!