Computerpraktikum Algebra

Thema 4 - Graphen und Lie-Algebren

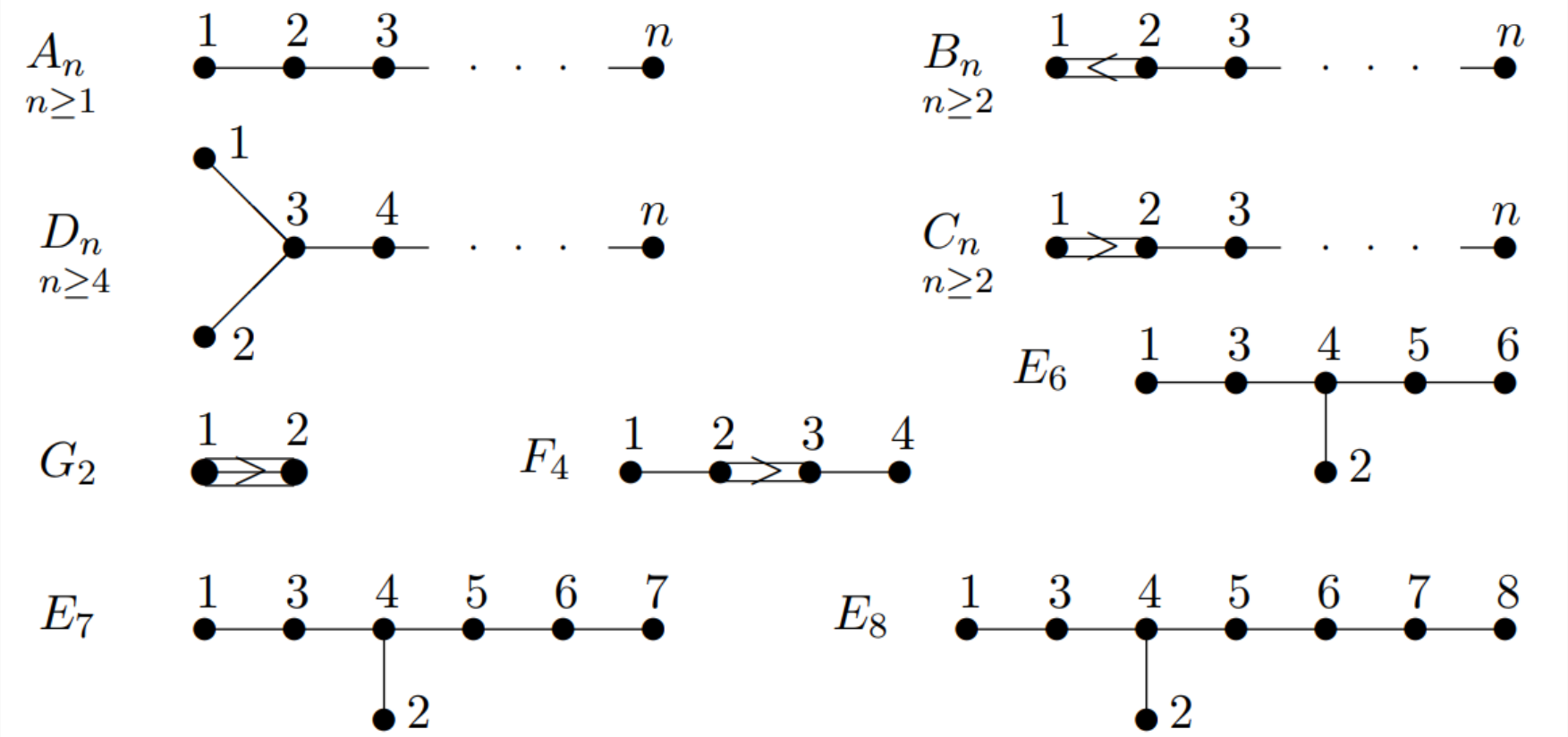
Pascal Bauer, Raphael Millon, Florian Haas

Sommersemester 2020

Table of contents

Theorie

- Wir betrachten Dynkin-Diagramme und die daraus konstuierbaren Gruppen.
- Dynkin-Diagramm sind spezielle Graphen, mit eventuell mehrfachen gerichteten Kanten.



Theorie

- Zu einem Graphen Γ kann eine Matrix $A(\Gamma) = (a_{ij})_{1 \leq i,j \leq n}$ wie folgt definiert werden:
- 1. Setze $a_{ii} = 2$ auf der gesamten Diagonalen.
  2. Setze $a_{ij} = 0$, falls $i \neq j$ und die Ecken i und j nicht verbunden sind.
  3. Setze $a_{ij} = a_{ji} = -1$, falls $i \neq j$ und die Ecken i und j einfach verbunden sind.
  4. Setze $a_{ij} = -d,\ a_{ji} = -1$, falls $i \neq j$ und die Ecken i und j d-fach in Richtung i verbunden sind.
- Für F_4 ergibt sich zum Beispiel

$$A(F_4) = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -2 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}.$$

- Somit kodieren sich Γ und $A(\Gamma)$ gegenseitig.

Theorie

- Für festes Γ definieren wir nun für $1 \leq i \leq n$ lineare Abbildungen gegeben durch $w_i(e_j) := e_j - a_{ij}e_i$ oder äquivalent $M_{\mathbb{Q}}(w_i) = I_n - E_{ii}A(\Gamma)$.
- Da $M_{\mathbb{Q}}(w_i)^2 = I_n - 2E_{ii}A(\Gamma) + (E_{ii}A(\Gamma))^2 = I_n$ ist die Abbildung $w_i \in GL_n(\mathbb{Q})$ und insbesondere diagonalisierbar mit Eigenwerten $\in \{-1, 1\}$.
- Jede Abbildung w_i beschreibt also eine Spiegelung.
- In unserem Projekt betrachteten wir die von allen w_i erzeugte Gruppe $W = \langle w_1, \ldots, w_n \rangle \subseteq GL_n(\mathbb{Q})$.
- Zudem wird $\Phi = \{w(e_j) \mid w \in W, 1 \leq j \leq n\}$ berechnet. Insbesondere ist Φ genau dann endlich wenn auch W endlich ist.

Showcase

Graph	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
$\lvert\Phi\rvert$	2	6	12	20	30	42	56	72
$\lvert W\rvert$	2	6	24	120	720	5040	40320	362880
Graph	–	B_2	B_3	B_4	B_5	B_6	B_7	B_8
$\lvert\Phi\rvert$	-	8	18	32	50	72	98	128
$\lvert W\rvert$	-	8	48	384	3840	46080	645120	10321920
Graph	–	C_2	C_3	C_4	C_5	C_6	C_7	C_8
$\lvert\Phi\rvert$	-	8	18	32	50	72	98	128
$\lvert W\rvert$	-	8	48	384	3840	46080	645120	10321920
Graph	–	–	–	D_4	D_5	D_6	D_7	D_8
$\lvert\Phi\rvert$	-	-	-	24	40	60	84	112
$\lvert W\rvert$	-	-	-	192	1920	23040	322560	5160960
Graph	–	–	–	–	–	E_6	E_7	E_8
$\lvert\Phi\rvert$	-	-	-	-	-	72	126	240
$\lvert W\rvert$	-	-	-	-	-	51840	2903040	696729600
Graph	–	G_2	–	F_4	–	–	–	–
$\lvert\Phi\rvert$	-	12	-	48	-	-	-	-
$\lvert W\rvert$	-	12	-	1152	-	-	-	-

Codebeispiele

Unser Code ist Open-Source verfügbar auf Github:

`https://github.com/raphaelMi/computerpraktikum-algebra`

Implementiert haben wir vier Funktionen:

- `gmat(X, n)`
- `glin(graph_matrix)`
- `gphi(linear_function_matrices)`
- `gw(linear_function_matrices)`

Codebeispiele gmat

```
# Generate matrix A(gamma)
for i in [1..n] do
        row := [];

        for j in [1..n] do

                if i = j then
                        row[j] := 2;
                else
                        row[j] := _grel(X, n, i, j);

                fi;

        od;

        mat[i] := row;
od;
```

Codebeispiele glin

```
for i in [1..n] do
        w_i := []; # Matrix for linear function w_i

        for k in [1..n] do
                row := []; # Row of w_i matrix

                for j in [1..n] do
                        if k = j then # Compute the unity vector e_j
                                e_j := 1;
                        else
                                e_j := 0;
                        fi;

                        if k = i then # Compute the unity vector e_i
                                e_i := 1;
                        else
                                e_i := 0;
                        fi;

                        row[j] := e_j - mat[i][j] * e_i; # Fill the matrix element-wise with the formula specified in the excersise
                od;
                w_i[k] := row;
        od;
```

Codebeispiele gphi

```
# Initialize phi with e_i
for i in [1..n] do
        e_i := [];

        for j in [1..n] do
                if i = j then # Generate the e_i
                        e_i[j] := 1;
                else
                        e_i[j] := 0;
                fi;
        od;

        AddSet(phi, e_i);

od;
```

Codebeispiele gphi

```
last_count := Length(phi); # Monitoring the count of phi

# Now we'll call phi on the elements (my multiplying the matrices) with each other until no new elements get added
while true do
        for ph in Iterator(phi) do
                for j in [1..n] do
                        AddSet(phi, matrices[j] * ph);
                od;
        od;

        # Check whether new elements got added
        if last_count = Length(phi) then
                break; # No new elements, break here
        else
                last_count := Length(phi);
        fi;
od;
```

Codebeispiele gw

```
additions := Immutable(w); # Elements that got added with the last iteration
newAdditions := Set([]); # Monitor the new elements that got added

while not Length(additions) = 0 do # Loop until no new elements got added

        for w1 in Iterator(additions) do
                for w2 in Iterator(w) do

                        prod := w1*w2; # Multiply the new matrices to the current set w

                                if not prod in w then
                                    AddSet(w, prod);
                                    AddSet(newAdditions, prod); # If the matrix is not in w, save it here
                                fi;
                od;
        od;

        additions := Immutable(newAdditions); # The additions of this iterations are the new additions
        newAdditions := Set([]); # Reset those
od;
```

Fragen ... und Antworten

```
presentation_running := true;

while presentation_running do
present();
od;

process_questions();
```