On a framework for Hillman–Grassl algorithms

NUMATA, Yasuhide

Shinshu univ.

(joint work w/ S. Okamura and K. Nakada.)
Outline

1 Introduction
   - Notation
     - The classical Hillman–Grassl algorithm

2 Our framework
   - Prototypical example
   - A H–G graph
   - Main results and Application
Partitions and Young diagrams

Let $\lambda$ be a partition of an integer, i.e.,

$$(\lambda_1, \lambda_2, \ldots, \lambda_l)$$

such that

$$i \leq i' \implies \lambda_i \geq \lambda_{i'}.$$ 

We regard $\lambda$ as the set

$$\{ (i, j) \mid 1 \leq j \leq \lambda_i \}$$

of boxes (or cells), and we use so-called English notation.
Hooks of Young diagrams

Let $\lambda'$ be the transposed Young diagram of $\lambda$, i.e.,

$$\{ (j, i) \mid (i, j) \in \lambda \}.$$ 

$$\lambda'_j = \# \text{ of boxes in the } j\text{-th column of } \lambda.$$ 

For $(i, j) \in \lambda$, we define the hook at $(i, j)$ of $\lambda$ by

$$H(i, j) = \{ (i, j') \in \lambda \mid j \leq j' \leq \lambda_i \} \cup \{ (i', j) \in \lambda \mid i \leq i' \leq \lambda'_j \}.$$ 

- $(i, \lambda_i)$ is the easternmost box in the hook $H(i, j)$.
- $(\lambda'_j, j)$ is the southernmost box in the hook $H(i, j)$. 
Consider a path from \((i, \lambda_i)\) to \((\lambda'_j, j)\) such that the direction of each step is south (\(\downarrow\)) or west (\(\leftarrow\)). In this talk, we call it a zigzag hook at \((i, j)\).

\[
\text{# of boxes in a zigzag hook at } (i, j) = \text{# of boxes in the hook } H(i, j) \text{ at } (i, j).
\]

- The west-first path is the hook at \((i, j)\).
- The south-first path is the rim hook at \((i, j)\).
Reverse plane partition

We call a map

\[ T : \lambda \rightarrow \mathbb{N} = \mathbb{Z}_{\geq 0} \]
\[ (i, j) \mapsto T_{ij} \]

such that

\[ i \leq i' \implies T_{ij} \leq T_{i'j} \]
\[ j \leq j' \implies T_{ij} \leq T_{ij'} \]

a reverse plane partition (RPP) on \( \lambda \).
Let \( \text{rpp}(\lambda) \) be the set of RPPs on \( \lambda \).
Reverse plane partition for an arbitrary poset

A Young diagram $\lambda$ is a poset by the following order:

$$(i, j) \geq (i', j') \iff i \leq i' \text{ and } j \leq j'$$

In this sense,

$\begin{align*}
T: \lambda \to \mathbb{N} \text{ is a RPP on } \lambda \\
\iff T: \lambda \to \mathbb{N} \text{ is an order-reversing map.}
\end{align*}$

For an arbitrary poset $P$, we call $T: P \to \mathbb{N}$ a RPP if $T$ is an order-reversing map. Let $\text{rpp}(P)$ be the set of RPPs on $P$. 
Introduction

The classical Hillman–Grassl algorithm

Outline

1 Introduction
   - Notation
   - The classical Hillman–Grassl algorithm

2 Our framework
   - Prototypical example
   - A H–G graph
   - Main results and Application
What is the Hillman–Grassl algorithm?

The classical H–G algorithm

- is an algorithm to obtain
  - a sequence of boxes of \( \lambda \)
  - from a RPP \( T \) on \( \lambda \).
- induces a weight-preserving bijection between
  - the set of RPP on \( \lambda \) and
  - the set of multisets of hooks of \( \lambda \)
  
  for each Young diagram \( \lambda \).

As a corollary to the bijection, we obtain the hook length formula.
An algorithm to remove a zigzag hook

**Input** a RPP $T$ on $\lambda$ such that $T_{1,\lambda_1} > 0$.

**Output** $T'$ and $j$.

**Proc.**

1. Let $i = 1$, $j = \lambda_1$, $Z = \emptyset$.
2. While $(i, j) \in \lambda$, do the following:
   1. Append $(i, j)$ to $Z$.
   2. If $T_{i,j-1} = T_{i,j}$, then
      1. add $-1$ to $j$;
      else
      1. add $1$ to $i$.
3. Let $T'_{ij} = \begin{cases} T_{ij} & (i, j) \notin Z \\ T_{ij} - 1 & (i, j) \in Z. \end{cases}$
An algorithm to remove a zigzag hook

Remark

This algorithm is an invertible algorithm.

Remark (on the output $T'$)

The output $T'$ is a reverse plane partition on $\lambda$.
The difference between $T'$ and $T$ is a zigzag hook at $(1, j)$.

Remark (on the output $j$)

If we apply this algorithm consecutively, then the outputs $j_1, j_2, \ldots$ satisfy $j_1 \geq j_2 \geq \cdots$. 
Introduction

The classical Hillman–Grassl algorithm

The classical H–G algorithm

Input a RPP $T$ on $\lambda$.

Output a sequence $\mathcal{H}$ of boxes of $\lambda$.

Proc.

1. Let $\mathcal{H}$ be the empty sequence.
2. For $i = 1, 2, \ldots$, do the following:
   1. While $T_{i\lambda i} > 0$, do the following:
      1. Let $T'$ and $j$ be the pair obtained from $T$ by the algorithm to remove a zigzag hook. (Since $T_{i'j} = 0$ for $i' < 0$, forget these rows.)
      2. Let $T$ be $T'$ (as a RPP on $\lambda$).
      3. Append $(i, j)$ to $\mathcal{H}$. 
The classical H–G algorithm

Remark
Since the algorithm to remove a zigzag hook is invertible, the algorithm is also invertible. The resulting sequence $\mathcal{H}$ is ordered in some order. Hence we can regard it as a multiset of boxes.

Remark
The analogues of the H–G algorithm for the other poset is known. E.g., shifted Young diagrams.

Our aim is
- to describe analogues of H–G algorithms uniformly, and
- to generalize them.
Outline

1 Introduction
   - Notation
   - The classical Hillman–Grassl algorithm

2 Our framework
   - Prototypical example
   - A H–G graph
   - Main results and Application
Recall an algorithm to remove a zigzag hook

**Input** a RPP $T$ on $\lambda$ such that $T_{1,\lambda_1} > 0$.

**Output** $T'$ and $j$.

**Proc.**

1. Let $i = 1$, $j = \lambda_1$, $Z = \emptyset$.
2. While $(i, j) \in \lambda$, do the following:
   1. Append $(i, j)$ to $Z$.
   2. If $T_{i,j-1} = T_{i,j}$, then
      1. add $-1$ to $j$;
      else
      1. add $1$ to $i$.
3. Let $T'_{i,j} = \begin{cases} T_{ij} & (i,j) \notin Z \\ T_{ij} - 1 & (i,j) \in Z \end{cases}$.
Recall an algorithm to remove a zigzag hook

**Input** a RPP $T$ on $\lambda$ such that $T_{1,\lambda_1} > 0$.

**Output** $T'$ and $j$.

**Proc.**
1. Let $i = 1$, $j = \lambda_1$, $Z = \emptyset$.
2. While $(i, j) \in \lambda$, do the following:
   1. Append $(i, j)$ to $Z$.
   2. If $T_{i,j-1} = T_{i,j}$, then
      1. add $-1$ to $j$;
      
      else
         1. add $1$ to $i$.

3. Let $T_{ij}' = \begin{cases} T_{ij}, & (i, j) \not\in Z \\ T_{ij} - 1, & (i, j) \in Z. \end{cases}$
On a framework for Hillman–Grassl algorithms

Our framework

Prototypical example

Refactor the primitive part of the algorithm

**Input** a RPP $T$ on $\lambda$ such that $T_{1,\lambda_1} > 0$.

**Output** $Z$ and $j$.

**Proc.**

1. Let $i = 1$, $j = \lambda_1$, $Z = \emptyset$.
2. While $(i, j) \in \lambda$, do the following:
   1. Append $(i, j)$ to $Z$.
   2. If $T_{i,j-1} = T_{i,j}$, then
      1. Let $j$ be $j - 1$
      else
      1. Let $i$ be $i + 1$. 
Refactor the primitive part of the algorithm

Input  a RPP $T$ on $\lambda$ such that $T_{1,\lambda_1} > 0$.
Output  $Z$ and $j$.
Proc.  1. Let $c = (1, \lambda_1)$, $Z = \emptyset$.
       2. While $c \in \lambda$, do the following:
          1. Append $c$ to $Z$.
          2. Let $c'$ be the box in the next hook in the same row as $c$. If $T_{c'} = T_c$, then
             1. move $c$ to the box of the next hook in the same row;
             else
             1. move $c$ to the next box of the same hook.
A H–G graph

To describe the primitive part, we rearrange the boxes in $\lambda$. Let

$$\Gamma = \left\{ (i, j) \mid \begin{array}{l} i \in \{ 1, 2, \ldots, \lambda_1 \}, \\ j \in \{ i, i + 1, \ldots, \#H(1, \lambda_i - i + 1) \} \end{array} \right\} \subset \mathbb{Z}^2.$$ 

Let $v$ be the map

$$v: \Gamma \to \lambda$$

$$(i, j) \mapsto (1 + j - i, \lambda_1 + 1 - i).$$

Add arrows $(i, j) \rightarrow (i + 1, j)$ and $(i, j) \rightarrow (i + 1, j + 1)$ to $\Gamma$. We call the labeled digraph $(\Gamma, v: \Gamma \to \lambda)$ a H–G graph.
Input a RPP $T$ on $\lambda$ such that $T_{1,\lambda_1} > 0$.

Output $Z$ and $c$.

Proc.

1. Let $c = (1, 1)$, $Z = \emptyset$.
2. While $c \in \Gamma$, do the following:
   1. Append $v(c)$ to $Z$.
   2. Let $c \rightarrow c'$, $c \starightarrow c''$. If $T(v(c)) = T(v(c'))$, then
      1. move $c$ via $\star$;
   else
      1. move $c$ via $\rightarrow$. 
Let $P$ be an arbitrary poset.
For any map $v: \Gamma \to P$, we can run the algorithm.
The algorithm is, however, not nice.
What does ‘nice’ mean...

- The output $T'$ should be a RPP on $P$.
- This algorithm should be an invertible algorithm.
- If we apply this algorithm consecutively, then the resulting boxes should be ordered.

We introduce some (minimal) condition for the map $v: \Gamma \to P$, to make the algorithm nice.
Outline

1 Introduction
   - Notation
   - The classical Hillman–Grassl algorithm

2 Our framework
   - Prototypical example
   - A H–G graph
   - Main results and Application
Fix nonnegative integers \( r, h_1, \ldots, h_r \).

Let

\[
\Gamma = \left\{ (i, j) \in \mathbb{Z}^2 \mid i \in \{1, \ldots, r\}, \ j \in \{i, i + 1, \ldots, h_i\} \right\}.
\]

Let \( \Delta' = \left\{ ((i, j), (i, j + 1)) \in \Gamma^2 \right\} \).

Fix a subset \( \Delta'' \subset \left\{ ((i, j), (i + 1, j + 1)) \in \Gamma^2 \right\} \).

We regard \( \Gamma \) as the digraph such that

- the set of vertices is \( \Gamma \);
- the set of arrows is \( \Delta' \cup \Delta'' \).
Let $P$ be a finite poset with the relation $\leq$. We write $x \prec y$ to denote that $x$ is covered by $y$. Fix a map $v: \Gamma \to P$. 
A technical notation to describe our condition

We call a quadruple \(((i, j), (i, j'); (i + h, j + h), (i + h, j' + h))\) of elements in \(\Gamma\) a \textit{ladder} if

1. \(j < j'\),
2. \(v(i + s, j + s) > v(i + s, j' + s)\) for \(s \in \{0, 1, \ldots, h\}\),
3. \((i + s, j + s) \rightarrow (i + s + 1, j + s + 1)\) for \(s \in \{0, 1, \ldots, h - 1\}\),
4. \((i + s, j' + s) \rightarrow (i + s + 1, j' + s + 1)\) for \(s \in \{0, 1, \ldots, h - 1\}\).

We define sets \(\tilde{\Xi}(i; j, j')\) and \(\hat{\Xi}(i; j, j')\) of ladders by

\[
\tilde{\Xi}(i; j, j') = \left\{ T \in \Gamma^2 \mid (T; (i, j), (i, j')) \text{ is a ladder} \right\}, \quad \text{and}
\hat{\Xi}(i; j, j') = \left\{ B \in \Gamma^2 \mid ((i, j), (i, j'); B) \text{ is a ladder} \right\}.
\]
Let \( \tilde{\Pi}((i, j), (i', j')) \) be the set of paths from \((i, j)\) to \((i', j')\) in \(\Gamma\).
We define \(\Pi((i, j), (i', j'))\) to be the set
\[
\left\{ ( (i_1, j_1), \ldots, (i_l, j_l) ) \in \tilde{\Pi}((i, j), (i', j')) \mid v(i_t, j_t) \neq v(i'_t, j'_t) \right\}.
\]
We also define
\[
\Pi = \bigcup_{i=1}^{r} \Pi((1, 1), (i, h_i)),
\]
\[
\bar{\Pi}(i, j) = \Pi((1, 1), (i, j)),
\]
\[
\hat{\Pi}(i, j) = \bigcup_{i' = i}^{r} \Pi((i, j), (i', h_{i'})).
\]
Hooks

For \((i, j) \in \Gamma\), we define \(\tilde{H}(i, j)\) and \(\hat{H}(i, j)\) by

\[
\begin{align*}
\tilde{H}(i, j) &= \{ v(k, k) \mid k \in \{1, 2, \ldots, i\} \} \\
&\quad \cup \{ v(i, k) \mid k \in \{i, i + 1, \ldots, j\} \}, \\
\hat{H}(i, j) &= \{ v(i, k) \mid k \in \{j, j + 1, \ldots, h_i\} \}.
\end{align*}
\]

For \(i \in \{1, 2, \ldots, r\}\), we define the hook \(H_{v(i,i)}\) at \(v(i, i)\) by

\[
H_{v(i,i)} = \tilde{H}(i, i) \cup \hat{H}(i, i) = \tilde{H}(i, h_i).
\]
On a framework for Hillman–Grassl algorithms
Our framework
A H–G graph

Definition

We call $\left(\Gamma, \Delta, \nu : \Gamma \to P\right)$ a $H–G$ graph for a finite poset $P$ if

1
2
3
4
5
6
7
Definition

We call \((\Gamma, \Delta, v : \Gamma \to P)\) a \textit{H–G graph} for a finite poset \(P\) if \(v(1, 1)\) is the maximum of \(\hat{H}(1, 1)\).
Definition

We call \((\Gamma, \Delta, v : \Gamma \to P)\) a \emph{H–G graph} for a finite poset \(P\) if

1. If \((i, j) \xrightarrow{\rightarrow} (i + 1, j + 1)\), then the following hold:
   1. \(\{ x \mid v(i, j) \preceq x \} \setminus \hat{H}(i, j) = \{ v(i + 1, j + 1) \}\).
   2. \(\{ x \mid x \preceq v(i + 1, j + 1) \} \setminus \hat{H}(i + 1, j + 1) = \{ v(i, j) \}\).
   3. \(v(i + 1, j + 1) \notin \hat{H}(i, j)\).
   4. \(v(i + 1, j + 1)\) is the maximum of \(\hat{H}(i + 1, j + 1)\).
On a framework for Hillman–Grassl algorithms

Our framework

A H–G graph

Definition

We call \((\Gamma, \Delta, \nu: \Gamma \to P)\) a \textit{H–G graph} for a finite poset \(P\) if

1. \(\hat{H}(i, j) = \emptyset\).

2. \(\check{H}(i, j) = \emptyset\).

3. If \((i, j) \not\rightarrow (i + 1, j + 1)\), then the following hold:
   1. \(\{ x \mid \nu(i, j) \prec x \} \setminus \hat{H}(i, j) = \emptyset\).
   2. \(\{ x \mid x \prec \nu(i + 1, j + 1) \} \setminus \hat{H}(i + 1, j + 1) = \emptyset\).
On a framework for Hillman–Grassl algorithms

Our framework

A H–G graph

**Definition**

We call \((\Gamma, \Delta, v: \Gamma \to P)\) a *H–G graph* for a finite poset \(P\) if

1. 
2. 
3. 
4. If \(((i_1, 1), \ldots, (i_j, j)) \in \bar{\Pi}(i, j)\), then \(v(i, j + 1) \not\in \{ v(i, t) \mid t \in \{ 1, \ldots, j \} \}\).
A H–G graph

Definition

We call \((\Gamma, \Delta, v: \Gamma \to P)\) a \(H\–G\ graph\) for a finite poset \(P\) if

1. 

2. 

3. 

4. 

5. If \(((i_j, j), \ldots, (i_e, e)) \in \hat{\Pi}(i, j),\) then 
   \(v(i, j - 1) \not\in \{ v(i, t) \mid t \in \{ j, \ldots, e \} \}.\)
Our framework

A H–G graph

Definition

We call \((\Gamma, \Delta, v : \Gamma \to P)\) a \textit{H–G graph} for a finite poset \(P\) if

1. There exists \(t\) such that \(v(i_m, m - w) = v(i_t, t)\); or
2. There exists \(t\) and \(t'\) such that \(((i_t, t), (i_{t'}, t')) \in \tilde{\Xi}(i_m; m - w, m)\).

If \(((i_1, 1), \ldots, (i_e, e)) \in \Pi\) and \(v(i_m, m - w) > v(i_m, m)\), then

1. There exists \(t\) such that \(v(i_m, m - w) = v(i_t, t)\); or
2. There exists \(t\) and \(t'\) such that \(((i_t, t), (i_{t'}, t')) \in \tilde{\Xi}(i_m; m - w, m)\).
On a framework for Hillman–Grassl algorithms

Our framework

A H–G graph

**Definition**

We call \((\Gamma, \Delta, v: \Gamma \rightarrow P)\) a *H–G graph* for a finite poset \(P\) if

1. there exists \(t\) such that \(v(i_m, m + w) = v(i_t, t)\); or
2. there exist \(t\) and \(t'\) such that \(((i_t, t), (i_{t'}, t')) \in \hat{\Xi}(i_m; m, m + w)\).

If \(((i_1, 1), \ldots, (i_e, e)) \in \Pi\) and \(v(i_m, m) \geq v(i_m, m + w)\), then
Let \((\Gamma, \Delta, v)\) be a H–G graph.
We call the set \(\{ v(k, k) \mid k \in \{1, 2, \ldots, r\} \}\) the first row of \(P\) w.r.t. \((\Gamma, \Delta, v)\).

A H–G graph is notion only for the first row of the poset \(P\). We also introduce notion for all rows of the poset \(P\).

**Definition**

We call \(\{(\Gamma_r, \Delta_r, v_r : \Gamma_r \to P_r) \mid r = 1, \ldots, k\}\) a H–G system for a poset \(P\) if the following conditions hold:

1. \(P = P_1 \supset P_2 \supset \cdots \supset P_k \supset P_{k+1} = \emptyset\).
2. For each \(r\), \((\Gamma_r, \Delta_r, v_r : \Gamma_r \to P_r)\) is a H–G graph for \(P_r\).
3. For each \(r\), \(P_r \setminus P_{r+1}\) is the first row of \(P_r\) w.r.t. \((\Gamma_r, \Delta_r, v_r : \Gamma_r \to P_r)\).
Input a reverse plane partition $T$ on $\lambda$ such that $T_{1,\lambda_1} > 0$.

Output $Z, c$.

Proc.  
1. Let $c = (1, 1), Z = \emptyset$.  
2. While $c \in \Gamma$, do the following:
   1. Append $v(c)$ to $Z$.
   2. If $c \rightarrow c'', T(v(c)) = T(v(c'))$ and $v(i_j + 1, j + 1) \notin Z$ then
      1. move $c$ via $\rightarrow$,  
      else
      1. move $c$ via $\rightarrow$.  

Outline

1 Introduction
   • Notation
   • The classical Hillman–Grassl algorithm

2 Our framework
   • Prototypical example
   • A H–G graph
   • Main results and Application
Let \((\Gamma, \Delta, v: \Gamma \rightarrow P)\) be a H–G graph.

**Theorem**

*Our algorithm for \((\Gamma, \Delta, v: \Gamma \rightarrow P)\) satisfies*

- *The output \(T'\) is a RPP on \(P\).*
- *This algorithm is invertible.*
- *If we apply this algorithm consecutively, then the resulting boxes is ordered.*
Let $R$ be the first row of $P$. Let

$$
\mathcal{R} = \left\{ (c_1, \ldots, c_k) \mid \begin{array}{l}
k = 0, 1, 2, \ldots \\
c_t \in R \\
c_{t-1} \leq c_t
\end{array} \right\}.
$$

**Theorem**

Our algorithm induces a bijection

$$
\varphi : \text{rpp}(P) \rightarrow \text{rpp}(P \setminus R) \times \mathcal{R}.
$$
Corollary

If \( \{ (\Gamma_r, \Delta_r, v_r : \Gamma_r \to P_r) \mid r = 1, \ldots, k \} \) is a H–G system for a poset \( P \), then we have a weight-preserving bijection between

- the set \( \text{rpp}(P) \) of \( P \)-partitions and
- the set of multisets of hooks.

The bijection induces a hook length formula.
Theorem

Let $P$ be a $d$-complete poset.

$P$ has a $H$–$G$ system (which is compatible with known hook structure).

$\Leftrightarrow$ $P$ is swivel-free.

Remark

‘sloant irreducible’ $d$-complete posets:

sweivel-free (1) Young diagrams, (2) shifted Young diagrams, (3) birds, (4) insets, (5) tailed insets, (6) banners, (7) nooks, (11) swivel shifteds;

not sweivel-free (8) swivels, (9) tailed swivels, (10) tagged swivels, (12) pumps, (13) tailed pumps, (14) near bats, (15) bat.
Remark

Let \((\Gamma, \Delta, \nu: \Gamma \to P)\) be a H–G graph for \(P\).

- The first row \(\{v(1,1), \ldots, v(l,l)\}\) of \(P\) is a poset-filter and a chain.
- The maximum element \(v(l,l)\) of the first row is a maximal element of \(P\).
- The hook \(H_{v(l,l)}\) at the element is a poset-filter of \(P\).

Hence, for \(d\)-complete posets including swivel, we can not construct a H–G system which is compatible with known hook structure.
Final remark.

Conjecture

If $P$ has a H–G system, then $P$ is a $d$-complete poset.