The Timed Plactic Monoid

Amritanshu Prasad

The Institute of Mathematical Sciences, Chennai
Homi Bhabha National Institute, Mumbai

21st May 2020
IMSc Algebraic Combinatorics Seminar

The Monoid of Words

The Monoid of Words

The Alphabet

$$A_n = \{1, \ldots, n\}$$

This is an *ordered* alphabet.

The Monoid of Words

The Alphabet

$$A_n = \{1, \ldots, n\}$$

This is an *ordered* alphabet.

The Monoid of Words

$$A_n^* = \{a_1 \cdots a_l \mid l \geq 0,\ a_i \in A_n\}.$$

The Monoid of Words

The Alphabet

$$A_n = \{1, \ldots, n\}$$

This is an *ordered* alphabet.

The Monoid of Words

$$A_n^* = \{a_1 \cdots a_l \mid l \geq 0,\ a_i \in A_n\}.$$

Product: concatentation of words.

The Monoid of Words

The Alphabet

$$A_n = \{1, \ldots, n\}$$

This is an *ordered* alphabet.

The Monoid of Words

$$A_n^* = \{a_1 \cdots a_l \mid l \geq 0,\ a_i \in A_n\}.$$

Product: concatentation of words.
Unit: the empty word $\emptyset$.

The Monoid of Words

The Alphabet

$$A_n = \{1, \ldots, n\}$$

This is an *ordered* alphabet.

The Monoid of Words

$$A_n^* = \{a_1 \cdots a_l \mid l \geq 0,\ a_i \in A_n\}.$$

Product: concatentation of words.
Unit: the empty word $\emptyset$.

Subwords

$$w = a_1 \cdots a_l$$
$$w' = a_{i_1} \cdots a_{i_k},\ 1 \leq i_1 < \cdots < i_k \leq l.$$

Then w' is said to be a *subword of* w.

The Longest Increasing Subword Problem

The Longest Increasing Subword Problem

Problem

Given a word $w \in A_n^*$, determine the length of the longest weakly increasing subword of w.

The Longest Increasing Subword Problem

Problem

Given a word $w \in A_n^*$, determine the length of the longest weakly increasing subword of w.

Schensted's one-pass algorithm

Idea: read the word from left to right, keeping track of the least last element the longest increasing subword of length r for each r.

Schensted's algorithm by example

$$w = 3523245111$$

$$p =$$

$$w' =$$

Schensted's algorithm by example

$$w = 3523245111$$

$$p =$$

$$w' =$$

Schensted's algorithm by example

$$w = 523245111$$

$$p = 3$$

$$w' =$$

Schensted's algorithm by example

$$w = 523245111$$

$$p = 3$$

$$w' =$$

Schensted's algorithm by example

$$w = 23245111$$

$$p = 35$$

$$w' =$$

Schensted's algorithm by example

$$w = 23245111$$

$$p = 35$$

$$w' =$$

Schensted's algorithm by example

$$w = 3245111$$
$$p = 25$$
$$w' = 3$$

Schensted's algorithm by example

$$w = 3245111$$
$$p = 25$$
$$w' = 3$$

Schensted's algorithm by example

$$w = 245111$$
$$p = 23$$
$$w' = 35$$

Schensted's algorithm by example

$$w = 245111$$
$$p = 23$$
$$w' = 35$$

Schensted's algorithm by example

$$w = 45111$$
$$p = 22$$
$$w' = 353$$

Schensted's algorithm by example

$$w = 45111$$
$$p = 22$$
$$w' = 353$$

Schensted's algorithm by example

$$w = 5111$$
$$p = 224$$
$$w' = 353$$

Schensted's algorithm by example

$$w = 5111$$
$$p = 224$$
$$w' = 353$$

Schensted's algorithm by example

$$w = 111$$
$$p = 2245$$
$$w' = 353$$

Schensted's algorithm by example

$$w = 111$$
$$p = 2245$$
$$w' = 353$$

Schensted's algorithm by example

$$w = 11$$
$$p = 1245$$
$$w' = 3532$$

Schensted's algorithm by example

$$w = 11$$
$$p = 1245$$
$$w' = 3532$$

Schensted's algorithm by example

$$w = 1$$
$$p = 1145$$
$$w' = 35322$$

Schensted's algorithm by example

$$w = 1$$
$$p = 1145$$
$$w' = 35322$$

Schensted's algorithm by example

$$w =$$
$$p = 1115$$
$$w' = 353224$$

Schensted's algorithm by example

$$w =$$
$$p = 1115$$
$$w' = 353224$$

The length of the longest increasing subword of $w = 3523245111$ is 4.

Schensted's algorithm by example

$$w =$$
$$p = 1115$$
$$w' = 353224$$

The length of the longest increasing subword of $w = 3523245111$ is 4.
The *shadow word* of $w = 3523245111$ is $w' = 353224$.

Schensted's algorithm by example

$$w =$$
$$p = 1115$$
$$w' = 353224$$

The length of the longest increasing subword of $w = 3523245111$ is 4.
The *shadow word* of $w = 3523245111$ is $w' = 353224$.
But let us continue this process, now with w' in place of w.

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= 353224 \\
p' &= \\
w'' &=
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= 353224 \\
p' &= \\
w'' &=
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= 53224 \\
p' &= 3 \\
w'' &=
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= 53224 \\
p' &= 3 \\
w'' &=
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= 3224 \\
p' &= 35 \\
w'' &=
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= 224 \\
p' &= 33 \\
w'' &= 5
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= 224 \\
p' &= 33 \\
w'' &= 5
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= 24 \\
p' &= 23 \\
w'' &= 53
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= 24 \\
p' &= 23 \\
w'' &= 53
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= 4 \\
p' &= 22 \\
w'' &= 533
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= 4 \\
p' &= 22 \\
w'' &= 533
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= 533
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= 533
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= 533 \\
p'' &= \\
w''' &=
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= 533 \\
p'' &= \\
w''' &=
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= 33 \\
p'' &= 5 \\
w''' &=
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= 33 \\
p'' &= 5 \\
w''' &=
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= 3 \\
p'' &= 3 \\
w''' &= 5
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= 3 \\
p'' &= 3 \\
w''' &= 5
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= \\
p'' &= 33 \\
w''' &= 5
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= \\
p'' &= 33 \\
w''' &= 5 \\
p''' &=
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= \\
p'' &= 33 \\
w''' &= 5 \\
p''' &=
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= \\
p'' &= 33 \\
w''' &= \\
p''' &= 5
\end{aligned}
$$

Schensted's algorithm by example

$$
\begin{aligned}
w &= \\
p &= 1115 \\
w' &= \\
p' &= 224 \\
w'' &= \\
p'' &= 33 \\
w''' &= \\
p''' &= 5
\end{aligned}
$$

The insertion tableaux of w

$$P(3523245111) = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 5 \\ \hline 2 & 2 & 4 \\ \cline{1-3} 3 & 3 \\ \cline{1-2} 5 \\ \cline{1-1} \end{array}.$$

Insertion Tableau

The insertion tableaux of w

$$P(3523245111) = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 5 \\ \hline 2 & 2 & 4 \\ \cline{1-3} 3 & 3 \\ \cline{1-2} 5 \\ \cline{1-1} \end{array} \text{ of shape } (4,3,2,1).$$

Insertion Tableau

The insertion tableaux of w

$$P(3523245111) = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 5 \\ \hline 2 & 2 & 4 \\ \cline{1-3} 3 & 3 \\ \cline{1-2} 5 \\ \cline{1-1} \end{array} \text{ of shape } (4,3,2,1).$$

Question

The length of the first row is the length of the longest increasing subword. What is the interpretation of the lengths of the remaining rows?

Greene's Theorem

$$w = a_1 \cdots a_l.$$

Greene's Theorem

$$w = a_1 \cdots a_l.$$

Consider two subwords:

$$u = a_{i_1} \cdots a_{i_k} \text{ and } v = a_{j_1} \cdots a_{j_r}.$$

Greene's Theorem

$$w = a_1 \cdots a_l.$$

Consider two subwords:

$$u = a_{i_1} \cdots a_{i_k} \text{ and } v = a_{j_1} \cdots a_{j_r}.$$

Disjoint Subwords

We say that u and v are disjoint if

$$\{i_1, \ldots, i_k\} \cap \{j_1, \ldots, j_r\} = \emptyset.$$

Greene's Theorem

Theorem

Suppose w is a word and $P(w)$ has shape $(\lambda_1, \ldots, \lambda_l)$. Then, for each k, $\lambda_1 + \cdots + \lambda_k$ is the maximum sum of lengths of k pairwise disjoint weakly increasing subwords.

Greene's Theorem

Theorem

Suppose w is a word and $P(w)$ has shape $(\lambda_1, \ldots, \lambda_l)$. Then, for each k, $\lambda_1 + \cdots + \lambda_k$ is the maximum sum of lengths of k pairwise disjoint weakly increasing subwords.

Greene invariants

$$a_k = \begin{array}{l}\text{max. sum of lengths of } k \text{ pairwise disjoint} \\ \text{weakly increasing subwords}\end{array} .$$

Greene's Theorem

Theorem

Suppose w is a word and $P(w)$ has shape $(\lambda_1, \ldots, \lambda_l)$. Then, for each k, $\lambda_1 + \cdots + \lambda_k$ is the maximum sum of lengths of k pairwise disjoint weakly increasing subwords.

Greene invariants

$$a_k = \begin{array}{l}\text{max. sum of lengths of } k \text{ pairwise disjoint} \\ \text{weakly increasing subwords}\end{array} .$$

Greene's theorem

$$a_k = \lambda_1 + \cdots + \lambda_k.$$

Proof of Greene's theorem

Knuth Moves

The Knuth moves (rewriting rules) on A^* are:

$$uyxzv \equiv uyzxv \text{ if } x < y \leq z,$$
$$uxzyv \equiv uzxyv \text{ if } x \leq y < z.$$

Greene's Proof

1. Greene invariants remain unchanged under Knuth moves.
2. $w \equiv$ reading word$(P(w))$
3. If w is the reading word of a tableau, then Greene's theorem is easy.

Proof of Greene's theorem

Knuth Moves

The Knuth moves (rewriting rules) on A^* are:

$$uyxzv \equiv uyzxv \text{ if } x < y \leq z,$$
$$uxzyv \equiv uzxyv \text{ if } x \leq y < z.$$

Greene's Proof

1. Greene invariants remain unchanged under Knuth moves.
2. $w \equiv$ reading word$(P(w))$
3. If w is the reading word of a tableau, then Greene's theorem is easy.

In this proof, the most non-trivial part is the identification of Knuth moves — simple enough for 1. to be provable, but strong enough for 2. to hold.

What is the reading word?

What is the reading word?

$$P(3523245111) = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 5 \\ \hline 2 & 2 & 4 \\ \cline{1-3} 3 & 3 \\ \cline{1-2} 5 \\ \cline{1-1} \end{array}$$

has reading word:

$$5\ 33\ 224\ 1115$$

What is the reading word?

$$P(3523245111) = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 5 \\ \hline 2 & 2 & 4 \\ \cline{1-3} 3 & 3 \\ \cline{1-2} 5 \\ \cline{1-1} \end{array}$$

has reading word:

$$5\ 33\ 224\ 1115$$

Step 2 in Greene's proof says that:

$$3523245111 \equiv 5332241115.$$

Theoretical Computer Science 126 (1994) 183–235
Elsevier

Fundamental Study

A theory of timed automata*

Rajeev Alur** and David L. Dill***

Computer Science Department, Stanford University, Stanford, CA 94305-2095, USA

Communicated by M.S. Paterson
Received November 1991
Revised November 1992

Abstract

Alur, R. and D.L. Dill, A theory of timed automata, Theoretical Computer Science 126 (1994) 183–235.

We propose *timed (finite) automata* to model the behavior of real-time systems over time. Our definition provides a simple, and yet powerful, way to annotate state-transition graphs with timing constraints using finitely many real-valued *clocks*. A timed automaton accepts *timed words* – infinite sequences in which a real-valued time of occurrence is associated with each symbol. We study timed automata from the perspective of formal language theory: we consider closure properties, decision problems, and subclasses. We consider both nondeterministic and deterministic transition structures, and both Büchi and Muller acceptance conditions. We show that nondeterministic timed automata are closed under union and intersection, but not under complementation, whereas deterministic timed Muller automata are closed under all Boolean operations. The main construction of the paper is an (PSPACE) algorithm for checking the emptiness of the language of a (nondeterministic) timed automaton. We also prove that the universality problem and the language inclusion problem are solvable only for the deterministic automata: both problems are undecidable (Π_1^1-hard) in the nondeterministic case and PSPACE-complete in the deterministic case. Finally, we discuss the application of this theory to automatic verification of real-time requirements of finite-state systems.

Timed Words

In *formal verification* of systems, a word represents a *sequence of events*.

Timed Words

In *formal verification* of systems, a word represents a *sequence of events*.
The formal verification of *timed systems* involves a *sequence of events with timestamps*.

Timed Words

In *formal verification* of systems, a word represents a *sequence of events*.
The formal verification of *timed systems* involves a *sequence of events with timestamps*.

$$\text{word: } a_1 \cdots a_l$$

$$\text{timed word: } a_1^{t_1} \cdots a_l^{t_l}$$

Timed Words

In *formal verification* of systems, a word represents a *sequence of events*.
The formal verification of *timed systems* involves a *sequence of events with timestamps*.

$$\text{word: } a_1 \cdots a_l$$
$$\text{timed word: } a_1^{t_1} \cdots a_l^{t_l}$$

Here t_i is a non-negative real number, which represents the *amount of time for which the letter a_i persisted*.

Timed Words

In *formal verification* of systems, a word represents a *sequence of events*.
The formal verification of *timed systems* involves a *sequence of events with timestamps*.

$$\text{word: } a_1 \cdots a_l$$
$$\text{timed word: } a_1^{t_1} \cdots a_l^{t_l}$$

Here t_i is a non-negative real number, which represents the *amount of time for which the letter a_i persisted*.

$$A_n^\dagger = \text{Monoid of timed words}$$

Timed Words

In *formal verification* of systems, a word represents a *sequence of events*.
The formal verification of *timed systems* involves a *sequence of events with timestamps*.

$$\text{word: } a_1 \cdots a_l$$
$$\text{timed word: } a_1^{t_1} \cdots a_l^{t_l}$$

Here t_i is a non-negative real number, which represents the *amount of time for which the letter a_i persisted*.

$$A_n^\dagger = \text{Monoid of timed words}$$

Contains the monoid A_n^* as the submoid of words $a_1^{t_1} \cdots a_l^{t_l}$, where $t_1, \cdots, t_l$ are positive integers.

Subwords of Timed Words

A timed word:

$$w = c_1^{t_1} \cdots c_k^{t_k}$$

of length $l(w) = t_1 + \cdots + t_k$ can be thought of as a right-continuous piecewise-constant function

$$w : [0, l(w)) \to A_n,$$

with $w(t) = a_i$ if $t_1 + \cdots + t_{i-1} \leq t < t_1 + \cdots + t_i$.

Subwords of Timed Words

A timed word:

$$w = c_1^{t_1} \cdots c_k^{t_k}$$

of length $l(w) = t_1 + \cdots + t_k$ can be thought of as a right-continuous piecewise-constant function

$$w : [0, l(w)) \to A_n,$$

with $w(t) = a_i$ if $t_1 + \cdots + t_{i-1} \leq t < t_1 + \cdots + t_i$.
Given a subset $S = [a_1, b_1) \cup \cdots [a_r, b_r) \subset [0, l(w))$, can construct a timed word w_S:

$$w_S(t) = w(u),$$

where u is such that $\mathrm{meas}([0, u) \cap S) = t$.

Subwords of Timed Words

A timed word:

$$w = c_1^{t_1} \cdots c_k^{t_k}$$

of length $l(w) = t_1 + \cdots + t_k$ can be thought of as a right-continuous piecewise-constant function

$$w : [0, l(w)) \to A_n,$$

with $w(t) = a_i$ if $t_1 + \cdots + t_{i-1} \leq t < t_1 + \cdots + t_i$.
Given a subset $S = [a_1, b_1) \cup \cdots [a_r, b_r) \subset [0, l(w))$, can construct a timed word w_S:

$$w_S(t) = w(u),$$

where u is such that $\mathrm{meas}([0, u) \cap S) = t$.
Subwords w_S and w_T are said to be *disjoint* if $S \cap T = \emptyset$.

Timed Row Insertion

A *timed row* is a weakly increasing timed word.

Timed Row Insertion

A *timed row* is a weakly increasing timed word.
Given a timed row u, the insertion $ROWINS(u, c^{t_c})$ of c^{t_c} into u is defined as follows:

Timed Row Insertion

A *timed row* is a weakly increasing timed word.
Given a timed row u, the insertion $ROWINS(u, c^{t_c})$ of c^{t_c} into u is defined as follows: if $u(t) \leq c$ for all $0 \leq t < l(u)$, then

$$ROWINS(u, c^{t_c}) = (\emptyset, uc^{t_c}).$$

Timed Row Insertion

A *timed row* is a weakly increasing timed word.
Given a timed row u, the insertion $ROWINS(u, c^{t_c})$ of c^{t_c} into u is defined as follows: if $u(t) \leq c$ for all $0 \leq t < l(u)$, then

$$ROWINS(u, c^{t_c}) = (\emptyset, uc^{t_c}).$$

Otherwise, there exists $0 \leq t < l(u)$ such that $u(t) > c$. Let

$$t_0 = \min\{0 \leq t < l(u) \mid u(t) > c\}.$$

Timed Row Insertion

A *timed row* is a weakly increasing timed word.
Given a timed row u, the insertion $ROWINS(u, c^{t_c})$ of c^{t_c} into u is defined as follows: if $u(t) \leq c$ for all $0 \leq t < l(u)$, then

$$ROWINS(u, c^{t_c}) = (\emptyset, uc^{t_c}).$$

Otherwise, there exists $0 \leq t < l(u)$ such that $u(t) > c$. Let

$$t_0 = \min\{0 \leq t < l(u) \mid u(t) > c\}.$$

Define

$$ROWINS(u, c^{t_c}) = \begin{cases} (u_{[t_0, t_0+t_c)}, u_{[0,t_0)} c^{t_c} u_{[t_0+t_c, l(w))}) & \text{if } l(u) - t_0 > t_c, \\ (u_{[t_0, l(u))}, u_{[0,t_0)} c^{t_c}) & \text{if } l(u) - t_0 \leq t_c. \end{cases}$$

Timed Row Insertion

A *timed row* is a weakly increasing timed word.
Given a timed row u, the insertion $ROWINS(u, c^{t_c})$ of c^{t_c} into u is defined as follows: if $u(t) \leq c$ for all $0 \leq t < l(u)$, then

$$ROWINS(u, c^{t_c}) = (\emptyset, uc^{t_c}).$$

Otherwise, there exists $0 \leq t < l(u)$ such that $u(t) > c$. Let

$$t_0 = \min\{0 \leq t < l(u) \mid u(t) > c\}.$$

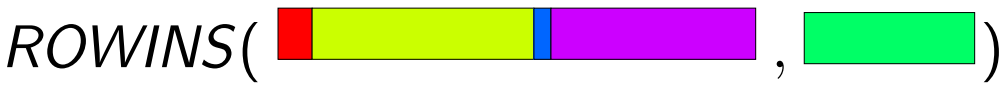
Define

$$ROWINS(u, c^{t_c}) = \begin{cases} (u_{[t_0, t_0+t_c)}, u_{[0,t_0)} c^{t_c} u_{[t_0+t_c, l(w))}) & \text{if } l(u) - t_0 > t_c, \\ (u_{[t_0, l(u))}, u_{[0,t_0)} c^{t_c}) & \text{if } l(u) - t_0 \leq t_c. \end{cases}$$

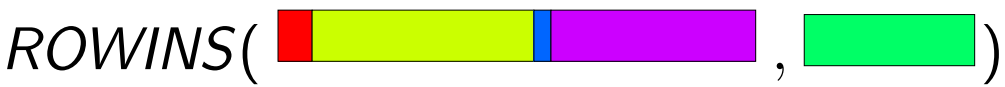
This is the natural extension of Schensted's algorithm for finding the length of the longest increasing subword.

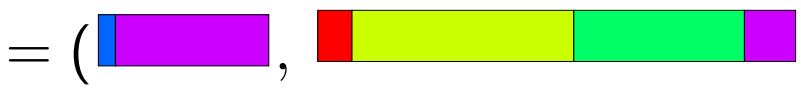
Visualization of row insertion

Visualization of row insertion



Visualization of row insertion





Timed Tableaux

A *timed tableau* is a timed word of the form $u_l u_{l-1} \cdots u_1$ such that

Timed Tableaux

A *timed tableau* is a timed word of the form $u_l u_{l-1} \cdots u_1$ such that

1. Each u_i is a timed row.

Timed Tableaux

A *timed tableau* is a timed word of the form $u_l u_{l-1} \cdots u_1$ such that

1. Each u_i is a timed row.
2. $l(u_1) \geq l(u_2) \geq \cdots l(u_l)$.

Timed Tableaux

A *timed tableau* is a timed word of the form $u_l u_{l-1} \cdots u_1$ such that

1. Each u_i is a timed row.
2. $l(u_1) \geq l(u_2) \geq \cdots l(u_l)$.
3. $u_i(t) > u_{i-1}(t)$ for $0 \leq t \leq l(u_i)$, for $i = 2, \ldots, l$.

Timed Tableaux

A *timed tableau* is a timed word of the form $u_l u_{l-1} \cdots u_1$ such that

1. Each u_i is a timed row.
2. $l(u_1) \geq l(u_2) \geq \cdots l(u_l)$.
3. $u_i(t) > u_{i-1}(t)$ for $0 \leq t \leq l(u_i)$, for $i = 2, \ldots, l$.

Insertion tableau of a timed word

Just as the insertion tableau of a word was defined using row insertion, the timed insertion tableau of a timed word can be defined using timed row insertion.

Insertion tableau of a timed word

Just as the insertion tableau of a word was defined using row insertion, the timed insertion tableau of a timed word can be defined using timed row insertion.

Using a colormap to represent the alphabet:

Insertion tableau of a timed word

Just as the insertion tableau of a word was defined using row insertion, the timed insertion tableau of a timed word can be defined using timed row insertion.

Using a colormap to represent the alphabet:

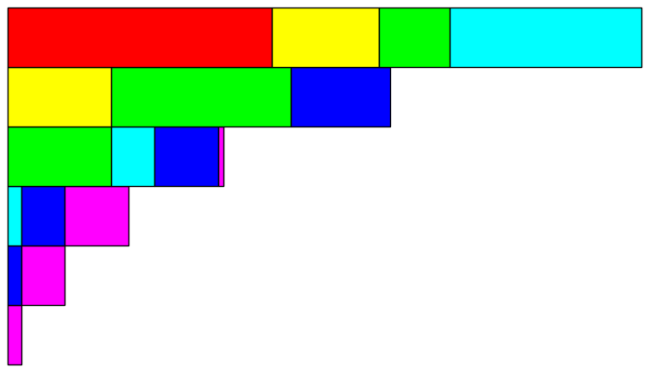
a timed word can be represented as a ribbon:

Insertion tableau of a timed word

Just as the insertion tableau of a word was defined using row insertion, the timed insertion tableau of a timed word can be defined using timed row insertion.
Using a colormap to represent the alphabet:

a timed word can be represented as a ribbon:

The insertion tableau of the above timed word is:



Greene's theorem for timed words

Greene'e theorem holds for timed words

Greene's theorem for timed words

Greene'e theorem holds for timed words

The only difficulty is the identification of a replacement for Knuth relations.

Knuth relations

Knuth's original relations

$$xzy \equiv zxy \text{ if } x \leq y < z,$$
$$yxz \equiv yzx \text{ if } x < y \leq z.$$

Knuth relations

Knuth's original relations

$$xzy \equiv zxy \text{ if } x \leq y < z,$$
$$yxz \equiv yzx \text{ if } x < y \leq z.$$

Knuth's relations for timed words

$xzy \equiv zxy$ when xyz is a timed row, $l(z) = l(y)$, and $\lim_{t \to l(y)^-} \mathbf{y}(t) < \mathbf{z}(0)$,

$yxz \equiv yzx$ when xyz is a timed row, $l(x) = l(y)$, and $\lim_{t \to l(x)^-} \mathbf{x}(t) < \mathbf{y}(0)$.

Main Step in Proof of Greene's Theorem

Lemma

If two timed words are timed Knuth equivalent, then they have the same Greene invariants.

Proof in the classical case

Consider the relation:

$$xzy \equiv zxy \text{ if } x \leq y < z.$$

wxzyu	wzxyu

Proof in the classical case

Consider the relation:

$$xzy \equiv zxy \text{ if } x \leq y < z.$$

wxzyu	wzxyu
w'zu'	w'zu'

Proof in the classical case

Consider the relation:

$$xzy \equiv zxy \text{ if } x \leq y < z.$$

wxzyu	wzxyu
w'zu'	w'zu'
w'xyu'	w'xyu'

Proof in the classical case

Consider the relation:

$$xzy \equiv zxy \text{ if } x \leq y < z.$$

wxzyu	wzxyu
w'zu'	w'zu'
w'xyu'	w'xyu'
w'xzu'	

Proof in the classical case

Consider the relation:

$$xzy \equiv zxy \text{ if } x \leq y < z.$$

wxzyu	wzxyu
w'zu'	w'zu'
w'xyu'	w'xyu'
w'xzu'	w'xyu'

Proof in the classical case

Consider the relation:

$$xzy \equiv zxy \text{ if } x \leq y < z.$$

wxzyu	wzxyu
w'zu'	w'zu'
w'xyu'	w'xyu'
w'xzu'	w'xyu'
w'xzu' w''yu''	

Proof in the classical case

Consider the relation:

$$xzy \equiv zxy \text{ if } x \leq y < z.$$

wxzyu	wzxyu
w'zu'	w'zu'
w'xyu'	w'xyu'
w'xzu'	w'xyu'
w'xzu' w''yu''	w'xyu'' w''zu'

Proof in the timed case

Consider the relation:

$xzy \equiv zxy$ when xyz is a timed row, $l(z) = l(y)$, and $\lim_{t \to l(y)^-} \mathbf{y}(t) < \mathbf{z}(0)$

Consider k timed row subwords of $wxzyu$ realizing $a_k(wxzyu)$:

$wxzyu$	$wzxyu$
$w_1 x_1 z_1 u_1$	
$w_2 x_2 z_2 u_2$	
$\vdots$	$\vdots$
$w_r x_r z_r u_r$	
$w_{r+1} x_{r+1} y_{r+1} u_{r+1}$	
$\vdots$	$\vdots$
$w_k x_k y_k u_k$	

Proof in the timed case

Consider the relation:

$xzy \equiv zxy$ when xyz is a timed row, $l(z) = l(y)$, and $\lim_{t \to l(y)^-} \mathbf{y}(t) < \mathbf{z}(0)$

Consider k timed subwords of $wxzyu$ realizing $a_k(wxzyu)$:
Case 1: $x_i = \emptyset$ for $i = 1, \ldots, r$

$wxzyu$	$wzxyu$
$w_1 x_1 z_1 u_1$	$w_1 z_1 u_1$
$w_2 x_2 z_2 u_2$	$w_2 z_2 u_2$
$\vdots$	$\vdots$
$w_r x_r z_r u_r$	$w_r z_r u_r$
$w_{r+1} x_{r+1} y_{r+1} u_{r+1}$	$w_{r+1} x_{r+1} y_{r+1} u_{r+1}$
$\vdots$	$\vdots$
$w_k x_k y_k u_k$	$w_k x_k y_k u_k$

Proof in the timed case

Consider the relation:

$xzy \equiv zxy$ when xyz is a timed row, $l(z) = l(y)$, and $\lim_{t \to l(y)^-} \mathbf{y}(t) < \mathbf{z}(0)$

Consider k timed subwords of $wxzyu$ realizing $a_k(wxzyu)$:
Case 2: $y_i = \emptyset$ for $i = r+1, \ldots, k$

$wxzyu$	$wzxyu$
$w_1 x_1 z_1 u_1$	$w_1 x_1 y u_1$
$w_2 x_2 z_2 u_2$	$w_2 x_2 u_2$
$\vdots$	$\vdots$
$w_r x_r z_r u_r$	$w_r x_r u_r$
$w_{r+1} x_{r+1} y_{r+1} u_{r+1}$	$w_{r+1} x_{r+1} u_{r+1}$
$\vdots$	$\vdots$
$w_k x_k y_k u_k$	$w_k x_k u_k$

Proof in the timed case

Consider the relation:

$xzy \equiv zxy$ when xyz is a timed row, $l(z) = l(y)$, and $\lim_{t \to l(y)^-} \mathbf{y}(t) < \mathbf{z}(0)$

Consider k timed subwords of $wxzyu$ realizing $a_k(wxzyu)$:
Case 3: $x_1 \neq \emptyset$ and $y_k \neq \emptyset$, $(\min x_1 \leq \min x_i,\ \max y_k \geq \max y_i)$

$wxzyu$	$wzxyu$
$w_1 x_1 z_1 u_1$	$w_1 x_0 y_0 u_k$
$w_2 x_2 z_2 u_2$	$w_2 z_2 u_2$
$\vdots$	$\vdots$
$w_r x_r z_r u_r$	$w_r z_r u_r$
$w_{r+1} x_{r+1} y_{r+1} u_{r+1}$	$w_{r+1} u_{r+1}$
$\vdots$	$\vdots$
$w_k x_k y_k u_k$	$w_k u_1$

Timed Pieri Rules

Timed Pieri Rules

A *timed column* in $A_n^\dagger$ is a word of the form:

$$n^{a_n} \cdots 2^{a_2} 1^{a_1},$$

where $a_1, \ldots, a_n \in [0, 1]$.

Timed Pieri Rules

A *timed column* in $A_n^\dagger$ is a word of the form:

$$n^{a_n} \cdots 2^{a_2} 1^{a_1},$$

where $a_1, \ldots, a_n \in [0, 1]$. Let $R^\dagger(A_n)$ and $C^\dagger(A_n)$ denote the sets of timed rows and timed columns.

Timed Pieri Rules

A *timed column* in $A_n^\dagger$ is a word of the form:

$$n^{a_n} \cdots 2^{a_2} 1^{a_1},$$

where $a_1, \ldots, a_n \in [0, 1]$. Let $R^\dagger(A_n)$ and $C^\dagger(A_n)$ denote the sets of timed rows and timed columns.

Pieri Rules

Insertion gives rise to bijections:

$$\mathrm{Tab}_\lambda^\dagger(A_n) \times R^\dagger(A_n) \tilde{\to} \coprod_{\mu-\lambda \text{ horiz. strip}} \mathrm{Tab}_\mu^\dagger(A_n),$$

$$\mathrm{Tab}_\lambda^\dagger(A_n) \times C^\dagger(A_n) \tilde{\to} \coprod_{\mu-\lambda \text{ vert. strip}} \mathrm{Tab}_\mu^\dagger(A_n).$$

Application: PL interpolation of RSK

RSK correspondence

Matrices with non-negative real entries are in bijective correspondence with pairs of timed tableau of the same shape.

$$A \leftrightarrow (P, Q)$$

Example

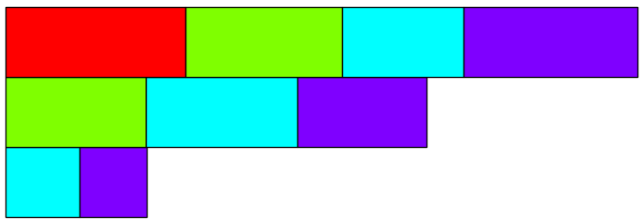
$$A = \begin{pmatrix} 0.16 & 0.29 & 0.68 & 0.44 \\ 0.29 & 0.70 & 0.38 & 0.45 \\ 0.32 & 0.29 & 0.43 & 0.70 \end{pmatrix}.$$

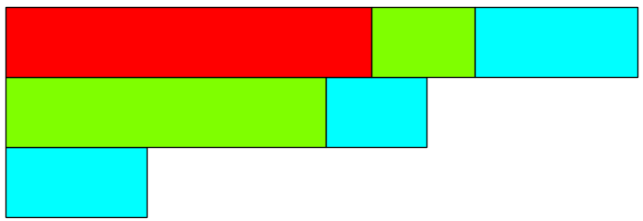
Then $(P, Q) = RSK(A)$ are given by:

$$P = 3^{0.32}4^{0.29}2^{0.60}3^{0.65}4^{0.55}1^{0.77}2^{0.67}3^{0.52}4^{0.75},$$
$$Q = 3^{0.61}2^{1.38}3^{0.43}1^{1.57}2^{0.45}3^{0.70}.$$

Using the colormap:

$RSK(A)$ is:

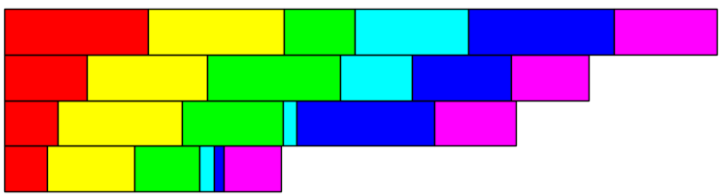


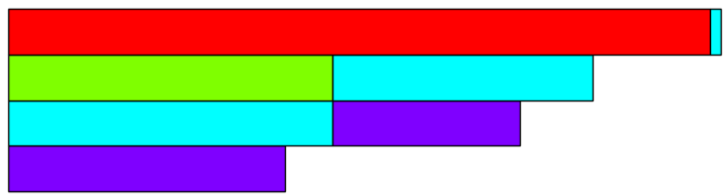


The Dual RSK Correspondence

$$A = \begin{pmatrix} 0.36 & 0.99 & 0.88 & 0.84 \\ 0.55 & 0.63 & 0.43 & 0.09 \\ 0.95 & 0.58 & 0.67 & 0.09 \\ 0.28 & 0.90 & 0.47 & 0.37 \end{pmatrix}$$

Then the dual RSK correspondence is given by:

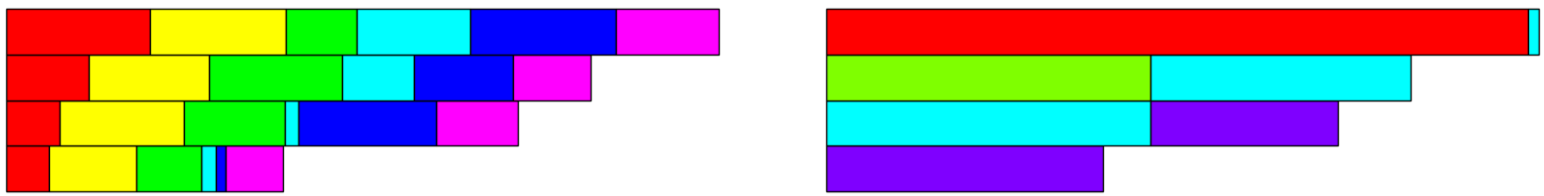




The Dual RSK Correspondence

$$A = \begin{pmatrix} 0.36 & 0.99 & 0.88 & 0.84 \\ 0.55 & 0.63 & 0.43 & 0.09 \\ 0.95 & 0.58 & 0.67 & 0.09 \\ 0.28 & 0.90 & 0.47 & 0.37 \end{pmatrix}$$

Then the dual RSK correspondence is given by:



Gale-Ryser Polytope

The dual RSK correspondence is a PL bijection from the Gale-Ryser polytope onto the set of pairs (P, Q) where P is a dual timed tableau and Q is a timed tableau, P and Q have the same shape.

Greene's Duality Theorem: An Unsolved Problem

Given $w \in A_n^*$, define

$$b_k = \begin{array}{l}\text{max. sum of lengths of } k \text{ pairwise disjoint} \\ \text{column subwords}\end{array} .$$

Greene's Duality Theorem: An Unsolved Problem

Given $w \in A_n^*$, define

$$b_k = \begin{array}{l}\text{max. sum of lengths of } k \text{ pairwise disjoint} \\ \text{column subwords}\end{array} .$$

Greene's Duality Theorem

Let $\mu_k = b_k - b_{k-1}$, and $\lambda_k = a_k - a_{k-1}$. Then μ and λ are mutually conjugate partitions.

Greene's Duality Theorem: An Unsolved Problem

Given $w \in A_n^*$, define

$$b_k = \begin{array}{l}\text{max. sum of lengths of } k \text{ pairwise disjoint} \\ \text{column subwords}\end{array} .$$

Greene's Duality Theorem

Let $\mu_k = b_k - b_{k-1}$, and $\lambda_k = a_k - a_{k-1}$. Then μ and λ are mutually conjugate partitions.

Problem

Formulate and prove Greene's Duality Theorem for timed words.