Deciding conjugacy for certain class of one-sided finite-type-Dyck shifts

Pavel Heller, joint work with Marie-Pierre Béal

9 April 2015
Journées SDA 2 du GDR IM
Université Paris-Est Marne-la-Vallée
Deciding conjugacy for certain class of one-sided finite-type-Dyck shifts

Deciding shift conjugacy: generally open problem

- decidable for 1-sided SFT’s [Williams, 73]
- unknown for 1-sided sofic shifts
- unknown for 2-sided SFT’s

Our focus: (1-sided) finite-type-Dyck shifts. [Béal, Blockelet, Dima, 2013]

We present an extension of Williams’ result.
Deciding conjugacy for certain class of one-sided finite-type-Dyck shifts

Deciding shift conjugacy: generally open problem

- decidable for 1-sided SFT’s [Williams, 73]
- unknown for 1-sided sofic shifts
- unknown for 2-sided SFT’s

Our focus: (1-sided) finite-type-Dyck shifts. [Béal, Blockelet, Dima, 2013]

We present an extension of Williams’ result.
Deciding conjugacy for certain class of one-sided finite-type-Dyck shifts

Deciding shift conjugacy: generally open problem

- decidable for 1-sided SFT’s [Williams, 73]
- unknown for 1-sided sofic shifts
- unknown for 2-sided SFT’s

Our focus: (1-sided) finite-type-Dyck shifts. [Béal, Blockelet, Dima, 2013]

We present an extension of Williams’ result.
Deciding conjugacy for certain class of one-sided finite-type-Dyck shifts

Deciding shift conjugacy: generally open problem

- decidable for 1-sided SFT’s [Williams, 73]
- unknown for 1-sided sofic shifts
- unknown for 2-sided SFT’s

Our focus: (1-sided) finite-type-Dyck shifts. [Béal, Blockelet, Dima, 2013]

We present an extension of Williams’ result.
Outline

Standard notions
  Shifts (of finite type)
  Conjugacy

1-sided finite-type-Dyck shifts
  Dyck shifts
  Finite-type-Dyck shifts

Conjugacy for 1-sided FTD shifts
  Deciding conjugacy
Outline

Standard notions
  Shifts (of finite type)
  Conjugacy

1-sided finite-type-Dyck shifts
  Dyck shifts
  Finite-type-Dyck shifts

Conjugacy for 1-sided FTD shifts
  Deciding conjugacy
Shifts

- A finite alphabet
- $A^*$ set of finite words over $A$
- $A^\mathbb{Z}$ set of bi-infinite sequences over $A$
- $A^{-\mathbb{N}}$ set left-infinite sequences over $A$

**Shift**

$X \subset A^\mathbb{Z}$ (resp. $A^{-\mathbb{N}}$) is called a (one-sided) shift if it is the set of sequences avoiding some set of forbidden words $F \subset A^*$. We denote $X = X_F$.

$X$ is equipped with a shift map and topology.
**Shifts**

- A finite alphabet
- $A^* \text{ set of finite words over } A$
- $A^\mathbb{Z} \text{ set of bi-infinite sequences over } A$
- $A^{-\mathbb{N}} \text{ set left-infinite sequences over } A$

**Shift**

$X \subset A^\mathbb{Z}$ (resp. $A^{-\mathbb{N}}$) is called a *(one-sided)* shift if it is the set of sequences avoiding some set of forbidden words $F \subset A^*$. We denote $X = X_F$.

$X$ is equipped with a shift map and topology.
Shifts

- A finite alphabet
- $A^*$ set of finite words over $A$
- $A^\mathbb{Z}$ set of bi-infinite sequences over $A$
- $A^{-\mathbb{N}}$ set left-infinite sequences over $A$

**Shift**

$X \subset A^\mathbb{Z}$ (resp. $A^{-\mathbb{N}}$) is called a *one-sided* shift if it is the set of sequences avoiding some set of forbidden words $F \subset A^*$. We denote $X = X_F$.

$X$ is equipped with a shift map and topology.
Shifts

- A finite alphabet
- $A^*$ set of finite words over $A$
- $A^\mathbb{Z}$ set of bi-infinite sequences over $A$
- $A^{-\mathbb{N}}$ set left-infinite sequences over $A$

**Shift**

$X \subset A^\mathbb{Z}$ (resp. $A^{-\mathbb{N}}$) is called a (one-sided) shift if it is the set of sequences avoiding some set of forbidden words $F \subset A^*$. We denote $X = X_F$.

$X$ is equipped with a shift map and topology.
**Sofic shift**

$X = X_F$ where $F$ is regular is called a *sofic shift*.

**Shift of finite type**

$X = X_F$ where $F$ is finite is called a *shift of finite type*.

A sofic shift can be represented by a finite automaton; a SFT by a local finite automaton.
**Sofic shift**

$X = X_F$ where $F$ is regular is called a *sofic shift*.

**Shift of finite type**

$X = X_F$ where $F$ is finite is called a *shift of finite type*.

A sofic shift can be represented by a finite automaton; a SFT by a local finite automaton.
**Sofic shift**

$X = X_F$ where $F$ is regular is called a *sofic shift*.

**Shift of finite type**

$X = X_F$ where $F$ is finite is called a *shift of finite type*.

A sofic shift can be represented by a finite automaton; a SFT by a local finite automaton.
Example: shift of finite type
Let $A = \{a, b, c\}$. Consider $X_F$ for $F = \{ba, bb, ac, cc\}$. Represented by the following local automaton.

![Automaton Diagram]

1. Let $A = \{a, b, c\}$.
2. Consider $X_F$ for $F = \{ba, bb, ac, cc\}$.
3. Represented by the following local automaton.
Shift conjugacy

Block map
Given $A, B$ alphabets and $X$ shift over $A$. $\Phi : A^\mathbb{Z} \to B^\mathbb{Z}$ is a block map if there are non-negative integers $m, a$ and a function $\phi : A^{m+1+a} \to B$ such that for any $x \in X$ and any $i$

$$\Phi(x)_i = \phi(x_{i-m} \cdots x_{i+a}).$$

Analogously can be defined for one-sided shifts, setting $a = 0$.

Conjugacy
Two shifts $X, Y$ are conjugate if there is a bijective block map $\Phi : X \to Y$. Note: the inverse is also block map.
Shift conjugacy

Block map
Given $A, B$ alphabets and $X$ shift over $A$. $\Phi : A^\mathbb{Z} \to B^\mathbb{Z}$ is a block map if there are non-negative integers $m, a$ and a function $\phi : A^{m+1+a} \to B$ such that for any $x \in X$ and any $i$

$$\Phi(x)_i = \phi(x_{i-m} \cdots x_{i+a}).$$

Analogously can be defined for one-sided shifts, setting $a = 0$.

Conjugacy
Two shifts $X, Y$ are conjugate if there is a bijective block map $\Phi : X \to Y$.
Note: the inverse is also block map.
Shift conjugacy

Block map

Given $A, B$ alphabets and $X$ shift over $A$. $\Phi : A^\mathbb{Z} \to B^\mathbb{Z}$ is a block map if there are non-negative integers $m, a$ and a function $\phi : A^{m+1+a} \to B$ such that for any $x \in X$ and any $i$

$$\Phi(x)_i = \phi(x_{i-m} \cdots x_{i+a}).$$

Analogously can be defined for one-sided shifts, setting $a = 0$.

Conjugacy

Two shifts $X, Y$ are conjugate if there is a bijective block map $\Phi : X \to Y$.

Note: the inverse is also block map.
Shift conjugacy

Block map
Given $A, B$ alphabets and $X$ shift over $A$. $\Phi : A^\mathbb{Z} \to B^\mathbb{Z}$ is a block map if there are non-negative integers $m, a$ and a function $\phi : A^{m+1+a} \to B$ such that for any $x \in X$ and any $i$

$$\Phi(x)_i = \phi(x_{i-m} \cdots x_{i+a}).$$

Analogously can be defined for one-sided shifts, setting $a = 0$.

Conjugacy
Two shifts $X$, $Y$ are conjugate if there is a bijective block map $\Phi : X \to Y$.
Note: the inverse is also block map.
Shift conjugacy

Block map
Given $A, B$ alphabets and $X$ shift over $A$. $\Phi : A^\mathbb{Z} \to B^\mathbb{Z}$ is a block map if there are non-negative integers $m, a$ and a function $\phi : A^{m+1+a} \to B$ such that for any $x \in X$ and any $i$

$$\Phi(x)_i = \phi(x_{i-m} \cdots x_{i+a}).$$

Analogously can be defined for one-sided shifts, setting $a = 0$.

Conjugacy
Two shifts $X, Y$ are conjugate if there is a bijective block map $\Phi : X \to Y$.
Note: the inverse is also block map.
Outline

Standard notions
   Shifts (of finite type)
   Conjugacy

1-sided finite-type-Dyck shifts
   Dyck shifts
   Finite-type-Dyck shifts

Conjugacy for 1-sided FTD shifts
   Deciding conjugacy
Dyck shifts consist of well-formed sequences of brackets.

E.g.

```
⋯))))((()())())⋯

⋯]](()[()]())⋯
```

NOT

```
⋯]](()[[]()())⋯
```

Similar rules can be introduced for any SFT defined on an alphabet of brackets.
Dyck shifts consist of well-formed sequences of brackets.

E.g.

\[
\cdots ))((()))()()\cdots \\
\cdots ]](()[[()]()\cdots \\
\]

NOT

\[
\cdots ]](()[[()]()\cdots \\
\]

Similar rules can be introduced for any SFT defined on an alphabet of brackets.
Dyck shifts consist of well-formed sequences of brackets. 

E.g.

\[
\cdots ) ( ( ( ( ) ) ) \cdots
\]

\[
\cdots ] [ ( ( ] ) ] ( ) \cdots
\]

NOT

\[
\cdots ] [ ( [ ) ) ] ( ) \cdots
\]

Similar rules can be introduced for any SFT defined on an alphabet of brackets.
Dyck shifts consist of well-formed sequences of brackets.

E.g.
\[
\cdots))((())(()))\cdots
\]
\[
\cdots][(()][()])\cdots
\]

NOT
\[
\cdots][(()][]())\cdots
\]

Similar rules can be introduced for any SFT defined on an alphabet of brackets.
Tripartitioned alphabet

Let the alphabet be divided into three disjoint sets of *call*, *return*, and *internal* symbols:

\[ A = A_c \sqcup A_r \sqcup A_i. \]

**Example**

Consider the SFT defined by the automaton, where \( A_c = \{ (, [ \} \), \( A_r = \{ )$, $] \}, \) \( A_d = \{ j, k \} \)
Tripartitioned alphabet

Let the alphabet be divided into three disjoint sets of call, return, and internal symbols:

\[ A = A_c \sqcup A_r \sqcup A_i. \]

Example

Consider the SFT defined by the automaton, where \( A_c = \{ (, [ \} \), \( A_r = \{ ), ] \} \), \( A_d = \{ j, k \} \).
Let’s add Dyck constraints. Prevent

\[ \cdots ( \left[ kjkj \right] ) kj( \cdots \]

from appearing in the shift...

The SFT already avoids some finite set of forbidden factors \( F \). So why not also forbid matching the pairs from the set \( U = \{ \langle (, \rangle, \langle [, \rangle \} \}. \)
Let’s add Dyck constraints. Prevent

\[ \cdots ( [ kjk ] jk ( \cdots \]

from appearing in the shift...

The SFT already avoids some finite set of forbidden factors \( F \). So why not also forbid matching the pairs from the set \( U = \{ \langle , [ \rangle, \langle [ , \rangle \} \).
Let’s add Dyck constraints. Prevent

\[ \cdots ( ( [ kjkj ] ] ) kj ( ( \cdots \]

from appearing in the shift...

The SFT already avoids some finite set of forbidden factors $F$. So why not also forbid matching the pairs from the set $U = \{ \langle ( , ] \rangle, \langle [ , ) \rangle \}$. 
Let’s add Dyck constraints. Prevent

\[ \cdots (\langle[[kj]k]\rangle)kj(\cdots \]

from appearing in the shift...

The SFT already avoids some finite set of forbidden factors \( F \). So why not also forbid matching the pairs from the set \( U = \{\langle(,]\rangle, \langle[,\rangle\rangle\} \). We can record the same information in the automaton by joining edges whose symbols are allowed to match.
But we could (dis)allow any other combination of call and return symbols. Consider

![Diagram]

corresponding to forbidden matching $U = \{ \langle (, ] \rangle \}$. We can also consider context. For example we may allow brackets matching only if at least one of them is preceded by $k$. Hence distinguishing

$$\cdots)jk((jk))(\cdots$$

from

$$\cdots)))((jk))(\cdots$$
But we could (dis)allow any other combination of call and return symbols. Consider

\[
\begin{array}{c}
\text{corresponding to forbidden matching } U = \{ \langle (, [) \rangle \}. \\
\end{array}
\]

We can also consider context. For example we may allow brackets matching only if at least one of them is preceded by \(k\). Hence distinguishing

\[
\cdots jk((jk))\cdots
\]

from

\[
\cdots)))(jk))\cdots
\]
But we could (dis)allow any other combination of call and return symbols. Consider

\[
\begin{align*}
\text{corresponding to forbidden matching } & U = \{ \langle (, ] \rangle \}. \\
\text{We can also consider context. For example we may allow brackets matching only if at least one of them is preceded by } & k. \text{ Hence distinguishing} \\
\cdots & jk((jk))(\cdots
\end{align*}
\]

from

\[
\cdots)))(((jk))(\cdots
\]
But we could (dis)allow any other combination of call and return symbols. Consider

corresponding to forbidden matching $U = \{ \langle(,] \rangle \}$. We can also consider context. For example we may allow brackets matching only if at least one of them is preceded by $k$. Hence distinguishing

$\cdots)j(k ((jk)) (\cdots$

from

$\cdots)))(((jk)) (\cdots$
**1-sided finite-type-Dyck shift**

We call *1-sided finite-type-Dyck shift* over alphabet $A$ any set $X \subseteq A^{-\mathbb{N}}$ that avoids some finite set of forbidden words $F \subseteq A^{m+1}$ and finite set of matching patterns $U \subseteq A^m A_c \times A^m A_r$ for some non-negative integer $m$. We denote $X = X_{F,U}$.

Some notes:

- Equivalently, 1-sided FTD shifts can be defined as left-infinite paths admissible in local automata with arbitrary matching relations between call and return edges. So-called Dyck automata.
- By construction, both (1-sided) Dyck shifts and SFT's are included.
- The language of factors is a visibly pushdown language.
1-sided finite-type-Dyck shift

We call 1-sided finite-type-Dyck shift over alphabet $A$ any set $X \subseteq A^{-\mathbb{N}}$ that avoids some finite set of forbidden words $F \subseteq A^{m+1}$ and finite set of matching patterns $U \subseteq A^m A_c \times A^m A_r$ for some non-negative integer $m$. We denote $X = X_{F,U}$.

Some notes:

- Equivalently, 1-sided FTD shifts can be defined as left-infinite paths admissible in local automata with arbitrary matching relations between call and return edges. So-called Dyck automata.
- By construction, both (1-sided) Dyck shifts and SFT's are included.
- The language of factors is a visibly pushdown language.
1-sided finite-type-Dyck shift

We call 1-sided finite-type-Dyck shift over alphabet $A$ any set $X \subseteq A^{-\mathbb{N}}$ that avoids some finite set of forbidden words $F \subseteq A^{m+1}$ and finite set of matching patterns $U \subseteq A^m A_c \times A^m A_r$ for some non-negative integer $m$. We denote $X = X_{F,U}$.

Some notes:

- Equivalently, 1-sided FTD shifts can be defined as left-infinite paths admissible in local automata with arbitrary matching relations between call and return edges. So-called Dyck automata.
- By construction, both (1-sided) Dyck shifts and SFT's are included.
- The language of factors is a visibly pushdown language.
1-sided finite-type-Dyck shift

We call 1-sided finite-type-Dyck shift over alphabet $A$ any set $X \subseteq A^{-\mathbb{N}}$ that avoids some finite set of forbidden words $F \subseteq A^{m+1}$ and finite set of matching patterns $U \subseteq A^mA_c \times A^mA_r$ for some non-negative integer $m$. We denote $X = X_{F,U}$.

Some notes:

- Equivalently, 1-sided FTD shifts can be defined as left-infinite paths admissible in local automata with arbitrary matching relations between call and return edges. So-called Dyck automata.
- By construction, both (1-sided) Dyck shifts and SFT's are included.
- The language of factors is a visibly pushdown language.
1-sided finite-type-Dyck shift

We call 1-sided finite-type-Dyck shift over alphabet $A$ any set $X \subseteq A^{-\mathbb{N}}$ that avoids some finite set of forbidden words $F \subseteq A^{m+1}$ and finite set of matching patterns $U \subseteq A^m A_c \times A^m A_r$ for some non-negative integer $m$. We denote $X = X_{F,U}$.

Some notes:

- Equivalently, 1-sided FTD shifts can be defined as left-infinite paths admissible in local automata with arbitrary matching relations between call and return edges. So-called Dyck automata.
- By construction, both (1-sided) Dyck shifts and SFT's are included.
- The language of factors is a visibly pushdown language.
1-sided finite-type-Dyck shift

We call 1-sided finite-type-Dyck shift over alphabet $A$ any set $X \subseteq A^{-\mathbb{N}}$ that avoids some finite set of forbidden words $F \subseteq A^{m+1}$ and finite set of matching patterns $U \subseteq A^m A_c \times A^m A_r$ for some non-negative integer $m$. We denote $X = X_{F,U}$.

Some notes:

- Equivalently, 1-sided FTD shifts can be defined as left-infinite paths admissible in local automata with arbitrary matching relations between call and return edges. So-called Dyck automata.
- By construction, both (1-sided) Dyck shifts and SFT’s are included.
- The language of factors is a visibly pushdown language.
Outline

Standard notions
  Shifts (of finite type)
  Conjugacy

1-sided finite-type-Dyck shifts
  Dyck shifts
  Finite-type-Dyck shifts

Conjugacy for 1-sided FTD shifts
  Deciding conjugacy
Proper block map
A block map $\Phi$ is proper when $\Phi(x)_j \in A_c$ (resp. $A_r, A_i$) if and only if $x_j \in A_c$ (resp. $A_r, A_i$).

Matched-return word
A word over tripartitioned alphabet $A$ is matched-return if it ”has no unmatched return symbols”. Formally, if each its prefix contains at least as many call symbols as return symbols.

MR-extensible shift
A shift $X$ is called MR-extensible if for any of its blocks, $u$ exists a non-empty matched-return word $v$ such that $uv$ is a block of $X$. 
Proper block map
A block map $\Phi$ is *proper* when $\Phi(x)_j \in A_c$ (resp. $A_r, A_i$) if and only if $x_j \in A_c$ (resp. $A_r, A_i$).

Matched-return word
A word over tripartitioned alphabet $A$ is *matched-return* if it ”has no unmatched return symbols”. Formally, if each its prefix contains at least as many call symbols as return symbols.

MR-extensible shift
A shift $X$ is called *MR-extensible* if for any of its blocks, $u$ exists a non-empty matched-return word $v$ such that $uv$ is a block of $X$. 
Proper block map
A block map $\Phi$ is proper when $\Phi(x)_j \in A_c$ (resp. $A_r, A_i$) if and only if $x_j \in A_c$ (resp. $A_r, A_i$).

Matched-return word
A word over tripartitioned alphabet $A$ is matched-return if it "has no unmatched return symbols". Formally, if each its prefix contains at least as many call symbols as return symbols.

MR-extensible shift
A shift $X$ is called MR-extensible if for any of its blocks, $u$ exists a non-empty matched-return word $v$ such that $uv$ is a block of $X$. 
Proper block map
A block map \( \Phi \) is *proper* when \( \Phi(x)_j \in A_c \) (resp. \( A_r, A_i \)) if and only if \( x_j \in A_c \) (resp. \( A_r, A_i \)).

Matched-return word
A word over tripartitioned alphabet \( A \) is *matched-return* if it "has no unmatched return symbols". Formally, if each its prefix contains at least as many call symbols as return symbols.

MR-extensible shift
A shift \( X \) is called *MR-extensible* if for any of its blocks, \( u \) exists a non-empty matched-return word \( v \) such that \( uv \) is a block of \( X \).
The result

Theorem
It is decidable in an effective way whether two one-sided finite-type-Dyck shifts which are MR-extensible are properly conjugate.

- Not all 1-sided FTD shifts are MR-extensible, but many are.
- Dyck and Motzkin shifts are MR-extensible.
- Our examples were MR-extensible.
The result

Theorem
It is decidable in an effective way whether two one-sided finite-type-Dyck shifts which are MR-extensible are properly conjugate.

- Not all 1-sided FTD shifts are MR-extensible, but many are.
- Dyck and Motzkin shifts are MR-extensible.
- Our examples were MR-extensible.
Deciding the conjugacy

In the process of deciding, the operations of *in-splitting* and *in-merging* Dyck graphs (multigraph with matching of some edges) are crucial.
Deciding the conjugacy

During the in-splitting a state is divided into two. Each new state receives a precise copy of its out-going edges (including the matching relations. The original in-coming edges are partitioned between the two new states.
Deciding the conjugacy

During the in-splitting a state is divided into two. Each new state receives a precise copy of its out-going edges (including the matching relations). The original in-coming edges are partitioned between the two new states.
Deciding the conjugacy

During the in-splitting a state is divided into two. Each new state receives a precise copy of its out-going edges (including the matching relations. The original in-coming edges are partitioned between the two new states.
Deciding the conjugacy

To decide conjugacy of two 1-sided FTD’S given by finite sets $F \subseteq A^{m+1}, U \subseteq A^m A_c \times A^m A_r$:

- construct from $F, U$, automaton of certain form recognizing the shift
- from now on disregard labels, work only with the underlying Dyck graph
- perform in-merges in the two graphs as long as possible
- check, whether they are isomorphic (in the Dyck graph sense, i.e. respecting the edge matching)
Deciding the conjugacy

To decide conjugacy of two 1-sided FTD’S given by finite sets $F \subseteq A^{m+1}, U \subseteq A^m A_c \times A^m A_r$:

- construct from $F, U$, automaton of certain form recognizing the shift
- from now on disregard labels, work only with the underlying Dyck graph
- perform in-merges in the two graphs as long as possible
- check, whether they are isomorphic (in the Dyck graph sense, i.e. respecting the edge matching)
Deciding the conjugacy

To decide conjugacy of two 1-sided FTD’s given by finite sets $F \subseteq A^{m+1}$, $U \subseteq A^m A_c \times A^m A_r$:

- construct from $F$, $U$, automaton of certain form recognizing the shift
- from now on disregard labels, work only with the underlying Dyck graph
- perform in-merges in the two graphs as long as possible
- check, whether they are isomorphic (in the Dyck graph sense, i.e. respecting the edge matching)
Deciding the conjugacy

To decide conjugacy of two 1-sided FTD’S given by finite sets \( F \subseteq A^{m+1}, U \subseteq A^m A_c \times A^m A_r \):

- construct from \( F, U \), automaton of certain form recognizing the shift
- from now on disregard labels, work only with the underlying Dyck graph
- perform in-merges in the two graphs as long as possible
- check, whether they are isomorphic (in the Dyck graph sense, i.e. respecting the edge matching)
Deciding the conjugacy

To decide conjugacy of two 1-sided FTD’S given by finite sets $F \subseteq A^{m+1}$, $U \subseteq A^m A_c \times A^m A_r$:

- construct from $F$, $U$, automaton of certain form recognizing the shift
- from now on disregard labels, work only with the underlying Dyck graph
- perform in-merges in the two graphs as long as possible
- check, whether they are isomorphic (in the Dyck graph sense, i.e. respecting the edge matching)