# On a question of Bob Gilman: Multi-pass Automata and Group Word Problems

Paul Schupp

University of Illinois

at Urbana-Champaign

**Joint work with**

**Tullio Ceccherini-Silberstein, Michel Coornaert, Francesca Fiorenzi**

September 2012, Hoboken

# The Chomsky Hierarchy of Formal Languages

Let $\Sigma$ be a finite *alphabet*. A *word* over $\Sigma$ is a finite sequence of elements of $\Sigma$, ie. a finite sequence of letters. $\Sigma^*$ denotes the set of all words over $\Sigma$. With the operation of concatenation of words, $\Sigma^*$ is the free monoid over $\Sigma$. A *language* $L$ is a subset of $\Sigma^*$.

1. *Regular languages* are the languages accepted by finite automata.
2. *Context-free languages* are the languages accepted by pushdown automata.
3. *Context-sensitive languages* are the languages accepted by linear bounded Turing machines.
   This is the same as the class of languages in linear space.
4. *Computably enumerable* languages are the languages accepted by Turing machines.

# The Chomsky Hierarchy of Formal Languages

Let $\Sigma$ be a finite *alphabet*. A *word* over $\Sigma$ is a finite sequence of elements of $\Sigma$, ie. a finite sequence of letters. $\Sigma^*$ denotes the set of all words over $\Sigma$. With the operation of concatenation of words, $\Sigma^*$ is the free monoid over $\Sigma$. A *language* $L$ is a subset of $\Sigma^*$.

1. *Regular languages* are the languages accepted by finite automata.
2. *Context-free languages* are the languages accepted by pushdown automata.
3. *Context-sensitive languages* are the languages accepted by linear bounded Turing machines.
   This is the same as the class of languages in linear space.
4. *Computably enumerable* languages are the languages accepted by Turing machines.

# The Chomsky Hierarchy of Formal Languages

Let $\Sigma$ be a finite *alphabet*. A *word* over $\Sigma$ is a finite sequence of elements of $\Sigma$, ie. a finite sequence of letters. $\Sigma^*$ denotes the set of all words over $\Sigma$. With the operation of concatenation of words, $\Sigma^*$ is the free monoid over $\Sigma$. A *language* $L$ is a subset of $\Sigma^*$.

1. *Regular languages* are the languages accepted by finite automata.
2. *Context-free languages* are the languages accepted by pushdown automata.
3. *Context-sensitive languages* are the languages accepted by linear bounded Turing machines.
   This is the same as the class of languages in linear space.
4. *Computably enumerable* languages are the languages accepted by Turing machines.

# The Chomsky Hierarchy of Formal Languages

Let $\Sigma$ be a finite *alphabet*. A *word* over $\Sigma$ is a finite sequence of elements of $\Sigma$, ie. a finite sequence of letters. $\Sigma^*$ denotes the set of all words over $\Sigma$. With the operation of concatenation of words, $\Sigma^*$ is the free monoid over $\Sigma$. A *language* $L$ is a subset of $\Sigma^*$.

1. *Regular languages* are the languages accepted by finite automata.
2. *Context-free languages* are the languages accepted by pushdown automata.
3. *Context-sensitive languages* are the languages accepted by linear bounded Turing machines.
   This is the same as the class of languages in linear space.
4. *Computably enumerable* languages are the languages accepted by Turing machines.

# The Chomsky Hierarchy of Formal Languages

Let $\Sigma$ be a finite *alphabet*. A *word* over $\Sigma$ is a finite sequence of elements of $\Sigma$, ie. a finite sequence of letters. $\Sigma^*$ denotes the set of all words over $\Sigma$. With the operation of concatenation of words, $\Sigma^*$ is the free monoid over $\Sigma$. A *language* $L$ is a subset of $\Sigma^*$.

1. *Regular languages* are the languages accepted by finite automata.
2. *Context-free languages* are the languages accepted by pushdown automata.
3. *Context-sensitive languages* are the languages accepted by linear bounded Turing machines.
   This is the same as the class of languages in linear space.
4. *Computably enumerable* languages are the languages accepted by Turing machines.

# The Chomsky Hierarchy of Formal Languages

Let $\Sigma$ be a finite *alphabet*. A *word* over $\Sigma$ is a finite sequence of elements of $\Sigma$, ie. a finite sequence of letters. $\Sigma^*$ denotes the set of all words over $\Sigma$. With the operation of concatenation of words, $\Sigma^*$ is the free monoid over $\Sigma$. A *language* $L$ is a subset of $\Sigma^*$.

1. *Regular languages* are the languages accepted by finite automata.
2. *Context-free languages* are the languages accepted by pushdown automata.
3. *Context-sensitive languages* are the languages accepted by linear bounded Turing machines.
   This is the same as the class of languages in linear space.
4. *Computably enumerable* languages are the languages accepted by Turing machines.

# Group Word Problems and Formal Languages

If we have a finitely generated presentation of a group $G = \langle X : R \rangle$, we describe elements of $G$ as words in the *group alphabet* $\Sigma = X \cup X^{-1}$. The Russian computer scientist Anisimov, in 1973, introduced the point of view of considering the word problem of $G$ as a formal language. So define *the word problem* of $G$ to be the formal language

$WP(G) = \{w \in \Sigma^* : w = 1\}$ in $G$. What does formal language theory

have to do with group theory? How do the formal language properties of $WP(G)$ relate to the algebric properties of $G$?

1. Thm. (Anisimov) $WP(G)$ is a regular language if and only if $G$ is finite.
2. Thm. (Muller - S) $WP(G)$ is a context-free language if and only if $G$ is virtualy free.
3. Thm. (Higman Embedding Theorem) $WP(G)$ is a computably enumerable language if and only if $G$ can be embedded in a finitely presented group.

# Group Word Problems and Formal Languages

If we have a finitely generated presentation of a group $G = \langle X : R \rangle$, we describe elements of $G$ as words in the *group alphabet* $\Sigma = X \cup X^{-1}$. The Russian computer scientist Anisimov, in 1973, introduced the point of view of considering the word problem of $G$ as a formal language. So define *the word problem* of $G$ to be the formal language

$WP(G) = \{w \in \Sigma^* : w = 1\}$ in $G$. What does formal language theory

have to do with group theory? How do the formal language properties of $WP(G)$ relate to the algebric properties of $G$?

1. Thm. (Anisimov) $WP(G)$ is a regular language if and only if $G$ is finite.
2. Thm. (Muller - S) $WP(G)$ is a context-free language if and only if $G$ is virtualy free.
3. Thm. (Higman Embedding Theorem) $WP(G)$ is a computably enumerable language if and only if $G$ can be embedded in a finitely presented group.

# Group Word Problems and Formal Languages

If we have a finitely generated presentation of a group $G = \langle X : R \rangle$, we describe elements of $G$ as words in the *group alphabet* $\Sigma = X \cup X^{-1}$. The Russian computer scientist Anisimov, in 1973, introduced the point of view of considering the word problem of $G$ as a formal language. So define *the word problem* of $G$ to be the formal language

$WP(G) = \{w \in \Sigma^* : w = 1\}$ in $G$. What does formal language theory have to do with group theory? How do the formal language properties of $WP(G)$ relate to the algebric properties of $G$?

1. Thm. (Anisimov) $WP(G)$ is a regular language if and only if $G$ is finite.
2. Thm. (Muller - S) $WP(G)$ is a context-free language if and only if $G$ is virtualy free.
3. Thm. (Higman Embedding Theorem) $WP(G)$ is a computably enumerable language if and only if $G$ can be embedded in a finitely presented group.

# Group Word Problems and Formal Languages

If we have a finitely generated presentation of a group $G = \langle X : R \rangle$, we describe elements of $G$ as words in the *group alphabet* $\Sigma = X \cup X^{-1}$. The Russian computer scientist Anisimov, in 1973, introduced the point of view of considering the word problem of $G$ as a formal language. So define *the word problem* of $G$ to be the formal language

$WP(G) = \{w \in \Sigma^* : w = 1\}$ in $G$. What does formal language theory have to do with group theory? How do the formal language properties of $WP(G)$ relate to the algebric properties of $G$?

1. Thm. (Anisimov) $WP(G)$ is a regular language if and only if $G$ is finite.
2. Thm. (Muller - S) $WP(G)$ is a context-free language if and only if $G$ is virtualy free.
3. Thm. (Higman Embedding Theorem) $WP(G)$ is a computably enumerable language if and only if $G$ can be embedded in a finitely presented group.

# Group Word Problems and Formal Languages

If we have a finitely generated presentation of a group $G = \langle X : R \rangle$, we describe elements of $G$ as words in the *group alphabet* $\Sigma = X \cup X^{-1}$. The Russian computer scientist Anisimov, in 1973, introduced the point of view of considering the word problem of $G$ as a formal language. So define *the word problem* of $G$ to be the formal language

$WP(G) = \{w \in \Sigma^* : w = 1\}$ in $G$. What does formal language theory

have to do with group theory? How do the formal language properties of $WP(G)$ relate to the algebric properties of $G$?

1. Thm. (Anisimov) $WP(G)$ is a regular language if and only if $G$ is finite.
2. Thm. (Muller - S) $WP(G)$ is a context-free language if and only if $G$ is virtualy free.
3. Thm. (Higman Embedding Theorem) $WP(G)$ is a computably enumerable language if and only if $G$ can be embedded in a finitely presented group.

# Group Word Problems and Formal Languages

If we have a finitely generated presentation of a group $G = \langle X : R \rangle$, we describe elements of $G$ as words in the *group alphabet* $\Sigma = X \cup X^{-1}$. The Russian computer scientist Anisimov, in 1973, introduced the point of view of considering the word problem of $G$ as a formal language. So define *the word problem* of $G$ to be the formal language

$WP(G) = \{w \in \Sigma^* : w = 1\}$ in $G$. What does formal language theory

have to do with group theory? How do the formal language properties of $WP(G)$ relate to the algebric properties of $G$?

1. Thm. (Anisimov) $WP(G)$ is a regular language if and only if $G$ is finite.
2. Thm. (Muller - S) $WP(G)$ is a context-free language if and only if $G$ is virtualy free.
3. Thm. (Higman Embedding Theorem) $WP(G)$ is a computably enumerable language if and only if $G$ can be embedded in a finitely presented group.

# Group Word Problems and Formal Languages

If we have a finitely generated presentation of a group $G = \langle X : R \rangle$, we describe elements of $G$ as words in the *group alphabet* $\Sigma = X \cup X^{-1}$. The Russian computer scientist Anisimov, in 1973, introduced the point of view of considering the word problem of $G$ as a formal language. So define *the word problem* of $G$ to be the formal language

$WP(G) = \{w \in \Sigma^* : w = 1\}$ in $G$. What does formal language theory

have to do with group theory? How do the formal language properties of $WP(G)$ relate to the algebric properties of $G$?

1. Thm. (Anisimov) $WP(G)$ is a regular language if and only if $G$ is finite.
2. Thm. (Muller - S) $WP(G)$ is a context-free language if and only if $G$ is virtualy free.
3. Thm. (Higman Embedding Theorem) $WP(G)$ is a computably enumerable language if and only if $G$ can be embedded in a finitely presented group.

# Group Word Problems and Formal Languages

If we have a finitely generated presentation of a group $G = \langle X : R \rangle$, we describe elements of $G$ as words in the *group alphabet* $\Sigma = X \cup X^{-1}$. The Russian computer scientist Anisimov, in 1973, introduced the point of view of considering the word problem of $G$ as a formal language. So define *the word problem* of $G$ to be the formal language

$WP(G) = \{w \in \Sigma^* : w = 1\}$ in $G$. What does formal language theory

have to do with group theory? How do the formal language properties of $WP(G)$ relate to the algebric properties of $G$?

1. Thm. (Anisimov) $WP(G)$ is a regular language if and only if $G$ is finite.
2. Thm. (Muller - S) $WP(G)$ is a context-free language if and only if $G$ is virtualy free.
3. Thm. (Higman Embedding Theorem) $WP(G)$ is a computably enumerable language if and only if $G$ can be embedded in a finitely presented group.

# Linear space is very difficult to deal with

It is very difficult to make any definitive statements about linear space. The famous example is the “LBA Problem”, the question of whether or not the classs of languages in linear space is closed under complementation. This was an open problem for more than twenty years. Everyone thought the answer was “No” but could not prove the result. Then at essentially the same time, Neil Immerman and

Szelepcsenyi really believed that the correct answer was “Yes”, in which case they just wrote down the proof. The proof is only about two pages and no only does not use any result proved in the intervening twenty years, it does not even introduce any new definitions. In fact, they proved that any reasonable space class is closed under complementation.

Whether or not nondeterminstic linear bounded automata are equivalent to deterministic linear bounded automata is still open.

# Linear space is very difficult to deal with

It is very difficult to make any definitive statements about linear space. The famous example is the “LBA Problem”, the question of whether or not the classs of languages in linear space is closed under complementation. This was an open problem for more than twenty years. Everyone thought the answer was “No” but could not prove the result. Then at essentially the same time, Neil Immerman and

Szelepcsenyi really believed that the correct answer was “Yes”, in which case they just wrote down the proof. The proof is only about two pages and no only does not use any result proved in the intervening twenty years, it does not even introduce any new definitions. In fact, they proved that any reasonable space class is closed under complementation.

Whether or not nondeterminstic linear bounded automata are equivalent to deterministic linear bounded automata is still open.

# Linear space is very difficult to deal with

It is very difficult to make any definitive statements about linear space. The famous example is the “LBA Problem”, the question of whether or not the classs of languages in linear space is closed under complementation. This was an open problem for more than twenty years. Everyone thought the answer was “No” but could not prove the result. Then at essentially the same time, Neil Immerman and

Szelepcsenyi really believed that the correct answer was “Yes”, in which case they just wrote down the proof. The proof is only about two pages and no only does not use any result proved in the intervening twenty years, it does not even introduce any new definitions. In fact, they proved that any reasonable space class is closed under complementation.

Whether or not nondeterminstic linear bounded automata are equivalent to deterministic linear bounded automata is still open.

# Linear space is very difficult to deal with

It is very difficult to make any definitive statements about linear space. The famous example is the “LBA Problem”, the question of whether or not the classs of languages in linear space is closed under complementation. This was an open problem for more than twenty years. Everyone thought the answer was “No” but could not prove the result. Then at essentially the same time, Neil Immerman and

Szelepcsenyi really believed that the correct answer was “Yes”, in which case they just wrote down the proof. The proof is only about two pages and no only does not use any result proved in the intervening twenty years, it does not even introduce any new definitions. In fact, they proved that any reasonable space class is closed under complementation.

Whether or not nondeterminstic linear bounded automata are equivalent to deterministic linear bounded automata is still open.

# Linear space is very difficult to deal with

It is very difficult to make any definitive statements about linear space. The famous example is the “LBA Problem”, the question of whether or not the classs of languages in linear space is closed under complementation. This was an open problem for more than twenty years. Everyone thought the answer was “No” but could not prove the result. Then at essentially the same time, Neil Immerman and

Szelepcsenyi really believed that the correct answer was “Yes”, in which case they just wrote down the proof. The proof is only about two pages and no only does not use any result proved in the intervening twenty years, it does not even introduce any new definitions. In fact, they proved that any reasonable space class is closed under complementation.

Whether or not nondeterminstic linear bounded automata are equivalent to deterministic linear bounded automata is still open.

# Too many group word problems

Most garden variety groups have their word problems in linear space. If a language is in linear space then it is decidable in single exponential time. So if $WP(G) \notin EXPTIME(n)$ then $WP(G)$ is not in linear space.

Bob Gilman has asked if there is a class of formal languages more general than context-free languages but less general than linear bounded languages for which one can say something about group word problems in the class. After a very preliminary look, the following class of languages at least seems interesting in that respect.

# Too many group word problems

Most garden variety groups have their word problems in linear space. If a language is in linear space then it is decidable in single exponential time. So if $WP(G) \notin EXPTIME(n)$ then $WP(G)$ is not in linear space.

Bob Gilman has asked if there is a class of formal languages more general than context-free languages but less general than linear bounded languages for which one can say something about group word problems in the class. After a very preliminary look, the following class of languages at least seems interesting in that respect.

# Too many group word problems

Most garden variety groups have their word problems in linear space. If a language is in linear space then it is decidable in single exponential time. So if $WP(G) \notin EXPTIME(n)$ then $WP(G)$ is not in linear space.

Bob Gilman has asked if there is a class of formal languages more general than context-free languages but less general than linear bounded languages for which one can say something about group word problems in the class. After a very preliminary look, the following class of languages at least seems interesting in that respect.

# Too many group word problems

Most garden variety groups have their word problems in linear space. If a language is in linear space then it is decidable in single exponential time. So if $WP(G) \notin EXPTIME(n)$ then $WP(G)$ is not in linear space.

Bob Gilman has asked if there is a class of formal languages more general than context-free languages but less general than linear bounded languages for which one can say something about group word problems in the class. After a very preliminary look, the following class of languages at least seems interesting in that respect.

# Multi-Pass Automata

Roughly speaking, a deterministic $k$-pass automaton $M$ works like an ordinary deterministic PDA in that it can only move forward on the read-only input tape, and has a pushdown stack and can read only the top letter on the stack. However, the automaton can read the input tape $k$-times. If $k = 1$ the machine is just a deterministic pushdown automaton.

There is a special right end-marker, denoted $\sharp$, which marks the end of an input word. There is a counter keeping track of which pass the automaton is on in reading the input. If the automaton reads the end marker $\sharp$ and the number of passes so far is less than $k$, then, depending on the number of the pass, on the control state and the top of the stack (including the case that the stack is empty), the machine changes state, the pass-counter is increased by1, and the reading head is reset to the beginning of the tape.

# Multi-Pass Automata

Roughly speaking, a deterministic $k$-pass automaton $M$ works like an ordinary deterministic PDA in that it can only move forward on the read-only input tape, and has a pushdown stack and can read only the top letter on the stack. However, the automaton can read the input tape $k$-times. If $k = 1$ the machine is just a deterministic pushdown automaton.

There is a special right end-marker, denoted $\sharp$, which marks the end of an input word. There is a counter keeping track of which pass the automaton is on in reading the input. If the automaton reads the end marker $\sharp$ and the number of passes so far is less than $k$, then, depending on the number of the pass, on the control state and the top of the stack (including the case that the stack is empty), the machine changes state, the pass-counter is increased by1, and the reading head is reset to the beginning of the tape.

# Multi-Pass Automata

Roughly speaking, a deterministic $k$-pass automaton $M$ works like an ordinary deterministic PDA in that it can only move forward on the read-only input tape, and has a pushdown stack and can read only the top letter on the stack. However, the automaton can read the input tape $k$-times. If $k = 1$ the machine is just a deterministic pushdown automaton.

There is a special right end-marker, denoted $\sharp$, which marks the end of an input word. There is a counter keeping track of which pass the automaton is on in reading the input. If the automaton reads the end marker $\sharp$ and the number of passes so far is less than $k$, then, depending on the number of the pass, on the control state and the top of the stack (including the case that the stack is empty), the machine changes state, the pass-counter is increased by1, and the reading head is reset to the beginning of the tape.

The automaton has two special halt states, $H_a$ which is accepting and $H_r$ which is rejecting. If the automaton reads the end-marker on pass $k$ then the machine, depending on its state and the top of the stack, halts in either $H_a$ or $H_r$. The machine $M$ *accepts* an input exactly if it halts in the accepting state $H_a$ on its final pass. As usual the language $L(M)$ accepted by $M$ is the set of all words accepted by $M$.

The automaton has two special halt states, $H_a$ which is accepting and $H_r$ which is rejecting. If the automaton reads the end-marker on pass $k$ then the machine, depending on its state and the top of the stack, halts in either $H_a$ or $H_r$. The machine $M$ *accepts* an input exactly if it halts in the accepting state $H_a$ on its final pass. As usual the language $L(M)$ accepted by $M$ is the set of all words accepted by $M$.

# An Example

Since we are interested in group word problems, our automata can continue working when they encounter an empty stack. As a

motivating example consider the word problem for the free abelian group of rank two.

Let $G = \mathbb{Z}^2$ with presentation $G = \langle a, b; ab = ba \rangle$.

The associated word problem is then the language consisting of words which have exponent sum 0 on both $a$ and $b$.

This word problem is accepted by a 2-pass automaton $M$.
One the first pass $M$ checks if the exponent sum on $a$ in $w$ is 0.
One the second pass $M$ checks if the exponent sum on $b$ in $w$ is 0.
$M$ accepts at the end of the second pass if and only if both conditions are met.

# An Example

Since we are interested in group word problems, our automata can continue working when they encounter an empty stack. As a motivating example consider the word problem for the free abelian group of rank two.

Let $G = \mathbb{Z}^2$ with presentation $G = \langle a, b; ab = ba \rangle$.

The associated word problem is then the language consisting of words which have exponent sum 0 on both $a$ and $b$.

This word problem is accepted by a 2-pass automaton $M$.
One the first pass $M$ checks if the exponent sum on $a$ in $w$ is 0.
One the second pass $M$ checks if the exponent sum on $b$ in $w$ is 0.
$M$ accepts at the end of the second pass if and only if both conditions are met.

# An Example

Since we are interested in group word problems, our automata can continue working when they encounter an empty stack. As a

motivating example consider the word problem for the free abelian group of rank two.

Let $G = \mathbb{Z}^2$ with presentation $G = \langle a, b; ab = ba \rangle$.

The associated word problem is then the language consisting of words which have exponent sum 0 on both $a$ and $b$.

This word problem is accepted by a 2-pass automaton $M$.
One the first pass $M$ checks if the exponent sum on $a$ in $w$ is 0.
One the second pass $M$ checks if the exponent sum on $b$ in $w$ is 0.
$M$ accepts at the end of the second pass if and only if both conditions are met.

# An Example

Since we are interested in group word problems, our automata can continue working when they encounter an empty stack. As a

motivating example consider the word problem for the free abelian group of rank two.

Let $G = \mathbb{Z}^2$ with presentation $G = \langle a, b; ab = ba \rangle$.

The associated word problem is then the language consisting of words which have exponent sum 0 on both $a$ and $b$.

This word problem is accepted by a 2-pass automaton $M$.
One the first pass $M$ checks if the exponent sum on $a$ in $w$ is 0.
One the second pass $M$ checks if the exponent sum on $b$ in $w$ is 0.
$M$ accepts at the end of the second pass if and only if both conditions are met.

# An Example

Since we are interested in group word problems, our automata can continue working when they encounter an empty stack. As a

motivating example consider the word problem for the free abelian group of rank two.

Let $G = \mathbb{Z}^2$ with presentation $G = \langle a, b; ab = ba \rangle$.

The associated word problem is then the language consisting of words which have exponent sum 0 on both $a$ and $b$.

This word problem is accepted by a 2-pass automaton $M$.
One the first pass $M$ checks if the exponent sum on $a$ in $w$ is 0.
One the second pass $M$ checks if the exponent sum on $b$ in $w$ is 0.
$M$ accepts at the end of the second pass if and only if both conditions are met.

# A Formal Definition of Multi-pass Automata

Let $\Sigma$ be a finite alphabet and let $k \geq 1$ be a positive integer. A *k-pass automaton* is a -tuple

$$M = (\{1, ..., k\}, Q, \Sigma, \Gamma, \sharp, \delta, q_0, \{H_a, H_r\})$$

where as usual, $Q$ is a finite set of *states*,
$\Sigma$ is the *input alphabet*,
$\Gamma \supseteq \Sigma$ is the *stack alphabet*
and $q_0 \in Q$ is the *initial state*. The *end-marker* $\sharp$ is a letter not in $\Gamma$. and

we assume that all input words end with $\sharp$
The distinct states $H_a$ and $H_r$ are not in $Q$. The *transition function* $\delta$ is

mainly a function

$$\delta \colon Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to Q \times (\{\varepsilon\} \cup \Gamma \cup \Gamma^2)$$

# A Formal Definition of Multi-pass Automata

Let $\Sigma$ be a finite alphabet and let $k \geq 1$ be a positive integer. A *k-pass automaton* is a -tuple

$$M = (\{1, ..., k\}, Q, \Sigma, \Gamma, \sharp, \delta, q_0, \{H_a, H_r\})$$

where as usual, $Q$ is a finite set of *states*,
$\Sigma$ is the *input alphabet*,
$\Gamma \supseteq \Sigma$ is the *stack alphabet*
and $q_0 \in Q$ is the *initial state*. The *end-marker* $\sharp$ is a letter not in $\Gamma$. and we assume that all input words end with $\sharp$
The distinct states $H_a$ and $H_r$ are not in $Q$. The *transition function* $\delta$ is mainly a function

$$\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to Q \times (\{\varepsilon\} \cup \Gamma \cup \Gamma^2)$$

# A Formal Definition of Multi-pass Automata

Let $\Sigma$ be a finite alphabet and let $k \geq 1$ be a positive integer. A *k-pass automaton* is a -tuple

$$M = (\{1, ..., k\}, Q, \Sigma, \Gamma, \sharp, \delta, q_0, \{H_a, H_r\})$$

where as usual, $Q$ is a finite set of *states*,
$\Sigma$ is the *input alphabet*,
$\Gamma \supseteq \Sigma$ is the *stack alphabet*
and $q_0 \in Q$ is the *initial state*. The *end-marker* $\sharp$ is a letter not in $\Gamma$. and

we assume that all input words end with $\sharp$
The distinct states $H_a$ and $H_r$ are not in $Q$. The *transition function* $\delta$ is

mainly a function

$$\delta \colon Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow Q \times (\{\varepsilon\} \cup \Gamma \cup \Gamma^2)$$

# A Formal Definition of Multi-pass Automata

Let $\Sigma$ be a finite alphabet and let $k \geq 1$ be a positive integer. A *k-pass automaton* is a -tuple

$$M = (\{1, ..., k\}, Q, \Sigma, \Gamma, \sharp, \delta, q_0, \{H_a, H_r\})$$

where as usual, $Q$ is a finite set of *states*,
$\Sigma$ is the *input alphabet*,
$\Gamma \supseteq \Sigma$ is the *stack alphabet*
and $q_0 \in Q$ is the *initial state*. The *end-marker* $\sharp$ is a letter not in $\Gamma$. and

we assume that all input words end with $\sharp$
The distinct states $H_a$ and $H_r$ are not in $Q$. The *transition function* $\delta$ is

mainly a function

$$\delta \colon Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to Q \times (\{\varepsilon\} \cup \Gamma \cup \Gamma^2)$$

There are two kinds of transitions here. The interpretation of

$$\delta(q, \sigma, \gamma) = (q', \lambda)$$

where $q, q' \in Q$, $\sigma \in \Sigma$, and $\gamma \in \Gamma \cup \{\varepsilon\}$ is that
if the machine is in state $q$ and reading the letter $\sigma$ on the input tape with $\gamma$ or $\varepsilon$ on top of the stack,
then the the automaton changes state to $q'$, replaces $\gamma$ by $\lambda$ and advances the input tape. Since we are considering deterministic machines, there is no loss of generality in having the automaton either delete the top letter, or rewrite it, or rewrite it and add a single letter.

There are two kinds of transitions here. The interpretation of

$$\delta(q, \sigma, \gamma) = (q', \lambda)$$

where $q, q' \in Q$, $\sigma \in \Sigma$, and $\gamma \in \Gamma \cup \{\varepsilon\}$ is that
if the machine is in state $q$ and reading the letter $\sigma$ on the input tape with $\gamma$ or $\varepsilon$ on top of the stack,
then the the automaton changes state to $q'$, replaces $\gamma$ by $\lambda$ and advances the input tape. Since we are considering deterministic

machines, there is no loss of generality in having the automaton either delete the top letter, or rewrite it, or rewrite it and add a single letter.

The interpretation of

$$\delta(q, \varepsilon, \gamma) = (q', \lambda)$$

where $q, q' \in Q$, $\sigma \in \Sigma$, and $\gamma \in \Gamma$ is that
if the machine is in state $q$
with $\gamma$ on top of the stack,
then, *independent* of the input letter
the the automaton changes state to $q'$, replaces $\gamma$ by $\lambda$.
In this case the input tape is NOT advanced. Such transitions are called $\varepsilon$-transitions. Since we are considering deterministic machines,

there are not both transitions $\delta(q, \sigma, \gamma)$ and $\delta(q, \varepsilon, \gamma)$. Note that the machine cannot make an $\varepsilon$-transition on empty stack. We will need this information later.

The interpretation of

$$\delta(q, \varepsilon, \gamma) = (q', \lambda)$$

where $q, q' \in Q$, $\sigma \in \Sigma$, and $\gamma \in \Gamma$ is that
if the machine is in state $q$
with $\gamma$ on top of the stack,
then, *independent* of the input letter
the the automaton changes state to $q'$, replaces $\gamma$ by $\lambda$.
In this case the input tape is NOT advanced. Such transitions are called $\varepsilon$-transitions. Since we are considering deterministic machines,

there are not both transitions $\delta(q, \sigma, \gamma)$ and $\delta(q, \varepsilon, \gamma)$. Note that the machine cannot make an $\varepsilon$-transition on empty stack. We will need this information later.

A $k$-pass automaton $M$ *accepts* a word $w \in \Sigma$ if,
when started in its initial state with an empty stack and with $w\sharp$ written on the input tape,
the automaton halts in state $H_a$ at the end of the $k$-th pass. We write

$M \vdash w$ if $M$ accepts $w$. The *language accepted* by $M$ is

$$L(M) := \{w \in \Sigma^* : M \vdash w\} \subset \Sigma^*$$

A *multi-pass language*) is a language accepted by a $k$-pass automaton for some $k$. Let $\mathcal{M}$ denote the class of all multi-pass languages.

A $k$-pass automaton $M$ *accepts* a word $w \in \Sigma$ if,
when started in its initial state with an empty stack and with $w\sharp$ written on the input tape,
the automaton halts in state $H_a$ at the end of the $k$-th pass. We write $M \vdash w$ if $M$ accepts $w$. The *language accepted* by $M$ is

$$L(M) := \{w \in \Sigma^* : M \vdash w\} \subset \Sigma^*$$

A *multi-pass language*) is a language accepted by a $k$-pass automaton for some $k$. Let $\mathcal{M}$ denote the class of all multi-pass languages.

A $k$-pass automaton $M$ *accepts* a word $w \in \Sigma$ if,
when started in its initial state with an empty stack and with $w\sharp$ written on the input tape,
the automaton halts in state $H_a$ at the end of the $k$-th pass. We write

$M \vdash w$ if $M$ accepts $w$. The *language accepted* by $M$ is

$$L(M) := \{w \in \Sigma^* : M \vdash w\} \subset \Sigma^*$$

A *multi-pass language*) is a language accepted by a $k$-pass automaton for some $k$. Let $\mathcal{M}$ denote the class of all multi-pass languages.

A $k$-pass automaton $M$ *accepts* a word $w \in \Sigma$ if,
when started in its initial state with an empty stack and with $w\sharp$ written on the input tape,
the automaton halts in state $H_a$ at the end of the $k$-th pass. We write

$M \vdash w$ if $M$ accepts $w$. The *language accepted* by $M$ is

$$L(M) := \{w \in \Sigma^* : M \vdash w\} \subset \Sigma^*$$

A *multi-pass language*) is a language accepted by a $k$-pass automaton for some $k$. Let $\mathcal{M}$ denote the class of all multi-pass languages.

# Closure under Inverse Homomorphism

The basic question about a class of formal languages is:
What closure properties does the class have?
So we need to investigate this question for the class $\mathcal{M}$ of multi-pass languages.

If $Z$ and $\Sigma$ are finite alphabets, a homomorphism

$$\phi : Z^* \to \Sigma^*$$

is defined by its images $\phi(\zeta_i) = u_i$.

Observation. The class $\mathcal{M}$ is closed under inverse homomorphism.
That is, if $\phi : Z^* \to \Sigma^*$ is a homomorphism and $L \subseteq \Sigma^*$ is multi-pass then $K = \{w \in Z^*, \phi(w) \in L\}$ is multi-pass.

# Closure under Inverse Homomorphism

The basic question about a class of formal languages is:
What closure properties does the class have?
So we need to investigate this question for the class $\mathcal{M}$ of multi-pass languages.

If $Z$ and $\Sigma$ are finite alphabets, a homomorphism

$$\phi : Z^* \to \Sigma^*$$

is defined by its images $\phi(\zeta_i) = u_i$.

Observation. The class $\mathcal{M}$ is closed under inverse homomorphism.
That is, if $\phi : Z^* \to \Sigma^*$ is a homomorphism and $L \subseteq \Sigma^*$ is multi-pass then $K = \{w \in Z^*, \phi(w) \in L\}$ is multi-pass.

# Closure under Inverse Homomorphism

The basic question about a class of formal languages is:
What closure properties does the class have?
So we need to investigate this question for the class $\mathcal{M}$ of multi-pass languages.

If $Z$ and $\Sigma$ are finite alphabets, a homomorphism

$$\phi : Z^* \rightarrow \Sigma^*$$

is defined by its images $\phi(\zeta_i) = u_i$.

Observation. The class $\mathcal{M}$ is closed under inverse homomorphism.
That is, if $\phi : Z^* \rightarrow \Sigma^*$ is a homomorphism and $L \subseteq \Sigma^*$ is multi-pass then $K = \{w \in Z^*, \phi(w) \in L\}$ is multi-pass.

# Closure under Inverse Homomorphism

The basic question about a class of formal languages is:
What closure properties does the class have?
So we need to investigate this question for the class $\mathcal{M}$ of multi-pass languages.

If $Z$ and $\Sigma$ are finite alphabets, a homomorphism

$$\phi : Z^* \to \Sigma^*$$

is defined by its images $\phi(\zeta_i) = u_i$.

Observation. The class $\mathcal{M}$ is closed under inverse homomorphism.
That is, if $\phi : Z^* \to \Sigma^*$ is a homomorphism and $L \subseteq \Sigma^*$ is multi-pass then $K = \{w \in Z^*, \phi(w) \in L\}$ is multi-pass.

Proof. Let $M$ accept $L$. Consider the multi-pass automaton $\widehat{M}$ over $Z$ which
on reading a letter $\zeta \in Z$ simulates $M$ on reading $\phi(\zeta)$.

Closure under inverse homorphism is the basic property needed to consider group word problems.

Proof. Let $M$ accept $L$. Consider the multi-pass automaton $\widehat{M}$ over $Z$ which
on reading a letter $\zeta \in Z$ simulates $M$ on reading $\phi(\zeta)$.

Closure under inverse homorphism is the basic property needed to consider group word problems.

Proof. Let $M$ accept $L$. Consider the multi-pass automaton $\widehat{M}$ over $Z$ which
on reading a letter $\zeta \in Z$ simulates $M$ on reading $\phi(\zeta)$.

Closure under inverse homorphism is the basic property needed to consider group word problems.

Observation. Whether or not a finitely generated group $G$ has a multi-pass word problem is independent of presentation. If $G$ has multi-pass word problem then every finitely generated subgroup of $G$ also has multi-pass word problem. Proof. Let $G = \langle X; R \rangle$ be a finitely

generated presentation of $G$.
such that $WP(G)$ is a multi-pass language.
Let $H = \langle Y; S \rangle$ be a finitely generated group and suppose that there is an injective homomorphism $\phi : H \to G$.
Then $w \in WP(H)$ if and only if $\phi(w) \in WP(G)$

and thus $WP(H)$ is multi-pass.

Observation. The class of groups with multi-pass word problem is closed under extension by finite groups. Once one has closure under finitely generated subgroups the argument is the same as for context-free groups.

Observation. Whether or not a finitely generated group $G$ has a multi-pass word problem is independent of presentation. If $G$ has multi-pass word problem then every finitely generated subgroup of $G$ also has multi-pass word problem. Proof. Let $G = \langle X; R \rangle$ be a finitely

generated presentation of $G$.
such that $WP(G)$ is a multi-pass language.
Let $H = \langle Y; S \rangle$ be a finitely generated group and suppose that there is an injective homomorphism $\phi : H \to G$.
Then $w \in WP(H)$ if and only if $\phi(w) \in WP(G)$

and thus $WP(H)$ is multi-pass.

Observation. The class of groups with multi-pass word problem is closed under extension by finite groups. Once one has closure under finitely generated subgroups the argument is the same as for context-free groups.

Observation. Whether or not a finitely generated group $G$ has a multi-pass word problem is independent of presentation. If $G$ has multi-pass word problem then every finitely generated subgroup of $G$ also has multi-pass word problem. Proof. Let $G = \langle X; R \rangle$ be a finitely

generated presentation of $G$.
such that $WP(G)$ is a multi-pass language.
Let $H = \langle Y; S \rangle$ be a finitely generated group and suppose that there is an injective homomorphism $\phi : H \to G$.
Then $w \in WP(H)$ if and only if $\phi(w) \in WP(G)$

and thus $WP(H)$ is multi-pass.

Observation. The class of groups with multi-pass word problem is closed under extension by finite groups. Once one has closure under finitely generated subgroups the argument is the same as for context-free groups.

Observation. Whether or not a finitely generated group $G$ has a multi-pass word problem is independent of presentation. If $G$ has multi-pass word problem then every finitely generated subgroup of $G$ also has multi-pass word problem. Proof. Let $G = \langle X; R \rangle$ be a finitely

generated presentation of $G$.
such that $WP(G)$ is a multi-pass language.
Let $H = \langle Y; S \rangle$ be a finitely generated group and suppose that there is an injective homomorphism $\phi : H \to G$.
Then $w \in WP(H)$ if and only if $\phi(w) \in WP(G)$

and thus $WP(H)$ is multi-pass.

Observation. The class of groups with multi-pass word problem is closed under extension by finite groups. Once one has closure under finitely generated subgroups the argument is the same as for context-free groups.

Observation. Whether or not a finitely generated group $G$ has a multi-pass word problem is independent of presentation. If $G$ has multi-pass word problem then every finitely generated subgroup of $G$ also has multi-pass word problem. Proof. Let $G = \langle X; R \rangle$ be a finitely

generated presentation of $G$.
such that $WP(G)$ is a multi-pass language.
Let $H = \langle Y; S \rangle$ be a finitely generated group and suppose that there is an injective homomorphism $\phi : H \to G$.
Then $w \in WP(H)$ if and only if $\phi(w) \in WP(G)$

and thus $WP(H)$ is multi-pass.

Observation. The class of groups with multi-pass word problem is closed under extension by finite groups. Once one has closure under finitely generated subgroups the argument is the same as for context-free groups.

Observation. Whether or not a finitely generated group $G$ has a multi-pass word problem is independent of presentation. If $G$ has multi-pass word problem then every finitely generated subgroup of $G$ also has multi-pass word problem. Proof. Let $G = \langle X; R \rangle$ be a finitely

generated presentation of $G$.
such that $WP(G)$ is a multi-pass language.
Let $H = \langle Y; S \rangle$ be a finitely generated group and suppose that there is an injective homomorphism $\phi : H \to G$.
Then $w \in WP(H)$ if and only if $\phi(w) \in WP(G)$

and thus $WP(H)$ is multi-pass.

Observation. The class of groups with multi-pass word problem is closed under extension by finite groups. Once one has closure under finitely generated subgroups the argument is the same as for context-free groups.

# Closure under Interleaved Products

## Definition

Let $\Sigma_1, \Sigma_2$ be two finite alphabets and let $L_i \subset \Sigma_i^*$ be multi-pass languages for $i = 1, 2$. Note that there is no hypothesis on how $\Sigma_1$ and $\Sigma_2$ overlap.

Let $\Sigma = \Sigma_1 \cup \Sigma_2$ and denote by $\pi_i : \Sigma^* : \Sigma_i^*$ the monoid homomophism defined by setting

$$\pi_i(a) = a \text{ if } a \in \Sigma_i \text{ and } \pi_i(a) = \varepsilon \text{ otherwise .}$$

We call the language

$$L = \{w \in \Sigma^* : \pi_i(w) \in L_i i = 1, 2\}$$

the *interleaved product* of the languages $L_1$ and $L_2$.

# Closure under Interleaved Products

## Definition

Let $\Sigma_1, \Sigma_2$ be two finite alphabets and let $L_i \subset \Sigma_i^*$ be multi-pass languages for $i = 1, 2$. Note that there is no hypothesis on how $\Sigma_1$ and $\Sigma_2$ overlap.
Let $\Sigma = \Sigma_1 \cup \Sigma_2$ and denote by $\pi_i : \Sigma^* : \Sigma_i^*$ the monoid homomophism defined by setting

$$\pi_i(a) = a \text{ if } a \in \Sigma_i \text{ and } \pi_i(a) = \varepsilon \text{ otherwise .}$$

We call the language

$$L = \{w \in \Sigma^* : \pi_i(w) \in L_i i = 1, 2\}$$

the *interleaved product* of the languages $L_1$ and $L_2$.

If the two alphabets are disjoint then $L$ is the shuffle product of $L_1$ and $L_2$.

If $L_1 = L_2$ then $L$ is the intersection of $L_1$ and $L_2$.
There does not seem to be a standard name if the overlap of the alphabets is arbitrary.

**Proposition**

*The interleaved product of multi-pass languages is again a multi-pass language.*

If the two alphabets are disjoint then $L$ is the shuffle product of $L_1$ and $L_2$.

If $L_1 = L_2$ then $L$ is the intersection of $L_1$ and $L_2$.
There does not seem to be a standard name if the overlap of the alphabets is arbitrary.

**Proposition**

*The interleaved product of multi-pass languages is again a multi-pass language.*

If the two alphabets are disjoint then $L$ is the shuffle product of $L_1$ and $L_2$.

If $L_1 = L_2$ then $L$ is the intersection of $L_1$ and $L_2$.
There does not seem to be a standard name if the overlap of the alphabets is arbitrary.

**Proposition**

*The interleaved product of multi-pass languages is again a multi-pass language.*

If the two alphabets are disjoint then $L$ is the shuffle product of $L_1$ and $L_2$.

If $L_1 = L_2$ then $L$ is the intersection of $L_1$ and $L_2$.
There does not seem to be a standard name if the overlap of the alphabets is arbitrary.

**Proposition**

*The interleaved product of multi-pass languages is again a multi-pass language.*

Let $L_i$ be accepted by a $k_i$-pass automaton $M_i$ and let $k = k_1 + k_2$. It is clear how to construct a $k$-pass automaton $\widehat{M}$ accepting the product of the $L_i$.
On the first $k_1$ passes $\widehat{M}$ simulates $M_1$ on the successive letters which are in $\Sigma_1$.

On reading the end-marker $\sharp$ at the end of pass $k_1$, the machine $\widehat{M}$ goes to different subsets of states depending on whether $M_1$ would halt and accept, or whether $M_1$ would reject.
In either case, the reading head is reset to the beginning of the input tape.
$\widehat{M}$ then begins simulating $M_2$ on the next $k_2$ passes on the letters belonging to $\Sigma_2$. On reading the end-marker at the end of pass $k_1 + k_2$,

if $M_2$ would accept and $M_1$ also accepted, then $\widehat{M}$ accepts.
If either would have rejected then $\widehat{M}$ rejects. Observation. The class of

multi-pass languages is closed under both intersection and union.

Let $L_i$ be accepted by a $k_i$-pass automaton $M_i$ and let $k = k_1 + k_2$.
It is clear how to construct a $k$-pass automaton $\widehat{M}$ accepting the product of the $L_i$.
On the first $k_1$ passes $\widehat{M}$ simulates $M_1$ on the successive letters which are in $\Sigma_1$.

On reading the end-marker $\sharp$ at the end of pass $k_1$, the machine $\widehat{M}$ goes to different subsets of states depending on whether $M_1$ would halt and accept, or whether $M_1$ would reject.
In either case, the reading head is reset to the beginning of the input tape.
$\widehat{M}$ then begins simulating $M_2$ on the next $k_2$ passes on the letters belonging to $\Sigma_2$. On reading the end-marker at the end of pass $k_1 + k_2$,

if $M_2$ would accept and $M_1$ also accepted, then $\widehat{M}$ accepts.
If either would have rejected then $\widehat{M}$ rejects. Observation. The class of

multi-pass languages is closed under both intersection and union.

Let $L_i$ be accepted by a $k_i$-pass automaton $M_i$ and let $k = k_1 + k_2$.
It is clear how to construct a $k$-pass automaton $\widehat{M}$ accepting the product of the $L_i$.
On the first $k_1$ passes $\widehat{M}$ simulates $M_1$ on the successive letters which are in $\Sigma_1$.

On reading the end-marker $\sharp$ at the end of pass $k_1$, the machine $\widehat{M}$ goes to different subsets of states depending on whether $M_1$ would halt and accept, or whether $M_1$ would reject.
In either case, the reading head is reset to the beginning of the input tape.
$\widehat{M}$ then begins simulating $M_2$ on the next $k_2$ passes on the letters belonging to $\Sigma_2$. On reading the end-marker at the end of pass $k_1 + k_2$,

if $M_2$ would accept and $M_1$ also accepted, then $\widehat{M}$ accepts.
If either would have rejected then $\widehat{M}$ rejects. Observation. The class of

multi-pass languages is closed under both intersection and union.

Let $L_i$ be accepted by a $k_i$-pass automaton $M_i$ and let $k = k_1 + k_2$. It is clear how to construct a $k$-pass automaton $\widehat{M}$ accepting the product of the $L_i$.
On the first $k_1$ passes $\widehat{M}$ simulates $M_1$ on the successive letters which are in $\Sigma_1$.

On reading the end-marker $\sharp$ at the end of pass $k_1$, the machine $\widehat{M}$ goes to different subsets of states depending on whether $M_1$ would halt and accept, or whether $M_1$ would reject.
In either case, the reading head is reset to the beginning of the input tape.
$\widehat{M}$ then begins simulating $M_2$ on the next $k_2$ passes on the letters belonging to $\Sigma_2$. On reading the end-marker at the end of pass $k_1 + k_2$,

if $M_2$ would accept and $M_1$ also accepted, then $\widehat{M}$ accepts.
If either would have rejected then $\widehat{M}$ rejects. Observation. The class of

multi-pass languages is closed under both intersection and union.

# Closure under direct products

Corollary. If the finitely generated groups $G_1$ and $G_2$ have multi-pass word problems,
then their direct product has a multi-pass word problem. All finitely

generated virtually free groups are multi-pass since they have deterministic context-free word problems. Thus $F_2 \times F_2$ is multi-pass.

Stallings' example of a finitely generated subgroup of $F_2 \times F_2$ which is not finitely presented is the kernel of the homomorphism

$$F_2 \times F_2 \to \langle t \rangle$$

defined by $a, b, c, d$ all go to $t$. So $\mathcal{M}$ contains groups which are not

finitely presented. Mikhailova's theorem shows that $F_2 \times F_2$ has

unsolvable membership problem.
So we begin to see unsolvable problems.

# Closure under direct products

Corollary. If the finitely generated groups $G_1$ and $G_2$ have multi-pass word problems,
then their direct product has a multi-pass word problem. All finitely

generated virtually free groups are multi-pass since they have deterministic context-free word problems. Thus $F_2 \times F_2$ is multi-pass.

Stallings' example of a finitely generated subgroup of $F_2 \times F_2$ which is not finitely presented is the kernel of the homomorphism

$$F_2 \times F_2 \to \langle t \rangle$$

defined by $a, b, c, d$ all go to $t$. So $\mathcal{M}$ contains groups which are not

finitely presented. Mikhailova's theorem shows that $F_2 \times F_2$ has

unsolvable membership problem.
So we begin to see unsolvable problems.

# Closure under direct products

Corollary. If the finitely generated groups $G_1$ and $G_2$ have multi-pass word problems,
then their direct product has a multi-pass word problem. All finitely

generated virtually free groups are multi-pass since they have deterministic context-free word problems. Thus $F_2 \times F_2$ is multi-pass.

Stallings' example of a finitely generated subgroup of $F_2 \times F_2$ which is not finitely presented is the kernel of the homomorphism

$$F_2 \times F_2 \to \langle t \rangle$$

defined by $a, b, c, d$ all go to $t$. So $\mathcal{M}$ contains groups which are not

finitely presented. Mikhailova's theorem shows that $F_2 \times F_2$ has

unsolvable membership problem.
So we begin to see unsolvable problems.

# Closure under direct products

Corollary. If the finitely generated groups $G_1$ and $G_2$ have multi-pass word problems,
then their direct product has a multi-pass word problem. All finitely

generated virtually free groups are multi-pass since they have deterministic context-free word problems. Thus $F_2 \times F_2$ is multi-pass.

Stallings' example of a finitely generated subgroup of $F_2 \times F_2$ which is not finitely presented is the kernel of the homomorphism

$$F_2 \times F_2 \to \langle t \rangle$$

defined by $a, b, c, d$ all go to $t$. So $\mathcal{M}$ contains groups which are not

finitely presented. Mikhailova's theorem shows that $F_2 \times F_2$ has

unsolvable membership problem.
So we begin to see unsolvable problems.

# Closure under direct products

Corollary. If the finitely generated groups $G_1$ and $G_2$ have multi-pass word problems,
then their direct product has a multi-pass word problem. All finitely

generated virtually free groups are multi-pass since they have deterministic context-free word problems. Thus $F_2 \times F_2$ is multi-pass.

Stallings' example of a finitely generated subgroup of $F_2 \times F_2$ which is not finitely presented is the kernel of the homomorphism

$$F_2 \times F_2 \to \langle t \rangle$$

defined by $a, b, c, d$ all go to $t$. So $\mathcal{M}$ contains groups which are not

finitely presented. Mikhailova's theorem shows that $F_2 \times F_2$ has

unsolvable membership problem.
So we begin to see unsolvable problems.

# Closure under direct products

Corollary. If the finitely generated groups $G_1$ and $G_2$ have multi-pass word problems,
then their direct product has a multi-pass word problem. All finitely

generated virtually free groups are multi-pass since they have deterministic context-free word problems. Thus $F_2 \times F_2$ is multi-pass.

Stallings' example of a finitely generated subgroup of $F_2 \times F_2$ which is not finitely presented is the kernel of the homomorphism

$$F_2 \times F_2 \to \langle t \rangle$$

defined by $a, b, c, d$ all go to $t$. So $\mathcal{M}$ contains groups which are not

finitely presented. Mikhailova's theorem shows that $F_2 \times F_2$ has

unsolvable membership problem.
So we begin to see unsolvable problems.

# Semi-direct Products

A very similar argument shows that if $G_1$ and $G_2$ are multi-pass and $G_2$ acts on $G_1$ by a *finite* group of automorphisms, then the corresponding semi-direct product is multi-pass. As before, check that the product of the letters representing elements of $G_2$ is the identity.

Using the state set, we can remember the multiplication table of the finite group of automorphisms and the image of each generator of $G_1$ under a given automorphism.

Now on reading a generator $x$ of $G_1$, simulate reading the image of $x$ under the automorphism associated to the product of the generators of $G_2$ read so far.

On reading a generator of $G_2$ update the automorphism. In particular, if $F = \langle x_1, ..., x_n \rangle$ is free and $\phi$ is an automorphism of $F$ of finite order then the mapping torus

$$\langle F, t : tx_it^{-1} = \phi(x_i) \rangle$$

# Semi-direct Products

A very similar argument shows that if $G_1$ and $G_2$ are multi-pass and $G_2$ acts on $G_1$ by a *finite* group of automorphisms, then the corresponding semi-direct product is multi-pass. As before, check that the product of the letters representing elements of $G_2$ is the identity.

Using the state set, we can remember the multiplication table of the finite group of automorphisms and the image of each generator of $G_1$ under a given automorphism.

Now on reading a generator $x$ of $G_1$, simulate reading the image of $x$ under the automorphism associated to the product of the generators of $G_2$ read so far.

On reading a generator of $G_2$ update the automorphism. In particular, if $F = \langle x_1, ..., x_n \rangle$ is free and $\phi$ is an automorphism of $F$ of finite order then the mapping torus

$$\langle F, t : tx_it^{-1} = \phi(x_i) \rangle$$

# Semi-direct Products

A very similar argument shows that if $G_1$ and $G_2$ are multi-pass and $G_2$ acts on $G_1$ by a *finite* group of automorphisms, then the corresponding semi-direct product is multi-pass. As before, check that the product of the letters representing elements of $G_2$ is the identity.

Using the state set, we can remember the multiplication table of the finite group of automorphisms and the image of each generator of $G_1$ under a given automorphism.

Now on reading a generator $x$ of $G_1$, simulate reading the image of $x$ under the automorphism associated to the product of the generators of $G_2$ read so far.

On reading a generator of $G_2$ update the automorphism. In particular, if $F = \langle x_1, ..., x_n \rangle$ is free and $\phi$ is an automorphism of $F$ of finite order then the mapping torus

$$\langle F, t : tx_it^{-1} = \phi(x_i) \rangle$$

# Semi-direct Products

A very similar argument shows that if $G_1$ and $G_2$ are multi-pass and $G_2$ acts on $G_1$ by a *finite* group of automorphisms, then the corresponding semi-direct product is multi-pass. As before, check that the product of the letters representing elements of $G_2$ is the identity.

Using the state set, we can remember the multiplication table of the finite group of automorphisms and the image of each generator of $G_1$ under a given automorphism.

Now on reading a generator $x$ of $G_1$, simulate reading the image of $x$ under the automorphism associated to the product of the generators of $G_2$ read so far.

On reading a generator of $G_2$ update the automorphism. In particular, if $F = \langle x_1, ..., x_n \rangle$ is free and $\phi$ is an automorphism of $F$ of finite order then the mapping torus

$$\langle F, t : tx_it^{-1} = \phi(x_i) \rangle$$

# Rewriting one-relator groups and mapping tori

The standard way to study a one-relator group is to rewrite the group as an HNN-extension of a one-relator group with shorter defining relator. This may require adding a root of a generator if no generator has exponent sum 0.

Observation. The one-relator groups

$$G_{m,n} = \langle xy^m xy^n \rangle, m, n \in \mathbb{Z}$$

are multi-pass.

Consider the group $\langle x, y; xy^{-2}xy^{-5} \rangle$.
So $\sigma_x = 2, \sigma_y = -7$. Add a square root to $y$.
Thus subsititue $x \to xy^7, y \to y^2$, giving

$$xy^7y^{-4}xy^7y^{-10} = xy^3xy^{-3}.$$

# Rewriting one-relator groups and mapping tori

The standard way to study a one-relator group is to rewrite the group as an HNN-extension of a one-relator group with shorter defining relator. This may require adding a root of a generator if no generator has exponent sum 0.

Observation. The one-relator groups

$$G_{m,n} = \langle xy^m xy^n \rangle, m, n \in \mathbb{Z}$$

are multi-pass.

Consider the group $\langle x, y; xy^{-2}xy^{-5} \rangle$.
So $\sigma_x = 2, \sigma_y = -7$. Add a square root to $y$.
Thus subsititue $x \to xy^7, y \to y^2$, giving

$$xy^7y^{-4}xy^7y^{-10} = xy^3xy^{-3}.$$

# Rewriting one-relator groups and mapping tori

The standard way to study a one-relator group is to rewrite the group as an HNN-extension of a one-relator group with shorter defining relator. This may require adding a root of a generator if no generator has exponent sum 0.

Observation. The one-relator groups

$$G_{m,n} = \langle xy^m xy^n \rangle, m, n \in \mathbb{Z}$$

are multi-pass.

Consider the group $\langle x, y; xy^{-2}xy^{-5} \rangle$.
So $\sigma_x = 2, \sigma_y = -7$. Add a square root to $y$.
Thus subsititue $x \to xy^7, y \to y^2$, giving

$$xy^7 y^{-4} xy^7 y^{-10} = xy^3 xy^{-3}.$$

We rewrite by subscripting occurrences of x by the exponent sum on $y$.
preceeding the occurrence, giving the relator $x_0 x_3$ in the base.
We can, of course, eliminate $x_3$ by a Tietze transformation. Giving

$$G = \langle x_0, x_1, x_2, y; y x_0 y^{-1} = x_1, y x_1 y^{-1} = x_2, y x_2 y^{-1} = {x_0}^{-1} \rangle$$

So $G$ is the mapping torus of the indicated automorphism.

We rewrite by subscripting occurrences of x by the exponent sum on $y$. preceeding the occurrence, giving the relator $x_0x_3$ in the base.
We can, of course, eliminate $x_3$ by a Tietze transformation. Giving

$$G = \langle x_0, x_1, x_2, y; yx_0y^{-1} = x_1, yx_1y^{-1} = x_2, yx_2y^{-1} = {x_0}^{-1} \rangle$$

So $G$ is the mapping torus of the indicated automorphism.

# How often does rewriting a two-generator one-relator group yield a mapping torus?

Obtaining a mapping torus is *not* a generic property. This is proved by Nathan Dunfield and Dylan Thurston in *A random tunnnel-number one 3-manifold does not fiber over the circle*. Computer experiments show that the fraction of two-generator one-relator groups which rewrite to mapping tori is between .90 and .92.

# How often does rewriting a two-generator one-relator group yield a mapping torus?

Obtaining a mapping torus is *not* a generic property. This is proved by Nathan Dunfield and Dylan Thurston in *A random tunnnel-number one 3-manifold does not fiber over the circle*. Computer experiments show that the fraction of two-generator one-relator groups which rewrite to mapping tori is between .90 and .92.

# How often does rewriting a two-generator one-relator group yield a mapping torus?

Obtaining a mapping torus is *not* a generic property. This is proved by Nathan Dunfield and Dylan Thurston in *A random tunnnel-number one* 3*-manifold does not fiber over the circle*. Computer experiments show that the fraction of two-generator one-relator groups which rewrite to mapping tori is between .90 and .92.

# Basic groups

Definition. A *basic* group is a group which is the free product of finitely many finite groups and a finitely generated free group.

The *canonical presentation* of a basic group is to take the multiplication table presentations for the finite factors and the free presentation for the free factor. In the canonical presentation, every element has a unique representation as a reduced word- no two successive letters come from the same finite factor and the word is reduced on the free generators.

A context-free language is *semi-simple* if it is accepted by a single state deterministic pushdown automata which accepts by empty stack and is allowed to continue working on empty stack.

# Basic groups

Definition. A *basic* group is a group which is the free product of finitely many finite groups and a finitely generated free group.

The *canonical presentation* of a basic group is to take the multiplication table presentations for the finite factors and the free presentation for the free factor. In the canonical presentation, every element has a unique representation as a reduced word- no two successive letters come from the same finite factor and the word is reduced on the free generators.

A context-free language is *semi-simple* if it is accepted by a single state deterministic pushdown automata which accepts by empty stack and is allowed to continue working on empty stack.

# Basic groups

Definition. A *basic* group is a group which is the free product of finitely many finite groups and a finitely generated free group.

The *canonical presentation* of a basic group is to take the multiplication table presentations for the finite factors and the free presentation for the free factor. In the canonical presentation, every element has a unique representation as a reduced word- no two successive letters come from the same finite factor and the word is reduced on the free generators.

A context-free language is *semi-simple* if it is accepted by a single state deterministic pushdown automata which accepts by empty stack and is allowed to continue working on empty stack.

# Basic groups

Definition. A *basic* group is a group which is the free product of finitely many finite groups and a finitely generated free group.

The *canonical presentation* of a basic group is to take the multiplication table presentations for the finite factors and the free presentation for the free factor. In the canonical presentation, every element has a unique representation as a reduced word- no two successive letters come from the same finite factor and the word is reduced on the free generators.

A context-free language is *semi-simple* if it is accepted by a single state deterministic pushdown automata which accepts by empty stack and is allowed to continue working on empty stack.

# Basic groups

Definition. A *basic* group is a group which is the free product of finitely many finite groups and a finitely generated free group.

The *canonical presentation* of a basic group is to take the multiplication table presentations for the finite factors and the free presentation for the free factor. In the canonical presentation, every element has a unique representation as a reduced word- no two successive letters come from the same finite factor and the word is reduced on the free generators.

A context-free language is *semi-simple* if it is accepted by a single state deterministic pushdown automata which accepts by empty stack and is allowed to continue working on empty stack.

Theorem. (Haring-Smith) The following are equivalent for a finitely generated group $G$.

1. $G$ is basic.
2. $G$ has a presentation such that $WP(G)$ is semi-simple.
3. $G$ has a presentation $\Pi$ such that in the Cayley graph $\Gamma(\Pi)$), there are only finitely many simple closed paths through a vertex.

Example. Consider the modular group $G = \langle x; x^2 \rangle * \langle b; b^3 \rangle$.

Shapiro's Question. Suppose that a finitely generated group $G$ has a presentation $\Pi$ such that in the Cayley graph $\Gamma(\Pi)$) geodesics are unique. What can one say about $G$?

Conjecture. $G$ is basic.

Theorem. (Haring-Smith) The following are equivalent for a finitely generated group $G$.

1. $G$ is basic.
2. $G$ has a presentation such that $WP(G)$ is semi-simple.
3. $G$ has a presentation $\Pi$ such that in the Cayley graph $\Gamma(\Pi)$), there are only finitely many simple closed paths through a vertex.

Example. Consider the modular group $G = \langle x; x^2 \rangle * \langle b; b^3 \rangle$.

Shapiro's Question. Suppose that a finitely generated group $G$ has a presentation $\Pi$ such that in the Cayley graph $\Gamma(\Pi)$) geodesics are unique. What can one say about $G$?

Conjecture. $G$ is basic.

Theorem. (Haring-Smith) The following are equivalent for a finitely generated group $G$.

1. $G$ is basic.
2. $G$ has a presentation such that $WP(G)$ is semi-simple.
3. $G$ has a presentation $\Pi$ such that in the Cayley graph $\Gamma(\Pi))$, there are only finitely many simple closed paths through a vertex.

Example. Consider the modular group $G = \langle x; x^2 \rangle * \langle b; b^3 \rangle$.

Shapiro's Question. Suppose that a finitely generated group $G$ has a presentation $\Pi$ such that in the Cayley graph $\Gamma(\Pi))$ geodesics are unique. What can one say about $G$?

Conjecture. $G$ is basic.

Theorem. (Haring-Smith) The following are equivalent for a finitely generated group $G$.

1. $G$ is basic.
2. $G$ has a presentation such that $WP(G)$ is semi-simple.
3. $G$ has a presentation $\Pi$ such that in the Cayley graph $\Gamma(\Pi))$, there are only finitely many simple closed paths through a vertex.

Example. Consider the modular group $G = \langle x; x^2 \rangle * \langle b; b^3 \rangle$.

Shapiro's Question. Suppose that a finitely generated group $G$ has a presentation $\Pi$ such that in the Cayley graph $\Gamma(\Pi))$ geodesics are unique. What can one say about $G$?

Conjecture. $G$ is basic.

Theorem. (Haring-Smith) The following are equivalent for a finitely generated group $G$.

1. $G$ is basic.
2. $G$ has a presentation such that $WP(G)$ is semi-simple.
3. $G$ has a presentation $\Pi$ such that in the Cayley graph $\Gamma(\Pi))$, there are only finitely many simple closed paths through a vertex.

Example. Consider the modular group $G = \langle x; x^2 \rangle * \langle b; b^3 \rangle$.

Shapiro's Question. Suppose that a finitely generated group $G$ has a presentation $\Pi$ such that in the Cayley graph $\Gamma(\Pi))$ geodesics are unique. What can one say about $G$?

Conjecture. $G$ is basic.

# Closure under complementation

Observation. The class of multi-pass languages is closed under complementation
The idea is of course, interchange to which of the special states $H_a$ and $H_r$ the automaton goes at the end of the final pass.

The possible problem is that the automaton could go into a loop making $\varepsilon$-transitions without advancing the tape and thus never read the final end-marker.
Show that every automaton is equivalent to a *normalized* automaton which always reads to the end-marker on the last pass. The proof is

exactly the same as the proof for deterministic pushdown automata as given in Hopcroft and Ullman.

# Closure under complementation

Observation. The class of multi-pass languages is closed under complementation

The idea is of course, interchange to which of the special states $H_a$ and $H_r$ the automaton goes at the end of the final pass.

The possible problem is that the automaton could go into a loop making $\varepsilon$-transitions without advancing the tape and thus never read the final end-marker.

Show that every automaton is equivalent to a *normalized* automaton which always reads to the end-marker on the last pass. The proof is

exactly the same as the proof for deterministic pushdown automata as given in Hopcroft and Ullman.

# Closure under complementation

Observation. The class of multi-pass languages is closed under complementation
The idea is of course, interchange to which of the special states $H_a$ and $H_r$ the automaton goes at the end of the final pass.

The possible problem is that the automaton could go into a loop making $\varepsilon$-transitions without advancing the tape and thus never read the final end-marker.
Show that every automaton is equivalent to a *normalized* automaton which always reads to the end-marker on the last pass. The proof is

exactly the same as the proof for deterministic pushdown automata as given in Hopcroft and Ullman.

# Closure under complementation

Observation. The class of multi-pass languages is closed under complementation
The idea is of course, interchange to which of the special states $H_a$ and $H_r$ the automaton goes at the end of the final pass.

The possible problem is that the automaton could go into a loop making $\varepsilon$-transitions without advancing the tape and thus never read the final end-marker.
Show that every automaton is equivalent to a *normalized* automaton which always reads to the end-marker on the last pass. The proof is exactly the same as the proof for deterministic pushdown automata as given in Hopcroft and Ullman.

# Closure under complementation

Observation. The class of multi-pass languages is closed under complementation
The idea is of course, interchange to which of the special states $H_a$ and $H_r$ the automaton goes at the end of the final pass.

The possible problem is that the automaton could go into a loop making $\varepsilon$-transitions without advancing the tape and thus never read the final end-marker.
Show that every automaton is equivalent to a *normalized* automaton which always reads to the end-marker on the last pass. The proof is

exactly the same as the proof for deterministic pushdown automata as given in Hopcroft and Ullman.

Observation. The membership problem for a multipass language is solvable in cubic time. (Undoubtedly in linear time.) Proof. Run the normalized automaton on the input.

Observation. However, the emptiness problem for multi-pass languages is undecidable. In formal language theory it is well-known

that deciding whether or not the intersection of two deterministic context-free languages is empty is undecidable. All such languages are multi-pass. One can represent valid computations of Turing

machines as the intersection of deterministic context-free languages.

Observation. The membership problem for a multipass language is solvable in cubic time. (Undoubtedly in linear time.) Proof. Run the normalized automaton on the input.

Observation. However, the emptiness problem for multi-pass languages is undecidable. In formal language theory it is well-known that deciding whether or not the intersection of two deterministic context-free languages is empty is undecidable. All such languages are multi-pass. One can represent valid computations of Turing machines as the intersection of deterministic context-free languages.

Observation. The membership problem for a multipass language is solvable in cubic time. (Undoubtedly in linear time.) Proof. Run the normalized automaton on the input.

Observation. However, the emptiness problem for multi-pass languages is undecidable. In formal language theory it is well-known that deciding whether or not the intersection of two deterministic context-free languages is empty is undecidable. All such languages are multi-pass. One can represent valid computations of Turing machines as the intersection of deterministic context-free languages.

Observation. The membership problem for a multipass language is solvable in cubic time. (Undoubtedly in linear time.) Proof. Run the normalized automaton on the input.

Observation. However, the emptiness problem for multi-pass languages is undecidable. In formal language theory it is well-known

that deciding whether or not the intersection of two deterministic context-free languages is empty is undecidable. All such languages are multi-pass. One can represent valid computations of Turing

machines as the intersection of deterministic context-free languages.

Of course, we have no good method for showing that a language is not multi-pass.

Conjecture. The free product $\mathbb{Z}^2 * \langle x; x^2 \rangle$ is not multi-pass.
This is probably on the borderline.

Of course, we have no good method for showing that a language is not multi-pass.

Conjecture. The free product $\mathbb{Z}^2 * \langle x; x^2 \rangle$ is not multi-pass.
This is probably on the borderline.

Of course, we have no good method for showing that a language is not multi-pass.

Conjecture. The free product $\mathbb{Z}^2 * \langle x; x^2 \rangle$ is not multi-pass.
This is probably on the borderline.

Thank You