

Reconfigurable Systems and Intractability

Gregory Puleo
gjp6297@rit.edu

Rochester Institute of Technology

July 29, 2008

Motivating Problem

Question

Given a reconfigurable system R , a set S of R -states, and an initial state \mathbf{u} , is there any way to get from \mathbf{u} to a state in S using legal generators of R ?

Motivating Problem

Question

Given a reconfigurable system R , a set S of R -states, and an initial state \mathbf{u} , is there any way to get from \mathbf{u} to a state in S using legal generators of R ? Is there a general algorithm for solving this problem?

Motivating Problem

Question

Given a reconfigurable system R , a set S of R -states, and an initial state \mathbf{u} , is there any way to get from \mathbf{u} to a state in S using legal generators of R ? Is there a general algorithm for solving this problem?

Technical Considerations

- The underlying graph \mathcal{G} , and the set Φ of generators, could be infinite. How do we represent them?

Motivating Problem

Question

Given a reconfigurable system R , a set S of R -states, and an initial state \mathbf{u} , is there any way to get from \mathbf{u} to a state in S using legal generators of R ? Is there a general algorithm for solving this problem?

Technical Considerations

- The underlying graph \mathcal{G} , and the set Φ of generators, could be infinite. How do we represent them?
- What model of computation are we using for our “general algorithm”?

Recursivity and Quasifiniteness

“Definition”

A set $S \subseteq \mathbb{N}$ is **recursive** if there is an algorithm which can decide, given any $n \in \mathbb{N}$, whether or not $n \in S$.

Recursivity and Quasifiniteness

“Definition”

A set $S \subseteq \mathbb{N}$ is **recursive** if there is an algorithm which can decide, given any $n \in \mathbb{N}$, whether or not $n \in S$. (More formally: if there is a Turing machine which halts on all inputs and accepts the set S .)

Recursivity and Quasifiniteness

“Definition”

A set $S \subseteq \mathbb{N}$ is **recursive** if there is an algorithm which can decide, given any $n \in \mathbb{N}$, whether or not $n \in S$. (More formally: if there is a Turing machine which halts on all inputs and accepts the set S .)

Definition

A reconfigurable system $R = (\mathcal{G}, \mathcal{A}, \Phi)$ is **quasifinite** if:

- \mathcal{A} is finite;
- $V(\text{SUP}\phi)$ is finite for all $\phi \in \Phi$;
- Φ is recursive; and
- $V(\mathcal{G})$ is finite or countable, and $E(\mathcal{G})$ is recursive.

Recursivity and Quasifiniteness

“Definition”

A set $S \subseteq \mathbb{N}$ is **recursive** if there is an algorithm which can decide, given any $n \in \mathbb{N}$, whether or not $n \in S$. (More formally: if there is a Turing machine which halts on all inputs and accepts the set S .)

Definition

A reconfigurable system $R = (\mathcal{G}, \mathcal{A}, \Phi)$ is **quasifinite** if:

- \mathcal{A} is finite;
- $V(\text{SUP}\phi)$ is finite for all $\phi \in \Phi$;
- Φ is recursive; and
- $V(\mathcal{G})$ is finite or countable, and $E(\mathcal{G})$ is recursive.

Definition

A function $f : A \rightarrow B$ is quasifinite if there is some $b_0 \in B$ such that $f^{-1}(B \setminus \{b_0\})$ is finite.

Refinement of the Problem

Decision Problem (CONFIGURABILITY)

INSTANCE. Quasifinite reconfigurable system R , finite set S of quasifinite R -states, quasifinite initial state \mathbf{u} .

QUESTION. Is it possible to move from \mathbf{u} to a state in S using the generators of R ?

Refinement of the Problem

Decision Problem (CONFIGURABILITY)

INSTANCE. Quasifinite reconfigurable system R , finite set S of quasifinite R -states, quasifinite initial state \mathbf{u} .

QUESTION. Is it possible to move from \mathbf{u} to a state in S using the generators of R ?

Theorem

CONFIGURABILITY is *undecidable*: there is no algorithm (Turing machine) which solves every instance of CONFIGURABILITY.

Refinement of the Problem

Decision Problem (CONFIGURABILITY)

INSTANCE. Quasifinite reconfigurable system R , finite set S of quasifinite R -states, quasifinite initial state \mathbf{u} .

QUESTION. Is it possible to move from \mathbf{u} to a state in S using the generators of R ?

Theorem

CONFIGURABILITY is *undecidable*: there is no algorithm (Turing machine) which solves every instance of CONFIGURABILITY.

Why?

Refinement of the Problem

Decision Problem (CONFIGURABILITY)

INSTANCE. Quasifinite reconfigurable system R , finite set S of quasifinite R -states, quasifinite initial state \mathbf{u} .

QUESTION. Is it possible to move from \mathbf{u} to a state in S using the generators of R ?

Theorem

CONFIGURABILITY is *undecidable*: there is no algorithm (Turing machine) which solves every instance of CONFIGURABILITY.

Why? Because the *word problem*, which is not solvable in general, reduces to CONFIGURABILITY.

Free Groups

Definition

Given a set X , let $X^{\pm 1}$ be the set of all elements of X together with a formal inverse \bar{x} for each $x \in X$. Let X^* be the set of all formal strings over $X^{\pm 1}$. For $w, z \in X^*$, say $w \sim z$ if w and z differ by finitely many insertions or deletions of inverse pairs $x\bar{x}$. Then $F_X = X^*/\sim$ is a group under concatenation, called the **free group on X** .

Intuitively, a free group is a group where the only relations between elements are the “trivial relations” like $x\bar{x} = e$.

Example

We may write F_n to indicate the free group on the set $\{1, \dots, n\}$. Then $\mathbb{Z} \cong F_1$.

An “Almost Free” Group

Example

Consider the group $\mathbb{Z} \times \mathbb{Z}$, and compare it to F_2 :

$\mathbb{Z} \times \mathbb{Z}$	F_2
Generated by $(0, 1)$ and $(1, 0)$ $(0, 1)$ and $(1, 0)$ commute	Generated by a and b
Satisfies no other nontrivial relations	Satisfies no nontrivial relations

An “Almost Free” Group

Example

Consider the group $\mathbb{Z} \times \mathbb{Z}$, and compare it to F_2 :

$\mathbb{Z} \times \mathbb{Z}$	F_2
Generated by $(0, 1)$ and $(1, 0)$ $(0, 1)$ and $(1, 0)$ commute	Generated by a and b
Satisfies no other nontrivial relations	Satisfies no nontrivial relations

Can we use this to get a succinct purely algebraic description of $\mathbb{Z} \times \mathbb{Z}$?

Finitely Presented Groups

Definition

Let X be a set, let $Y \subseteq X^*$ be a finite set of words over X , and let N be the normal closure of Y in F_X .

Finitely Presented Groups

Definition

Let X be a set, let $Y \subseteq X^*$ be a finite set of words over X , and let N be the normal closure of Y in F_X . Then F_X/N is said to be a **finitely presented group**, and the tuple $\langle X|Y \rangle$ is said to be a **presentation** for F_X/N . We write $F_X/N = \langle X|Y \rangle$.

Finitely Presented Groups

Definition

Let X be a set, let $Y \subseteq X^*$ be a finite set of words over X , and let N be the normal closure of Y in F_X . Then F_X/N is said to be a **finitely presented group**, and the tuple $\langle X|Y \rangle$ is said to be a **presentation** for F_X/N . We write $F_X/N = \langle X|Y \rangle$.

Informally, $\langle X|Y \rangle$ is the “most free” group which satisfies the trivial relations together with relations of the form $y = e$ for $y \in Y$.

Finitely Presented Groups

Definition

Let X be a set, let $Y \subseteq X^*$ be a finite set of words over X , and let N be the normal closure of Y in F_X . Then F_X/N is said to be a **finitely presented group**, and the tuple $\langle X|Y \rangle$ is said to be a **presentation** for F_X/N . We write $F_X/N = \langle X|Y \rangle$.

Informally, $\langle X|Y \rangle$ is the “most free” group which satisfies the trivial relations together with relations of the form $y = e$ for $y \in Y$.

Example

- $\mathbb{Z}_k \cong \langle a|a^k \rangle$.
- $\mathbb{Z} \times \mathbb{Z} \cong \langle a, b|ab\bar{a}\bar{b} \rangle$.

The Word Problem

Definition

Given a finitely presentation $\langle X|Y \rangle$ for some group G , the **word problem** for $\langle X|Y \rangle$ is the following decision problem:

The Word Problem

Definition

Given a finitely presentation $\langle X|Y \rangle$ for some group G , the **word problem** for $\langle X|Y \rangle$ is the following decision problem: Given a word $w \in X^*$, is w equivalent to the identity in G ?

The Word Problem

Definition

Given a finitely presentation $\langle X|Y \rangle$ for some group G , the **word problem** for $\langle X|Y \rangle$ is the following decision problem: Given a word $w \in X^*$, is w equivalent to the identity in G ?

Theorem (Novikov 1955, Boone 1959)

There exists a finitely presented group with unsolvable word problem.

Reconfigurable Systems and Word Problems

Theorem

$w \equiv e$ if and only if w can be written in the form

$$w = g_1 y_1 g_1^{-1} g_2 y_2 g_2^{-1} \cdots g_k y_k g_k^{-1}$$

with $g_i \in X^$ and $y_i \in Y$.*



Reconfigurable Systems and Word Problems

Theorem

$w \equiv e$ if and only if w can be written in the form

$$w = g_1 y_1 g_1^{-1} g_2 y_2 g_2^{-1} \cdots g_k y_k g_k^{-1}$$

with $g_i \in X^$ and $y_i \in Y$.*



Corollary

$w \equiv e$ if and only if w can reach the empty string by a finite chain of insertions/deletions of inverse pairs $a\bar{a}$ or elements of Y .



Reconfigurable Systems and Word Problems

Theorem

$w \equiv e$ if and only if w can be written in the form

$$w = g_1 y_1 g_1^{-1} g_2 y_2 g_2^{-1} \cdots g_k y_k g_k^{-1}$$

with $g_i \in X^*$ and $y_i \in Y$. □

Corollary

$w \equiv e$ if and only if w can reach the empty string by a finite chain of insertions/deletions of inverse pairs $a\bar{a}$ or elements of Y . □

In other words, “being the identity” is expressible in terms of chains of **local, reversible** operations.

Reconfigurable Systems and Word Problems

Theorem

$w \equiv e$ if and only if w can be written in the form

$$w = g_1 y_1 g_1^{-1} g_2 y_2 g_2^{-1} \cdots g_k y_k g_k^{-1}$$

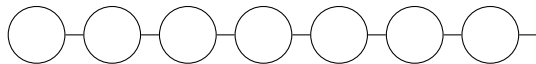
with $g_i \in X^*$ and $y_i \in Y$. □

Corollary

$w \equiv e$ if and only if w can reach the empty string by a finite chain of insertions/deletions of inverse pairs $a\bar{a}$ or elements of Y . □

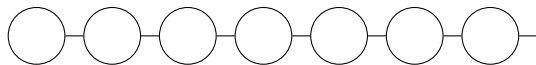
In other words, “being the identity” is expressible in terms of chains of **local, reversible** operations. These operations have a natural expression as the generators of a reconfigurable system.

The Construction



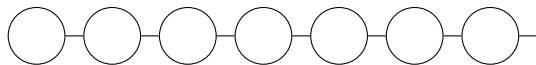
- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.

The Construction



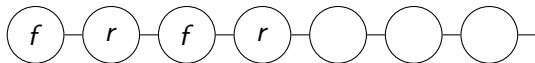
- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.
- $V(\mathcal{G}) = \mathbb{N}$, $E(\mathcal{G}) = \{n, n+1 \mid n \in \mathbb{N}\}$.

The Construction



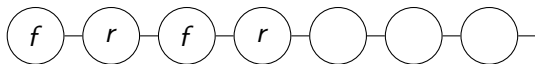
- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.
- $V(\mathcal{G}) = \mathbb{N}$, $E(\mathcal{G}) = \{n, n+1 \mid n \in \mathbb{N}\}$.
- $\mathcal{A} = X^{\pm 1} \cup \{\sqcup\}$.

The Construction



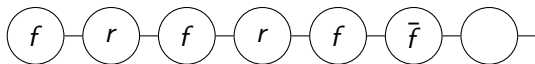
- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.
- $V(\mathcal{G}) = \mathbb{N}$, $E(\mathcal{G}) = \{n, n+1 \mid n \in \mathbb{N}\}$.
- $\mathcal{A} = X^{\pm 1} \cup \{\sqcup\}$.
- \mathbf{u} has the word w written across the leftmost vertices.

The Construction



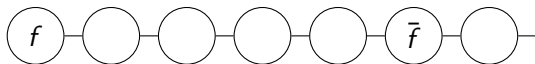
- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.
- $V(\mathcal{G}) = \mathbb{N}$, $E(\mathcal{G}) = \{n, n+1 \mid n \in \mathbb{N}\}$.
- $\mathcal{A} = X^{\pm 1} \cup \{\sqcup\}$.
- \mathbf{u} has the word w written across the leftmost vertices.
- For each inverse pair $x\bar{x}$ and each relator in Y , we have a generator that lets us insert or remove that string.

The Construction



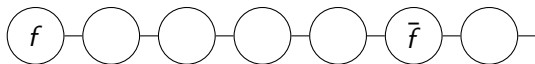
- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.
- $V(\mathcal{G}) = \mathbb{N}$, $E(\mathcal{G}) = \{n, n+1 \mid n \in \mathbb{N}\}$.
- $\mathcal{A} = X^{\pm 1} \cup \{\sqcup\}$.
- \mathbf{u} has the word w written across the leftmost vertices.
- For each inverse pair $x\bar{x}$ and each relator in Y , we have a generator that lets us insert or remove that string.

The Construction



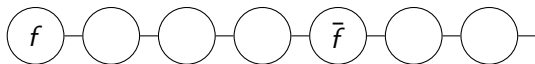
- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.
- $V(\mathcal{G}) = \mathbb{N}$, $E(\mathcal{G}) = \{n, n+1 \mid n \in \mathbb{N}\}$.
- $\mathcal{A} = X^{\pm 1} \cup \{\sqcup\}$.
- \mathbf{u} has the word w written across the leftmost vertices.
- For each inverse pair $x\bar{x}$ and each relator in Y , we have a generator that lets us insert or remove that string.

The Construction



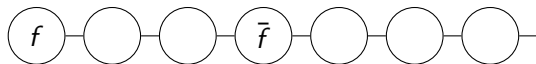
- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.
- $V(\mathcal{G}) = \mathbb{N}$, $E(\mathcal{G}) = \{n, n+1 \mid n \in \mathbb{N}\}$.
- $\mathcal{A} = X^{\pm 1} \cup \{\sqcup\}$.
- \mathbf{u} has the word w written across the leftmost vertices.
- For each inverse pair $x\bar{x}$ and each relator in Y , we have a generator that lets us insert or remove that string.
- We also have generators that let us switch any letter with an adjacent blank.

The Construction



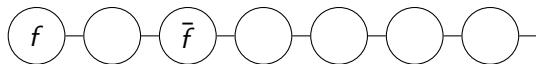
- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.
- $V(\mathcal{G}) = \mathbb{N}$, $E(\mathcal{G}) = \{n, n+1 \mid n \in \mathbb{N}\}$.
- $\mathcal{A} = X^{\pm 1} \cup \{\sqcup\}$.
- \mathbf{u} has the word w written across the leftmost vertices.
- For each inverse pair $x\bar{x}$ and each relator in Y , we have a generator that lets us insert or remove that string.
- We also have generators that let us switch any letter with an adjacent blank.

The Construction



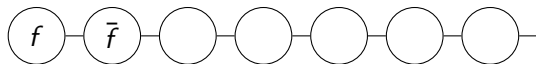
- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.
- $V(\mathcal{G}) = \mathbb{N}$, $E(\mathcal{G}) = \{n, n+1 \mid n \in \mathbb{N}\}$.
- $\mathcal{A} = X^{\pm 1} \cup \{\sqcup\}$.
- \mathbf{u} has the word w written across the leftmost vertices.
- For each inverse pair $x\bar{x}$ and each relator in Y , we have a generator that lets us insert or remove that string.
- We also have generators that let us switch any letter with an adjacent blank.

The Construction



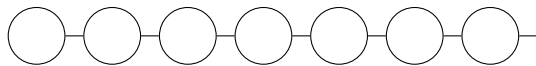
- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.
- $V(\mathcal{G}) = \mathbb{N}$, $E(\mathcal{G}) = \{n, n+1 \mid n \in \mathbb{N}\}$.
- $\mathcal{A} = X^{\pm 1} \cup \{\sqcup\}$.
- \mathbf{u} has the word w written across the leftmost vertices.
- For each inverse pair $x\bar{x}$ and each relator in Y , we have a generator that lets us insert or remove that string.
- We also have generators that let us switch any letter with an adjacent blank.

The Construction



- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.
- $V(\mathcal{G}) = \mathbb{N}$, $E(\mathcal{G}) = \{n, n+1 \mid n \in \mathbb{N}\}$.
- $\mathcal{A} = X^{\pm 1} \cup \{\sqcup\}$.
- \mathbf{u} has the word w written across the leftmost vertices.
- For each inverse pair $x\bar{x}$ and each relator in Y , we have a generator that lets us insert or remove that string.
- We also have generators that let us switch any letter with an adjacent blank.

The Construction



- Consider the group $D_3 = \langle r, f \mid r^3, f^2, (rf)^2 \rangle$, $w = frfr$.
- $V(\mathcal{G}) = \mathbb{N}$, $E(\mathcal{G}) = \{n, n+1 \mid n \in \mathbb{N}\}$.
- $\mathcal{A} = X^{\pm 1} \cup \{\sqcup\}$.
- \mathbf{u} has the word w written across the leftmost vertices.
- For each inverse pair $x\bar{x}$ and each relator in Y , we have a generator that lets us insert or remove that string.
- We also have generators that let us switch any letter with an adjacent blank.
- S is just the all-blank state; we can get here iff $w \equiv e$.

Proof of Undecidability

Theorem

CONFIGURABILITY *is undecidable.*

Proof of Undecidability

Theorem

CONFIGURABILITY *is undecidable*.

Proof.

Suppose not.

Proof of Undecidability

Theorem

CONFIGURABILITY *is undecidable*.

Proof.

Suppose not. Then there is an algorithm which solves every instance of CONFIGURABILITY.

Proof of Undecidability

Theorem

CONFIGURABILITY *is undecidable*.

Proof.

Suppose not. Then there is an algorithm which solves every instance of CONFIGURABILITY. But this gives an algorithm for solving every instance of the word problem:

Proof of Undecidability

Theorem

CONFIGURABILITY *is undecidable*.

Proof.

Suppose not. Then there is an algorithm which solves every instance of CONFIGURABILITY. But this gives an algorithm for solving every instance of the word problem:

- Let $\langle X|Y \rangle$, and $w \in X^*$, be given.

Proof of Undecidability

Theorem

CONFIGURABILITY *is undecidable*.

Proof.

Suppose not. Then there is an algorithm which solves every instance of CONFIGURABILITY. But this gives an algorithm for solving every instance of the word problem:

- Let $\langle X|Y \rangle$, and $w \in X^*$, be given.
- Translate the word problem into an instance of CONFIGURABILITY.

Proof of Undecidability

Theorem

CONFIGURABILITY *is undecidable*.

Proof.

Suppose not. Then there is an algorithm which solves every instance of CONFIGURABILITY. But this gives an algorithm for solving every instance of the word problem:

- Let $\langle X|Y \rangle$, and $w \in X^*$, be given.
- Translate the word problem into an instance of CONFIGURABILITY.
- Use our algorithm to figure out whether \mathbf{u} can reach the blank state in the CONFIGURABILITY instance.

Proof of Undecidability

Theorem

CONFIGURABILITY *is undecidable*.

Proof.

Suppose not. Then there is an algorithm which solves every instance of CONFIGURABILITY. But this gives an algorithm for solving every instance of the word problem:

- Let $\langle X|Y \rangle$, and $w \in X^*$, be given.
- Translate the word problem into an instance of CONFIGURABILITY.
- Use our algorithm to figure out whether \mathbf{u} can reach the blank state in the CONFIGURABILITY instance.
- $w \equiv e$ if and only if it was possible for \mathbf{u} to reach the blank state.

Proof of Undecidability

Theorem

CONFIGURABILITY *is undecidable*.

Proof.

Suppose not. Then there is an algorithm which solves every instance of CONFIGURABILITY. But this gives an algorithm for solving every instance of the word problem:

- Let $\langle X|Y \rangle$, and $w \in X^*$, be given.
- Translate the word problem into an instance of CONFIGURABILITY.
- Use our algorithm to figure out whether \mathbf{u} can reach the blank state in the CONFIGURABILITY instance.
- $w \equiv e$ if and only if it was possible for \mathbf{u} to reach the blank state.

Since there are f.p. groups with unsolvable word problem, no such general algorithm for the word problem can exist.

Proof of Undecidability

Theorem

CONFIGURABILITY *is undecidable*.

Proof.

Suppose not. Then there is an algorithm which solves every instance of CONFIGURABILITY. But this gives an algorithm for solving every instance of the word problem:

- Let $\langle X|Y \rangle$, and $w \in X^*$, be given.
- Translate the word problem into an instance of CONFIGURABILITY.
- Use our algorithm to figure out whether \mathbf{u} can reach the blank state in the CONFIGURABILITY instance.
- $w \equiv e$ if and only if it was possible for \mathbf{u} to reach the blank state.

Since there are f.p. groups with unsolvable word problem, no such general algorithm for the word problem can exist. Therefore, there cannot be an algorithm for solving every instance of CONFIGURABILITY. \square

Other Results

- Another approach to undecidability: simulate Turing machines directly

Other Results

- Another approach to undecidability: simulate Turing machines directly
- Case where \mathcal{G} , \mathcal{A} finite
 - ▶ First result: NP-hard, by reduction from HAMILTONIAN PATH
 - ▶ Stronger result: PSPACE-complete, by simulating “nondeterministic constraint logic”

Further Reading



R.C. Lyndon and P.E. Schupp.
Combinatorial Group Theory.
Springer, 1977.



R. Ghrist and V. Peterson.
The Geometry and Topology of Reconfiguration.
Advances in Applied Mathematics **38** (2006), 302–323.