# Conjugacy in Higman-Thompson groups

David Robertson

15th April, 2015

# Crash course on groups

- Group: set with a nice binary operation
  - $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{C}$ with addition
  - non zero reals, invertible matrices with multiplication
  - invertible maps $f : X \to X$ with function composition

# Crash course on groups

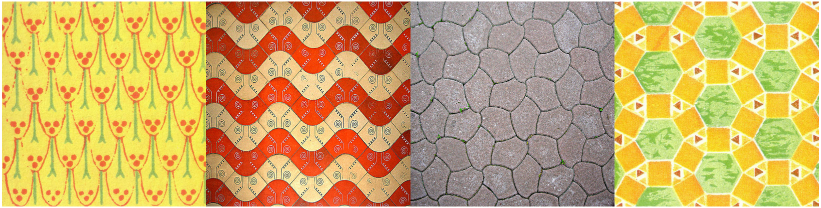
- Group: set with a nice binary operation
  - $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{C}$ with addition
  - non zero reals, invertible matrices with multiplication
  - invertible maps $f\colon X \to X$ with function composition
- 'Nice': operation has identity $e$ and inverses $x^{-1}$
  - $ex = xe = x$ e.g. $0$, $1$, $x \mapsto x$
  - $xx^{-1} = x^{-1}x = e$ e.g. $-x$, $1/x$, $f(x) \mapsto x$
  - $x(yz) = (xy)z$

# Crash course on groups

- Group: set with a nice binary operation
  - $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{C}$ with addition
  - non zero reals, invertible matrices with multiplication
  - invertible maps $f \colon X \to X$ with function composition
- 'Nice': operation has identity $e$ and inverses $x^{-1}$
  - $ex = xe = x$ e.g. $0$, $1$, $x \mapsto x$
  - $xx^{-1} = x^{-1}x = e$ e.g. $-x$, $1/x$, $f(x) \mapsto x$
  - $x(yz) = (xy)z$
- Used (e.g.) to describe/measure symmetry

# Crash course on groups

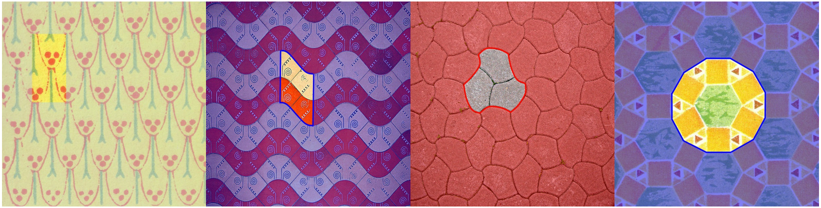
- Group: set with a nice binary operation
  - $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{C}$ with addition
  - non zero reals, invertible matrices with multiplication
  - invertible maps $f : X \to X$ with function composition
- 'Nice': operation has identity $e$ and inverses $x^{-1}$
  - $ex = xe = x$ e.g. $0$, $1$, $x \mapsto x$
  - $xx^{-1} = x^{-1}x = e$ e.g. $-x$, $1/x$, $f(x) \mapsto x$
  - $x(yz) = (xy)z$
- Used (e.g.) to describe/measure symmetry



Images from Wikipedia

# Crash course on groups

- Group: set with a nice binary operation
  - $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{C}$ with addition
  - non zero reals, invertible matrices with multiplication
  - invertible maps $f: X \to X$ with function composition
- 'Nice': operation has identity $e$ and inverses $x^{-1}$
  - $ex = xe = x$ — e.g. $0$, $1$, $x \mapsto x$
  - $xx^{-1} = x^{-1}x = e$ — e.g. $-x$, $1/x$, $f(x) \mapsto x$
  - $x(yz) = (xy)z$
- Used (e.g.) to describe/measure symmetry



Images from Wikipedia

# The conjugacy problem

Given a group $G$ and elements $x$, $y$ can you find a conjugator $z$?

$$z^{-1}xz = y$$

# The conjugacy problem

Given a group $G$ and elements $x$, $y$ can you find a conjugator $z$?

$$z^{-1}xz = y$$

[$G$, $x$, $y$ given in terms of a presentation]

- This problem is undecidable (P. Novikov, 1954)
- Can't write a single program which correctly answers "Yes" or "No" in finite time for every group $G$.

# The conjugacy problem

Given a group $G$ and elements $x$, $y$ can you find a conjugator $z$?

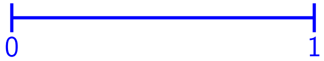
$$z^{-1}xz = y$$

[$G$, $x$, $y$ given in terms of a presentation]

- This problem is undecidable (P. Novikov, 1954)
- Can't write a single program which correctly answers "Yes" or "No" in finite time for every group $G$.
- Cryptosystem?
- Can do this for specific groups, *i.e.* with $G$ fixed.

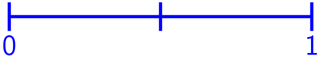
# A recipe for $F$

- Take the interval $[0, 1]$.



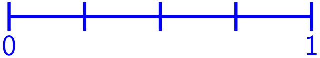
# A recipe for $F$

- Take the interval $[0, 1]$.
- Chop it in half.



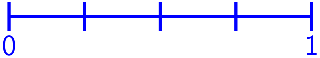
# A recipe for $F$

- Take the interval [0, 1].
- Chop it in half.
- If you like, chop some of the halves in half.



# A recipe for $F$

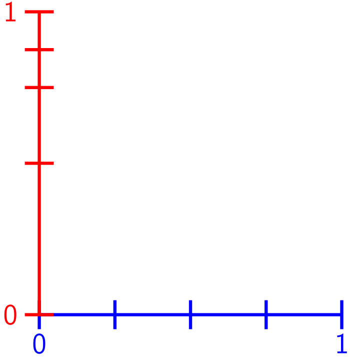
- Take the interval [0, 1].
- Chop it in half.
- If you like, chop some of the halves in half.
- Continue until you get bored.



# A recipe for $F$

- Take the interval [0, 1].
- Chop it in half.
- If you like, chop some of the halves in half.
- Continue until you get bored.

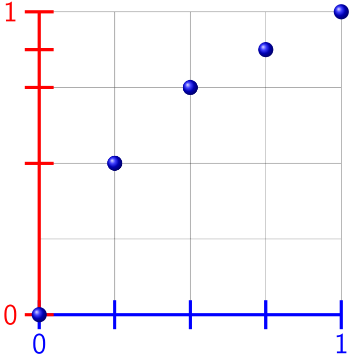
- Do the same to a second copy of $[0, 1]$. Same number of chops!



# A recipe for $F$

- Take the interval [0, 1].
- Chop it in half.
- If you like, chop some of the halves in half.
- Continue until you get bored.

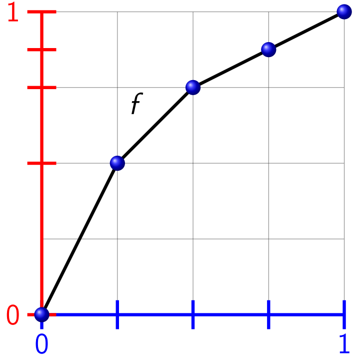
- Do the same to a second copy of $[0, 1]$. Same number of chops!
- Use the intervals as axes and join the dots.



# A recipe for $F$

- Take the interval [0, 1].
- Chop it in half.
- If you like, chop some of the halves in half.
- Continue until you get bored.

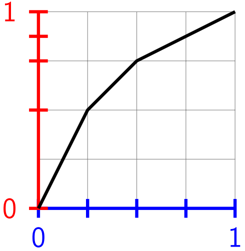
- Do the same to a second copy of $[0, 1]$. Same number of chops!
- Use the intervals as axes and join the dots.




Functions $f$ like this are the elements of Thompson's group $F$.

# Thompson's other groups $T$ and $V$

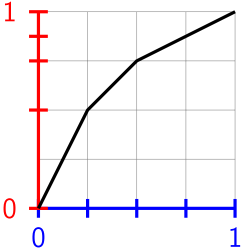
$F$



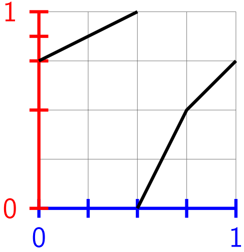

$F$: increasing functions

# Thompson's other groups $T$ and $V$

$$F < T$$



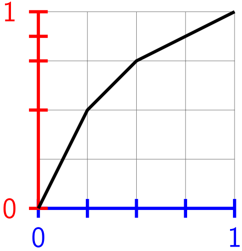

$F$: increasing functions



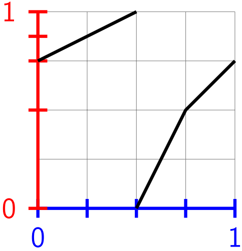

$T$: don't have to start at $(0,0)$

# Thompson's other groups $T$ and $V$

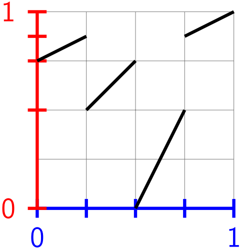
$$F < T < V$$




$F$: increasing functions




$T$: don't have to start at $(0, 0)$




$V$: don't have to be continuous

# People find these groups interesting

$T$ and $V$ are finitely presented, infinite simple groups (rare!)

- Finitely Presented: described by finitely many equations.
- Simple: can't 'compress' the group to get a smaller one.

# People find these groups interesting

$T$ and $V$ are finitely presented, infinite simple groups (rare!)

- Finitely Presented: described by finitely many equations.
- Simple: can't 'compress' the group to get a smaller one.

$V$ contains a copy of every finite group $G$:

- $G \hookrightarrow \mathcal{S}_n \hookrightarrow \mathcal{S}_{2^m}$
- Take two full binary trees of depth $m$, each with $2^m$ leaves.
- $\sigma \in \mathcal{S}_{2^m}$ corresponds to a permutation of the leaves.

# People find these groups interesting

$T$ and $V$ are finitely presented, infinite simple groups (rare!)

- Finitely Presented: described by finitely many equations.
- Simple: can't 'compress' the group to get a smaller one.

$V$ contains a copy of every finite group $G$:

- $G \hookrightarrow \mathcal{S}_n \hookrightarrow \mathcal{S}_{2^m}$
- Take two full binary trees of depth $m$, each with $2^m$ leaves.
- $\sigma \in \mathcal{S}_{2^m}$ corresponds to a permutation of the leaves.

People are also interested in whether $F$ is amenable or not,

whatever that means...

## Conjugacy algorithms in $V$

1974 Higman described a conjugacy algorithm in a series of lectures.

- Works for $V = G_{2,1}$ and generalisations $G_{n,r}$.

# Conjugacy algorithms in $V$

1974 Higman described a conjugacy algorithm in a series of lectures.

- Works for $V = G_{2,1}$ and generalisations $G_{n,r}$.
- Different tools used to address conjugacy since:
  - 1997 Guba, Sapir: diagram groups
  - 2007 Belk, Matucci: strand diagrams
  - 2010 Salazar-Díaz: revealing tree pairs
  - 2011 Bleak *et al.*: train tracks and flow graphs

# Conjugacy algorithms in $V$

1974 Higman described a conjugacy algorithm in a series of lectures.

- Works for $V = G_{2,1}$ and generalisations $G_{n,r}$.
- Different tools used to address conjugacy since:

  1997 Guba, Sapir: diagram groups

  2007 Belk, Matucci: strand diagrams

  2010 Salazar-Díaz: revealing tree pairs

  2011 Bleak *et al.*: train tracks and flow graphs

2014 Barker used Higman's ideas to solve power conjugacy in $V$.

# Conjugacy algorithms in $V$

1974 Higman described a conjugacy algorithm in a series of lectures.

- Works for $V = G_{2,1}$ and generalisations $G_{n,r}$.
- Different tools used to address conjugacy since:

  1997 Guba, Sapir: diagram groups
  2007 Belk, Matucci: strand diagrams
  2010 Salazar-Díaz: revealing tree pairs
  2011 Bleak *et al.*: train tracks and flow graphs

2014 Barker used Higman's ideas to solve power conjugacy in $V$.

Given $x, y$ can you solve $z^{-1}x^a z = y^b$?

# Conjugacy algorithms in $V$

1974 Higman described a conjugacy algorithm in a series of lectures.

- Works for $V = G_{2,1}$ and generalisations $G_{n,r}$.
- Different tools used to address conjugacy since:

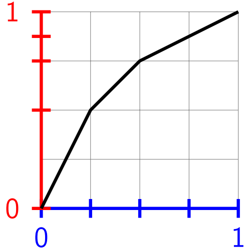
  1997 Guba, Sapir: diagram groups
  2007 Belk, Matucci: strand diagrams
  2010 Salazar-Díaz: revealing tree pairs
  2011 Bleak *et al.*: train tracks and flow graphs

2014 Barker used Higman's ideas to solve power conjugacy in $V$.

Given $x, y$ can you solve $z^{-1}x^a z = y^b$?

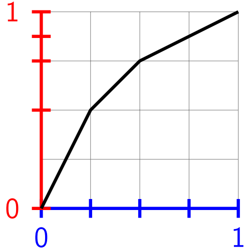
2015 Barker, Duncan and R. Generalisation to $G_{n,r}$ and corrections.
Proof of concept implementation.

# Interval partitions $\to$ trees




How can we store this in memory?

# Interval partitions $\to$ trees

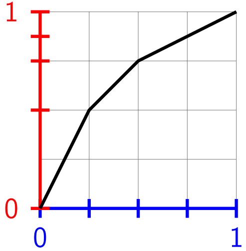



How can we store this in memory?

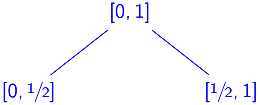
$[0, 1]$

$[0, 1]$

# Interval partitions → trees

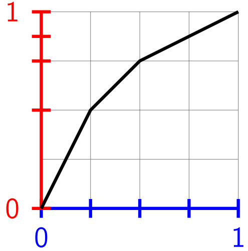



How can we store this in memory?

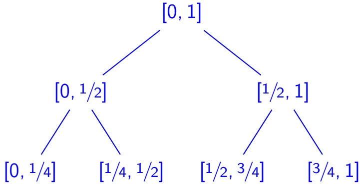



[0, 1]

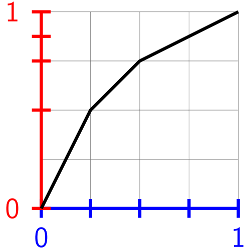
# Interval partitions → trees



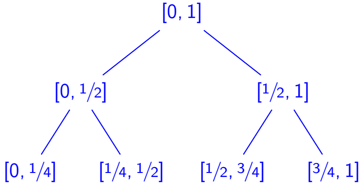

How can we store this in memory?

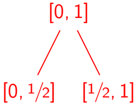



# Interval partitions → trees

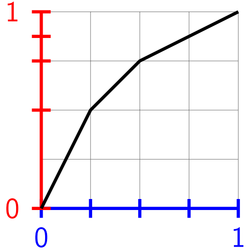



How can we store this in memory?

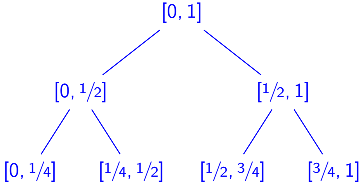


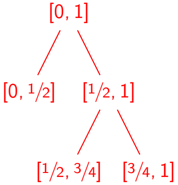



# Interval partitions → trees

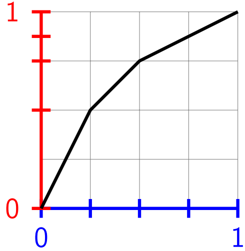



How can we store this in memory?

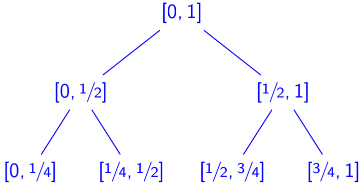


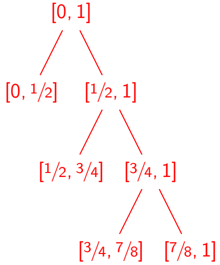



# Interval partitions → trees




How can we store this in memory?



# Trees $\rightarrow$ paths and words

- Trees aren't always easy to work with
- Can only tell if you're at the top (root) or bottom (leaf)
- Recursively delegate to children

# Trees $\to$ paths and words

- Trees aren't always easy to work with
- Can only tell if you're at the top (root) or bottom (leaf)
- Recursively delegate to children
- Quickly becomes confusing! for me, at least...
- Where does $f$ send $[{}^{34}/_{128}, {}^{35}/_{128}]$?

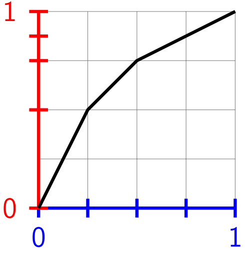
# Trees $\to$ paths and words

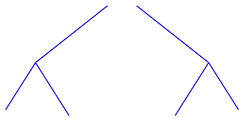
- Trees aren't always easy to work with
- Can only tell if you're at the top (root) or bottom (leaf)
- Recursively delegate to children
- Quickly becomes confusing! for me, at least...
- Where does $f$ send $[{}^{34}/_{128}, {}^{35}/_{128}]$?

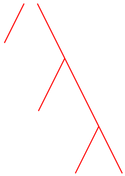
Higman described paths in the tree using an algebra. Introduce labels:

$$\text{Root} \mapsto x \qquad \text{left} \mapsto \alpha_1 \qquad \text{right} \mapsto \alpha_2$$

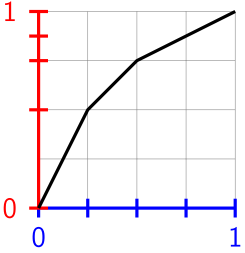
# Trees → paths and words

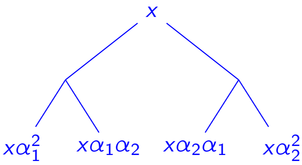


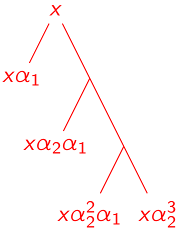





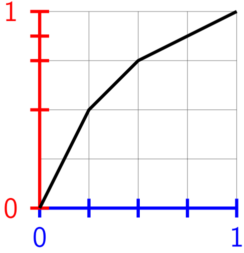
# Trees → paths and words

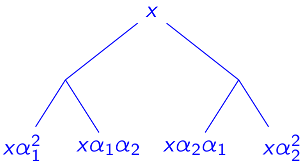


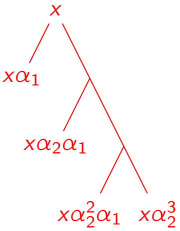



# Trees → paths and words



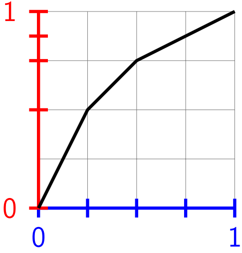



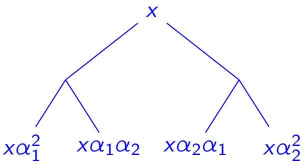



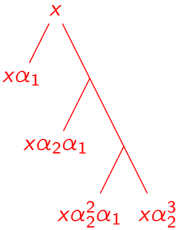

Maps specified by lists of domain and range words.

$$\begin{aligned} x\alpha_1^2 &\mapsto x\alpha_1 & \qquad x\alpha_2\alpha_1 &\mapsto x\alpha_2^2\alpha_1 \\ x\alpha_1\alpha_2 &\mapsto x\alpha_2\alpha_1 & \qquad x\alpha_2^2 &\mapsto x\alpha_2^3 \end{aligned}$$

# Trees → paths and words




Maps specified by lists of domain and range words.

$$\begin{array}{llcll} x\alpha_1^2 & \mapsto x\alpha_1 & \quad & x\alpha_2\alpha_1 & \mapsto x\alpha_2^2\alpha_1 \\ x\alpha_1\alpha_2 & \mapsto x\alpha_2\alpha_1 & \quad & x\alpha_2^2 & \mapsto x\alpha_2^3 \end{array}$$

Easier to compute components, e.g.

$$\begin{array}{ccccccc} \ldots & \mapsto & x\alpha_1^3 & \mapsto & x\alpha_1^2 & \mapsto & x\alpha_1 \\ \ldots & \mapsto & [0, 1/8] & \mapsto & [0, 1/4] & \mapsto & [0, 1/2] \end{array}$$

# Components ($\approx$ orbits)

- Pick your favourite word e.g. $w = x\alpha_1^2 \leftrightarrow [0, 1/4]$.
- Compute component of $w$ until you can't any more:

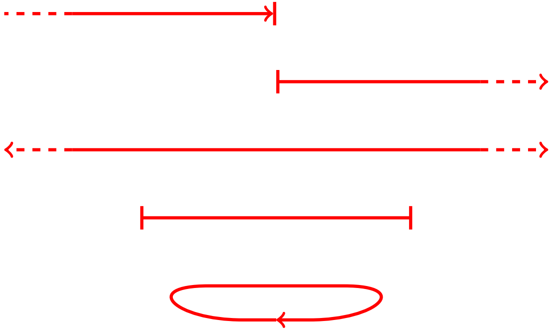
$$\ldots, f^{-2}, f^{-1}(w), w, f(w), f^2(w), \ldots$$

# Components ($\approx$ orbits)

- Pick your favourite word e.g. $w = x\alpha_1^2 \leftrightarrow [0, 1/4]$.
- Compute component of $w$ until you can't any more:

$$\ldots, f^{-2}, f^{-1}(w), w, f(w), f^2(w), \ldots$$

Components come in five different shapes:



# Components and Conjugacy

1. Break down a map $f \in V$ into components.

# Components and Conjugacy

1. Break down a map $f \in V$ into components.
2. If $g \in V$ is conjugate to $f$, the components of $g$ must match the components of $f$.

   e.g. periodic $\mapsto$ periodic, with the same period

# Components and Conjugacy

1. Break down a map $f \in V$ into components.
2. If $g \in V$ is conjugate to $f$, the components of $g$ must match the components of $f$.
   e.g. periodic $\mapsto$ periodic, with the same period
3. Only finitely many matchings—check them all!

# Components and Conjugacy

1. Break down a map $f \in V$ into components.
2. If $g \in V$ is conjugate to $f$, the components of $g$ must match the components of $f$.

   e.g. periodic $\mapsto$ periodic, with the same period
3. Only finitely many matchings—check them all!
4. If one works: we get a conjugator $h$.

# Components and Conjugacy

1. Break down a map $f \in V$ into components.
2. If $g \in V$ is conjugate to $f$, the components of $g$ must match the components of $f$.

   e.g. periodic $\mapsto$ periodic, with the same period
3. Only finitely many matchings—check them all!
4. If one works: we get a conjugator $h$.
5. If none of them work: no conjugator exists.

# Implementation

- My job: get a computer to do this.
- Like programming a fancy calculator!
- Wrote in Python: something I knew and quick to work with.

# Implementation

- My job: get a computer to do this.
- Like programming a fancy calculator!
- Wrote in Python: something I knew and quick to work with.
- Summer project that was more like a semester project...

# Implementation

- My job: get a computer to do this.
- Like programming a fancy calculator!
- Wrote in Python: something I knew and quick to work with.
- Summer project that was more like a semester project...
- Other tools exist to do calculations in $V$, but not to solve the conjugacy problem.

Code is on GitHub. Come and find me if you want a demo!

# Sphinx: comments in source code

```
def format(word):
    """Turns a sequence of integers representing a *word* into [...]
        >>> format([2, −1, 2, −2, 0])
        'x2 a1 x2 a2 L'
        >>> format([])
        The Spanish Inquisition
    """
    if len(word) == 0:
        return "<the empty word>"
    return " ".join(_char(i) for i in word)
```

Sphinx generates nice HTML documentation and runs tests based on *"""comments like this"""*.

# Sphinx: doctest

```
H:\thompsons_v\docs>make doctest
[...]
****************************************************
File "thompson.word.rst", line 10, in default
Failed example:
    format([])
Expected:
    The Spanish Inquisition
Got:
    '<the empty word>'
*****************************************************
1 items had failures:
   1 of 100 in default
100 tests in 1 items.
99 passed and 1 failed.
***Test Failed*** 1 failures.
```

# Other lessons learned

- Small test suites—catch bugs before they happen
- Generate random examples
- Immutable words
- Document the code

# Future Work

Code

- More testing
- Complexity analysis

Theory

- Simultaneous conjugacy
  Given $x_1, \ldots, x_n; y_1, \ldots, y_n$ find a single conjugator $z$ such that
  $$z^{-1} x_i z = y_i, \quad \forall i$$
- Try to solve different kinds of equations?
- Transfer to more general Thompson-like groups $V(\Sigma)$?