Extending the concept of an automatic group

Murray Elder, (U Newcastle, Australia)
Jennifer Taback, (Bowdoin, Maine USA)

AustMS 2013, Algebra special session

G a group with finite generating set X

would like to compute efficiently, *i.e.*

- recognise equality
- see when u, v differ by multiplication by $x \in$ X
- find a representative for elements

quickly.

Automatic groups were designed for this purpose

Automatic group

(G, X) is **automatic** if there is a language L$\subseteq$ X*

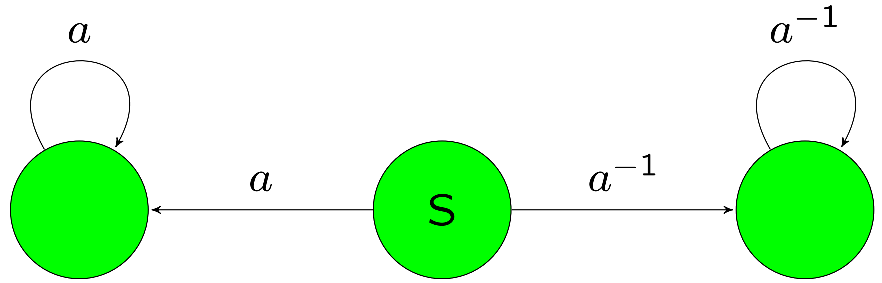
in bijection with group elements

so that L is regular (recognised by a FSA)

and for each $x \in$ X there is a FSA to check whether $u, v \in$ L satisfy $v = ux$

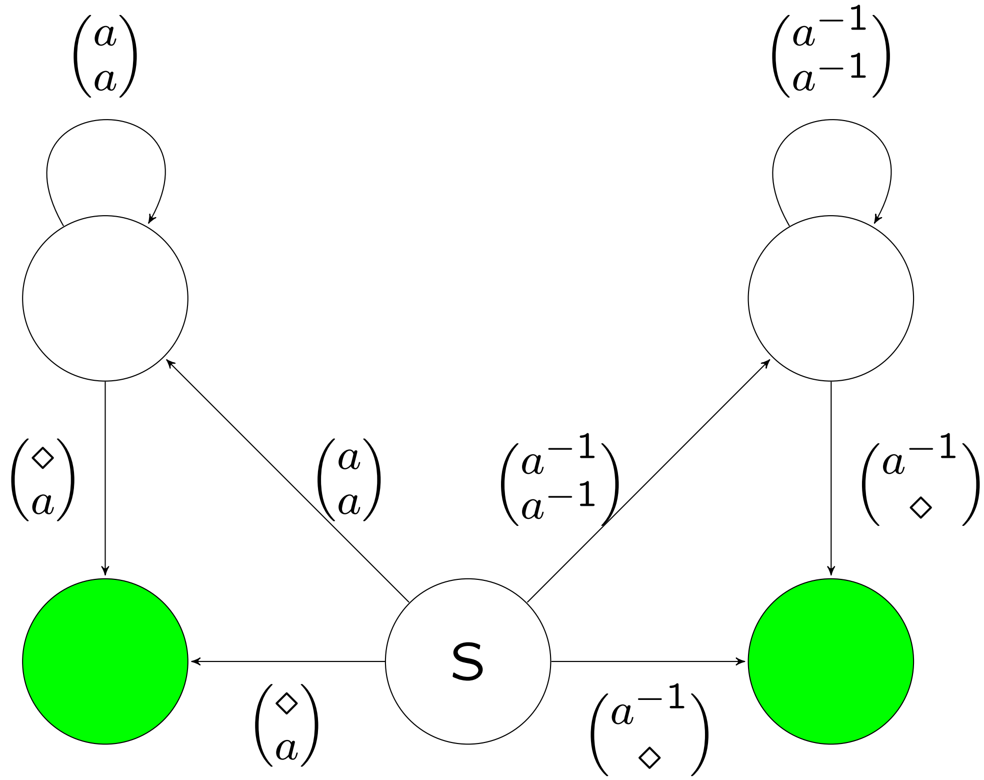
Example

$\mathbb{Z} = \langle a \rangle$ is automatic with $\mathsf{L} = \{a^i \mid i \in \mathbb{Z}\}$.



Example

$\mathbb{Z} = \langle a \rangle$ is automatic with $\mathsf{L} = \{a^i \mid i \in \mathbb{Z}\}$.



Graph automatic group

Kharlampovich, Khoussainov and Miasnikov modified the definition as follows:

let G be a group with (finite or not)† generating set X, and Λ be a **finite set of symbols**

† they didn't say X could be infinite, but its no problem

Graph automatic group

(G, Λ, X) is **graph automatic** if there is a language L$\subseteq$ Λ* st

$\overline{\ }$: L $\rightarrow$ G is a bijection

L is regular (recognised by a FSA)

for each $x \in$ X there is a FSA to check whether $u, v \in$ L satisfy $\overline{v} = \overline{u}x$

$\mathcal{C}$-graph automatic group

Jen Taback and I extended the definition a bit further:

let $\mathcal{C}$ be a class of formal languages

eg. context-free
k-counter
indexed
poly-context-free
context-sensitive ...

$\mathcal{C}$-graph automatic group

(G, Λ, X) is **$\mathcal{C}$-graph automatic** if there is a language L$\subseteq$ Λ* st

$\overline{}$: L $\to$ G is a bijection

L is in $\mathcal{C}$

for each $x \in$ X there is a $\mathcal{C}$-automaton to check whether $u, v \in$ L satisfy $\overline{v} = \overline{u}x$

$\mathcal{C}$-graph automatic group

Before I go through an example, here is what we proved:

Thm if (G, X, Λ) is k-counter-graph automatic then there is an algorithm that:

on input a word $w \in X^*$ of length n,

computes the L-word (in Λ^*) for w in time n^{k+2}

Cor if we know the L-word for the identity, we can solve the *word problem* in polynomial time

Examples

We proved that Thompson's group F and Baumslag-Solitar groups BS$(m,n) = \langle a,t \mid ta^m t^{-1} = a^n \rangle$ are 3-counter-graph automatic.

KKM proved BS$(1,n)$ are graph automatic, and asked about F.

See our paper(s) to appear soon on the arXiv. The F result is joint with **Sharif Younes** (undergraduate student project at Bowdoin).

Examples

We proved that Thompson's group F and Baumslag-Solitar groups BS$(m, n) = \langle a, t \mid ta^m t^{-1} = a^n \rangle$ are 3-counter-graph automatic.

KKM proved BS$(1, n)$ are graph automatic, and asked about F.

See our paper(s) to appear soon on the arXiv. The F result is joint with **Sharif Younes** (undergraduate student project at Bowdoin).

The example I will give here is the free group F_∞ with countably infinite basis.

Kharlampovich, Khoussainov and Miasnikov

Example

$F_\infty = \langle x_1, x_2, \cdots \mid \ \rangle$ is not finitely generated, so cannot be automatic (the language L must be over a finite alphabet)

We can represent a (freely reduced) word in the generators x_i, x_i^{-1} $i = 1, 2, 3, \ldots$ as follows.

- map x_i to $p1^i$
- map x_i^{-1} to $n1^i$

For example $(x_3)^2\,(x_4)^{-1}$ is $p111p111n1111$

Example

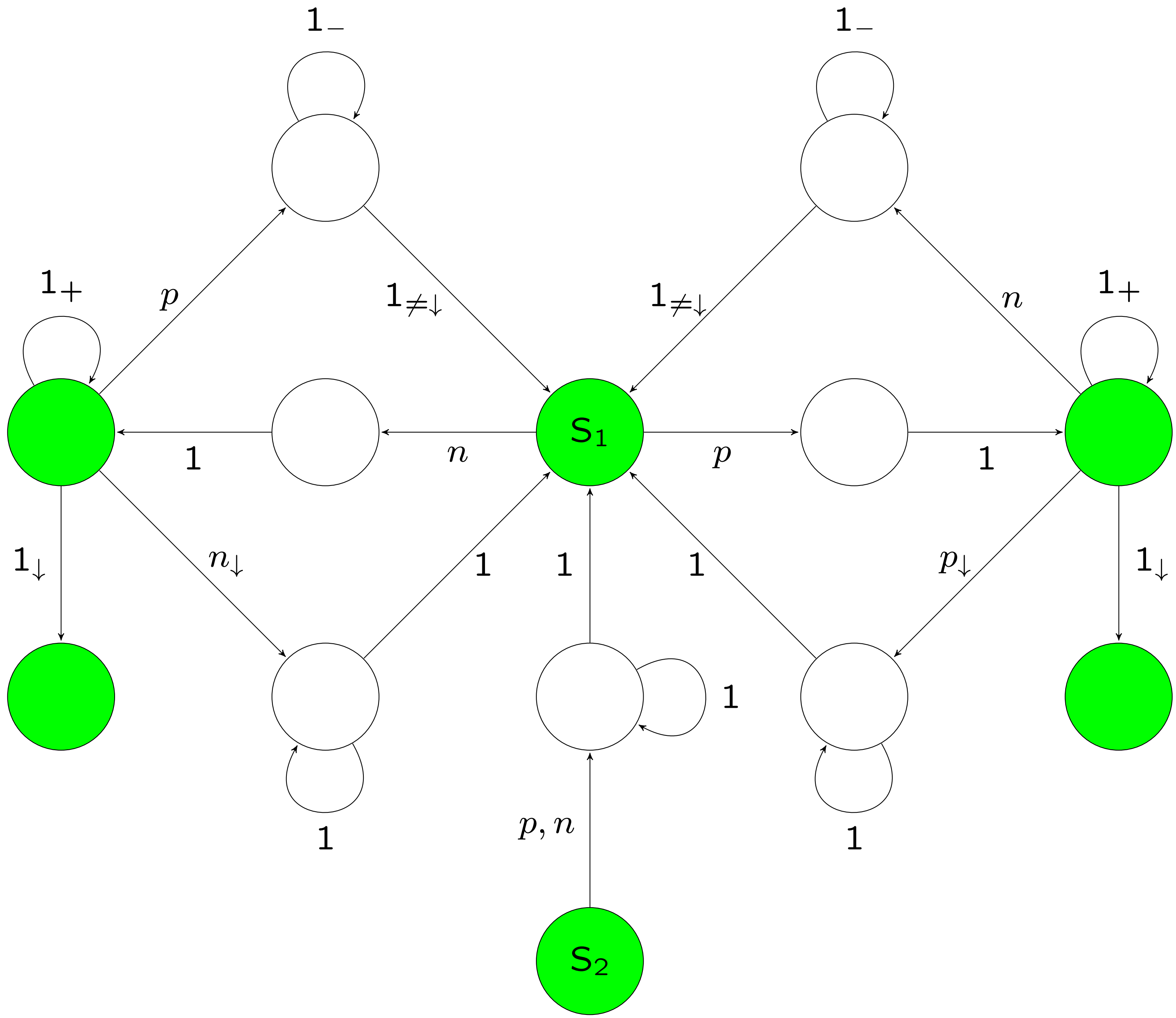
let L_1 be the set of strings

$$\underbrace{p/n\ 1^{i_1}\ p/n\ 1^{i_2}}\ \underbrace{p/n\ 1^{i_3}\ p/n\ 1^{i_4}}\ \ldots\ \underbrace{p/n\ 1^{i_{2n-1}}\ p/n\ 1^{i_{2n}}}\ p/n\ 1^{i_{2n+1}}$$

and L_2 the set of strings

$$p/n\ 1^{i_1}\ \underbrace{p/n\ 1^{i_2}\ p/n\ 1^{i_3}}\ \underbrace{p/n\ 1^{i_4}\ p/n\ 1^{i_5}}\ \ldots\ \underbrace{p/n\ 1^{i_{2n}}\ p/n\ 1^{i_{2n+1}}}\ p/n\ 1^{i_{2n+1}}$$

where underbrace means the pair is freely reduced
(*i.e.* $p1^in1^i, n1^ip1^i$ not allowed)



1_-
1_-
1_+
p
$1_{\neq\downarrow}$
$1_{\neq\downarrow}$
n
1_+
S_1
1
n
p
1
$1_\downarrow$
$n_\downarrow$
1
1
1
$p_\downarrow$
$1_\downarrow$
1
1
1
p, n
S_2

Example

So $L=L_1 \cap L_2$ and is accepted by the (non-blind, non-deterministic) 2-counter automata that is is the intersection of the two machines (start at S_1 for L_1, S_2 for L_2)

Example

So L=$L_1 \cap L_2$ and is accepted by the (non-blind, non-deterministic) 2-counter automata that is is the intersection of the two machines (start at S_1 for L_1, S_2 for L_2)

Define $\otimes(\mathsf{L},\mathsf{L})$ to be the set of strings

$$\otimes(u,v) = \begin{cases} \binom{u_1}{v_1}\binom{u_2}{v_2}\ldots\binom{u_i}{v_i}\binom{\diamond}{v_{i+1}}\ldots\binom{\diamond}{v_j} & \text{if} \quad |u| \leq |v| \\ \\ \binom{u_1}{v_1}\binom{u_2}{v_2}\ldots\binom{u_j}{v_j}\binom{u_{i+1}}{\diamond}\ldots\binom{u_j}{\diamond} & \text{if} \quad |u| > |v| \end{cases}$$

Example

Then L_{x_i} is the set of strings in $\otimes(\mathsf{L},\mathsf{L})$ of the form

$$\binom{r_1}{r_1}\binom{1}{1}^{\eta_1}\binom{r_2}{r_2}\binom{1}{1}^{\eta_2}\cdots\binom{r_k}{r_k}\binom{1}{1}^{\eta_k}\binom{\diamond}{p}\binom{\diamond}{1}^{i}$$

if $r_k = p$ or $\eta_k \neq i$,

and

$$\binom{r_1}{r_1}\binom{1}{1}^{\eta_1}\binom{r_2}{r_2}\binom{1}{1}^{\eta_2}\cdots\binom{r_{k-1}}{r_{k-1}}\binom{1}{1}^{\eta_{k-1}}\binom{n}{\diamond}\binom{1}{\diamond}^{i}.$$

See paper for details. But its easy — you just intersect (a modified version of) the above machine with a FSA to check the suffix

Thanks − and for more :

http://www.stevens.edu/algebraic/GTI/

Dmytro Savchuk (University of South Florida)

An Example of an Automatic Graph of Intermediate Growth

noon Thurs New York time (2am Fri Syd time)

Paper(s) to appear on the arXiv very soon