Studying monoids that model concurrency

Sarah Rees

University of Newcastle

NBSAN, Durham, 15th December 2025

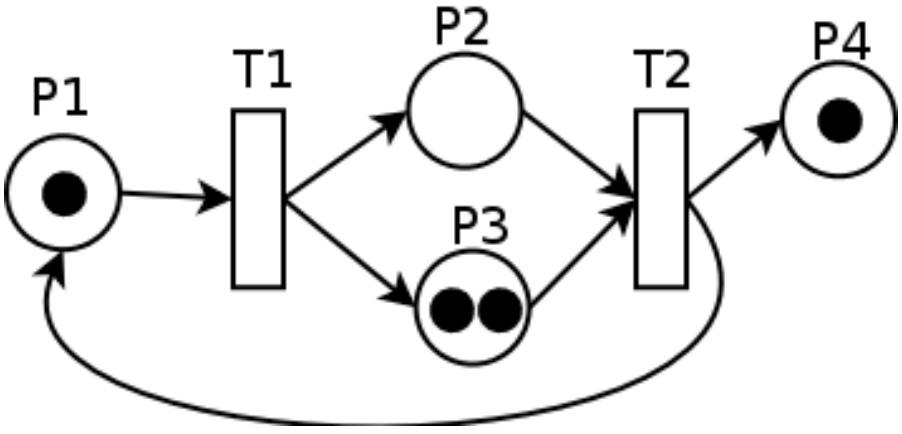
Introduction, plan for today

Joint work of mine with Ascencio-Martin, Britnell, Duncan, Francoeur, Koutny (ABDFKR) to set up and study algebraic models of concurrent computation. Today's plan:

- I: Introduction, defn. of petri nets as models of concurrency, traces and trace monoids. Generalising to comtraces, step traces, and associated monoids.
  The problems we'd like to solve.
- II: Normal forms for traces, comtraces and step traces.
- III: Folding in free groups, finitely presented groups and monoids.
- IV: And where next? e.g.
  - do we really have the right models? (Probably not)
  - are we attacking the right problems?

I: Petri nets and trace monoids

Petri nets (Carl Petri, 1939-2010) model concurrent systems (computer networks, rail networks, many biological systems). A **net** is a finite, directed bipartite graph, with vertex classes S (the **states/places**), A (the **actions/transitions)**. Arrows (arcs) $s \to a \to s'$ indicates that a acts on s, with result s'.

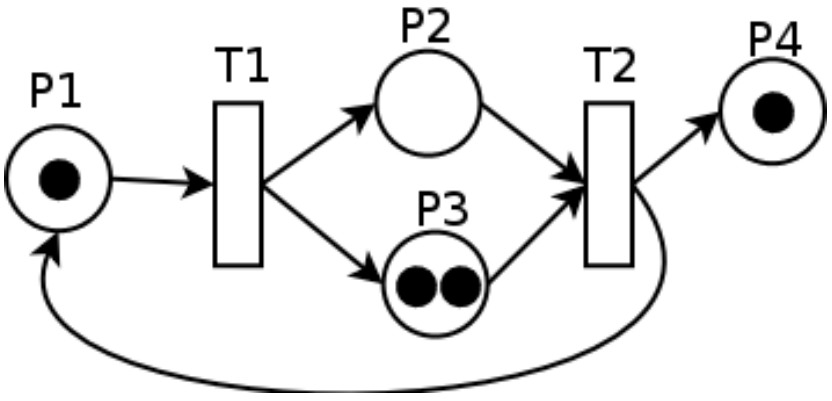


Example of a Petri net found on wikipedia, ©Msoos.
Rectangles depict transitions, white circles places, black circles input tokens (need to be in place for a transition to fire).

For each a, we assume that $\mathrm{in}(a), \mathrm{out}(a) \neq \emptyset$, and $\mathrm{in}(a) \cap \mathrm{out}(a) = \emptyset$. We define $\mathrm{lk}(a) := \mathrm{in}(a) \cup \mathrm{out}(a)$. We call a sequence of actions that might be performed within a system an **action sequence**.

In the example, we start with input in $P1$ only. Then any alternating string of $T1$ and $T2$ is an action sequence.

Actions sequences, equivalence, traces, trace monoids



Example of a Petri net found on wikipedia, ©Msoos.

Actions $a, b \in A$ are **independent** if $\mathrm{lk}(a) \cap \mathrm{lk}(b) = \emptyset$; then, action sequences $uabv$ and $ubav$ have the same effect. The equiv. reln. on action sequences A^* induced by independence on A is called **trace equivalence**, and its equivalence classes are called **traces**.

(Mazurkiewicz87,89) defined the **trace monoid** $\mathcal{T} := \langle A \mid R \rangle$, where $R := \{ab = ba : a, b \in A \text{ are independent}\}$; its elements are traces. As an abstract monoid it has been well studied (eg CartierFoata69), has many names, eg **right-angled Artin monoid** (RAAM). RAAMs embed in **right-angled Artin groups** (RAAGs), very familiar to group theorists.

Generalising traces to comtraces and step traces

JanickiKoutny95 and JanickiEtAl16 defined models of **combined traces** (**comtraces**) and **step traces** that recognise that two or more out of (i) the simultaneous application of actions a and b, (ii) the sequence ab and (iii) the sequence ba might have the same effect. For $a, b \in A$, we write $a \sim_{\mathrm{sim}} b$ if a and b might be performed simultaneously, call the (symmetric) relation sim **simultaneity**, and define $\mathbb{A} \subset 2^A$ to be the set of cliques of sim, which we call **steps**. We describe a computation as a sequence of steps.

Irreflexive, binary relations $\sim_{\mathrm{inl}}$ and $\sim_{\mathrm{ser}}$ on A naturally extend to $\mathbb{A}$:

interleaving For $a, b \in A$, $a \sim_{\mathrm{inl}} b$ if a, b have the same effect when performed in either order, but not when performed simultaneously (symmetric, by defn.),

serialisation For $a, b \in A$, $a \sim_{\mathrm{ser}} b$ if within a sequence ab and $\{a, b\}$ are interchangeable (so $\mathrm{ser} \subseteq \mathrm{sim}$). This is not nec. symmetric, but it may be that $a \sim_{\mathrm{ser}} b$ and $b \sim_{\mathrm{ser}} a$ for certain pairs a, b. For such pairs, $a \not\sim_{\mathrm{inl}} b$ (by defn of $\sim_{\mathrm{inl}}$).

By defn. com- and step traces are the equiv. classes of ser and $\mathrm{inl} \cup \mathrm{ser}$

Definitions of comtrace and step trace monoids, $\mathcal{C}$ and $\mathcal{S}$

Given a finite set A of actions, and simultaneity relation $\sim_{\mathrm{sim}}$, we define

$$\mathbb{A} := \{S \subset A : x \sim_{\mathrm{sim}} y, \forall x \neq y \in S\}$$

Given also a serialisation relation ser, we define the comtrace monoid

$$\mathcal{C} := \langle \mathbb{A} \mid ST = S \cup T \text{ when } S \sim_{\mathrm{ser}} T \text{`} \rangle.$$

Given both a serialisation relation ser and interleaving relation inl, where $\mathrm{ser} \cap \mathrm{ser}^{-1} \cap \mathrm{inl} = \emptyset$, we define the step trace monoid

$$\mathcal{S} := \langle \mathbb{A} \mid ST = TS \text{ when } S \sim_{\mathrm{inl}} T, ST = S \cup T \text{ when } S \sim_{\mathrm{ser}} T \text{`} \rangle.$$

So, elements of any trace, comtrace or step trace monoid correspond to equivalence classes $[w]$ of words w over the generating set A or $\mathbb{A}$, with equivalence determined by the defining relations.

But we can also describe elements graphically, as we shall describe later.

Problems relating to step trace monoids

In general, we want to develop geometric and algebraic/combinatorial methods to study step trace monoids, their substructures, automorphisms and homomorphisms, as well as related structures that might contain them (note that step trace monoids embed in groups), also to study generalised Petri nets that are modelled by com- and step traces. Some problems are already solved for trace monoids, a few for comtrace monoids.

- Find good (geodesic?) normal forms for step trace monoids. Known geodesic normal forms over A for trace monoids are not geodesic over $\mathbb{A}$ using standard word length, so not for step trace monoids.
- Find good descriptions of substructures, eg :
  - good generating sets for submonoids (folding?),
  - rational subset membership (in restricted situations, where we have a chance of success).
- Find step trace analogues of big theorems for trace monoids.
- Examine reachability in Petri nets, verificatn. in (extended) Petri nets.

Some problems can be solved for step trace monoids by considering them as trace monoids, since in fact every step trace monoid is a trace monoid.

In fact, every step trace monoid is a trace monoid

Given a monoid presentation $\langle S|R\rangle$, we define **Tietze transformations**:

- (T1) adding to R a relation $u = v$ that is a consequence of R,
- (T2) deleting from R a relation $u = v$ that is a consequence of $R \setminus \{u = v\}$,
- (T3) adding to S a new generator s, and to R a new relation $s = w$, for some $w \in S^*$,
- (T4) deleting from S a generator s, and from R a relation $s = w$, and replacing all other occurrences of s within relations in R by the word w.

Proposition (ABDFKR23)

Any step trace monoid can be transformed to a trace monoid by a sequence of Tietze transformations.

So step trace monoids embed in RAAMs, and hence in RAAGs. But note that during the transformation from step trace monoid to trace monoid the generating set is changed.

Subrelations

The four binary relations $\mathrm{sim} \setminus \mathrm{ser}$, inl, $\mathrm{ser} \cap \mathrm{ser}^{-1} =: \mathrm{con}$, and $\mathrm{ser} \setminus \mathrm{ser}^{-1}$ are pairwise disjoint on A and $\mathbb{A}$. (**Notn:** $\mathrm{ser}^{-1} := \{(b, a) : (a, b) \in \mathrm{ser}\}$.)

We note that the relation $\mathrm{sim} \setminus (\mathrm{ser} \cup \mathrm{ser}^{-1})$ holds between a and b iff $\{a, b\}$ is not equivalent to either ab or ba; ie iff $\{a, b\}$ cannot be split as a generator.

The union of the remaining three relations on $\mathbb{A}$ is the relation defining the step trace monoid.

We are interested in situations where one or more of those four relations is empty (never holds for $a, b \in A$), and hence the associated step trace monoids lie in a subclass, which we might be better able to understand. In particular we get traces when $\mathrm{sim} = \emptyset$, comtraces when $\mathrm{inl} = \emptyset$.

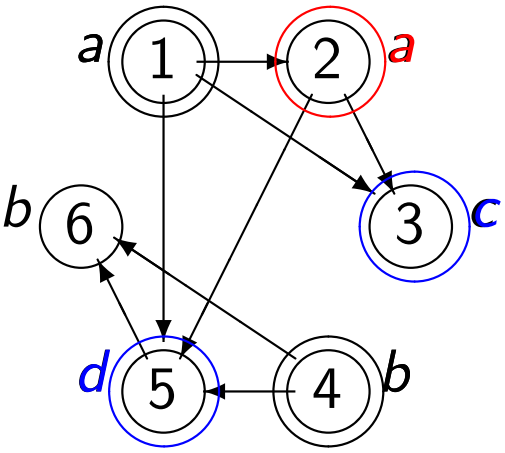
Different monoids may provide slightly restricted models of concurrency. But some of the related monoids may be more tractable for computation than others.

II: Dependence graphs for elts of trace monoids

We defined elts of trace, com- and step trace monoids to be equiv. classes of action seq. But **dependence graphs** describe elts graphically, help us to find normal forms. Mazurkiewicz defined dependence graphs for trace monoids, associating an acyclic labelled digraph on $\{1, \ldots, n\}$ to each sequence $w = a_1 \ldots a_n$.

Vertex i has label a_i, and for $i < j$, (i, j) is an edge $\iff$ $a_i = a_j$ or $a_i a_j \neq a_j a_i$. Equivalent sequences define isomorphic labelled digraphs.

e.g: Where $A = \{a, b, c, d\}$, $R = \{ab = ba,\ bc = cb,\ cd = dc\}$, the dependence graph of *aacbdb* is shown; it's also associated with sequences *aabcdb*, *abacdba*, with vertices in orders $1, 2, 4, 3, 5, 6$ and $1, 4, 2, 3, 5, 6$.

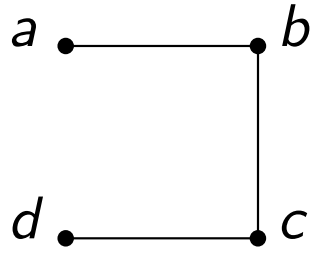


Left greedy algorithm uses dependence graph to find action sequence in normal form for associated trace. Max set of vertices with no **IN** arrows is selected and deleted, associated letters moved to left of word, in a preferred order, process repeated on remaining graph and word until graph is empty. Here first we select two vertices ringed black, then one red, two blue, one remaining vertex, to get *abacdb*.

Graphical rep. of relations, dep. graphs for products

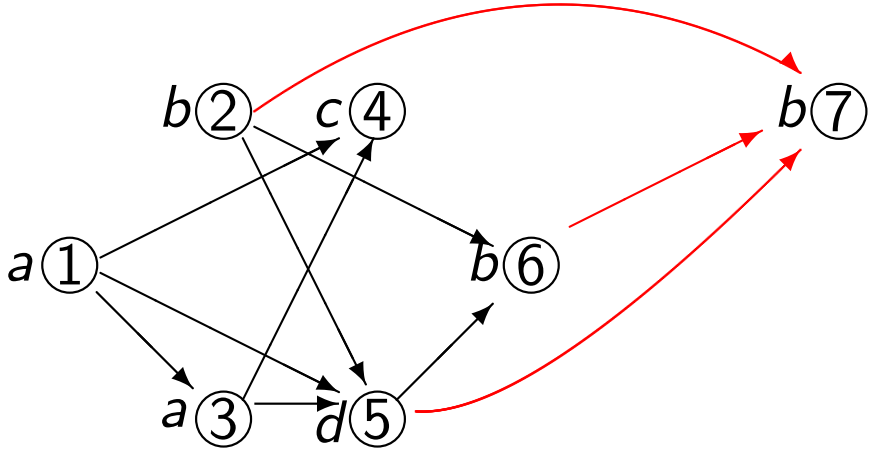
We can use a commutation graph to display the trace monoid relations.

eg $\left\langle \begin{matrix} a, b \\ c, d \end{matrix} \middle| \, , \; \begin{matrix} ab = ba,\ bc = cb \\ cd = dc \end{matrix} \right\rangle$ has commutation graph:



And we can join together the dependence graphs of monoid elements to get the dependence graph of their products. In this case it makes a lot of sense to order the vertices of a dependence graph roughly left to right.

Example: dependence graph for the product of *abacdb* and *b*. Edges between the two subgraphs are shown in red.



Dependence graphs for elts of com- and step trace monoids

We consider an element of a comtrace or step trace monoid to be an equivalence class $[w]$ of action sequences $w = \alpha_1 \dots \alpha_n$, $\alpha_i \in \mathbb{A}$. Modifying ideas of KoutnyEtAl, we define the dependence graph associated to w (and $[w]$) to be the acyclic labelled digraph with $\Sigma_i |\alpha_i|$ vertices, each with a label from A. For each $a, b \in A$, each $i \leq j$ with $a \in \alpha_i, b \in \alpha_j$, labelling vertices u, v resp., if (u, v) is an edge, it has one of four types:

type 1 $\bullet \longrightarrow \bullet$ when $i < j$ & $(a, b) \notin \mathrm{ser} \cup \mathrm{inl}$ (**precedence**),

type 2 $\bullet - - \rightarrow \bullet$ when $(a, b) \in \mathrm{ser} \setminus \mathrm{ser}^{-1}$ (**weak precedence**),

type 3 $\bullet \longrightarrow \bullet$ when $(a, b) \in \mathrm{inl}$.

type 4 $\bullet - - - \bullet$ when $i = j$ & $(a, b) \in \mathrm{sim} \setminus (\mathrm{ser} \cup \mathrm{ser}^{-1})$.

No edge when $(a, b) \in \mathrm{ser} \cap \mathrm{ser}^{-1}$,

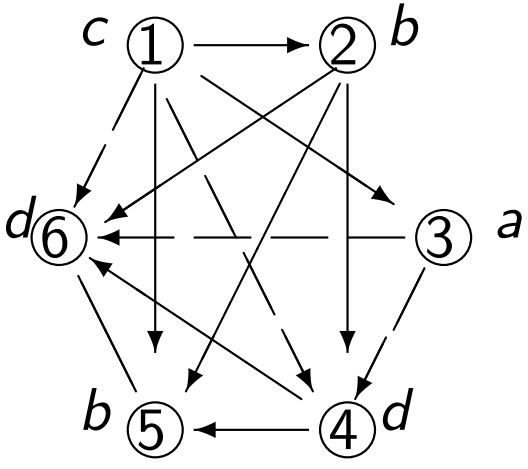
Edges of types 1,2 and 4 (but not 3) occur in comtraces. Types 1–4 can all occur in step trace monoids. NB: independence and hence non-edges are defined slightly differently here from their defs. in trace monoids.

Example for a comtrace monoid

In the comtrace monoid

$$M = \left\langle \begin{array}{l} a, b, c, d, \{a, b\}, \\ \{a, d\}, \{b, d\}, \{a, b, d\}, \{c, d\} \end{array} \middle| \begin{array}{l} ab = \{a, b\} = ba, \\ ad = \{a, d\}, cd = \{c, d\} \end{array} \right\rangle$$

the dependence graphs don't have type 3 edges. For $cbad\{b, d\}$ we have dependence graph:



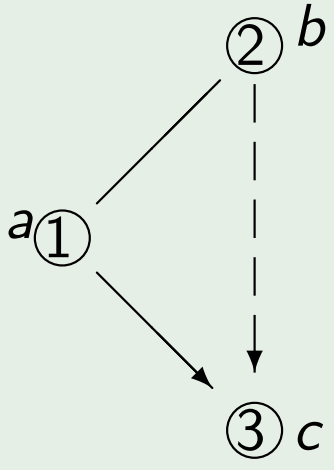
A left greedy algorithm finds an action sequence in normal form for the associated comtrace. The set of vertices for the first step of the normal form is selected as a maximal sim clique of vertices into which there is no directed incoming path containing a type 1 arrow (type 4 edges can be traversed in either direction). We select first c, then b, a to get the step $\{a, b\}$, then d, then b, d, and so get the comtrace $c\underline{\{}a, b\}d\{b, d\}$. This LG algorithm gives geodesic words for comtraces, but doesn't work for all step traces; in fact no LG algorithm for step traces can give a geodesic normal form.

A bad example in a step trace monoid

In a com- or step trace monoid a step trace is geodesic iff it contains a minimal number of steps. The LG algorithm described for comtraces chooses a leftmost step of maximal size.

The dependence graph of a step trace may contain type 3 edges, which must also be considered in the search for a leftmost step of maximal size. But however leftmost steps are found,there are examples of step trace monoids in which an LG algorithm cannot give a geodesic normal form.

Example



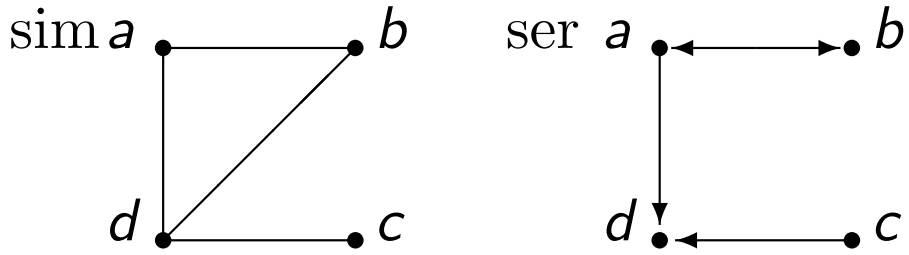
$M = \langle a, b, c, \{b, c\} \mid ab = ba, bc = \{b, c\} \rangle$

The set $\{abc, bac, a\{b, c\}\}$ is a step trace, whose dependence graph is shown, in which $a\{b, c\}$ is geodesic.
But an LG algorithm might select either a or b as a first step, and b is not the first step of a geodesic.

Graphical rep. of relations, dep. graphs for products

For com- and step trace monoids, we can display relations graphically.

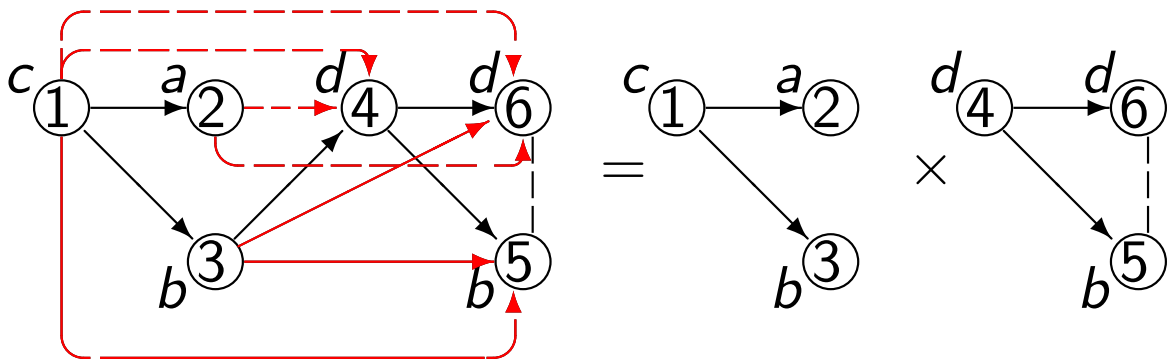
eg In our comtrace monoid M, we have:



In a step trace monoid, since $\mathrm{ser} \cap \mathrm{ser}^{-1} \cap \mathrm{inl} = \emptyset$, we can put ser & inl on the same graph, with inl edge undirected.

As in trace monoids we can also represent products graphically.

e.g. the dependence graph for $c\{a,b\}d\{b,d\}$, also found as the product of graphs for $c\{a,b\}$ and $d\{a,b\}$:



On left, red edges join the two factors. Dependence graphs can be big; we can save space by using Hasse diagrams (excluding edges implied by transitivity of the partial order associated with precedence).

Searching for normal forms for step traces: (1) abelian monoids

Our example of a step trace monoid for which the LG algorithm cannot find a geodesic normal form already suggests that step trace monoids must present more challenges than com trace monoids.

In particular,
let M be a commutative step trace monoid, for which any two actions $a, b \in A$ commute. Let $n := |A|$, and let w be a word of length $2n$ in which every element of A occurs twice. Then the problem of finding a normal form for w reduces to the graph clique problem, which is NP-hard in general (in terms of n).

However, we believe that, for a word written over the generators of an abelian submonoid $\langle B \rangle$, we can find a (close to) minimal length word over $\mathbb{B}$ in polynomial time.

Searching for normal forms for step traces: (2) a polynomial time reduction

Suppose we have a step trace system on an alphabet $\mathbb{A}$, and suppose that we are given a element w in the step trace monoid, together with a a sequence $s_1, \ldots, s_n$ of steps, and we know that w is equal to the product of $s_1, \ldots, s_n$ **in some order**. Can we find that order in polynomial time?

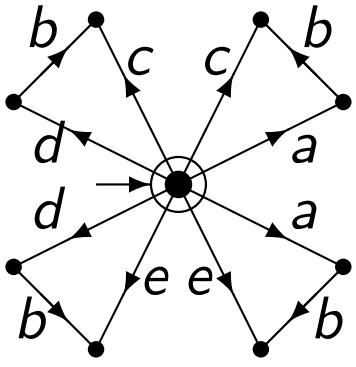
If so, then we can find a geodesic normal form for a step trace in polynomial time.

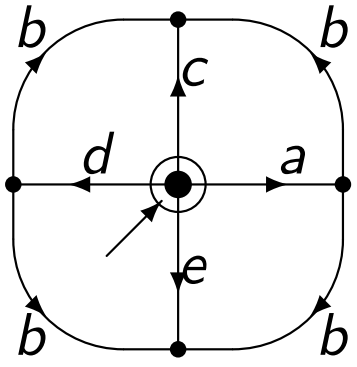
III: Stallings folding algorithm

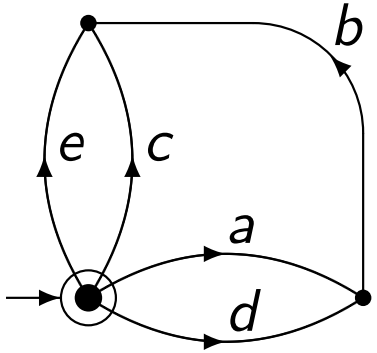
Given a set Y of words in $F(X)$ ($|X|, |Y| < \infty$), the algorithm constructs the **Stallings graph** $\mathcal{M}$ for the subgp $H = \langle Y \rangle$, by defn. the minimal reversible FSA over X accepting freely reduced w iff $w \in H$. The directed graph $\mathcal{M}$ has **start** vertex/state s_0, and accepts w if w labels a path from s_0 to s_0. Knowledge of $\mathcal{M}$ is effective for computation with H.

Algorithm (Stallings folding)

Initially: Directed edge-labelled **rose** graph $\mathcal{M}_0$, with $|Y|$ petals attached to a single vertex s_0, each petal a cycle labelled by an element of Y.
Fold by identifying any pair of directed edges with the same label that have common source or common target, until no more folding is possible.
Trim by deleting any edge not in a cycle. **Resulting graph** defines FSA $\mathcal{M}$.







Stallings folding: why it works, why it's useful

Stallings folding computes $\mathcal{M}$ as the unique minimal FSA with language the free reduction of $L(\mathcal{M}_0)$. By Myhill-Nerode(1957,8), the minimal FSA $\mathcal{M}'$ for the language $L(\mathcal{M}'_0)$ of an FSA $\mathcal{M}'_0$ is found by identifying all vertices of $\mathcal{M}'_0$ with the same **future**; the future of a vertex/state s is the set of words labelling paths from s to acceptance by $\mathcal{M}'_0$.
Since $\mathcal{M}_0$ is invertible, two vertices of $\mathcal{M}_0$ have (modulo free reduction) the same future $\iff$ they are identified by a sequence of foldings. **(*)**
And hence Stallings folding must terminate for any fg subgroup of a free group. But the statement (*) isn't valid for submonoids of the free monoid, when we don't have inverses and the corresponding FSA $\mathcal{M}_0$ is not invertible.; and (as in gp. case, not det.).
The Stallings graph of fg subgroups H_1, H_2 of a free group allows

- identification of **Nielsen** gens. as cycles (correp. max. subtree),
- soln. of membership and finiteness problems for H_i,
- answers to qns. about finite index, normality
- computation of Stallings graph for $H_1 \cap H_2$.

and more.

Kharlampovich et al. 2017, folding for automatic groups

For $G = \langle X \rangle$, $H = \langle Y \rangle \subseteq G$, $|X|, |Y| < \infty$, the **Schreier graph** $\mathrm{Schreier}_X(G, H)$ has the right cosets of H in G as vertices, directed edges (Hg, Hgx); a word w is in $H \iff w$ labels a circuit based at vtx H.
Let L be a regular language of representatives of G. The **Stallings graph** wrt L, $\mathrm{Stallings}_L(H)$, is the subgraph of $\mathrm{Schreier}(G, H)$ spanned by loops at H labelled by words in $L \cap H$. In general, $\mathrm{Stallings}_L(G)$ is not finite.

H is **L-quasi-convex** if $\exists k$ st , for $w \in L \cap H$, the path from 1 in $\mathrm{Cay}(G)$ labelled by w stays within distance k of H.

Theorem (KharlampovichEtAl2017, aka KMW)

Let G be automatic, with rational reps. L. Then $\exists$ partial algorithm that computes $\mathrm{Stallings}_L(H)$, halting iff H is L-quasi-convex.

NB: in particular this result applies to quasi-convex subgroups of hyperbolic groups and RAAGs.

KMW 2017, folding for automatic groups, (2)

The proof of KMW proof adapted Stallings' methods with a multi-stage procedure constructing a sequence $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_i, \ldots$.
Stage i runs as follows, where $\widehat{R}$ is cyclic closure of R.

Attach a loop labelled by r at v, for each $r \in \widehat{R} \cup \{xx^{-1} : x \in X\}$, each vertex v of $\mathcal{M}_{i-1}$, yielding $\mathcal{M}_i^0$

Fold $\mathcal{M}_i^0$ to get $\mathcal{M}_i$, by identifying any pair of directed edges in $\mathcal{M}_i^0$ with the same label that have common source or common target, until no more folding is possible.

All the FSA in this process are invertible, ie every edge can be traversed in two directions. So folding of $\mathcal{M}_i^0$ yields the unique deterministic minimal FSA accepting the same language.

Further modifications to deal with monoids and their submonoids

We (ABDFKR) use a construction modelled on KMW to extend that result to rational subsets of automatic groups that satisfy an asynchronous quasi-convexity property.

For rational subsets the algorithm always terminates in a finite no. of steps. But we don't yet know how to see when it has terminated,

For submonoids we can use the test of KMW to see that the algorithm has terminated. For RAAGs we have our own test.

We use the same basic construction, variants of both **attachment** and **folding**. But everything is complicated by absence of inverses. We need

- to be able to replace an FSA by another one accepting the same language up to free reduction.
- to adjust folding to replace our FSA with a deterministic FSA , in the absence of inverses
- a variant of L-quasiconvexity to ensure termination.

Termination in surface groups

Let $G := \pi_1(S_g)$ for a surface of genus g, which is a hyperbolic group, with a Dehn algorithm. Let L be the set of geodesics in G.

We have identified a set of constraints on a finite subset $T \subseteq G$ which are sufficient to ensure that the submonoid $< T >_+$ is L-quasiconvex, and hence that our algorithm terminates, given input T, to find an automaton recognising geodesic representatives of the elements of $< T >_+$.

Given a finite subset $U \subseteq G$, can we construct T satisfying those constraints such that $< T >_+ = < U >_+$? If so, we can test, for any finite $U \subseteq G$, whether $< U >_+$ is quasiconvex.

IV: A model based on intervals

Koutny et al. are developing a new model for concurrency, where actions (which take time) are represented as intervals rather than vertices (of a dependence graph).

Two different notions of precedence of actions must be considered:

strong precedence $a < b$, if a finishes before b starts,

weak precedence $a \sqsubset b$, if a starts before b starts, and a ends before b ends, but a does not finish before b starts (so $a \not< b$).

In this setup, finding an analogue of steps is tricky. So there's still a lot to do to interpret these models algebraically.