Automata theoretic techniques for computation in monoids

J. D. Mitchell

May 18, 2026

Finitely presented monoids

A ***monoid presentation*** is a pair $\langle A \mid R \rangle$ where $R \subseteq A^* \times A^*$ is a set of ***relations*** on A^*.

The monoid M defined by the presentation $\langle A \mid R \rangle$ is $A^*/R^\#$ where $R^\# \subseteq A^* \times A^*$ is the least congruence on A^* containing R.

A monoid M is a ***finitely presented monoid*** if it is defined by $\langle A \mid R \rangle$ and both A and R are finite.

Throughout the talk we will assume that $M = \langle A \mid R \rangle$ is a finite presentation.

Example:

Finitely presented monoids

A ***monoid presentation*** is a pair $\langle A \mid R \rangle$ where $R \subseteq A^* \times A^*$ is a set of ***relations*** on A^*.

The monoid M defined by the presentation $\langle A \mid R \rangle$ is $A^*/R^\#$ where $R^\# \subseteq A^* \times A^*$ is the least congruence on A^* containing R.

A monoid M is a ***finitely presented monoid*** if it is defined by $\langle A \mid R \rangle$ and both A and R are finite.

Throughout the talk we will assume that $M = \langle A \mid R \rangle$ is a finite presentation.

Example: $\langle a \mid a^{m+r} = a^m \rangle$

Finitely presented monoids

A ***monoid presentation*** is a pair $\langle A \mid R \rangle$ where $R \subseteq A^* \times A^*$ is a set of ***relations*** on A^*.

The monoid M defined by the presentation $\langle A \mid R \rangle$ is $A^*/R^\#$ where $R^\# \subseteq A^* \times A^*$ is the least congruence on A^* containing R.

A monoid M is a ***finitely presented monoid*** if it is defined by $\langle A \mid R \rangle$ and both A and R are finite.

Throughout the talk we will assume that $M = \langle A \mid R \rangle$ is a finite presentation.

Example: $\langle a, b \mid (ba)^2 b^2 ab = a, ab^2 aba = b \rangle$

Finitely presented monoids

A ***monoid presentation*** is a pair $\langle A \mid R \rangle$ where $R \subseteq A^* \times A^*$ is a set of ***relations*** on A^*.

The monoid M defined by the presentation $\langle A \mid R \rangle$ is $A^*/R^\#$ where $R^\# \subseteq A^* \times A^*$ is the least congruence on A^* containing R.

A monoid M is a ***finitely presented monoid*** if it is defined by $\langle A \mid R \rangle$ and both A and R are finite.

Throughout the talk we will assume that $M = \langle A \mid R \rangle$ is a finite presentation.

Example:

$$\begin{aligned}\langle a, b, c, A, B, C \mid & Aa = aA = Bb = bB = Cc = cC = 1,\\ & bbABaBcbCCAbaBBccBCbccBCb = 1,\\ & ccBCbCacAABcbCCaaCAcaaCAc = 1,\\ & aaCAcAbaBBCacAAbbABabbABa = 1\rangle\end{aligned}$$

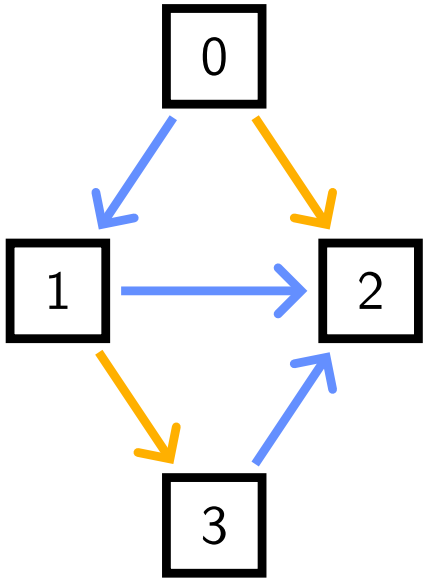
Aims

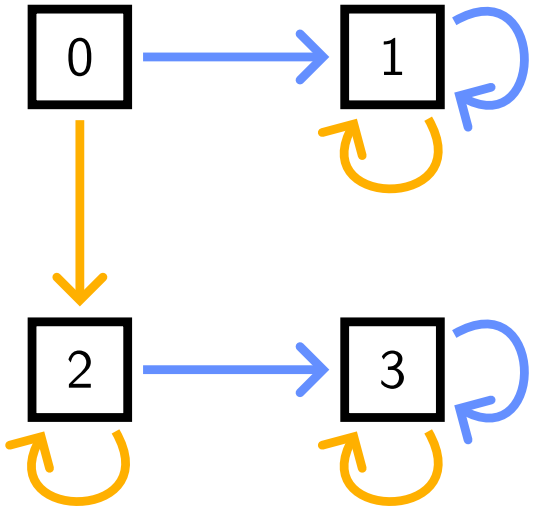
Given $M = \langle A|R\rangle$ can we:

- ✯ solve the word problem?
- ✯ determine finiteness?
- ✯ check for redundancy of relations in R?
- ✯ show that $M \not\cong \langle B \mid S\rangle$?
- ✯ determine the one-sided or two-sided congruences of M?
- ✯ ...

Word Graphs

Let A be an alphabet. A graph $\Gamma = (N, E)$ with a finite set of nodes $N \subseteq \mathbb{N}$ where $0 \in N$ and edges $E \subseteq N \times A \times N$ is called a ***word graph***.





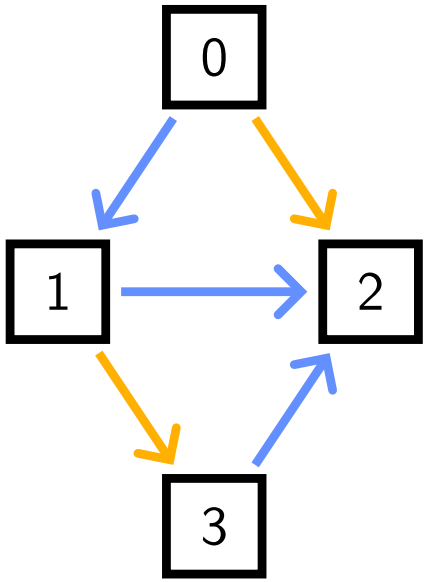
blue $= b$, yellow $= y$

A word graph is essentially just an automata without any starting or accept states.

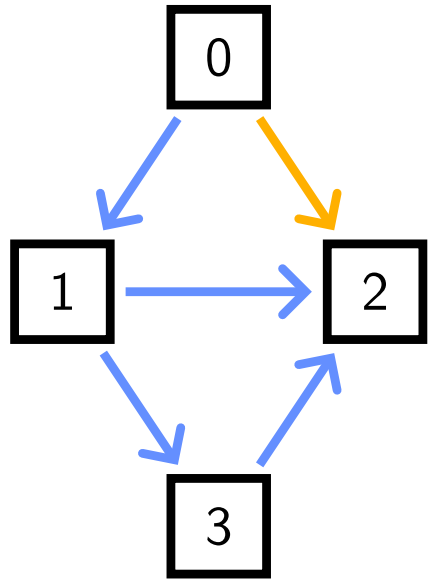
Example: the (right) Cayley graph of any monoid.

Determinism

A word graph $\Gamma = (N, E)$ is called ***deterministic*** if for every node $\alpha \in N$ and every letter $a \in A$ there exists at most one edge of the form $(\alpha, a, \beta) \in E$.



deterministic

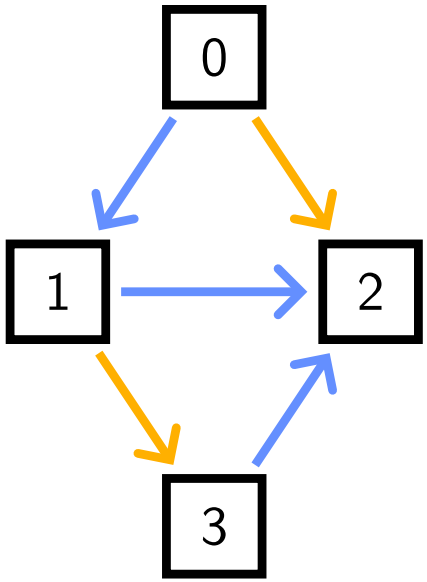


non-deterministic

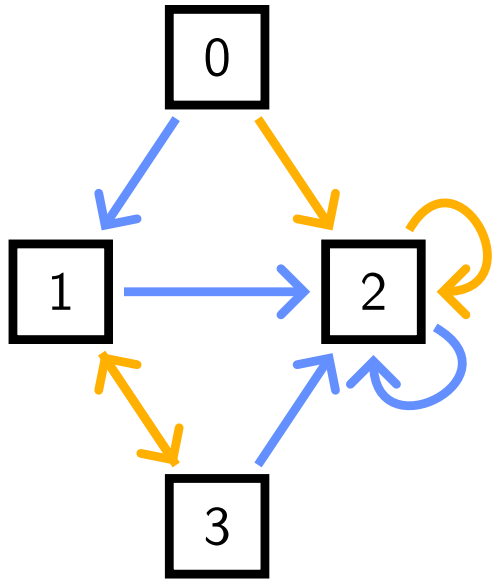
blue $= b$, yellow $= y$

Completeness

A word graph $\Gamma = (N, E)$ is ***complete*** if for every node $\alpha \in N$ and every letter $a \in A$ there exists some node $\beta \in \Gamma$ such that $(\alpha, a, \beta) \in E$.



not complete



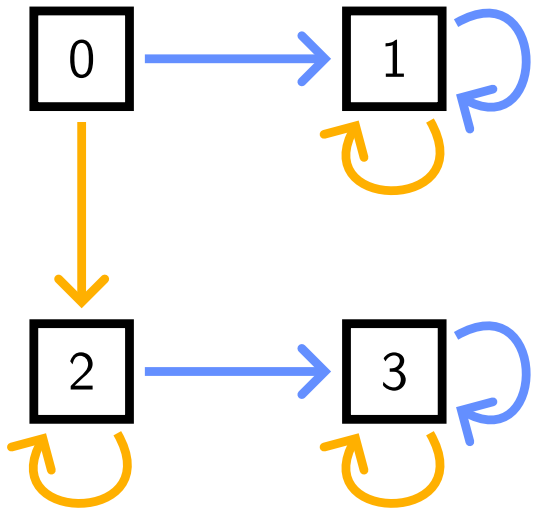
complete

blue $= b$, yellow $= y$

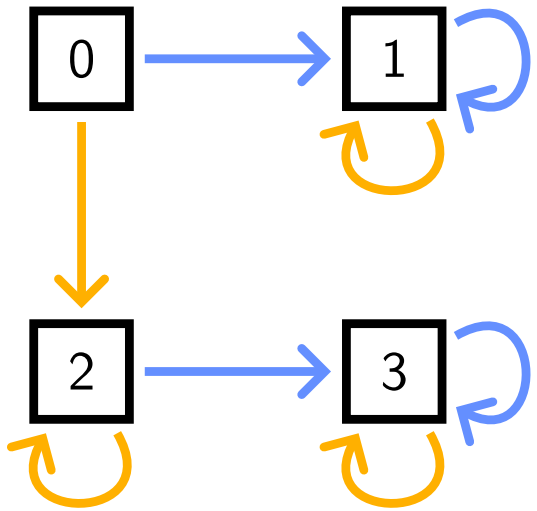
Compatible

A word graph $\Gamma = (N, E)$ is ***compatible*** with $R \subseteq A^* \times A^*$ if Γ is deterministic and for every node $\alpha \in N$ and for every $(u, v) \in R$ there exists a node $\beta \in N$ such that both u and v label α-β paths in Γ.

blue $= b$, yellow $= y$



compatible with $(by)^2 = b$



not compatible with $by = yb$

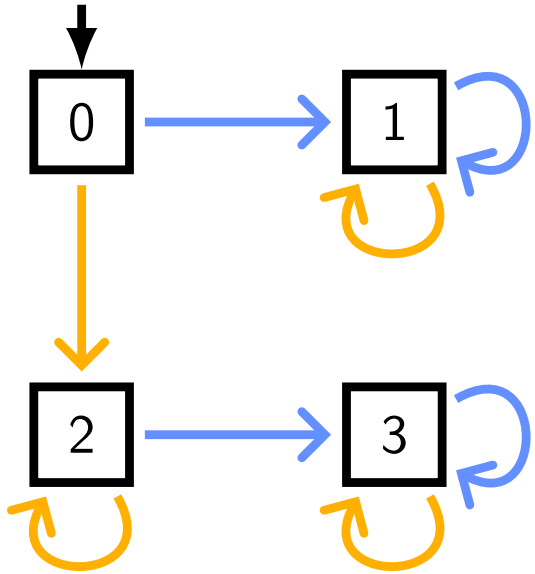
Standard, 1/2

Let $\Gamma = (N, E)$ be a word graph over A. For every $\alpha \in N$, let $w_\alpha \in A^*$ denote the lenlex least word labelling a 0-α path in Γ. Then $\Gamma = (N, E)$ is ***standard*** if:

- $N = \{0, \ldots, |N| - 1\}$;
- Γ is complete;
- Γ is deterministic;
- every node is reachable from $0 \in N$;
- $\alpha < \beta$ if and only if $w_\alpha < w_\beta$.

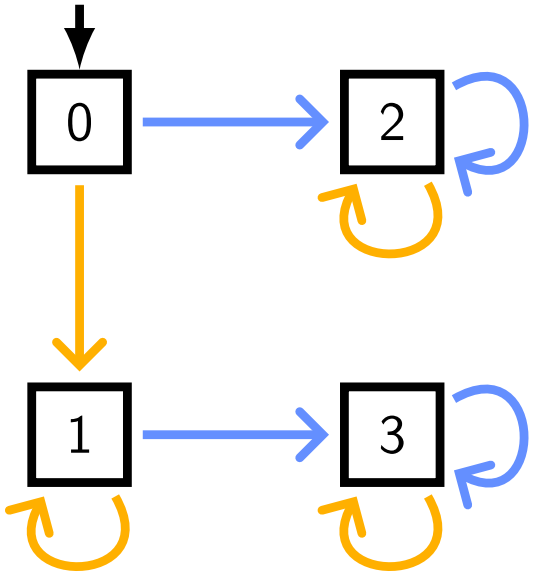
Standard, 2/2

blue $= b$, yellow $= y$, and $b < y$



standard

$w_0 = \varepsilon < w_1 = b < w_2 = y < w_3 = yb$



not standard

$w_0 = \varepsilon < w_1 = y \not\leq w_2 = b$

Word graphs and right congruences

Theorem

There is a one-to-one correspondence between the right congruences of $M = \langle A \mid R \rangle$ and the standard word graphs over A compatible with R.

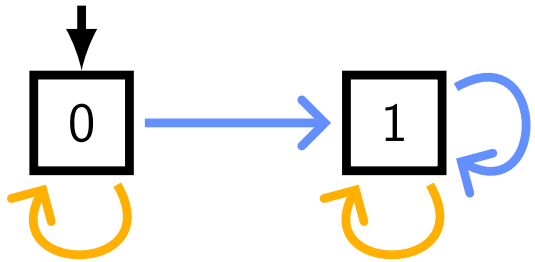
Word graphs and right congruences

Theorem

There is a one-to-one correspondence between the right congruences of $M = \langle A \mid R \rangle$ and the standard word graphs over A compatible with R.

If $M = \langle b, y \mid b^3 = b, y^2 = y, (yb)^2 = b \rangle = \{1, b, y, yb\}$, then

blue $= b$, yellow $= y$

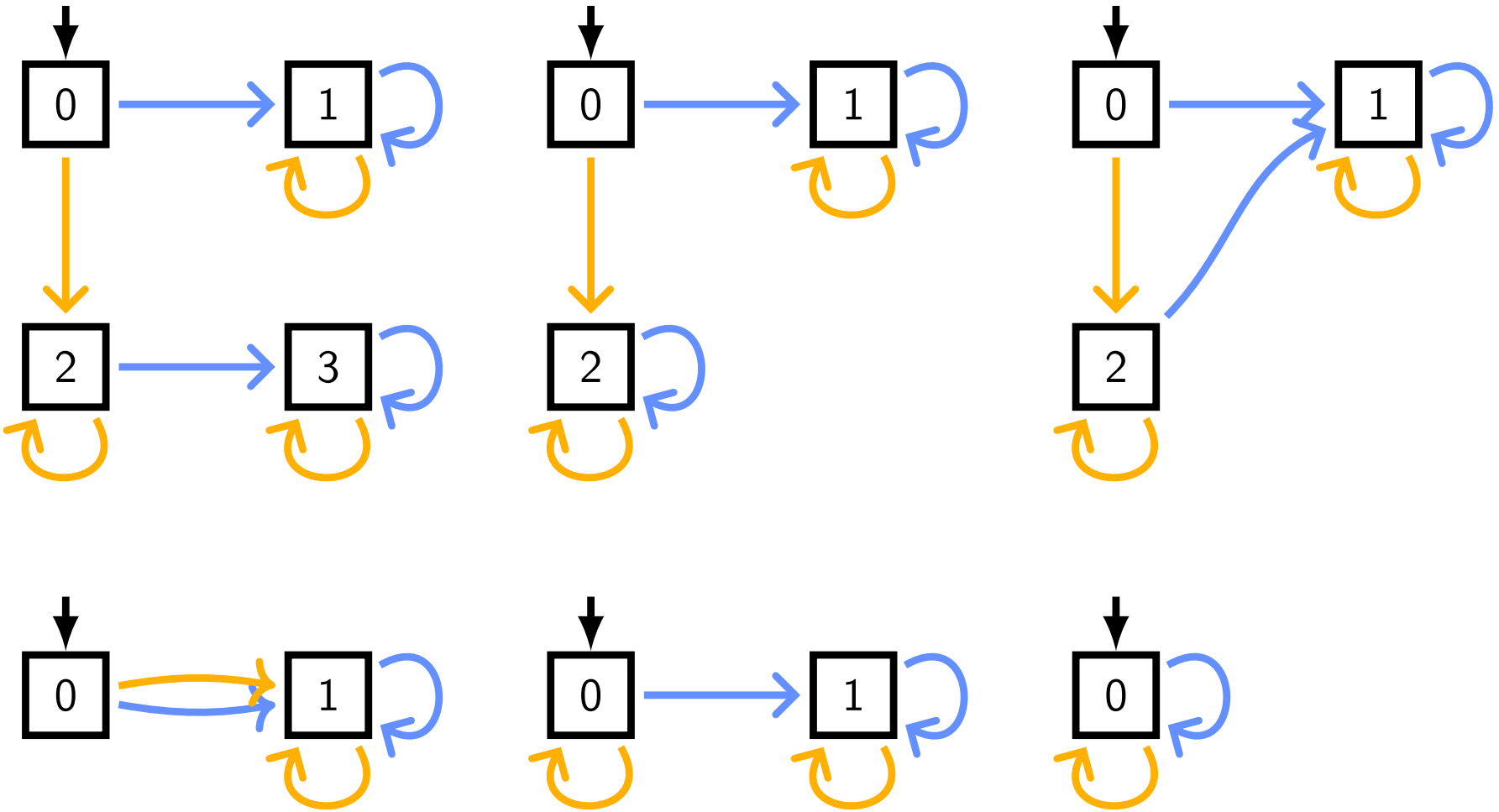


is standard, and so represents the right congruence:

$$\{\{1, y\}, \{b, yb\}\}.$$

The right congruences of
$M = \langle b, y \mid b^3 = b, y^2 = y, (yb)^2 = b \rangle = \{1, b, y, yb\}$ are:

blue = b, yellow = y



Todd-Coxeter for monoids

What is?

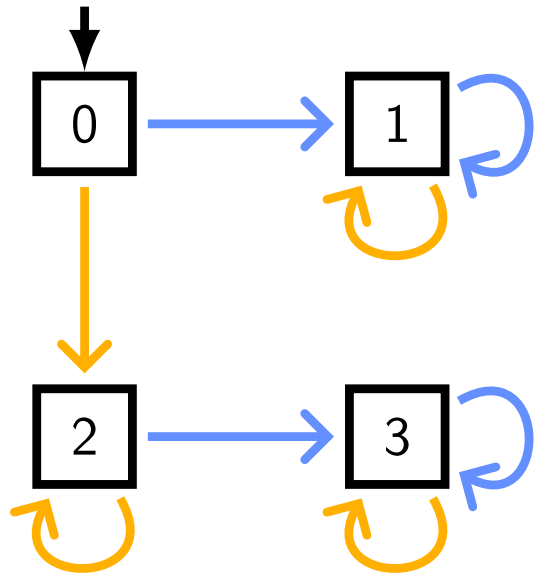
Aim: Find the (right) Cayley graph $\Gamma = (N, E)$ of the monoid $M = \langle A|R\rangle$.

Starts with the word graph $\Gamma = (N := \{0\}, E := \varnothing)$.

At every step Γ is some sort of "approximation" of the right Cayley graph.

There are three moves you can make at any point.

$$\langle b, y \mid b^3 = b, y^2 = y, (yb)^2 = b\rangle \longrightarrow$$



blue $= b$, yellow $= y$

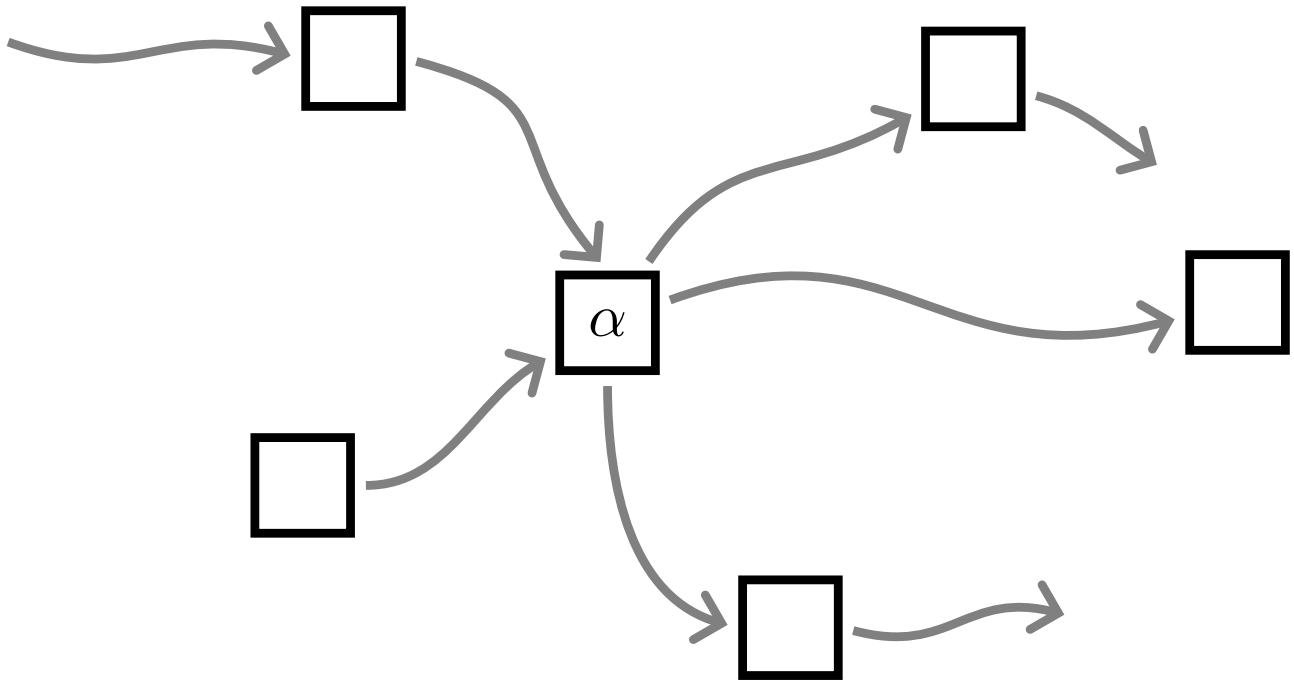
Todd-Coxeter for monoids

Move one — define a new node and an edge to it

If $\alpha \in N$ and $a \in A$ but there's no edge labelled a with source α, then

$$\Gamma \leftarrow (N \cup \{\beta\}, E \cup (\alpha, a, \beta))$$

for some $\beta \notin N$.



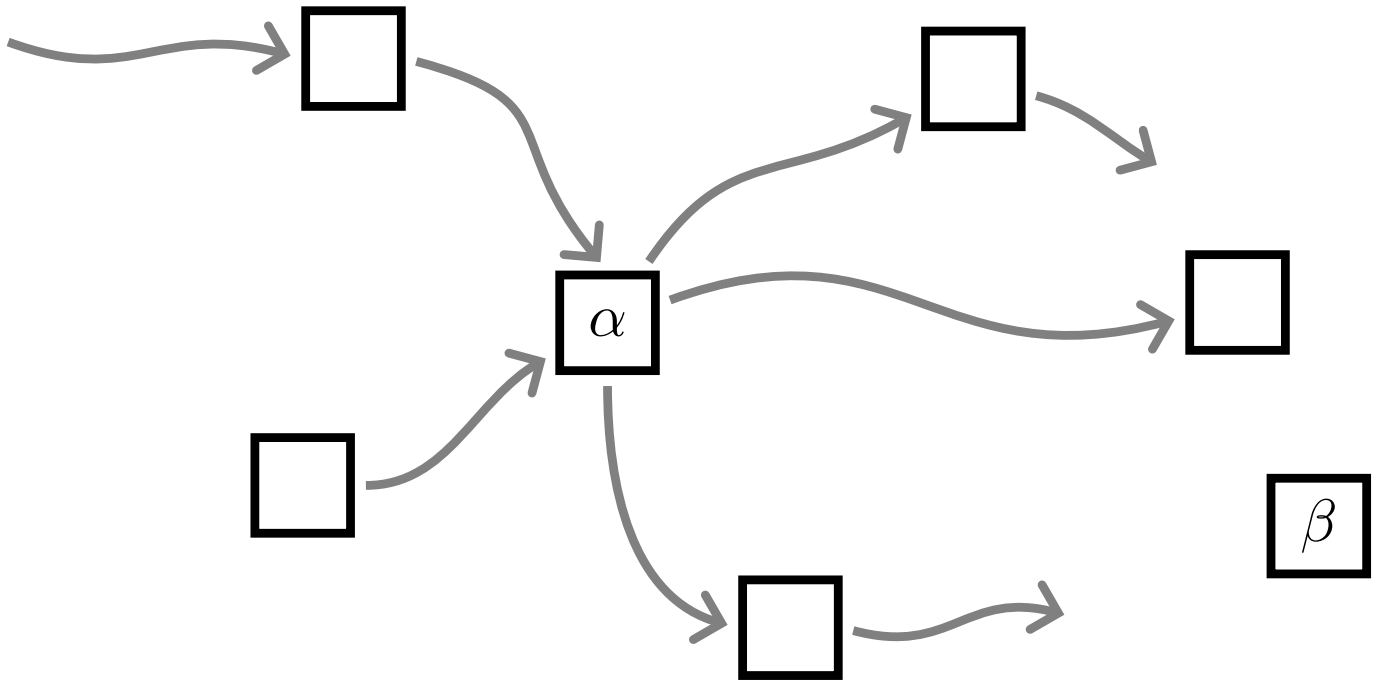
Todd-Coxeter for monoids

Move one — define a new node and an edge to it

If $\alpha \in N$ and $a \in A$ but there's no edge labelled a with source α, then

$$\Gamma \leftarrow (N \cup \{\beta\}, E \cup (\alpha, a, \beta))$$

for some $\beta \notin N$.



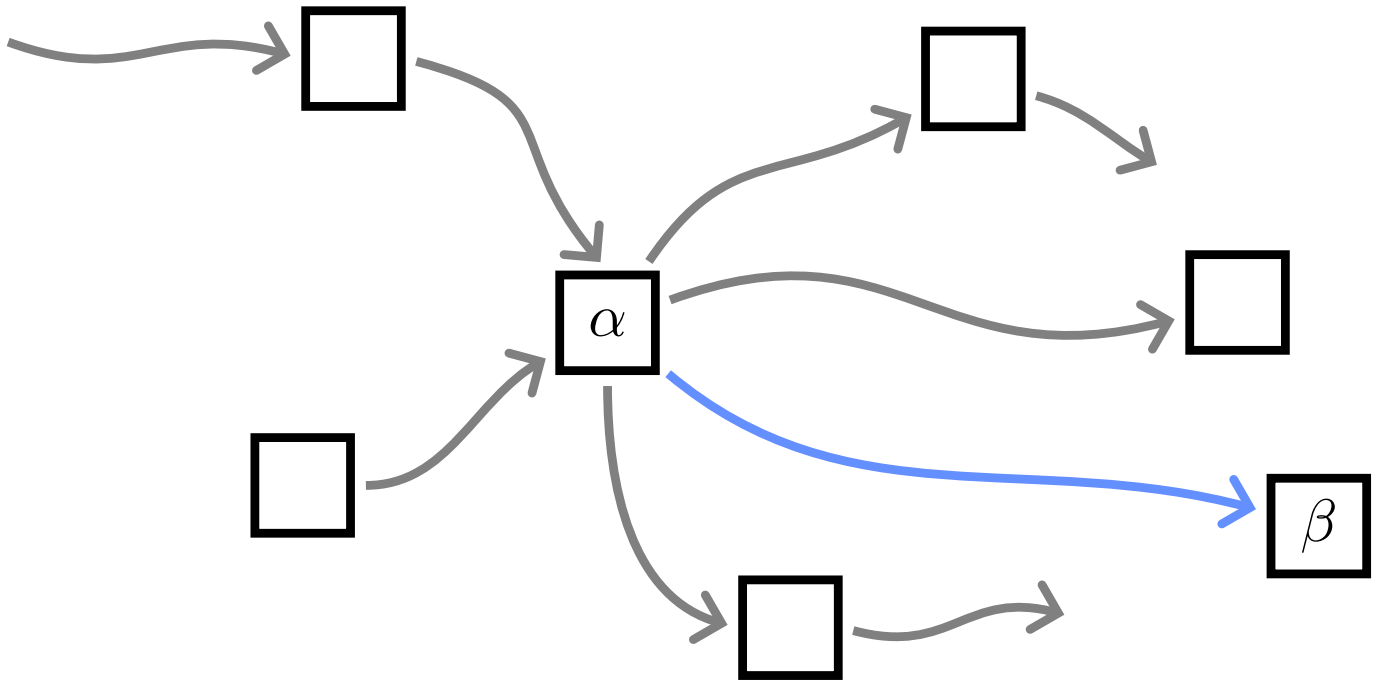
Todd-Coxeter for monoids

Move one — define a new node and an edge to it

If $\alpha \in N$ and $a \in A$ but there's no edge labelled a with source α, then

$$\Gamma \leftarrow (N \cup \{\beta\}, E \cup (\alpha, a, \beta))$$

for some $\beta \notin N$.

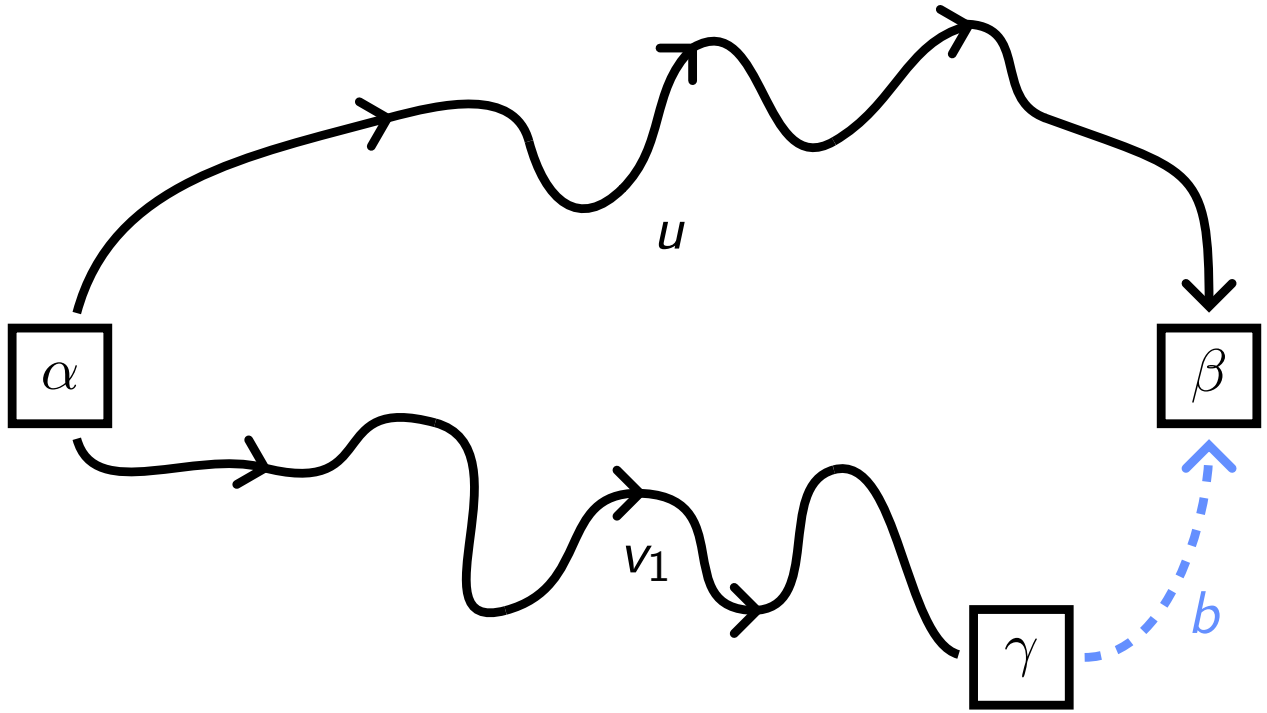


Todd-Coxeter for monoids

Move two — follow the paths labelled by a relation

Suppose that $\alpha \in N$ and $(u, v_1 b) \in R$ where:

- ✯ $b \in A$;
- ✯ $u \in A^*$ labels an α-β path; and
- ✯ $v_1 \in A^*$ labels an α-γ path.



Then $\Gamma \leftarrow (N, E \cup \{(\gamma, b, \beta)\})$.

Todd-Coxeter for monoids

Move three — determinize

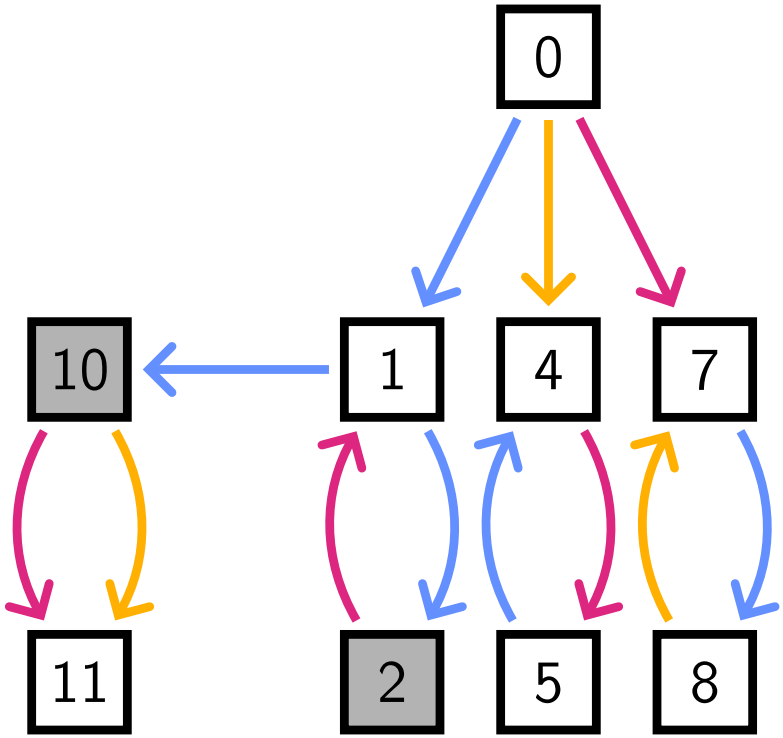
For all $(\alpha, a, \beta), (\alpha, a, \gamma) \in E$ where $\beta < \gamma$, do:

- ✯ for every edge (δ, b, γ) add (δ, b, β) and remove (δ, b, γ);
- ✯ for every edge (γ, c, ζ) add (β, c, ζ) and remove (γ, c, ζ);
- ✯ remove the node γ;

while Γ is not deterministic.

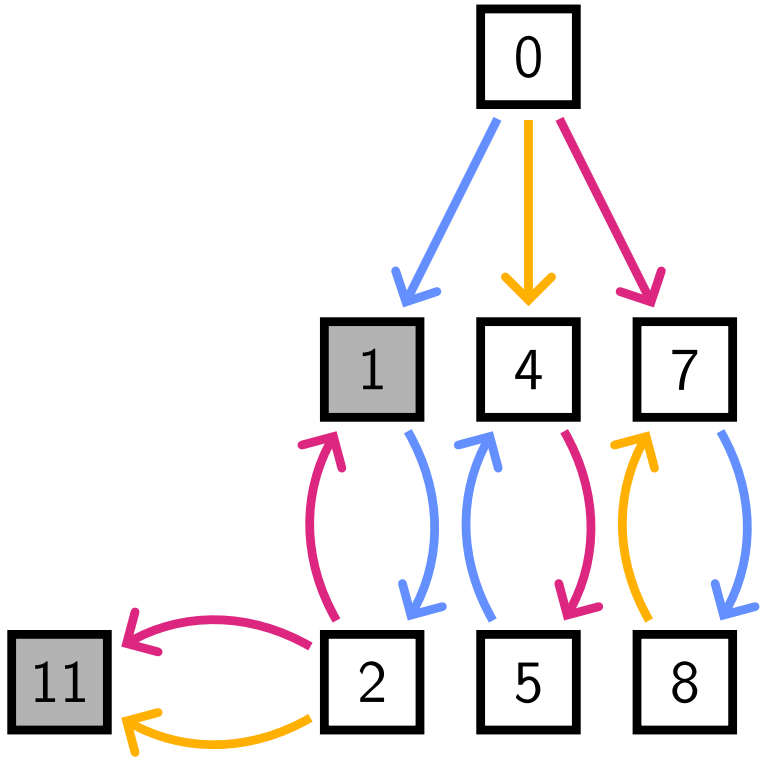
Todd-Coxeter for monoids

Move three — determinize



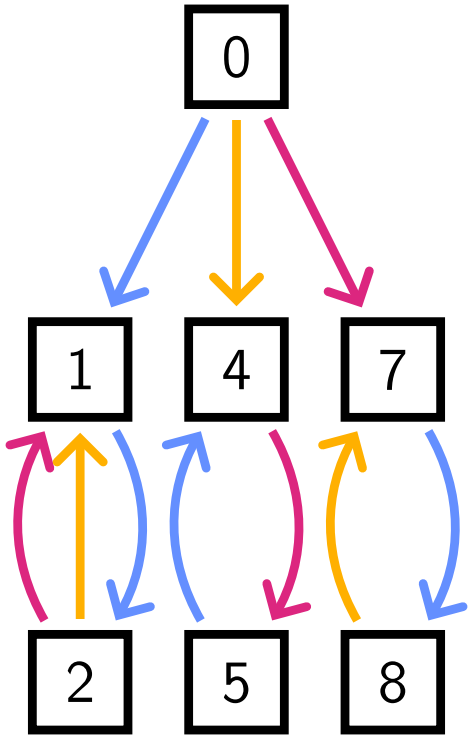
Todd-Coxeter for monoids

Move three — determinize



Todd-Coxeter for monoids

Move three — determinize



Todd-Coxeter is not a single algorithm, or an algorithm at all

It's any sequence of the previous three steps:

- ✯ define a new node;
- ✯ follow the paths labelled by a relation;
- ✯ determinize;

subject to some fairly weak assumptions.

There are infinitely many such procedures, which we refer to as ***congruence enumerations***.

Todd-Coxeter is not a single algorithm, or an algorithm at all

It's any sequence of the previous three steps:

- ✯ define a new node;
- ✯ follow the paths labelled by a relation;
- ✯ determinize;

subject to some fairly weak assumptions.

There are infinitely many such procedures, which we refer to as ***congruence enumerations***.

A congruence enumeration ***terminates*** if the word graph Γ is complete deterministic and compatible with R.

Todd-Coxeter is not a single algorithm, or an algorithm at all

It's any sequence of the previous three steps:

- ✯ define a new node;
- ✯ follow the paths labelled by a relation;
- ✯ determinize;

subject to some fairly weak assumptions.

There are infinitely many such procedures, which we refer to as ***congruence enumerations***.

A congruence enumeration ***terminates*** if the word graph Γ is complete deterministic and compatible with R.

Theorem

A congruence enumeration terminates for $M = \langle A|R\rangle$ if and only if M is finite. If a congruence enumeration terminates, the word graph is the right Cayley graph of M with respect to A.

Todd-Coxeter is not a single algorithm, or an algorithm at all

It's any sequence of the previous three steps:

- ✯ define a new node;
- ✯ follow the paths labelled by a relation;
- ✯ determinize;

subject to some fairly weak assumptions.

There are infinitely many such procedures, which we refer to as ***congruence enumerations***.

A congruence enumeration ***terminates*** if the word graph Γ is complete deterministic and compatible with R.

Theorem

A congruence enumeration terminates for $M = \langle A|R\rangle$ if and only if M is finite. If a congruence enumeration terminates, the word graph is the right Cayley graph of M with respect to A.

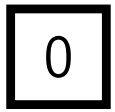
There are two main strategies for performing congruence enumerations.

HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$



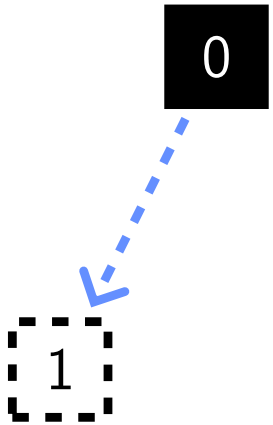
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid \underline{byr = b}, yrb = y, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 1 $(0, b)$:



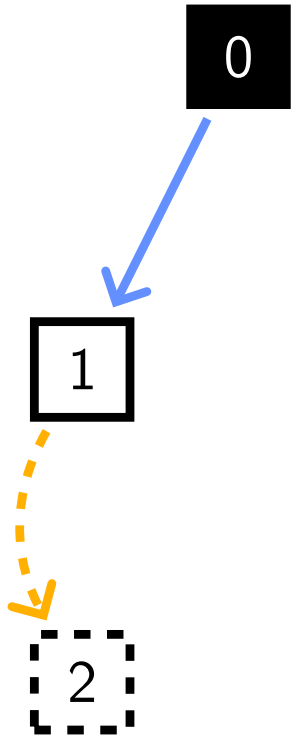
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid \underline{byr = b}, yrb = y, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 1 $(1, y)$:



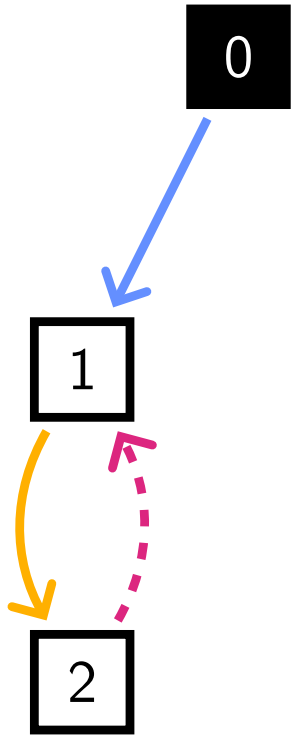
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid \underline{byr = b}, yrb = y, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 2 ($byr = b, 0$):



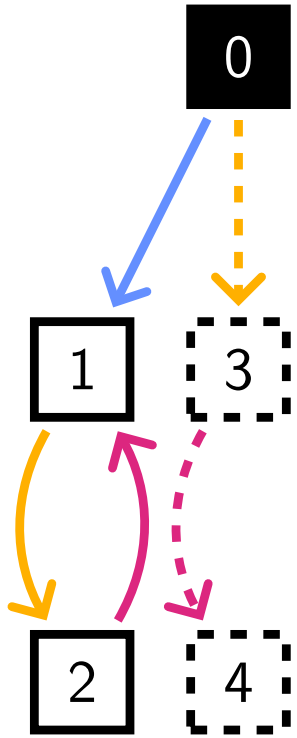
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, \underline{yrb = y}, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 1 $(y, 0) + (r, 3)$:



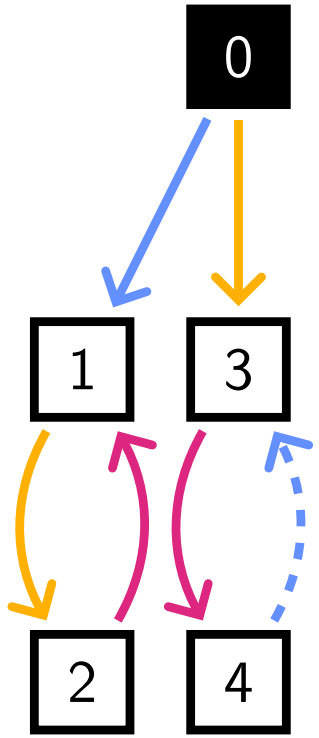
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, \underline{yrb = y}, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 2 $(yrb = y, 0)$:



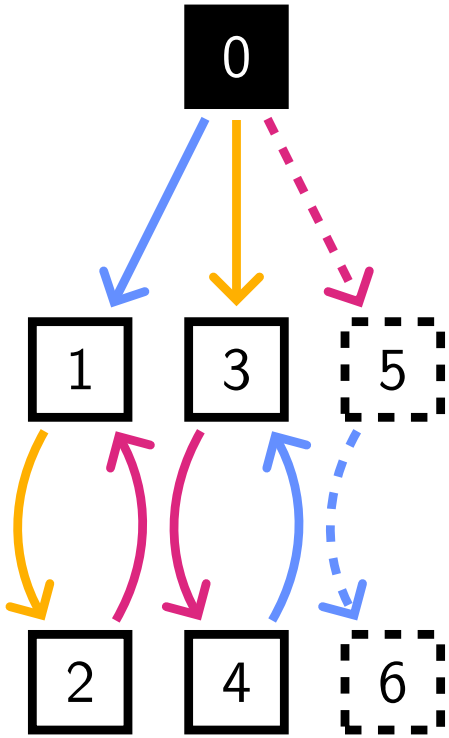
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, \underline{rby = r} \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 1 $(r, 0) + (b, 5)$:



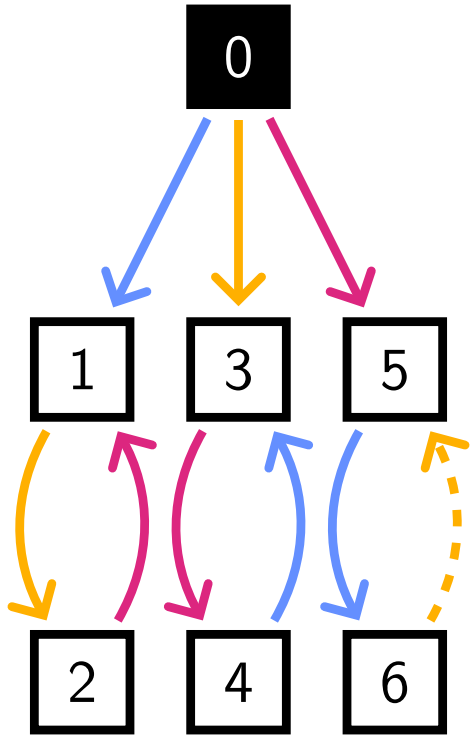
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, \underline{rby = r} \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 2 $(rby = r, 0)$:



HLT strategy

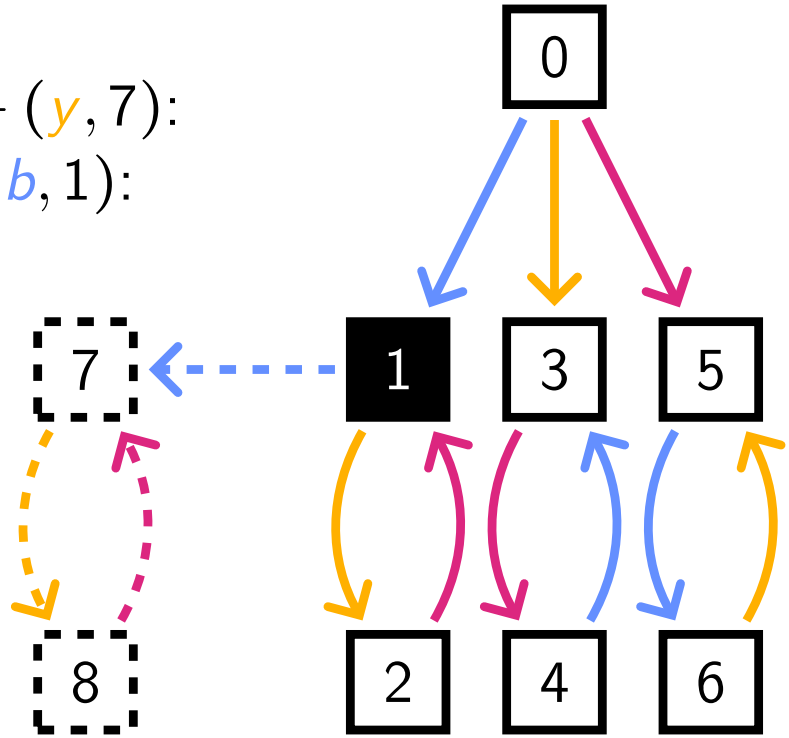
Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid \underline{byr = b}, yrb = y, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 1 $(b, 1) + (y, 7)$:

Move 2 $(byr = b, 1)$:



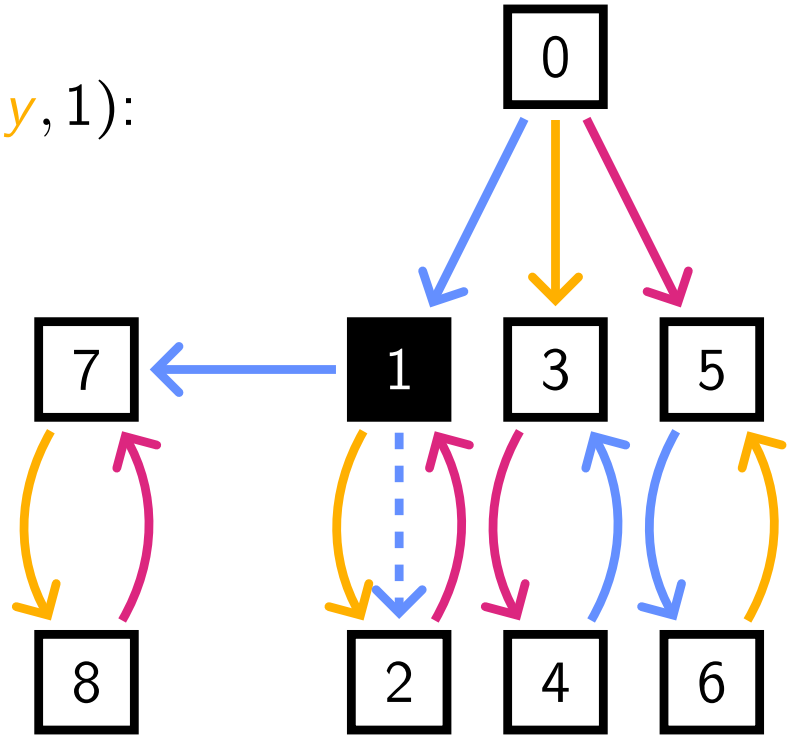
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, \underline{yrb = y}, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 2 $(yrb = y, 1)$:



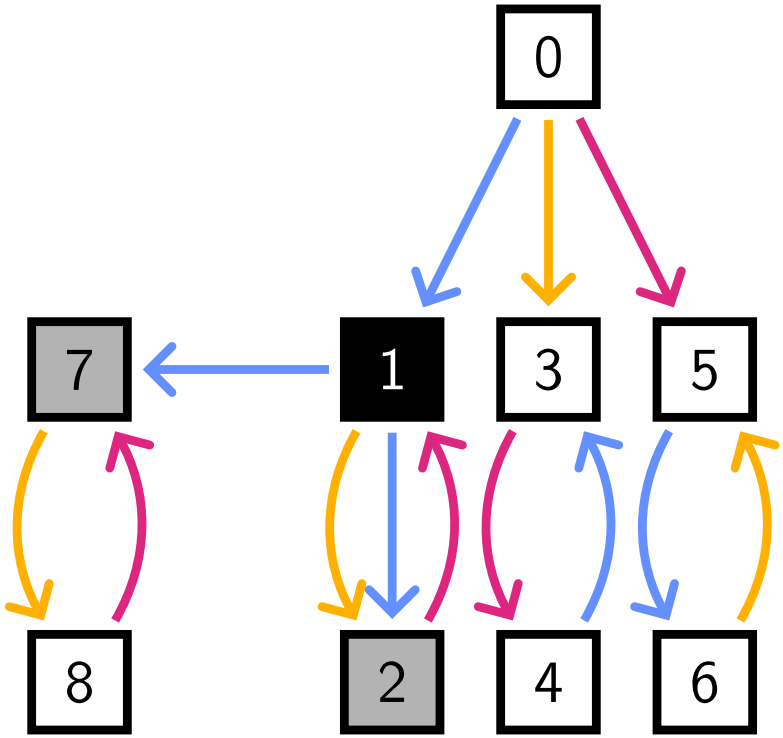
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, \underline{yrb = y}, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 3 $(2, 7)$:



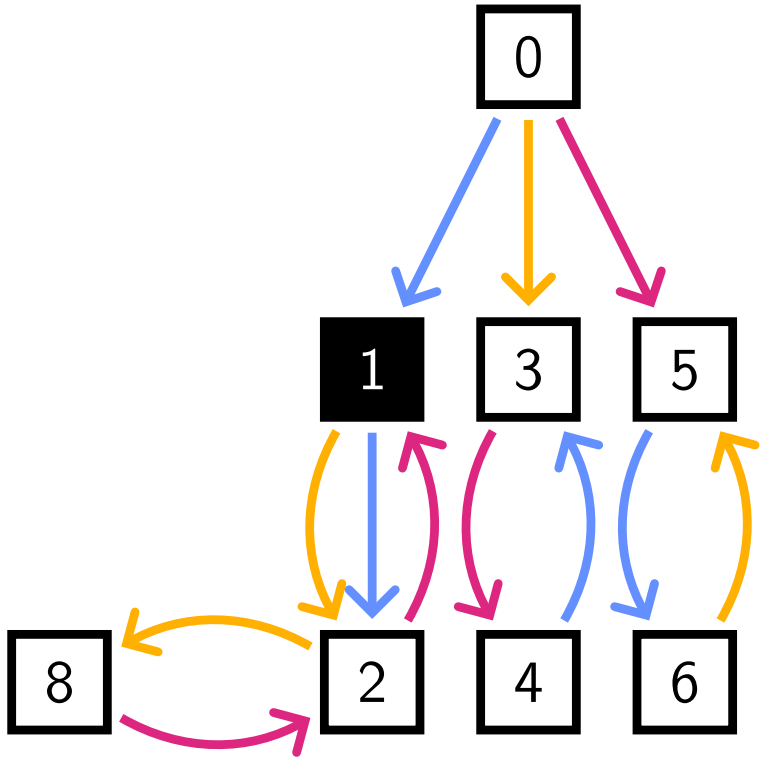
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, \underline{yrb = y}, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 3 $(2, 7)$:



HLT strategy

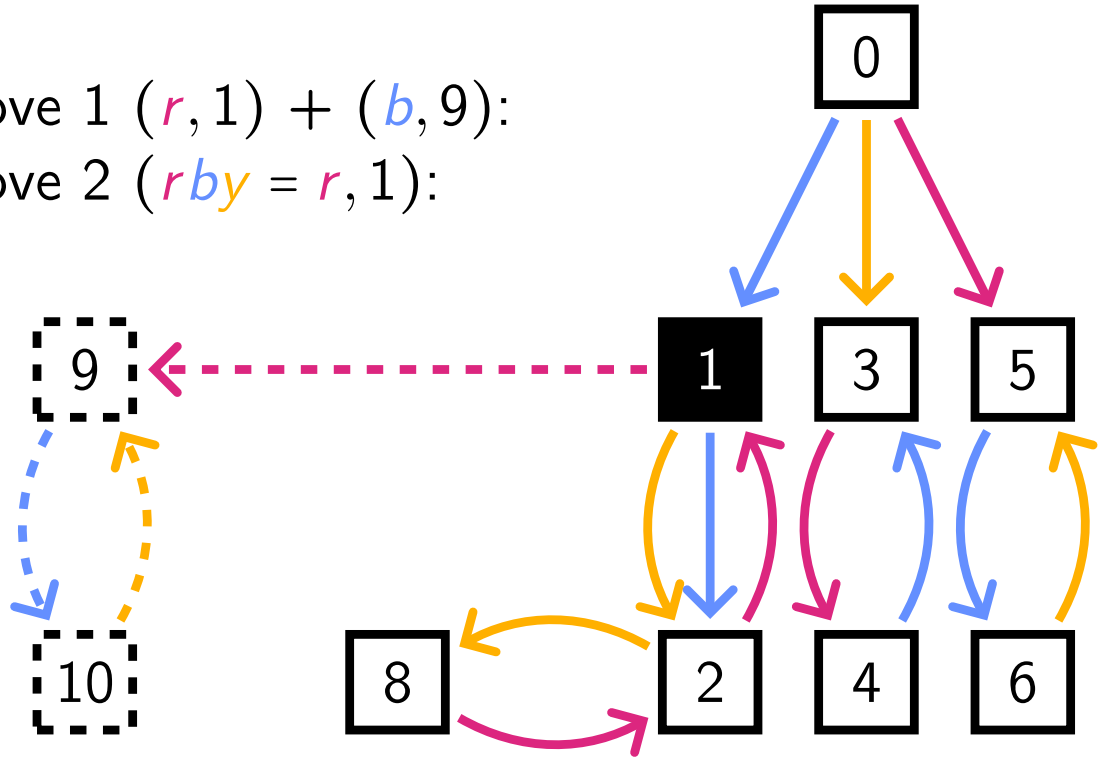
Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, \underline{rby = r} \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 1 $(r, 1) + (b, 9)$:

Move 2 $(rby = r, 1)$:



HLT strategy

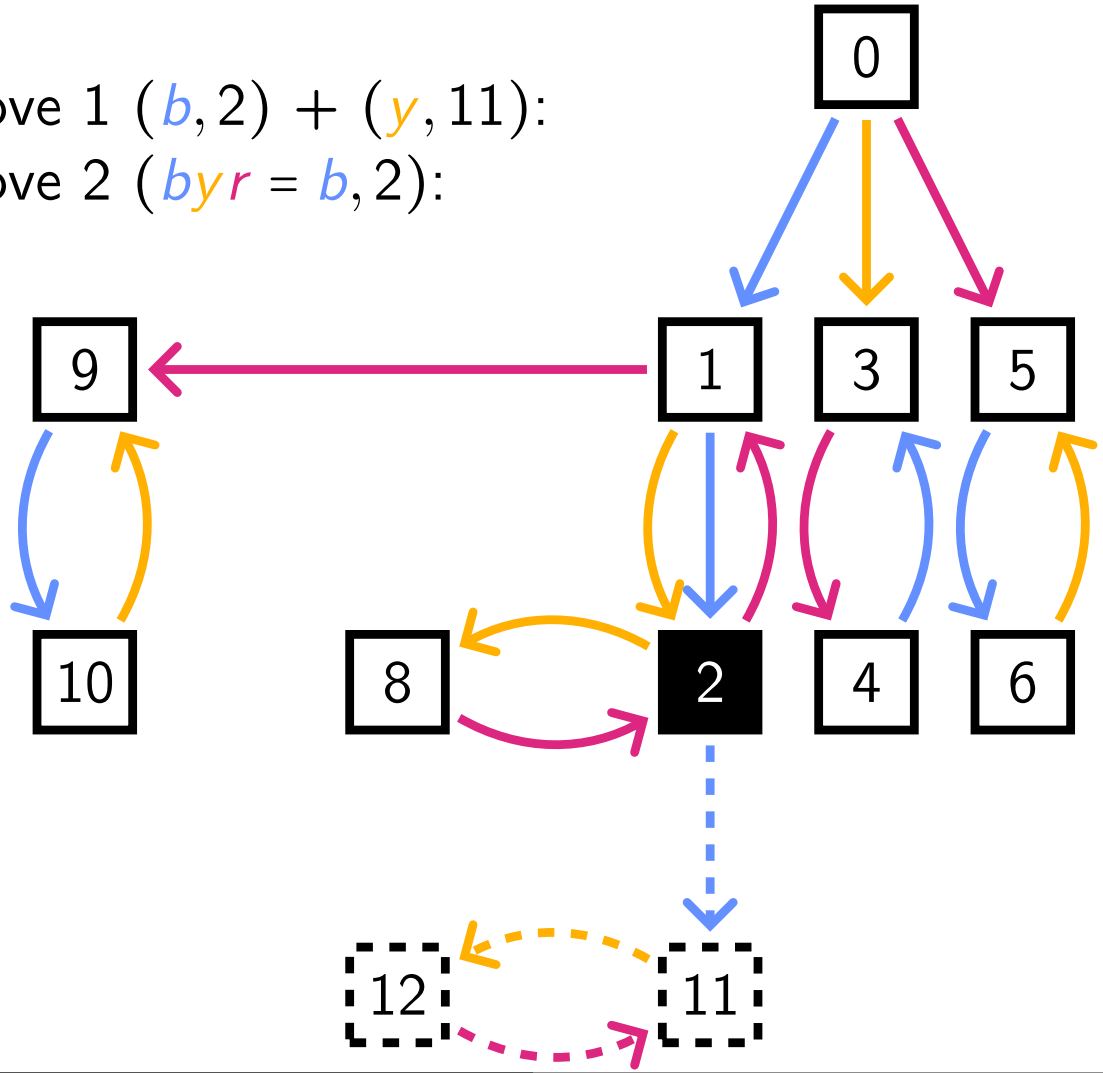
Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid \underline{byr = b}, yrb = y, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 1 $(b, 2) + (y, 11)$:

Move 2 $(byr = b, 2)$:



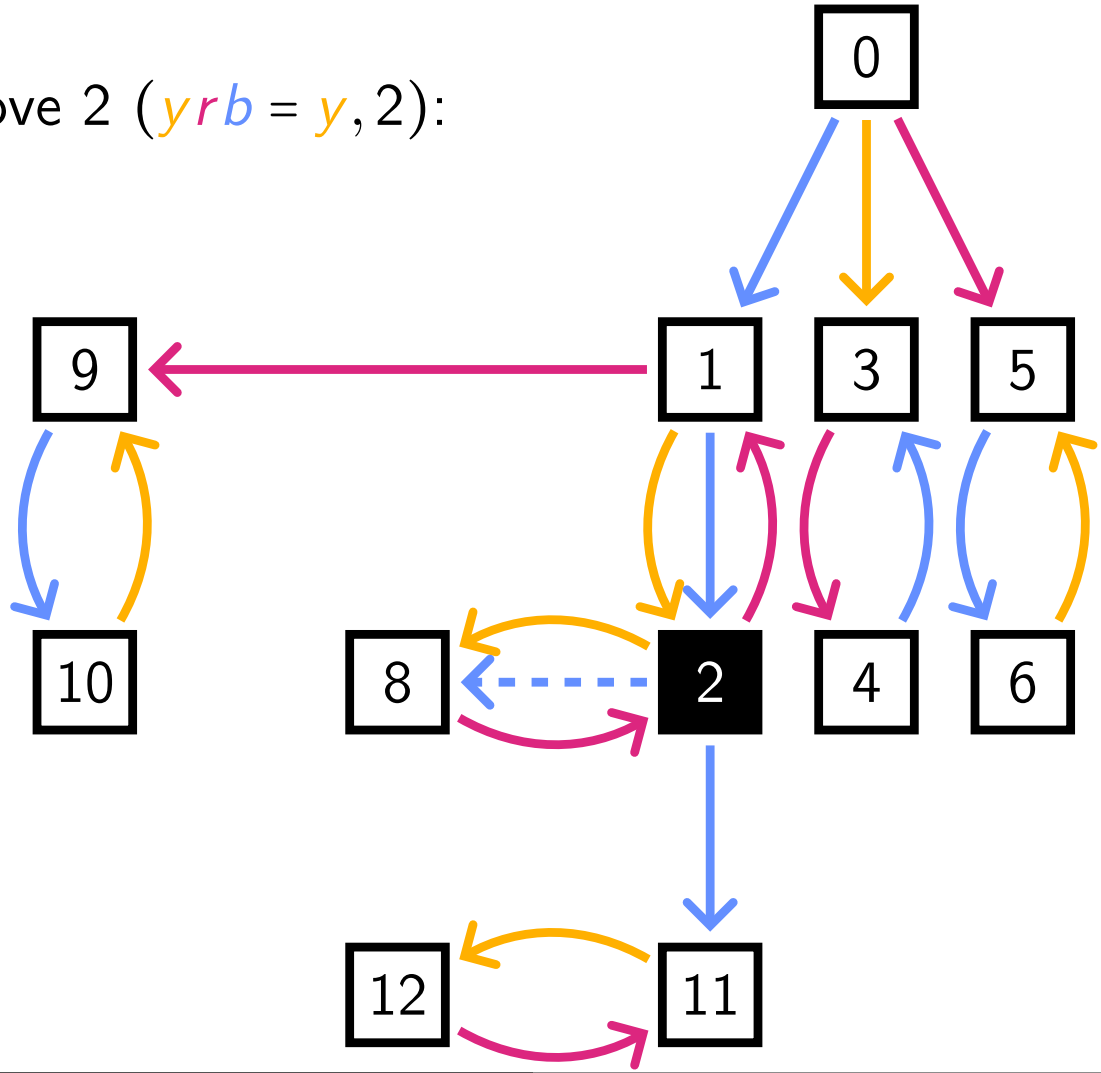
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, \underline{yrb = y}, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 2 ($yrb = y, 2$):



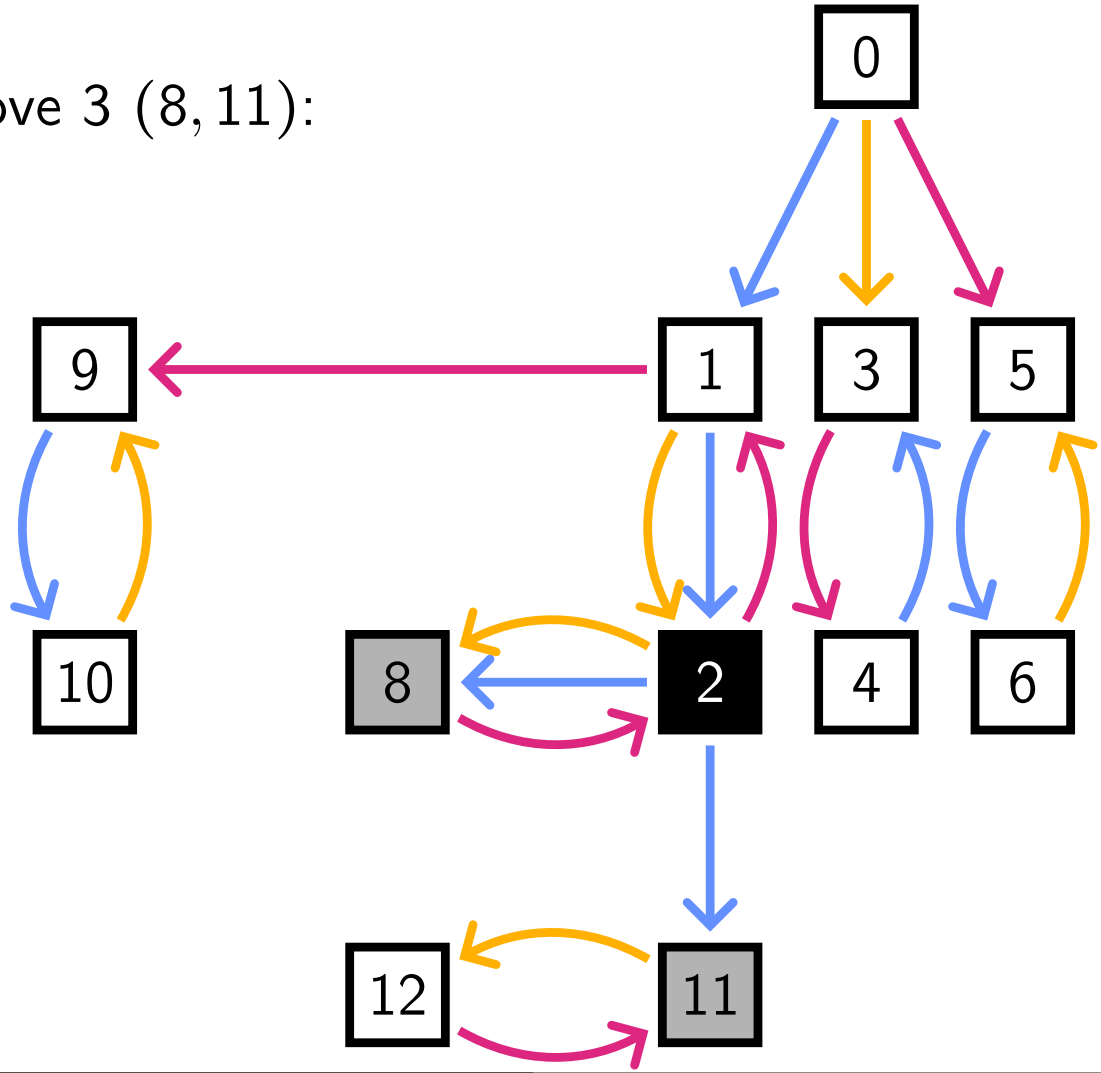
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, \underline{yrb = y}, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 3 $(8, 11)$:



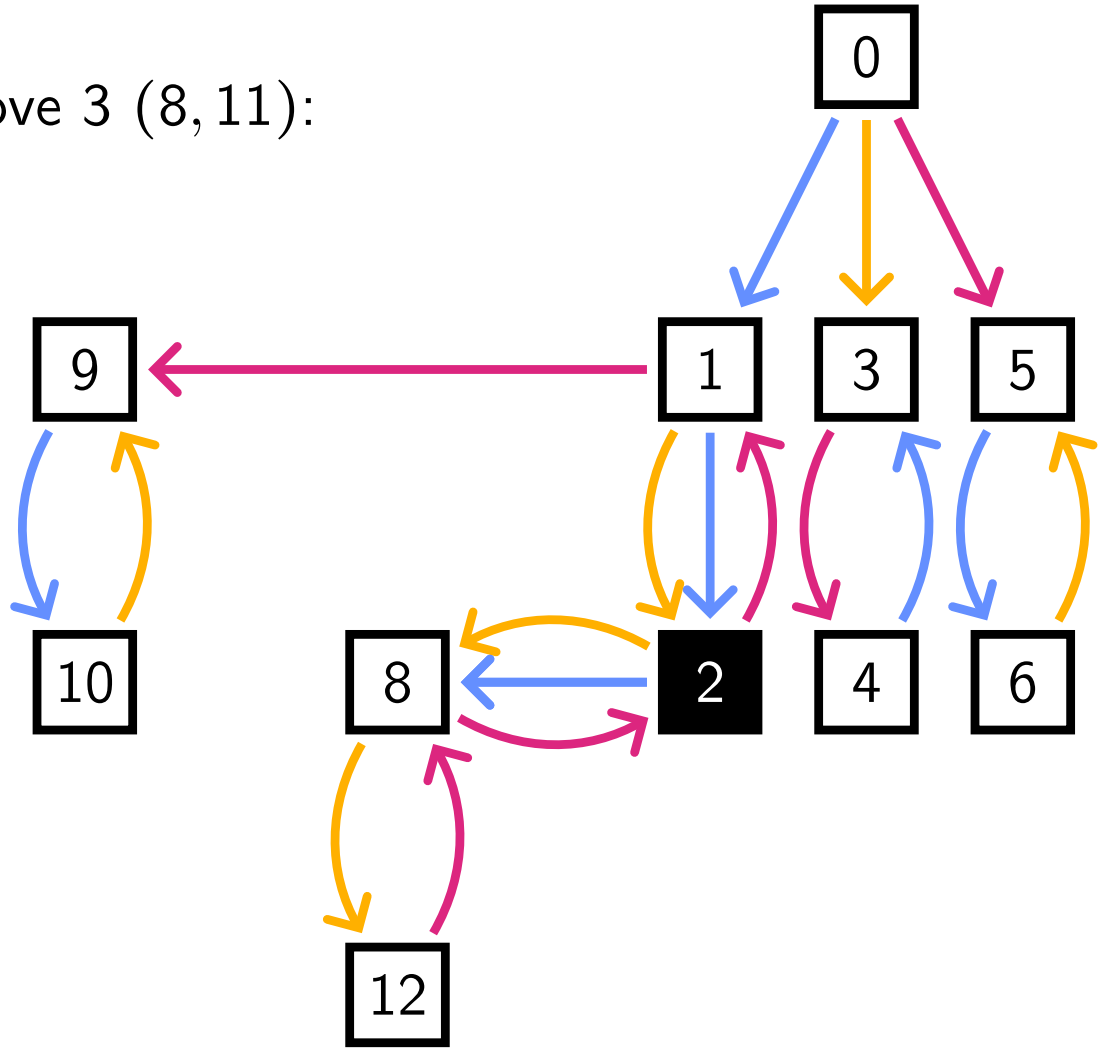
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, \underline{yrb = y}, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 3 $(8, 11)$:



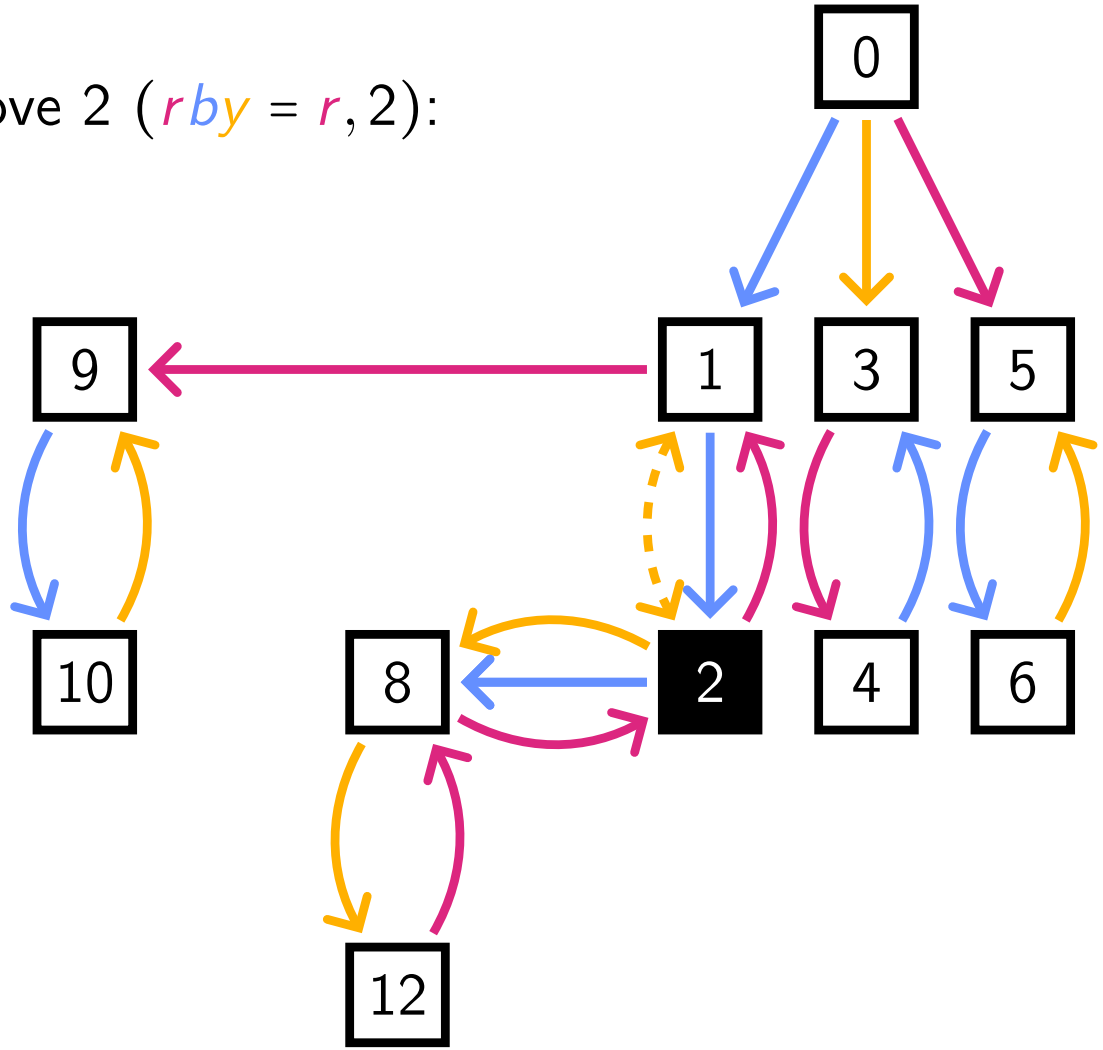
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, \underline{rby = r} \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 2 $(rby = r, 2)$:



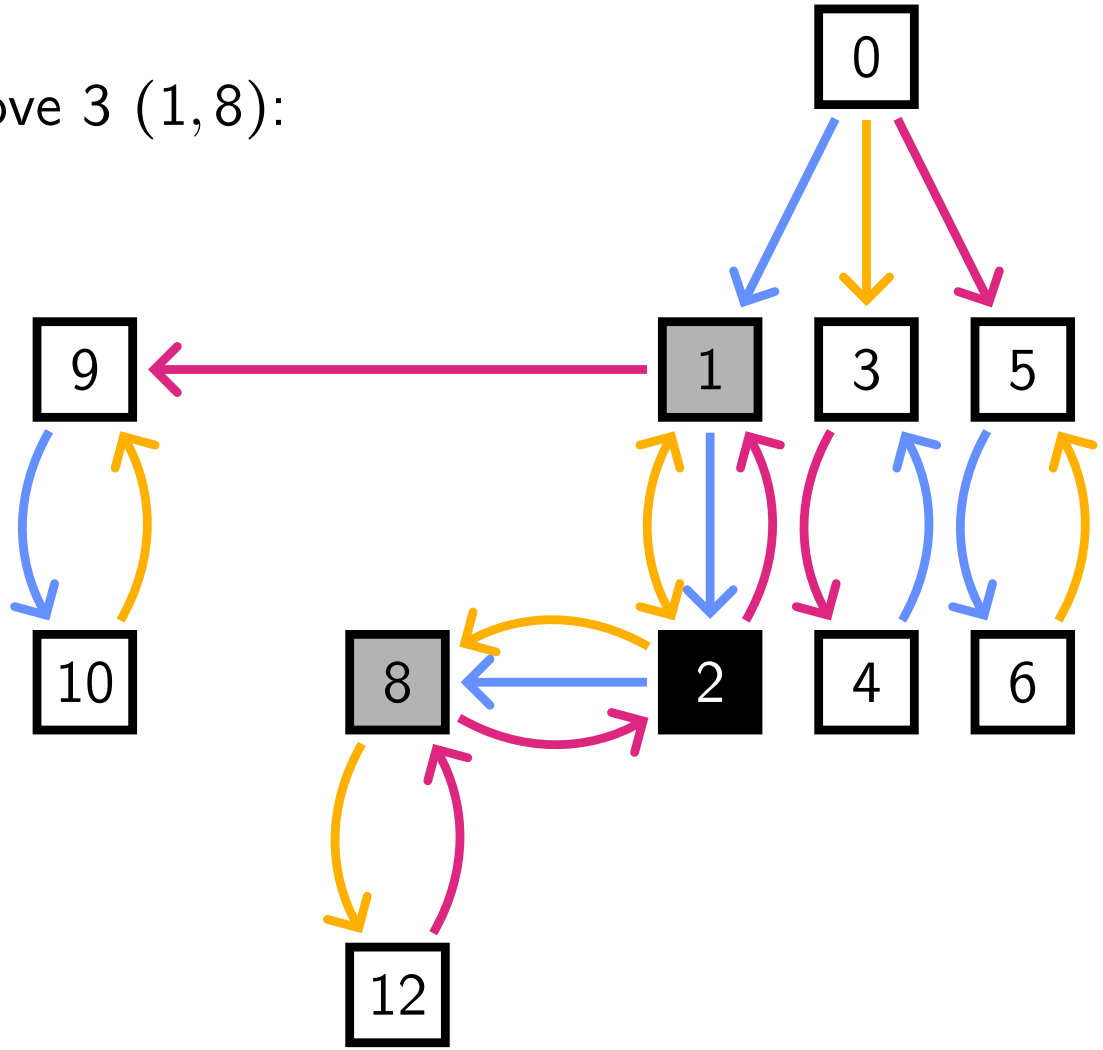
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, \underline{rby = r} \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 3 $(1, 8)$:



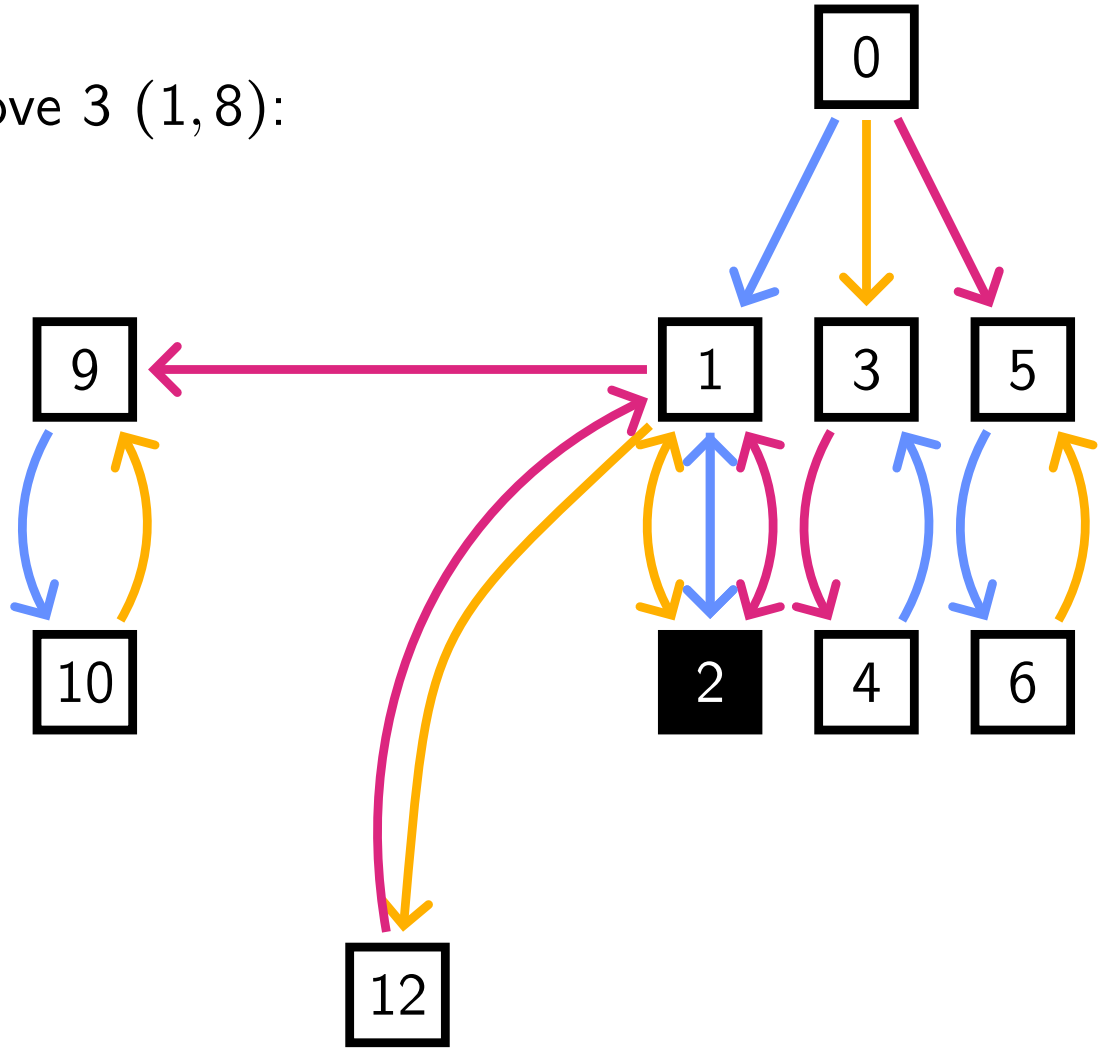
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, \underline{rby = r} \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 3 $(1, 8)$:



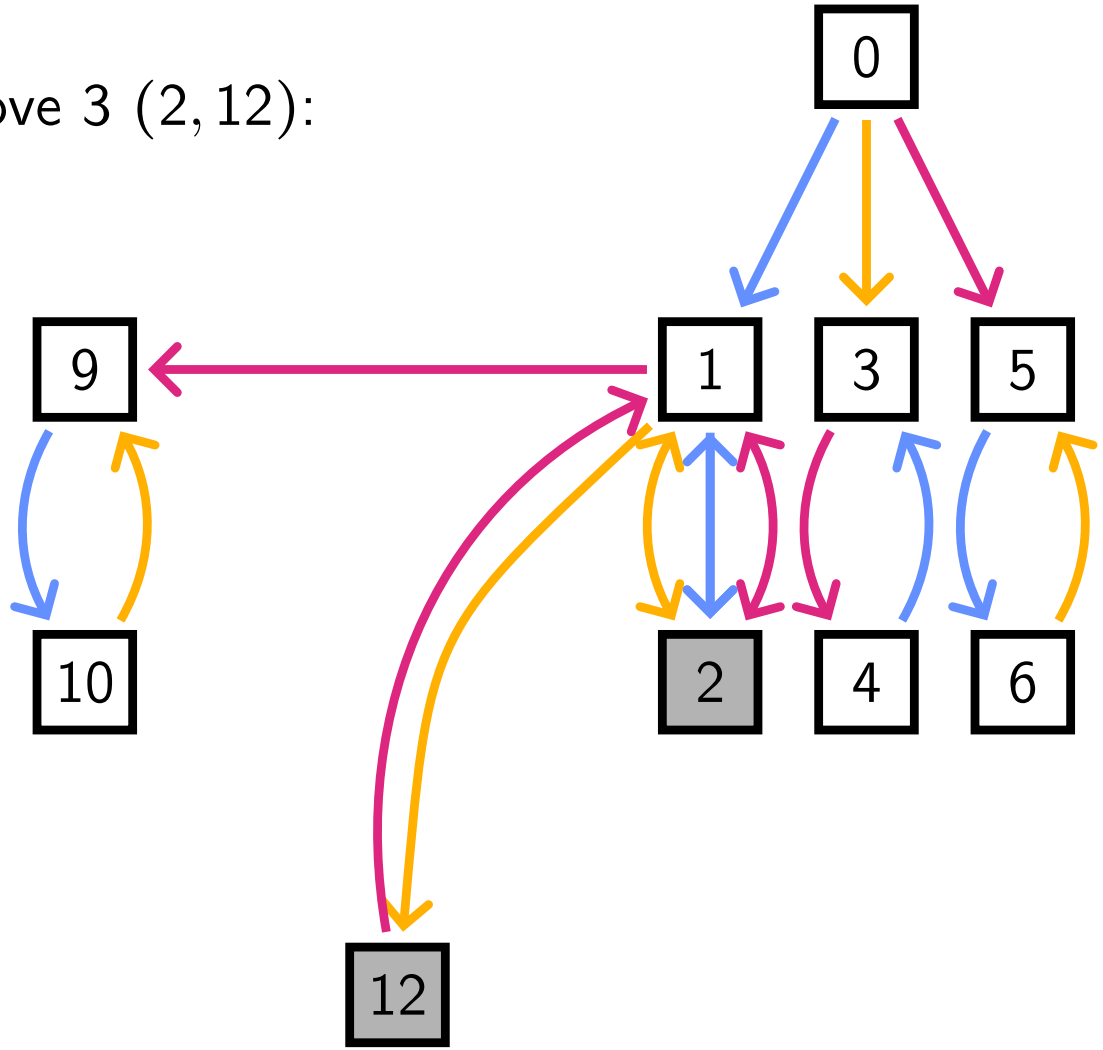
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, \underline{rby = r} \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 3 $(2, 12)$:



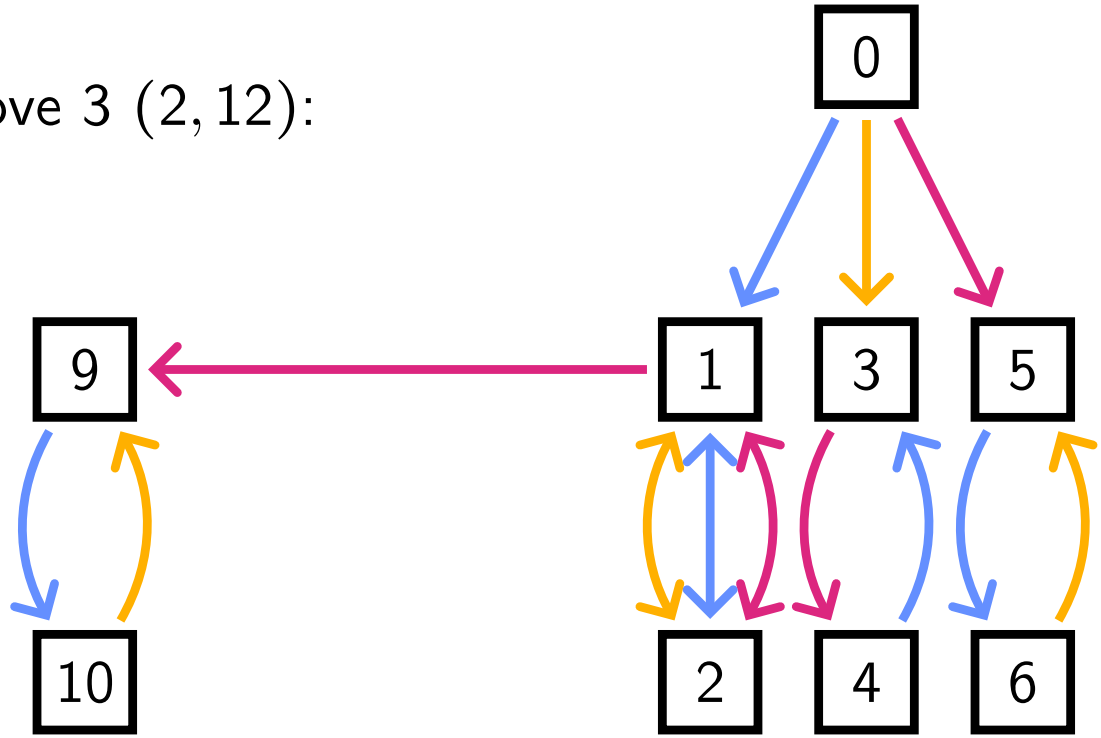
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, \underline{rby = r} \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 3 $(2, 12)$:



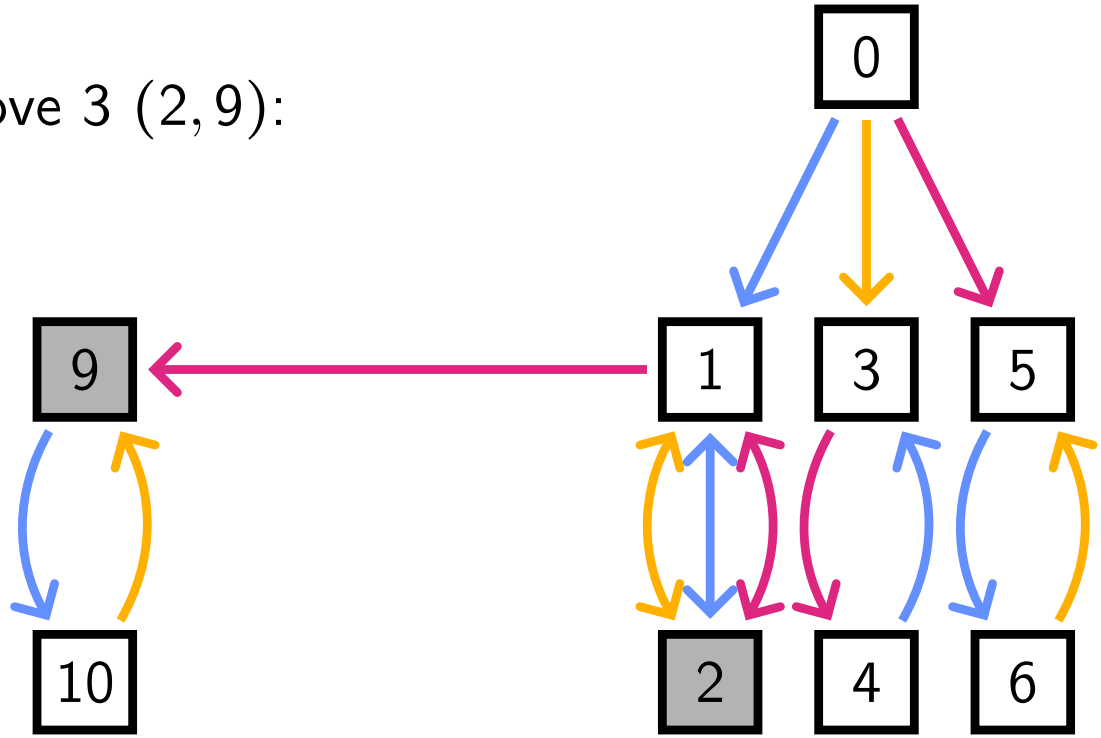
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, \underline{rby = r} \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 3 $(2, 9)$:



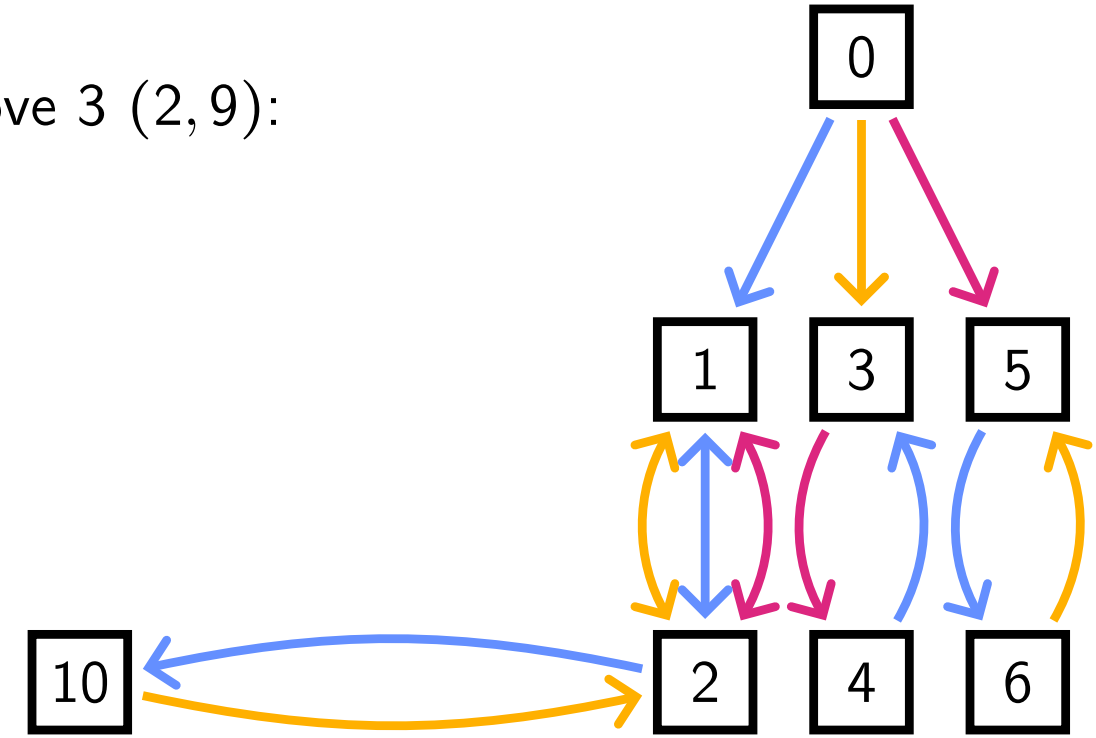
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, \underline{rby = r} \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 3 $(2, 9)$:



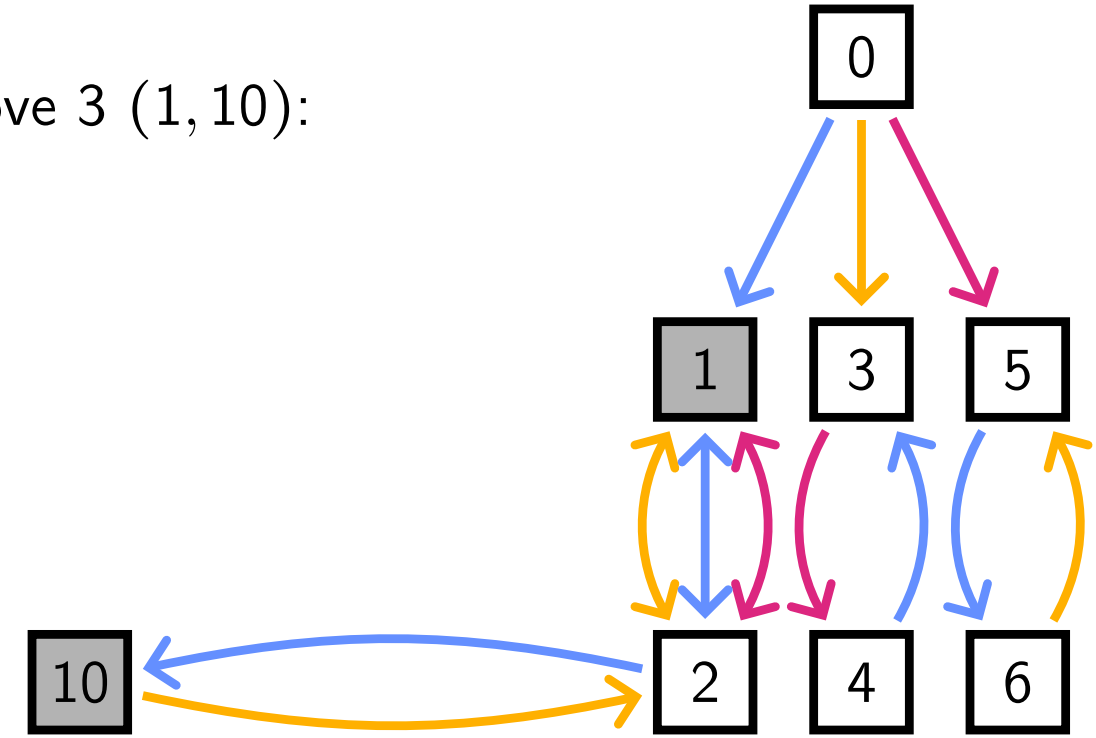
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, \underline{rby = r} \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 3 $(1, 10)$:



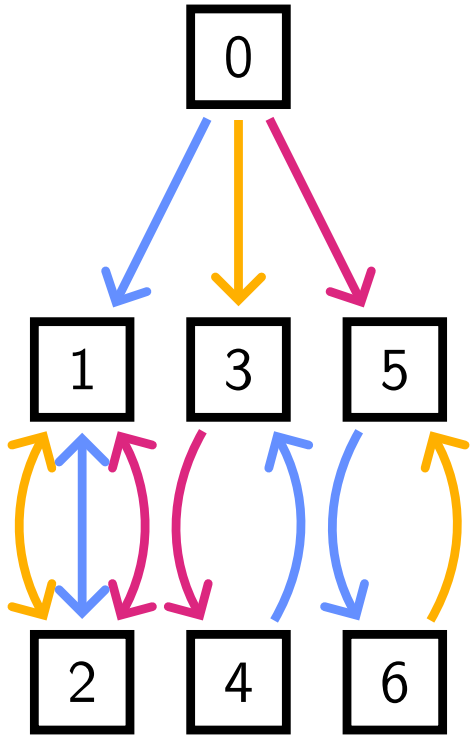
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, \underline{rby = r} \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Move 3 $(1, 10)$:



HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Continuing in this way...

. . .

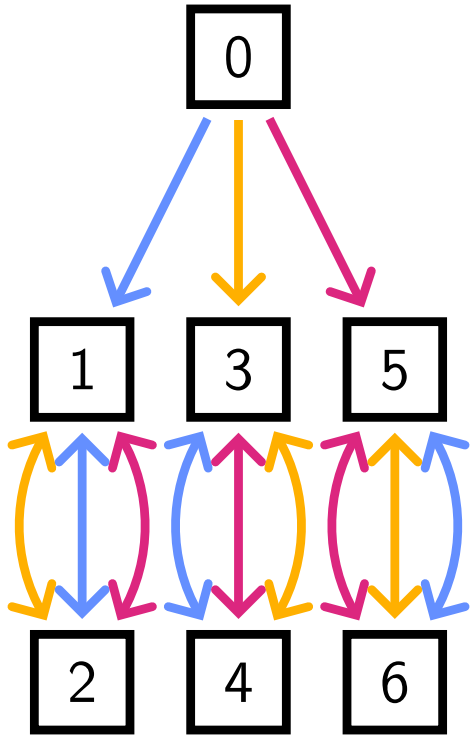
HLT strategy

Haselgrove, Leech, and Trotter

$$\langle b, y, r \mid byr = b, yrb = y, rby = r \rangle,$$

blue $= b$, yellow $= y$, red $= r$

Finally...



Capabilities + limitations

Our implementation of Todd-Coxeter can:

- ✯ generate ∼ 250 million nodes per second;
- ✯ up to ∼ 4 billion nodes in total;
- ✯ determine redundant relations in a presentation;
- ✯ prove non-triviality;
- ✯ find 1-sided congruences;
- ✯ can be used to compute congruence lattices of finite monoids;
- ✯ is almost as good as the best "coset enumerators" for groups;
- ✯ is memory bound.

Applications

- Tom Aird, JDM, and Murray Whyte, *Short presentations for transformation monoids*, 2024 https://arxiv.org/abs/2406.19294.
- M. Anagnostopoulou-Merkouri, Z. Mesyan, JDM, *Properties of Congruence Lattices of Graph Inverse Semigroups*, 2024 https://doi.org/10.1142/S0218196724500139
- M. Brookes, J. East, C. Miller, JDM, N. Ruskuc, *Heights of one- and two-sided congruence lattices of semigroups*, 2024 https://doi.org/10.2140/pjm.2024.333.17
- J. East, JDM, N. Ruskuc, M. Torpey, *Congruence lattices of finite diagram monoids*, 2018 https://doi.org/10.1016/j.aim.2018.05.016

The low-index congruence algorithm

Suppose that M is a monoid (finite or infinite) defined by a finite presentation $\langle A|R\rangle$ and $n \in \mathbb{N}$.

Aim: to compute all right congruences of M with $\leq n$ classes.

The general idea is to construct, via a backtrack search, all standard word graphs compatible with R with at most n nodes.

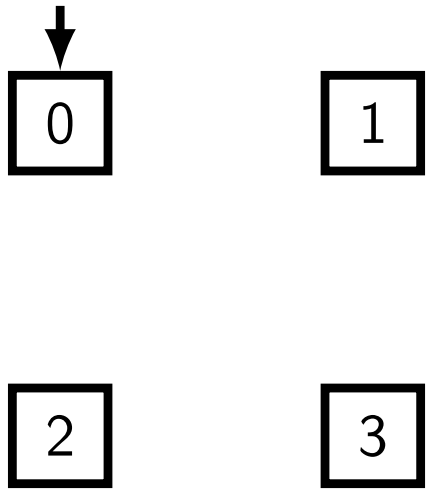
This is essentially the same as Sims low index subgroup algorithm from 1994.

Uses move 2 from Todd-Coxeter.

Low index congruence algorithm

$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle$ and $n = 4$

blue $= b$, yellow $= y$



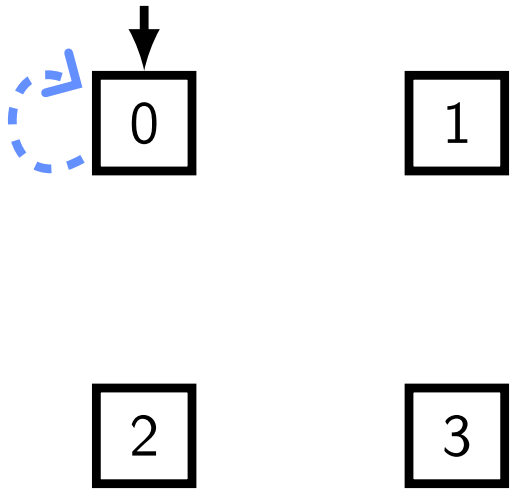
Stack: $\bot, 0 \longrightarrow 0$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Pop



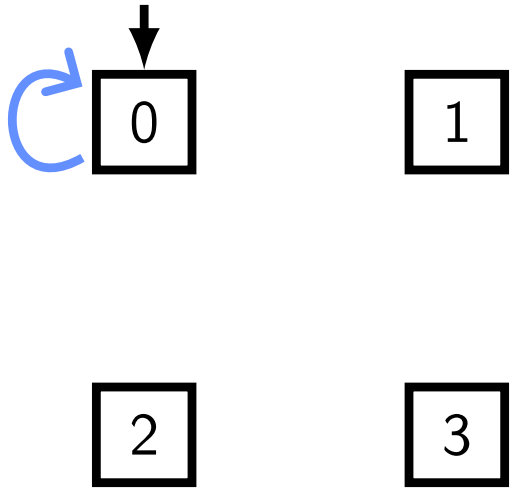
Stack: $\perp$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Push



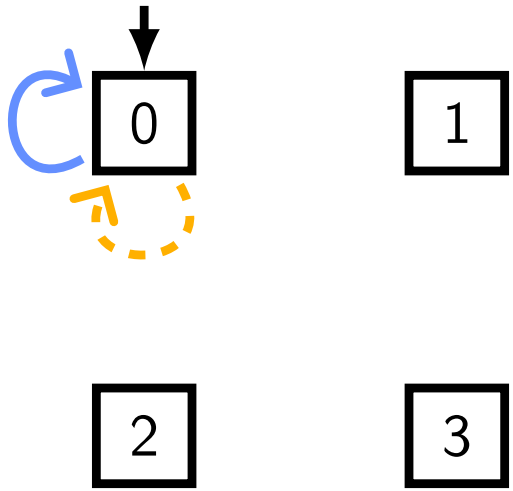
Stack: $\bot, 0 \longrightarrow 1, 0 \longrightarrow 1, 0 \longrightarrow 0$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Pop



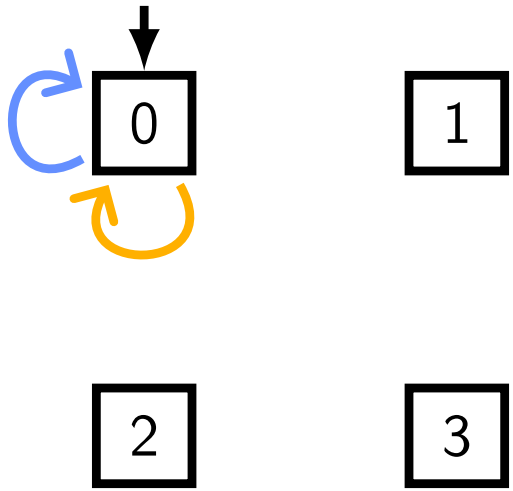
Stack: $\bot, 0 \longrightarrow 1, 0 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Good



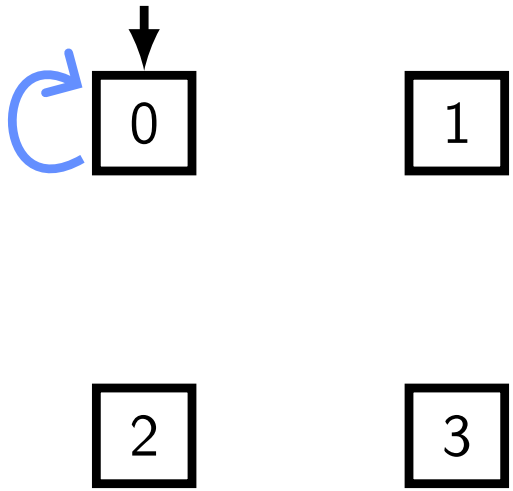
Stack: $\bot, 0 \longrightarrow 1, 0 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Backtrack



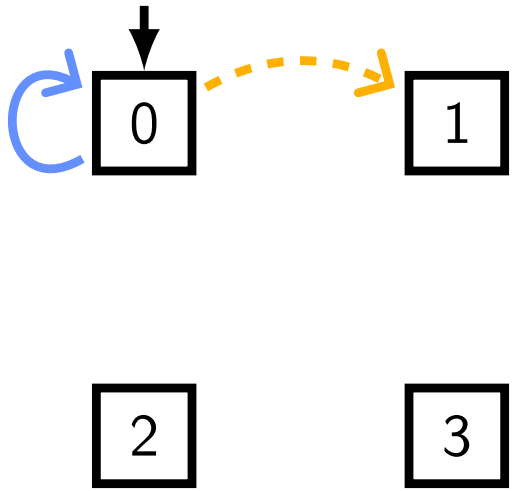
Stack: $\bot, 0 \longrightarrow 1, 0 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Pop



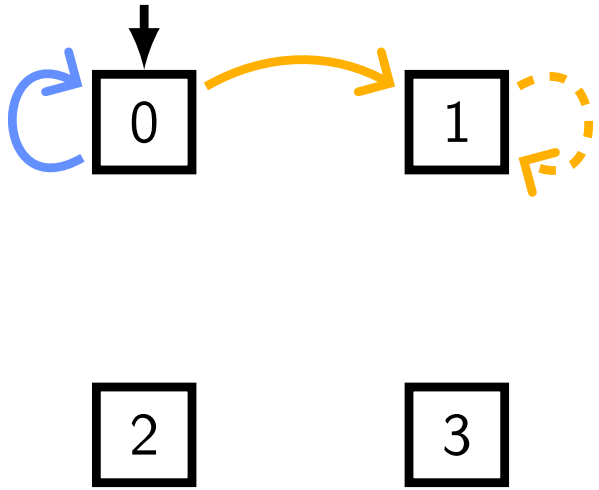
Stack: $\bot, 0 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Move 2 $(y^2 = y, 0)$



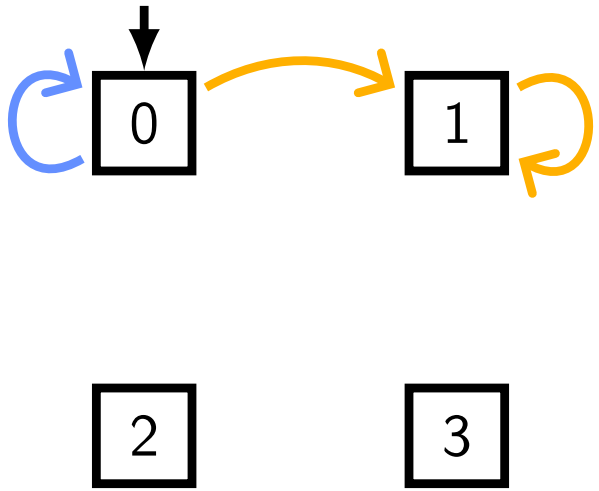
Stack: $\bot, 0 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Push



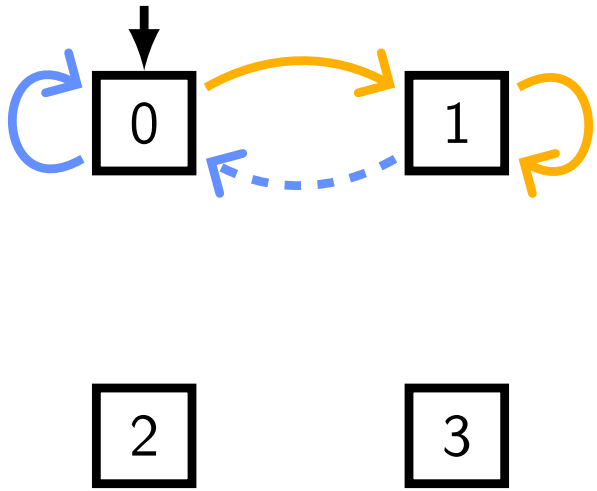
Stack: $\bot, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 1, 1 \longrightarrow 0$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Pop



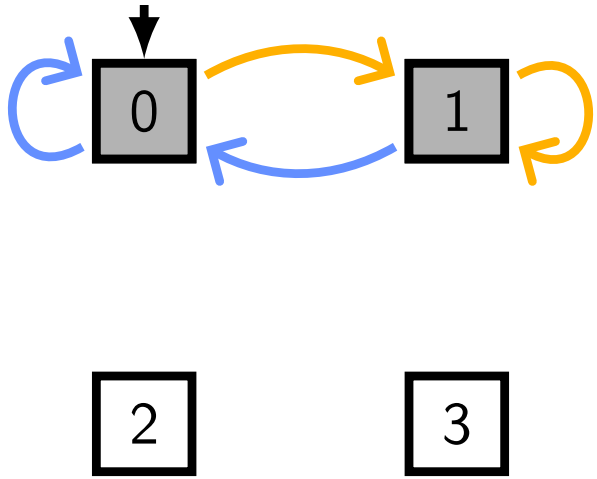
Stack: $\bot, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Incompatible with $(by)^2 = b$ at 0:



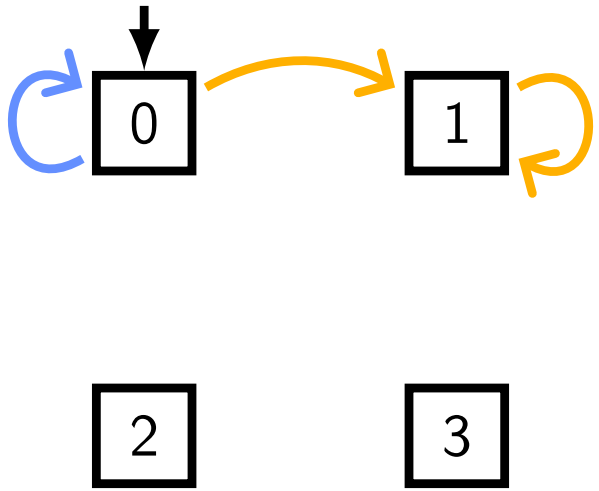
Stack: $\bot, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Backtrack



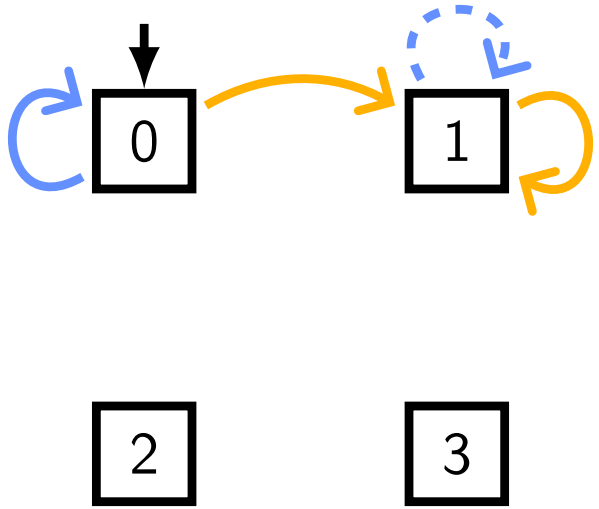
Stack: $\bot, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Pop



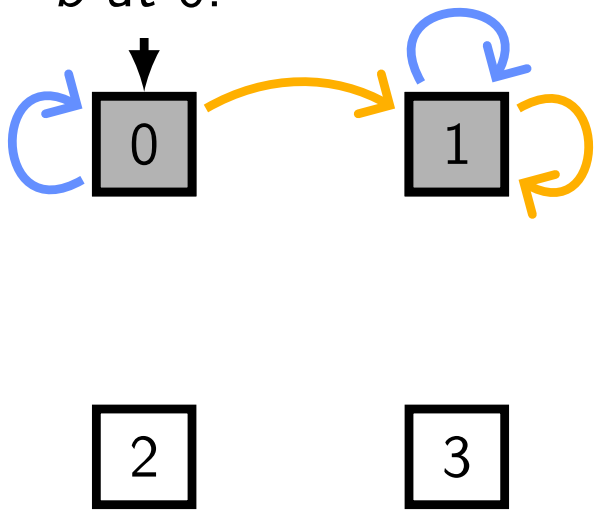
Stack: $\bot, 0 \longrightarrow 1, 1 \longrightarrow 2$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Incompatible with $(by)^2 = b$ at 0:



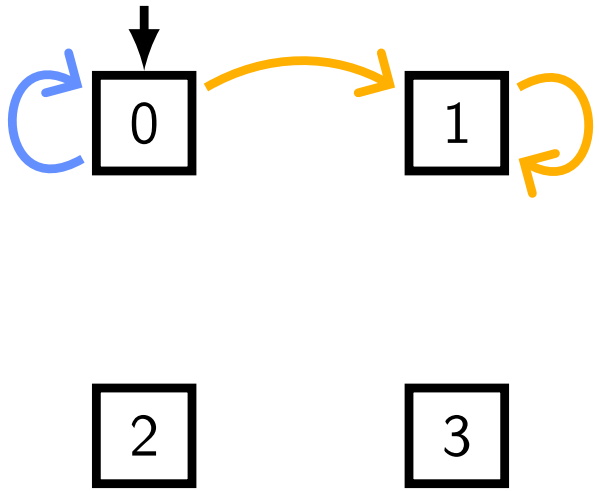
Stack: $\bot, 0 \longrightarrow 1, 1 \longrightarrow 2$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Backtrack



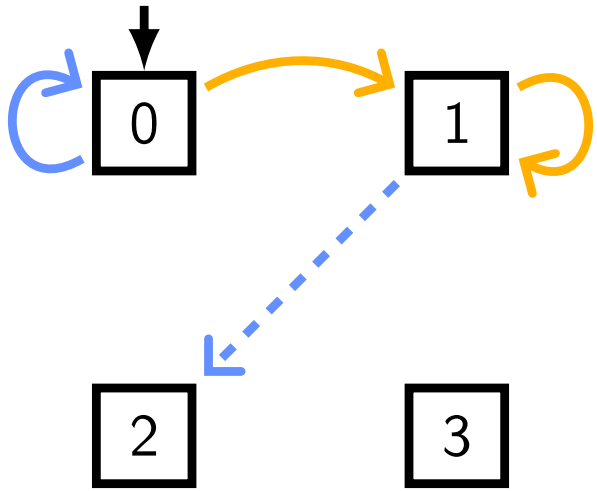
Stack: $\bot, 0 \longrightarrow 1, 1 \longrightarrow 2$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Pop



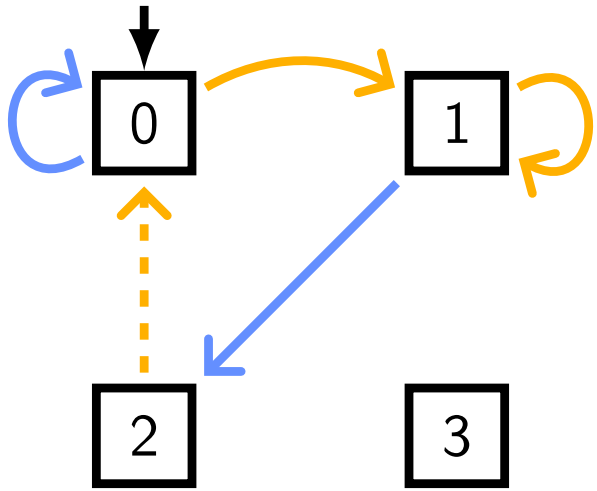
Stack: $\bot, 0 \longrightarrow 1$

Low index congruence algorithm

$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle$ and $n = 4$

blue $= b$, yellow $= y$

Move 2 $((by)^2 = b, 0)$



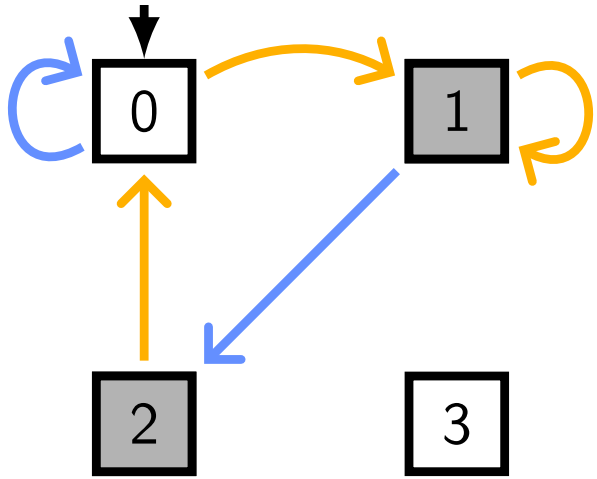
Stack: $\bot, 0 \longrightarrow 1$

Low index congruence algorithm

$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle$ and $n = 4$

blue $= b$, yellow $= y$

Incompatible with $(by)^2 = b$ at 1:



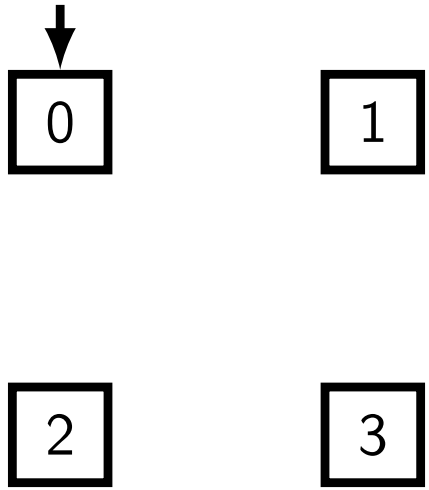
Stack: $\bot, 0 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Backtrack



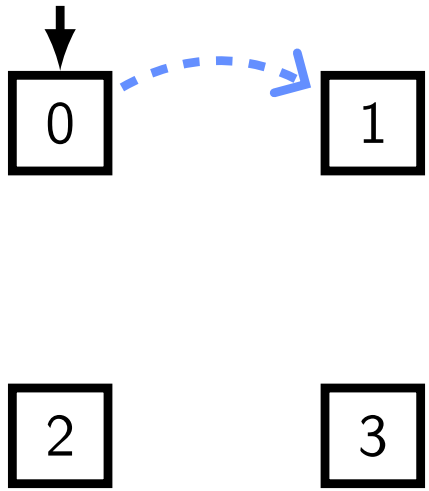
Stack: $\bot, 0 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Pop



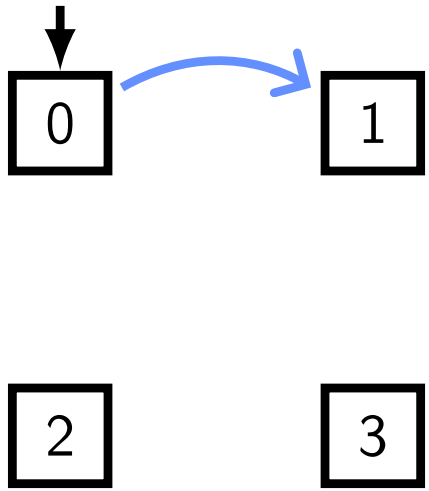
Stack: $\perp$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Push



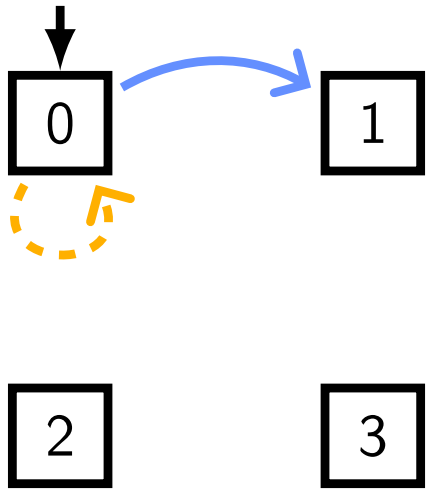
Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1, 0 \longrightarrow 0$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Pop



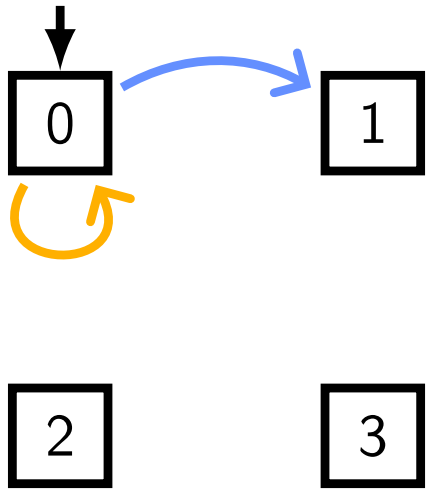
Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Push



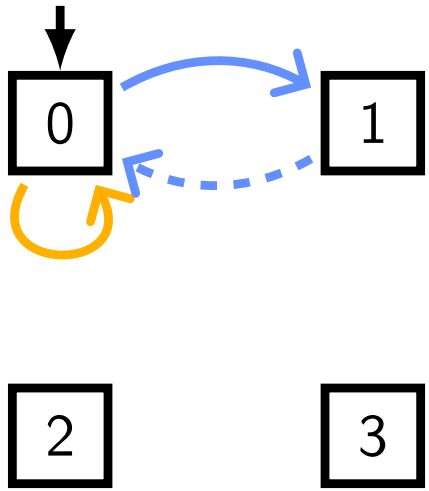
Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 1, 1 \longrightarrow 0$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Pop



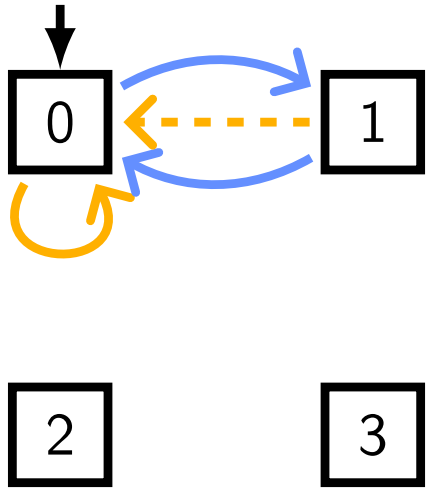
Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Move 2 $((by)^2 = b, 1)$:



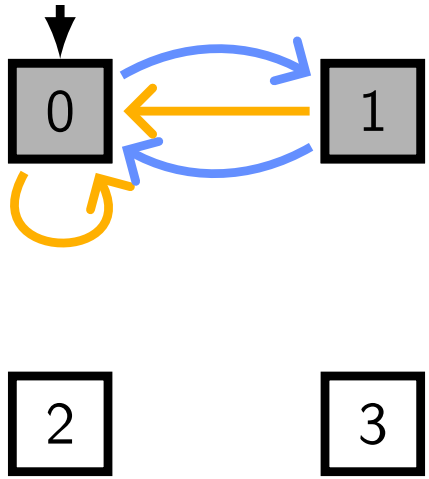
Stack: $\perp, 0 \longrightarrow 2, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Incompatible with $(by)^2 = b$ at 0:



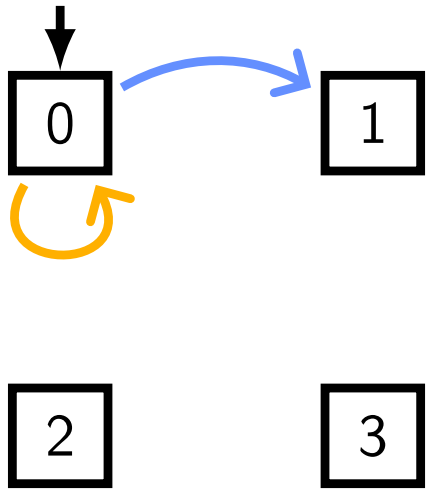
Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Backtrack



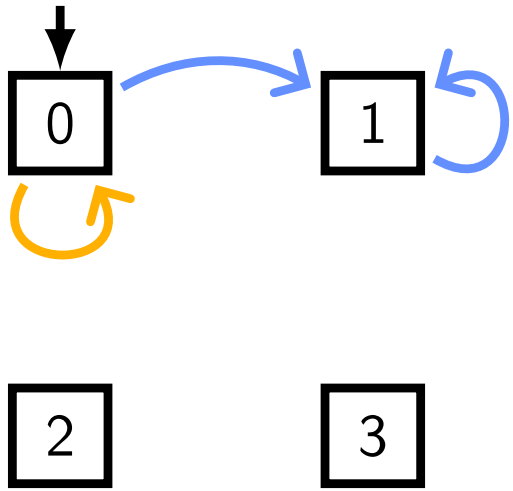
Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Pop



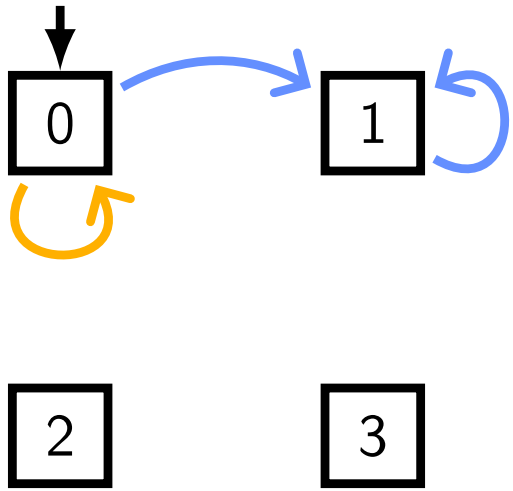
Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1, 1 \longrightarrow 2$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Push



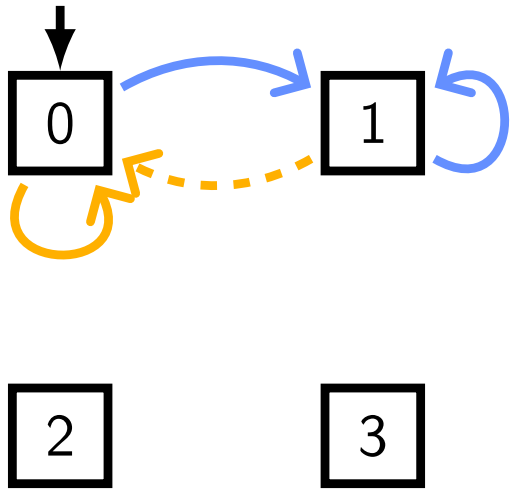
Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 2, 1 \longrightarrow 1, 1 \longrightarrow 0$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Pop



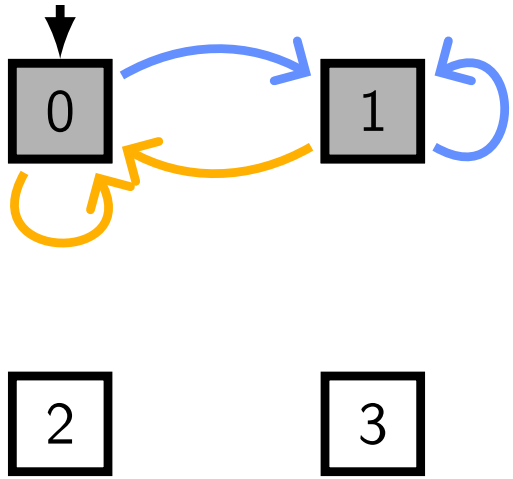
Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 2, 1 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Incompatible with $(by)^2 = b$ at 1



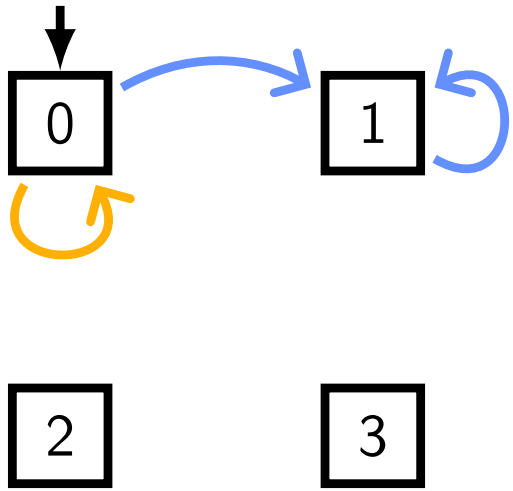
Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 2, 1 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Backtrack



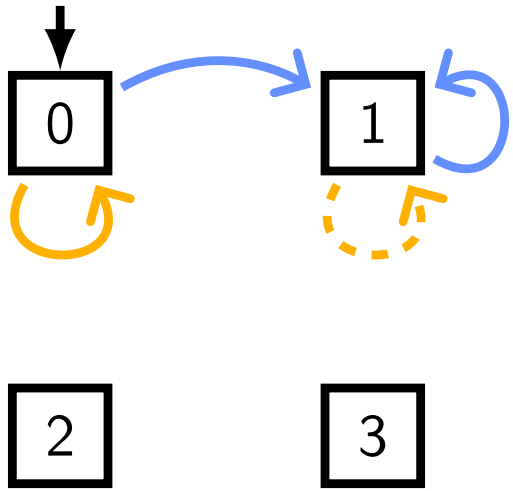
Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 2, 1 \longrightarrow 1$

Low index congruence algorithm

$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

Pop



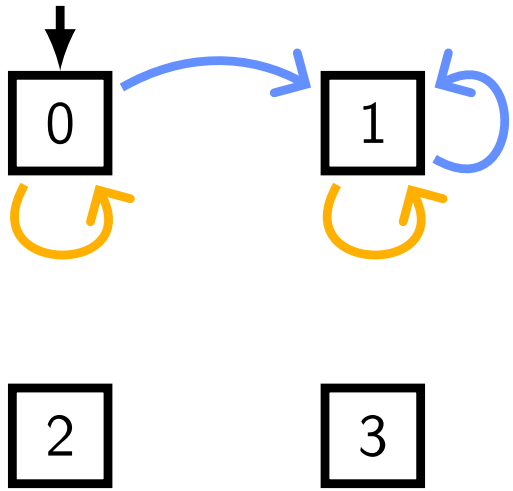
Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 2$

Low index congruence algorithm

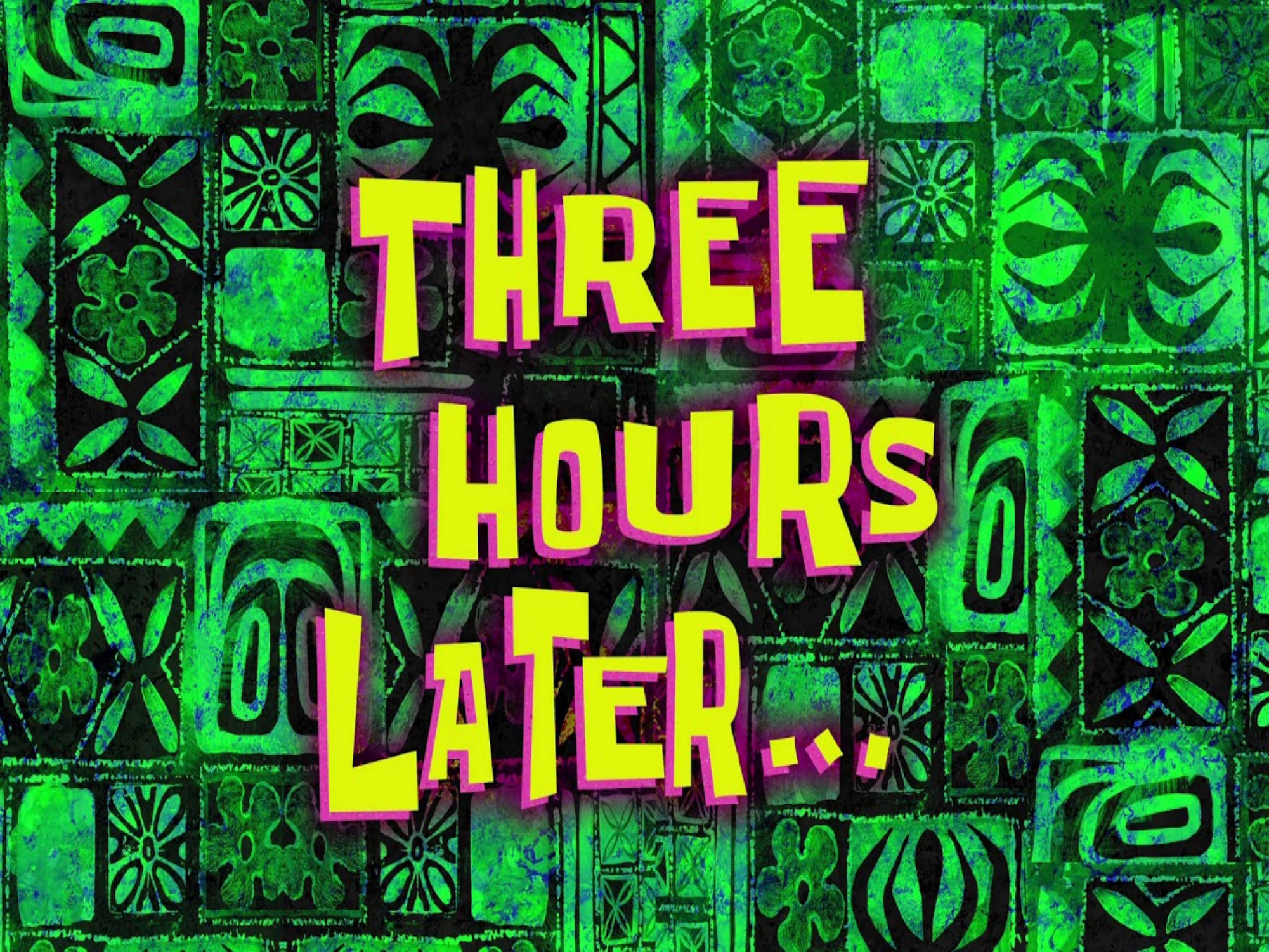
$$\langle b, y \mid b^3 = b, y^2 = y, (by)^2 = b \rangle \text{ and } n = 4$$

blue $= b$, yellow $= y$

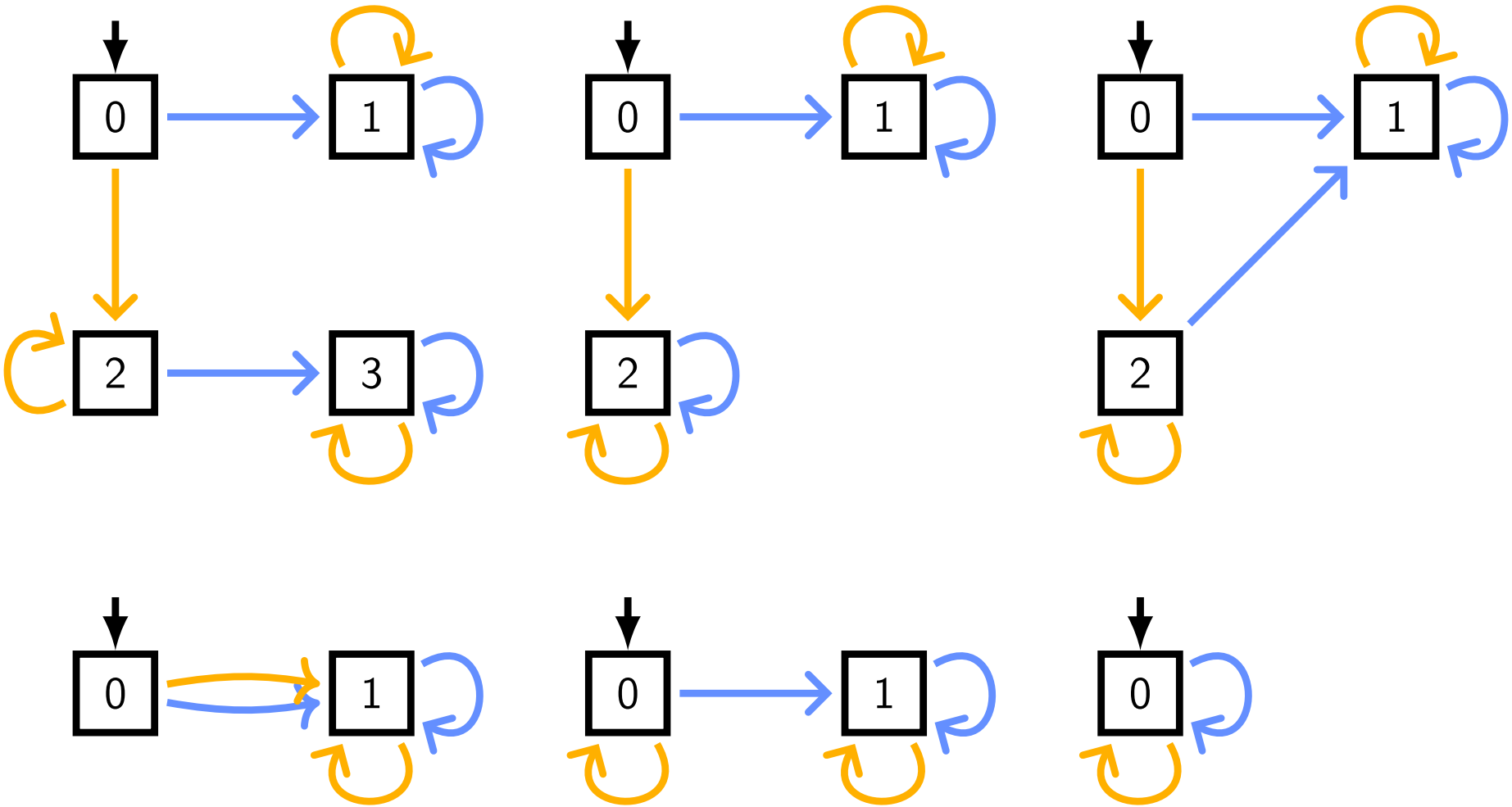
Good!



Stack: $\bot, 0 \longrightarrow 2, 0 \longrightarrow 1, 1 \longrightarrow 2, 1 \longrightarrow 2$


THREE
HOURS
LATER...

The right congruences of $M = \langle a, b \mid a^3 = a, b^2 = b, (ab)^2 = a \rangle$ are:



Capabilities + limitations

The implementation of the low-index congruence algorithm:

- ✯ can find one-sided or two-sided congruences;
- ✯ can find Rees congruences;
- ✯ can find all congruences including or excluding a given set of pairs;
- ✯ can find minimal and maximal congruences;
- ✯ can find congruences ρ such that M/ρ is a group;
- ✯ can be used to find faithful small degree transformation representations of finite monoids;
- ✯ show that two presentations define non-isomorphic monoids;
- ✯ is extremely sensitive to the input presentation;
- ✯ is parallelised;
- ✯ uses a fixed amount of memory and is time bound.

Numbers of 1-sided congruences

n	$\lvert T_n\rvert$	$\lvert \mathcal{L}_l(T_n)\rvert$	$\lvert \mathcal{L}_r(T_n)\rvert$
1	1	1	1

Numbers of 1-sided congruences

n	$\lvert T_n\rvert$	$\lvert \mathcal{L}_l(T_n)\rvert$	$\lvert \mathcal{L}_r(T_n)\rvert$
1	1	1	1
2	4	4	7

Numbers of 1-sided congruences

n	$\lvert T_n\rvert$	$\lvert \mathcal{L}_l(T_n)\rvert$	$\lvert \mathcal{L}_r(T_n)\rvert$
1	1	1	1
2	4	4	7
3	27	120	287

Numbers of 1-sided congruences

n	$\lvert T_n\rvert$	$\lvert \mathcal{L}_l(T_n)\rvert$	$\lvert \mathcal{L}_r(T_n)\rvert$
1	1	1	1
2	4	4	7
3	27	120	287
4	256	120,121	22,069,828

Numbers of 1-sided congruences

The ***stylic monoid*** with totally ordered generating set X is defined by

$$x^2 = x$$
$$yzx = yxz \text{ if } x < y \leq z$$
$$xzy = zxy \text{ if } x \leq y < z.$$

If $|X| = 4$, then this monoid has size 52 and $1,431,795,100$ left congruences.

Applications


M. Anagnostopoulou-Merkouri, R. Cirpons, JDM, M. Tsalakou *Computing finite index congruences of finitely presented semigroups and monoids*, 2025 `https://doi.org/10.1090/mcom/4136`


R. Cirpons, JDM, J. East, *Transformation representations of diagram monoids*, 2026 `https://doi.org/10.1093/imrn/rnag041`


R. Cirpons, JDM, Y. Péresse, *Minimal and maximal 1-sided congruences of the full transformation monoids*, 202X in preparation.