Context

Tools and proof techniques for systems equivalence

Methodology:

1. First do it naively
2. Then improve the associated proof method

Deterministic finite automata

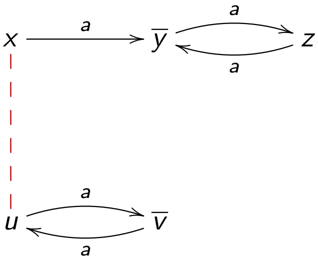
The states x and u are language equivalent

$$\begin{tikzcd}
x \arrow[r, "a"] & \overline{y} \arrow[r, bend left, "a"] & z \arrow[l, bend left, "a"]
\end{tikzcd}$$

$$\begin{tikzcd}
u \arrow[r, bend left, "a"] & \overline{v} \arrow[l, bend left, "a"]
\end{tikzcd}$$

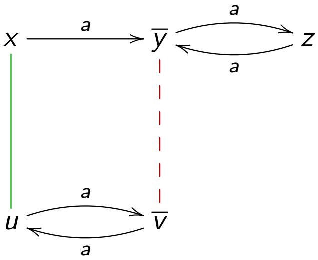
Deterministic finite automata

The states x and u are language equivalent



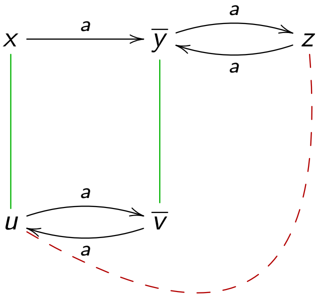
Deterministic finite automata

The states x and u are language equivalent



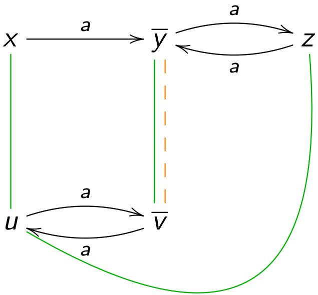
Deterministic finite automata

The states x and u are language equivalent



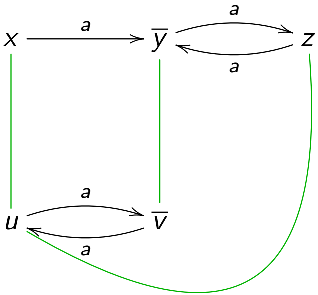
Deterministic finite automata

The states x and u are language equivalent



Deterministic finite automata

The states x and u are language equivalent



Deterministic finite automata

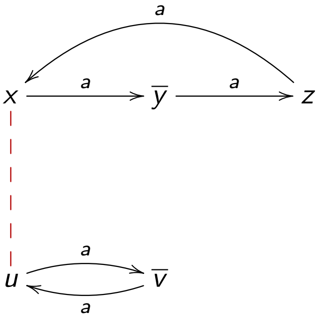
x and u are **not** equivalent

$$\begin{tikzcd} x \arrow[r, "a"] & \overline{y} \arrow[r, "a"] & z \arrow[ll, bend right, "a"'] \end{tikzcd}$$

$$\begin{tikzcd} u \arrow[r, bend left, "a"] & \overline{v} \arrow[l, bend left, "a"] \end{tikzcd}$$

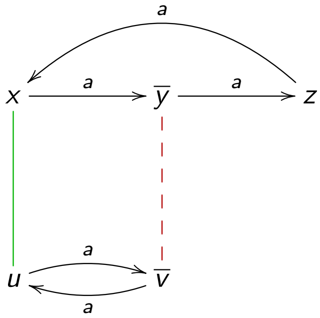
Deterministic finite automata

x and u are **not** equivalent



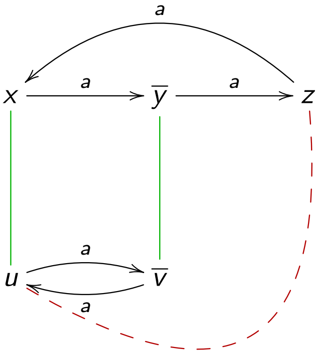
Deterministic finite automata

x and u are **not** equivalent



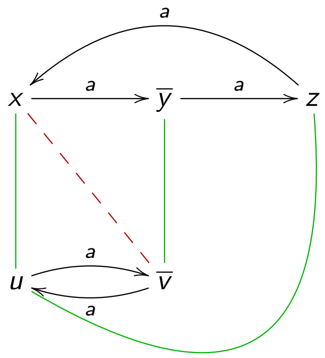
Deterministic finite automata

x and u are **not** equivalent



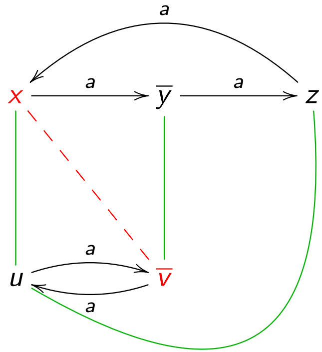
Deterministic finite automata

x and u are **not** equivalent



Deterministic finite automata

x and u are **not** equivalent



Deterministic finite automata

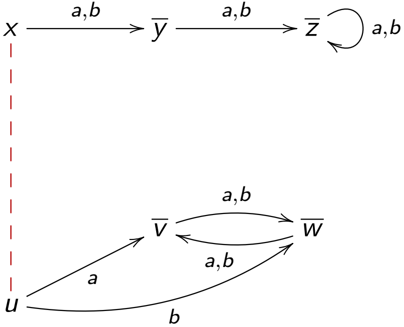
Last example, with two letters

$$\begin{tikzcd} x \arrow[r, "{a,b}"] & \overline{y} \arrow[r, "{a,b}"] & \overline{z} \arrow[loop right, "{a,b}"] \end{tikzcd}$$

$$\begin{tikzcd} & \overline{v} \arrow[r, bend left, "{a,b}"] & \overline{w} \arrow[l, bend left, "{a,b}"] \\ u \arrow[ur, "a"'] \arrow[urr, bend right, "b"'] & & \end{tikzcd}$$

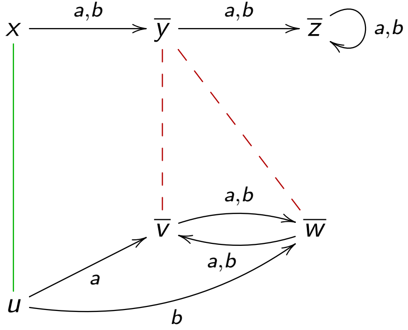
Deterministic finite automata

Last example, with two letters



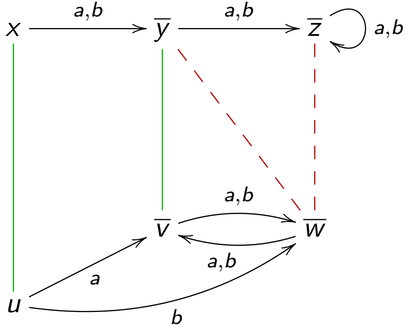
Deterministic finite automata

Last example, with two letters



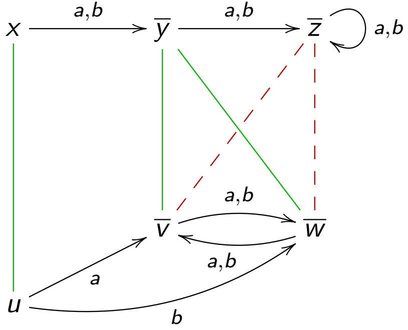
Deterministic finite automata

Last example, with two letters



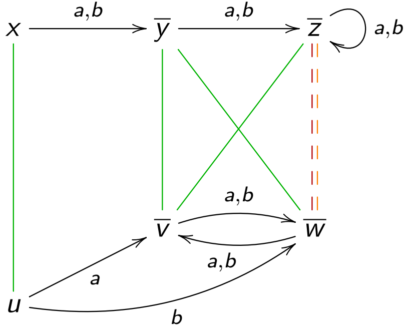
Deterministic finite automata

Last example, with two letters



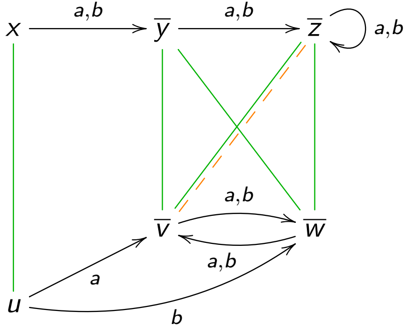
Deterministic finite automata

Last example, with two letters



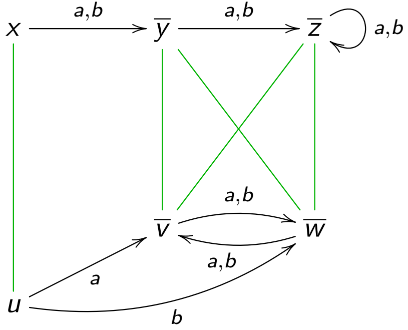
Deterministic finite automata

Last example, with two letters



Deterministic finite automata

Last example, with two letters



Correctness

- A relation R is a proof of equivalence (bisimulation) if $x \; R \; y$ entails
  - $o(x) = o(y)$;
  - for all a, $t_a(x) \; R \; t_a(y)$.

Correctness

- A relation R is a proof of equivalence (bisimulation) if $x \; R \; y$ entails
  - $o(x) = o(y)$;
  - for all a, $t_a(x) \; R \; t_a(y)$.
- *Theorem:* $L(x) = L(y)$ iff
  there exists a bisimulation R with $x \; R \; y$

Correctness

- A relation R is a proof of equivalence (bisimulation) if $x\ R\ y$ entails
  - $o(x) = o(y)$;
  - for all a, $t_a(x)\ R\ t_a(y)$.
- *Theorem:* $L(x) = L(y)$ iff
  there exists a bisimulation R with $x\ R\ y$

The previous algorithm attempts to construct a bisimulation

Complexity

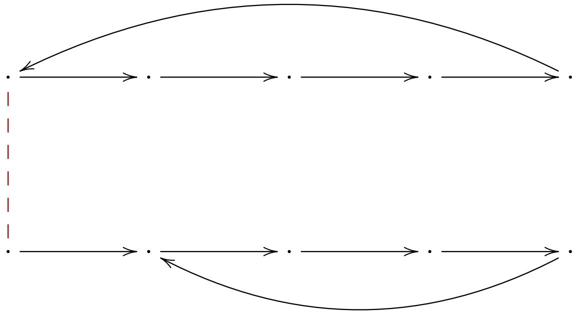
The previous algorithm is quadratic

$$\begin{tikzcd} \cdot \arrow[r] & \cdot \arrow[r] & \cdot \arrow[r] & \cdot \arrow[r] & \cdot \arrow[llll, bend right] \end{tikzcd}$$

$$\begin{tikzcd} \cdot \arrow[r] & \cdot \arrow[r] & \cdot \arrow[r] & \cdot \arrow[r] & \cdot \arrow[lll, bend left] \end{tikzcd}$$

Complexity

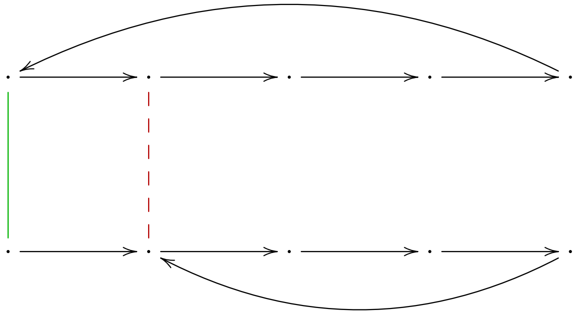
The previous algorithm is quadratic



0 pairs

Complexity

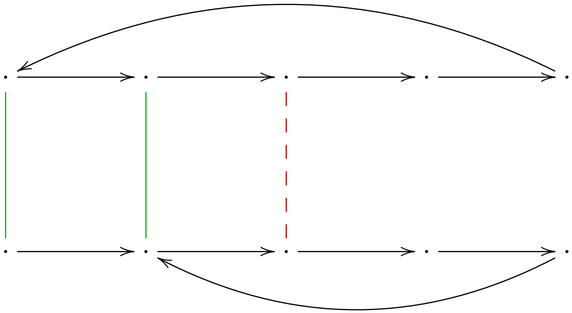
The previous algorithm is quadratic



1 pairs

Complexity

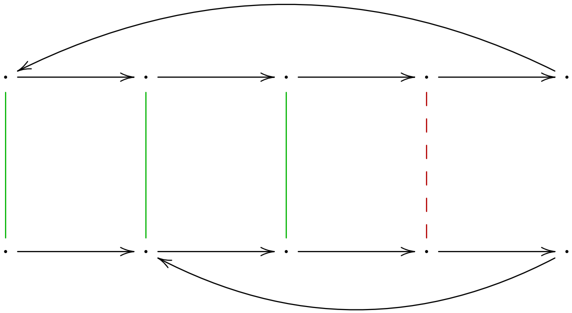
The previous algorithm is quadratic



2 pairs

Complexity

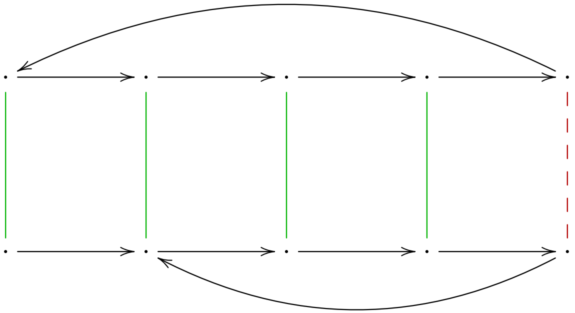
The previous algorithm is quadratic



3 pairs

Complexity

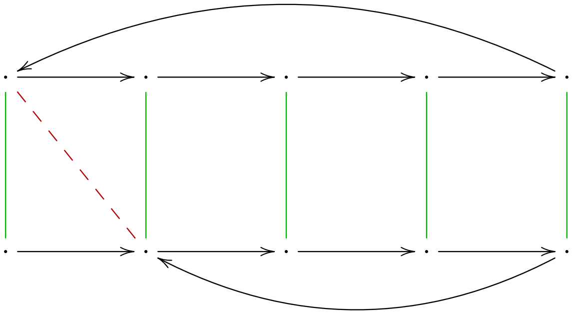
The previous algorithm is quadratic



4 pairs

Complexity

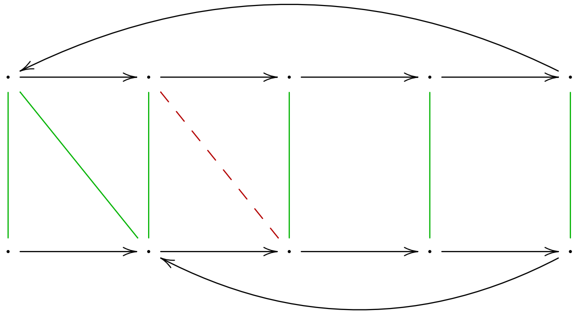
The previous algorithm is quadratic



5 pairs

Complexity

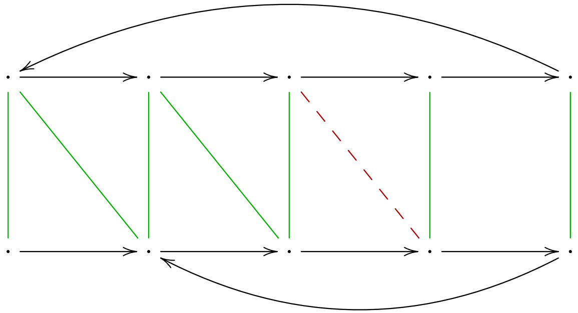
The previous algorithm is quadratic



6 pairs

Complexity

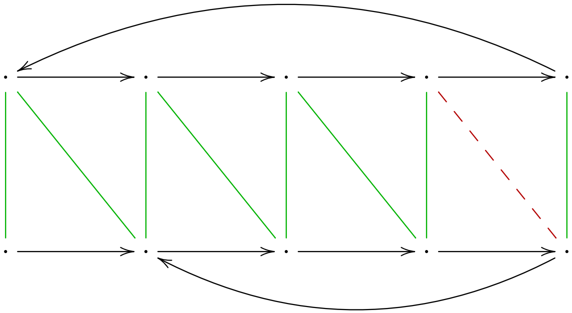
The previous algorithm is quadratic



7 pairs

Complexity

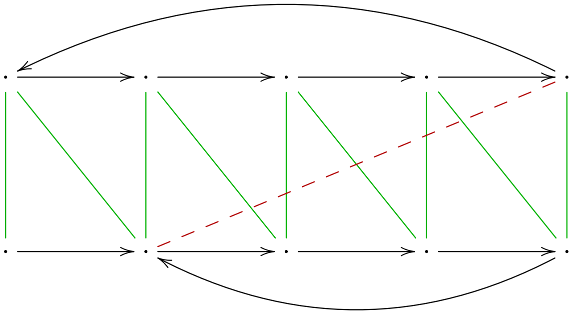
The previous algorithm is quadratic



8 pairs

Complexity

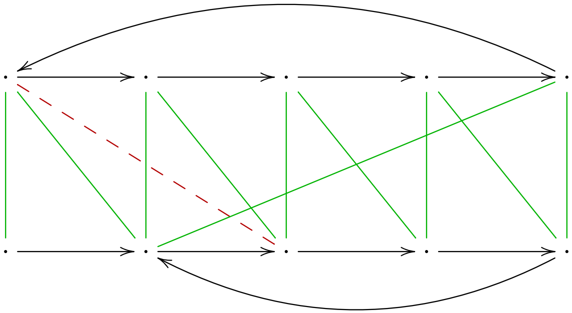
The previous algorithm is quadratic



9 pairs

Complexity

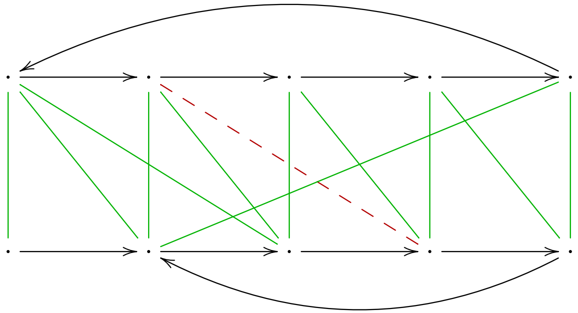
The previous algorithm is quadratic



10 pairs

Complexity

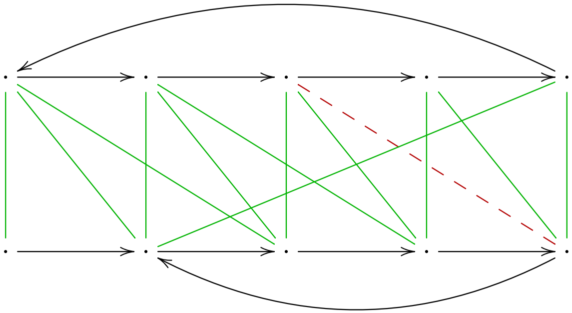
The previous algorithm is quadratic



11 pairs

Complexity

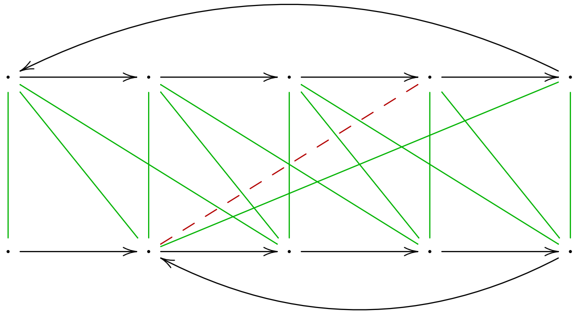
The previous algorithm is quadratic



12 pairs

Complexity

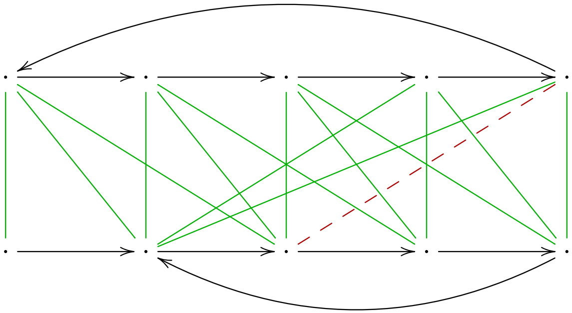
The previous algorithm is quadratic



13 pairs

Complexity

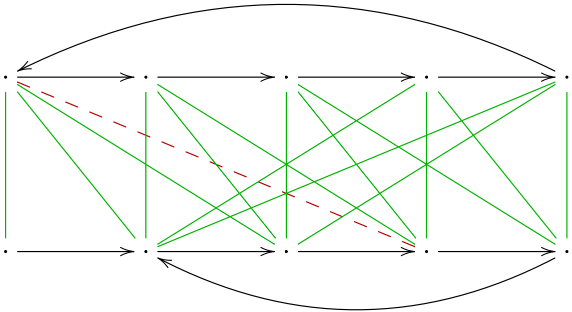
The previous algorithm is quadratic



14 pairs

Complexity

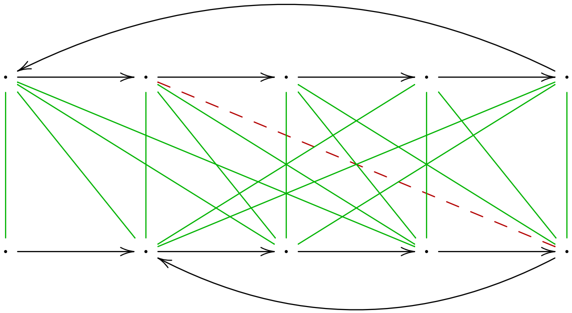
The previous algorithm is quadratic



15 pairs

Complexity

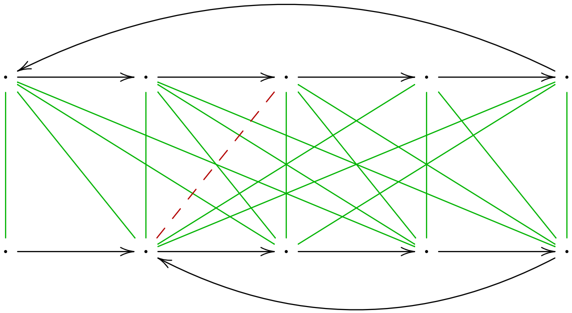
The previous algorithm is quadratic



16 pairs

Complexity

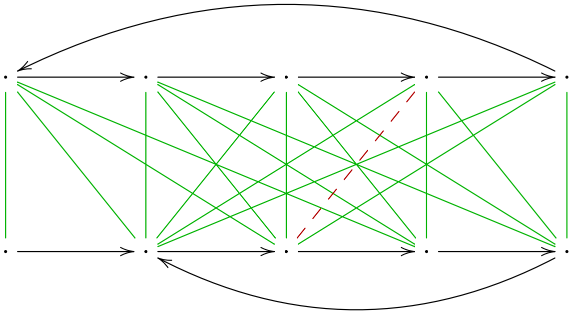
The previous algorithm is quadratic



17 pairs

Complexity

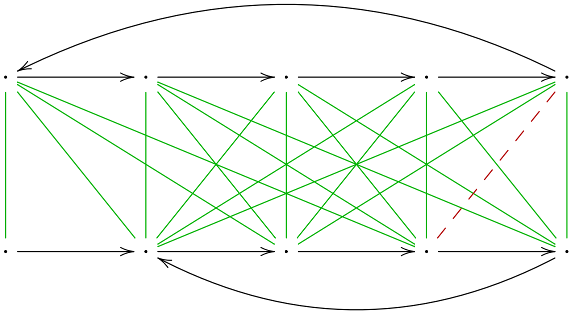
The previous algorithm is quadratic



18 pairs

Complexity

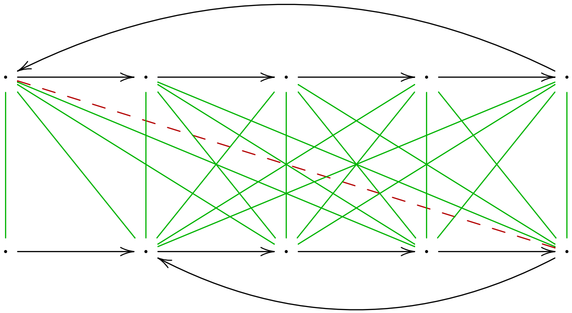
The previous algorithm is quadratic



19 pairs

Complexity

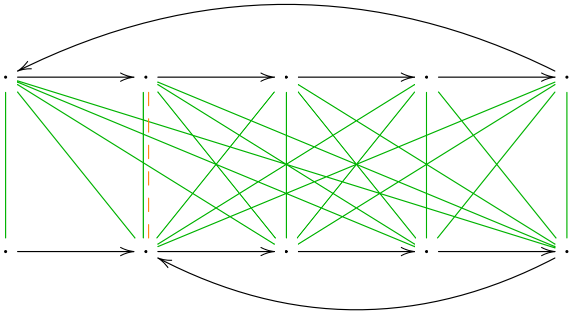
The previous algorithm is quadratic



20 pairs

Complexity

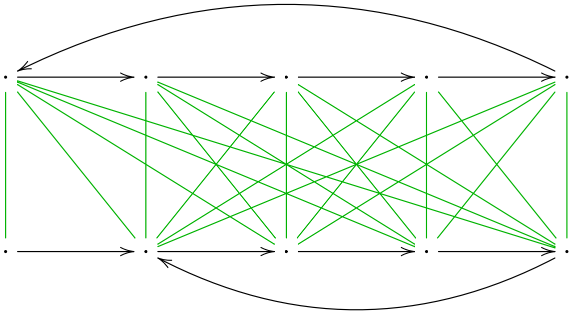
The previous algorithm is quadratic



21 pairs

Complexity

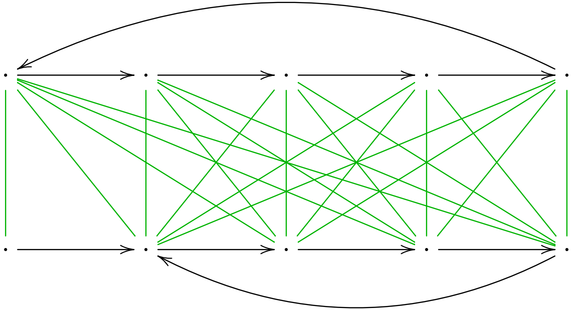
The previous algorithm is quadratic



21 pairs

First improvement

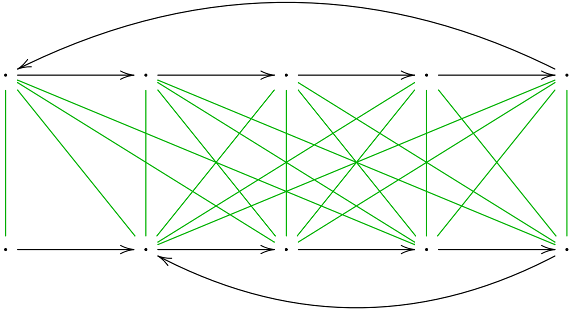
One can stop much earlier



21 pairs

First improvement

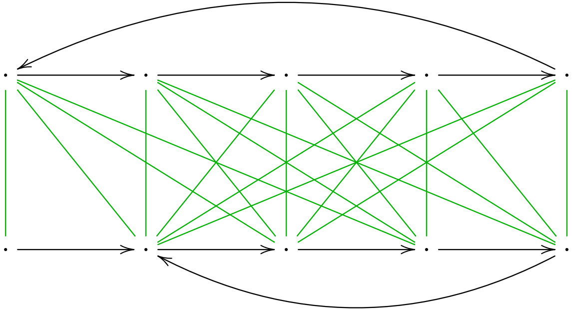
One can stop much earlier



~~21~~ 20 pairs

First improvement

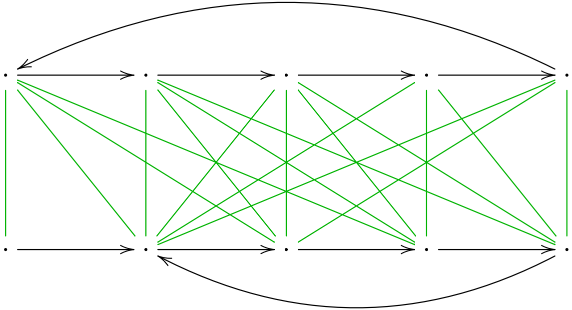
One can stop much earlier



~~21~~ 19 pairs

First improvement

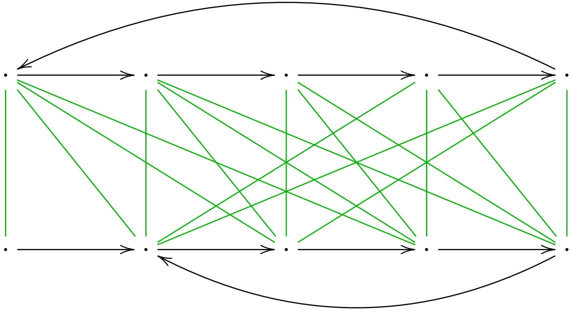
One can stop much earlier



~~21~~ 18 pairs

First improvement

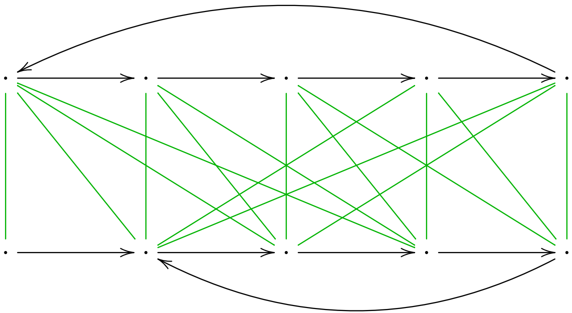
One can stop much earlier



~~21~~ 17 pairs

First improvement

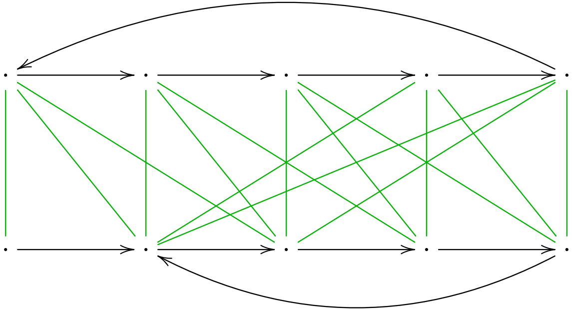
One can stop much earlier



~~21~~ 16 pairs

First improvement

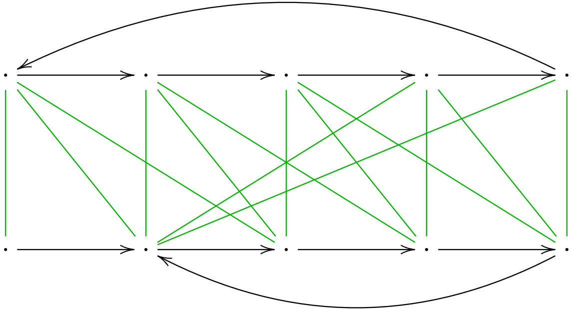
One can stop much earlier



~~21~~ 15 pairs

First improvement

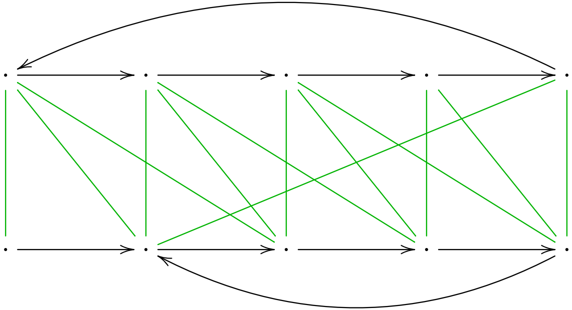
One can stop much earlier



~~21~~ 14 pairs

First improvement

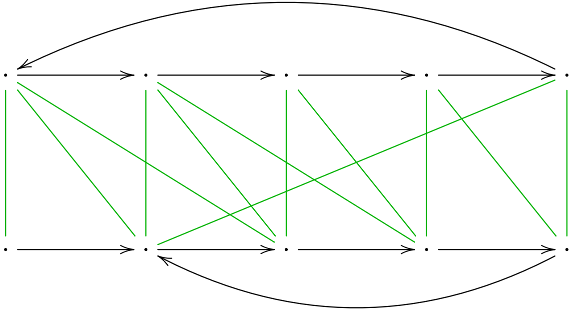
One can stop much earlier



~~21~~ 13 pairs

First improvement

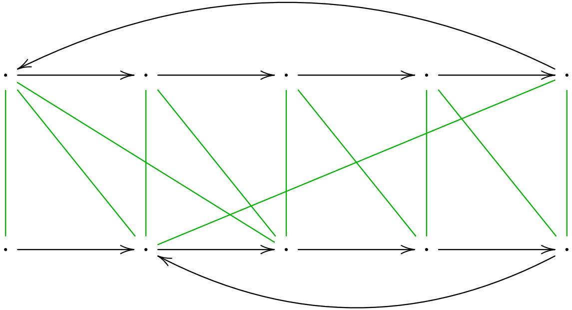
One can stop much earlier



~~21~~ 12 pairs

First improvement

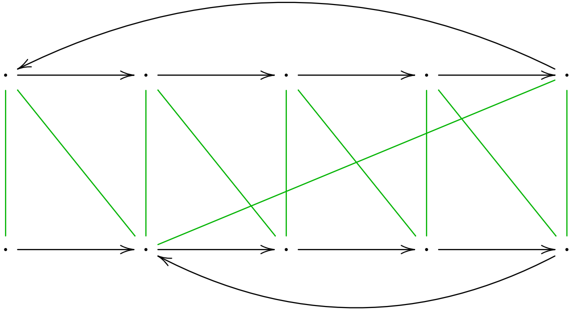
One can stop much earlier



~~21~~ 11 pairs

First improvement

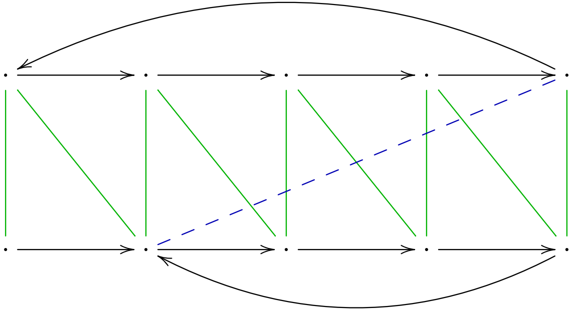
One can stop much earlier



~~21~~ 10 pairs

First improvement

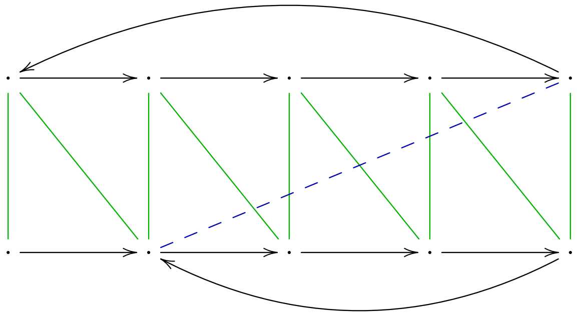
One can stop much earlier



~~21~~ 9 pairs

First improvement

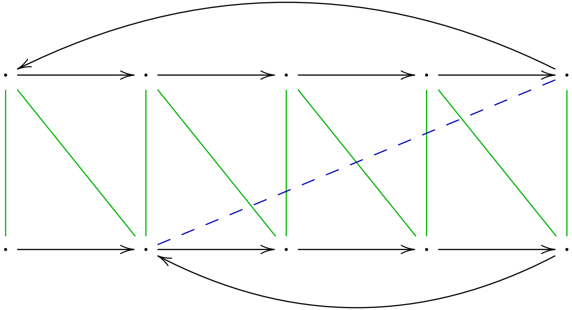
One can stop much earlier



[Hopcroft and Karp '71]

First improvement

One can stop much earlier



[Hopcroft and Karp '71]

Complexity: almost linear

[Tarjan '75]

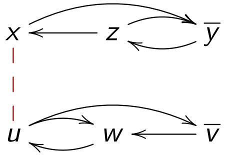
Correctness of the improvement

Correctness of HK algorithm, revisited:

- The previous relation is not a bisimulation - proof of equivalence
- But can be completed to one using equivalence - transitivity
- Hopcroft and Karp's algorithm ('71) attempts to construct a bisimulation up to equivalence

Non-Deterministic Automata

Use Hopcroft and Karp on the fly, through the powerset construction:



Non-Deterministic Automata

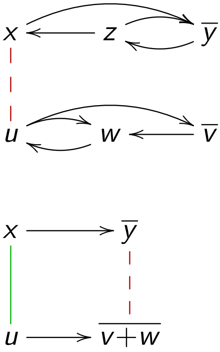
Use Hopcroft and Karp on the fly, through the powerset construction:

$$\begin{tikzcd}
x \arrow[rr, bend left] \arrow[d, dash, dashed] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}$$

$$\begin{tikzcd}
x \arrow[d, dash, dashed] \\
u
\end{tikzcd}$$

Non-Deterministic Automata

Use Hopcroft and Karp on the fly, through the powerset construction:



Non-Deterministic Automata

Use Hopcroft and Karp on the fly, through the powerset construction:

$$
\begin{tikzcd}
x \arrow[d, dash, dashed] \arrow[rr, bend left] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}
$$

$$
\begin{tikzcd}
x \arrow[r] \arrow[d, dash] & \overline{y} \arrow[r] \arrow[d, dash] & z \arrow[d, dash, dashed] \\
u \arrow[r] & \overline{v+w} \arrow[r] & u+w
\end{tikzcd}
$$

Non-Deterministic Automata

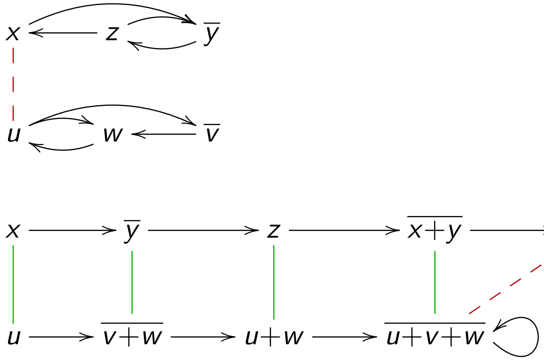
Use Hopcroft and Karp on the fly, through the powerset construction:

$$
\begin{tikzcd}
x \arrow[rr, bend left] \arrow[d, dash, dashed] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left=40] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}
$$

$$
\begin{tikzcd}
x \arrow[r] \arrow[d, dash] & \overline{y} \arrow[r] \arrow[d, dash] & z \arrow[r] \arrow[d, dash] & \overline{x+y} \arrow[d, dash, dashed] \\
u \arrow[r] & \overline{v+w} \arrow[r] & u+w \arrow[r] & \overline{u+v+w}
\end{tikzcd}
$$

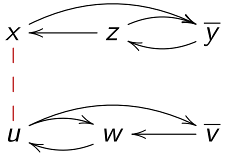
Non-Deterministic Automata

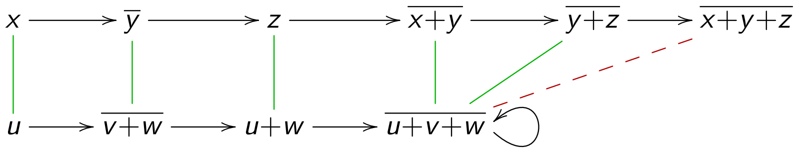
Use Hopcroft and Karp on the fly, through the powerset construction:



Non-Deterministic Automata

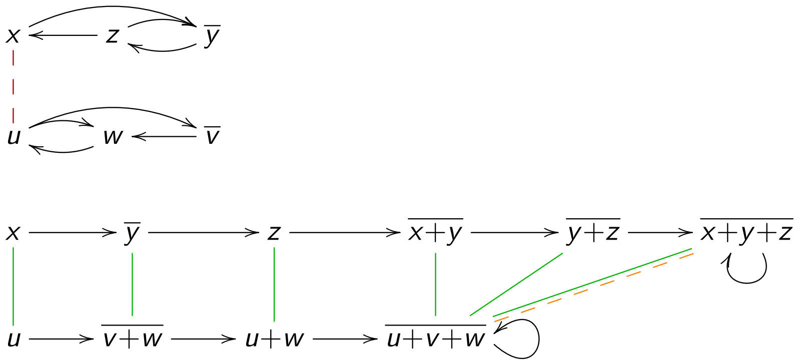
Use Hopcroft and Karp on the fly, through the powerset construction:





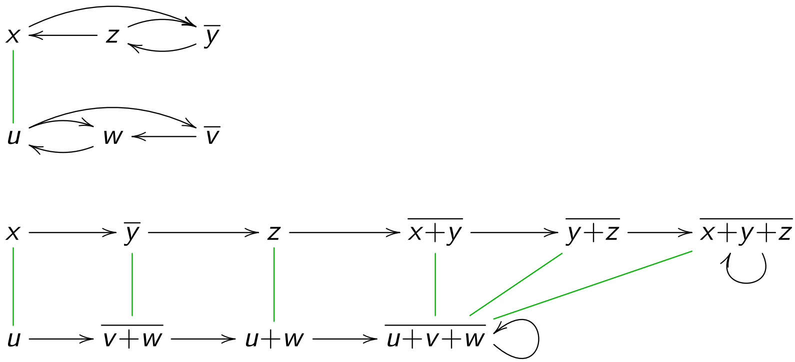
Non-Deterministic Automata

Use Hopcroft and Karp on the fly, through the powerset construction:



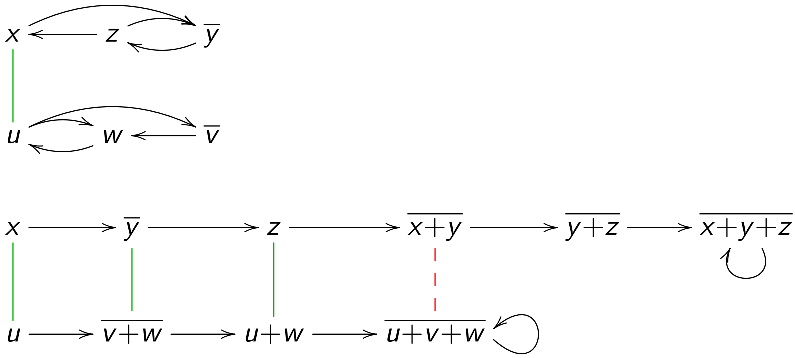
Non-Deterministic Automata

Use Hopcroft and Karp on the fly, through the powerset construction:



Non-Deterministic Automata

One can do better:



Non-Deterministic Automata

One can do better:

$$\begin{tikzcd}
x \arrow[rr, bend left] \arrow[d, dash] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}$$

$$\begin{array}{cl}
 & (x,\ u) \\
+ & (y,\ v{+}w) \\
\hline
= & (x{+}y,\ u{+}v{+}w)
\end{array}$$

$$\begin{tikzcd}
x \arrow[r] \arrow[d, dash] & \overline{y} \arrow[r] \arrow[d, dash] & z \arrow[r] \arrow[d, dash] & \overline{x+y} \arrow[r] \arrow[d, dashed, dash] & \overline{y+z} \arrow[r] & \overline{x+y+z} \arrow[loop below] \\
u \arrow[r] & \overline{v+w} \arrow[r] & u+w \arrow[r] & \overline{u+v+w} \arrow[loop right] & &
\end{tikzcd}$$

Non-Deterministic Automata

One can do better:

$$
\begin{tikzcd}
x \arrow[rr, bend left, ""] \arrow[d, dash] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}
$$

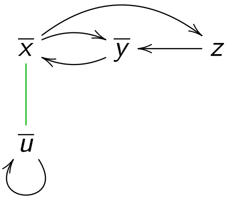
$$
\begin{array}{rl}
 & (x,\ u) \\
+ & (y,\ v{+}w) \\
\hline
= & (x{+}y,\ u{+}v{+}w)
\end{array}
$$

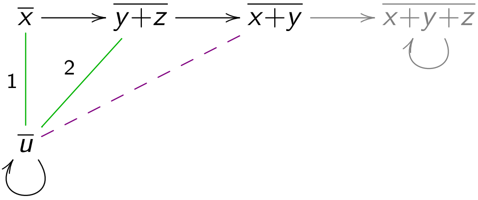
$$
\begin{tikzcd}
x \arrow[r] \arrow[d, dash] & \overline{y} \arrow[r] \arrow[d, dash] & z \arrow[r] \arrow[d, dash] & \overline{x+y} \arrow[r] \arrow[d, dash, dashed] & \overline{y+z} \arrow[r] & \overline{x+y+z} \arrow[loop below] \\
u \arrow[r] & \overline{v+w} \arrow[r] & u+w \arrow[r] & \overline{u+v+w} \arrow[loop right] & &
\end{tikzcd}
$$

using bisimulations up to union

Non-Deterministic Automata

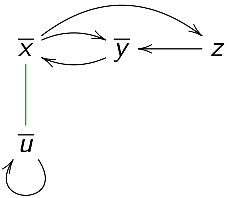
One can do even better:



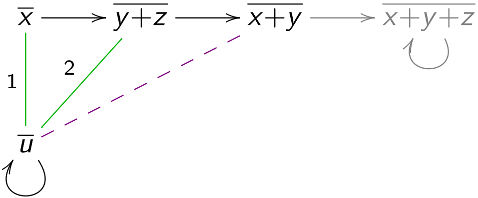


Non-Deterministic Automata

One can do even better:

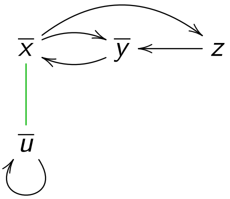


$$
\begin{aligned}
x+y &= u+y && (1)\\
&= y+z+y && (2)\\
&= y+z &&\\
&= u && (2)
\end{aligned}
$$

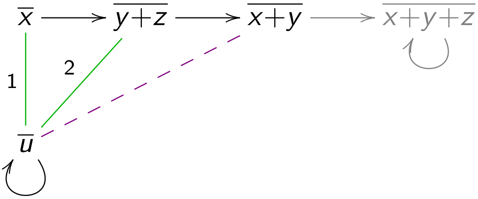


Non-Deterministic Automata

One can do even better:

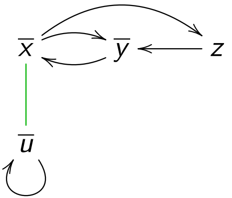


$$
\begin{aligned}
x+y &= u+y && (1)\\
&= y+z+y && (2)\\
&= y+z &&\\
&= u && (2)
\end{aligned}
$$

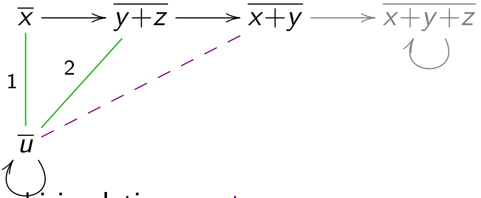


Non-Deterministic Automata

One can do even better:



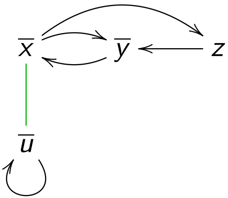
$$
\begin{aligned}
x+y &= u+y && (1)\\
&= y+z+y && (2)\\
&= y+z &&\\
&= u && (2)
\end{aligned}
$$



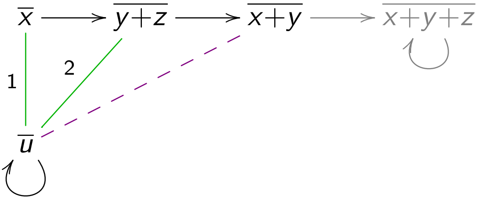
using bisimulations up to congruence

Non-Deterministic Automata

One can do even better:



$$
\begin{aligned}
x+y &= u+y && (1)\\
&= y+z+y && (2)\\
&= y+z &&\\
&= u && (2)
\end{aligned}
$$



this yield to the HKC algorithm [Bonchi, Pous'13]