Constant delay algorithms for regular document spanners

Fernando Florenzano

Cristian Riveros

Domagoj Vrgoč

From PUC Chile

Martín Ugarte

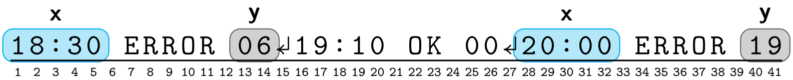
Stijn Vansummeren

From Université Libre de Bruxelles

Rule-based information extraction by example

```
18:30 ERROR 06
19:10 OK 00
20:00 ERROR 19
```

"Extract all pairs (time,id) of ERROR events"



Rule: RGX formula

$$\Sigma^* \cdot \mathbf{x}\{\delta\delta : \delta\delta\} \cdot \textvisiblespace\texttt{ERROR}\textvisiblespace \cdot \mathbf{y}\{\delta\delta\} \cdot \Sigma^*$$

$$\delta = (0 + 1 + \ldots + 9)$$

Output: mappings

x	**y**
$[1, 6\rangle$	$[13, 15\rangle$
$[28, 33\rangle$	$[40, 42\rangle$

Rule-based information extraction by example

Problem: Evaluation of rules in information extraction.
Input: RGX formula *R* and document *d*.
Output: Enumerate all mappings of *d* that satisfy *R*.

1	8	:	3	0		E	R	R	O	R		0	6	↲	1	9	:	1	0		O	K		0	0	↲	2	0	:	0	0		E	R	R	O	R		1	9
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41

Rule: RGX formula

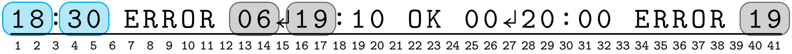
$$\Sigma^* \cdot \mathbf{x}\{\delta\delta : \delta\delta\} \cdot \text{␣}\texttt{ERROR}\text{␣} \cdot \mathbf{y}\{\delta\delta\} \cdot \Sigma^*$$

$$\delta = (0 + 1 + \ldots + 9)$$

Output: mappings

x	**y**
$[1, 6\rangle$	$[13, 15\rangle$
$[28, 33\rangle$	$[40, 42\rangle$

Unfortunately, the output can easily become exponential



```
18:30 ERROR 06↲19:10 OK 00↲20:00 ERROR 19
```

Rule: RGX formula

$$\Sigma^* \cdot \mathbf{x_1}\{\delta\delta\} \cdot \Sigma^* \cdot \mathbf{x_2}\{\delta\delta\} \cdot \Sigma^*$$

$$\delta \;=\; (0 + 1 + \ldots + 9)$$

Output: mappings

$\mathbf{x_1}$	$\mathbf{x_2}$
$[1,3\rangle$	$[4,6\rangle$
$[1,3\rangle$	$[13,15\rangle$
$\vdots$	$\vdots$
$[1,3\rangle$	$[40,42\rangle$
$[4,6\rangle$	$[13,15\rangle$
$[4,6\rangle$	$[16,18\rangle$
$\vdots$	$\vdots$

$\Theta(|d|^2)$

In general, a RGX formula with k variables
can have an output of size $\Theta(|d|^k)$.

Constant delay algorithms to the rescue

Definition

Given a RGX rule R and a document d,
a constant delay algorithm is a **two-phase** enumeration algorithm:

1. **Preprocessing** phase: linear in $|d|$ and, hopefully, linear in $|R|$.
2. **Enumeration** phase: constant time between two consecutive outputs.

Can we have an **efficient** constant delay algorithm for RGX formulas?

In this paper, we propose a constant delay algorithm for variable-set automata

Specifically, our contributions are:

1. We study the class of extended and deterministic variable-set automata.
2. We give a **simple** constant delay algorithm for deterministic functional extended variable-set automata.
3. We extend this algorithm for the full class of variable-set automata and spanner algebra.
4. We study the complexity of counting the number of output mappings.

In this talk: only the **main ideas** of the constant delay algorithm.

Outline

Variable-set automata and their variants

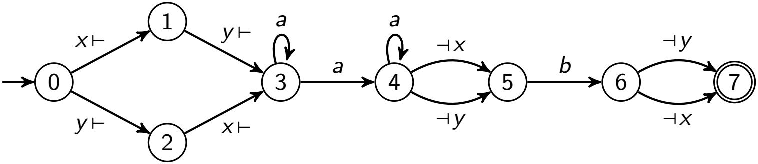
The constant delay algorithm

Outline

Variable-set automata and their variants

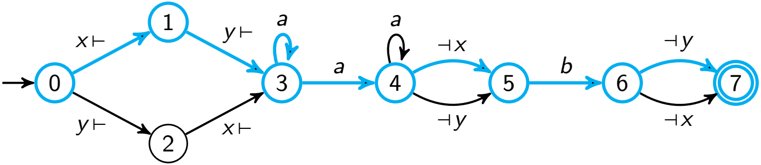
The constant delay algorithm

Variable-set automata (VA)



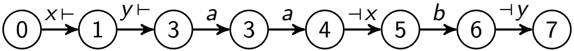
document: $\underline{\underset{1}{\texttt{a}}\,\underset{2}{\texttt{a}}\,\underset{3}{\texttt{b}}}$

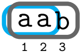
Variable-set automata (VA)



document: aab

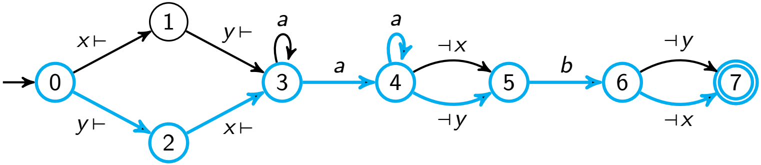
1 2 3





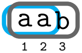
$x = [1, 3\rangle, y = [1, 4\rangle$

Variable-set automata (VA)



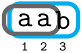
document: $\underline{\texttt{aab}}$
1 2 3

$$0 \xrightarrow{x\vdash} 1 \xrightarrow{y\vdash} 3 \xrightarrow{a} 3 \xrightarrow{a} 4 \xrightarrow{\dashv x} 5 \xrightarrow{b} 6 \xrightarrow{\dashv y} 7$$



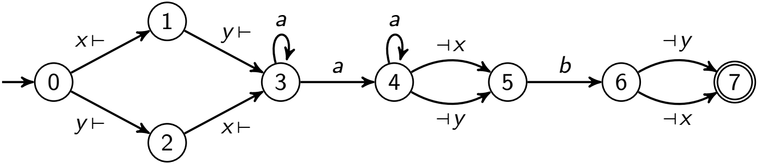
$x = [1,3\rangle, y = [1,4\rangle$

$$0 \xrightarrow{y\vdash} 2 \xrightarrow{x\vdash} 3 \xrightarrow{a} 4 \xrightarrow{a} 4 \xrightarrow{\dashv y} 5 \xrightarrow{b} 6 \xrightarrow{\dashv x} 7$$



$x = [1,4\rangle, y = [1,3\rangle$

Variable-set automata (VA)



document: $\underline{\mathtt{a}\,\mathtt{a}\,\mathtt{b}}$ $_{1\;2\;3}$

Theorem (Freydenberger17,MRV18)

The evaluation problem of variable-set automata is NP-complete.

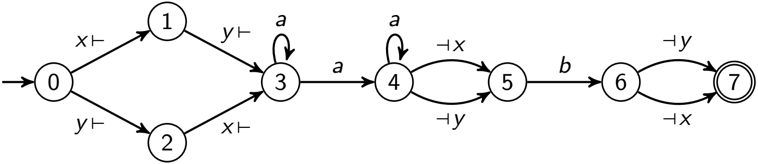
How do we restrict VA to have constant delay algorithms?

Problematic behaviors of VA and their classes

1. Functional VA
2. Extended VA
3. Deterministic VA

Problematic behaviors of VA and their classes

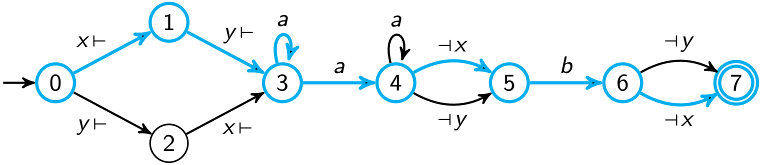
1. **Functional VA**
2. Extended VA
3. Deterministic VA



Problem: A VA can have accepting runs that are NOT valid.

Problematic behaviors of VA and their classes

1. **Functional VA**
2. Extended VA
3. Deterministic VA



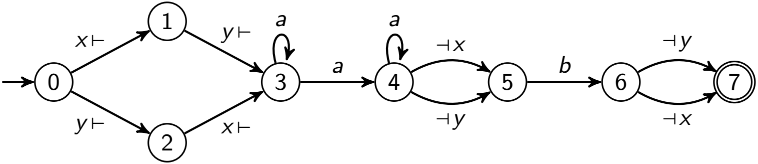
Problem: A VA can have accepting runs that are NOT valid.

Example of an accepting run that is not valid

$$0 \xrightarrow{x\vdash} 1 \xrightarrow{y\vdash} 3 \xrightarrow{a} 3 \xrightarrow{a} 4 \xrightarrow{\dashv x} 5 \xrightarrow{b} 6 \xrightarrow{\dashv x} 7$$ ✗

Problematic behaviors of VA and their classes

1. **Functional VA**
2. Extended VA
3. Deterministic VA

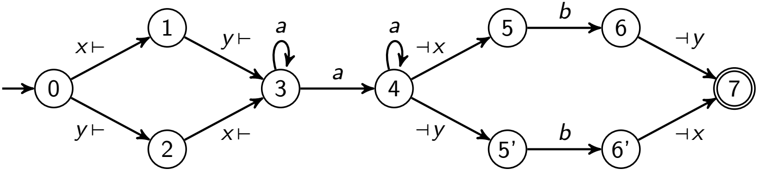


Definition: functional VA

A VA is functional if **every accepting run is a valid run.**

Problematic behaviors of VA and their classes

1. **Functional VA**
2. Extended VA
3. Deterministic VA



Definition: functional VA

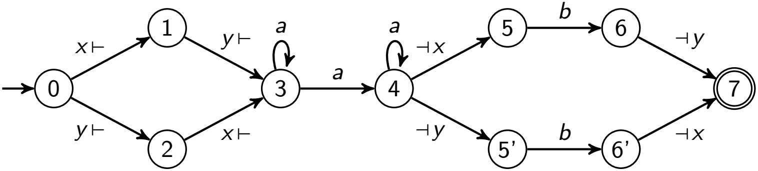
A VA is functional if **every accepting run is a valid run.**

Theorem (FKRV15)

Every VA is equivalent to a functional VA of at most exponential size.

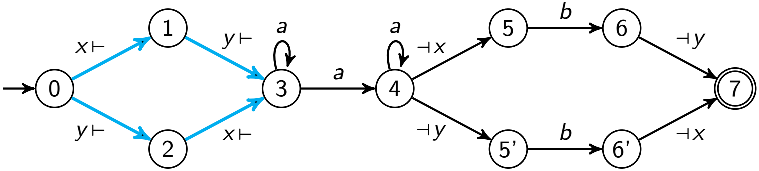
Problematic behaviors of VA and their classes

1. Functional VA 2. **Extended VA** 3. Deterministic VA



Problematic behaviors of VA and their classes

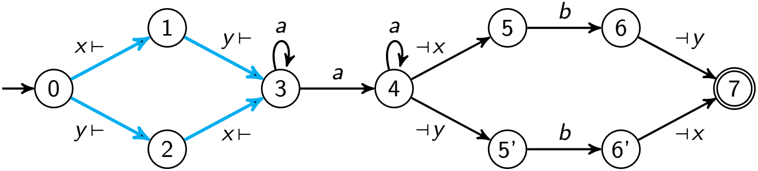
1. Functional VA
2. **Extended VA**
3. Deterministic VA



Problem: VA can use several paths of variables for the same extraction of spans.

Problematic behaviors of VA and their classes

1. Functional VA
2. **Extended VA**
3. Deterministic VA

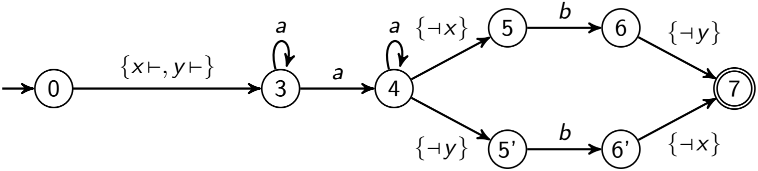


Definition: extended VA

An extended VA uses **transitions extended with sets of variables** such that between each pair of letters at most one of these transitions are used.

Problematic behaviors of VA and their classes

1. Functional VA
2. **Extended VA**
3. Deterministic VA



Definition: extended VA

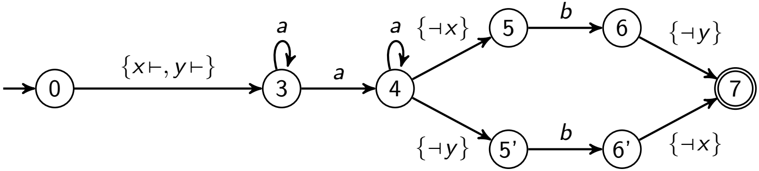
An extended VA uses **transitions extended with sets of variables** such that between each pair of letters at most one of these transitions are used.

Theorem

Every VA is equivalent to an extended VA of at most exponential size.

Problematic behaviors of VA and their classes

1. Functional VA
2. Extended VA
3. **Deterministic VA**



Problem: A VA can have several runs that witness the same output.

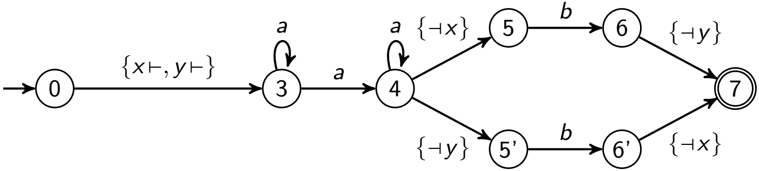
Example of several runs with the same input/output

$$0 \xrightarrow{\{x\vdash, y\vdash\}} 3 \xrightarrow{a} 3 \xrightarrow{a} 4 \xrightarrow{\{\dashv x\}} 5 \xrightarrow{b} 6 \xrightarrow{\{\dashv y\}} 7$$

$$0 \xrightarrow{\{x\vdash, y\vdash\}} 3 \xrightarrow{a} 4 \xrightarrow{a} 4 \xrightarrow{\{\dashv x\}} 5 \xrightarrow{b} 6 \xrightarrow{\{\dashv y\}} 7$$

Problematic behaviors of VA and their classes

1. Functional VA
2. Extended VA
3. **Deterministic VA**

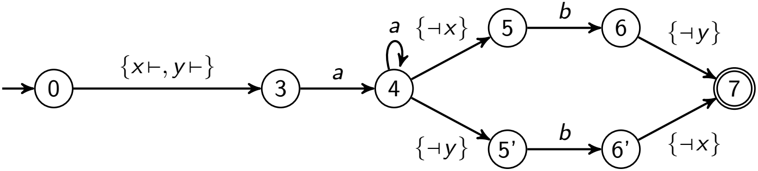


Definition: deterministic (Input/Output) VA

An extended VA is deterministic if the **transition relation is a function**.

Problematic behaviors of VA and their classes

1. Functional VA 2. Extended VA 3. **Deterministic VA**



Definition: deterministic (Input/Output) VA

An extended VA is deterministic if the **transition relation is a function**.

Theorem

Every extended VA is equivalent to a deterministic extended VA
of at most exponential size.

Outline

Variable-set automata and their variants

The constant delay algorithm

The constant delay algorithm for extended VA

Given an deterministic and functional extended VA $\mathcal{A} = (Q, q_0, F, \delta)$.

```
procedure $\textsc{Evaluate}(\mathcal{A},\ a_1 \dots a_n)$
    for all $q \in Q \setminus \{q_0\}$ do
        $list_q \leftarrow \epsilon$
    $list_{q_0} \leftarrow [\bot]$
    for $i := 1$ to $n$ do
        $\textsc{Capturing}(i)$
        $\textsc{Reading}(i)$
    $\textsc{Capturing}(n+1)$
    $\textsc{Enumerate}(\{list_q\}_{q \in Q}, F)$
```

```
procedure $\textsc{Capturing}(i)$
    for all $q \in Q$ do
        $list_q^{old} \leftarrow list_q.\mathtt{l}azycopy$
    for all $q \in Q$ with $list_q^{old} \neq \epsilon$ do
        for all $S \in \text{Markers}_\delta(q)$ do
            node $\leftarrow \mathtt{N}ode((S, i), list_q^{old})$
            $p \leftarrow \delta(q, S)$
            $list_p.\mathtt{a}dd(\text{node})$
```

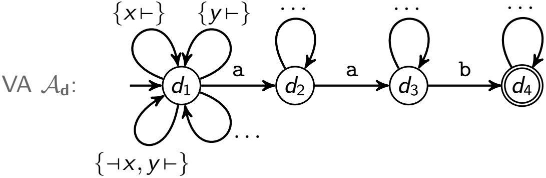
```
procedure $\textsc{Reading}(i)$
    for all $q \in Q$ do
        $list_q^{old} \leftarrow list_q$
        $list_q \leftarrow \epsilon$
    for all $q \in Q$ with $list_q^{old} \neq \epsilon$ do
        $p \leftarrow \delta(q, a_i)$
        $list_p.\mathtt{a}ppend(list_q^{old})$
```

Sketch idea of the constant delay algorithm in 3 steps

Given an deterministic and functional extended VA $\mathcal{A} = (Q, q_0, F, \delta)$.

1. Convert the document d into a deterministic extended VA $\mathcal{A}_d$.

document d: `aab`
1 2 3

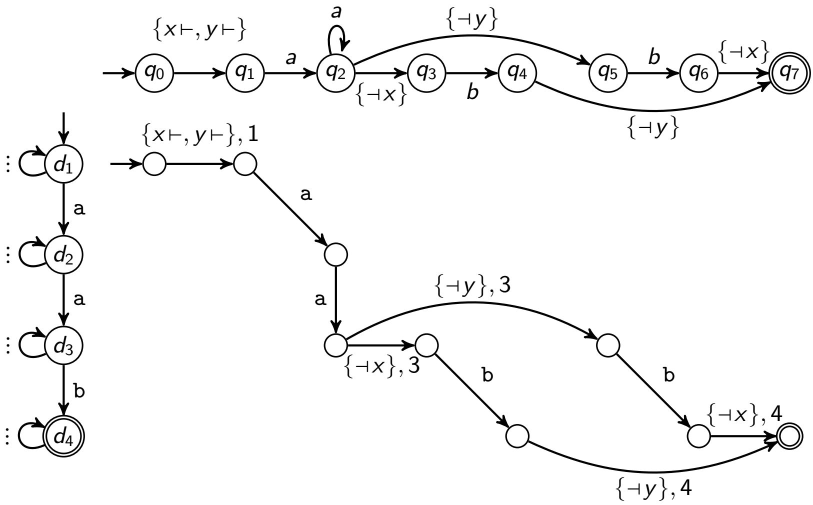


Sketch idea of the constant delay algorithm in 3 steps

Given an deterministic and functional extended VA $\mathcal{A} = (Q, q_0, F, \delta)$.

1. Convert the document d into a deterministic extended VA $\mathcal{A}_d$.
2. Build the product between $\mathcal{A}$ and $\mathcal{A}_d$, and annotate the variable transitions with the position of d where they take place.

2. Build the product between $\mathcal{A}$ and $\mathcal{A}_d$

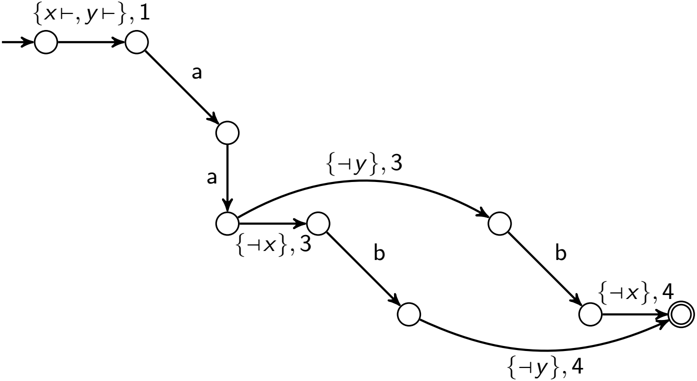


Sketch idea of the constant delay algorithm in 3 steps

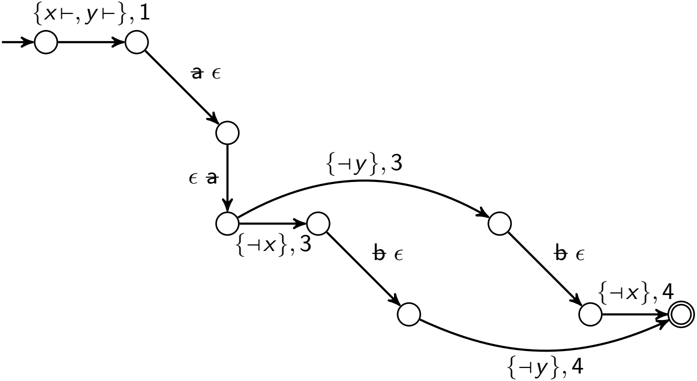
Given an deterministic and functional extended VA $\mathcal{A} = (Q, q_0, F, \delta)$.

1. Convert the document d into a deterministic eVA $\mathcal{A}_d$.
2. Build the product between $\mathcal{A}$ and $\mathcal{A}_d$, and annotate the variable transitions with the position of d where they take place.
3. Replace all the letters in the transitions of $\mathcal{A} \times \mathcal{A}_d$ with ε, and construct the "forward" ε-closure of the resulting graph.

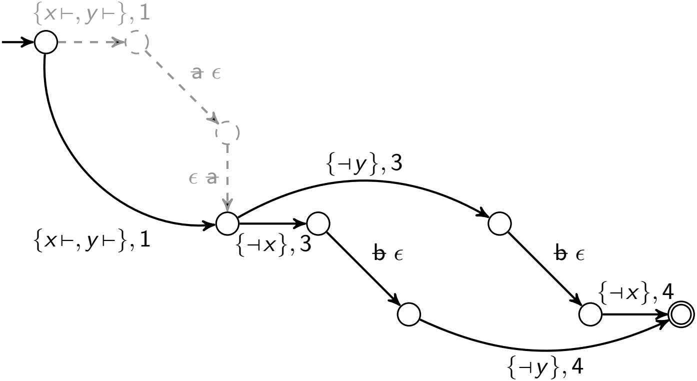
3. Replace all the letters with ϵ-transitions and construct the forward ε-closure



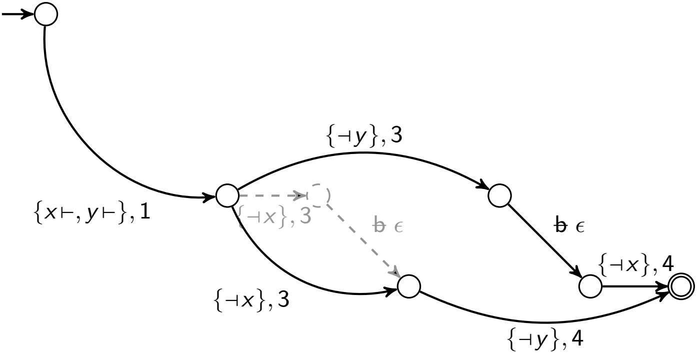
3. Replace all the letters with ϵ-transitions and construct the forward ε-closure



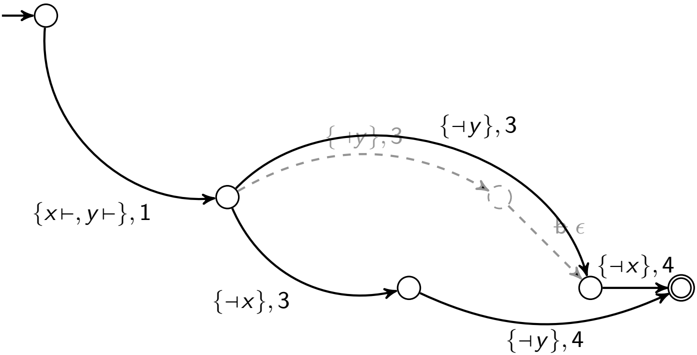
3. Replace all the letters with ϵ-transitions and construct the forward ε-closure



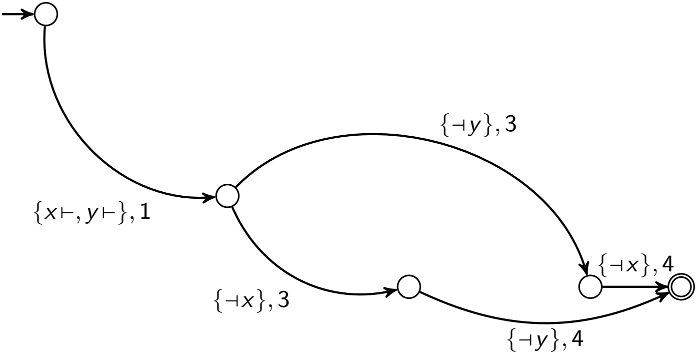
3. Replace all the letters with ϵ-transitions and construct the forward ε-closure



3. Replace all the letters with ϵ-transitions and construct the forward ε-closure



3. Replace all the letters with ϵ-transitions and construct the forward ε-closure



Given that the VA is **functional**, **extended** and **deterministic**:

- each path in the graph corresponds exactly to an output mapping, and
- every path is different (i.e. there are no duplicates).

Sketch idea of the constant delay algorithm in 3 steps

Given an deterministic and functional extended VA $\mathcal{A} = (Q, q_0, F, \delta)$.

1. Convert the document d into a deterministic eVA $\mathcal{A}_d$.
2. Build the product between $\mathcal{A}$ and $\mathcal{A}_d$, and annotate the variable transitions with the position of d where they take place.
3. Replace all the letters in the transitions of $\mathcal{A} \times \mathcal{A}_d$ with ε, and construct the "forward" ε-closure of the resulting graph.

Finally, we enumerate all paths from the resulting acyclic labeled graph which can easily be done with **constant delay** between outputs.

Efficiency of the constant delay algorithm

Given a VA $\mathcal{A}$ and a document d if:

$$\begin{aligned} n &= \#\text{states of } \mathcal{A} \\ m &= \#\text{transitions of } \mathcal{A} \\ l &= \#\text{number of variables of } \mathcal{A} \end{aligned}$$

Class of regular spanners	**Precomputation phase**
deterministic functional extended VA	$(n+m)\cdot\lvert d\rvert$
functional extended VA	$2^n\cdot m\cdot\lvert d\rvert$
functional VA / functional RGX	$2^n\cdot(n^2+\lvert\Sigma\rvert)\cdot\lvert d\rvert$
VA / RGX	$(2^n 5^\ell+2^n 3^\ell\lvert\Sigma\rvert)\cdot\lvert d\rvert$

In the paper, we give some evidences that
the exponential blow-up of functional extended VA seems unavoidable.

Conclusions and future work

- We provide a **simple** constant delay algorithm for evaluating deterministic functional extended VA.
- We extend this algorithm for the full class of variable-set automata and (also) regular spanner algebra.

Future work:

1. Code the algorithm and show that it works in practice.
2. Extend the algorithm to include other features used in rule-based information extraction.

Thanks!