Regular Expressions with Counting: Weak versus Strong Determinism

Wouter Gelade[a] Marc Gyssens[a] Wim Martens[b]

[a] Hasselt University, Belgium
[b] TU Dortmund, Germany

Introduction

Regular Expressions

Regular Expressions are used in a wide array of applications (Bioinformatics, Programming Languages, Model Checking, XML Schema Languages, etc.)

Introduction

Regular Expressions

Regular Expressions are used in a wide array of applications (Bioinformatics, Programming Languages, Model Checking, XML Schema Languages, etc.)

Standard Regular Expressions (REG(Σ))

- $\emptyset$, ε, and every $a \in \Sigma$ are in REG(Σ)
- for standard regular expressions r and s,
  rs, $r+s$, $r?$, and r^* are in REG(Σ)

Introduction

Regular Expressions

Regular Expressions are used in a wide array of applications (Bioinformatics, Programming Languages, Model Checking, XML Schema Languages, etc.)

Standard Regular Expressions (REG(Σ))

- $\emptyset$, ε, and every $a \in \Sigma$ are in REG(Σ)
- for standard regular expressions r and s,

  rs, $r + s$, $r?$, and r^* are in REG(Σ)

To keep users happy...
many applications add operators (counting, negation, intersection,...)

Introduction

To keep users happy...
many applications add operators (counting, negation, intersection,...)

Regular Expressions with Counting ($\mathrm{REG}^{\#}(\Sigma)$)

- All $\mathrm{REG}(\Sigma)$ are $\mathrm{REG}^{\#}(\Sigma)$
- If r is a $\mathrm{REG}^{\#}(\Sigma)$, then $r^{k,\ell}$ for $k \leq \ell \in \mathbb{N}$ is also a $\mathrm{REG}^{\#}(\Sigma)$

Introduction

To keep users happy...
many applications add operators (counting, negation, intersection,...)

Regular Expressions with Counting ($\mathrm{REG}^{\#}(\Sigma)$)

- All $\mathrm{REG}(\Sigma)$ are $\mathrm{REG}^{\#}(\Sigma)$
- If r is a $\mathrm{REG}^{\#}(\Sigma)$, then $r^{k,\ell}$ for $k \leq \ell \in \mathbb{N}$ is also a $\mathrm{REG}^{\#}(\Sigma)$

Example: $(ab)^{3,5}$
(matches *ababab*, *abababab*, and *ababababab*)

Introduction

To keep users happy...
many applications add operators (counting, negation, intersection,...)

Regular Expressions with Counting ($\mathrm{REG}^{\#}(\Sigma)$)

- All $\mathrm{REG}(\Sigma)$ are $\mathrm{REG}^{\#}(\Sigma)$
- If r is a $\mathrm{REG}^{\#}(\Sigma)$, then $r^{k,\ell}$ for $k \leq \ell \in \mathbb{N}$ is also a $\mathrm{REG}^{\#}(\Sigma)$

Example: $(ab)^{3,5}$
(matches *ababab*, *abababab*, and *ababababab*)

Counting is used in, e.g.,...

- XML Schema
- egrep
- Perl patterns

Outline

Deterministic Regular Expressions

Deterministic regular expressions exist to facilitate matching

(also called one-unambiguous regular expressions)

Deterministic Regular Expressions

Deterministic regular expressions exist to facilitate matching
(also called one-unambiguous regular expressions)

"When matching a string from left to right,
it's always clear which position in the expression to match next"

Deterministic Regular Expressions

Deterministic regular expressions exist to facilitate matching
(also called one-unambiguous regular expressions)

"When matching a string from left to right,
it's always clear which position in the expression to match next"

Example

- $c(a+b)^*a$ is not deterministic
- $cb^*a(b^*a)^*$ is deterministic and equivalent

Deterministic Regular Expressions

Deterministic regular expressions exist to facilitate matching
(also called one-unambiguous regular expressions)

"When matching a string from left to right,
it's always clear which position in the expression to match next"

Example

- $c(a+b)^*a$ is not deterministic
- $cb^*a(b^*a)^*$ is deterministic and equivalent

Deterministic expressions are used in, e.g., ...

- Document Type Definitions (DTD)
- SGML
- XML Schema

Strong and Weak Determinism

Weak determinism: what we just saw

Strong and Weak Determinism

Weak determinism: what we just saw

Strong determinism: weak determinism, plus
"It should also be clear which operator to use next"

Strong and Weak Determinism

Weak determinism: what we just saw

Strong determinism: weak determinism, plus
"It should also be clear which operator to use next"

Example

- $(a^*)^*$ is not strongly deterministic
- a^* is strongly deterministic and equivalent

Strong and Weak Determinism

Weak determinism: what we just saw

Strong determinism: weak determinism, plus
"It should also be clear which operator to use next"

Example

- $(a^*)^*$ is not strongly deterministic
- a^* is strongly deterministic and equivalent

Notation

- $\mathrm{DET}_S(\Sigma)$: strongly deterministic $\mathrm{REG}(\Sigma)$
- $\mathrm{DET}_W(\Sigma)$: weakly deterministic $\mathrm{REG}(\Sigma)$

Weak / Strong Determinism with Counting

Weak: "When matching a string from left to right,
it's always clear which position in the expression to match next"

Example

- $(b?a^{2,3})^{3,3}b$ is not weakly deterministic (witness: $aaaaaab\ldots$)
- $(b?a^{2,3})^{2,2}b$ is weakly deterministic

Weak / Strong Determinism with Counting

Weak: "When matching a string from left to right,
it's always clear which position in the expression to match next"

Example

- $(b?a^{2,3})^{3,3}b$ is not weakly deterministic (witness: $aaaaaab\ldots$)
- $(b?a^{2,3})^{2,2}b$ is weakly deterministic

Strong: "It should also be clear which operator to use next"

Weak / Strong Determinism with Counting

Weak: "When matching a string from left to right,
it's always clear which position in the expression to match next"

Example

- $(b?a^{2,3})^{3,3}b$ is not weakly deterministic (witness: $aaaaaab\ldots$)
- $(b?a^{2,3})^{2,2}b$ is weakly deterministic

Strong: "It should also be clear which operator to use next"

Example

- $(a^{1,2})^{3,4}$ is weakly deterministic, but not strongly deterministic
- $(a^{2,2})^{3,4}$ is weakly and strongly deterministic

Where are we going?

- XML Schema uses weakly deterministic expressions with counting

Where are we going?

- XML Schema uses weakly deterministic expressions with counting
- What do we know about these?

Where are we going?

- XML Schema uses weakly deterministic expressions with counting
- What do we know about these?
  - Does this class have a nice “deterministic” automata model?

Where are we going?

- XML Schema uses weakly deterministic expressions with counting
- What do we know about these?
  - Does this class have a nice "deterministic" automata model?
  - Is it decidable whether a regular language can be defined with a weakly deterministic expression with counting?

Where are we going?

- XML Schema uses weakly deterministic expressions with counting
- What do we know about these?
  - Does this class have a nice "deterministic" automata model?
  - Is it decidable whether a regular language can be defined with a weakly deterministic expression with counting?
  - What's the complexity for, e.g., membership, inclusion testing?

Where are we going?

- XML Schema uses weakly deterministic expressions with counting
- What do we know about these?
  - Does this class have a nice "deterministic" automata model?
  - Is it decidable whether a regular language can be defined with a weakly deterministic expression with counting?
  - What's the complexity for, e.g., membership, inclusion testing?

We'll see that weak and strong determinism are very different

in expressions with counting

Where are we going?

- XML Schema uses weakly deterministic expressions with counting
- What do we know about these?
  - Does this class have a nice "deterministic" automata model?
  - Is it decidable whether a regular language can be defined with a weakly deterministic expression with counting?
  - What's the complexity for, e.g., membership, inclusion testing?

We'll see that weak and strong determinism are very different

in expressions with counting

Do we want weak or strong determinism?

Outline

What you need to know about REG(Σ)

Theorem (Implicit in Brüggmann-Klein, 1993; Brügg.-Klein,Wood, 1998)

Expressive power in a picture:

$$DET_S(\Sigma) = DET_W(\Sigma) \subsetneq REG(\Sigma)$$

What you need to know about REG(Σ)

Theorem (Implicit in Brüggmann-Klein, 1993; Brügg.-Klein,Wood, 1998)

Expressive power in a picture:

$$DET_S(\Sigma) = DET_W(\Sigma) \subsetneq REG(\Sigma)$$

Theorem (Brüggemann-Klein and Wood, 1998)

Given expression r, deciding whether there exists a deterministic expression for $L(r)$ is in **EXPTIME**

What you need to know about REG(Σ)

Theorem (Implicit in Brüggmann-Klein, 1993; Brügg.-Klein,Wood, 1998)

Expressive power in a picture:

$$DET_S(\Sigma) = DET_W(\Sigma) \subsetneq REG(\Sigma)$$

Theorem (Brüggemann-Klein and Wood, 1998)

Given expression r, deciding whether there exists a deterministic expression for $L(r)$ is in **EXPTIME**

Theorem (Implicit in Brüggmann-Klein, 1993)

Every weakly deterministic expression can be made strongly deterministic in linear time

What you need to know

Brüggemann-Klein and Wood, 1998

Testing weak determinism of an expression is easy ($\mathcal{O}(n^2)$)

Core operation: Glushkov(r)

Consider $r = c(a+b)^*a$

What you need to know

Brüggemann-Klein and Wood, 1998

Testing weak determinism of an expression is easy ($\mathcal{O}(n^2)$)

Core operation: Glushkov(r)

Consider $r = c(a+b)^*a \rightsquigarrow c_1(a_2+b_3)^*a_4$

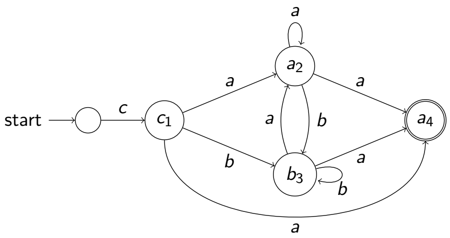
What you need to know

Brüggemann-Klein and Wood, 1998

Testing weak determinism of an expression is easy ($\mathcal{O}(n^2)$)

Core operation: Glushkov(r)

Consider $r = c(a+b)^*a \rightsquigarrow c_1(a_2+b_3)^*a_4$



Expression r is deterministic iff Glushkov(r) is a DFA

Complexity of (Weakly) Deterministic Expressions

- **MEMBERSHIP**: Given string w and expression r, is $w \in L(r)$?
- **INCLUSION**: Given expressions r_1, r_2, is $L(r_1) \subseteq L(r_2)$?
- **INTERSECTION**: Given expressions $r_1, \ldots, r_n$, is $\bigcap_i L(r_i) \neq \emptyset$?

Complexity of (Weakly) Deterministic Expressions

- **MEMBERSHIP**: Given string w and expression r, is $w \in L(r)$?
- **INCLUSION**: Given expressions r_1, r_2, is $L(r_1) \subseteq L(r_2)$?
- **INTERSECTION**: Given expressions $r_1, \ldots, r_n$, is $\bigcap_i L(r_i) \neq \emptyset$?

Theorem

For (weakly) deterministic expressions:

- **MEMBERSHIP** *is in* $\mathcal{O}(n^2)$
- **INCLUSION***: in* **PTIME** *[Stearns, Hunt 1981]*
- **INTERSECTION***:* **PSPACE***-complete*

[Mar., Neven, Schwentick 2004]

Questions

The Situation for deterministic expressions...

	Without counting	
Expressiveness	$\text{DET}_S(\Sigma) = \text{DET}_W(\Sigma) \subsetneq \text{REG}(\Sigma)$	
Succinctness	$\text{DET}_S(\Sigma) \approx \text{DET}_W(\Sigma)$	
Det-Test	easy (Glushkov)	
$\exists$-Det-Test	**EXPTIME**	
Membership	$\mathcal{O}(n^2)$	
Complexity	**PTIME**/ **PSPACE**	

Questions

The Situation for deterministic expressions...

	Without counting	With counting
Expressiveness	$\mathrm{DET}_S(\Sigma) = \mathrm{DET}_W(\Sigma) \subsetneq \mathrm{REG}(\Sigma)$	
Succinctness	$\mathrm{DET}_S(\Sigma) \approx \mathrm{DET}_W(\Sigma)$	
Det-Test	easy (Glushkov)	
$\exists$-Det-Test	**EXPTIME**	
Membership	$\mathcal{O}(n^2)$	
Complexity	**PTIME**/ **PSPACE**	

Questions

The Situation for deterministic expressions...

	Without counting	With counting
Expressiveness	$\mathrm{DET}_S(\Sigma) = \mathrm{DET}_W(\Sigma) \subsetneq \mathrm{REG}(\Sigma)$	???
Succinctness	$\mathrm{DET}_S(\Sigma) \approx \mathrm{DET}_W(\Sigma)$	???
Det-Test	easy (Glushkov)	??
∃-Det-Test	**EXPTIME**	???
Membership	$\mathcal{O}(n^2)$	??
Complexity	**PTIME**/ **PSPACE**	??

Outline

Outline

Expressive Power

Theorem

In terms of expressive power,
$DET_S(\Sigma) = DET_W(\Sigma) = DET_S^{\#}(\Sigma) \subsetneq DET_W^{\#}(\Sigma) \subsetneq REG(\Sigma)$ *(if* $|\Sigma| > 1$*)*

Expressive Power

Theorem

In terms of expressive power,

$$DET_S(\Sigma) = DET_W(\Sigma) = DET_S^{\#}(\Sigma) \subsetneq DET_W^{\#}(\Sigma) \subsetneq REG(\Sigma) \text{ (if } |\Sigma| > 1\text{)}$$

$$DET_S(\Sigma) = DET_W(\Sigma) = DET_S^{\#}(\Sigma) = DET_W^{\#}(\Sigma) \subsetneq REG(\Sigma) \text{ (if } |\Sigma| = 1\text{)}$$

Expressive Power

Theorem

In terms of expressive power,

$DET_S(\Sigma) = DET_W(\Sigma) = DET_S^{\#}(\Sigma) \subsetneq DET_W^{\#}(\Sigma) \subsetneq REG(\Sigma)$ *(if* $|\Sigma| > 1$*)*

$DET_S(\Sigma) = DET_W(\Sigma) = DET_S^{\#}(\Sigma) = DET_W^{\#}(\Sigma) \subsetneq REG(\Sigma)$ *(if* $|\Sigma| = 1$*)*

The equalities...

- $\mathrm{DET}_W(\Sigma) = \mathrm{DET}_S^{\#}(\Sigma)$ $(|\Sigma| > 1)$
- $\mathrm{DET}_W(\Sigma) = \mathrm{DET}_W^{\#}(\Sigma)$ $(|\Sigma| = 1)$

are non-trivial!

Witness separating languages:

- $(a^{2,3}b?)^*$ is in $\mathrm{DET}_W^{\#}(\Sigma)$, but not in $\mathrm{DET}_W(\Sigma)$
- $(aaa)^*(a + aa)$ is in $\mathrm{REG}(\Sigma)$, but not in $\mathrm{DET}_W(\Sigma)$

Outline

Succinctness

Theorem

In terms of succinctness,
$DET^{\#}_{W}(\Sigma)$ *is exponentially smaller than* $DET^{\#}_{S}(\Sigma)$

Succinctness

Theorem

In terms of succinctness,
$DET^{\#}_{W}(\Sigma)$ *is exponentially smaller than* $DET^{\#}_{S}(\Sigma)$

More precisely,
for every $n \in \mathbb{N}$, there's a $\mathrm{DET}^{\#}_{W}(\Sigma)$ r of size $\mathcal{O}(n)$ such that every $\mathrm{DET}^{\#}_{S}(\Sigma)$ for $L(r)$ is of size at least 2^n

Succinctness

Theorem

In terms of succinctness,
$DET_W^{\#}(\Sigma)$ *is exponentially smaller than* $DET_S^{\#}(\Sigma)$

More precisely,
for every $n \in \mathbb{N}$, there's a $\mathrm{DET}_W^{\#}(\Sigma)$ r of size $\mathcal{O}(n)$ such that every $\mathrm{DET}_S^{\#}(\Sigma)$ for $L(r)$ is of size at least 2^n

$r = (a^{2^n+1, 2^{n+1}})^{1,2}$
"all strings of as of length $2^n + 1$ until 2^{n+2}, but not of length $2^{n+1} + 1$"

Succinctness

Theorem

In terms of succinctness,
$\mathrm{DET}^{\#}_{W}(\Sigma)$ *is exponentially smaller than* $\mathrm{DET}^{\#}_{S}(\Sigma)$

More precisely,
for every $n \in \mathbb{N}$, there's a $\mathrm{DET}^{\#}_{W}(\Sigma)$ r of size $\mathscr{O}(n)$ such that every $\mathrm{DET}^{\#}_{S}(\Sigma)$ for $L(r)$ is of size at least 2^n

$r = (a^{2^n+1,2^{n+1}})^{1,2}$
"all strings of as of length $2^n + 1$ until 2^{n+2}, but not of length $2^{n+1} + 1$"

Corollary

The above theorem holds for unary languages

Outline

Counter Automata

Counter Automaton: $(Q, q_0, C, \delta, F, \tau)$

Ingredients:

- Q: states, q_0: initial state

Counter Automata

Counter Automaton: $(Q, q_0, C, \delta, F, \tau)$

Ingredients:

- Q: states, q_0: initial state
- C: counter variables

Counter Automata

Counter Automaton: $(Q, q_0, C, \delta, F, \tau)$

Ingredients:

- Q: states, q_0: initial state
- C: counter variables
- $\alpha : C \to \mathbb{N}$ assigns values to counters

Counter Automata

Counter Automaton: $(Q, q_0, C, \delta, F, \tau)$

Ingredients:

- Q: states, q_0: initial state
- C: counter variables
- $\alpha : C \to \mathbb{N}$ assigns values to counters
- Transitions are guarded: $\delta \subset Q \times \Sigma \times \text{Guard}(C) \times \text{Update}(C) \times Q$

Guard(C): Boolean combination of true, false, $c = k$, $c < k$
Update(C): set of statements $c{+}{+}$, reset(c)

Counter Automata

Counter Automaton: $(Q, q_0, C, \delta, F, \tau)$

Ingredients:

- Q: states, q_0: initial state
- C: counter variables
- $\alpha : C \to \mathbb{N}$ assigns values to counters
- Transitions are guarded: $\delta \subset Q \times \Sigma \times \text{Guard}(C) \times \text{Update}(C) \times Q$
- $F : Q \to \text{Guard}(C)$ acceptance function

Guard(C): Boolean combination of true, false, $c = k$, $c < k$
Update(C): set of statements $c{+}{+}$, reset(c)

Counter Automata

Counter Automaton: $(Q, q_0, C, \delta, F, \tau)$

Ingredients:

- Q: states, q_0: initial state
- C: counter variables
- $\alpha : C \to \mathbb{N}$ assigns values to counters
- Transitions are guarded: $\delta \subset Q \times \Sigma \times \text{Guard}(C) \times \text{Update}(C) \times Q$
- $F : Q \to \text{Guard}(C)$ acceptance function
- $\tau : C \to \mathbb{N}$ assigns maximum values to counters

Guard(C): Boolean combination of true, false, $c = k$, $c < k$
Update(C): set of statements $c{+}{+}$, reset(c)

Counter Automata

Counter Automaton: $(Q, q_0, C, \delta, F, \tau)$

Ingredients:

- Q: states, q_0: initial state
- C: counter variables
- $\alpha : C \to \mathbb{N}$ assigns values to counters
- Transitions are guarded: $\delta \subset Q \times \Sigma \times \text{Guard}(C) \times \text{Update}(C) \times Q$
- $F : Q \to \text{Guard}(C)$ acceptance function
- $\tau : C \to \mathbb{N}$ assigns maximum values to counters

Guard(C): Boolean combination of true, false, $c = k$, $c < k$
Update(C): set of statements $c{+}{+}$, reset(c)

Remark

These are very similar to [McQueen]

Counter Automata: Determinism

Configuration

(q, α), where q is a state, and $\alpha : C \to \mathbb{N}$

Counter Automata: Determinism

Configuration

(q, α), where q is a state, and $\alpha : C \rightarrow \mathbb{N}$

Determinism

- For every reachable configuration (q, α),

Counter Automata: Determinism

Configuration

(q, α), where q is a state, and $\alpha : C \longrightarrow \mathbb{N}$

Determinism

- For every reachable configuration (q, α),
- for every $a \in \Sigma$,

Counter Automata: Determinism

Configuration

(q, α), where q is a state, and $\alpha : C \to \mathbb{N}$

Determinism

- For every reachable configuration (q, α),
- for every $a \in \Sigma$,
- there's at most one transition (q, a, ϕ, π, q') with $\alpha \models \phi$

Counter Automata: Determinism

Configuration

(q, α), where q is a state, and $\alpha : C \rightarrow \mathbb{N}$

Determinism

- For every reachable configuration (q, α),
- for every $a \in \Sigma$,
- there's at most one transition (q, a, ϕ, π, q') with $\alpha \models \phi$

Note

Testing determinism is **PSPACE**-complete

From Expressions to Automata

We extend the Glushkov construction to $\mathrm{REG}^{\#}(\Sigma)$
Denote the construction by $\mathrm{Glushkov}^{\#}$

From Expressions to Automata

We extend the Glushkov construction to $\mathrm{REG}^{\#}(\Sigma)$
Denote the construction by $\mathrm{Glushkov}^{\#}$

Theorem

For every expression $r \in REG^{\#}(\Sigma)$,

- $L(r) = L(Glushkov^{\#}(r))$, *and*
- *r is strongly deterministic $\Leftrightarrow$ $Glushkov^{\#}(r)$ is deterministic*

Outline

Testing Determinism

Theorem

- *Testing weak determinism for* $REG^{\#}(\Sigma)$ *is in time* $\mathcal{O}(n^3)$
  (Kilpeläinen and Tuhkanen 2007)

Testing Determinism

Theorem

- *Testing weak determinism for* $REG^{\#}(\Sigma)$ *is in time* $\mathcal{O}(n^3)$
  (Kilpeläinen and Tuhkanen 2007)

Theorem

Testing strong determinism for $REG^{\#}(\Sigma)$ *is in time* $\mathcal{O}(n^3)$

Inclusion, Intersection

What should we expect? To put you in the right mood

Theorem (Gelade, Mar., Neven 2007)

- **INCLUSION** *for* $REG^{\#}(\Sigma)$ *is* **EXPSPACE**-*complete*
- **INTERSECTION** *for* $REG^{\#}(\Sigma)$ *is* **PSPACE**-*complete*

Inclusion, Intersection

What should we expect? To put you in the right mood

Theorem (Gelade, Mar., Neven 2007)

- **INCLUSION** *for* $REG^{\#}(\Sigma)$ *is* **EXPSPACE**-*complete*
- **INTERSECTION** *for* $REG^{\#}(\Sigma)$ *is* **PSPACE**-*complete*

Theorem

- **INCLUSION** *for* $DET^{\#}_{S}(\Sigma)$ *is in* **PSPACE** *(from automata)*
- **INTERSECTION** *for* $DET^{\#}_{S}(\Sigma)$ *and* $DET^{\#}_{W}(\Sigma)$ *is* **PSPACE**-*complete*
- **MEMBERSHIP** *for* $DET^{\#}_{S}(\Sigma)$ *is in* $\mathcal{O}(n^3)$ *(from automata)*

Outline

Concluding Remarks

The Situation for deterministic expressions. . .

	Without counting	
Expressiveness	$\mathrm{DET}_S(\Sigma) = \mathrm{DET}_W(\Sigma)$	
Succinctness	$\mathrm{DET}_S(\Sigma) \approx \mathrm{DET}_W(\Sigma)$	
Det-Test	easy (Glushkov)	
$\exists$-Det-Test	**EXPTIME**	
Membership	$\mathcal{O}(n^2)$	
Complexity	**PTIME**/ **PSPACE**	

Concluding Remarks

The Situation for deterministic expressions...

	Without counting	With counting
Expressiveness	$\text{DET}_S(\Sigma) = \text{DET}_W(\Sigma)$	$\text{DET}^{\#}_S(\Sigma) \subsetneq \text{DET}^{\#}_W(\Sigma)$
Succinctness	$\text{DET}_S(\Sigma) \approx \text{DET}_W(\Sigma)$	strong $>_{\text{exp}}$ weak
Det-Test	easy (Glushkov)	easy (+ Glushkov$^{\#}$)
$\exists$-Det-Test	**EXPTIME**	2**EXPTIME** (strong)
Membership	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
Complexity	**PTIME**/ **PSPACE**	**PSPACE** (strong)

Where are we going?

- XML Schema uses weakly deterministic expressions with counting
- What do we know about these?
  - Does this class have a nice “deterministic” automata model?
  - Is it decidable whether a regular language can be defined with a weakly deterministic expression with counting?
  - What’s the complexity for, e.g., membership, inclusion testing?

Weak and strong determinism are very different

in expressions with counting

Do we want weak or strong determinism?

Thank you for listening