Relative edit-distance problem between input-driven pushdown languages

Hyunjoon Cheon[1] Yo-Sub Han[1] Sang-Ki Ko[2] Kai Salomaa[3]

[1]Department of Computer Science, Yonsei University, Republic of Korea

[2]AI Research Center, Korea Electronics Technology Institute, Republic of Korea

[3]School of Computing, Queen's University, Canada

DLT 2019
August 6th, 2019
Warsaw, Poland

Overview

Overview

Backgrounds

Inclusion problems

Relative edit-distance problems

XML Error detection

When we use an XML document...

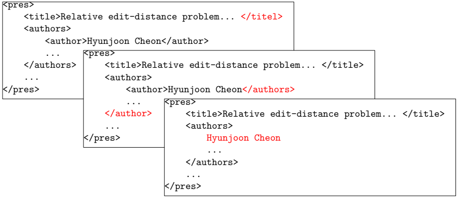
- Open new context while reading a opening tag
- Read texts
- Close the current context while reading a closing tag

```
<pres>
    <title>Relative edit-distance problem... </title>
    <authors>
        <author>Hyunjoon Cheon</author>
        ...
    </authors>
    ...
</pres>
```

XML Error detection

When we use an XML document...

- Open new context while reading a opening tag
- Read texts
- Close the current context while reading a closing tag

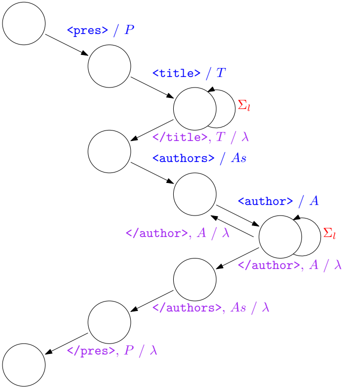


How different is a set of documents from XML schema?
$\rightarrow$ Relative edit-distance problem of Input-driven languages

Input-driven pushdown automata

An *Input-driven pushdown automaton* (IDPDA) is a restricted form of a PDA that

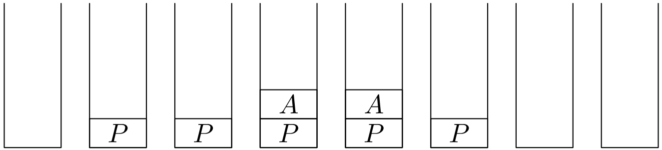
1. pushes a symbol into the stack while reading a call symbol,
2. pops a symbol from the stack if it exists while reading a return symbol,
3. reads a local symbol without modifying the stack.



Input-driven pushdown automata

Definition

An IDPDA $A = (Q, \Sigma, \Gamma, \delta, s, F)$ has its alphabet and transitions, each of which belongs to one of call, return and local sets.



$$s \xrightarrow{<} p \xrightarrow{a} q \xrightarrow{<} r \xrightarrow{b} s \xrightarrow{>} p \xrightarrow{>} p \xrightarrow{>} t$$

Input-driven pushdown automata

Definition

We define each transition set to be:

- a set δ_c of call transitions over $(Q \times \Sigma_c) \times (Q \times \Gamma)$,
- a set δ_r of return transitions over $(Q \times \Sigma_r \times \Gamma_\perp) \times Q$,
- a set δ_l of local transitions over $(Q \times \Sigma_l) \times Q$,

where $\perp$ represents an empty stack and $\Gamma_\perp = \Gamma \cup \{\perp\}$.

Known properties for IDPDAs

	Emptiness	Universality	Complement
NFA	NL[1]	PSPACE[2]	EXPTIME[3]
IDPDA[4]	**P**	**EXPTIME**	**EXPTIME**
PDA[5]	P	Undec.	Undec.

[1] Jones, Space-bounded reducibility among combinatorial problems, Journal of Computer and System Sciences, 1975

[2] Meyer and Stockmeyer, The equivalence problem for regular expression with squaring requires exponential space, 13th symp. on Switching and Automata Theory, 1972

[3] Sakoda and Sipser, Nondeterminism and the size of two way finite automata, 10th ACM symp. on Theory of computing, 1978.

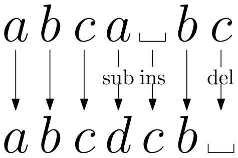
[4] Alur and Madhusudan, Visibly pushdown languages, 16th ACM symp. on Theory of computing., 2004.

[5] Hopcroft, Motwani and Ullman, Introduction to Language, Automata and Computation (2nd Ed.), 2000.

Edit-distance

Definition

The *edit-distance* $d(x, y)$ between two strings x and y is the minimum number of edit operations to change x into y (or vice versa).



Example

$d(a, b) = 1$, $d(a\underline{bcabc}, \underline{bcabc}a) = 2$.

Edit-distance over languages

Definition

We define the edit-distance between two languages L_1 and L_2 to be:

$$d(L_1, L_2) = \min_{x \in L_1, y \in L_2} d(x, y).$$

Example

L_1 \ L_2	<a>	<ab>	<a<>>
ab	2	2	4
<b>	1	1	3

$$d(L_1, L_2) = d(L_2, L_1) = 1$$

Relative edit-distance

Definition

The maximum value of the edit-distance from every string in L_1 (called "source") to L_2 (called "target").

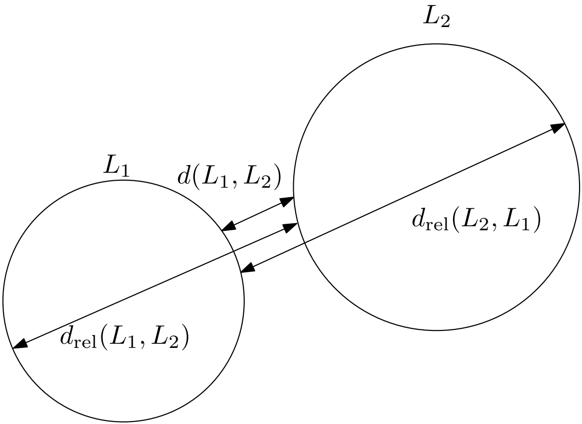
$$d_{\mathsf{rel}}(L_1, L_2) = \sup_{w_1 \in L_1} \inf_{w_2 \in L_2} d(w_1, w_2)$$

Example

L_1 \ L_2	<a>	<ab>	<a<>>
ab	2	2	4
<b>	1	1	3

$$d_{\mathsf{rel}}(L_1, L_2) = 2, \quad d_{\mathsf{rel}}(L_2, L_1) = 3$$

Relative edit-distance



A visual representation of the edit-distance and the relative variant

Proposed problem

Problem (Inclusion problem)

Given two languages L_1 and L_2, determine whether or not $L_1 \subseteq L_2$.

Problem (Relative edit-distance problem)

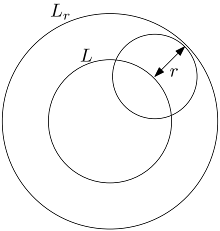
Given two languages L_1 and L_2 and a fixed integer r, determine whether or not $d_{\mathit{rel}}(L_1, L_2) \leq r$.

Neighborhood languages

Definition

Given a language L, distance metric d and radius r, the *radius* r *neighborhood language* of L on d is,

$$L_r = \{w \in \Sigma^* \mid (\exists x \in L)\, d(w, x) \leq r\}.$$



A visual representation of a neighbourhood language

Overview

Inclusion problem

Definition

Given two languages L_1 and L_2, the inclusion problem from L_1 to L_2 is deciding whether or not $L_1 \subseteq L_2$.

Inclusion from NFA to $NIDPDA$

Theorem

Given an NFA A and an NIDPDA B, deciding whether or not $L(A) \subseteq L(B)$ is EXPTIME-complete.

Inclusion from NFA to $NIDPDA$

Proof of EXPTIME upperbound.

- $L(A) \subseteq L(B) \iff L(A) \cap L(B)^c = \emptyset$.
- Complementation of an NIDPDA is done in EXPTIME due to determinization. [**Alur2004**]
- Intersection emptiness can be done in polynomial time.

□

Proof of EXPTIME lowerbound.

Universality problem for NIDPDAs ($\Sigma^* \subseteq L(B)$) is EXPTIME-complete.

□

Inclusion from $(D)IDPDA$ to $DIDPDA$

Theorem

Given an NIDPDA A and a DIDPDA B, deciding whether or not $L(A) \subseteq L(B)$ can be solved in polynomial time.

Proof.

- It is well-known that complementing B is easy; exchange the set of final states and the set of non-final states.
- Intersection emptiness between $L(A)$ and $L(B)^c$ can be done in polynomial time.

$\square$

Inclusion from $(D)IDPDA$ to NFA

Theorem

Given an NIDPDA (or a DIDPDA) A and an NFA B, deciding whether or not $L(A) \subseteq L(B)$ is EXPTIME*-complete.*

Proof strategy

- EXPTIME upper bound: Find a complement of B
- EXPTIME lower bound: Show a reduction from linear-space ATM membership test, which is EXPTIME-c.

Inclusion from $(D)IDPDA$ to NFA: Upper bound

Theorem

Given a DIDPDA A and an NFA B, deciding whether or not $L(A) \subseteq L(B)$ is in EXPTIME*.*

Proof.

- $L(A) \subseteq L(B)$ is equivalent to $L(A) \cap L(B)^c = \emptyset$.
- Complementing NFA can be done in EXPTIME.
- Intersection between DIDPDA and NFA can be done in polynomial time in their sizes.
- Testing emptiness on NIDPDA can also be done in polynomial time in its size.

$\square$

Inclusion from $(D)IDPDA$ to NFA: ATM

Definition (Alternating Turing Machine [**Chandra1976**])

An *alternating Turing machine* (ATM) $M = (Q, \Sigma, \delta, q_0, g)$ has:

- a set Q of states,
- a alphabet Σ,
- a transition function $\delta : Q \times \Sigma \to 2^{Q\times\Sigma\times\{L,R\}}$,
- a initial state $q_0 \in Q$,
- a type mapping function $g : Q \to \{\vee, \wedge, \text{accept}, \text{reject}\}$.

Inclusion from $(D)IDPDA$ to NFA: ATM

Definition (Accepting configuration)

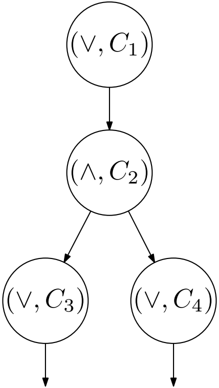
A configuration C is *accepting* on an ATM M iff:

- C is on an accept state ($g =$ accept),
- C is on an existential state ($g = \vee$) and one of its next configurations is accepting,
- C is on a universal state ($g = \wedge$) and all of its next configurations are accepting.

Inclusion from $(D)IDPDA$ to NFA: ATM

Definition (Computation tree)

A computation tree of a configuration C_1 on an ATM M is a list of C_1's possible accepting configurations, in a form of tree.



Inclusion from $(D)IDPDA$ to NFA: Lower bound

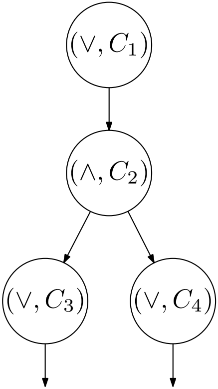
Proof strategy for EXPTIME lower bound.

We prove it by using reduction from the membership test on a linear-space ATM with an input w.

1. Encode the computation tree of the given input.
2. Construct an NFA B that rejects the strings matching with some valid criteria (defined later slides).
3. Construct an NIDPDA A that accepts the strings matching with other valid criteria.
4. Test if there exists an accepting computation in $L(A) \cap L(B)^c$.
   $L(A) \cap L(B)^c = \emptyset \iff L(A) \subseteq L(B)$.

Inclusion from $(D)IDPDA$ to NFA: Lower bound

1. Encoding a computation tree into a string.



Encode such computation tree into a string.:

$$C_1(\underline{C_2^R \$_l} C_3(\ldots)\$_r \overline{C_4}(\ldots))$$

Note that the underlined symbols and '(' are call symbols, and overlined symbols and ')' are return symbols.

Inclusion from $(D)IDPDA$ to NFA: Lower bound

1. Encoding a computation tree into a string.

$$C_1(\underline{C_2^R}\$_l C_3(\ldots)\$_r\overline{C_4}(\ldots))$$

Valid conditions:

- The string encodes a tree (Its structure is valid).
- The initial configuration, C_1, is q_0w.
- The successive configurations are all valid.
- Every final configuration is accepting.

Inclusion from $(D)IDPDA$ to NFA: Lower bound

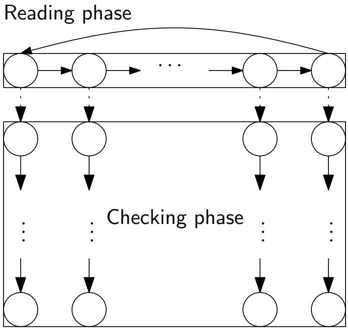
2. Constructing an NFA.

The NFA accepts the strings that satisfy one of the following conditions:

- C_1 is not the initial configuration,
- at least one of final configurations (that is followed by ')') is not an accepting configuration or
- one of computations $C_1 \to C_2$, $C_2 \to C_3$ is invalid.

Inclusion from $(D)IDPDA$ to NFA: Lower bound

2. Constructing an NFA.



Note that such NFA can have exactly one nondeterministic transition on all of its accepting computations.

Inclusion from $(D)IDPDA$ to NFA: Lower bound

2. Constructing an NFA.

$$
\begin{array}{rl}
 & w_1 w_2 \ldots \underline{w_{i-1} q w_i} w_{i+1} \cdots \\
\text{(move L)} & w_1 w_2 \ldots \underline{q' w_{i-1} w'_i} w_{i+1} \ldots \\
\text{(move R)} & w_1 w_2 \ldots \underline{w_{i-1} w'_i q'} w_{i+1} \ldots
\end{array}
$$

Since each transition changes at most three characters in sequence, it is enough to check their matching after n symbols using an NFA.

Inclusion from $(D)IDPDA$ to NFA: Lower bound

3. Constructing an NIDPDA.

The NIDPDA accepts the string such that

- computation $C_2 \to C_4$ is correct and
- it encodes a tree.

Inclusion from $(D)IDPDA$ to NFA: Lower bound

4. Showing equivalence.
   - $L(A) \cap L(B)^c = \emptyset$ means that the linear-space ATM does not accept the configuration C_1.
   - $L(A) \cap L(B)^c = \emptyset \iff L(A) \subseteq L(B)$, which is the inclusion problem.

Therefore, the problem is EXPTIME-hard. □

Theorem

Given an (D)IDPDA A and an NFA B, deciding whether $L(A) \subseteq L(B)$ is EXPTIME-complete.

Inclusion from $DPDA$ to $DIDPDA$

Theorem

The problem is undecidable.

Proof.

Reduction from TM emptiness.

- For a Turing machine M, we can encode its computation in the form of $w_1 \# w_2^R \# \ldots$.
- The DPDA A checks that w_1 is initial and the transitions from even to odd configuration are valid.
- The DIDPDA B checks that the transitions from odd to even configuration are valid.
- $L(A) \cap L(B)$ is the set of all valid computations of M.
- $L(A) \cap L(B) = \emptyset \iff L(A) \subseteq L(B)^c$.

$\square$

Inclusion from $DIDPDA$ to $DPDA$

Theorem

The problem is undecidable.

Proof.

Similar to the previous proof; swap the roles of DPDA and DIDPDA. □

Overview

Relative edit-distance problem

Definition

Given two languages L_1 and L_2, a fixed integer $r \in \mathbb{N}$, the relative edit-distance problem is deciding whether or not $d_{\mathsf{rel}}(L_1, L_2) \leq r$.

Relative problem from $NIDPDA$ to NFA

Theorem

Given an (D)IDPDA A and an (D)FA B, the relative edit-distance problem from $L(A)$ to $L(B)$ is EXPTIME*-complete.*

Proof strategy

- EXPTIME upper bound: Design the neighbourhood of $L(B)$.
- EXPTIME lower bound: Show a reduction from the inclusion problem from DIDPDA to NFA.

Relative problem from $NIDPDA$ to NFA

Proof of EXPTIME upperbound.

1. We can construct an NFA B_r for the neighbourhood of $L(B)$ in polynomial time in the size of B. [**NgRS15DLT**; **Povarov07**]
2. $L(A) \subseteq L(B_r)$, which is the inclusion problem from an NIDPDA to an NFA, is EXPTIME-complete.

Therefore, the problem is decidable in EXPTIME. $\square$

Relative problem from $NIDPDA$ to NFA

Proof of EXPTIME lowerbound.

1. On the previous reduction of the membership test on a linear-space ATM, we have a DIDPDA A and an NFA B.
2. Substitute the symbol of the nondeterministic transitions on B to a unique symbol to make the computation deterministic.
3. The language for DFA B_D has relative edit-distance 1 from $L(A)$, i.e. $d_{\mathsf{rel}}(L(A), L(B_D)) \leq 1$.

Therefore, the problem is EXPTIME-hard. □

Relative problem from $NIDPDA$ to NFA

We have proved that

- given an NIDPDA A and an NFA B, the problem is EXPTIME,
- given a DIDPDA A and a DFA B, the problem is EXPTIME-hard.

Therefore, the relative edit-distance problem from an (D)IDPDA to an (D)FA is EXPTIME-complete.

Relative problem in $NIDPDA$

Theorem

Given an NIDPDAs A and B, the relative edit-distance problem from $L(A)$ to $L(B)$ is EXPTIME*-complete.*

Proof strategy

- EXPTIME upper bound: Design the neighbourhood of $L(B)$.
- EXPTIME lower bound: Reduction from the inclusion problem in NIDPDAs.

Neighbourhood of $NIDPDA$

We need to construct the radius r neighbourhood of an NIDPDA to show its upper bound.

Theorem

Let an NIDPDA A has n states and m stack symbols. The neighbourhood of $L(A)$ of radius 1 can be recognized by an NIDPDA B with $O(mn^2)$ states and $n+1$ stack symbols.

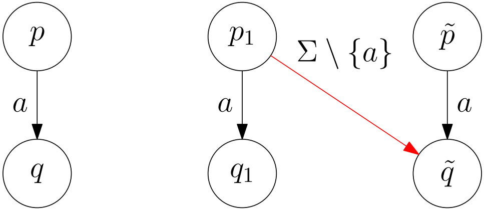
Insertion and Deletion.

Due to Okhotin and Salomaa [**OkhotinS19**], we know that the neighbourhood automaton has $O(nm)$ states and $m+1$ stack symbols..

Neighbourhood of $NIDPDA$

Substitution of symbols of the same type.

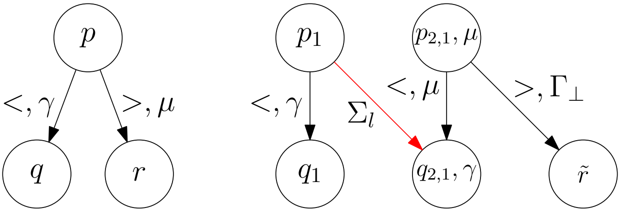
- Make two copies Q_1 and $\tilde{Q}$ of Q.
- Copy every transition from Q to Q_1.
- Add new transitions from Q_1 to $\tilde{Q}$ according to the original transition.



Neighbourhood of $NIDPDA$

Substitution from a call symbol to a local symbol.

- Make two copies $Q_{2,1}$ of $Q1$.
- Add new transitions from $Q1$ to $Q_{2,1} \times \Gamma$ by the substitution.



For other substitution cases between non-local and local, we use

- $Q_{2,1} \times \Gamma$ for local to return.
- $Q_{2,2} \times \Gamma_\perp$ for return to local or local to call, starts from $(s_{2,2}, \perp)$.

Neighbourhood of $NIDPDA$

Substitution from a call symbol to a return symbol.

Since the difference of stack height before and after the substitution is 2, we use $Q_{3,1} \times \Gamma^2$ in this case.
For the case of substitution from a return to a call, we use $Q_{3,2} \times \Gamma_\perp^2$ instead of that.

Neighbourhood of $NIDPDA$

Construction summary.

The neighbourhood IDPDA $B = (Q_B, \Sigma, \Gamma, \delta, I_B, F_B)$ consists of:

- $Q_B = Q_1 \cup (Q_{2,1} \times \Gamma) \cup (Q_{2,2} \times \Gamma_\perp) \cup (Q_{3,1} \times \Gamma^2) \cup (Q_{3,2} \times \Gamma_\perp^2) \cup \tilde{Q}$,
- $I_B = \{q_1, (q_{2,2}, \perp), (q_{3,2}, \perp\perp) \mid q \in I\}$,
- $F_B = \{\tilde{q}, (q_{2,1}, \gamma), (q_{3,1}, \gamma\mu) \mid q \in F, \gamma, \mu \in \Gamma\}$.

It uses $O(nm^2)$ states and a new stack symbol $\perp$. $\square$

Neighbourhood of $NIDPDA$

Theorem

Given an NIDPDA A with n states and m stack symbols, the radius r neighbourhood of $L(A)$ on edit-distance with fixed r has $n \cdot (m + r)^{2r}$ states, which is a polynomial with respect to the size of A.

Proof.

Iterating the previous construction for a neighbourhood IDPDA. $\square$

Relative problem in $NIDPDA$

Theorem

Given two NIDPDAs A and B, the relative edit-distance problem from $L(A)$ to $L(B)$ is EXPTIME-complete.

Proof strategy

- EXPTIME upper bound: Design the neighbourhood of $L(B)$.
- EXPTIME lower bound: Reduction from the inclusion problem in NIDPDAs.

Relative problem in $NIDPDA$

Proof of EXPTIME upperbound.

- The neighbourhood NIDPDA B_r has polynomially many states when we have a fixed r.
- The complement of B_r has an exponential number of states.

Therefore, the problem is decidable in EXPTIME. □

Relative problem in $NIDPDA$

Proof of EXPTIME lowerbound.

- $d_{\mathsf{rel}}(L(A), L(B)) \leq 0 \iff L(A) \subseteq L(B)$
- The inclusion problem between two NIDPDAs is EXPTIME-complete [**Alur2004**].

Therefore, the problem is EXPTIME-hard. $\square$

Relative problem in $NIDPDA$

We show that the relative distance problem between two NIDPDAs is both EXPTIME and EXPTIME-hard.
Therefore, this problem is EXPTIME-complete.

Completing the problem table

Theorem

Given a DPDA A and a DIDPDA B, the relative edit-distance problem from $L(A)$ to $L(B)$ is undecidable.

Proof.

Due to the fact that the inclusion problem from DPDA to DIDPDA is undecidable. □

What we have showed

We proved the inclusion problem

- from NFAs to NIDPDAs to be EXPTIME-complete,
- from NIDPDAs to NFAs to be EXPTIME-complete,
- between DPDAs and DIDPDAs to be undecidable.

We also showed the relative edit-distance problem

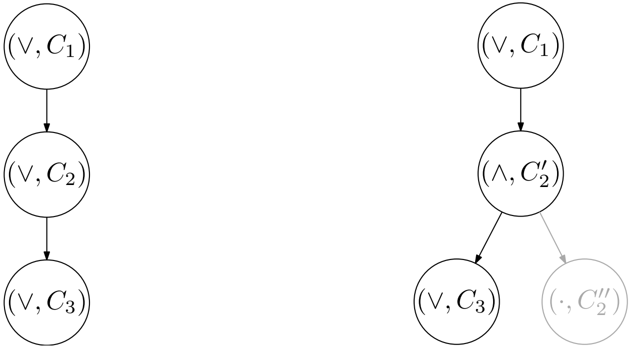
- from NIDPDAs to NFAs to be EXPTIME-complete,
- between two NIDPDAs to be EXPTIME-complete,
- from DPDAs to DIDPDAs to be undecidable.

Open problems

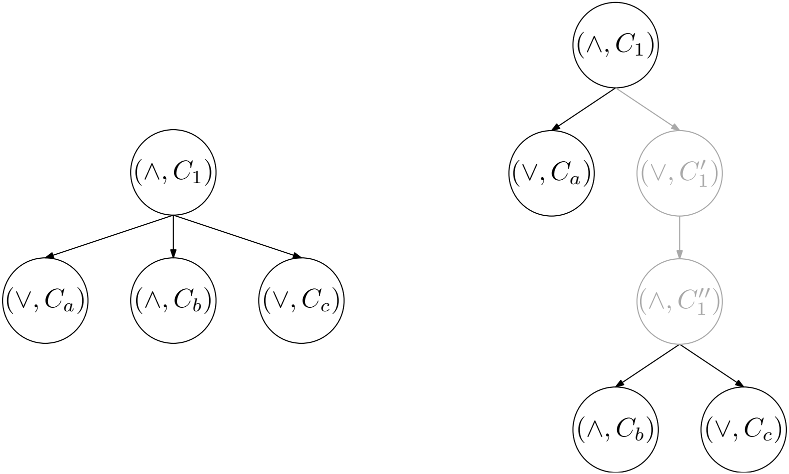
- The exact complexity of relative edit-distance problems without fixing r.
  - For instance of the relative problem between two NIDPDAs, its upper bound of complexity is 2EXPTIME on binary r by using the given proof.
- An approximation algorithm that computes the relative edit-distance between two languages efficiently.
- The relative edit-distance problem for other classes of languages.

Thank you

Normalizing an ATM computation tree



Normalizing an ATM computation tree



Complete summary

<table>
<tr><th>L_1 \ L_2</th><th>DFA</th><th>NFA</th><th>DIDPDA</th><th>IDPDA</th><th>DPDA</th><th>PDA</th></tr>
<tr><td>(D)FA</td><td rowspan="3">P</td><td>PSPACE-c.</td><td rowspan="2">P</td><td rowspan="2">EXPTIME-c.</td><td>P</td><td rowspan="3">Undec.</td></tr>
<tr><td>(D)IDPDA</td><td rowspan="2">EXPTIME-c.</td><td rowspan="2">Undec.</td></tr>
<tr><td>(D)PDA</td><td colspan="2">Undec.</td></tr>
</table>

The complexities of deciding $L_1 \subseteq L_2$.

<table>
<tr><th>L_1 \ L_2</th><th>(D)FA</th><th>(D)IDPDA</th><th>(D)PDA</th></tr>
<tr><td>(D)FA</td><td>PSPACE-c</td><td rowspan="2">EXPTIME-c.</td><td rowspan="3">Undec.</td></tr>
<tr><td>(D)IDPDA</td><td rowspan="2">EXPTIME-c.</td></tr>
<tr><td>(D)PDA</td><td>Undec.</td></tr>
</table>

The complexities of deciding $d_{\mathsf{rel}}(L_1, L_2) \leq r$.