The double-sided Dyck language is not a non-branching ℓMCFL.

Paul Gallot[1] Sylvain Salvati[2] Makoto Kanazawa[3]

[1]Université de Lille
Inria Lille
Agence Nationale pour la Recherche

[2]Université de Lille
Inria Lille
Agence Nationale pour la Recherche

[3]Hosei University

December 7, 2018

Summary

- Definitions
- Proof of the result
- Consequences

Double-sided Dyck language O_1

The double-sided Dyck language is the commutative closure of the Dyck language one one pair of parenthesis:

$$O_1 = \{u \in \{a, b\}^* \mid |u|_a = |u|_b\}$$

Double-sided Dyck language O_1

The double-sided Dyck language is the commutative closure of the Dyck language one one pair of parenthesis:

$$O_1 = \{u \in \{a, b\}^* \mid |u|_a = |u|_b\}$$

We can also define it as $O_1 = \mathrm{Scramble}((ab)^*)$.

Double-sided Dyck language O_1

The double-sided Dyck language is the commutative closure of the Dyck language one one pair of parenthesis:

$$O_1 = \{u \in \{a, b\}^* \mid |u|_a = |u|_b\}$$

We can also define it as $O_1 = \mathrm{Scramble}((ab)^*)$.

Also called the mix language.

Non-branching Linear Multiple Context-Free Grammar

A non-branching ℓMCFG G is a tuple (N, Σ, P, S)

Non-branching Linear Multiple Context-Free Grammar

A non-branching ℓMCFG G is a tuple (N, Σ, P, S)

For example $N = \{S^{(1)}, A^{(2)}\}$, $\Sigma = \{a, b\}$

Non-branching Linear Multiple Context-Free Grammar

A non-branching ℓMCFG G is a tuple (N, Σ, P, S)

For example $N = \{S^{(1)}, A^{(2)}\}$, $\Sigma = \{a, b\}$
and $P = \{$

$$S \to A_1 A_2$$

$$A_1, A_2 \to aA_1, aA_2$$

$$A_1, A_2 \to bA_1, bA_2$$

$$A_1, A_2 \to \varepsilon, \varepsilon$$

$\}$

Non-branching Linear Multiple Context-Free Grammar

A non-branching ℓMCFG G is a tuple (N, Σ, P, S)

For example $N = \{S^{(1)}, A^{(2)}\}$, $\Sigma = \{a, b\}$
and $P = \{$

$$S \to A_1 A_2$$

$$A_1, A_2 \to aA_1, aA_2$$

$$A_1, A_2 \to bA_1, bA_2$$

$$A_1, A_2 \to \varepsilon, \varepsilon$$

$\}$

$$L = \{w\, w \mid w \in \{a, b\}^*\}$$

Example of run

We derive the word *baba*:

$$S$$

Example of run

We derive the word *baba*:

$$S \to A_1 A_2$$

Example of run

We derive the word *baba*:

$$S \to A_1 A_2 \to bA_1 bA_2$$

Example of run

We derive the word *baba*:

$$S \to A_1 A_2 \to bA_1 bA_2 \to baA_1 baA_2$$

Example of run

We derive the word *baba*:

$$S \to A_1 A_2 \to bA_1 bA_2 \to baA_1 baA_2 \to baba$$

Non-branching Linear Multiple Context-Free Grammars

A non-branching ℓMCFG is a tuple (N, Σ, P, S) where:

Non-branching Linear Multiple Context-Free Grammars

A non-branching ℓMCFG is a tuple (N, Σ, P, S) where:

Every non-terminal has a fixed multiplicity.

Non-branching Linear Multiple Context-Free Grammars

A non-branching ℓMCFG is a tuple (N, Σ, P, S) where:

Every non-terminal has a fixed multiplicity.

Rules are of the form :

$$A_1, \ldots, A_n \rightarrow u_1, \ldots, u_n$$

where there is a non terminal $B^{(m)}$ such that $u_1 \ldots u_n$ is a word over $\Sigma \cup \{B_1, \ldots, B_m\}$

Non-branching Linear Multiple Context-Free Grammars

A non-branching ℓMCFG is a tuple (N, Σ, P, S) where:

Every non-terminal has a fixed multiplicity.

Rules are of the form :

$$A_1, \ldots, A_n \rightarrow u_1, \ldots, u_n$$

where there is a non terminal $B^{(m)}$ such that $u_1 \ldots u_n$ is a word over $\Sigma \cup \{B_1, \ldots, B_m\}$

each B_i appears at most once in each rule (linearity)

Non-branching Linear Multiple Context-Free Grammars

A non-branching ℓMCFG is a tuple (N, Σ, P, S) where:

Every non-terminal has a fixed multiplicity.

Rules are of the form :

$$A_1, \ldots, A_n \to u_1, \ldots, u_n$$

where there is a non terminal $B^{(m)}$ such that $u_1 \ldots u_n$ is a word over $\Sigma \cup \{B_1, \ldots, B_m\}$

each B_i appears at most once in each rule (linearity) and we can normalise grammars so that each B_i appears exactly once.

Normalisation of the grammar

Assume there exists a non-branching ℓMCFG G,

Normalisation of the grammar

Assume there exists a non-branching ℓMCFG G,

we normalise G so that:

Normalisation of the grammar

Assume there exists a non-branching ℓMCFG G,

we normalise G so that:

- the non-terminals are $S^{(1)}$ and $A^{(n)}$ where n is the maximum multiplicity of a non-terminal of G,

Normalisation of the grammar

Assume there exists a non-branching ℓMCFG G,

we normalise G so that:

- the non-terminals are $S^{(1)}$ and $A^{(n)}$ where n is the maximum multiplicity of a non-terminal of G,
- each rule writes exactly one a and one b

Normalisation of the grammar

Assume there exists a non-branching ℓMCFG G,

we normalise G so that:

- the non-terminals are $S^{(1)}$ and $A^{(n)}$ where n is the maximum multiplicity of a non-terminal of G,
- each rule writes exactly one a and one b (except for $S \to \varepsilon$).

Case $n = 1$

$$S \rightarrow abS$$

$$S \rightarrow baS$$

$$S \rightarrow aSb$$

$$S \rightarrow bSa$$

$$S \rightarrow Sab$$

$$S \rightarrow Sba$$

$$S \rightarrow \varepsilon$$

Counter-example word for $n = 1$

$$w_1 = aabbbbaa$$

Counter-example word for $n = 1$

$$w_1 = aabbbbaa$$

No rule for S can apply.

Derivation of w_1 with multiplicity 2

$$S$$

Derivation of w_1 with multiplicity 2

$$S \to A_1 ab A_2$$

Derivation of w_1 with multiplicity 2

$$S \to A_1 abA_2 \to A_1 aabbA_2$$

Derivation of w_1 with multiplicity 2

$$S \to A_1 abA_2 \to A_1 aabbA_2 \to A_1 aabbbA_2 a$$

Derivation of w_1 with multiplicity 2

$$S \to A_1 ab A_2 \to A_1 aabb A_2 \to A_1 aabbb A_2 a$$

$$\to A_1 aabbbb A_2 aa$$

Derivation of w_1 with multiplicity 2

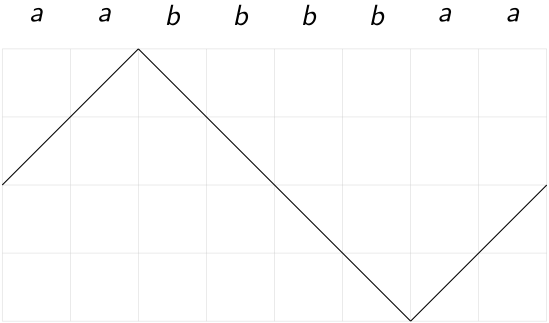
$$S \to A_1 ab A_2 \to A_1 aabb A_2 \to A_1 aabbb A_2 a$$

$$\to A_1 aabbbb A_2 aa \to aabbbbaa$$

Derivation of w_1 with multiplicity 2

$$S \to A_1 ab A_2 \to A_1 aabb A_2 \to A_1 aabbb A_2 a$$

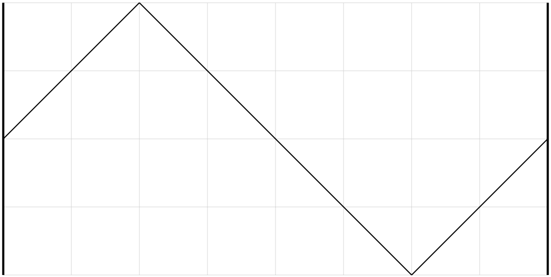
$$\to A_1 aabbbb A_2 aa \to aabbbbaa$$



Derivation of w_1 with multiplicity 2

$$S \to A_1 ab A_2 \to A_1 aabb A_2 \to A_1 aabbb A_2 a$$

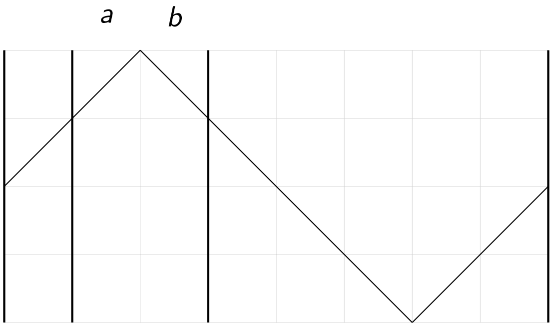
$$\to A_1 aabbbb A_2 aa \to aabbbbaa$$



Derivation of w_1 with multiplicity 2

$$S \to A_1 abA_2 \to A_1 aabbA_2 \to A_1 aabbbA_2 a$$

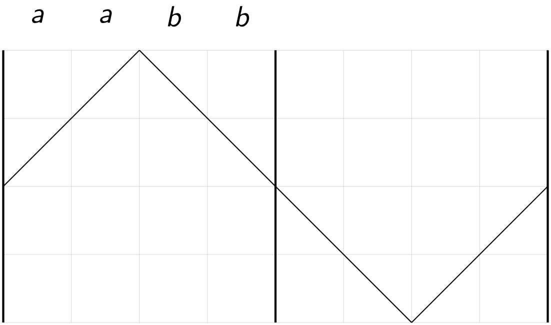
$$\to A_1 aabbbbA_2 aa \to aabbbbaa$$



Derivation of w_1 with multiplicity 2

$$S \to A_1abA_2 \to A_1aabbA_2 \to A_1aabbbA_2a$$

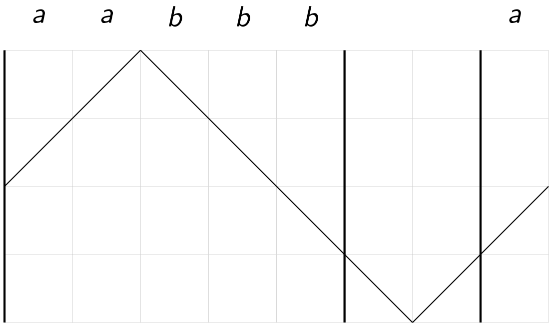
$$\to A_1aabbbbA_2aa \to aabbbbaa$$



Derivation of w_1 with multiplicity 2

$$S \to A_1abA_2 \to A_1aabbA_2 \to A_1aabbbA_2a$$

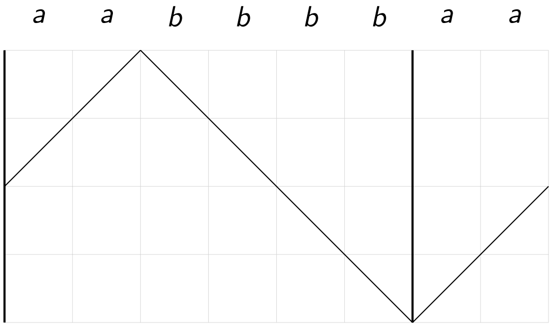
$$\to A_1aabbbbA_2aa \to aabbbbaa$$



Derivation of w_1 with multiplicity 2

$$S \to A_1 abA_2 \to A_1 aabbA_2 \to A_1 aabbbA_2 a$$

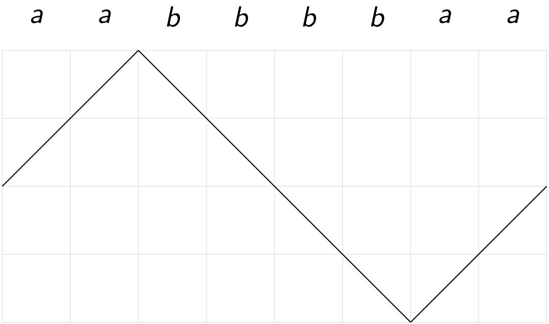
$$\to A_1 aabbbbA_2 aa \to aabbbbaa$$



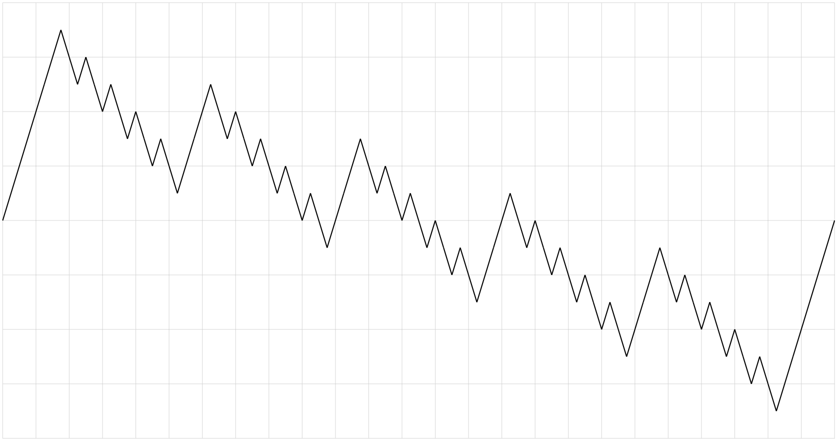
Derivation of w_1 with multiplicity 2

$$S \to A_1 abA_2 \to A_1 aabbA_2 \to A_1 aabbbA_2 a$$

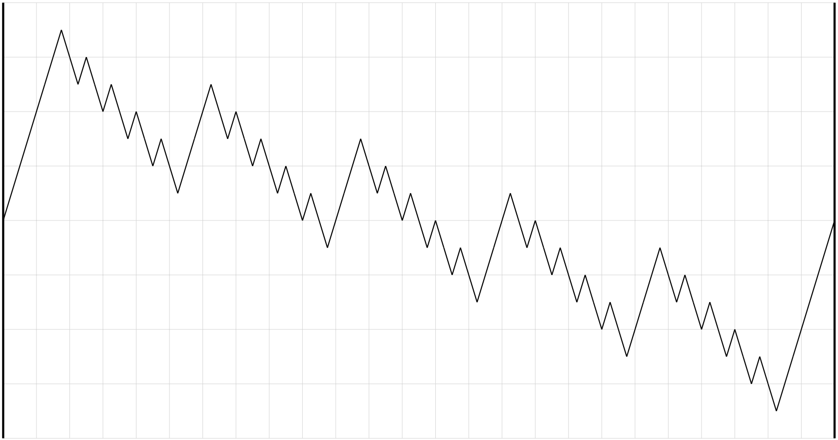
$$\to A_1 aabbbbA_2 aa \to aabbbbaa$$



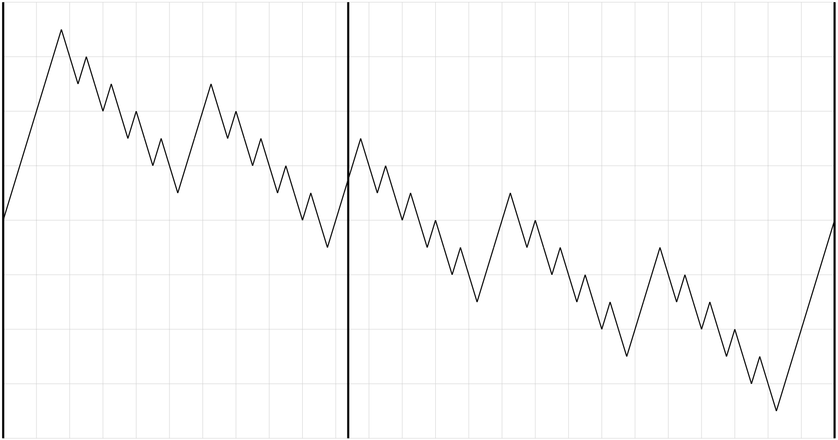
Counter-example word for $k = 2$



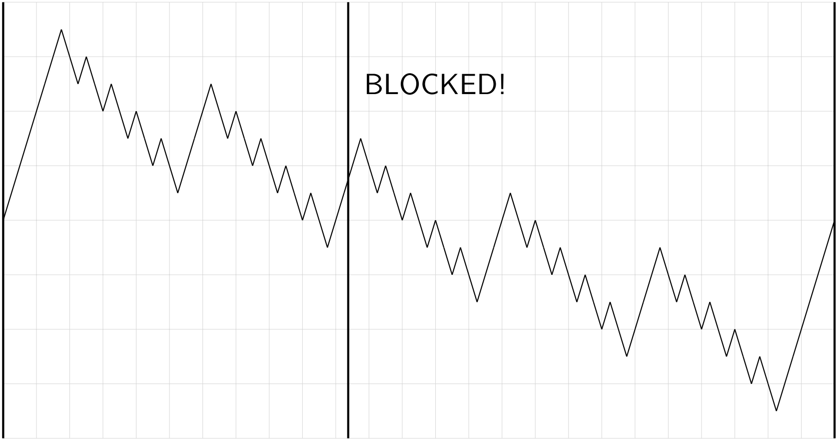
Counter-example word for $k = 2$



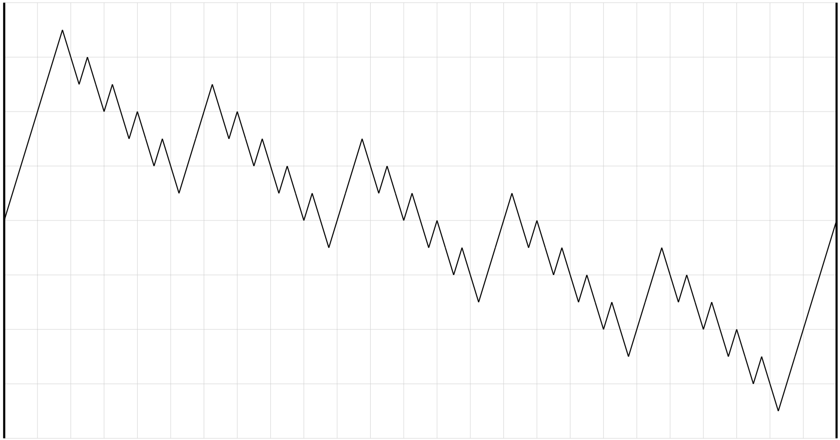
Counter-example word for $k = 2$



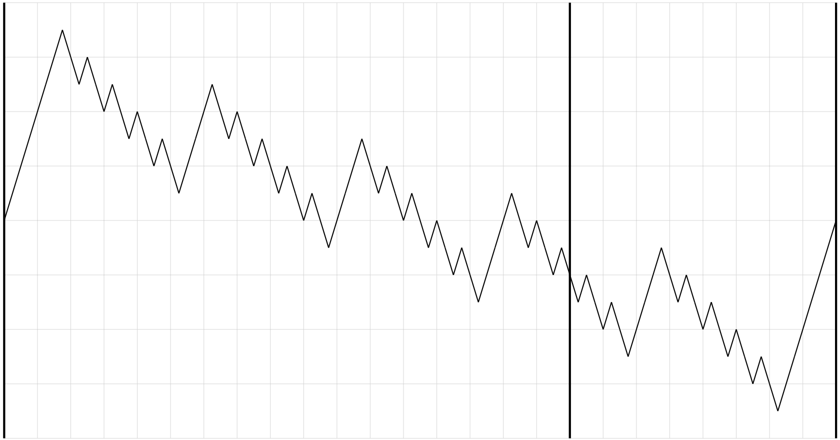
Counter-example word for $k = 2$



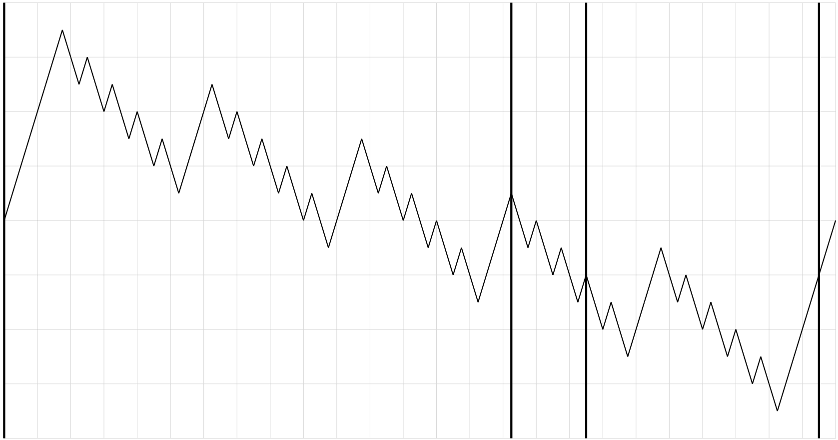
Counter-example word for $k = 2$



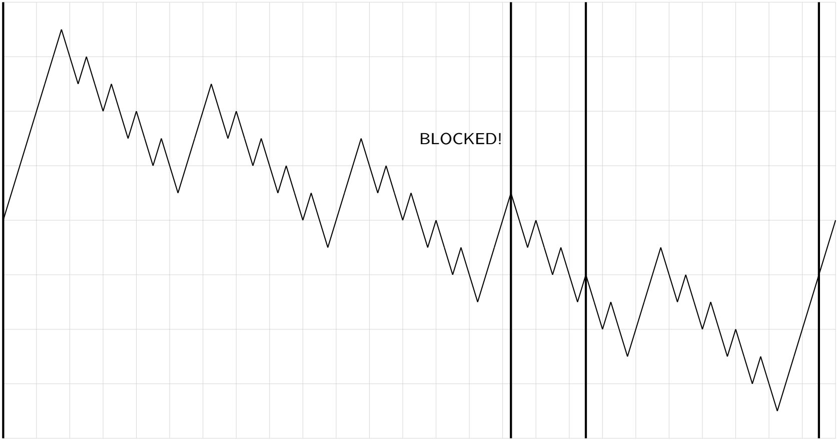
Counter-example word for $k = 2$



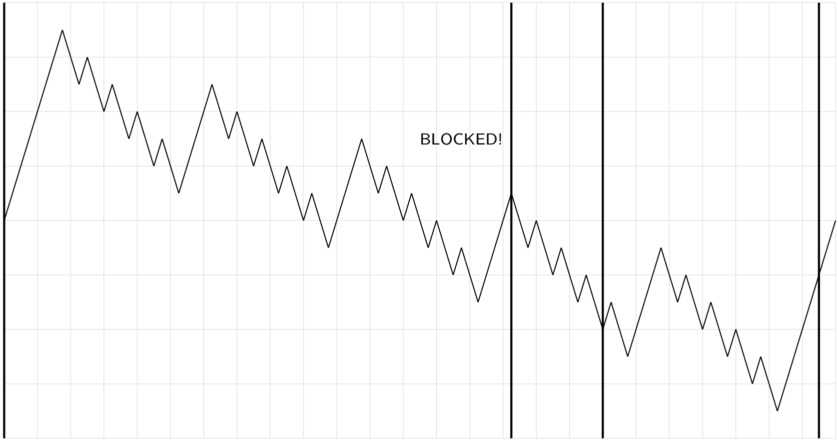
Counter-example word for $k = 2$



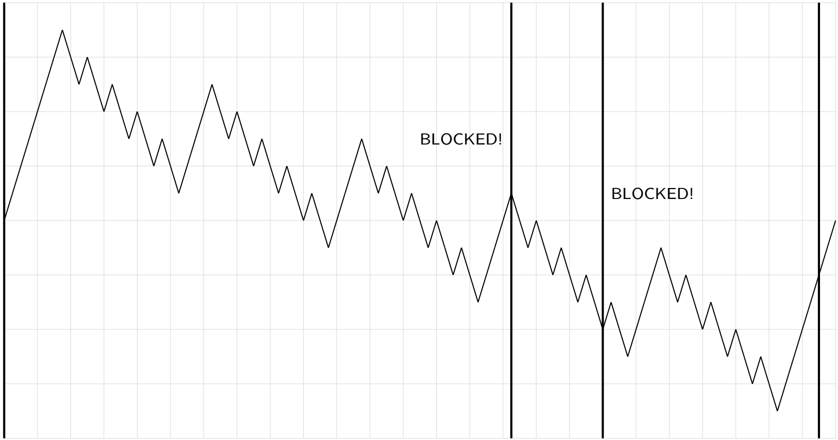
Counter-example word for $k = 2$



Counter-example word for $k = 2$



Counter-example word for $k = 2$



The main result

Theorem

The language O_1 is not derived by any non-branching ℓMCFG.

EDT0L

The language $L = \{ww \mid w \in \{a, b\}\}$

EDT0L

The language $L = \{ww \mid w \in \{a, b\}\}$

is recognised by the EDT0L (N, Σ, J, P, S) with:

set of non-terminals: $N = \{S, A\}$

set of table symbols: $J = \{0, 1\}$

set of rules P:

$P(0)$	$P(1)$
$S \to AA$	$S \to AA$
$A \to aA$	$A \to bA$

EDT0L

An EDT0L system is a tuple (N, Σ, J, P, S),

EDT0L

An EDT0L system is a tuple (N, Σ, J, P, S),

J is a finite set of table symbols and

EDT0L

An EDT0L system is a tuple (N, Σ, J, P, S),

J is a finite set of table symbols and
P associates, to every table symbol, a substitution of the non-terminals.

O_1 is not an EDT0L language

Theorem

The language O_1 is not derived by any EDT0L.

O_3 is not an *IO* language

O_3 is the commutative closure of the language $(abcd)^*$:

$$O_3 = \{u \mid |u|_a = |u|_b = |u|_c = |u|_d\}$$

O_3 is not an IO language

O_3 is the commutative closure of the language $(abcd)^*$:

$$O_3 = \{u \mid |u|_a = |u|_b = |u|_c = |u|_d\}$$

Theorem

The language O_3 is not IO.

Summing up

Theorem

The language O_1 is not derived by any non-branching ℓMCFG nor any EDT0L.

Theorem

The language O_3 is not IO.

Thank you for your attention