The Deducibility Problem for Certain Extensions of the Full Non-Associative Lambek Calculus is Undecidable

August Sangalli
(Joint Work With Nick Galatos)

University of Denver

TACL 2026

Substructural Logics

LJ - is a sequent calculus for intuitionistic logic. (order and multiplicity of hypotheses do not matter)

Substructural Logics

LJ - is a sequent calculus for intuitionistic logic. (order and multiplicity of hypotheses do not matter)
FL - Obtained by removing the structural rules of exchange, weakening, and contraction. ($\mathbf{FL}$ is resource sensitive)

Substructural Logics

LJ - is a sequent calculus for intuitionistic logic. (order and multiplicity of hypotheses do not matter)
FL - Obtained by removing the structural rules of exchange, weakening, and contraction. ($\mathbf{FL}$ is resource sensitive)

Substructural logics are axiomatic extensions/restrictions of $\mathbf{FL}$.

Substructural Logics

LJ - is a sequent calculus for intuitionistic logic. (order and multiplicity of hypotheses do not matter)
FL - Obtained by removing the structural rules of exchange, weakening, and contraction. ($\mathbf{FL}$ is resource sensitive)

Substructural logics are axiomatic extensions/restrictions of $\mathbf{FL}$.
Examples: Relevance logic, linear logic, many-valued logic.

Substructural Logics

LJ - is a sequent calculus for intuitionistic logic. (order and multiplicity of hypotheses do not matter)
FL - Obtained by removing the structural rules of exchange, weakening, and contraction. ($\mathbf{FL}$ is resource sensitive)

Substructural logics are axiomatic extensions/restrictions of $\mathbf{FL}$.
Examples: Relevance logic, linear logic, many-valued logic.

L ($\mathbf{FL}$ without $\vee$ and $\wedge$) was introduced by Lambek in the context of mathematical linguistics .

Substructural Logics

LJ - is a sequent calculus for intuitionistic logic. (order and multiplicity of hypotheses do not matter)
FL - Obtained by removing the structural rules of exchange, weakening, and contraction. ($\mathbf{FL}$ is resource sensitive)

Substructural logics are axiomatic extensions/restrictions of $\mathbf{FL}$.
Examples: Relevance logic, linear logic, many-valued logic.

L ($\mathbf{FL}$ without $\vee$ and $\wedge$) was introduced by Lambek in the context of mathematical linguistics .
Lambek also introduced **nL** (removes associativity) to capture linguistic contexts in which association of words matter.

Substructural Logics

LJ - is a sequent calculus for intuitionistic logic. (order and multiplicity of hypotheses do not matter)
FL - Obtained by removing the structural rules of exchange, weakening, and contraction. ($\mathbf{FL}$ is resource sensitive)

Substructural logics are axiomatic extensions/restrictions of $\mathbf{FL}$.
Examples: Relevance logic, linear logic, many-valued logic.

L ($\mathbf{FL}$ without $\vee$ and $\wedge$) was introduced by Lambek in the context of mathematical linguistics .
Lambek also introduced **nL** (removes associativity) to capture linguistic contexts in which association of words matter.
Ex: Let's eat Grandma! vs Let's eat, Grandma!

Substructural Logics

LJ - is a sequent calculus for intuitionistic logic. (order and multiplicity of hypotheses do not matter)
FL - Obtained by removing the structural rules of exchange, weakening, and contraction. ($\mathbf{FL}$ is resource sensitive)

Substructural logics are axiomatic extensions/restrictions of $\mathbf{FL}$.
Examples: Relevance logic, linear logic, many-valued logic.

L ($\mathbf{FL}$ without $\vee$ and $\wedge$) was introduced by Lambek in the context of mathematical linguistics .
Lambek also introduced **nL** (removes associativity) to capture linguistic contexts in which association of words matter.
Ex: Let's eat Grandma! vs Let's eat, Grandma!

nFL is $\mathbf{FL}$ without associativity.

Residuated Lattices

A *not necessarily associative residuated lattice (nRL)* is an algebraic structure $\mathbf{R} = (R, \vee, \wedge, \cdot, \backslash, /, 1)$ so that $(R, \vee, \wedge)$ is a lattice, $(R, \cdot, 1)$ is a groupoid and the *law of residuation* holds: For all $x, y, z \in R$,

$$x \cdot y \leqslant z \iff y \leqslant x \backslash z \iff x \leqslant z / y$$

where $\leqslant$ is the induced lattice order.

Residuated Lattices

A *not necessarily associative residuated lattice (nRL)* is an algebraic structure $\mathbf{R} = (R, \vee, \wedge, \cdot, \backslash, /, 1)$ so that $(R, \vee, \wedge)$ is a lattice, $(R, \cdot, 1)$ is a groupoid and the *law of residuation* holds: For all $x, y, z \in R$,

$$x \cdot y \leqslant z \iff y \leqslant x \backslash z \iff x \leqslant z / y$$

where $\leqslant$ is the induced lattice order.
If we restrict $(R, \cdot, 1)$ to be a monoid, then $\mathbf{R}$ is a *residuated lattice*.

Residuated Lattices

A *not necessarily associative residuated lattice (nRL)* is an algebraic structure $\mathbf{R} = (R, \vee, \wedge, \cdot, \backslash, /, 1)$ so that $(R, \vee, \wedge)$ is a lattice, $(R, \cdot, 1)$ is a groupoid and the *law of residuation* holds: For all $x, y, z \in R$,

$$x \cdot y \leqslant z \iff y \leqslant x \backslash z \iff x \leqslant z / y$$

where $\leqslant$ is the induced lattice order.
If we restrict $(R, \cdot, 1)$ to be a monoid, then $\mathbf{R}$ is a *residuated lattice*.
Examples Boolean algebras, Heyting algebras, MV-algebras, ℓ-groups.

Residuated Lattices

A *not necessarily associative residuated lattice (nRL)* is an algebraic structure $\mathbf{R} = (R, \vee, \wedge, \cdot, \backslash, /, 1)$ so that $(R, \vee, \wedge)$ is a lattice, $(R, \cdot, 1)$ is a groupoid and the *law of residuation* holds: For all $x, y, z \in R$,

$$x \cdot y \leqslant z \iff y \leqslant x \backslash z \iff x \leqslant z / y$$

where $\leqslant$ is the induced lattice order.
If we restrict $(R, \cdot, 1)$ to be a monoid, then $\mathbf{R}$ is a *residuated lattice*.
Examples Boolean algebras, Heyting algebras, MV-algebras, ℓ-groups.

Varieties of (n)RLs serve as the algebraic semantics (in the sense of Blok and Pigozzi) for substructural logics.

Residuated Lattices

A *not necessarily associative residuated lattice (nRL)* is an algebraic structure $\mathbf{R} = (R, \vee, \wedge, \cdot, \backslash, /, 1)$ so that $(R, \vee, \wedge)$ is a lattice, $(R, \cdot, 1)$ is a groupoid and the *law of residuation* holds: For all $x, y, z \in R$,

$$x \cdot y \leqslant z \iff y \leqslant x \backslash z \iff x \leqslant z / y$$

where $\leqslant$ is the induced lattice order.
If we restrict $(R, \cdot, 1)$ to be a monoid, then $\mathbf{R}$ is a *residuated lattice*.
Examples Boolean algebras, Heyting algebras, MV-algebras, ℓ-groups.

Varieties of (n)RLs serve as the algebraic semantics (in the sense of Blok and Pigozzi) for substructural logics.
Recall that a quasi-equation is the universal closure of implications of the following form:

$$s_1 = t_1 \ \& \ \ldots \ \& \ s_n = t_n \ \Rightarrow \ s = t$$

2-tag Systems

Theorem [Chvalovsky JSL 2015]: The deducibility problem problem for $\mathbf{nFL}$ is undecidable.

2-tag Systems

Theorem [Chvalovsky JSL 2015]: The deducibility problem problem for $\mathbf{nFL}$ is undecidable. (The quasi-equational theory for nRL is undecidable)

2-tag Systems

Theorem [Chvalovsky JSL 2015]: The deducibility problem problem for $\mathbf{nFL}$ is undecidable. (The quasi-equational theory for nRL is undecidable) We extend this result to all logics in the interval from $\mathbf{nFL}$ to $\mathbf{FL}$.

2-tag Systems

Theorem [Chvalovsky JSL 2015]: The deducibility problem problem for $\mathbf{nFL}$ is undecidable. (The quasi-equational theory for nRL is undecidable) We extend this result to all logics in the interval from $\mathbf{nFL}$ to $\mathbf{FL}$.

2-tag Systems

Theorem [Chvalovsky JSL 2015]: The deducibility problem problem for $\mathbf{nFL}$ is undecidable. (The quasi-equational theory for nRL is undecidable) We extend this result to all logics in the interval from $\mathbf{nFL}$ to $\mathbf{FL}$.

A *2-tag system* consists of a finite alphabet $\mathcal{T} = \{a_1, ..., a_n\}$ and a function $\tau : \mathcal{T} \to \mathcal{T}^*$.

2-tag Systems

Theorem [Chvalovsky JSL 2015]: The deducibility problem problem for $\mathbf{nFL}$ is undecidable. (The quasi-equational theory for nRL is undecidable) We extend this result to all logics in the interval from $\mathbf{nFL}$ to $\mathbf{FL}$.

A *2-tag system* consists of a finite alphabet $\mathcal{T} = \{a_1, ..., a_n\}$ and a function $\tau : \mathcal{T} \to \mathcal{T}^*$.
Define the 1-step computation relation $\rightsquigarrow_1$ for words of length at least 2 by $a_i a_j u \rightsquigarrow_1 u\tau_i$ where $a_i, a_j \in \mathcal{T}$ and $u \in \mathcal{T}^*$.

2-tag Systems

Theorem [Chvalovsky JSL 2015]: The deducibility problem problem for $\mathbf{nFL}$ is undecidable. (The quasi-equational theory for nRL is undecidable) We extend this result to all logics in the interval from $\mathbf{nFL}$ to $\mathbf{FL}$.

A *2-tag system* consists of a finite alphabet $\mathcal{T} = \{a_1, ..., a_n\}$ and a function $\tau : \mathcal{T} \to \mathcal{T}^*$.
Define the 1-step computation relation $\rightsquigarrow_1$ for words of length at least 2 by $a_i a_j u \rightsquigarrow_1 u\tau_i$ where $a_i, a_j \in \mathcal{T}$ and $u \in \mathcal{T}^*$.

$\rightsquigarrow$:= reflexive and transitive closure of $\rightsquigarrow_1$.

2-tag Systems

Theorem [Chvalovsky JSL 2015]: The deducibility problem problem for $\mathbf{nFL}$ is undecidable. (The quasi-equational theory for nRL is undecidable) We extend this result to all logics in the interval from $\mathbf{nFL}$ to $\mathbf{FL}$.

A *2-tag system* consists of a finite alphabet $\mathcal{T} = \{a_1, ..., a_n\}$ and a function $\tau : \mathcal{T} \to \mathcal{T}^*$.
Define the 1-step computation relation $\rightsquigarrow_1$ for words of length at least 2 by $a_i a_j u \rightsquigarrow_1 u\tau_i$ where $a_i, a_j \in \mathcal{T}$ and $u \in \mathcal{T}^*$.

$\rightsquigarrow$:= reflexive and transitive closure of $\rightsquigarrow_1$.

We say that $w \in \mathcal{T}^*$ is *accepted* if there exists some $a \in \mathcal{T} \cup \{\varepsilon\}$ for which $w \rightsquigarrow a$.

2-tag Systems

Theorem [Chvalovsky JSL 2015]: The deducibility problem problem for $\mathbf{nFL}$ is undecidable. (The quasi-equational theory for nRL is undecidable) We extend this result to all logics in the interval from $\mathbf{nFL}$ to $\mathbf{FL}$.

A *2-tag system* consists of a finite alphabet $\mathcal{T} = \{a_1, ..., a_n\}$ and a function $\tau : \mathcal{T} \to \mathcal{T}^*$.
Define the 1-step computation relation $\rightsquigarrow_1$ for words of length at least 2 by $a_i a_j u \rightsquigarrow_1 u\tau_i$ where $a_i, a_j \in \mathcal{T}$ and $u \in \mathcal{T}^*$.

$\rightsquigarrow$:= reflexive and transitive closure of $\rightsquigarrow_1$.

We say that $w \in \mathcal{T}^*$ is *accepted* if there exists some $a \in \mathcal{T} \cup \{\varepsilon\}$ for which $w \rightsquigarrow a$. Denote the set of accepted words by $\mathrm{Acc}(\tau)$.

2-tag Systems

Theorem [Chvalovsky JSL 2015]: The deducibility problem problem for $\mathbf{nFL}$ is undecidable. (The quasi-equational theory for nRL is undecidable) We extend this result to all logics in the interval from $\mathbf{nFL}$ to $\mathbf{FL}$.

A *2-tag system* consists of a finite alphabet $\mathcal{T} = \{a_1, ..., a_n\}$ and a function $\tau : \mathcal{T} \to \mathcal{T}^*$.
Define the 1-step computation relation $\rightsquigarrow_1$ for words of length at least 2 by $a_i a_j u \rightsquigarrow_1 u\tau_i$ where $a_i, a_j \in \mathcal{T}$ and $u \in \mathcal{T}^*$.

$\rightsquigarrow$:= reflexive and transitive closure of $\rightsquigarrow_1$.

We say that $w \in \mathcal{T}^*$ is *accepted* if there exists some $a \in \mathcal{T} \cup \{\varepsilon\}$ for which $w \rightsquigarrow a$. Denote the set of accepted words by $\mathrm{Acc}(\tau)$.

Fact: The halting problem for 2-tag systems is undecidable.

2-tag Systems

Theorem [Chvalovsky JSL 2015]: The deducibility problem problem for $\mathbf{nFL}$ is undecidable. (The quasi-equational theory for nRL is undecidable) We extend this result to all logics in the interval from $\mathbf{nFL}$ to $\mathbf{FL}$.

A *2-tag system* consists of a finite alphabet $\mathcal{T} = \{a_1, ..., a_n\}$ and a function $\tau : \mathcal{T} \to \mathcal{T}^*$.
Define the 1-step computation relation $\rightsquigarrow_1$ for words of length at least 2 by $a_i a_j u \rightsquigarrow_1 u\tau_i$ where $a_i, a_j \in \mathcal{T}$ and $u \in \mathcal{T}^*$.

$\rightsquigarrow$:= reflexive and transitive closure of $\rightsquigarrow_1$.

We say that $w \in \mathcal{T}^*$ is *accepted* if there exists some $a \in \mathcal{T} \cup \{\varepsilon\}$ for which $w \rightsquigarrow a$. Denote the set of accepted words by $\mathrm{Acc}(\tau)$.

Fact: The halting problem for 2-tag systems is undecidable. In fact, there exists a 2-tag system for which $\mathrm{Acc}(\tau)$ is undecidable.

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$.

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\underline{a_1}a_1a_2$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\underline{a_1}a_1a_2 \rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\underline{a_1}a_1a_2 \rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3 = \underline{a_2}a_2a_3$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\underline{a_1}a_1a_2 \rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3 = \underline{a_2}a_2a_3$$
$$\rightsquigarrow_1 \cancel{a_2a_2}\underline{a_3}a_1$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\underline{a_1}a_1a_2 \rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3 = \underline{a_2}a_2a_3$$
$$\rightsquigarrow_1 \cancel{a_2a_2}\underline{a_3}a_1 = \underline{a_3}a_1$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\begin{aligned}
\underline{a_1}a_1a_2 &\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3 = \underline{a_2}a_2a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_2}\underline{a_3}a_1 = \underline{a_3}a_1 \\
&\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1
\end{aligned}$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\underline{a_1}a_1a_2 \rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3 = \underline{a_2}a_2a_3$$
$$\rightsquigarrow_1 \cancel{a_2a_2}\underline{a_3}a_1 = \underline{a_3}a_1$$
$$\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1 = \underline{a_1}a_1$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\begin{aligned}
\underline{a_1}a_1a_2 &\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3 = \underline{a_2}a_2a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_2}\underline{a_3}a_1 = \underline{a_3}a_1 \\
&\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1 = \underline{a_1}a_1 \\
&\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_3
\end{aligned}$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\begin{aligned}
\underline{a_1}a_1a_2 &\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3 = \underline{a_2}a_2a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_2}\underline{a_3}a_1 = \underline{a_3}a_1 \\
&\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1 = \underline{a_1}a_1 \\
&\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_3 = \underline{a_2}a_3
\end{aligned}$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\begin{aligned}
\underline{a_1}a_1a_2 &\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3 = \underline{a_2}a_2a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_2}\underline{a_3}a_1 = \underline{a_3}a_1 \\
&\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1 = \underline{a_1}a_1 \\
&\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_3 = \underline{a_2}a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_3}a_1
\end{aligned}$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\begin{aligned}
\underline{a_1}a_1a_2 &\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3 = \underline{a_2}a_2a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_2}\underline{a_3}a_1 = \underline{a_3}a_1 \\
&\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1 = \underline{a_1}a_1 \\
&\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_3 = \underline{a_2}a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_3}a_1 = a_1
\end{aligned}$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\begin{aligned}
\underline{a_1}a_1a_2 &\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3 = \underline{a_2}a_2a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_2}\underline{a_3}a_1 = \underline{a_3}a_1 \\
&\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1 = \underline{a_1}a_1 \\
&\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_3 = \underline{a_2}a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_3}a_1 = a_1 \in \mathcal{T}
\end{aligned}$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\begin{aligned}
\underline{a_1}a_1a_2 &\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3 = \underline{a_2}a_2a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_2}\underline{a_3}a_1 = \underline{a_3}a_1 \\
&\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1 = \underline{a_1}a_1 \\
&\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_3 = \underline{a_2}a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_3}a_1 = a_1 \in \mathcal{T}
\end{aligned}$$

Therefore, $a_1a_1a_2 \in \mathsf{Acc}(\tau)$.

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_1a_2$:

$$\begin{aligned}
\underline{a_1}a_1a_2 &\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_2a_3 = \underline{a_2}a_2a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_2}\underline{a_3}a_1 = \underline{a_3}a_1 \\
&\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1 = \underline{a_1}a_1 \\
&\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_2}a_3 = \underline{a_2}a_3 \\
&\rightsquigarrow_1 \cancel{a_2a_3}a_1 = a_1 \in \mathcal{T}
\end{aligned}$$

Therefore, $a_1a_1a_2 \in \mathsf{Acc}(\tau)$.

In fact, $a_2a_2a_3, a_3a_1, a_1a_1, a_2a_3, a_1 \in \mathsf{Acc}(\tau)$.

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$.

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_2a_3$:

$$\underline{a_1}a_2a_3 \rightsquigarrow_1 \cancel{a_1a_2}\underline{a_3}a_2a_3$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_2a_3$:

$$\underline{a_1}a_2a_3 \rightsquigarrow_1 \cancel{a_1a_2}\underline{a_3}a_2a_3 = \underline{a_3}a_2a_3$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_2a_3$:

$$\underline{a_1}a_2a_3 \rightsquigarrow_1 \cancel{a_1a_2}\underline{a_3}a_2a_3 = \underline{a_3}a_2a_3$$

$$\rightsquigarrow_1 \cancel{a_3a_2}\underline{a_3}a_1a_1$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_2a_3$:

$$\underline{a_1}a_2a_3 \rightsquigarrow_1 \cancel{a_1a_2}\underline{a_3}a_2a_3 = \underline{a_3}a_2a_3$$
$$\rightsquigarrow_1 \cancel{a_3a_2}\underline{a_3}a_1a_1 = \underline{a_3}a_1a_1$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_2a_3$:

$$\underline{a_1}a_2a_3 \rightsquigarrow_1 \cancel{a_1a_2}\underline{a_3}a_2a_3 = \underline{a_3}a_2a_3$$
$$\rightsquigarrow_1 \cancel{a_3a_2}\underline{a_3}a_1a_1 = \underline{a_3}a_1a_1$$
$$\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1a_1$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_2a_3$:

$$\underline{a_1}a_2a_3 \rightsquigarrow_1 \cancel{a_1a_2}\underline{a_3}a_2a_3 = \underline{a_3}a_2a_3$$
$$\rightsquigarrow_1 \cancel{a_3a_2}\underline{a_3}a_1a_1 = \underline{a_3}a_1a_1$$
$$\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1a_1 = \underline{a_1}a_1a_1$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_2a_3$:

$$\begin{aligned}
\underline{a_1}a_2a_3 &\rightsquigarrow_1 \cancel{a_1a_2}\underline{a_3}a_2a_3 = \underline{a_3}a_2a_3 \\
&\rightsquigarrow_1 \cancel{a_3a_2}\underline{a_3}a_1a_1 = \underline{a_3}a_1a_1 \\
&\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1a_1 = \underline{a_1}a_1a_1 \\
&\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_1}a_2a_3
\end{aligned}$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_2a_3$:

$$\begin{aligned}
\underline{a_1}a_2a_3 &\rightsquigarrow_1 \cancel{a_1a_2}\underline{a_3}a_2a_3 = \underline{a_3}a_2a_3 \\
&\rightsquigarrow_1 \cancel{a_3a_2}\underline{a_3}a_1a_1 = \underline{a_3}a_1a_1 \\
&\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1a_1 = \underline{a_1}a_1a_1 \\
&\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_1}a_2a_3 = \underline{a_1}a_2a_3
\end{aligned}$$

Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_2a_3$:

$$\begin{aligned}
\underline{a_1}a_2a_3 &\rightsquigarrow_1 \cancel{a_1a_2}\underline{a_3}a_2a_3 = \underline{a_3}a_2a_3 \\
&\rightsquigarrow_1 \cancel{a_3a_2}\underline{a_3}a_1a_1 = \underline{a_3}a_1a_1 \\
&\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1a_1 = \underline{a_1}a_1a_1 \\
&\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_1}a_2a_3 = \underline{a_1}a_2a_3 \rightsquigarrow_1 \dots
\end{aligned}$$

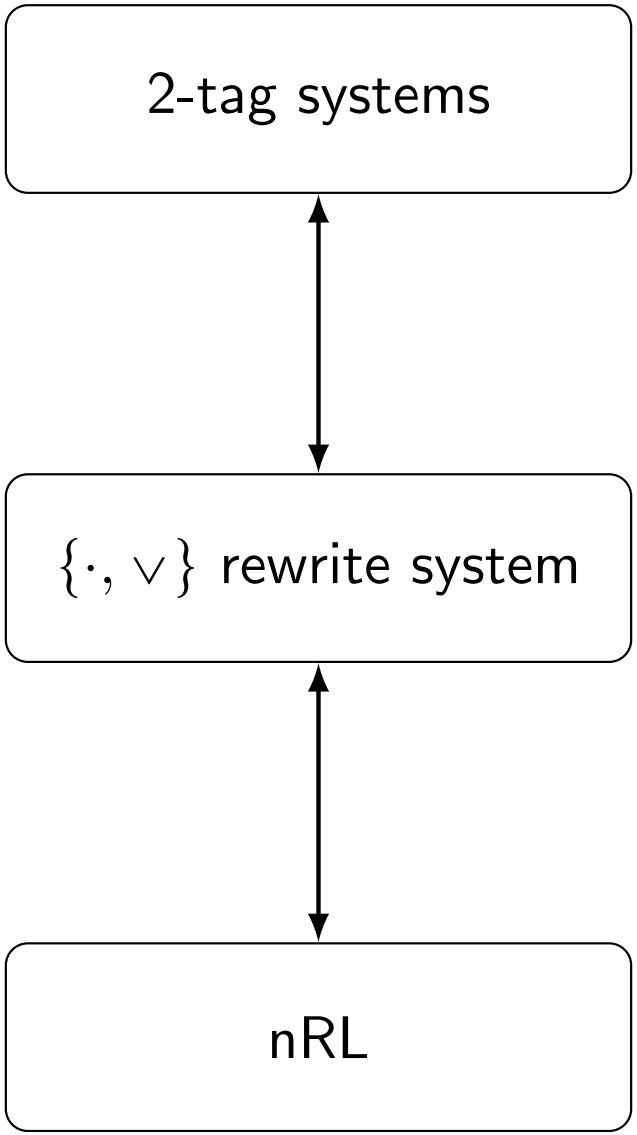
Examples

Let $\mathcal{T} := \{a_1, a_2, a_3\}$, $\tau_1 := a_2a_3$, $\tau_2 := a_1$, and $\tau_3 := a_1a_1$. For the word $w := a_1a_2a_3$:

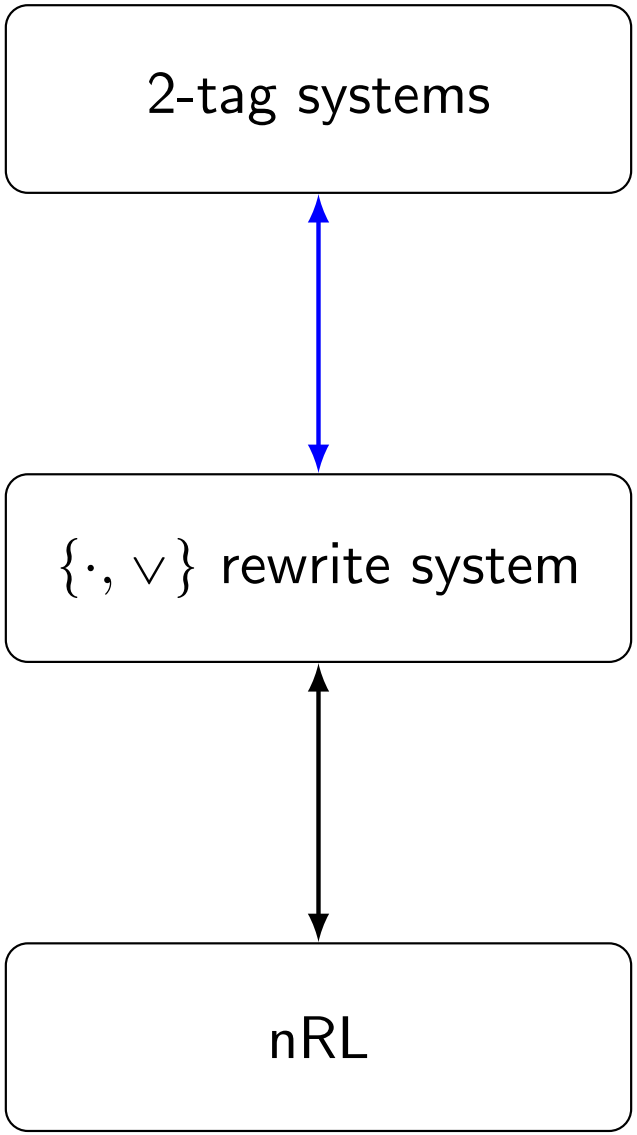
$$\begin{aligned}
\underline{a_1}a_2a_3 &\rightsquigarrow_1 \cancel{a_1a_2}\underline{a_3}a_2a_3 = \underline{a_3}a_2a_3 \\
&\rightsquigarrow_1 \cancel{a_3a_2}\underline{a_3}a_1a_1 = \underline{a_3}a_1a_1 \\
&\rightsquigarrow_1 \cancel{a_3a_1}\underline{a_1}a_1a_1 = \underline{a_1}a_1a_1 \\
&\rightsquigarrow_1 \cancel{a_1a_1}\underline{a_1}a_2a_3 = \underline{a_1}a_2a_3 \rightsquigarrow_1 \ldots
\end{aligned}$$

Therefore, $a_1a_2a_3 \notin \mathsf{Acc}(\tau)$.

Overview



Overview



Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1 a_2 a_3 \rightsquigarrow_1 a_3 a_2 a_3$.

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1 a_2 a_3 \rightsquigarrow_1 a_3 a_2 a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3$

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.
Our 2-tag derivation becomes $a_1^2a_3 \rightsquigarrow_1 a_3^2a_3$.

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.
Our 2-tag derivation becomes $a_1^2a_3 \rightsquigarrow_1 a_3^2a_3$.

We assume that every word below is right associated.

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.
Our 2-tag derivation becomes $a_1^2a_3 \rightsquigarrow_1 a_3^2a_3$.

We assume that every word below is right associated.
E.g. $a_1^2a_3^2a_1^3a_2$ stands for $a_1^2(a_3^2(a_1^3a_2))$. Formally $\overrightarrow{a_1^2a_3^4a_5^6a_7} = a_1^2(a_3^4(a_5^6a_7))$

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.
Our 2-tag derivation becomes $a_1^2a_3 \rightsquigarrow_1 a_3^2a_3$.

We assume that every word below is right associated.
E.g. $a_1^2a_3^2a_1^3a_2$ stands for $a_1^2(a_3^2(a_1^3a_2))$. Formally $\overrightarrow{a_1^2a_3^4a_5^6a_7} = a_1^2(a_3^4(a_5^6a_7))$

Applying the *naive* rule $\varepsilon \leqslant \tau_i$

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.
Our 2-tag derivation becomes $a_1^2a_3 \rightsquigarrow_1 a_3^2a_3$.

We assume that every word below is right associated.
E.g. $a_1^2a_3^2a_1^3a_2$ stands for $a_1^2(a_3^2(a_1^3a_2))$. Formally $\overrightarrow{a_1^2a_3^4a_5^6a_7} = a_1^2(a_3^4(a_5^6a_7))$

Applying the *naive* rule $\varepsilon \leqslant \tau_i$ gives $a_1^2a_3 \leqslant a_1^2\tau_1a_3$

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.
Our 2-tag derivation becomes $a_1^2a_3 \rightsquigarrow_1 a_3^2a_3$.

We assume that every word below is right associated.
E.g. $a_1^2a_3^2a_1^3a_2$ stands for $a_1^2(a_3^2(a_1^3a_2))$. Formally $\overrightarrow{a_1^2a_3^4a_5^6a_7} = a_1^2(a_3^4(a_5^6a_7))$

Applying the *naive* rule $\varepsilon \leqslant \tau_i$ gives $a_1^2a_3 \leqslant a_1^2\tau_1a_3 = a_1^2a_2^3a_3$.

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.
Our 2-tag derivation becomes $a_1^2a_3 \rightsquigarrow_1 a_3^2a_3$.

We assume that every word below is right associated.
E.g. $a_1^2a_3^2a_1^3a_2$ stands for $a_1^2(a_3^2(a_1^3a_2))$. Formally $\overrightarrow{a_1^2a_3^4a_5^6a_7} = a_1^2(a_3^4(a_5^6a_7))$

Applying the *naive* rule $\varepsilon \leqslant \tau_i$ gives $a_1^2a_3 \leqslant a_1^2\tau_1a_3 = a_1^2a_2^3a_3$.
We represent $a_1^2a_3$ by $a_1^2a_3X$ and **introduce the rule** $a_jX \leqslant \overline{a_j\tau_i}X$:

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.
Our 2-tag derivation becomes $a_1^2a_3 \rightsquigarrow_1 a_3^2a_3$.

We assume that every word below is right associated.
E.g. $a_1^2a_3^2a_1^3a_2$ stands for $a_1^2(a_3^2(a_1^3a_2))$. Formally $\overrightarrow{a_1^2a_3^4a_5^6a_7} = a_1^2(a_3^4(a_5^6a_7))$

Applying the *naive* rule $\varepsilon \leqslant \tau_i$ gives $a_1^2a_3 \leqslant a_1^2\tau_1a_3 = a_1^2a_2^3a_3$.
We represent $a_1^2a_3$ by $a_1^2a_3X$ and **introduce the rule** $a_jX \leqslant \overline{a_j\tau_i}X$:
$a_1^2a_3X \leqslant a_1^2\overline{a_3\tau_1}X$

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.
Our 2-tag derivation becomes $a_1^2a_3 \rightsquigarrow_1 a_3^2a_3$.

We assume that every word below is right associated.
E.g. $a_1^2a_3^2a_1^3a_2$ stands for $a_1^2(a_3^2(a_1^3a_2))$. Formally $\overrightarrow{a_1^2a_3^4a_5^6a_7} = a_1^2(a_3^4(a_5^6a_7))$

Applying the *naive* rule $\varepsilon \leqslant \tau_i$ gives $a_1^2a_3 \leqslant a_1^2\tau_1a_3 = a_1^2a_2^3a_3$.
We represent $a_1^2a_3$ by $a_1^2a_3X$ and **introduce the rule** $a_jX \leqslant \overline{a_j\tau_i}X$:
$a_1^2a_3X \leqslant a_1^2\overline{a_3\tau_1}X = a_1^2a_3^2a_3X$

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.
Our 2-tag derivation becomes $a_1^2a_3 \rightsquigarrow_1 a_3^2a_3$.

We assume that every word below is right associated.
E.g. $a_1^2a_3^2a_1^3a_2$ stands for $a_1^2(a_3^2(a_1^3a_2))$. Formally $\overrightarrow{a_1^2a_3^4a_5^6a_7} = a_1^2(a_3^4(a_5^6a_7))$

Applying the *naive* rule $\varepsilon \leqslant \tau_i$ gives $a_1^2a_3 \leqslant a_1^2\tau_1a_3 = a_1^2a_2^3a_3$.
We represent $a_1^2a_3$ by $a_1^2a_3X$ and **introduce the rule** $a_jX \leqslant \overline{a_j\tau_i}X$:
$a_1^2a_3X \leqslant a_1^2\overline{a_3\tau_1}X = a_1^2a_3^2a_3X$ Formally: $a_1^2(a_3X)$

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.
Our 2-tag derivation becomes $a_1^2a_3 \rightsquigarrow_1 a_3^2a_3$.

We assume that every word below is right associated.
E.g. $a_1^2a_3^2a_1^3a_2$ stands for $a_1^2(a_3^2(a_1^3a_2))$. Formally $\overrightarrow{a_1^2a_3^4a_5^6a_7} = a_1^2(a_3^4(a_5^6a_7))$

Applying the *naive* rule $\varepsilon \leqslant \tau_i$ gives $a_1^2a_3 \leqslant a_1^2\tau_1a_3 = a_1^2a_2^3a_3$.
We represent $a_1^2a_3$ by $a_1^2a_3X$ and **introduce the rule** $a_jX \leqslant \overline{a_j\tau_i}X$:
$a_1^2a_3X \leqslant a_1^2\overline{a_3\tau_1}X = a_1^2a_3^2a_3X$ Formally: $a_1^2(a_3X) \leqslant a_1^2(a_3^2(a_3X))$.

Pairing Convention & Appending Rules

In the previous 2-tag system, $a_1a_2a_3 \rightsquigarrow_1 a_3a_2a_3$.
However, our rewrite system will be on right associated $\cdot$ terms.
$a_1(a_2a_3) \leqslant ?$ The issue here is that $a_1 \cdot a_2$ is not a subterm of $a_1 \cdot (a_2 \cdot a_3)$.

$\overline{a_1a_2a_3} = a_1^2a_3 \qquad \overline{a_2a_3a_1a_2} = a_2^3a_1^2$
We expand the alphabet to include symbols a_i^j and represent each word $w \in \mathcal{T}^*$ by $\overline{w}$.
Our 2-tag derivation becomes $a_1^2a_3 \rightsquigarrow_1 a_3^2a_3$.

We assume that every word below is right associated.
E.g. $a_1^2a_3^2a_1^3a_2$ stands for $a_1^2(a_3^2(a_1^3a_2))$. Formally $\overrightarrow{a_1^2a_3^4a_5^6a_7} = a_1^2(a_3^4(a_5^6a_7))$

Applying the *naive* rule $\varepsilon \leqslant \tau_i$ gives $a_1^2a_3 \leqslant a_1^2\tau_1a_3 = a_1^2a_2^3a_3$.
We represent $a_1^2a_3$ by $a_1^2a_3X$ and **introduce the rule** $a_jX \leqslant \overline{a_j\tau_i}X$:
$a_1^2a_3X \leqslant a_1^2\overline{a_3\tau_1}X = a_1^2a_3^2a_3X$ Formally: $a_1^2(a_3X) \leqslant a_1^2(a_3^2(a_3X))$.

Similarly, applying the *naive* rule $a_i^j \leqslant \varepsilon$ would allow us to delete *any* pair.

Full $\{\cdot, \vee\}$ Rewrite System

$w \in \mathcal{T}^*$ is represented by $e\overline{w}X$

Full $\{\cdot, \vee\}$ Rewrite System

$w \in \mathcal{T}^*$ is represented by $e\overline{w}X$ (formally $\overrightarrow{e\overline{w}X}$)

Full $\{\cdot, \vee\}$ Rewrite System

$$w \in \mathcal{T}^* \text{ is represented by } e\overline{w}X \text{ (formally } \overrightarrow{e\overline{w}X}\text{)}$$

- To control the alterations of appending and deleting, we introduce the letters e, e' and X, X'.

Full $\{\cdot, \vee\}$ Rewrite System

$$w \in \mathcal{T}^* \text{ is represented by } e\overline{w}X \text{ (formally } \overrightarrow{e\overline{w}X}\text{)}$$

- To control the alterations of appending and deleting, we introduce the letters e, e' and X, X'.
- We use the convention of append first then delete.

Full $\{\cdot, \vee\}$ Rewrite System

$$w \in \mathcal{T}^* \text{ is represented by } e\overline{w}X \text{ (formally } \overrightarrow{e\overline{w}X}\text{)}$$

- To control the alterations of appending and deleting, we introduce the letters e, e' and X, X'.
- We use the convention of append first then delete.
- We introduce 30 rewrite rules (instructions) denoted by P.

Full $\{\cdot, \vee\}$ Rewrite System

$$w \in \mathcal{T}^* \text{ is represented by } e\overline{w}X \text{ (formally } \overrightarrow{e\overline{w}X}\text{)}$$

- To control the alterations of appending and deleting, we introduce the letters e, e' and X, X'.
- We use the convention of append first then delete.
- We introduce 30 rewrite rules (instructions) denoted by P.
- Utilize $\vee$ to branch off into parallel computations that verify a rule has been applied correctly.

Full $\{\cdot, \vee\}$ Rewrite System

$$w \in \mathcal{T}^* \text{ is represented by } e\overline{w}X \text{ (formally } \overrightarrow{e\overline{w}X}\text{)}$$

- To control the alterations of appending and deleting, we introduce the letters e, e' and X, X'.
- We use the convention of append first then delete.
- We introduce 30 rewrite rules (instructions) denoted by P.
- Utilize $\vee$ to branch off into parallel computations that verify a rule has been applied correctly.
- For a given 2-tag system $\tau : \mathcal{T} \to \mathcal{T}^*$ let R_τ designate our $\{\cdot, \vee\}$ rewrite system.

Correctness of Rewrite System

Recall the 2-tag system where $\tau_1 = a_2a_3, \tau_2 = a_1, \tau_3 = a_1a_1$. Note: $a_2a_3 \rightsquigarrow_1 a_1$

$$ea_2^3X$$

Correctness of Rewrite System

Recall the 2-tag system where $\tau_1 = a_2a_3, \tau_2 = a_1, \tau_3 = a_1a_1$. Note: $a_2a_3 \rightsquigarrow_1 a_1$

$(26) : a_j^k X \leqslant \overline{a_j^k \tau_i} X' \vee a_j^k A_i$

$$ea_2^3 X \leqslant e(a_2^3 a_1 X' \vee a_2^3 A_2)$$

Correctness of Rewrite System

Recall the 2-tag system where $\tau_1 = a_2a_3, \tau_2 = a_1, \tau_3 = a_1a_1$. Note: $a_2a_3 \rightsquigarrow_1 a_1$

$(26) : a_j^k X \leqslant \overline{a_j^k \tau_i} X' \vee a_j^k A_i$

$$\begin{aligned} ea_2^3 X &\leqslant e(a_2^3 a_1 X' \vee a_2^3 A_2) \\ &= ea_2^3 a_1 X' \vee ea_2^3 A_2 \end{aligned}$$

Correctness of Rewrite System

Recall the 2-tag system where $\tau_1 = a_2a_3, \tau_2 = a_1, \tau_3 = a_1a_1$. Note: $a_2a_3 \rightsquigarrow_1 a_1$

$(26) : a_j^k X \leqslant \overline{a_j^k \tau_i} X' \vee a_j^k A_i$ $(2) : ea_i^j A_i \leqslant eA$

$$\begin{aligned} ea_2^3 X &\leqslant e(a_2^3 a_1 X' \vee a_2^3 A_2) \\ &= ea_2^3 a_1 X' \vee ea_2^3 A_2 \\ &\leqslant ea_2^3 a_1 X' \vee eA \end{aligned}$$

Correctness of Rewrite System

Recall the 2-tag system where $\tau_1 = a_2a_3, \tau_2 = a_1, \tau_3 = a_1a_1$. Note: $a_2a_3 \rightsquigarrow_1 a_1$

$(26) : a_j^k X \leqslant \overline{a_j^k \tau_i} X' \vee a_j^k A_i$ $(2) : ea_i^j A_i \leqslant eA$ $(29) : a_i^j \leqslant e' \vee d'$

$$\begin{aligned} ea_2^3 X &\leqslant e(a_2^3 a_1 X' \vee a_2^3 A_2) \\ &= ea_2^3 a_1 X' \vee ea_2^3 A_2 \\ &\leqslant ea_2^3 a_1 X' \vee eA \\ &\leqslant e(e' \vee d')a_1 X' \vee eA \end{aligned}$$

Correctness of Rewrite System

Recall the 2-tag system where $\tau_1 = a_2a_3, \tau_2 = a_1, \tau_3 = a_1a_1$. Note: $a_2a_3 \rightsquigarrow_1 a_1$

$(26) : a_j^k X \leqslant \overline{a_j^k \tau_i} X' \vee a_j^k A_i$ $(2) : ea_i^j A_i \leqslant eA$ $(29) : a_i^j \leqslant e' \vee d'$

$$\begin{aligned} ea_2^3 X &\leqslant e(a_2^3 a_1 X' \vee a_2^3 A_2) \\ &= ea_2^3 a_1 X' \vee ea_2^3 A_2 \\ &\leqslant ea_2^3 a_1 X' \vee eA \\ &\leqslant e(e' \vee d')a_1 X' \vee eA \\ &= ee' a_1 X' \vee ed' a_1 X' \vee eA \end{aligned}$$

Correctness of Rewrite System

Recall the 2-tag system where $\tau_1 = a_2a_3, \tau_2 = a_1, \tau_3 = a_1a_1$. Note: $a_2a_3 \rightsquigarrow_1 a_1$

$(26) : a_j^k X \leqslant \overline{a_j^k \tau_i} X' \vee a_j^k A_i$ $(2) : ea_i^j A_i \leqslant eA$ $(29) : a_i^j \leqslant e' \vee d'$

$(10) : X' \leqslant D'$

$$\begin{aligned} ea_2^3 X &\leqslant e(a_2^3 a_1 X' \vee a_2^3 A_2) \\ &= ea_2^3 a_1 X' \vee ea_2^3 A_2 \\ &\leqslant ea_2^3 a_1 X' \vee eA \\ &\leqslant e(e' \vee d')a_1 X' \vee eA \\ &= ee' a_1 X' \vee ed' a_1 X' \vee eA \\ &\leqslant ee' a_1 X' \vee ed' a_1 D' \vee eA \end{aligned}$$

Correctness of Rewrite System

Recall the 2-tag system where $\tau_1 = a_2a_3, \tau_2 = a_1, \tau_3 = a_1a_1$. Note: $a_2a_3 \rightsquigarrow_1 a_1$

$(26) : a_j^k X \leqslant \overline{a_j^k \tau_i} X' \vee a_j^k A_i$ $(2) : ea_i^j A_i \leqslant eA$ $(29) : a_i^j \leqslant e' \vee d'$

$(10) : X' \leqslant D'$ $(6) : d' a_i D' \leqslant d' D'$

$$
\begin{aligned}
ea_2^3 X &\leqslant e(a_2^3 a_1 X' \vee a_2^3 A_2) \\
&= ea_2^3 a_1 X' \vee ea_2^3 A_2 \\
&\leqslant ea_2^3 a_1 X' \vee eA \\
&\leqslant e(e' \vee d') a_1 X' \vee eA \\
&= ee' a_1 X' \vee ed' a_1 X' \vee eA \\
&\leqslant ee' a_1 X' \vee ed' a_1 D' \vee eA \\
&\leqslant ee' a_1 X' \vee ed' D' \vee eA
\end{aligned}
$$

Correctness of Rewrite System

Recall the 2-tag system where $\tau_1 = a_2a_3, \tau_2 = a_1, \tau_3 = a_1a_1$. Note: $a_2a_3 \rightsquigarrow_1 a_1$

$(26) : a_j^k X \leqslant \overline{a_j^k \tau_i} X' \vee a_j^k A_i$ $(2) : ea_i^j A_i \leqslant eA$ $(29) : a_i^j \leqslant e' \vee d'$

$(10) : X' \leqslant D'$ $(6) : d'a_i D' \leqslant d'D'$ $(5) : ed'D' \leqslant eA$

$$
\begin{aligned}
ea_2^3 X &\leqslant e(a_2^3 a_1 X' \vee a_2^3 A_2) \\
&= ea_2^3 a_1 X' \vee ea_2^3 A_2 \\
&\leqslant ea_2^3 a_1 X' \vee eA \\
&\leqslant e(e' \vee d')a_1 X' \vee eA \\
&= ee'a_1 X' \vee ed'a_1 X' \vee eA \\
&\leqslant ee'a_1 X' \vee ed'a_1 D' \vee eA \\
&\leqslant ee'a_1 X' \vee ed'D' \vee eA \\
&\leqslant ee'a_1 X' \vee eA \vee eA
\end{aligned}
$$

Correctness of Rewrite System

Recall the 2-tag system where $\tau_1 = a_2a_3, \tau_2 = a_1, \tau_3 = a_1a_1$. Note: $a_2a_3 \rightsquigarrow_1 a_1$

$(26) : a_j^k X \leqslant \overline{a_j^k \tau_i} X' \vee a_j^k A_i$ $(2) : ea_i^j A_i \leqslant eA$ $(29) : a_i^j \leqslant e' \vee d'$

$(10) : X' \leqslant D'$ $(6) : d'a_iD' \leqslant d'D'$ $(5) : ed'D' \leqslant eA$ $(24) : e'a_iX' \leqslant e'X'$

$$
\begin{aligned}
ea_2^3X &\leqslant e(a_2^3a_1X' \vee a_2^3A_2) \\
&= ea_2^3a_1X' \vee ea_2^3A_2 \\
&\leqslant ea_2^3a_1X' \vee eA \\
&\leqslant e(e' \vee d')a_1X' \vee eA \\
&= ee'a_1X' \vee ed'a_1X' \vee eA \\
&\leqslant ee'a_1X' \vee ed'a_1D' \vee eA \\
&\leqslant ee'a_1X' \vee ed'D' \vee eA \\
&\leqslant ee'a_1X' \vee eA \vee eA \\
&\leqslant ee'X' \vee eA \vee eA
\end{aligned}
$$

Correctness of Rewrite System

Recall the 2-tag system where $\tau_1 = a_2a_3, \tau_2 = a_1, \tau_3 = a_1a_1$. Note: $a_2a_3 \rightsquigarrow_1 a_1$

$(26) : a_j^k X \leqslant \overline{a_j^k \tau_i} X' \vee a_j^k A_i$ $(2) : ea_i^j A_i \leqslant eA$ $(29) : a_i^j \leqslant e' \vee d'$

$(10) : X' \leqslant D'$ $(6) : d'a_i D' \leqslant d'D'$ $(5) : ed'D' \leqslant eA$ $(24) : e'a_i X' \leqslant e'X'$

$(23) : ee'X' \leqslant eX$

$$
\begin{aligned}
ea_2^3 X &\leqslant e(a_2^3 a_1 X' \vee a_2^3 A_2) \\
&= ea_2^3 a_1 X' \vee ea_2^3 A_2 \\
&\leqslant ea_2^3 a_1 X' \vee eA \\
&\leqslant e(e' \vee d')a_1 X' \vee eA \\
&= ee'a_1 X' \vee ed'a_1 X' \vee eA \\
&\leqslant ee'a_1 X' \vee ed'a_1 D' \vee eA \\
&\leqslant ee'a_1 X' \vee ed'D' \vee eA \\
&\leqslant ee'a_1 X' \vee eA \vee eA \\
&\leqslant ee'X' \vee eA \vee eA \leqslant eX \vee eA \vee eA
\end{aligned}
$$

Correctness of Rewrite System

Recall the 2-tag system where $\tau_1 = a_2a_3, \tau_2 = a_1, \tau_3 = a_1a_1$. Note: $a_2a_3 \rightsquigarrow_1 a_1$

$(26) : a_j^k X \leqslant \overline{a_j^k \tau_i} X' \vee a_j^k A_i$ $(2) : ea_i^j A_i \leqslant eA$ $(29) : a_i^j \leqslant e' \vee d'$

$(10) : X' \leqslant D'$ $(6) : d'a_i D' \leqslant d'D'$ $(5) : ed'D' \leqslant eA$ $(24) : e'a_i X' \leqslant e'X'$

$(23) : ee'X' \leqslant eX$

$$
\begin{aligned}
ea_2^3 X &\leqslant e(a_2^3 a_1 X' \vee a_2^3 A_2) \\
&= ea_2^3 a_1 X' \vee ea_2^3 A_2 \\
&\leqslant ea_2^3 a_1 X' \vee eA \\
&\leqslant e(e' \vee d')a_1 X' \vee eA \\
&= ee'a_1 X' \vee ed'a_1 X' \vee eA \\
&\leqslant ee'a_1 X' \vee ed'a_1 D' \vee eA \\
&\leqslant ee'a_1 X' \vee ed'D' \vee eA \\
&\leqslant ee'a_1 X' \vee eA \vee eA \\
&\leqslant ee'X' \vee eA \vee eA \leqslant eX \vee eA \vee eA \Rightarrow ea_2^3 X \in \mathsf{Acc}(R_\tau)
\end{aligned}
$$

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Correct derivation:

$\underline{a_1}a_2 \rightsquigarrow_1 \cancel{a_1 a_2} a_3$ which implies $a_1a_2 \in \mathsf{Acc}(\tau)$.

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Correct derivation:

$\underline{a_1}a_2 \rightsquigarrow_1 \cancel{a_1a_2}a_3$ which implies $a_1a_2 \in \mathsf{Acc}(\tau)$.

Double appending glitch:

$\underline{a_1}a_2$

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Correct derivation:

$\underline{a_1}a_2 \rightsquigarrow_1 \cancel{a_1a_2}a_3$ which implies $a_1a_2 \in \mathsf{Acc}(\tau)$.

Double appending glitch:

$\underline{a_1}a_2 \rightsquigarrow_1 a_1a_2\underline{a_3}$

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Correct derivation:

$\underline{a_1}a_2 \rightsquigarrow_1 \cancel{a_1a_2}a_3$ which implies $a_1a_2 \in \mathsf{Acc}(\tau)$.

Double appending glitch:

$\underline{a_1}a_2 \rightsquigarrow_1 a_1a_2\underline{a_3} \rightsquigarrow a_1a_2a_3a_3\underline{a_3}a_3$

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Correct derivation:

$\underline{a_1}a_2 \rightsquigarrow_1 \cancel{a_1a_2}a_3$ which implies $a_1a_2 \in \mathsf{Acc}(\tau)$.

Double appending glitch:

$\underline{a_1}a_2 \rightsquigarrow_1 a_1a_2\underline{a_3} \rightsquigarrow a_1a_2a_3a_3\underline{a_3}a_3 \rightsquigarrow_1 \cancel{a_1a_2a_3a_3}\underline{a_3}a_3$

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Correct derivation:

$\underline{a_1}a_2 \rightsquigarrow_1 \cancel{a_1a_2}a_3$ which implies $a_1a_2 \in \mathsf{Acc}(\tau)$.

Double appending glitch:

$\underline{a_1}a_2 \rightsquigarrow_1 a_1a_2\underline{a_3} \rightsquigarrow a_1a_2a_3a_3\underline{a_3}a_3 \rightsquigarrow_1 \cancel{a_1a_2}a_3\cancel{a_3}\underline{a_3}a_3 \rightsquigarrow_1 \ldots$

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Correct derivation:

$\underline{a_1}a_2 \rightsquigarrow_1 \cancel{a_1a_2}a_3$ which implies $a_1a_2 \in \mathsf{Acc}(\tau)$.

Double appending glitch:

$\underline{a_1}a_2 \rightsquigarrow_1 a_1a_2\underline{a_3} \rightsquigarrow a_1a_2a_3a_3\underline{a_3}a_3 \rightsquigarrow_1 \cancel{a_1a_2a_3a_3}\underline{a_3}a_3 \rightsquigarrow_1 \ldots$

Detection in rewrite system:

$$ea_1^2X$$

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Correct derivation:

$\underline{a_1}a_2 \rightsquigarrow_1 \cancel{a_1a_2}a_3$ which implies $a_1a_2 \in \mathsf{Acc}(\tau)$.

Double appending glitch:

$\underline{a_1}a_2 \rightsquigarrow_1 a_1a_2\underline{a_3} \rightsquigarrow a_1a_2a_3a_3\underline{a_3}a_3 \rightsquigarrow_1 \cancel{a_1a_2a_3a_3}\underline{a_3}a_3 \rightsquigarrow_1 \ldots$

Detection in rewrite system:

$(26) : a_j^k X \leqslant \overline{a_j\tau_i}X' \vee a_j^k A_i$

$$ea_1^2X \leqslant ea_1^2a_3X' \vee ea_1^2A_1$$

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Correct derivation:

$\underline{a_1}a_2 \rightsquigarrow_1 \cancel{a_1a_2}a_3$ which implies $a_1a_2 \in \mathsf{Acc}(\tau)$.

Double appending glitch:

$\underline{a_1}a_2 \rightsquigarrow_1 a_1a_2\underline{a_3} \rightsquigarrow a_1a_2a_3a_3\underline{a_3}a_3 \rightsquigarrow_1 \cancel{a_1a_2a_3a_3}\underline{a_3}a_3 \rightsquigarrow_1 \ldots$

Detection in rewrite system:

$(26) : a_j^k X \leqslant \overline{a_j\tau_i}X' \vee a_j^k A_i$ $(27) : a_j X' \leqslant \overline{a_j\tau_i}X \vee a_j A_i$

$$\begin{aligned} ea_1^2 X &\leqslant ea_1^2 a_3 X' \vee ea_1^2 A_1 \\ &\leqslant ea_1^2 a_3^3 a_3^3 X \vee ea_1^2 a_3 A_1' \vee ea_1^2 A_1 \end{aligned}$$

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Correct derivation:

$\underline{a_1}a_2 \rightsquigarrow_1 \cancel{a_1a_2}a_3$ which implies $a_1a_2 \in \mathsf{Acc}(\tau)$.

Double appending glitch:

$\underline{a_1}a_2 \rightsquigarrow_1 a_1a_2\underline{a_3} \rightsquigarrow a_1a_2a_3a_3\underline{a_3}a_3 \rightsquigarrow_1 \cancel{a_1a_2a_3a_3}\underline{a_3}a_3 \rightsquigarrow_1 \ldots$

Detection in rewrite system:

$(26) : a_j^k X \leqslant \overline{a_j\tau_i}X' \vee a_j^k A_i$ $(27) : a_j X' \leqslant \overline{a_j\tau_i}X \vee a_j A_i$

$(13) : a_j^k a_l A_i' \leqslant a_j^k A_i'$

$$\begin{aligned} ea_1^2X &\leqslant ea_1^2a_3X' \vee ea_1^2A_1 \\ &\leqslant ea_1^2a_3^3a_3^3X \vee ea_1^2a_3A_1' \vee ea_1^2A_1 \\ &\leqslant ea_1^2a_3^3a_3^3X \vee ea_1^2A_1' \vee ea_1^2A_1 \end{aligned}$$

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Correct derivation:

$\underline{a_1}a_2 \rightsquigarrow_1 \cancel{a_1a_2}a_3$ which implies $a_1a_2 \in \mathsf{Acc}(\tau)$.

Double appending glitch:

$\underline{a_1}a_2 \rightsquigarrow_1 a_1a_2\underline{a_3} \rightsquigarrow a_1a_2a_3a_3\underline{a_3}a_3 \rightsquigarrow_1 \cancel{a_1a_2a_3a_3}\underline{a_3}a_3 \rightsquigarrow_1 \ldots$

Detection in rewrite system:

$(26) : a_j^k X \leqslant \overline{a_j\tau_i}X' \vee a_j^k A_i$ $(27) : a_j X' \leqslant \overline{a_j\tau_i}X \vee a_j A_i$

$(13) : a_j^k a_l A_i' \leqslant a_j^k A_i'$ $(12) : e'a_i^j A_i' \leqslant e'A$

$$\begin{aligned} ea_1^2 X &\leqslant ea_1^2 a_3 X' \vee ea_1^2 A_1 \\ &\leqslant ea_1^2 a_3^3 a_3^3 X \vee ea_1^2 a_3 A_1' \vee ea_1^2 A_1 \\ &\leqslant ea_1^2 a_3^3 a_3^3 X \vee ea_1^2 A_1' \vee ea_1^2 A_1 \leqslant ?? \end{aligned}$$

Rewrite System Does Not Over Compute

Consider the 2-tag system where $\tau_1 := a_3$, $\tau_2 := a_2$, and $\tau_3 := a_3a_3a_3$.

Correct derivation:

$\underline{a_1}a_2 \rightsquigarrow_1 \cancel{a_1a_2}a_3$ which implies $a_1a_2 \in \mathsf{Acc}(\tau)$.

Double appending glitch:

$\underline{a_1}a_2 \rightsquigarrow_1 a_1a_2\underline{a_3} \rightsquigarrow a_1a_2a_3a_3\underline{a_3}a_3 \rightsquigarrow_1 \cancel{a_1a_2a_3a_3}\underline{a_3}a_3 \rightsquigarrow_1 \ldots$

Detection in rewrite system:

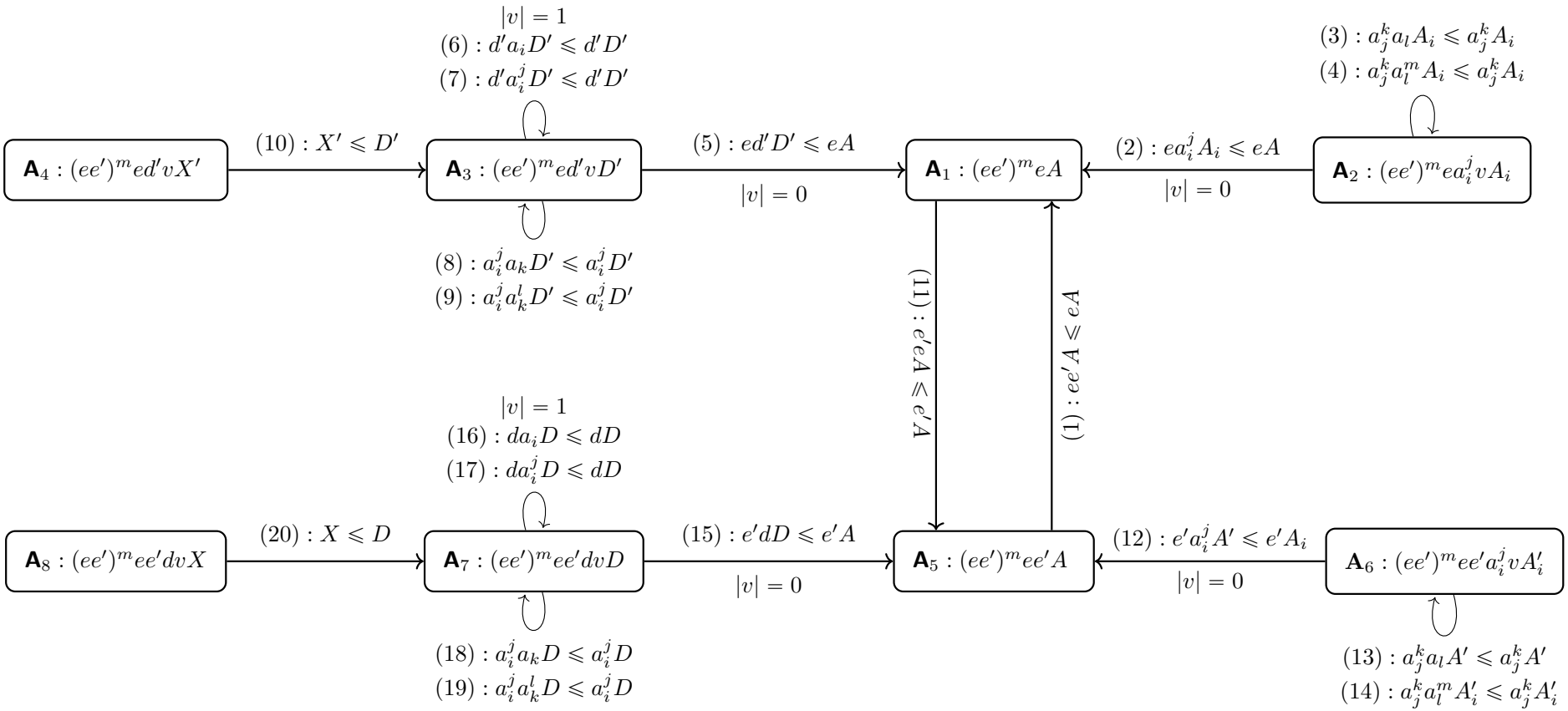
$(26) : a_j^k X \leqslant \overline{a_j\tau_i}X' \vee a_j^k A_i$ $(27) : a_j X' \leqslant \overline{a_j\tau_i}X \vee a_j A_i$

$(13) : a_j^k a_l A_i' \leqslant a_j^k A_i'$ $(12) : e' a_i^j A_i' \leqslant e' A$

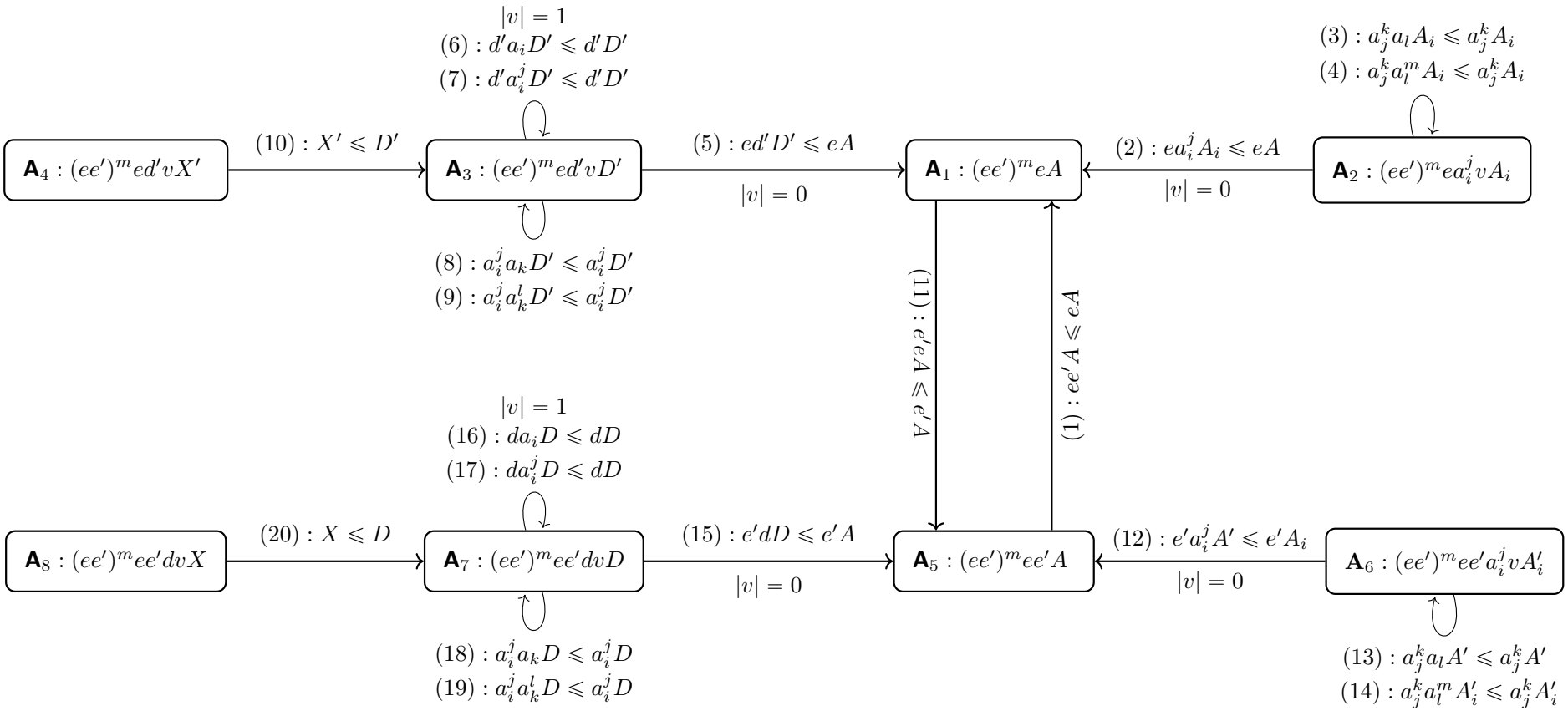
$$\begin{aligned} ea_1^2 X &\leqslant ea_1^2 a_3 X' \vee ea_1^2 A_1 \\ &\leqslant ea_1^2 a_3^3 a_3^3 X \vee ea_1^2 a_3 A_1' \vee ea_1^2 A_1 \\ &\leqslant ea_1^2 a_3^3 a_3^3 X \vee ea_1^2 A_1' \vee ea_1^2 A_1 \leqslant ?? \end{aligned}$$

Similarly, the auxiliary configurations from the deletion rules $(29) : a_i^j \leqslant e' \vee d'$ and $(30) : a_i^j \leqslant e \vee d$ prohibit deleting before appending.

Rewrite System Successfully Simulates Accepted Words

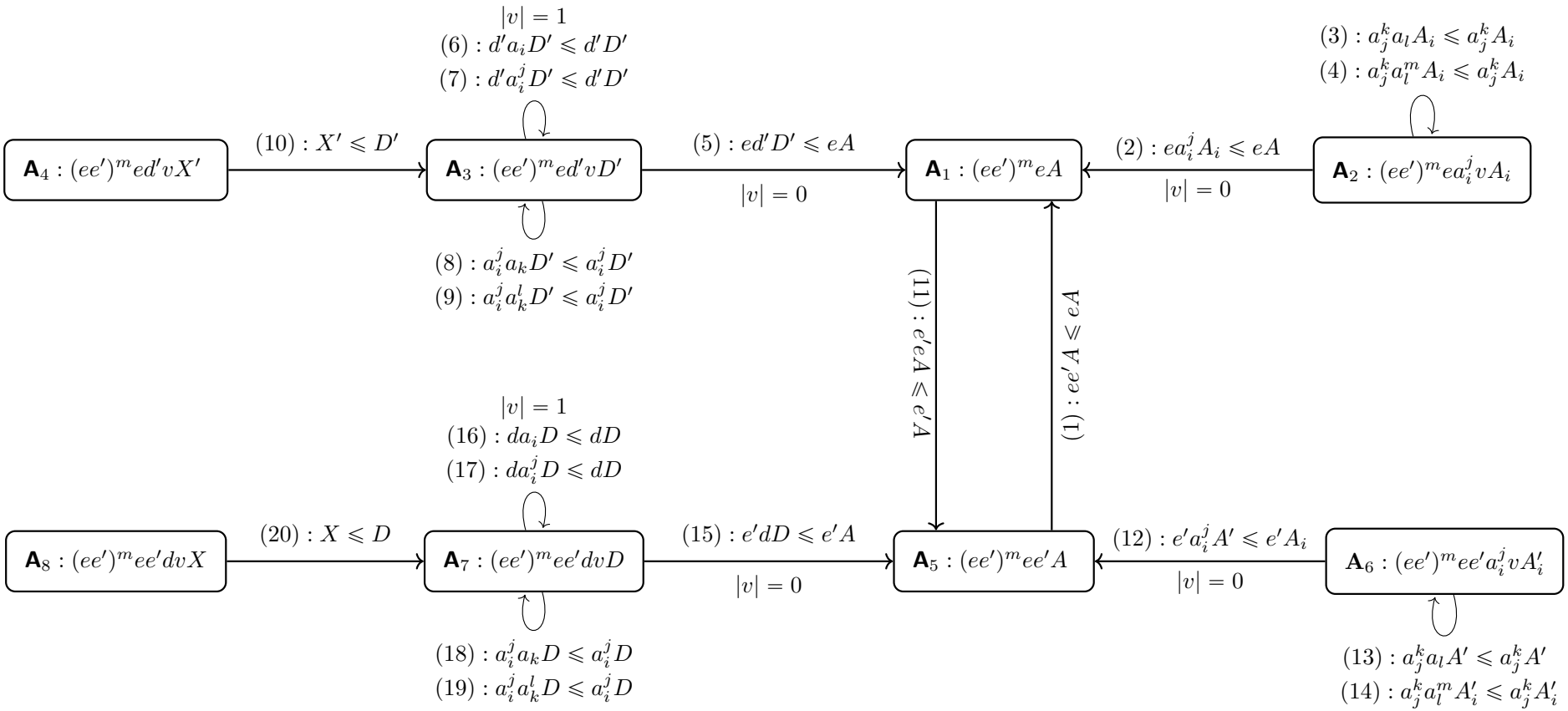


Rewrite System Successfully Simulates Accepted Words



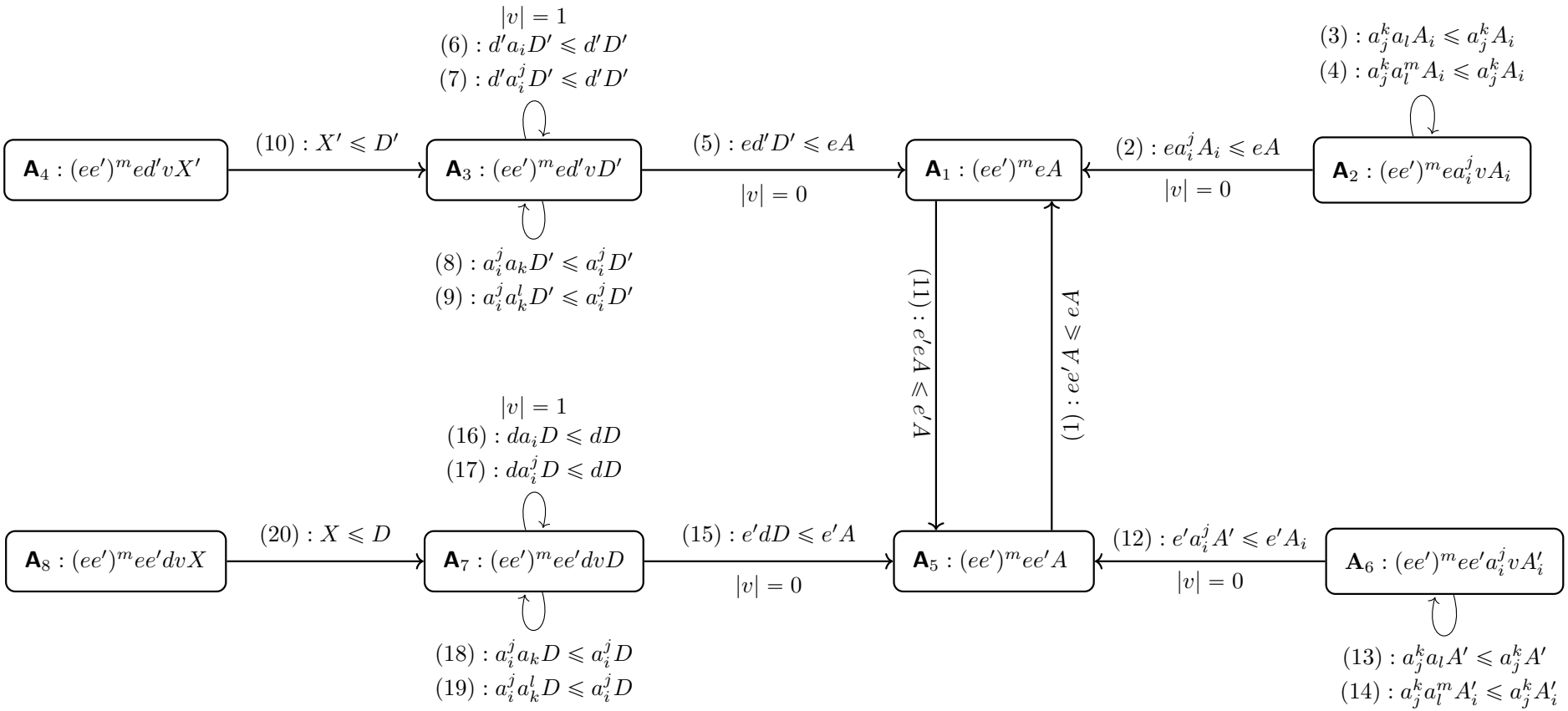
$$ea_2^3X \leqslant \underbrace{ea_2^3a_1X'}_{\text{Main}} \vee \underbrace{ea_2^3A_2}_{\text{Aux:}\mathbf{A}_2}$$

Rewrite System Successfully Simulates Accepted Words



$$ea_2^3X \leqslant \underbrace{ea_2^3a_1X'}_{\text{Main}} \vee \underbrace{ea_2^3A_2}_{\text{Aux:}\mathbf{A}_2} \leqslant \underbrace{ee'a_1X'}_{\text{Main}} \vee \underbrace{ed'a_1X'}_{\text{Aux:}\mathbf{A}_4} \vee \underbrace{ea_2^3A_2}_{\text{Aux:}\mathbf{A}_2}$$

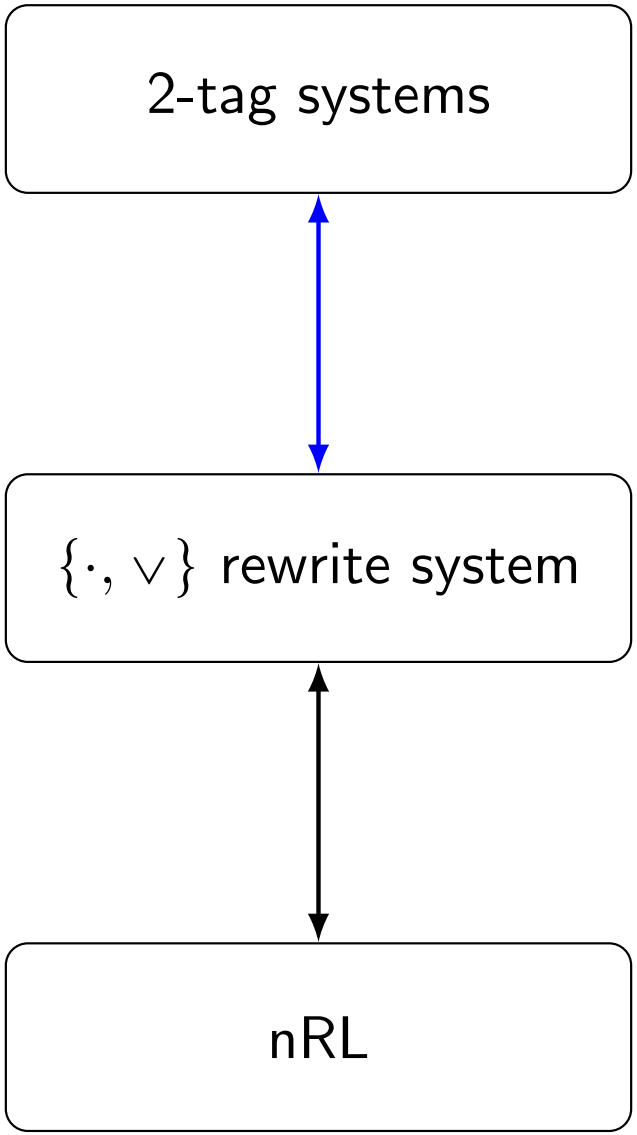
Rewrite System Successfully Simulates Accepted Words



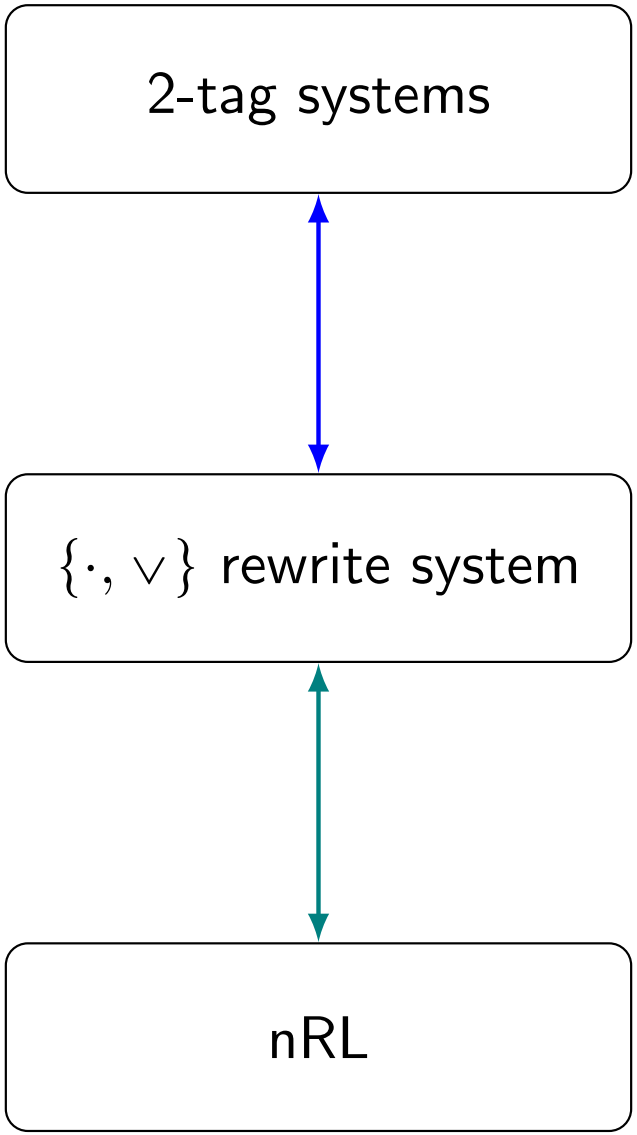
$$ea_2^3X \leqslant \underbrace{ea_2^3a_1X'}_{\text{Main}} \vee \underbrace{ea_2^3A_2}_{\text{Aux:}\mathbf{A}_2} \leqslant \underbrace{ee'a_1X'}_{\text{Main}} \vee \underbrace{ed'a_1X'}_{\text{Aux:}\mathbf{A}_4} \vee \underbrace{ea_2^3A_2}_{\text{Aux:}\mathbf{A}_2}$$

Fact: Let $\tau : \mathcal{T} \to \mathcal{T}^*$ be a 2-tag system: $w \in \mathsf{Acc}(\tau)$ iff $\overrightarrow{e\overline{w}X} \in \mathsf{Acc}(R_\tau)$.

Overview



Overview



Main Result

Theorem: Let $\mathcal{V}$ be an variety in the interval $[\mathsf{RL}, \mathsf{nRL}]$ and $\tau : \mathcal{T} \to \mathcal{T}^*$ a 2-tag system. Then for each $w \in \mathcal{T}^*$:
$w \in \mathsf{Acc}(\tau)$

Main Result

Theorem: Let $\mathcal{V}$ be an variety in the interval $[\mathsf{RL}, \mathsf{nRL}]$ and $\tau : \mathcal{T} \to \mathcal{T}^*$ a 2-tag system. Then for each $w \in \mathcal{T}^*$:
$w \in \mathsf{Acc}(\tau)$ iff $\overrightarrow{e\overline{w}X} \in \mathsf{Acc}(R_\tau)$

Main Result

Theorem: Let $\mathcal{V}$ be an variety in the interval $[\mathsf{RL}, \mathsf{nRL}]$ and $\tau : \mathcal{T} \to \mathcal{T}^*$ a 2-tag system. Then for each $w \in \mathcal{T}^*$:
$w \in \mathsf{Acc}(\tau)$ iff $\overrightarrow{e\overline{w}X} \in \mathsf{Acc}(R_\tau)$ iff $\mathcal{V} \models (\&P \implies \overrightarrow{e\overline{w}X} \leqslant eX \vee eA)$.

Main Result

Theorem: Let $\mathcal{V}$ be an variety in the interval $[\mathsf{RL}, \mathsf{nRL}]$ and $\tau : \mathcal{T} \to \mathcal{T}^*$ a 2-tag system. Then for each $w \in \mathcal{T}^*$:
$w \in \mathsf{Acc}(\tau)$ iff $\overrightarrow{e\overline{w}X} \in \mathsf{Acc}(R_\tau)$ iff $\mathcal{V} \models (\&P \implies \overrightarrow{e\overline{w}X} \leqslant eX \vee eA)$.
Soundness: $\mathsf{nRL} \models$ the above quasi-equation.

Main Result

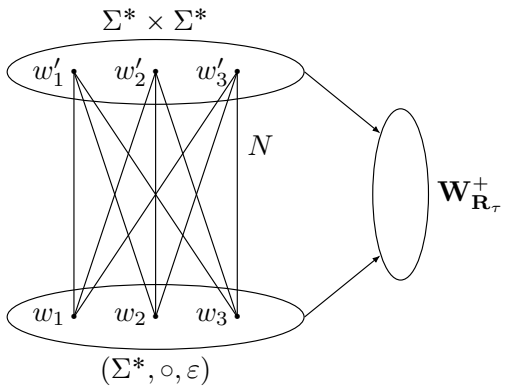
Theorem: Let $\mathcal{V}$ be an variety in the interval $[\mathsf{RL}, \mathsf{nRL}]$ and $\tau : \mathcal{T} \to \mathcal{T}^*$ a 2-tag system. Then for each $w \in \mathcal{T}^*$:
$w \in \mathsf{Acc}(\tau)$ iff $\overrightarrow{e\overline{w}X} \in \mathsf{Acc}(R_\tau)$ iff $\mathcal{V} \models (\&P \implies \overrightarrow{e\overline{w}X} \leqslant eX \vee eA)$.
Soundness: $\mathsf{nRL} \models$ the above quasi-equation.
Completeness: Define $\mathbf{W}_{\mathsf{R}_\tau} := (\Sigma^*, \Sigma^* \times \Sigma^*, N_{R_\tau}, \circ, \varepsilon)$ where $xN_{R_\tau}(y, z) \iff \overrightarrow{yxz} \in \mathsf{Acc}(\tau)$ for $x, y, z \in \Sigma^*$, Σ is the alphabet of R_τ, $(\Sigma^*, \circ, \varepsilon)$ is the free monoid over Σ.

Main Result

Theorem: Let $\mathcal{V}$ be an variety in the interval $[\mathsf{RL}, \mathsf{nRL}]$ and $\tau : \mathcal{T} \to \mathcal{T}^*$ a 2-tag system. Then for each $w \in \mathcal{T}^*$:
$w \in \mathsf{Acc}(\tau)$ iff $\overrightarrow{e\overline{w}X} \in \mathsf{Acc}(R_\tau)$ iff $\mathcal{V} \models (\&P \implies \overrightarrow{e\overline{w}X} \leqslant eX \vee eA)$.
Soundness: $\mathsf{nRL} \models$ the above quasi-equation.
Completeness: Define $\mathbf{W}_{\mathsf{R}_\tau} := (\Sigma^*, \Sigma^* \times \Sigma^*, N_{R_\tau}, \circ, \varepsilon)$ where $xN_{R_\tau}(y, z) \iff \overrightarrow{yxz} \in \mathsf{Acc}(\tau)$ for $x, y, z \in \Sigma^*$, Σ is the alphabet of R_τ, $(\Sigma^*, \circ, \varepsilon)$ is the free monoid over Σ.
$\mathbf{W}_{\mathsf{R}_\tau}$ is a *residuated frame* which implies that $\mathbf{W}^+_{R_\tau} \in \mathsf{nRL}$.

Main Result

Theorem: Let $\mathcal{V}$ be an variety in the interval [RL, nRL] and $\tau : \mathcal{T} \to \mathcal{T}^*$ a 2-tag system. Then for each $w \in \mathcal{T}^*$:
$w \in \mathsf{Acc}(\tau)$ iff $\overrightarrow{e\overline{w}X} \in \mathsf{Acc}(R_\tau)$ iff $\mathcal{V} \models (\&P \implies \overrightarrow{e\overline{w}X} \leqslant eX \vee eA)$.
Soundness: nRL $\models$ the above quasi-equation.
Completeness: Define $\mathbf{W}_{\mathsf{R}_\tau} := (\Sigma^*, \Sigma^* \times \Sigma^*, N_{R_\tau}, \circ, \varepsilon)$ where $xN_{R_\tau}(y, z) \iff \overrightarrow{yxz} \in \mathsf{Acc}(\tau)$ for $x, y, z \in \Sigma^*$, Σ is the alphabet of R_τ, $(\Sigma^*, \circ, \varepsilon)$ is the free monoid over Σ.
$\mathbf{W}_{\mathsf{R}_\tau}$ is a *residuated frame* which implies that $\mathbf{W}^+_{R_\tau} \in$ nRL.



Corollary: Any variety in [RL, nRL] has an undecidable word problem/quasi-eq. theory. (Any logic in $[\mathbf{nFL}, \mathbf{FL}]$ has an undecidable deducibility problem)

Future Work

- Extending to nRL $+ (xy = yx) + (x \leqslant x^2)$ $(\mathbf{nFL_{ec}})$

Future Work

- Extending to nRL $+ (xy = yx) + (x \leqslant x^2)$ $(\mathbf{nFL_{ec}})$
- What other structural rules can be added while maintaining undecidability? (E.g. knotted rules, simple equations, etc)

Future Work

- Extending to nRL + $(xy = yx)$ + $(x \leqslant x^2)$ $(\mathbf{nFL_{ec}})$
- What other structural rules can be added while maintaining undecidability? (E.g. knotted rules, simple equations, etc)
- Can we obtain undecidability of the equational theory for certain subvarieties of nRL?

Thank You!!