Finding bases in linear categories using rewriting.

Benjamin Dupont

Institut Camille Jordan, Université Lyon 1

Algebra Seminar, Ottawa

Outline

I. Rewriting theory

II. String rewriting

III. Rewriting in linear 2-categories

IV. Extension to rewriting modulo

I. Rewriting theory

Rewriting theory

- Rewriting is a combinatorial theory of equivalence classes.
    - Consists in orienting the equations.

Rewriting theory

- Rewriting is a combinatorial theory of equivalence classes.
  - Consists in orienting the equations.
  - Thue '14: rewriting in semi-groups.

.

Rewriting theory

- Rewriting is a combinatorial theory of equivalence classes.
  - Consists in orienting the equations.
  - Thue '14: rewriting in semi-groups.
  - Church-Rosser '36: lambda-calculus and beta-reductions.

.

Rewriting theory

- Rewriting is a combinatorial theory of equivalence classes.
  - Consists in orienting the equations.
  - Thue '14: rewriting in semi-groups.
  - Church-Rosser '36: lambda-calculus and beta-reductions.
  - Newman '42: abstract rewriting.

Rewriting theory

- Rewriting is a combinatorial theory of equivalence classes.
  - Consists in orienting the equations.
  - Thue '14: rewriting in semi-groups.
  - Church-Rosser '36: lambda-calculus and beta-reductions.
  - Newman '42: abstract rewriting.
  - Knuth-Bendix '70, Nivat '72: completion procedures, characterization of local confluence in terms of overlappings.

Rewriting theory

- Rewriting is a combinatorial theory of equivalence classes.
  - Consists in orienting the equations.
  - Thue '14: rewriting in semi-groups.
  - Church-Rosser '36: lambda-calculus and beta-reductions.
  - Newman '42: abstract rewriting.
  - Knuth-Bendix '70, Nivat '72: completion procedures, characterization of local confluence in terms of overlappings.
- Algebraic rewriting: deduce properties of an algebraic structure presented by generators and relations.

Rewriting theory

- Rewriting is a combinatorial theory of equivalence classes.
  - Consists in orienting the equations.
  - Thue '14: rewriting in semi-groups.
  - Church-Rosser '36: lambda-calculus and beta-reductions.
  - Newman '42: abstract rewriting.
  - Knuth-Bendix '70, Nivat '72: completion procedures, characterization of local confluence in terms of overlappings.
- Algebraic rewriting: deduce properties of an algebraic structure presented by generators and relations.
  - Computation of syzygies, i.e. relations among relations.

Rewriting theory

- Rewriting is a combinatorial theory of equivalence classes.
  - Consists in orienting the equations.
  - Thue '14: rewriting in semi-groups.
  - Church-Rosser '36: lambda-calculus and beta-reductions.
  - Newman '42: abstract rewriting.
  - Knuth-Bendix '70, Nivat '72: completion procedures, characterization of local confluence in terms of overlappings.
- Algebraic rewriting: deduce properties of an algebraic structure presented by generators and relations.
  - Computation of syzygies, i.e. relations among relations.
  - Computation of linear bases.

Rewriting theory

- Rewriting is a combinatorial theory of equivalence classes.
  - Consists in orienting the equations.
  - Thue '14: rewriting in semi-groups.
  - Church-Rosser '36: lambda-calculus and beta-reductions.
  - Newman '42: abstract rewriting.
  - Knuth-Bendix '70, Nivat '72: completion procedures, characterization of local confluence in terms of overlappings.
- Algebraic rewriting: deduce properties of an algebraic structure presented by generators and relations.
  - Computation of syzygies, i.e. relations among relations.
  - Computation of linear bases.
  - Proofs of Koszulity.

Rewriting theory

- Rewriting is a combinatorial theory of equivalence classes.
  - Consists in orienting the equations.
  - Thue '14: rewriting in semi-groups.
  - Church-Rosser '36: lambda-calculus and beta-reductions.
  - Newman '42: abstract rewriting.
  - Knuth-Bendix '70, Nivat '72: completion procedures, characterization of local confluence in terms of overlappings.
- Algebraic rewriting: deduce properties of an algebraic structure presented by generators and relations.
  - Computation of syzygies, i.e. relations among relations.
  - Computation of linear bases.
  - Proofs of Koszulity.
  - Computation of free resolutions and cofibrant replacements, Anick '84.

Algebraic contexts of rewriting

- Rewriting has been developed for various algebraic structures:
  - String rewriting systems, Thue.

Algebraic contexts of rewriting

- Rewriting has been developed for various algebraic structures:
  - String rewriting systems, Thue.
  - Universal algebra (term rewriting systems), Knuth-Bendix '70.

Algebraic contexts of rewriting

- Rewriting has been developed for various algebraic structures:
  - String rewriting systems, Thue.
  - Universal algebra (term rewriting systems), Knuth-Bendix '70.
  - Commutative algebras, Buchberger '65.

Algebraic contexts of rewriting

- Rewriting has been developed for various algebraic structures:
  - String rewriting systems, Thue.
  - Universal algebra (term rewriting systems), Knuth-Bendix '70.
  - Commutative algebras, Buchberger '65.
  - Associative algebras, Bokut '76, Bergman '78, Mora '86.

Algebraic contexts of rewriting

- Rewriting has been developed for various algebraic structures:
  - String rewriting systems, Thue.
  - Universal algebra (term rewriting systems), Knuth-Bendix '70.
  - Commutative algebras, Buchberger '65.
  - Associative algebras, Bokut '76, Bergman '78, Mora '86.
  - Operads, Dotsenko-Khoroshkin '10.

Algebraic contexts of rewriting

- Rewriting has been developed for various algebraic structures:
  - String rewriting systems, Thue.
  - Universal algebra (term rewriting systems), Knuth-Bendix '70.
  - Commutative algebras, Buchberger '65.
  - Associative algebras, Bokut '76, Bergman '78, Mora '86.
  - Operads, Dotsenko-Khoroshkin '10.
  - Higher-dimensional globular strict categories, Guiraud-Malbos '09.

Algebraic contexts of rewriting

- Rewriting has been developed for various algebraic structures:
  - String rewriting systems, Thue.
  - Universal algebra (term rewriting systems), Knuth-Bendix '70.
  - Commutative algebras, Buchberger '65.
  - Associative algebras, Bokut '76, Bergman '78, Mora '86.
  - Operads, Dotsenko-Khoroshkin '10.
  - Higher-dimensional globular strict categories, Guiraud-Malbos '09.
- **Objective:** Develop rewriting methods to study diagrammatic algebras that arise in representation theory.

Algebraic contexts of rewriting

- Rewriting has been developed for various algebraic structures:
  - String rewriting systems, Thue.
  - Universal algebra (term rewriting systems), Knuth-Bendix '70.
  - Commutative algebras, Buchberger '65.
  - Associative algebras, Bokut '76, Bergman '78, Mora '86.
  - Operads, Dotsenko-Khoroshkin '10.
  - Higher-dimensional globular strict categories, Guiraud-Malbos '09.
- **Objective:** Develop rewriting methods to study diagrammatic algebras that arise in representation theory.
  - Khovanov-Lauda-Rouquier (KLR) algebras which categorify quantum groups.
  - Heisenberg categorifications.
  - Partition, Brauer and Birman-Wenzl algebras.

Algebraic contexts of rewriting

- Rewriting has been developed for various algebraic structures:
  - String rewriting systems, Thue.
  - Universal algebra (term rewriting systems), Knuth-Bendix '70.
  - Commutative algebras, Buchberger '65.
  - Associative algebras, Bokut '76, Bergman '78, Mora '86.
  - Operads, Dotsenko-Khoroshkin '10.
  - Higher-dimensional globular strict categories, Guiraud-Malbos '09.
- **Objective:** Develop rewriting methods to study diagrammatic algebras that arise in representation theory.
  - Khovanov-Lauda-Rouquier (KLR) algebras which categorify quantum groups.
  - Heisenberg categorifications.
  - Partition, Brauer and Birman-Wenzl algebras.
- **Questions:**
  - Solve the word problem: decide the equality of two diagrams.
  - Computation of linear bases.
  - Computation of coherent presentations.
  - Explicit proofs of categorification results.

Algebraic contexts of rewriting

- Rewriting has been developed for various algebraic structures:
  - String rewriting systems, Thue.
  - Universal algebra (term rewriting systems), Knuth-Bendix '70.
  - Commutative algebras, Buchberger '65.
  - Associative algebras, Bokut '76, Bergman '78, Mora '86.
  - Operads, Dotsenko-Khoroshkin '10.
  - Higher-dimensional globular strict categories, Guiraud-Malbos '09.
- **Objective:** Develop rewriting methods to study diagrammatic algebras that arise in representation theory.
  - Khovanov-Lauda-Rouquier (KLR) algebras which categorify quantum groups.
  - Heisenberg categorifications.
  - Partition, Brauer and Birman-Wenzl algebras.
- **Questions:**
  - Solve the word problem: decide the equality of two diagrams.
  - Computation of linear bases. ✓
  - Computation of coherent presentations.
  - Explicit proofs of categorification results.

II. String rewriting

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc, \;\; ada \to dc, \;\; bc \to dab, \;\; db \to c, \;\; dcb \to acc.$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc,\ ada \to dc,\ bc \to dab,\ db \to c,\ dcb \to acc.$$

$$abc$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc, ada \to dc,\ bc \to dab,\ db \to c,\ dcb \to acc.$$

$$\begin{tikzcd} & abc \arrow[dl] \\ bcc & \end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc, \;\; ada \to dc, \;\; bc \to dab, db \to c, \;\; dcb \to acc.$$

$$\begin{tikzcd}
 & abc \arrow[dl] \arrow[dr] & \\
bcc & & adab
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc,\ \ ada \to dc,\ \ \ bc \to dab, db \to c,\ \ dcb \to acc.$$

$$\begin{tikzcd}
 & abc \arrow[dl] \arrow[dr] & \\
bcc \arrow[d] & & adab \\
dabc & &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc,\ \ ada \to dc,\ \ bc \to dab,\ \ db \to c,\ \ dcb \to acc.$$

$$\begin{tikzcd}
 & abc \arrow[dl] \arrow[dr] & \\
bcc \arrow[d] & & adab \\
dabc & &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc,\ ada \to dc,\ \ bc \to dab,\ \ db \to c,\ \ dcb \to acc.$$

$$\begin{tikzcd}
 & abc \arrow[dl] \arrow[dr] & \\
bcc \arrow[d] & & adab \\
dabc \arrow[d] & & \\
dbcc & &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc,\; ada \to dc,\; bc \to dab,\; db \to c,\; dcb \to acc.$$

$$\begin{tikzcd}
 & abc \arrow[dl] \arrow[dr] & \\
bcc \arrow[d] & & adab \\
dabc \arrow[d] & & \\
dbcc & &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc,\ \ ada \to dc,\ \ \ bc \to dab, db \to c,\ \ dcb \to acc.$$

$$\begin{tikzcd}
 & & abc \arrow[dl] \arrow[dr] & \\
 & bcc \arrow[d] & & adab \\
 & dabc \arrow[d] & & \\
 & dbcc \arrow[dl] & & \\
ddabc & & &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc, \;\; ada \to dc, \;\; bc \to dab, \;\; db \to c, dcb \to acc.$$

$$\begin{tikzcd}
 & & abc \arrow[dl] \arrow[dr] & \\
 & bcc \arrow[d] & & adab \\
 & dabc \arrow[d] & & \\
 & dbcc \arrow[dl] \arrow[dr] & & \\
ddabc & & ccc &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc,\ \ ada \to dc,\ \ bc \to dab,\ \ db \to c,\ \ dcb \to acc.$$

$$\begin{tikzcd}
 & & abc \arrow[dl] \arrow[dr] & \\
 & bcc \arrow[d] & & adab \\
 & dabc \arrow[d] & & \\
 & dbcc \arrow[dl] \arrow[dr] & & \\
ddabc & & ccc &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc, \;\; ada \to dc, \;\; bc \to dab, \;\; db \to c, \;\; dcb \to acc.$$

$$\begin{tikzcd}
 & & abc \arrow[dl] \arrow[dr] & \\
 & bcc \arrow[d] & & adab \\
 & dabc \arrow[d] & & \\
 & dbcc \arrow[dl] \arrow[dr] & & \\
ddabc \arrow[d] & & ccc & \\
\cdots & & &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc, \quad ada \to dc, bc \to dab, \; db \to c, \; dcb \to acc.$$

$$\begin{tikzcd}
 & & abc \arrow[dl] \arrow[dr] & \\
 & bcc \arrow[d] & & adab \arrow[dl] \\
 & dabc \arrow[d] & dcb & \\
 & dbcc \arrow[dl] \arrow[dr] & & \\
ddabc \arrow[d] & & ccc & \\
\cdots & & &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

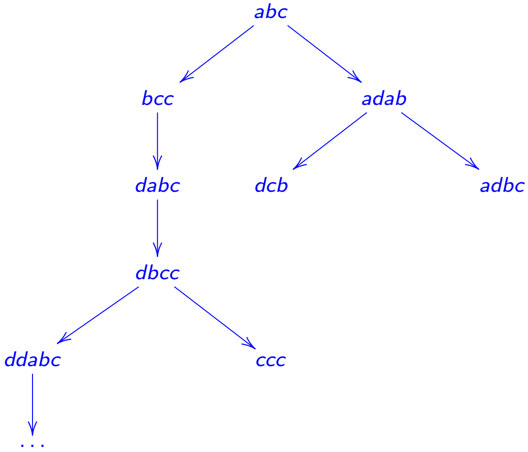
$$ab \to bc, ada \to dc,\ \ bc \to dab,\ \ db \to c,\ \ dcb \to acc.$$

$$\begin{tikzcd}
 & & abc \arrow[dl] \arrow[dr] & & \\
 & bcc \arrow[d] & & adab \arrow[dl] \arrow[dr] & \\
 & dabc \arrow[d] & dcb & & adbc \\
 & dbcc \arrow[dl] \arrow[dr] & & & \\
ddabc \arrow[d] & & ccc & & \\
\cdots & & & &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

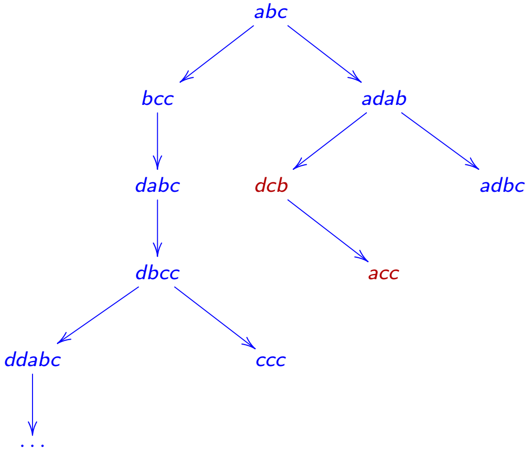
$$ab \to bc, \;\; ada \to dc, \;\; bc \to dab, \;\; db \to c, \;\; dcb \to acc.$$



String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc, \ ada \to dc, \ bc \to dab, \ db \to c, \quad dcb \to acc.$$



String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc, \;\; ada \to dc, \;\; bc \to dab, \;\; db \to c, \;\; dcb \to acc.$$

$$\begin{tikzcd}
 & & abc \arrow[dl] \arrow[dr] & & \\
 & bcc \arrow[d] & & adab \arrow[dl] \arrow[dr] & \\
 & dabc \arrow[d] & dcb \arrow[dr] & & adbc \\
 & dbcc \arrow[dl] \arrow[dr] & & acc & \\
ddabc \arrow[d] & & ccc & & \\
\cdots & & & &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc, \;\; ada \to dc, \;\; bc \to dab, \;\; db \to c, dcb \to acc.$$

$$\begin{tikzcd}
 & & abc \arrow[dl] \arrow[dr] & & \\
 & bcc \arrow[d] & & adab \arrow[dl] \arrow[dr] & \\
 & dabc \arrow[d] & dcb \arrow[dr] & & adbc \arrow[dl] \\
 & dbcc \arrow[dl] \arrow[dr] & & acc & \\
ddabc \arrow[d] & & ccc & & \\
\cdots & & & &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

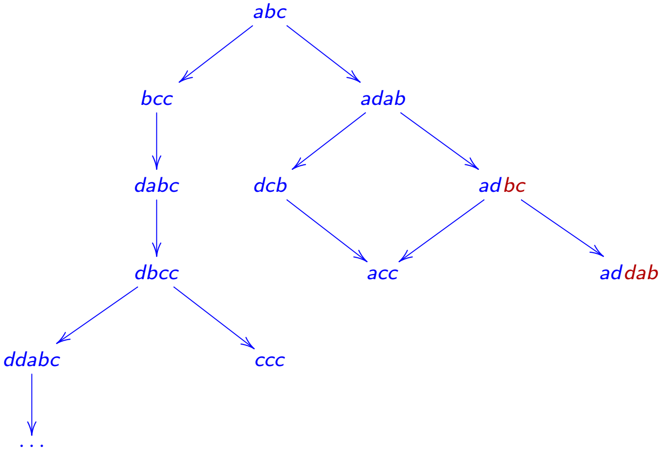
$$ab \to bc,\ \ ada \to dc,\ \ bc \to dab,\ \ db \to c,\ \ dcb \to acc.$$

$$\begin{tikzcd}
 & & abc \arrow[dl] \arrow[dr] & & \\
 & bcc \arrow[d] & & adab \arrow[dl] \arrow[dr] & \\
 & dabc \arrow[d] & dcb \arrow[dr] & & adbc \arrow[dl] \\
 & dbcc \arrow[dl] \arrow[dr] & & acc & \\
ddabc \arrow[d] & & ccc & & \\
\cdots & & & &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc,\ \ ada \to dc,\ \ \ bc \to dab, db \to c,\ \ dcb \to acc.$$



String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc, \;\; ada \to dc, \;\; bc \to dab, \;\; db \to c, \;\; dcb \to acc.$$

$$\begin{tikzcd}
 & & abc \arrow[dl] \arrow[dr] & & & \\
 & bcc \arrow[d] & & adab \arrow[dl] \arrow[dr] & & \\
 & dabc \arrow[d] & dcb \arrow[dr] & & adbc \arrow[dl] \arrow[dr] & \\
 & dbcc \arrow[dl] \arrow[dr] & & acc & & addab \\
ddabc \arrow[d] & & ccc & & & \\
\cdots & & & & &
\end{tikzcd}$$

String rewriting

- **Example**: $X := \{a, b, c, d\}$ an alphabet and we consider the 5 rewriting rules:

$$ab \to bc, \;\; ada \to dc, \;\; bc \to dab, \;\; db \to c, \;\; dcb \to acc.$$

$$\begin{tikzcd}
 & & abc \arrow[dl] \arrow[dr] & & & \\
 & bcc \arrow[d] & & adab \arrow[dl] \arrow[dr] & & \\
 & dabc \arrow[d] & dcb \arrow[dr] & & adbc \arrow[dl] \arrow[dr] & \\
 & dbcc \arrow[dl] \arrow[dr] & & acc & & addab \arrow[d] \\
ddabc \arrow[d] & & ccc & & & \dots \\
\dots & & & & &
\end{tikzcd}$$

Normal forms and termination

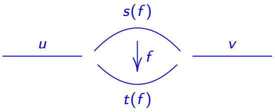
- Let (X, R) be a string rewriting system and X^* the free monoid on X. A rewriting step of (X, R) is a reduction

$$us(f)v \to ut(f)v, \text{ for } u, v \in X^* \text{ and } f : s(f) \to t(f) \in R$$

Normal forms and termination

- Let (X, R) be a string rewriting system and X^* the free monoid on X. A rewriting step of (X, R) is a reduction

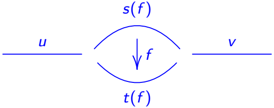
$us(f)v \to ut(f)v$, for $u, v \in X^*$ and $f : s(f) \to t(f) \in R$



Normal forms and termination

- Let (X, R) be a string rewriting system and X^* the free monoid on X. A rewriting step of (X, R) is a reduction

$$us(f)v \to ut(f)v, \text{ for } u, v \in X^* \text{ and } f : s(f) \to t(f) \in R$$

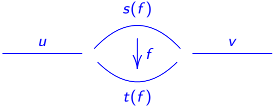


- An element x of X^* is a normal form if there does not exist y in X^* such that $x \to y$.

Normal forms and termination

- Let (X, R) be a string rewriting system and X^* the free monoid on X. A rewriting step of (X, R) is a reduction

$$us(f)v \to ut(f)v, \text{ for } u, v \in X^* \text{ and } f : s(f) \to t(f) \in R$$



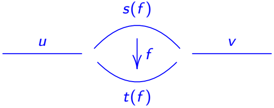
- An element x of X^* is a normal form if there does not exist y in X^* such that $x \to y$.

- (X, R) is terminating if there does not exist any infinite rewriting sequence in (X, R).

Normal forms and termination

- Let (X, R) be a string rewriting system and X^* the free monoid on X. A rewriting step of (X, R) is a reduction

$us(f)v \to ut(f)v$, for $u, v \in X^*$ and $f : s(f) \to t(f) \in R$



- An element x of X^* is a normal form if there does not exist y in X^* such that $x \to y$.

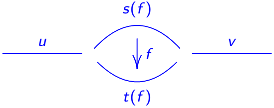
- (X, R) is terminating if there does not exist any infinite rewriting sequence in (X, R).

- If (X, R) terminates, each element $x \in X^*$ admits at least one normal form.

Normal forms and termination

- Let (X, R) be a string rewriting system and X^* the free monoid on X. A rewriting step of (X, R) is a reduction

$us(f)v \to ut(f)v$, for $u, v \in X^*$ and $f : s(f) \to t(f) \in R$



- An element x of X^* is a normal form if there does not exist y in X^* such that $x \to y$.

- (X, R) is terminating if there does not exist any infinite rewriting sequence in (X, R).

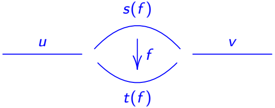
- If (X, R) terminates, each element $x \in X^*$ admits at least one normal form.

- If (X, R) is convergent, i.e. both terminating and confluent, each element $x \in X^*$ admits a unique normal form, denoted by $\hat{x}$.

Normal forms and termination

- Let (X, R) be a string rewriting system and X^* the free monoid on X. A rewriting step of (X, R) is a reduction

$$us(f)v \to ut(f)v, \text{ for } u, v \in X^* \text{ and } f : s(f) \to t(f) \in R$$



- An element x of X^* is a normal form if there does not exist y in X^* such that $x \to y$.

- (X, R) is terminating if there does not exist any infinite rewriting sequence in (X, R).

- If (X, R) terminates, each element $x \in X^*$ admits at least one normal form.

- If (X, R) is convergent, i.e. both terminating and confluent, each element $x \in X^*$ admits a unique normal form, denoted by $\hat{x}$.

- **Example:**

$$\begin{tikzcd} a \arrow[rr, bend left] \arrow[dr] & & a' \arrow[ll, bend left] \arrow[dl] \\ & b & \end{tikzcd}$$

Confluence and branchings

- A branching (resp. local branching) of (X, R) is:

$$\begin{tikzcd} & v \\ u \arrow[ur, bend left, "f"] \arrow[dr, bend right, "g"'] & \\ & w \end{tikzcd}$$

where f are g rewriting paths (resp. rewriting steps) and u, v, w are in X^*.

Confluence and branchings

- A branching (resp. local branching) of (X, R) is:

$$
\begin{tikzcd}
 & v \arrow[dr, dotted, "f'"] & \\
u \arrow[ur, bend left, "f"] \arrow[dr, bend right, "g"'] & & u' \\
 & w \arrow[ur, dotted, "g'"'] &
\end{tikzcd}
$$

where f are g rewriting paths (resp. rewriting steps) and u, v, w are in X^*.

- A (local) branching is confluent if there exists rewriting paths that close the diagram.

Confluence and branchings

- A branching (resp. local branching) of (X, R) is:

$$
\begin{tikzcd}
 & v \arrow[dr, dotted, "f'"] & \\
u \arrow[ur, bend left, "f"] \arrow[dr, bend right, "g"'] & & u' \\
 & w \arrow[ur, dotted, "g'"'] &
\end{tikzcd}
$$

  where f are g rewriting paths (resp. rewriting steps) and u, v, w are in X^*.

- A (local) branching is confluent if there exists rewriting paths that close the diagram.

- **Theorem (Newman Lemma):** If (X, R) is terminating, local confluence is equivalent to confluence.

Confluence and branchings

- A branching (resp. local branching) of (X, R) is:

$$\begin{tikzcd} & v \arrow[dr, dotted, "f'"] & \\ u \arrow[ur, bend left, "f"] \arrow[dr, bend right, "g"'] & & u' \\ & w \arrow[ur, dotted, "g'"'] & \end{tikzcd}$$

  where f are g rewriting paths (resp. rewriting steps) and u, v, w are in X^*.

- A (local) branching is confluent if there exists rewriting paths that close the diagram.

- **Theorem (Newman Lemma):** If (X, R) is terminating, local confluence is equivalent to confluence.

- Local branchings are divided into 3 families:

Confluence and branchings

- A branching (resp. local branching) of (X, R) is:

$$\begin{tikzcd} & v \arrow[dr, dotted, "f'"] & \\ u \arrow[ur, bend left, "f"] \arrow[dr, bend right, "g"'] & & u' \\ & w \arrow[ur, dotted, "g'"'] & \end{tikzcd}$$

  where f are g rewriting paths (resp. rewriting steps) and u, v, w are in X^*.

- A (local) branching is confluent if there exists rewriting paths that close the diagram.

- **Theorem (Newman Lemma):** If (X, R) is terminating, local confluence is equivalent to confluence.

- Local branchings are divided into 3 families:

$$\begin{tikzcd} u \arrow[r, bend left, "f"] \arrow[r, bend right, "f"'] & v \end{tikzcd}$$

Aspherical

Confluence and branchings

- A branching (resp. local branching) of (X, R) is:

$$\begin{tikzcd}
 & v \arrow[dr, dotted, "f'"] & \\
u \arrow[ur, bend left, "f"] \arrow[dr, bend right, "g"'] & & u' \\
 & w \arrow[ur, dotted, "g'"'] &
\end{tikzcd}$$

  where f are g rewriting paths (resp. rewriting steps) and u, v, w are in X^*.

- A (local) branching is confluent if there exists rewriting paths that close the diagram.

- **Theorem (Newman Lemma):** If (X, R) is terminating, local confluence is equivalent to confluence.

- Local branchings are divided into 3 families:

$$\begin{tikzcd}
u \arrow[r, bend left, "f"] \arrow[r, bend right, "f"'] & v
\end{tikzcd}$$

Aspherical

$$\begin{tikzcd}
 & u'v \\
uv \arrow[ur, bend left, "fv"] \arrow[dr, bend right, "ug"'] & \\
 & uv'
\end{tikzcd}$$

Peiffer

Confluence and branchings

- A branching (resp. local branching) of (X, R) is:

$$\begin{tikzcd}[row sep=small] & v \arrow[dr, dotted, "f'"] & \\ u \arrow[ur, bend left, "f"] \arrow[dr, bend right, "g"'] & & u' \\ & w \arrow[ur, dotted, "g'"'] & \end{tikzcd}$$

  where f are g rewriting paths (resp. rewriting steps) and u, v, w are in X^*.

- A (local) branching is confluent if there exists rewriting paths that close the diagram.

- **Theorem (Newman Lemma):** If (X, R) is terminating, local confluence is equivalent to confluence.

- Local branchings are divided into 3 families:

$$\begin{tikzcd} u \arrow[r, bend left, "f"] \arrow[r, bend right, "f"'] & v \end{tikzcd}$$

Aspherical

$$\begin{tikzcd}[row sep=small] & u'v \\ uv \arrow[ur, bend left, "fv"] \arrow[dr, bend right, "ug"'] & \\ & uv' \end{tikzcd}$$

Peiffer

$\cdots$

Overlappings

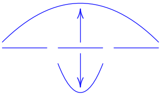
Critical branchings

- Local branchings are compared by the order $\sqsubseteq$ generated by $(f, g) \sqsubseteq (ufv, ugv)$ for $u,v \in X^*$. A critical branching is a minimal branching for $\sqsubseteq$.

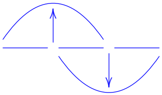
Critical branchings

- Local branchings are compared by the order $\sqsubseteq$ generated by $(f, g) \sqsubseteq (ufv, ugv)$ for $u,v \in X^*$. A critical branching is a minimal branching for $\sqsubseteq$.

- There are two forms of critical branchings:



and

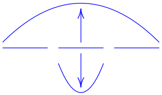
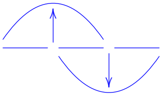


.

Critical branchings

- Local branchings are compared by the order $\sqsubseteq$ generated by $(f, g) \sqsubseteq (ufv, ugv)$ for $u,v \in X^*$. A critical branching is a minimal branching for $\sqsubseteq$.

- There are two forms of critical branchings:

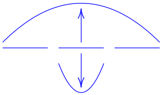
 and  .

- **Theorem (Critical pair lemma):** (X, R) is locally confluent iff all its critical branchings are confluent.

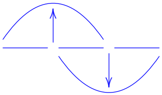
Critical branchings

- Local branchings are compared by the order $\sqsubseteq$ generated by $(f, g) \sqsubseteq (ufv, ugv)$ for $u, v \in X^*$. A critical branching is a minimal branching for $\sqsubseteq$.

- There are two forms of critical branchings:



and



.

- **Theorem (Critical pair lemma):** (X, R) is locally confluent iff all its critical branchings are confluent.

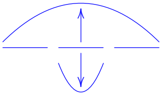
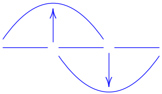
- Proof is case by case: aspherical and Peiffer branchings are always confluent.

$$\begin{tikzcd} & v \arrow[dr, "1_v"] & \\ u \arrow[ur, "f"] \arrow[dr, "f"'] & & v \\ & v \arrow[ur, "1_v"'] & \end{tikzcd} \qquad \begin{tikzcd} & u'v \arrow[dr, "u'g"] & \\ uv \arrow[ur, "fv"] \arrow[dr, "ug"'] & & u'v' \\ & uv' \arrow[ur, "fv'"'] & \end{tikzcd}$$

Critical branchings

- Local branchings are compared by the order $\sqsubseteq$ generated by $(f, g) \sqsubseteq (ufv, ugv)$ for $u, v \in X^*$. A critical branching is a minimal branching for $\sqsubseteq$.

- There are two forms of critical branchings:

 and  .

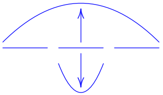
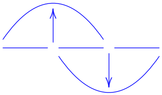
- **Theorem (Critical pair lemma):** (X, R) is locally confluent iff all its critical branchings are confluent.

- Proof is case by case: aspherical and Peiffer branchings are always confluent. For overlappings (f, g), there exists (h, k) such that $f = uhv$ and $g = ukv$.

$$
\begin{tikzcd}
 & v \arrow[dr, bend left, "1_v"] & \\
u \arrow[ur, bend left, "f"] \arrow[dr, bend right, "f"'] & & v \\
 & v \arrow[ur, bend right, "1_v"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & u'v \arrow[dr, bend left, "u'g"] & \\
uv \arrow[ur, bend left, "fv"] \arrow[dr, bend right, "ug"'] & & u'v' \\
 & uv' \arrow[ur, bend right, "fv'"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & x \arrow[dr, bend left, "h'"] & \\
w \arrow[ur, bend left, "h"] \arrow[dr, bend right, "k"'] & & w' \\
 & y \arrow[ur, bend right, "k'"'] &
\end{tikzcd}
$$

Critical branchings

- Local branchings are compared by the order $\sqsubseteq$ generated by $(f, g) \sqsubseteq (ufv, ugv)$ for $u, v \in X^*$. A critical branching is a minimal branching for $\sqsubseteq$.
- There are two forms of critical branchings:

 and  .

- **Theorem (Critical pair lemma):** (X, R) is locally confluent iff all its critical branchings are confluent.
- Proof is case by case: aspherical and Peiffer branchings are always confluent. For overlappings (f, g), there exists (h, k) such that $f = uhv$ and $g = ukv$.

$$\begin{tikzcd}[row sep=small] & v \arrow[dr, "1_v"] & \\ u \arrow[ur, "f"] \arrow[dr, "f"'] & & v \\ & v \arrow[ur, "1_v"'] & \end{tikzcd}
\qquad
\begin{tikzcd}[row sep=small] & u'v \arrow[dr, "u'g"] & \\ uv \arrow[ur, "fv"] \arrow[dr, "ug"'] & & u'v' \\ & uv' \arrow[ur, "fv'"'] & \end{tikzcd}
\qquad
\begin{tikzcd}[row sep=small] & x \arrow[dr, "h'"] & \\ w \arrow[ur, "h"] \arrow[dr, "k"'] & & w' \\ & y \arrow[ur, "k'"'] & \end{tikzcd}
\rightsquigarrow
\begin{tikzcd}[row sep=small] & uxv \arrow[dr, "uh'v"] & \\ uwv \arrow[ur, "f=uhv"] \arrow[dr, "g=ukv"'] & & uw'v \\ & uyv \arrow[ur, "uk'v"'] & \end{tikzcd}$$

The word problem

- Consider M a monoid presented by generators X and relations $R^{\textbf{n-o}}$, i.e.

$$M \simeq X / \equiv_{R^{\textbf{n-o}}},$$

that is $u = v$ in M iff $\overline{u} \overset{R}{\leftrightarrow} \overline{v}$ in X^* for representatives $\overline{u}$ and $\overline{v}$ of u and v in X^*.

The word problem

- Consider M a monoid presented by generators X and relations $R^{\textbf{n-o}}$, i.e.

$$M \simeq X/\equiv_{R^{\textbf{n-o}}},$$

that is $u = v$ in M iff $\overline{u} \overset{R}{\leftrightarrow} \overline{v}$ in X^* for representatives $\overline{u}$ and $\overline{v}$ of u and v in X^*.

- **Word problem:** given u and v in X^*, does $u = v$ in M ?

The word problem

- Consider M a monoid presented by generators X and relations $R^{\textbf{n-o}}$, i.e.

$$M \simeq X / \equiv_{R^{\textbf{n-o}}},$$

that is $u = v$ in M iff $\overline{u} \overset{R}{\leftrightarrow} \overline{v}$ in X^* for representatives $\overline{u}$ and $\overline{v}$ of u and v in X^*.

- **Word problem:** given u and v in X^*, does $u = v$ in M ?

- **Partial answer:** Fix an orientation R of rules in $R^{\textbf{n-o}}$. If (X, R) is convergent, this problem is decidable using the normal form algorithm.

The word problem

- Consider M a monoid presented by generators X and relations $R^{\text{n-o}}$, i.e.

$$M \simeq X/\equiv_{R^{\text{n-o}}},$$

that is $u = v$ in M iff $\overline{u} \overset{R}{\leftrightarrow} \overline{v}$ in X^* for representatives $\overline{u}$ and $\overline{v}$ of u and v in X^*.

- **Word problem:** given u and v in X^*, does $u = v$ in M ?

- **Partial answer:** Fix an orientation R of rules in $R^{\text{n-o}}$. If (X, R) is convergent, this problem is decidable using the normal form algorithm.

Input : u, $v \in X^*$
Result: Boolean $u = v$ in M ?
Reduce u in $\hat{u}$;
Reduce v in $\hat{v}$;
if $\hat{u} = \hat{v}$ **then**
  True
else
  False
end

Examples

Example. $X = \{a\}$ and $R = \{aa \xrightarrow{\alpha} 1\}$.

Examples

Example. $X = \{a\}$ and $R = \{aa \xrightarrow{\alpha} 1\}$.

- Termination: the number of a is strictly decreasing.

Examples

Example. $X = \{a\}$ and $R = \{aa \xrightarrow{\alpha} 1\}$.

- Termination: the number of a is strictly decreasing.

- One confluent critical branching.

$$
\begin{tikzcd}
 & a \\
\underline{a\overline{a}}\overline{a} \arrow[ur, bend left, "a\alpha"] \arrow[dr, bend right, "\alpha a"'] & \\
 & a
\end{tikzcd}
$$

Examples

Example. $X = \{a\}$ and $R = \{aa \xrightarrow{\alpha} 1\}$.

- Termination: the number of a is strictly decreasing.

- One confluent critical branching.

$$\begin{tikzcd} & a \\ \underline{a\overline{a}}\overline{a} \arrow[ur, bend left, "a\alpha"] \arrow[dr, bend right, "\alpha a"'] & \\ & a \end{tikzcd}$$

Exemple. $X = \{s, t\}$ and $R = \{sts \xrightarrow{\alpha} tst\}$. $s =$  $t =$   $=$ 

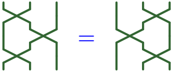
Examples

Example. $X = \{a\}$ and $R = \{aa \overset{\alpha}{\to} 1\}$.

- Termination: the number of a is strictly decreasing.

- One confluent critical branching.

$$\begin{tikzcd} & a \\ \underline{a\overline{a}}\overline{a} \arrow[ur, bend left, "a\alpha"] \arrow[dr, bend right, "\alpha a"'] & \\ & a \end{tikzcd}$$

Exemple. $X = \{s, t\}$ and $R = \{sts \overset{\alpha}{\to} tst\}$. $s =$  $t =$  

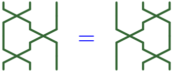
- Termination: lexicographic order on $s > t$.

Examples

Example. $X = \{a\}$ and $R = \{aa \xrightarrow{\alpha} 1\}$.

- Termination: the number of a is strictly decreasing.
- One confluent critical branching.

$$\begin{tikzcd}[row sep=small]
 & a \\
\underline{a\overline{a}}\overline{a} \arrow[ur, bend left, "a\alpha"] \arrow[dr, bend right, "\alpha a"'] & \\
 & a
\end{tikzcd}$$

Exemple. $X = \{s, t\}$ and $R = \{sts \xrightarrow{\alpha} tst\}$. $s =$  $t =$  

- Termination: lexicographic order on $s > t$.
- One non-confluent critical branching.

$$\begin{tikzcd}[row sep=small]
 & sttst \\
\underline{st\overline{s}}\overline{ts} \arrow[ur, bend left, "st\alpha"] \arrow[dr, bend right, "\alpha ts"'] & \\
 & tstts
\end{tikzcd}$$

Knuth-Bendix completion

Input : (X, R) terminating + termination order $>$

$\mathcal{KB}(R) := R$;

$\mathcal{C}_b :=$ { critical branchings } ;

while $\mathcal{C}_b \neq \emptyset$ **do**

Pick $(f : u \to v, g : u \to w)$ in $\mathcal{C}_b$;

$\mathcal{C}_b := \mathcal{C}_b \backslash \{(f, g)\}$;

Reduce v in $\hat{v}$ wrt R ;

Reduce w in $\hat{w}$ wrt R ;

if $\hat{v} \neq \hat{w}$ **then**

if $\hat{v} > \hat{w}$ **then**

$\mathcal{KB}(R) := \mathcal{KB}(R) \cup \{\alpha : \hat{v} \to \hat{w}\}$

else

$\mathcal{KB}(R) := \mathcal{KB}(R) \cup \{\alpha : \hat{w} \to \hat{v}\}$

end

else

end

$\mathcal{C}_b := \mathcal{C}_b \cup$ {critical branchings generated by α }

end

Knuth-Bendix completion

- This algorithm may not terminate.
- If it does, it returns $(X, \mathcal{KB}(R))$ which is convergent and presents the same monoid.

Example. $X = \{s, t\}$ and $R = \{sts \xrightarrow{\alpha} tst\}$ with lexicographic order on $s > t$,

Knuth-Bendix completion

- This algorithm may not terminate.
- If it does, it returns $(X, \mathcal{KB}(R))$ which is convergent and presents the same monoid.

Example. $X = \{s, t\}$ and $R = \{sts \overset{\alpha}{\to} tst\}$ with lexicographic order on $s > t$,

$$\begin{tikzcd}
 & sttst \\
ststs \arrow[ur, bend left, "st\alpha"] \arrow[dr, bend right, "\alpha ts"'] & \\
 & tstts
\end{tikzcd}$$

Knuth-Bendix completion

- This algorithm may not terminate.
- If it does, it returns $(X, \mathcal{KB}(R))$ which is convergent and presents the same monoid.

Example. $X = \{s, t\}$ and $R = \{sts \xrightarrow{\alpha} tst\}$ with lexicographic order on $s > t$,

$$\begin{tikzcd}
 & sttst \arrow[dd, bend left, "\beta"] \\
\underline{st\overline{s}}\overline{ts} \arrow[ur, bend left, "st\alpha"] \arrow[dr, bend right, "\alpha ts"'] & \\
 & tstts
\end{tikzcd}$$

Knuth-Bendix completion

- This algorithm may not terminate.
- If it does, it returns $(X, \mathcal{KB}(R))$ which is convergent and presents the same monoid.

Example. $X = \{s, t\}$ and $R = \{sts \overset{\alpha}{\to} tst\}$ with lexicographic order on $s > t$,

$$
\begin{tikzcd}
 & sttst \arrow[dd, bend left, "\beta"] & & \\
\underline{st\overline{s}}\overline{ts} \arrow[ur, bend left, "st\alpha"] \arrow[dr, bend right, "\alpha ts"'] & & & \underline{stt\overline{st}}\overline{s} \\
 & tstts & &
\end{tikzcd}
$$

Knuth-Bendix completion

- This algorithm may not terminate.
- If it does, it returns $(X, \mathcal{KB}(R))$ which is convergent and presents the same monoid.

Example. $X = \{s, t\}$ and $R = \{sts \xrightarrow{\alpha} tst\}$ with lexicographic order on $s > t$,

$$\begin{tikzcd} & sttst \arrow[dd, "\beta"] \\ ststs \arrow[ur, bend left, "st\alpha"] \arrow[dr, bend right, "\alpha ts"'] & \\ & tstts \end{tikzcd} \qquad \begin{tikzcd} & stttst \\ sttsts \arrow[ur, bend left, "stt\alpha"] \arrow[dr, bend right, "\beta s"'] & \\ & tsttss \end{tikzcd}$$

Knuth-Bendix completion

- This algorithm may not terminate.
- If it does, it returns $(X, \mathcal{KB}(R))$ which is convergent and presents the same monoid.

Example. $X = \{s, t\}$ and $R = \{sts \xrightarrow{\alpha} tst\}$ with lexicographic order on $s > t$,

$$\begin{tikzcd}
 & sttst \arrow[dd, bend left, "\beta"] \\
\underline{st\overline{s}}\overline{ts} \arrow[ur, "st\alpha"] \arrow[dr, "\alpha ts"'] & \\
 & tstts
\end{tikzcd}
\qquad
\begin{tikzcd}
 & stttst \arrow[dd, bend left, "\gamma"] \\
\underline{stt\overline{st}}\overline{s} \arrow[ur, "stt\alpha"] \arrow[dr, "\beta s"'] & \\
 & tsttss
\end{tikzcd}$$

Knuth-Bendix completion

- This algorithm may not terminate.
- If it does, it returns $(X, \mathcal{KB}(R))$ which is convergent and presents the same monoid.

Example. $X = \{s, t\}$ and $R = \{sts \xrightarrow{\alpha} tst\}$ with lexicographic order on $s > t$,

$$
\begin{tikzcd}
 & sttst \arrow[dd, bend left, "\beta"] \\
\underline{st\overline{s}}\overline{ts} \arrow[ur, bend left, "st\alpha"] \arrow[dr, bend right, "\alpha ts"'] & \\
 & tstts
\end{tikzcd}
\qquad
\begin{tikzcd}
 & stttst \arrow[dd, bend left, "\gamma"] \\
\underline{stt\overline{st}}\overline{s} \arrow[ur, bend left, "stt\alpha"] \arrow[dr, bend right, "\beta s"'] & \\
 & tsttss
\end{tikzcd}
\qquad
\underline{stt\overline{st}}\overline{tst}
$$

Knuth-Bendix completion

- This algorithm may not terminate.
- If it does, it returns $(X, \mathcal{KB}(R))$ which is convergent and presents the same monoid.

Example. $X = \{s, t\}$ and $R = \{sts \xrightarrow{\alpha} tst\}$ with lexicographic order on $s > t$,

$$\begin{tikzcd}[column sep=small]
 & sttst \arrow[dd, bend left, "\beta"] \\
ststs \arrow[ur, "st\alpha"] \arrow[dr, "\alpha ts"'] & \\
 & tstts
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=small]
 & stttst \arrow[dd, bend left, "\gamma"] \\
sttsts \arrow[ur, "stt\alpha"] \arrow[dr, "\beta s"'] & \\
 & tsttss
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=small]
 & stttstts \\
sttsttst \arrow[ur, "\beta tst"] \arrow[dr, "stt\beta"'] & \\
 & tsttstst
\end{tikzcd}$$

Knuth-Bendix completion

- This algorithm may not terminate.
- If it does, it returns $(X, \mathcal{KB}(R))$ which is convergent and presents the same monoid.

Example. $X = \{s, t\}$ and $R = \{sts \xrightarrow{\alpha} tst\}$ with lexicographic order on $s > t$,

$$
\begin{tikzcd}
 & sttst \arrow[dd, bend left, "\beta"] \\
\underline{st\overline{s}}\overline{ts} \arrow[ur, "st\alpha"] \arrow[dr, "\alpha ts"'] & \\
 & tstts
\end{tikzcd}
\qquad
\begin{tikzcd}
 & stttst \arrow[dd, bend left, "\gamma"] \\
\underline{stt\overline{st}}\overline{s} \arrow[ur, "stt\alpha"] \arrow[dr, "\beta s"'] & \\
 & tsttss
\end{tikzcd}
\qquad
\begin{tikzcd}
 & stttstts \arrow[dr, "\beta ts"] & \\
\underline{stt\overline{st}}\overline{tst} \arrow[ur, "\beta tst"] \arrow[dr, "stt\beta"'] & & tsttssts \arrow[dl, "tstts\alpha"] \\
 & tsttstst &
\end{tikzcd}
$$

Knuth-Bendix completion

- This algorithm may not terminate.
- If it does, it returns $(X, \mathcal{KB}(R))$ which is convergent and presents the same monoid.

  Example. $X = \{s, t\}$ and $R = \{sts \overset{\alpha}{\to} tst\}$ with lexicographic order on $s > t$,

$$
\begin{tikzcd}
 & sttst \arrow[dd, bend left, "\beta"] \\
\underline{st\overline{s}}\overline{ts} \arrow[ur, bend left, "st\alpha"] \arrow[dr, bend right, "\alpha ts"'] & \\
 & tstts
\end{tikzcd}
\qquad
\begin{tikzcd}
 & stttst \arrow[dd, bend left, "\gamma"] \\
\underline{stt\overline{st}}\overline{s} \arrow[ur, bend left, "stt\alpha"] \arrow[dr, bend right, "\beta s"'] & \\
 & tsttss
\end{tikzcd}
\qquad
\begin{tikzcd}
 & stttstts \arrow[dr, bend left, "\beta ts"] & \\
\underline{stt\overline{st}}\overline{tst} \arrow[ur, bend left, "\beta tst"] \arrow[dr, bend right, "stt\beta"'] & & tsttssts \arrow[dl, bend left, "tstts\alpha"] \\
 & tsttstst &
\end{tikzcd}
\qquad \cdots
$$

- Kapur & Narendran , '85: The monoid B_3^+ does not admit a finite convergent presentation with 2 generators.

Knuth-Bendix completion

- $X = \{s, t, a\}$ and $R = \{ta \stackrel{\alpha}{\to} as \ , \ st \stackrel{\beta}{\to} a\}$ presents the same monoid. It terminates for the lexicographic order on $s > t > a$.

Knuth-Bendix completion

- $X = \{s, t, a\}$ and $R = \{ta \overset{\alpha}{\to} as \; , \; st \overset{\beta}{\to} a\}$ presents the same monoid. It terminates for the lexicographic order on $s > t > a$.

$$\begin{tikzcd}
 & aa \\
sta \arrow[ur, bend left, "\beta a"] \arrow[dr, bend right, "s\alpha"'] & \\
 & sas
\end{tikzcd}$$

Knuth-Bendix completion

- $X = \{s, t, a\}$ and $R = \{ta \overset{\alpha}{\to} as\ ,\ st \overset{\beta}{\to} a\}$ presents the same monoid. It terminates for the lexicographic order on $s > t > a$.

$$\begin{tikzcd}
 & aa \\
sta \arrow[ur, bend left, "\beta a"] \arrow[dr, bend right, "s\alpha"'] & \\
 & sas \arrow[uu, bend right, "\gamma"']
\end{tikzcd}$$

Knuth-Bendix completion

- $X = \{s, t, a\}$ and $R = \{ta \xrightarrow{\alpha} as \ , \ st \xrightarrow{\beta} a\}$ presents the same monoid. It terminates for the lexicographic order on $s > t > a$.

$$
\begin{tikzcd}
 & aa \\
sta \arrow[ur, bend left, "\beta a"] \arrow[dr, bend right, "s\alpha"'] & \\
 & sas \arrow[uu, bend right, "\gamma"']
\end{tikzcd}
\qquad
\begin{tikzcd}
 & aat \\
sast \arrow[ur, bend left, "\gamma t"] \arrow[dr, bend right, "sa\beta"'] & \\
 & saa
\end{tikzcd}
$$

Knuth-Bendix completion

- $X = \{s, t, a\}$ and $R = \{ta \xrightarrow{\alpha} as \ , \ st \xrightarrow{\beta} a\}$ presents the same monoid. It terminates for the lexicographic order on $s > t > a$.

$$
\begin{tikzcd}
 & aa \\
sta \arrow[ur, bend left, "\beta a"] \arrow[dr, bend right, "s\alpha"'] & \\
 & sas \arrow[uu, bend right, "\gamma"']
\end{tikzcd}
\qquad
\begin{tikzcd}
 & aat \\
sast \arrow[ur, bend left, "\gamma t"] \arrow[dr, bend right, "sa\beta"'] & \\
 & saa \arrow[uu, bend right, "\delta"']
\end{tikzcd}
$$

Knuth-Bendix completion

- $X = \{s, t, a\}$ and $R = \{ta \xrightarrow{\alpha} as \ , \ st \xrightarrow{\beta} a\}$ presents the same monoid. It terminates for the lexicographic order on $s > t > a$.

$$\begin{tikzcd}[row sep=small]
 & aa \\
sta \arrow[ur, bend left, "\beta a"] \arrow[dr, bend right, "s\alpha"'] & \\
 & sas \arrow[uu, bend right, "\gamma"']
\end{tikzcd}
\qquad
\begin{tikzcd}[row sep=small]
 & aat \\
sast \arrow[ur, bend left, "\gamma t"] \arrow[dr, bend right, "sa\beta"'] & \\
 & saa \arrow[uu, bend right, "\delta"']
\end{tikzcd}
\qquad
\begin{tikzcd}[row sep=small]
 & aaas & \\
sasas \arrow[ur, bend left, "\gamma as"] \arrow[dr, bend right, "sa\gamma"'] & & aata \arrow[ul, bend right, "aa\alpha"'] \\
 & saaa \arrow[ur, bend right, "\delta a"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}[row sep=small]
 & aaaa & aaast \arrow[l, "aaa\beta"'] \\
sasaa \arrow[ur, bend left, "\gamma aa"] \arrow[dr, bend right, "sa\delta"'] & & \\
 & saaat \arrow[r, "\delta at"'] & aatat \arrow[uu, bend right, "aa\alpha t"']
\end{tikzcd}$$

Knuth-Bendix completion

- $X = \{s, t, a\}$ and $R = \{ta \xrightarrow{\alpha} as\ ,\ st \xrightarrow{\beta} a\}$ presents the same monoid. It terminates for the lexicographic order on $s > t > a$.

$$\begin{tikzcd}
 & aa \\
sta \arrow[ur, bend left, "\beta a"] \arrow[dr, bend right, "s\alpha"'] & \\
 & sas \arrow[uu, bend right, "\gamma"']
\end{tikzcd}
\qquad
\begin{tikzcd}
 & aat \\
sast \arrow[ur, bend left, "\gamma t"] \arrow[dr, bend right, "sa\beta"'] & \\
 & saa \arrow[uu, bend right, "\delta"']
\end{tikzcd}
\qquad
\begin{tikzcd}
 & aaas & \\
sasas \arrow[ur, bend left, "\gamma as"] \arrow[dr, bend right, "sa\gamma"'] & & aata \arrow[ul, bend right, "aa\alpha"'] \\
 & saaa \arrow[ur, bend right, "\delta a"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & aaaa & aaast \arrow[l, "aaa\beta"'] \\
sasaa \arrow[ur, bend left, "\gamma aa"] \arrow[dr, bend right, "sa\delta"'] & & \\
 & saaat \arrow[r, "\delta at"'] & aatat \arrow[uu, bend right, "aa\alpha t"']
\end{tikzcd}$$

- The string rewriting system $< s, t, a \mid ta \xrightarrow{\alpha} as\ ,\ st \xrightarrow{\beta} a\ ,\ sas \xrightarrow{\gamma} aa\ ,\ saa \xrightarrow{\delta} aat\ >$ is a convergent presentation of B_3^+.

III. Rewriting in linear 2-categories

Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.

Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.
- **Example**: For $n \in \mathbb{N}$, the nil Hecke algebra $\mathcal{NH}_n$ is presented by
  - generators ξ_i for $1 \leq i \leq n$ and ∂_i for $1 \leq i < n$;

Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.
- **Example**: For $n \in \mathbb{N}$, the nil Hecke algebra $\mathcal{NH}_n$ is presented by
  - generators ξ_i for $1 \leq i \leq n$ and ∂_i for $1 \leq i < n$;
  - relations:

$$\xi_i \xi_j = \xi_j \xi_i$$

$$\partial_i \xi_j = \xi_j \partial_i \quad \text{si } |i - j| > 1$$

$$\partial_i \partial_j = \partial_j \partial_i \quad \text{si } |i - j| > 1$$

$$\partial_i^2 = 0$$

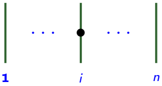
$$\partial_i \partial_{i+1} \partial_i = \partial_{i+1} \partial_i \partial_{i+1}$$

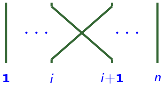
$$\xi_i \partial_i - \partial_i \xi_{i+1} = 1$$

$$\partial_i \xi_i - \xi_{i+1} \partial_i = 1$$

Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.
- **Example**: For $n \in \mathbb{N}$, the nil Hecke algebra $\mathcal{NH}_n$ is presented by
  - generators ξ_i for $1 \leq i \leq n$ and ∂_i for $1 \leq i < n$;

$\xi_i =$ 

, $\partial_i =$ 

  - relations:

$$\xi_i \xi_j = \xi_j \xi_i$$

$$\partial_i \xi_j = \xi_j \partial_i \quad \text{si } |i-j| > 1$$

$$\partial_i \partial_j = \partial_j \partial_i \quad \text{si } |i-j| > 1$$

$$\partial_i^2 = 0$$

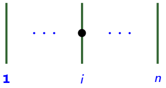
$$\partial_i \partial_{i+1} \partial_i = \partial_{i+1} \partial_i \partial_{i+1}$$

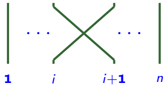
$$\xi_i \partial_i - \partial_i \xi_{i+1} = 1$$

$$\partial_i \xi_i - \xi_{i+1} \partial_i = 1$$

Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.
- **Example**: For $n \in \mathbb{N}$, the nil Hecke algebra $\mathcal{NH}_n$ is presented by
  - generators ξ_i for $1 \leq i \leq n$ and ∂_i for $1 \leq i < n$;

$\xi_i =$ 

, $\partial_i =$ 

  - relations:

$$\xi_i \xi_j = \xi_j \xi_i$$

$$\partial_i \xi_j = \xi_j \partial_i \quad \text{si } |i-j| > 1$$

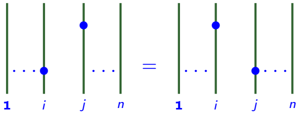
$$\partial_i \partial_j = \partial_j \partial_i \quad \text{si } |i-j| > 1$$

$$\partial_i^2 = 0$$

$$\partial_i \partial_{i+1} \partial_i = \partial_{i+1} \partial_i \partial_{i+1}$$

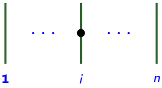
$$\xi_i \partial_i - \partial_i \xi_{i+1} = 1$$

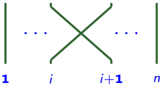
$$\partial_i \xi_i - \xi_{i+1} \partial_i = 1$$



Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.
- **Example**: For $n \in \mathbb{N}$, the nil Hecke algebra $\mathcal{NH}_n$ is presented by
  - generators ξ_i for $1 \leq i \leq n$ and ∂_i for $1 \leq i < n$;

$\xi_i =$ 

, $\partial_i =$ 

  - relations:

$$\xi_i \xi_j = \xi_j \xi_i$$

$$\partial_i \xi_j = \xi_j \partial_i \quad \text{si } |i-j| > 1$$

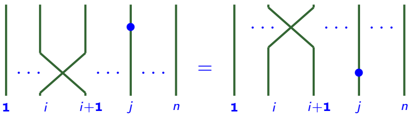
$$\partial_i \partial_j = \partial_j \partial_i \quad \text{si } |i-j| > 1$$

$$\partial_i^2 = 0$$

$$\partial_i \partial_{i+1} \partial_i = \partial_{i+1} \partial_i \partial_{i+1}$$

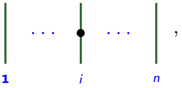
$$\xi_i \partial_i - \partial_i \xi_{i+1} = 1$$

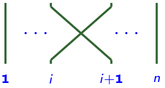
$$\partial_i \xi_i - \xi_{i+1} \partial_i = 1$$



Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.
- **Example**: For $n \in \mathbb{N}$, the nil Hecke algebra $\mathcal{NH}_n$ is presented by
  - generators ξ_i for $1 \leq i \leq n$ and ∂_i for $1 \leq i < n$;

$\xi_i =$ 

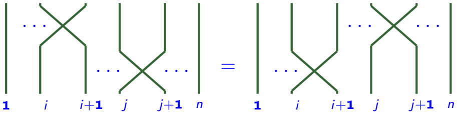
, $\partial_i =$ 

  - relations:

$$\xi_i\xi_j = \xi_j\xi_i$$

$$\partial_i\xi_j = \xi_j\partial_i \quad \text{si } |i-j| > 1$$

$$\partial_i\partial_j = \partial_j\partial_i \quad \text{si } |i-j| > 1$$



$$\partial_i^2 = 0$$

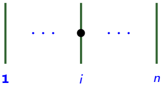
$$\partial_i\partial_{i+1}\partial_i = \partial_{i+1}\partial_i\partial_{i+1}$$

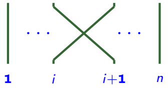
$$\xi_i\partial_i - \partial_i\xi_{i+1} = 1$$

$$\partial_i\xi_i - \xi_{i+1}\partial_i = 1$$

Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.
- **Example**: For $n \in \mathbb{N}$, the nil Hecke algebra $\mathcal{NH}_n$ is presented by
  - generators ξ_i for $1 \leq i \leq n$ and ∂_i for $1 \leq i < n$;

$\xi_i =$  ,

$\partial_i =$ 

  - relations:

$$\xi_i \xi_j = \xi_j \xi_i$$

$$\partial_i \xi_j = \xi_j \partial_i \quad \text{si } |i-j| > 1$$

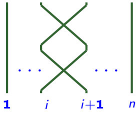
$$\partial_i \partial_j = \partial_j \partial_i \quad \text{si } |i-j| > 1$$

$$\partial_i^2 = 0$$

$$\partial_i \partial_{i+1} \partial_i = \partial_{i+1} \partial_i \partial_{i+1}$$

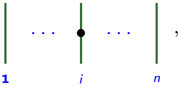
$$\xi_i \partial_i - \partial_i \xi_{i+1} = 1$$

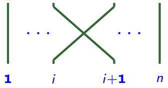
$$\partial_i \xi_i - \xi_{i+1} \partial_i = 1$$

 $= 0$

Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.
- **Example**: For $n \in \mathbb{N}$, the nil Hecke algebra $\mathcal{NH}_n$ is presented by
  - generators ξ_i for $1 \leq i \leq n$ and ∂_i for $1 \leq i < n$;

$\xi_i =$ 

, $\partial_i =$ 

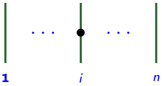
  - relations:

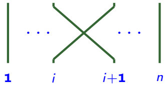
$$\xi_i \xi_j = \xi_j \xi_i$$
$$\partial_i \xi_j = \xi_j \partial_i \quad \text{si } |i-j| > 1$$
$$\partial_i \partial_j = \partial_j \partial_i \quad \text{si } |i-j| > 1$$
$$\partial_i^2 = 0$$
$$\partial_i \partial_{i+1} \partial_i = \partial_{i+1} \partial_i \partial_{i+1}$$
$$\xi_i \partial_i - \partial_i \xi_{i+1} = 1$$
$$\partial_i \xi_i - \xi_{i+1} \partial_i = 1$$

$= 0$

Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.
- **Example**: For $n \in \mathbb{N}$, the nil Hecke algebra $\mathcal{NH}_n$ is presented by
  - generators ξ_i for $1 \leq i \leq n$ and ∂_i for $1 \leq i < n$;

$\xi_i =$ 

, $\partial_i =$ 

  - relations:

$$\xi_i \xi_j = \xi_j \xi_i$$

$$\partial_i \xi_j = \xi_j \partial_i \quad \text{si } |i-j| > 1$$

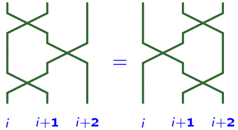
$$\partial_i \partial_j = \partial_j \partial_i \quad \text{si } |i-j| > 1$$

$$\partial_i^2 = 0$$

$$\partial_i \partial_{i+1} \partial_i = \partial_{i+1} \partial_i \partial_{i+1}$$

$$\xi_i \partial_i - \partial_i \xi_{i+1} = 1$$

$$\partial_i \xi_i - \xi_{i+1} \partial_i = 1$$

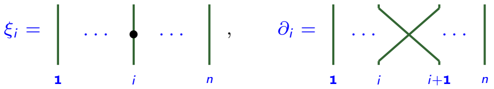


Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.

- **Example**: For $n \in \mathbb{N}$, the nil Hecke algebra $\mathcal{NH}_n$ is presented by

  - generators ξ_i for $1 \leq i \leq n$ and ∂_i for $1 \leq i < n$;



  - relations:

$$\xi_i \xi_j = \xi_j \xi_i$$

$$\partial_i \xi_j = \xi_j \partial_i \quad \text{si } |i-j| > 1$$

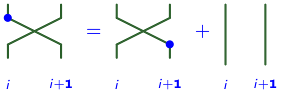
$$\partial_i \partial_j = \partial_j \partial_i \quad \text{si } |i-j| > 1$$

$$\partial_i^2 = 0$$

$$\partial_i \partial_{i+1} \partial_i = \partial_{i+1} \partial_i \partial_{i+1}$$

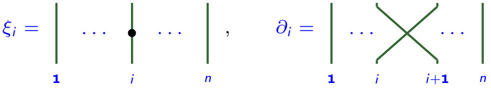
$$\xi_i \partial_i - \partial_i \xi_{i+1} = 1$$

$$\partial_i \xi_i - \xi_{i+1} \partial_i = 1$$



Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.
- **Example**: For $n \in \mathbb{N}$, the nil Hecke algebra $\mathcal{NH}_n$ is presented by
  - generators ξ_i for $1 \leq i \leq n$ and ∂_i for $1 \leq i < n$;



  - relations:

$$\xi_i\xi_j = \xi_j\xi_i$$

$$\partial_i\xi_j = \xi_j\partial_i \quad \text{si } |i-j| > 1$$

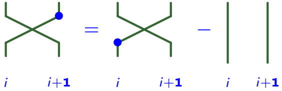
$$\partial_i\partial_j = \partial_j\partial_i \quad \text{si } |i-j| > 1$$

$$\partial_i^2 = 0$$

$$\partial_i\partial_{i+1}\partial_i = \partial_{i+1}\partial_i\partial_{i+1}$$

$$\xi_i\partial_i - \partial_i\xi_{i+1} = 1$$

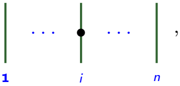
$$\partial_i\xi_i - \xi_{i+1}\partial_i = 1$$

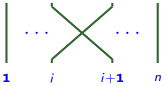


Diagrammatic algebras

- Diagrammatic algebra: algebra admitting a presentation by generators and relations depicted by diagrams.

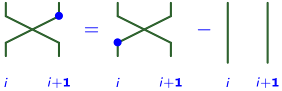
- **Example**: For $n \in \mathbb{N}$, the nil Hecke algebra $\mathcal{NH}_n$ is presented by
  - generators ξ_i for $1 \leq i \leq n$ and ∂_i for $1 \leq i < n$;

$\xi_i =$ 

, $\partial_i =$ 

  - relations:

$$\xi_i \xi_j = \xi_j \xi_i$$
$$\partial_i \xi_j = \xi_j \partial_i \quad \text{si } |i-j| > 1$$
$$\partial_i \partial_j = \partial_j \partial_i \quad \text{si } |i-j| > 1$$
$$\partial_i^2 = 0$$
$$\partial_i \partial_{i+1} \partial_i = \partial_{i+1} \partial_i \partial_{i+1}$$
$$\xi_i \partial_i - \partial_i \xi_{i+1} = 1$$
$$\partial_i \xi_i - \xi_{i+1} \partial_i = 1$$



- We realize these algebras as endomorphism spaces of a linear 2-category.

Linear monoidal categories and linear 2-categories

- A $\mathbb{K}$-linear strict monoidal category is a category $\mathcal{A}$ equipped with

Linear monoidal categories and linear 2-categories

- A $\mathbb{K}$-linear strict monoidal category is a category $\mathcal{A}$ equipped with
  - a tensor product $\otimes : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ which is associative.

Linear monoidal categories and linear 2-categories

- A $\mathbb{K}$-linear strict monoidal category is a category $\mathcal{A}$ equipped with
  - a tensor product $\otimes : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ which is associative.
  - a unit object $\mathbb{1}$ such that $1 \otimes A = A = A \otimes 1$ for all object of $\mathcal{A}$.

Linear monoidal categories and linear 2-categories

- A $\mathbb{K}$-linear strict monoidal category is a category $\mathcal{A}$ equipped with
  - a tensor product $\otimes : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ which is associative.
  - a unit object $\mathbb{1}$ such that $1 \otimes A = A = A \otimes 1$ for all object of $\mathcal{A}$.
  - for any object A,B of $\mathcal{A}$, $\mathcal{A}(A, B)$ is a $\mathbb{K}$-vector space.

Linear monoidal categories and linear 2-categories

- A $\mathbb{K}$-linear strict monoidal category is a category $\mathcal{A}$ equipped with
  - a tensor product $\otimes : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ which is associative.
  - a unit object $\mathbb{1}$ such that $1 \otimes A = A = A \otimes 1$ for all object of $\mathcal{A}$.
  - for any object A,B of $\mathcal{A}$, $\mathcal{A}(A, B)$ is a $\mathbb{K}$-vector space.
  - composition and tensor products of morphisms are $\mathbb{K}$-bilinear.

Linear monoidal categories and linear 2-categories

- A $\mathbb{K}$-linear strict monoidal category is a category $\mathcal{A}$ equipped with
  - a tensor product $\otimes : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ which is associative.
  - a unit object $\mathbb{1}$ such that $1 \otimes A = A = A \otimes 1$ for all object of $\mathcal{A}$.
  - for any object A,B of $\mathcal{A}$, $\mathcal{A}(A, B)$ is a $\mathbb{K}$-vector space.
  - composition and tensor products of morphisms are $\mathbb{K}$-bilinear.

- A $\mathbb{K}$-linear 2-category is the data of a 2-category $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2)$ such that:
  - for all p, q in $\mathcal{C}_\mathbf{1}$, $\mathcal{C}_\mathbf{2}(p, q)$ is a $\mathbb{K}$-vector space.
  - $\star_\mathbf{0}$ and $\star_\mathbf{1}$-composition of 1-cells are $\mathbb{K}$-bilinear.

Linear monoidal categories and linear 2-categories

- A $\mathbb{K}$-linear strict monoidal category is a category $\mathcal{A}$ equipped with
  - a tensor product $\otimes : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ which is associative.
  - a unit object $\mathbb{1}$ such that $1 \otimes A = A = A \otimes 1$ for all object of $\mathcal{A}$.
  - for any object A,B of $\mathcal{A}$, $\mathcal{A}(A, B)$ is a $\mathbb{K}$-vector space.
  - composition and tensor products of morphisms are $\mathbb{K}$-bilinear.
- A $\mathbb{K}$-linear 2-category is the data of a 2-category $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2)$ such that:
  - for all p, q in $\mathcal{C}_\mathbf{1}$, $\mathcal{C}_\mathbf{2}(p, q)$ is a $\mathbb{K}$-vector space.
  - $\star_\mathbf{0}$ and $\star_\mathbf{1}$-composition of 1-cells are $\mathbb{K}$-bilinear.
- When $\mathcal{C}_0 = \{*\}$, these two objects are the same.

Linear monoidal categories and linear 2-categories

- A $\mathbb{K}$-linear strict monoidal category is a category $\mathcal{A}$ equipped with
  - a tensor product $\otimes : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ which is associative.
  - a unit object $\mathbb{1}$ such that $1 \otimes A = A = A \otimes 1$ for all object of $\mathcal{A}$.
  - for any object A,B of $\mathcal{A}$, $\mathcal{A}(A, B)$ is a $\mathbb{K}$-vector space.
  - composition and tensor products of morphisms are $\mathbb{K}$-bilinear.
- A $\mathbb{K}$-linear 2-category is the data of a 2-category $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2)$ such that:
  - for all p, q in $\mathcal{C}_1$, $\mathcal{C}_2(p, q)$ is a $\mathbb{K}$-vector space.
  - $\star_0$ and $\star_1$-composition of 1-cells are $\mathbb{K}$-bilinear.
- When $\mathcal{C}_0 = \{*\}$, these two objects are the same.

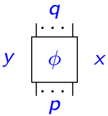
$$\text{objects of } \mathcal{A} \leftrightarrow 1\text{-cells of } \mathcal{C}$$

$$\text{morphisms of } \mathcal{A} \leftrightarrow 2\text{-cells of } \mathcal{C}$$

$$\otimes \leftrightarrow \star_0, \qquad \text{composition of morphisms} \leftrightarrow \star_1$$

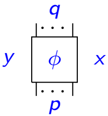
String diagrams

- A 2-cell $\phi : p \Rightarrow q$ with $p, q : x \to y$ in a linear 2-category $\mathcal{C}$ can be depicted by a string diagram:

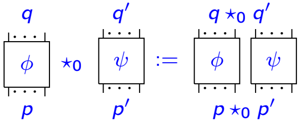


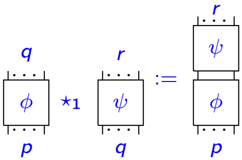
String diagrams

- A 2-cell $\phi : p \Rightarrow q$ with $p, q : x \to y$ in a linear 2-category $\mathcal{C}$ can be depicted by a string diagram:



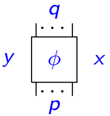
- Compositions:



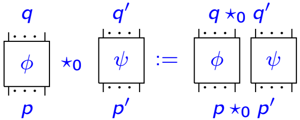


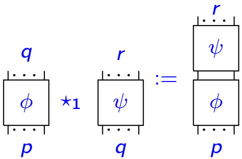
String diagrams

- A 2-cell $\phi : p \Rightarrow q$ with $p,q : x \to y$ in a linear 2-category $\mathcal{C}$ can be depicted by a string diagram:

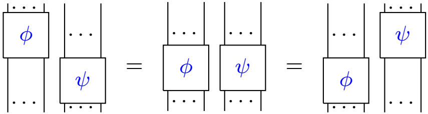


- Compositions:





- These compositions satisfy the exchange law:



Presentations by linear $(3, 2)$-polygraphs

- Polygraphs (Burroni - Street) are presentations by generators and relations of higher-dimensional globular strict categories.
  - Linear 2-categories are presented by rewriting systems called linear $(3, 2)$-polygraphs.

Presentations by linear $(3,2)$-polygraphs

- Polygraphs (Burroni - Street) are presentations by generators and relations of higher-dimensional globular strict categories.
  - Linear 2-categories are presented by rewriting systems called linear $(3,2)$-polygraphs.

- A 1-polygraph is a directed graph (P_1, P_0, s_0, t_0), on which we construct the free 1-category P_1^*.

Presentations by linear $(3,2)$-polygraphs

- Polygraphs (Burroni - Street) are presentations by generators and relations of higher-dimensional globular strict categories.
  - Linear 2-categories are presented by rewriting systems called linear $(3,2)$-polygraphs.

- A 1-polygraph is a directed graph (P_1, P_0, s_0, t_0), on which we construct the free 1-category P_1^*.

- $P_0 = \{*\}, P_1 = \{1\}, \star_0 = +, P_1^* = \mathbb{N}$,

Presentations by linear $(3,2)$-polygraphs

- Polygraphs (Burroni - Street) are presentations by generators and relations of higher-dimensional globular strict categories.
  - Linear 2-categories are presented by rewriting systems called linear $(3,2)$-polygraphs.

- A 1-polygraph is a directed graph (P_1, P_0, s_0, t_0), on which we construct the free 1-category P_1^*.

- We consider a cellular extension P_2 of P_1^*, that is a set equipped with s_1, t_1: $P_2 \to P_1^*$.

- $P_0 = \{*\}, P_1 = \{1\}, \star_0 = +, P_1^* = \mathbb{N}$,

Presentations by linear $(3,2)$-polygraphs

- Polygraphs (Burroni - Street) are presentations by generators and relations of higher-dimensional globular strict categories.
  - Linear 2-categories are presented by rewriting systems called linear $(3,2)$-polygraphs.

- A 1-polygraph is a directed graph (P_1, P_0, s_0, t_0), on which we construct the free 1-category P_1^*.

- We consider a cellular extension P_2 of P_1^*, that is a set equipped with s_1, t_1: $P_2 \to P_1^*$.

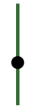
- $P_0 = \{*\}, P_1 = \{1\}, \star_0 = +, P_1^* = \mathbb{N}$,

- $P_2 = \{$  $: 2 \to 2,$  $: 1 \to 1\}$

Presentations by linear $(3,2)$-polygraphs

- Polygraphs (Burroni - Street) are presentations by generators and relations of higher-dimensional globular strict categories.
    - Linear 2-categories are presented by rewriting systems called linear $(3,2)$-polygraphs.

- A 1-polygraph is a directed graph (P_1, P_0, s_0, t_0), on which we construct the free 1-category P_1^*.
- We consider a cellular extension P_2 of P_1^*, that is a set equipped with s_1, t_1: $P_2 \to P_1^*$.
- We construct the free 2-category P_2^* on P_2.

- $P_0 = \{*\}, P_1 = \{1\}, \star_0 = +, P_1^* = \mathbb{N}$,
- $P_2 = \{$  $: 2 \to 2,$  $: 1 \to 1\}$

Presentations by linear $(3,2)$-polygraphs

- Polygraphs (Burroni - Street) are presentations by generators and relations of higher-dimensional globular strict categories.
  - Linear 2-categories are presented by rewriting systems called linear $(3,2)$-polygraphs.

- A 1-polygraph is a directed graph (P_1, P_0, s_0, t_0), on which we construct the free 1-category P_1^*.
- $P_0 = \{*\}, P_1 = \{1\}, \star_0 = +, P_1^* = \mathbb{N}$,
- We consider a cellular extension P_2 of P_1^*, that is a set equipped with s_1, t_1: $P_2 \to P_1^*$.
- $P_2 = \{$  $: 2 \to 2,$  $: 1 \to 1\}$
- We construct the free 2-category P_2^* on P_2.
- $P_2^* = \{$ diagrams formed by horizontal and vertical compositions of crossings and dots$\}$

Presentations by linear $(3, 2)$-polygraphs

- Polygraphs (Burroni - Street) are presentations by generators and relations of higher-dimensional globular strict categories.
  - Linear 2-categories are presented by rewriting systems called linear $(3, 2)$-polygraphs.

- A 1-polygraph is a directed graph (P_1, P_0, s_0, t_0), on which we construct the free 1-category P_1^*.
- We consider a cellular extension P_2 of P_1^*, that is a set equipped with s_1, t_1: $P_2 \to P_1^*$.
- We construct the free 2-category P_2^* on P_2.
- We construct the free linear 2-category P_2^ℓ on P_2:

$$P_2^\ell(x, y) = \mathbb{K}[P_2^*(x, y)]$$

for any 1-cells x and y in P_2^*.

- $P_0 = \{*\}$, $P_1 = \{1\}$, $\star_0 = +$, $P_1^* = \mathbb{N}$,
- $P_2 = \{$  $: 2 \to 2,$  $: 1 \to 1\}$
- $P_2^* = \{$ diagrams formed by horizontal and vertical compositions of crossings and dots$\}$

Presentations by linear $(3, 2)$-polygraphs

- Polygraphs (Burroni - Street) are presentations by generators and relations of higher-dimensional globular strict categories.
  - Linear 2-categories are presented by rewriting systems called linear $(3, 2)$-polygraphs.

- A 1-polygraph is a directed graph (P_1, P_0, s_0, t_0), on which we construct the free 1-category P_1^*.
- We consider a cellular extension P_2 of P_1^*, that is a set equipped with s_1, t_1: $P_2 \to P_1^*$.
- We construct the free 2-category P_2^* on P_2.
- We construct the free linear 2-category P_2^ℓ on P_2:

$$P_2^\ell(x, y) = \mathbb{K}[P_2^*(x, y)]$$

for any 1-cells x and y in P_2^*.

- $P_0 = \{*\}$, $P_1 = \{1\}$, $\star_0 = +$, $P_1^* = \mathbb{N}$,
- $P_2 = \{$  $: 2 \to 2,$  $: 1 \to 1\}$
- $P_2^* = \{$ diagrams formed by horizontal and vertical compositions of crossings and dots$\}$
- $P_2^\ell = \{\mathbb{K} -$ linear combinations of diagrams in $P_2^*\}$

Presentations by linear $(3,2)$-polygraphs

- Polygraphs (Burroni - Street) are presentations by generators and relations of higher-dimensional globular strict categories.
  - Linear 2-categories are presented by rewriting systems called linear $(3,2)$-polygraphs.

- A 1-polygraph is a directed graph (P_1, P_0, s_0, t_0), on which we construct the free 1-category P_1^*.
- We consider a cellular extension P_2 of P_1^*, that is a set equipped with s_1, t_1: $P_2 \to P_1^*$.
- We construct the free 2-category P_2^* on P_2.
- We construct the free linear 2-category P_2^ℓ on P_2:

$$P_2^\ell(x,y) = \mathbb{K}[P_2^*(x,y)]$$

  for any 1-cells x and y in P_2^*.
- We consider a cellular extension P_3 of P_2^ℓ, corresponding to an orientation of the relations.

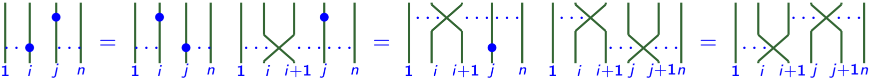
- $P_0 = \{*\}$, $P_1 = \{1\}$, $\star_0 = +$, $P_1^* = \mathbb{N}$,
- $P_2 = \{$  $: 2 \to 2,$  $: 1 \to 1\}$
- $P_2^* = \{$ diagrams formed by horizontal and vertical compositions of crossings and dots$\}$
- $P_2^\ell = \{\mathbb{K} -$ linear combinations of diagrams in $P_2^*\}$

Presentations by linear $(3, 2)$-polygraphs

- **Example** : for the nil Hecke algebras,

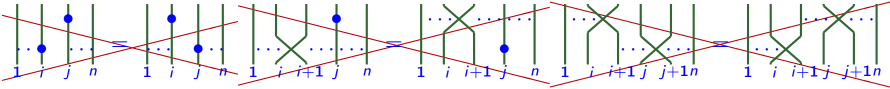
Presentations by linear $(3, 2)$-polygraphs

- **Example** : for the nil Hecke algebras,



Presentations by linear $(3, 2)$-polygraphs

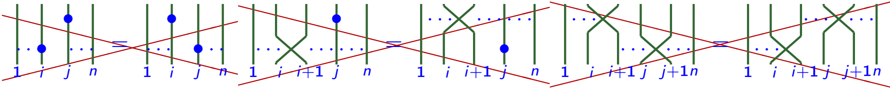
- **Example** : for the nil Hecke algebras,



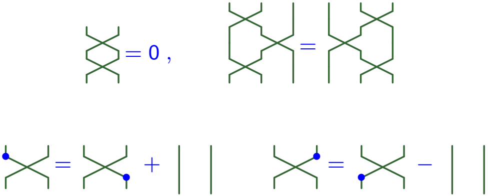
  - These are exchange laws.

Presentations by linear $(3,2)$-polygraphs

- **Example** : for the nil Hecke algebras,

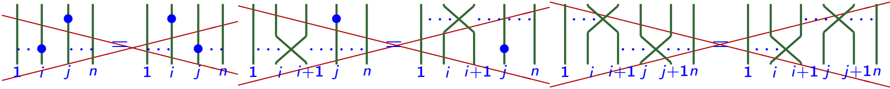


  - These are exchange laws.

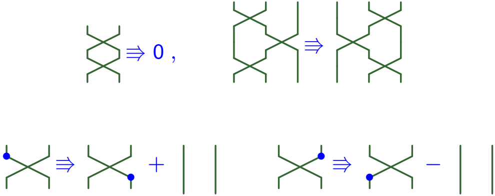


Presentations by linear $(3,2)$-polygraphs

- **Example** : for the nil Hecke algebras,

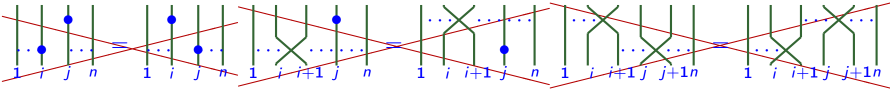


  - These are exchange laws.

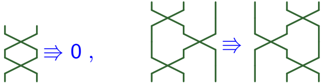


Presentations by linear $(3, 2)$-polygraphs

- **Example** : for the nil Hecke algebras,



  - These are exchange laws.

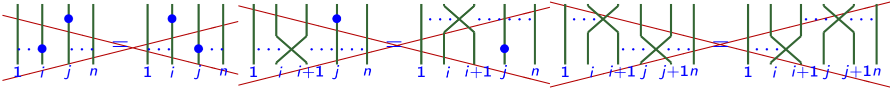


- This choice of cellular extension defines a linear $(3, 2)$-polygraph presenting a linear 2-category encoding the nil Hecke algebras.

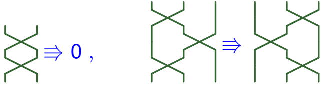
$$\mathrm{End}_{\mathcal{C}}(n) \simeq \mathcal{NH}_n$$

Presentations by linear $(3,2)$-polygraphs

- **Example** : for the nil Hecke algebras,



  - These are exchange laws.



- This choice of cellular extension defines a linear $(3,2)$-polygraph presenting a linear 2-category encoding the nil Hecke algebras.

$$\mathrm{End}_{\mathcal{C}}(n) \simeq \mathcal{NH}_n$$

- It is left-monomial, that is each source of a 3-cell is a monomial.

Linear rewriting

- Restriction of the set of rewritings due to the linear context:

.

Linear rewriting

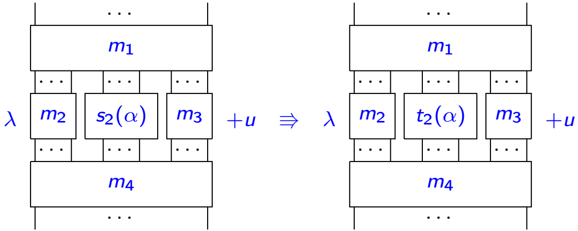
- Restriction of the set of rewritings due to the linear context: if $u \to v$, then $-u \Rightarrow -v$, and so $v = (u + v) - u \Rightarrow u + v - v = u$.

Linear rewriting

- Restriction of the set of rewritings due to the linear context: if $u \to v$, then $-u \Rightarrow -v$, and so $v = (u+v) - u \cancel{\Rightarrow} u + v - v = u$.

Linear rewriting

- Restriction of the set of rewritings due to the linear context: if $u \to v$, then $-u \Rrightarrow -v$, and so $v = (u+v) - u \cancel{\Rrightarrow} u + v - v = u$.
- A rewriting step of a linear $(3,2)$-polygraph is a 3-cell of the form

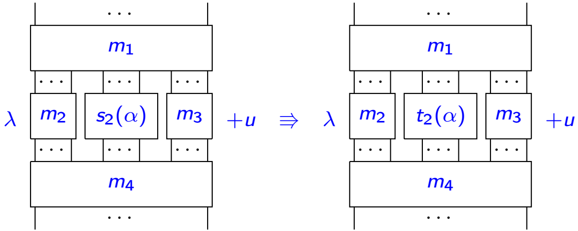


where $\alpha \in P_3$, and the monomial $m_1 \star_1 (m_2 \star_0 s_2(\alpha) \star_0 m_3) \star_1 m_4$ does not appear in the monomial decomposition of u.

Linear rewriting

- Restriction of the set of rewritings due to the linear context: if $u \to v$, then $-u \Rrightarrow -v$, and so $v = (u+v) - u \cancel{\Rrightarrow} u + v - v = u$.

- A rewriting step of a linear $(3,2)$-polygraph is a 3-cell of the form



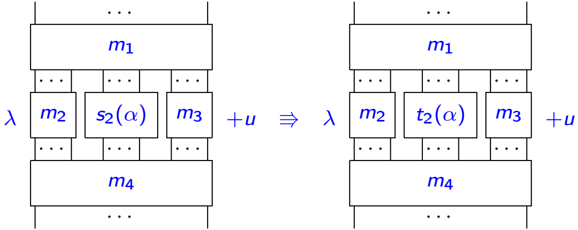
  where $\alpha \in P_3$, and the monomial $m_1 \star_1 (m_2 \star_0 s_2(\alpha) \star_0 m_3) \star_1 m_4$ does not appear in the monomial decomposition of u.

- **Newman lemma**: A terminating linear $(3,2)$-polygraph is confluent if and only if it is locally confluent.

Linear rewriting

- Restriction of the set of rewritings due to the linear context: if $u \to v$, then $-u \Rightarrow -v$, and so $v = (u+v) - u \cancel{\Rightarrow} u + v - v = u$.

- A rewriting step of a linear $(3,2)$-polygraph is a 3-cell of the form



  where $\alpha \in P_3$, and the monomial $m_1 \star_1 (m_2 \star_0 s_2(\alpha) \star_0 m_3) \star_1 m_4$ does not appear in the monomial decomposition of u.

- **Newman lemma**: A terminating linear $(3,2)$-polygraph is confluent if and only if it is locally confluent.

- **Critical pair lemma**: A **terminating** linear $(3,2)$-polygraph is locally confluent if and only if its critical branchings are confluent.

Critical pair lemma fails without termination

- Consider a linear rewriting system on generators x,y, z and rules $\alpha : xy \to xz$ and $\beta : zt \to 2yt$.

Critical pair lemma fails without termination

- Consider a linear rewriting system on generators x,y, z and rules $\alpha : xy \to xz$ and $\beta : zt \to 2yt$.
- It has no critical branching.

Critical pair lemma fails without termination

- Consider a linear rewriting system on generators x,y, z and rules $\alpha : xy \to xz$ and $\beta : zt \to 2yt$.
- It has no critical branching.
- Consider the Peiffer branching

$$
\begin{tikzcd}
 & 2xzt \\
xyt + xzt \arrow[ur, bend left, "\alpha t + xzt"] \arrow[dr, bend right, "xyt + x\beta"'] & \\
 & 3xyt
\end{tikzcd}
$$

Critical pair lemma fails without termination

- Consider a linear rewriting system on generators x,y, z and rules $\alpha : xy \to xz$ and $\beta : zt \to 2yt$.
- It has no critical branching.
- Consider the Peiffer branching

$$\begin{tikzcd}
 & 2xzt \arrow[dr, dotted, bend left, "xzt + x\beta"] & \\
xyt + xzt \arrow[ur, bend left, "\alpha t + xzt"] \arrow[dr, bend right, "xyt + x\beta"'] & & xzt + 2xyt \\
 & 3xyt \arrow[ur, dotted, bend right, "\alpha t + 2xyt"'] &
\end{tikzcd}$$

Critical pair lemma fails without termination

- Consider a linear rewriting system on generators x,y, z and rules $\alpha : xy \to xz$ and $\beta : zt \to 2yt$.
- It has no critical branching.
- Consider the Peiffer branching

$$
\begin{tikzcd}
 & & 4xyt \\
 & 2xzt \arrow[ur, bend left, "2x\beta"] \arrow[dr, dotted, "xzt + x\beta"] & \\
xyt + xzt \arrow[ur, bend left, "\alpha t + xzt"] \arrow[dr, bend right, "xyt + x\beta"'] & & xzt + 2xyt \\
 & 3xyt \arrow[ur, dotted, "\alpha t + 2xyt"'] &
\end{tikzcd}
$$

Critical pair lemma fails without termination

- Consider a linear rewriting system on generators x,y, z and rules $\alpha : xy \to xz$ and $\beta : zt \to 2yt$.

- It has no critical branching.

- Consider the Peiffer branching

$$
\begin{tikzcd}
 & & 4xyt \arrow[r, "4\alpha t"] & 4xzt \arrow[r, "4x\beta"] & \cdots \\
 & 2xzt \arrow[ur, bend left, "2x\beta"] \arrow[dr, dotted, bend left, "xzt + x\beta"] & & & \\
xyt + xzt \arrow[ur, bend left, "\alpha t + xzt"] \arrow[dr, bend right, "xyt + x\beta"'] & & xzt + 2xyt & & \\
 & 3xyt \arrow[ur, dotted, bend right, "\alpha t + 2xyt"'] & & &
\end{tikzcd}
$$

Critical pair lemma fails without termination

- Consider a linear rewriting system on generators x,y, z and rules $\alpha : xy \to xz$ and $\beta : zt \to 2yt$.
- It has no critical branching.
- Consider the Peiffer branching

$$
\begin{tikzcd}
 & & 4xyt \arrow[r, "4\alpha t"] & 4xzt \arrow[r, "4x\beta"] & \cdots \\
 & 2xzt \arrow[ur, bend left, "2x\beta"] \arrow[dr, dotted, "xzt + x\beta"] & & & \\
xyt + xzt \arrow[ur, bend left, "\alpha t + xzt"] \arrow[dr, bend right, "xyt + x\beta"'] & & xzt + 2xyt & & \\
 & 3xyt \arrow[ur, dotted, "\alpha t + 2xyt"'] \arrow[dr, bend right, "3\alpha t"'] & & & \\
 & & 3xzt \arrow[r, "3x\beta"'] & 6xyt \arrow[r, "6\alpha t"'] & \cdots
\end{tikzcd}
$$

Critical branchings of linear $(3, 2)$-polygraphs

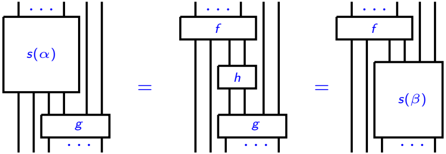
- A critical branching is a branching on a minimal string diagram.

Critical branchings of linear $(3,2)$-polygraphs

- A critical branching is a branching on a minimal string diagram.
- There are 3 different forms of critical branchings:

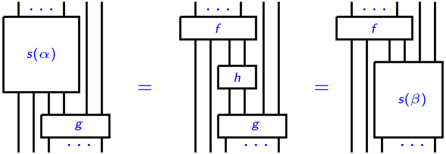
Critical branchings of linear $(3, 2)$-polygraphs

- A critical branching is a branching on a minimal string diagram.
- There are 3 different forms of critical branchings:
  - Regular critical branchings:

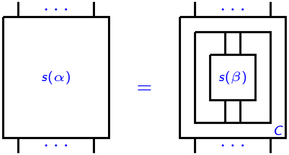


Critical branchings of linear $(3, 2)$-polygraphs

- A critical branching is a branching on a minimal string diagram.
- There are 3 different forms of critical branchings:
  - Regular critical branchings:

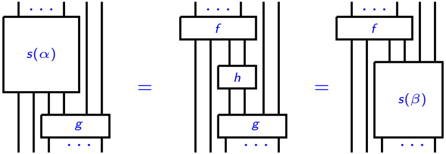


  - Inclusion critical branchings:

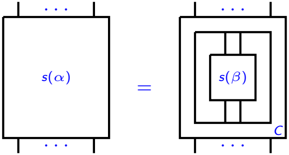


Critical branchings of linear $(3, 2)$-polygraphs

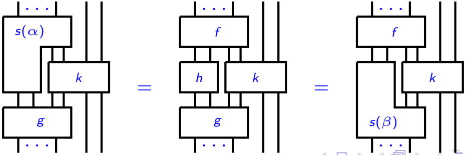
- A critical branching is a branching on a minimal string diagram.
- There are 3 different forms of critical branchings:
  - Regular critical branchings:



  - Inclusion critical branchings:



  - Right-indexed (also left-indexed, multi-indexed) critical branchings:



Linear bases from convergence

- P a convergent left-monomial linear $(3, 2)$-polygraph.

Linear bases from convergence

- P a convergent left-monomial linear $(3,2)$-polygraph.
- $\mathcal{C}$ the linear 2-category it presents.

Linear bases from convergence

- P a convergent left-monomial linear $(3,2)$-polygraph.
- $\mathcal{C}$ the linear 2-category it presents.
- **Theorem (Alleaume):** For any parallel 1-cells p and q of $\mathcal{C}$, the set of monomials in normal form for P with 1-source p and 1-target q is a linear basis of $\mathcal{C}_2(p,q)$.

Linear bases from convergence

- P a convergent left-monomial linear $(3, 2)$-polygraph.
- $\mathcal{C}$ the linear 2-category it presents.
- **Theorem (Alleaume):** For any parallel 1-cells p and q of $\mathcal{C}$, the set of monomials in normal form for P with 1-source p and 1-target q is a linear basis of $\mathcal{C}_2(p, q)$.
  - Termination: the monomials in normal form span $\mathcal{C}_{\mathbf{2}}(p, q)$.

Linear bases from convergence

- P a convergent left-monomial linear $(3,2)$-polygraph.
- $\mathcal{C}$ the linear 2-category it presents.
- **Theorem (Alleaume):** For any parallel 1-cells p and q of $\mathcal{C}$, the set of monomials in normal form for P with 1-source p and 1-target q is a linear basis of $\mathcal{C}_2(p,q)$.
  - Termination: the monomials in normal form span $\mathcal{C}_{\mathbf{2}}(p,q)$.
  - Confluence: if a 2-cell reduces into two different linear combinations of monomials in normal form, they are equal by confluence and since P is left-monomial.

Example: the Khovanov-Lauda-Rouquier (KLR) algebras

- These algebras have been defined in the process of categorifying a quantum group $\mathbf{U}_q(\mathfrak{g})$ associated with a symmetrizable Kac-Moody algebra $\mathfrak{g}$.

Example: the Khovanov-Lauda-Rouquier (KLR) algebras

- These algebras have been defined in the process of categorifying a quantum group $\mathbf{U}_q(\mathfrak{g})$ associated with a symmetrizable Kac-Moody algebra $\mathfrak{g}$.
- Let Γ be the Dynkin graph of $\mathfrak{g}$, and I its set of vertices. Fix:

Example: the Khovanov-Lauda-Rouquier (KLR) algebras

- These algebras have been defined in the process of categorifying a quantum group $\mathbf{U}_q(\mathfrak{g})$ associated with a symmetrizable Kac-Moody algebra $\mathfrak{g}$.
- Let Γ be the Dynkin graph of $\mathfrak{g}$, and I its set of vertices. Fix:
  - an element $\mathcal{V} = \sum_{i \in I} \nu_i . i \in \mathbb{N}[I]$,

Example: the Khovanov-Lauda-Rouquier (KLR) algebras

- These algebras have been defined in the process of categorifying a quantum group $\mathbf{U}_q(\mathfrak{g})$ associated with a symmetrizable Kac-Moody algebra $\mathfrak{g}$.
- Let Γ be the Dynkin graph of $\mathfrak{g}$, and I its set of vertices. Fix:
  - an element $\mathcal{V} = \sum_{i \in I} \nu_i . i \in \mathbb{N}[I]$, $\rightsquigarrow$ algebra $R(\mathcal{V})$

Example: the Khovanov-Lauda-Rouquier (KLR) algebras

- These algebras have been defined in the process of categorifying a quantum group $\mathbf{U}_q(\mathfrak{g})$ associated with a symmetrizable Kac-Moody algebra $\mathfrak{g}$.
- Let Γ be the Dynkin graph of $\mathfrak{g}$, and I its set of vertices. Fix:
  - an element $\mathcal{V} = \sum_{i \in I} \nu_i . i \in \mathbb{N}[I]$, $\rightsquigarrow$ algebra $R(\mathcal{V})$
  - a bilinear form $\cdot$ on $\mathbb{Z}[I]$ with values in $\{0, 1\}$,

Example: the Khovanov-Lauda-Rouquier (KLR) algebras

- These algebras have been defined in the process of categorifying a quantum group $\mathbf{U}_q(\mathfrak{g})$ associated with a symmetrizable Kac-Moody algebra $\mathfrak{g}$.
- Let Γ be the Dynkin graph of $\mathfrak{g}$, and I its set of vertices. Fix:
  - an element $\mathcal{V} = \sum_{i \in I} \nu_i . i \in \mathbb{N}[I]$, $\rightsquigarrow$ algebra $R(\mathcal{V})$
  - a bilinear form $\cdot$ on $\mathbb{Z}[I]$ with values in $\{0, 1\}$,

Example: the Khovanov-Lauda-Rouquier (KLR) algebras

- These algebras have been defined in the process of categorifying a quantum group $\mathbf{U}_q(\mathfrak{g})$ associated with a symmetrizable Kac-Moody algebra $\mathfrak{g}$.
- Let Γ be the Dynkin graph of $\mathfrak{g}$, and I its set of vertices. Fix:
  - an element $\mathcal{V} = \sum_{i \in I} \nu_i . i \in \mathbb{N}[I]$, $\rightsquigarrow$ algebra $R(\mathcal{V})$
  - a bilinear form $\cdot$ on $\mathbb{Z}[I]$ with values in $\{0, 1\}$,
  - the set $\mathrm{Seq}(\mathcal{V})$ of sequences of length m of elements of Γ, where i appears $\mathcal{V}_i$ times.

Example: the Khovanov-Lauda-Rouquier (KLR) algebras

- These algebras have been defined in the process of categorifying a quantum group $\mathbf{U}_q(\mathfrak{g})$ associated with a symmetrizable Kac-Moody algebra $\mathfrak{g}$.
- Let Γ be the Dynkin graph of $\mathfrak{g}$, and I its set of vertices. Fix:
  - an element $\mathcal{V} = \sum_{i \in I} \nu_i . i \in \mathbb{N}[I]$, $\rightsquigarrow$ algebra $R(\mathcal{V})$
  - a bilinear form $\cdot$ on $\mathbb{Z}[I]$ with values in $\{0, 1\}$,
  - the set $\mathrm{Seq}(\mathcal{V})$ of sequences of length m of elements of Γ, where i appears $\mathcal{V}_i$ times.
  - **Example:** $\mathrm{Seq}(2i + j) = \{iij, iji, jii\}$

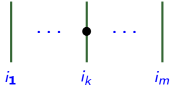
Example: the Khovanov-Lauda-Rouquier (KLR) algebras

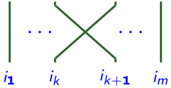
- These algebras have been defined in the process of categorifying a quantum group $\mathbf{U}_q(\mathfrak{g})$ associated with a symmetrizable Kac-Moody algebra $\mathfrak{g}$.
- Let Γ be the Dynkin graph of $\mathfrak{g}$, and I its set of vertices. Fix:
  - an element $\mathcal{V} = \sum_{i \in I} \nu_i . i \in \mathbb{N}[I]$, $\rightsquigarrow$ algebra $R(\mathcal{V})$
  - a bilinear form $\cdot$ on $\mathbb{Z}[I]$ with values in $\{0, 1\}$,
  - the set $\mathrm{Seq}(\mathcal{V})$ of sequences of length m of elements of Γ, where i appears $\mathcal{V}_i$ times.
  - **Example:** $\mathrm{Seq}(2i + j) = \{iij, iji, jii\}$
- **Theorem [Khovanov-Lauda '08]:** If $R = \bigoplus_{\mathcal{V} \in \mathbb{N}[I]} R(\mathcal{V})$,

$$K_0(R - \mathrm{pmod}) \simeq \mathbf{U}_q^-(\mathfrak{g})$$

Presentation of the KLR algebras

- For $\mathbf{i} = i_1 \dots i_m \in \mathrm{Seq}(\mathcal{V})$, generators

$x_{k,\mathbf{i}} =$ 

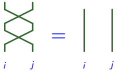
and $\tau_{k,\mathbf{i}} =$ 

- Relations:

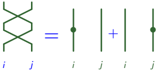
i) For $i \in I$,

 $= 0$

ii) For $i, j \in I$ s.t $i \cdot j = 0$,

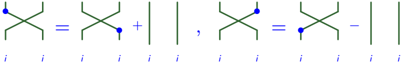


iii) For $i, j \in I$ s.t $i \cdot j = -1$,

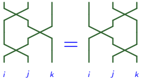


iv) For $i, j \in I$,

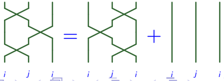
v) For $i \in I$,



vi) For $i, j, k \in I$, unless $i = k$ and $i \cdot j = -1$,

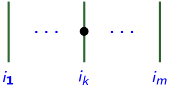
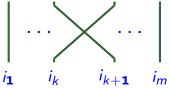


vii) For $i, j \in I$ s.t $i \cdot j = -1$,



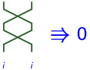
Presentation of the KLR algebras

- For $\mathbf{i} = i_1 \dots i_m \in \mathrm{Seq}(\mathcal{V})$, generators

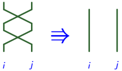
$x_{k,\mathbf{i}} =$  and $\tau_{k,\mathbf{i}} =$ 

- Relations:

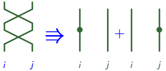
i) For $i \in I$,



ii) For $i, j \in I$ s.t $i \cdot j = 0$,

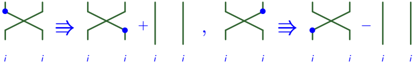


iii) For $i, j \in I$ s.t $i \cdot j = -1$,

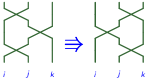


iv) For $i, j \in I$,

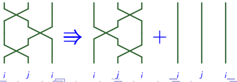
v) For $i \in I$,



vi) For $i, j, k \in I$, unless $i = k$ and $i \cdot j = -1$,



vii) For $i, j \in I$ s.t $i \cdot j = -1$,



Convergent presentation

- **Theorem [D. ’17]:** This linear $(3, 2)$-polygraph is convergent.

Convergent presentation

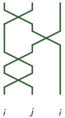
- **Theorem [D. '17]:** This linear $(3,2)$-polygraph is convergent.
  - Termination: the number of crossings decreases and the dots move to the bottom.

Convergent presentation

- **Theorem [D. '17]:** This linear $(3,2)$-polygraph is convergent.
  - Termination: the number of crossings decreases and the dots move to the bottom.
  - Confluence: exhaustive study of all critical branchings.

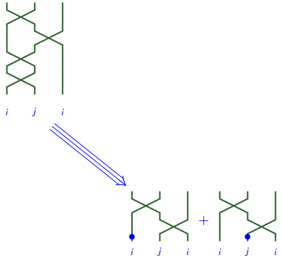
Convergent presentation

- **Theorem [D. '17]:** This linear $(3,2)$-polygraph is convergent.
  - Termination: the number of crossings decreases and the dots move to the bottom.
  - Confluence: exhaustive study of all critical branchings.



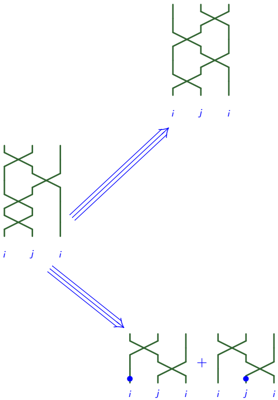
Convergent presentation

- **Theorem [D. '17]:** This linear $(3,2)$-polygraph is convergent.
  - Termination: the number of crossings decreases and the dots move to the bottom.
  - Confluence: exhaustive study of all critical branchings.



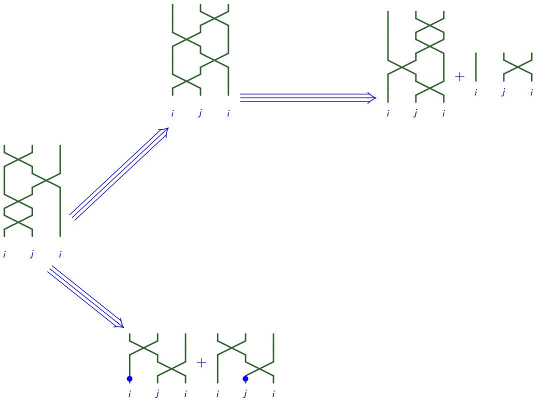
Convergent presentation

- **Theorem [D. '17]:** This linear $(3,2)$-polygraph is convergent.
  - Termination: the number of crossings decreases and the dots move to the bottom.
  - Confluence: exhaustive study of all critical branchings.



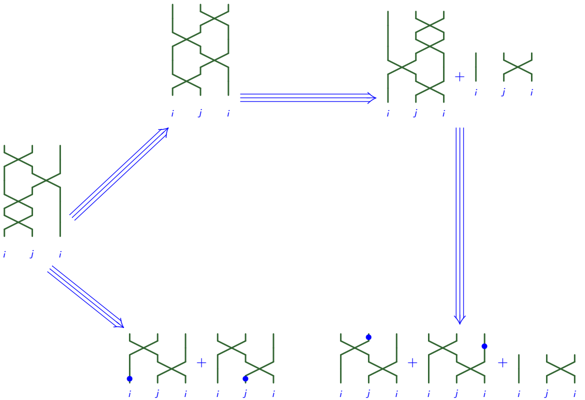
Convergent presentation

- **Theorem [D. '17]:** This linear $(3,2)$-polygraph is convergent.
  - Termination: the number of crossings decreases and the dots move to the bottom.
  - Confluence: exhaustive study of all critical branchings.



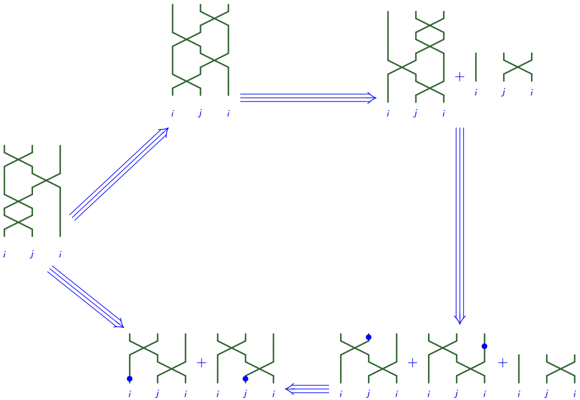
Convergent presentation

- **Theorem [D. '17]:** This linear $(3,2)$-polygraph is convergent.
  - Termination: the number of crossings decreases and the dots move to the bottom.
  - Confluence: exhaustive study of all critical branchings.



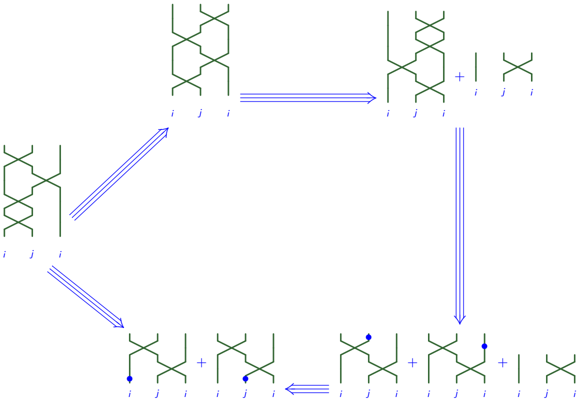
Convergent presentation

- **Theorem [D. '17]:** This linear $(3,2)$-polygraph is convergent.
  - Termination: the number of crossings decreases and the dots move to the bottom.
  - Confluence: exhaustive study of all critical branchings.



Convergent presentation

- **Theorem [D. '17]:** This linear $(3,2)$-polygraph is convergent.
  - Termination: the number of crossings decreases and the dots move to the bottom.
  - Confluence: exhaustive study of all critical branchings.

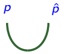


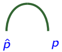
- **Corollary:** Diagrams corresponding to minimal permutations in the Coxeter presentation of the symmetric groups and dots placed at the bottom of each strand give bases of these algebras.

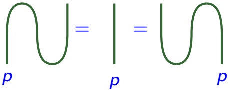
IV. Extension to rewriting modulo

Rewriting modulo

- Some structural relations may make the analysis of confluence difficult.
  - **Example:** Adjunction relations in pivotal linear 2-categories. If p is a 1-cell, a left-adjoint of p is a 1-cell $\hat{p}$ such that there are 2-cells

$\eta_p : 1 \Rightarrow p \star_0 \hat{p}$, $\quad \varepsilon_p : \hat{p} \star_0 p \Rightarrow 1$, 

, 

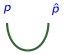
satisfying 

.

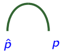
Rewriting modulo

- Some structural relations may make the analysis of confluence difficult.
  - **Example:** Adjunction relations in pivotal linear 2-categories. If p is a 1-cell, a left-adjoint of p is a 1-cell $\hat{p}$ such that there are 2-cells

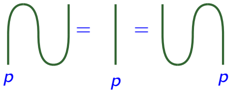
$\eta_p : 1 \Rightarrow p \star_0 \hat{p}$, $\quad \varepsilon_p : \hat{p} \star_0 p \Rightarrow 1$,



,



satisfying

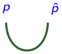


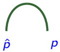
.

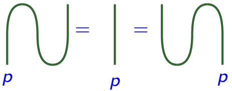
  - We rewrite modulo these rules, with a set R of oriented relations and a set E of non-oriented axioms.

Rewriting modulo

- Some structural relations may make the analysis of confluence difficult.
  - **Example:** Adjunction relations in pivotal linear 2-categories. If p is a 1-cell, a left-adjoint of p is a 1-cell $\hat{p}$ such that there are 2-cells

$\eta_p : 1 \Rightarrow p \star_0 \hat{p}, \quad \varepsilon_p : \hat{p} \star_0 p \Rightarrow 1,$ 

, 

satisfying 

.

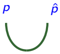
  - We rewrite modulo these rules, with a set R of oriented relations and a set E of non-oriented axioms.

- Three paradigms of rewriting modulo:

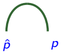
Rewriting modulo

- Some structural relations may make the analysis of confluence difficult.
  - **Example:** Adjunction relations in pivotal linear 2-categories. If p is a 1-cell, a left-adjoint of p is a 1-cell $\hat{p}$ such that there are 2-cells

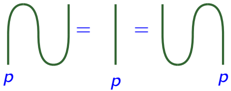
$\eta_p : 1 \Rightarrow p \star_0 \hat{p}$, $\quad \varepsilon_p : \hat{p} \star_0 p \Rightarrow 1$,



,



satisfying



.

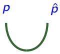
  - We rewrite modulo these rules, with a set R of oriented relations and a set E of non-oriented axioms.

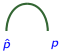
- Three paradigms of rewriting modulo:
  - Rewriting with rules in R, but confluence modulo E, Huet '80

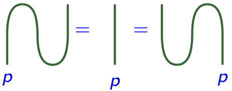
$$\begin{tikzcd} u \arrow[r, "R"] \arrow[d, leftrightarrow, "E"'] & u' \arrow[r, dotted, "R"] & w \arrow[d, leftrightarrow, "E"] \\ v \arrow[r, "R"'] & v' \arrow[r, dotted, "R"'] & w' \end{tikzcd}$$

Rewriting modulo

- Some structural relations may make the analysis of confluence difficult.
  - **Example:** Adjunction relations in pivotal linear 2-categories. If p is a 1-cell, a left-adjoint of p is a 1-cell $\hat{p}$ such that there are 2-cells

$\eta_p : 1 \Rightarrow p \star_0 \hat{p}$, $\varepsilon_p : \hat{p} \star_0 p \Rightarrow 1$, 

, 

satisfying 

.

  - We rewrite modulo these rules, with a set R of oriented relations and a set E of non-oriented axioms.

- Three paradigms of rewriting modulo:
  - Rewriting with rules in R, but confluence modulo E, Huet '80

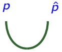
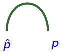
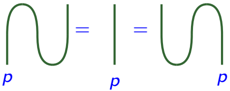
$$\begin{tikzcd} u \arrow[r, "R"] \arrow[d, leftrightarrow, "E"'] & u' \arrow[r, dotted, "R"] & w \arrow[d, leftrightarrow, "E"] \\ v \arrow[r, "R"'] & v' \arrow[r, dotted, "R"'] & w' \end{tikzcd}$$

  - Rewriting with R on E-equivalence classes:

$$\begin{tikzcd} u \arrow[r, dotted, "{}_E R_E"] \arrow[d, "E"'] & v \arrow[d, "E"] \\ u' \arrow[r, "R"'] & v' \end{tikzcd}$$

Rewriting modulo

- Some structural relations may make the analysis of confluence difficult.
  - **Example:** Adjunction relations in pivotal linear 2-categories. If p is a 1-cell, a left-adjoint of p is a 1-cell $\hat{p}$ such that there are 2-cells

$\eta_p : 1 \Rightarrow p \star_0 \hat{p}, \quad \varepsilon_p : \hat{p} \star_0 p \Rightarrow 1,$  ,  satisfying  .

  - We rewrite modulo these rules, with a set R of oriented relations and a set E of non-oriented axioms.

- Three paradigms of rewriting modulo:
  - Rewriting with rules in R, but confluence modulo E, Huet '80

$$\begin{tikzcd} u \arrow[r, "R"] \arrow[d, leftrightarrow, "E"'] & u' \arrow[r, dotted, "R"] & w \arrow[d, leftrightarrow, "E"] \\ v \arrow[r, "R"'] & v' \arrow[r, dotted, "R"'] & w' \end{tikzcd}$$

  - Rewriting with R on E-equivalence classes:

$$\begin{tikzcd} u \arrow[r, dotted, "{}_E R_E"] \arrow[d, "E"'] & v \arrow[d, "E"] \\ u' \arrow[r, "R"'] & v' \end{tikzcd}$$

  - Rewriting system modulo: (R, E, S) such that $R \subseteq S \subseteq {}_E R_E$, Jouannaud-Kirchner '84.

Results

- Confluence modulo:

$$\begin{tikzcd}
u \arrow[r, "S^*"] \arrow[d, leftrightarrow, "E"'] & u' \arrow[r, dotted, "S^*"] & w \arrow[d, dotted, "E"] \\
v \arrow[r, "S^*"'] & v' \arrow[r, dotted, "S^*"'] & w'
\end{tikzcd}$$

Results

- Confluence modulo:

$$\begin{tikzcd} u \arrow[r, "S^*"] \arrow[d, leftrightarrow, "E"'] & u' \arrow[r, dotted, "S^*"] & w \arrow[d, dotted, "E"] \\ v \arrow[r, "S^*"'] & v' \arrow[r, dotted, "S^*"'] & w' \end{tikzcd}$$

- **Theorem [D. - Malbos '18], Critical pair lemma modulo** : For (R, E, S) such that ${}_E R_E$ is terminating, S is confluent modulo E if and only if its critical branchings modulo E of the form

$$\begin{tikzcd} u \arrow[r, "S^{*(1)}"] \arrow[d, "||"'] & v \arrow[r, dotted, "S^*"] & v' \arrow[d, dotted, "E^{\top}"] \\ u \arrow[r, "R^{*(1)}"'] & w \arrow[r, dotted, "S^*"'] & w' \end{tikzcd} \qquad \begin{tikzcd} u \arrow[r, "S^{*(1)}"] \arrow[d, "E^{\top(1)}"'] & v \arrow[r, dotted, "S^*"] & v' \arrow[d, dotted, "E^{\top}"] \\ u' \arrow[rr, dotted, "S^*"'] & & w \end{tikzcd}$$

are confluent modulo E.

Results

- Confluence modulo:

$$\begin{tikzcd} u \arrow[r, "S^*"] \arrow[d, leftrightarrow, "E"'] & u' \arrow[r, dotted, "S^*"] & w \arrow[d, dotted, "E"] \\ v \arrow[r, "S^*"'] & v' \arrow[r, dotted, "S^*"'] & w' \end{tikzcd}$$

- **Theorem [D. - Malbos '18], Critical pair lemma modulo** : For (R, E, S) such that ${}_E R_E$ is terminating, S is confluent modulo E if and only if its critical branchings modulo E of the form

$$\begin{tikzcd} u \arrow[r, "S^{*(1)}"] \arrow[d, "\parallel"'] & v \arrow[r, dotted, "S^*"] & v' \arrow[d, dotted, "E^{\top}"] \\ u \arrow[r, "R^{*(1)}"'] & w \arrow[r, dotted, "S^*"'] & w' \end{tikzcd} \qquad \begin{tikzcd} u \arrow[r, "S^{*(1)}"] \arrow[d, "E^{\top(1)}"'] & v \arrow[r, dotted, "S^*"] & v' \arrow[d, dotted, "E^{\top}"] \\ u' \arrow[rr, dotted, "S^*"'] & & w \end{tikzcd}$$

  are confluent modulo E.

- **Theorem [D. '19]** Let (R, E, S) be a linear $(3, 2)$-polygraph modulo and $\mathcal{C}$ the category presented by $R \coprod E$, such that S is terminating and confluent modulo E.

Results

- Confluence modulo:

$$\begin{tikzcd} u \arrow[r, "S^*"] \arrow[d, leftrightarrow, "E"'] & u' \arrow[r, dotted, "S^*"] & w \arrow[d, dotted, "E"] \\ v \arrow[r, "S^*"'] & v' \arrow[r, dotted, "S^*"'] & w' \end{tikzcd}$$

- **Theorem [D. - Malbos '18], Critical pair lemma modulo** : For (R, E, S) such that ${}_E R_E$ is terminating, S is confluent modulo E if and only if its critical branchings modulo E of the form

$$\begin{tikzcd} u \arrow[r, "S^{*(1)}"] \arrow[d, "\|"'] & v \arrow[r, dotted, "S^*"] & v' \arrow[d, dotted, "E^\top"] \\ u \arrow[r, "R^{*(1)}"'] & w \arrow[r, dotted, "S^*"'] & w' \end{tikzcd} \qquad \begin{tikzcd} u \arrow[r, "S^{*(1)}"] \arrow[d, "E^{\top(1)}"'] & v \arrow[r, dotted, "S^*"] & v' \arrow[d, dotted, "E^\top"] \\ u' \arrow[rr, dotted, "S^*"'] & & w \end{tikzcd}$$

 are confluent modulo E.

- **Theorem [D. '19]** Let (R, E, S) be a linear $(3, 2)$-polygraph modulo and $\mathcal{C}$ the category presented by $R \coprod E$, such that S is terminating and confluent modulo E.

 Then, for all parallel 1-cells p and q, the set of monomials in the E-normal forms of monomials in normal form for S gives a basis of $\mathcal{C}_2(p, q)$.

Example: The 2-category $\mathcal{KLR}(\mathfrak{sl}_2)$

- Let $\mathcal{KLR}$ be the linear 2-category defined by:

Example: The 2-category $\mathcal{KLR}(\mathfrak{sl}_2)$

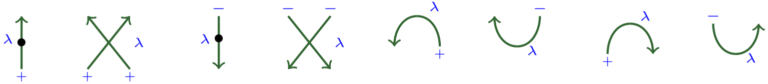
- Let $\mathcal{KLR}$ be the linear 2-category defined by:
  - $\mathcal{KLR}_{\mathbf{0}} = X$ weight lattice of a Kac-Moody algebra,

Example: The 2-category $\mathcal{KLR}(\mathfrak{sl}_2)$

- Let $\mathcal{KLR}$ be the linear 2-category defined by:
  - $\mathcal{KLR}_{\mathbf{0}} = X$ weight lattice of a Kac-Moody algebra,
  - $\mathcal{KLR}_{\mathbf{1}} = \{\underline{\varepsilon} = (\varepsilon_{\mathbf{1}}, \ldots, \varepsilon_{\ell(\varepsilon)})$ with $\varepsilon_i \in \{-,+\}\}$.

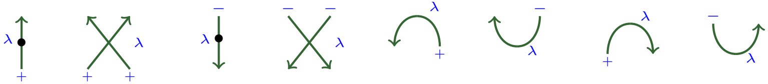
Example: The 2-category $\mathcal{KLR}(\mathfrak{sl}_2)$

- Let $\mathcal{KLR}$ be the linear 2-category defined by:
  - $\mathcal{KLR}_{\mathbf{0}} = X$ weight lattice of a Kac-Moody algebra,
  - $\mathcal{KLR}_{\mathbf{1}} = \{\underline{\varepsilon} = (\varepsilon_{\mathbf{1}}, \ldots, \varepsilon_{\ell(\varepsilon)})$ with $\varepsilon_i \in \{-,+\}\}$.
  - $\mathcal{KLR}_{\mathbf{2}}$ is the set of following generating 2-cells



Example: The 2-category $\mathcal{KLR}(\mathfrak{sl}_2)$

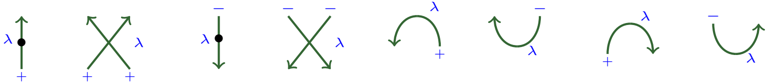
- Let $\mathcal{KLR}$ be the linear 2-category defined by:
  - $\mathcal{KLR}_\mathbf{0} = X$ weight lattice of a Kac-Moody algebra,
  - $\mathcal{KLR}_\mathbf{1} = \{\underline{\varepsilon} = (\varepsilon_\mathbf{1}, \dots, \varepsilon_{\ell(\varepsilon)})$ with $\varepsilon_i \in \{-,+\}\}$.
  - $\mathcal{KLR}_\mathbf{2}$ is the set of following generating 2-cells



- subject to the following relations:

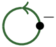
Example: The 2-category $\mathcal{KLR}(\mathfrak{sl}_2)$

- Let $\mathcal{KLR}$ be the linear 2-category defined by:
  - $\mathcal{KLR}_{\mathbf{0}} = X$ weight lattice of a Kac-Moody algebra,
  - $\mathcal{KLR}_{\mathbf{1}} = \{\underline{\varepsilon} = (\varepsilon_{\mathbf{1}}, \dots, \varepsilon_{\ell(\varepsilon)})$ with $\varepsilon_i \in \{-,+\}\}$.
  - $\mathcal{KLR}_{\mathbf{2}}$ is the set of following generating 2-cells



- subject to the following relations:
  - KLR algebras relations for both orientations.
  - Bubble relations:

n  $\lambda \Rrightarrow \begin{cases} 1_{\mathbf{1}_\lambda} & \text{if } n = h-1 \\ 0 & \text{if } n < h-1 \end{cases}$; λ  $n \Rrightarrow \begin{cases} 1_{\mathbf{1}_\lambda} & \text{if } n = -h-1 \\ 0 & \text{if } n < -h-1 \end{cases}$

$h-\mathbf{1}+\alpha$  $\lambda \Rrightarrow -\sum_{l=\mathbf{1}}^{\alpha}$ $h-\mathbf{1}+\alpha-l$  λ  $-h-\mathbf{1}+l$ for all $\lambda \in X$ and $\alpha > 0$

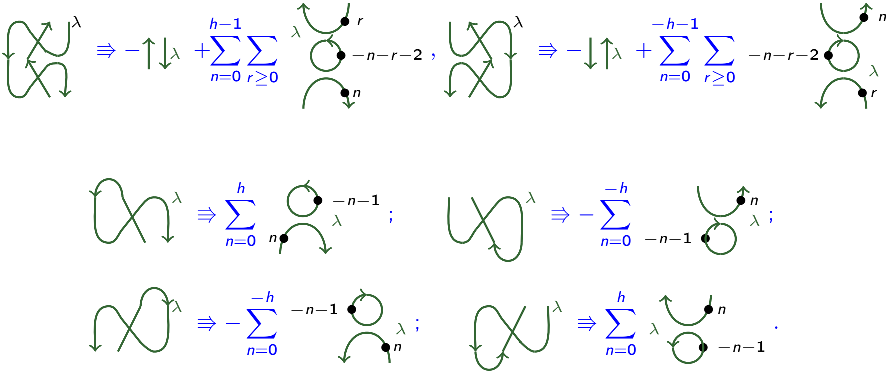
Example: The 2-category $\mathcal{KLR}(\mathfrak{sl}_2)$

- Isotopy relations: 

Example: The 2-category $\mathcal{KLR}(\mathfrak{sl}_2)$

- Isotopy relations:

- Quantum relations:



- Bubble slide relations.