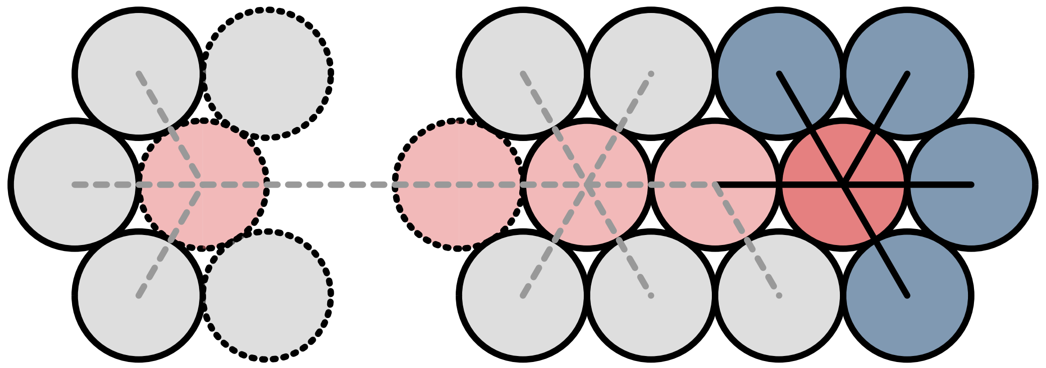
Weak Unit Disk Contact Representations for Graphs **without Embedding**

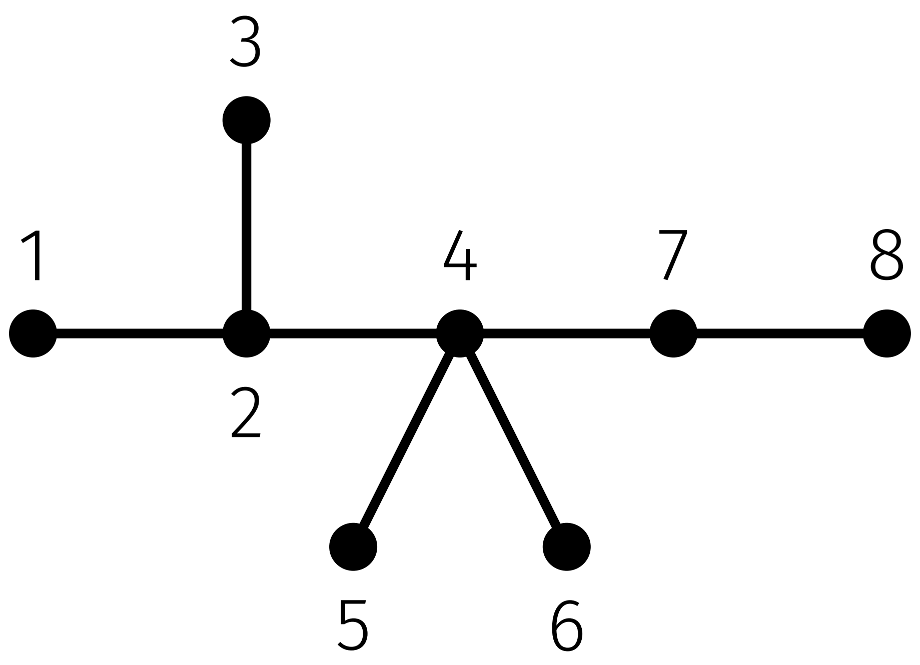
Jonas Cleve, Freie Universität Berlin



EuroCG 2020 – March 16–18 – Würzburg, Germany

Unit Disk Contact Representations

Unit disk contact representation (UDCR) of $G = (V, E)$:

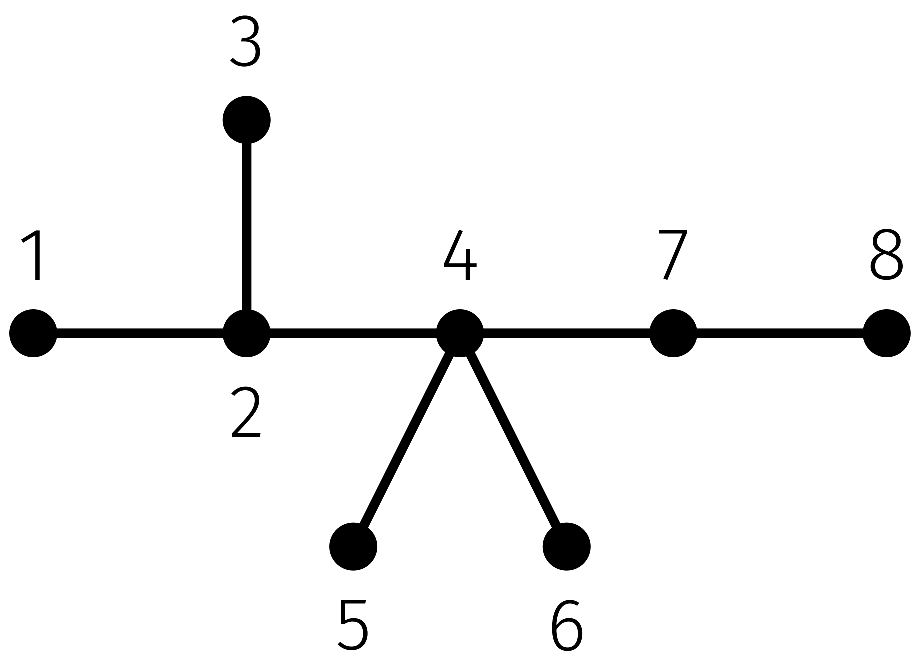


graph G

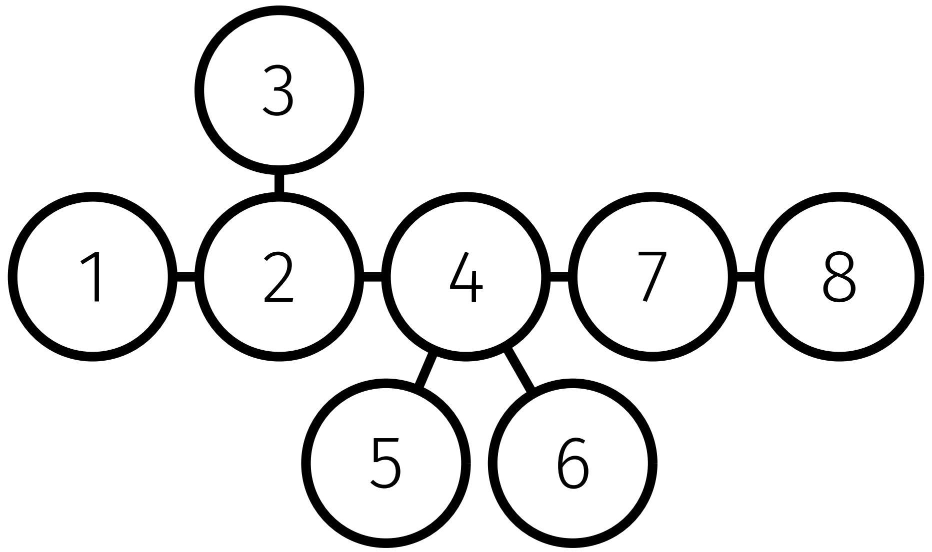
Unit Disk Contact Representations

Unit disk contact representation (UDCR) of $G = (V, E)$:

- one **unit-disk** $D(v)$ **per node** v s. t.



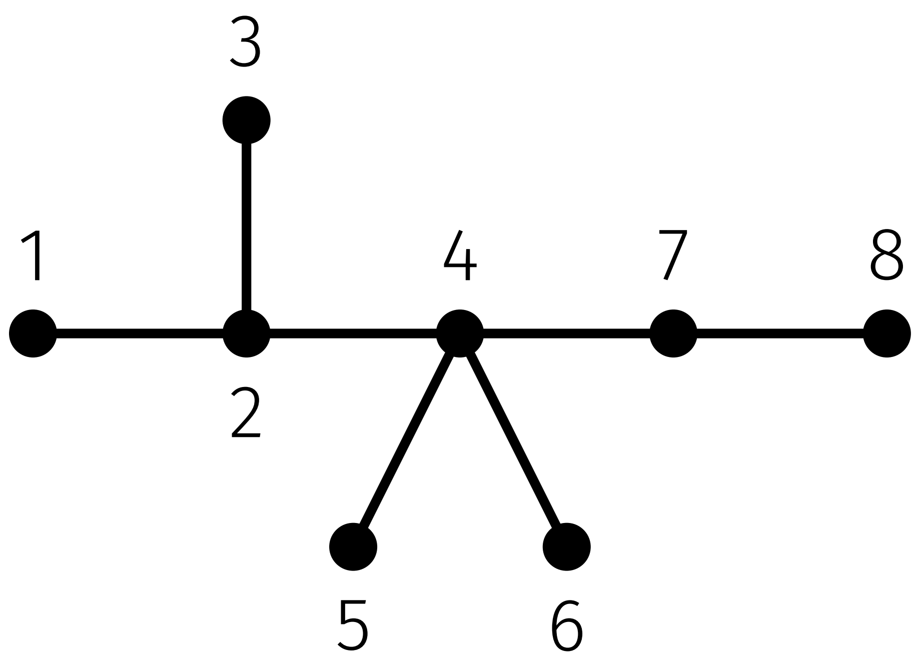
graph G



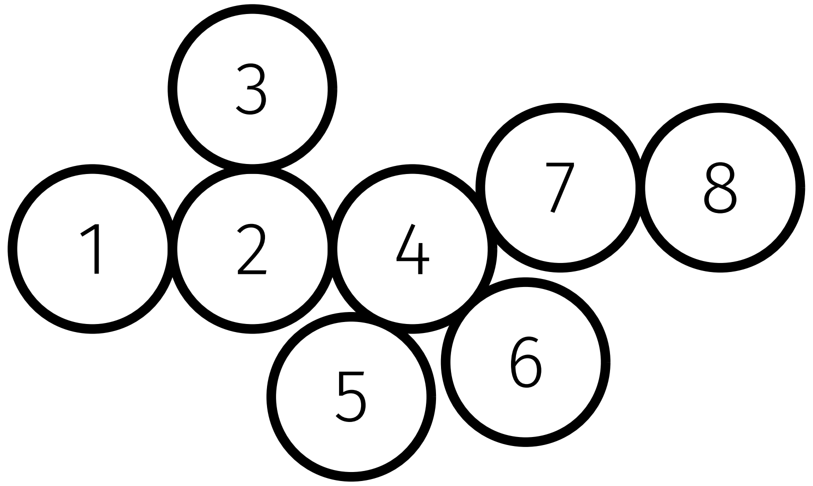
Unit Disk Contact Representations

Unit disk contact representation (UDCR) of $G = (V, E)$:

- one **unit-disk** $D(v)$ **per node** v s. t.
- $\{u, v\} \in E \iff D(u), D(v)$ **touch**.



graph G



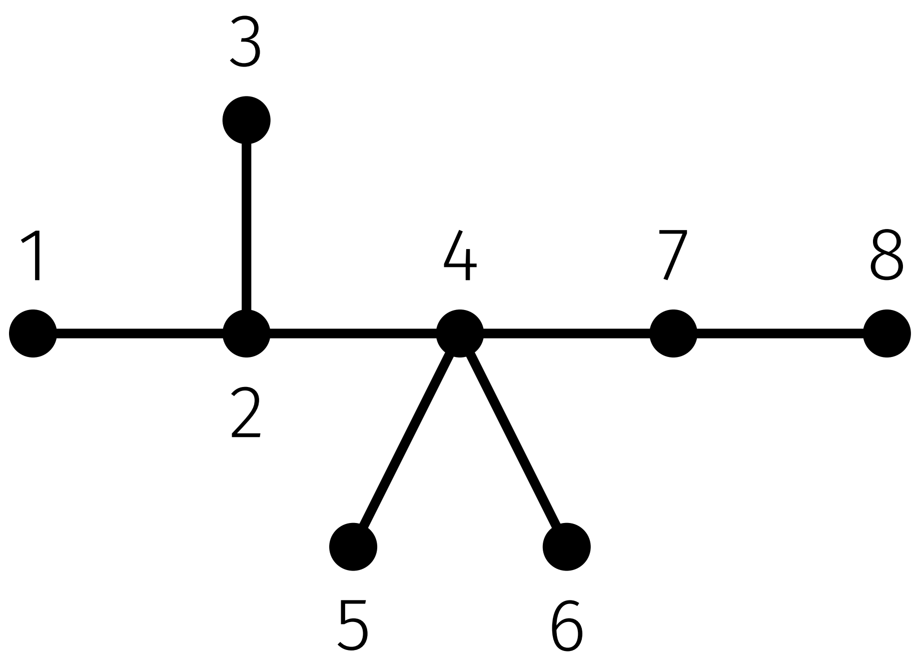
UDCR of G

Unit Disk Contact Representations

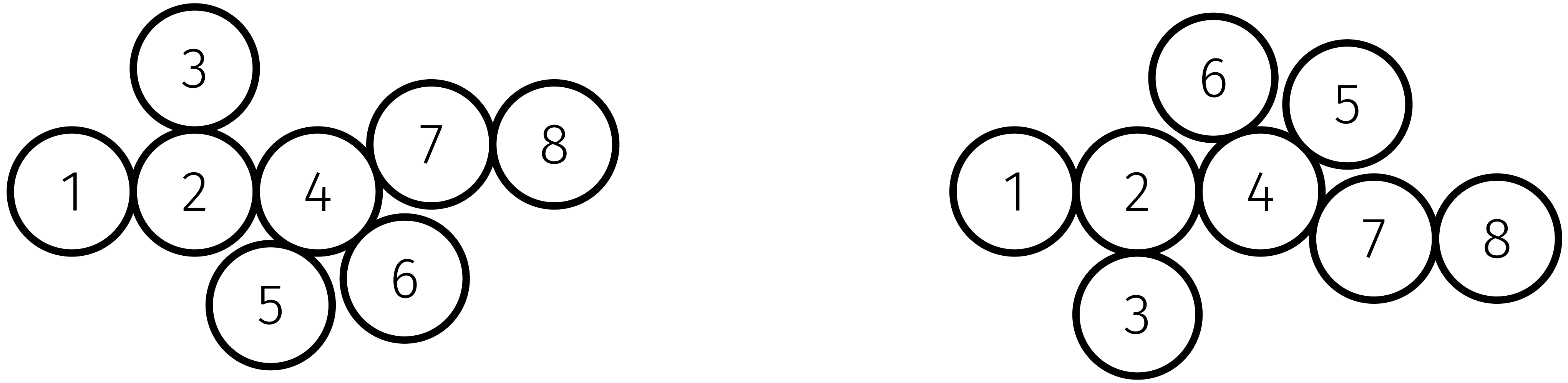
Unit disk contact representation (UDCR) of $G = (V, E)$:

- one **unit-disk** $D(v)$ **per node** v s. t.
- $\{u, v\} \in E \iff D(u), D(v)$ **touch**.

No embedding: **neighbor order** can be chosen arbitrarily



graph G

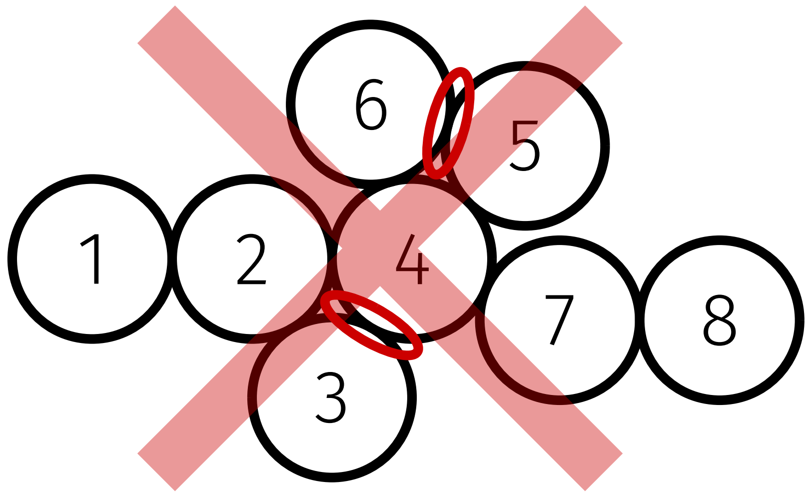


UDCRs of G

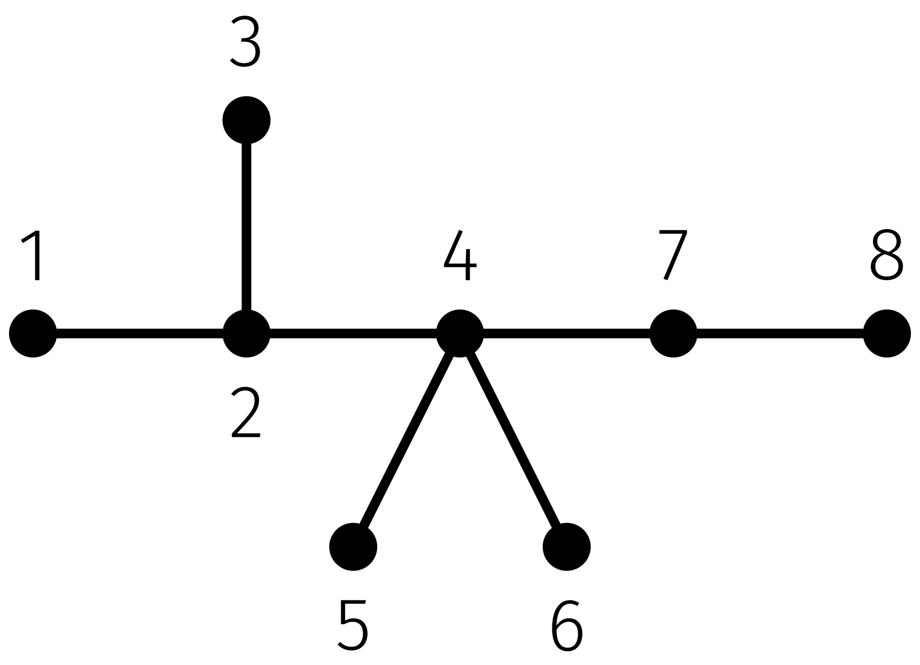
Unit Disk Contact Representations

Unit disk contact representation (UDCR) of $G = (V, E)$:

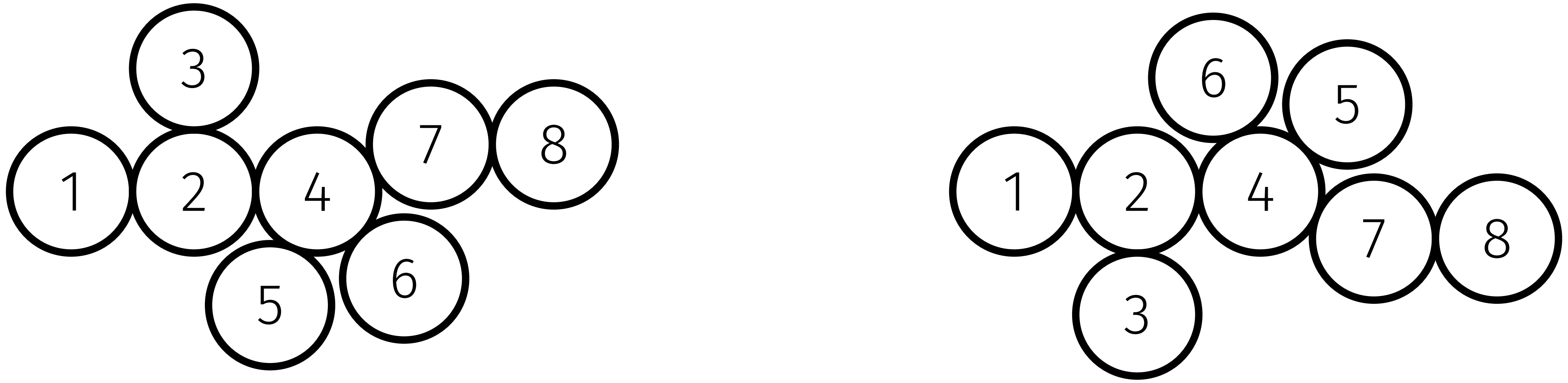
- one **unit-disk** $D(v)$ **per node** v s. t.
- $\{u, v\} \in E \iff D(u), D(v)$ **touch**.



No embedding: **neighbor order** can be chosen arbitrarily



graph G

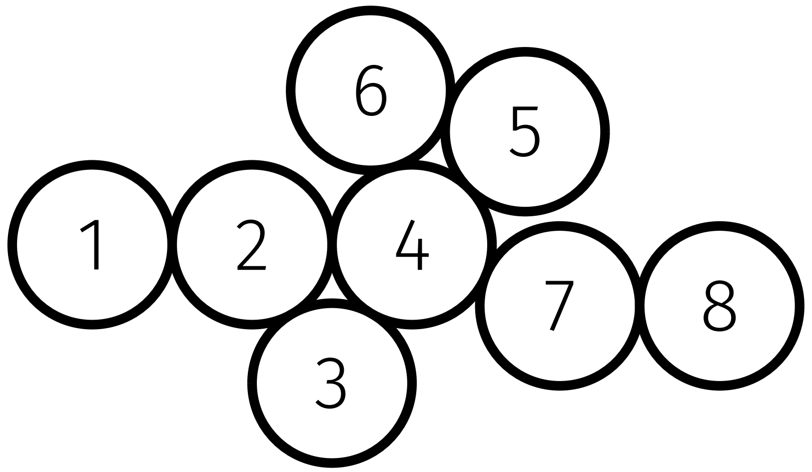


UDCRs of G

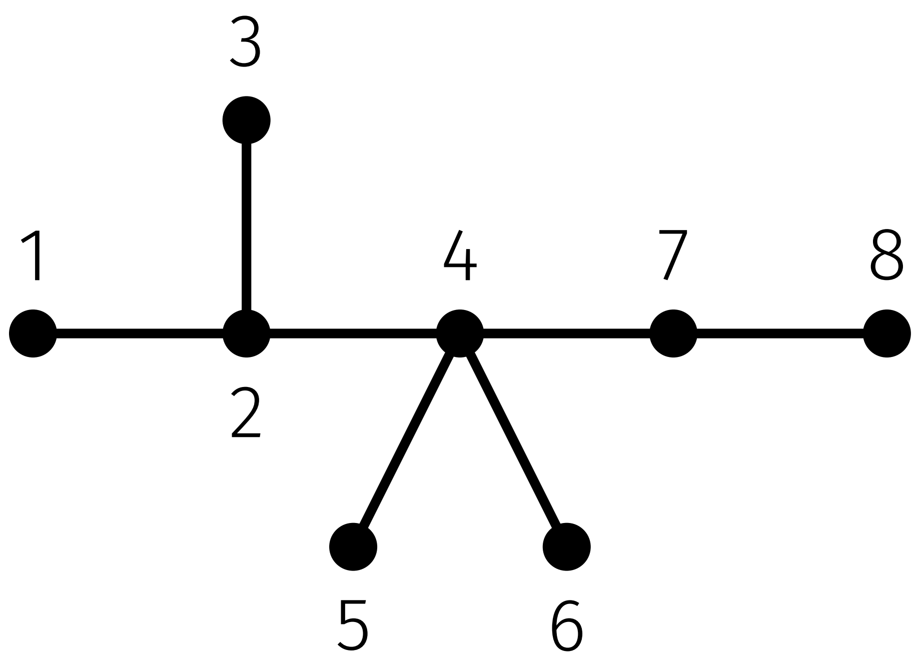
Unit Disk Contact Representations

Weak **unit disk contact representation (WUDCR)** of $G = (V, E)$:

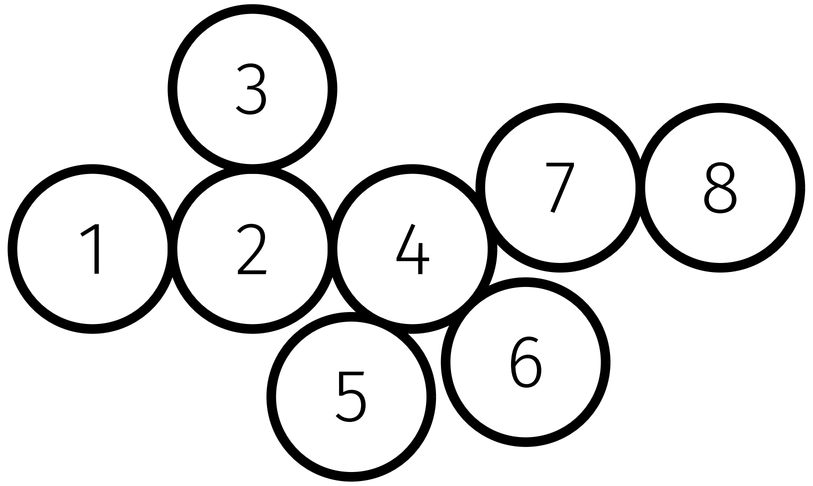
- one **unit-disk** $D(v)$ **per node** v s. t.
- $\{u, v\} \in E \implies D(u), D(v)$ **touch**.

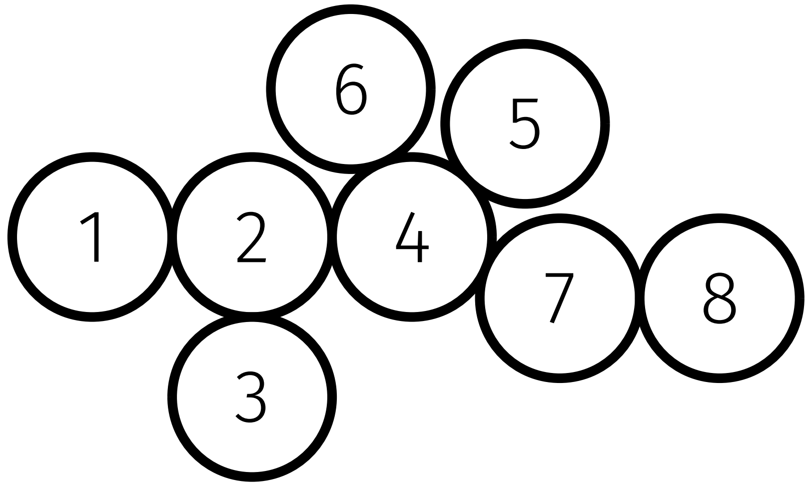


No embedding: **neighbor order** can be chosen arbitrarily



graph G





WUDCRs of G

Results

Freie Universität Berlin

Existing Work: NP-hardness and construction algorithms.

	star	caterpillar	tree
Weak UDCR		Cleve; 2020	Cleve; 2020
Weak Emb. UDCR		Chiu, Cleve, Nöllenburg; 2019	

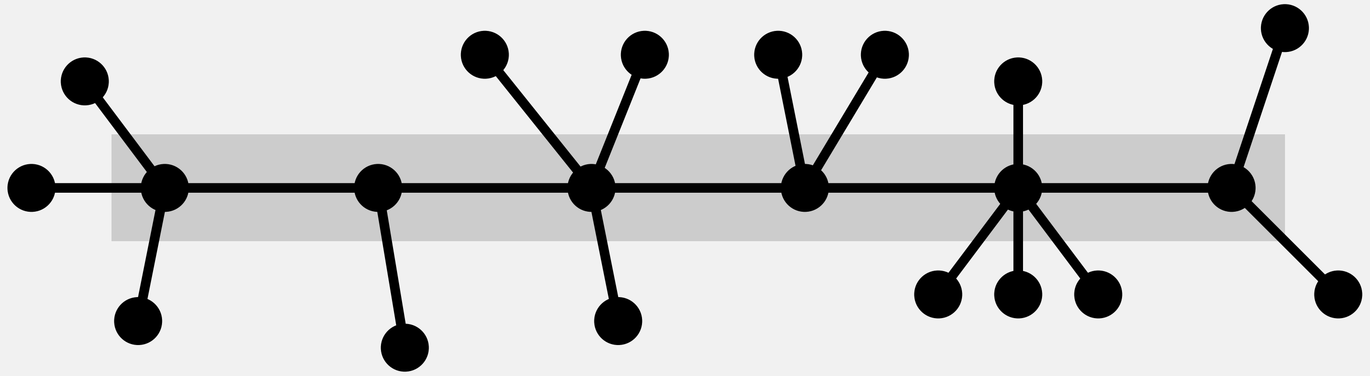
Results

Freie Universität Berlin

Existing Work: NP-hardness and construction algorithms.

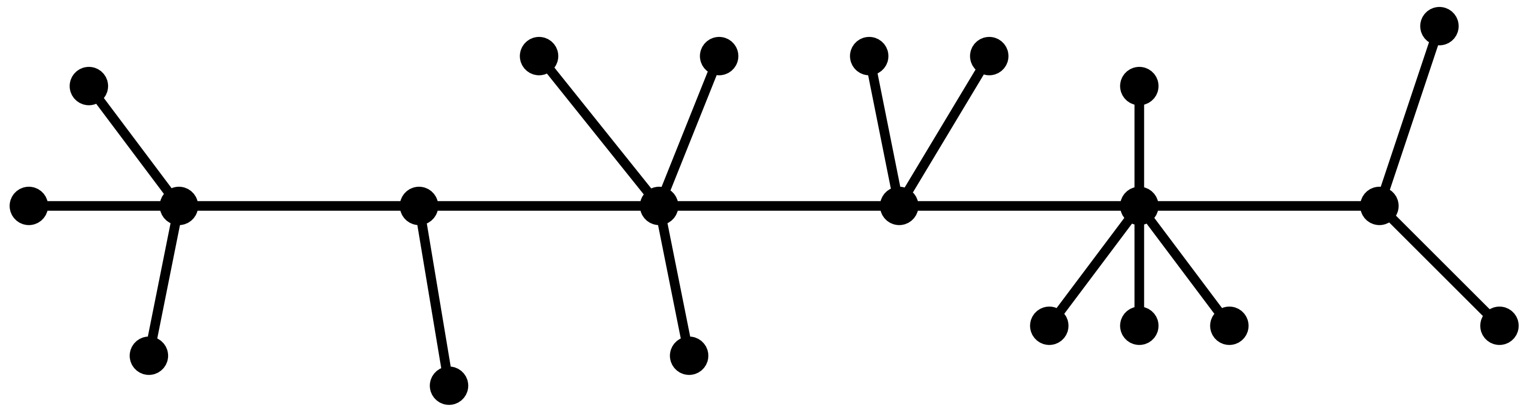
	star	caterpillar	tree
Weak UDCR		**Cleve; 2020**	**Cleve; 2020**
Weak Emb. UDCR		Chiu, Cleve, Nöllenburg; 2019	

Reminder:
- A **caterpillar** is a tree
- **backbone** path plus leaves

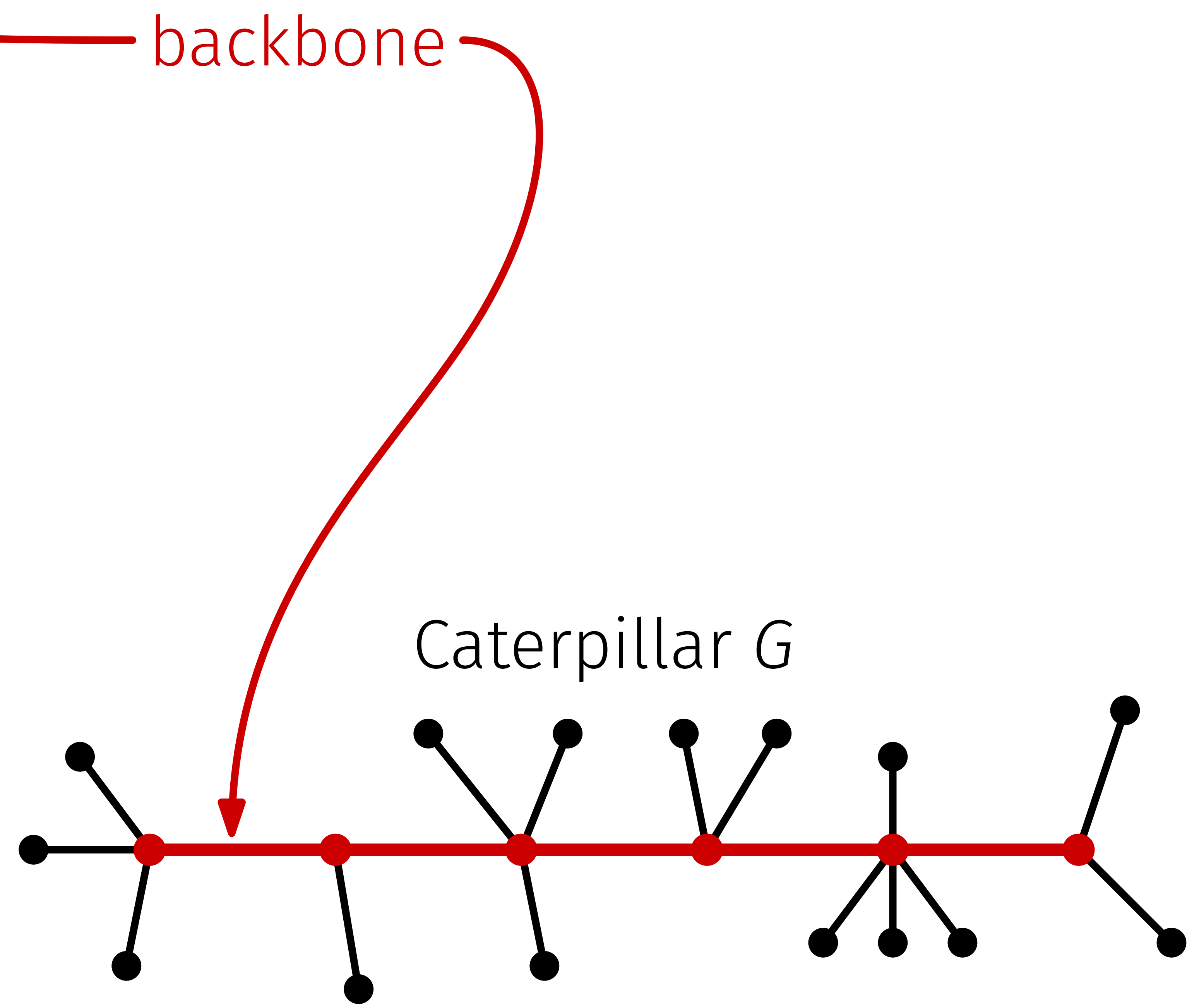


Algorithm:

Caterpillar G

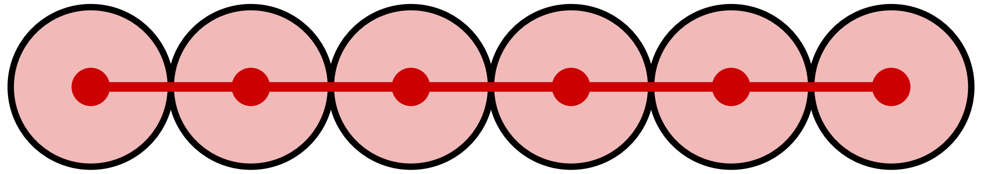


Algorithm: Find longest path $v_0, v_1, \ldots, v_k, v_{k+1}$

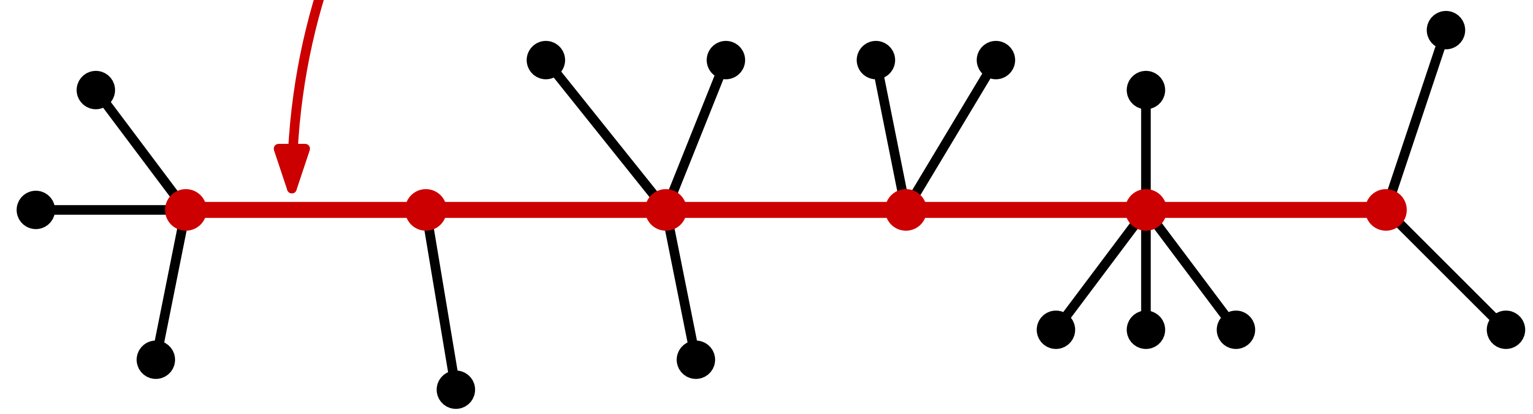


Algorithm: Find longest path $v_0, \underbrace{v_1, \ldots, v_k}_{\text{backbone}}, v_{k+1}$

Place backbone disks horizontally



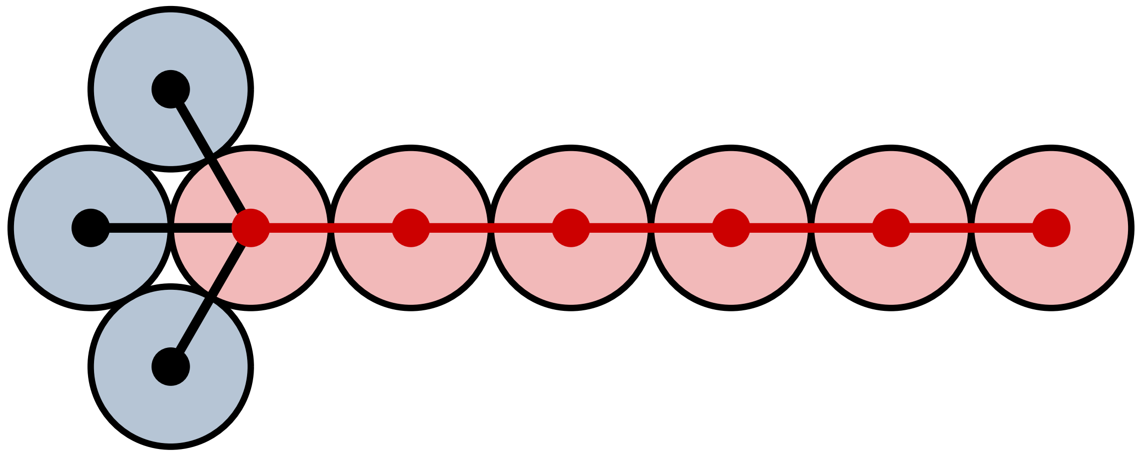
Caterpillar G



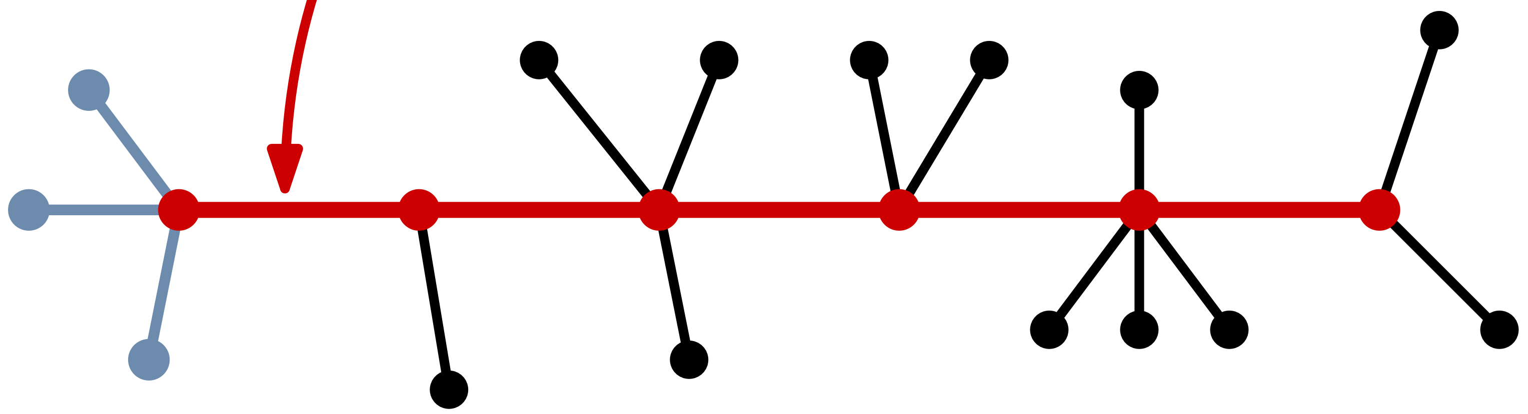
Algorithm: Find longest path $v_0, \underbrace{v_1, \ldots, v_k}_{\text{backbone}}, v_{k+1}$

Place backbone disks horizontally

Place leaf disks at leftmost position



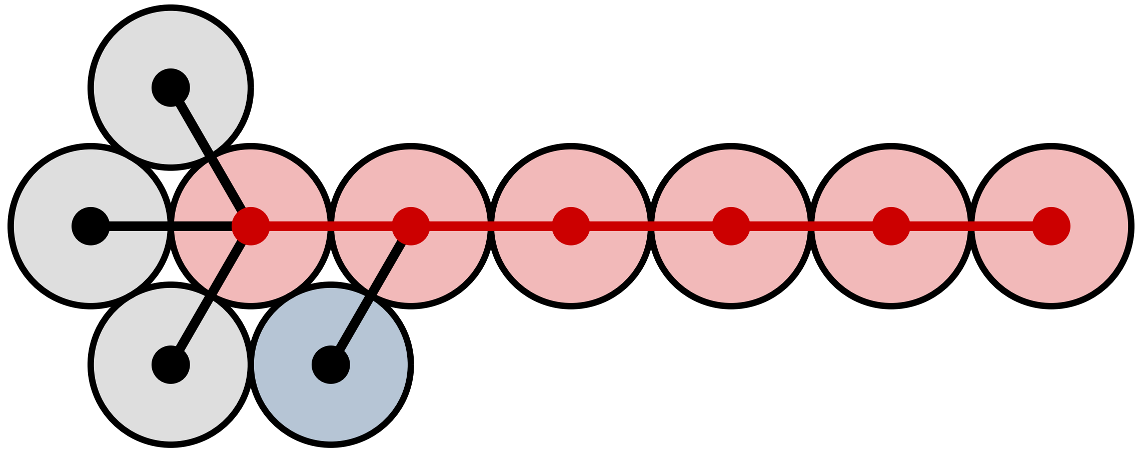
Caterpillar G



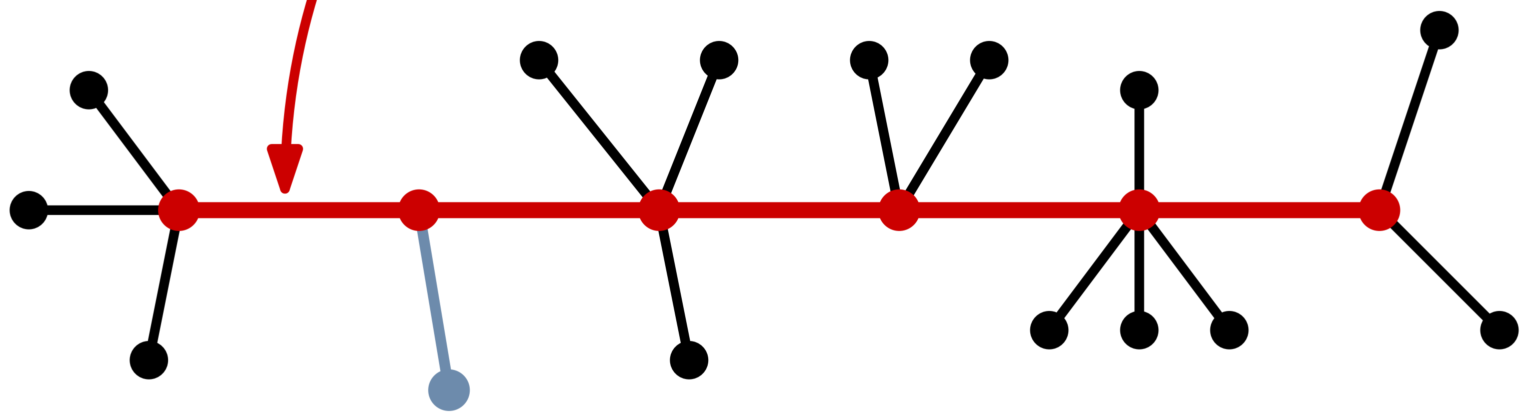
Algorithm: Find longest path $v_0, \underbrace{v_1, \ldots, v_k}_{\text{backbone}}, v_{k+1}$

Place backbone disks horizontally

Place leaf disks at leftmost position



Caterpillar G

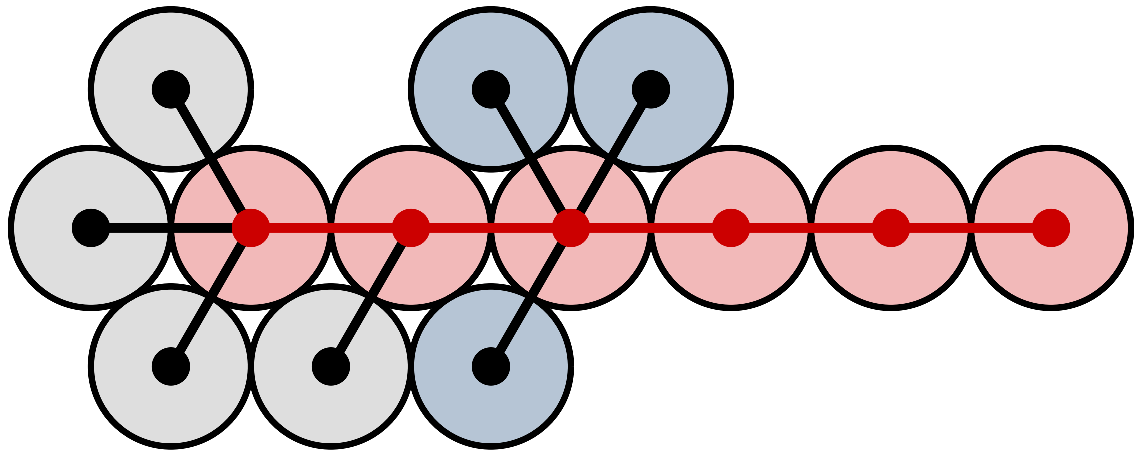


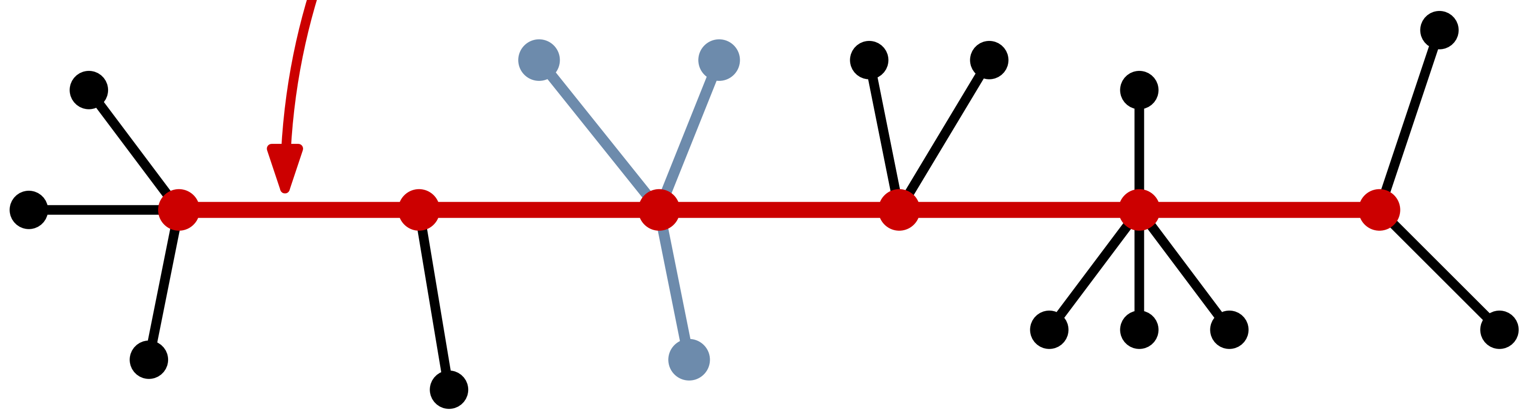
Recognizing Caterpillars in Linear Time

Algorithm: Find longest path $v_0, \underbrace{v_1, \ldots, v_k}_{\text{backbone}}, v_{k+1}$

Place backbone disks horizontally

Place leaf disks at leftmost position



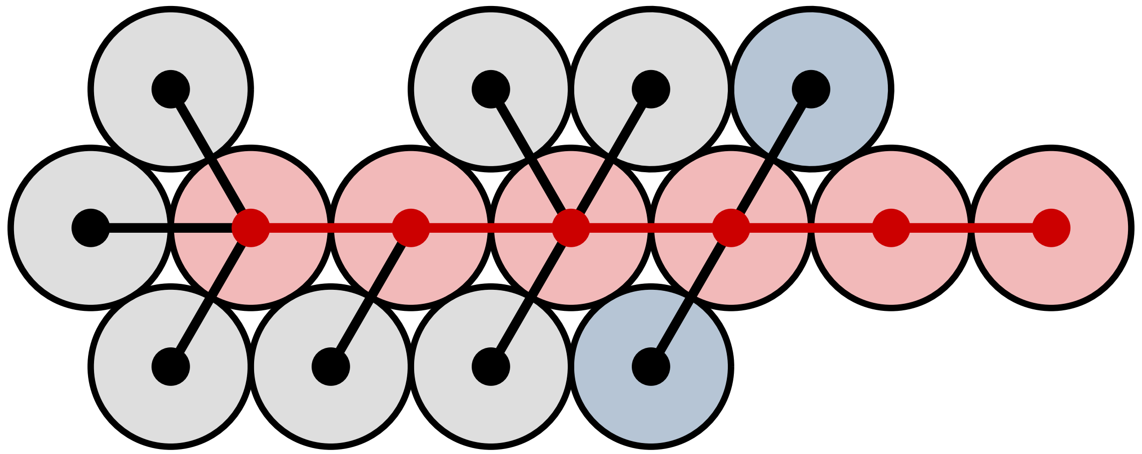


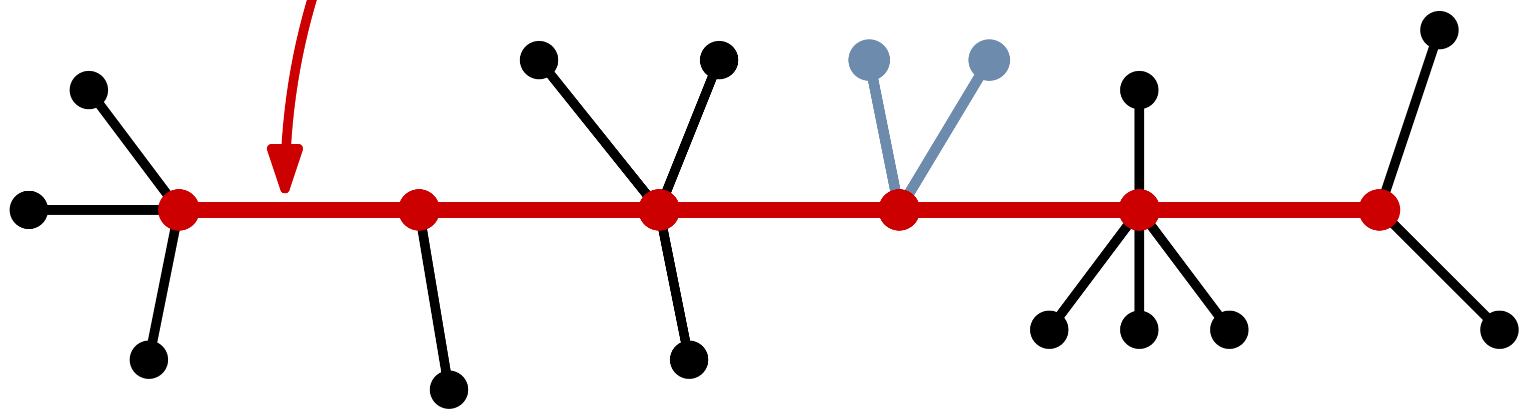
Algorithm: Find longest path $v_0, v_1, \ldots, v_k, v_{k+1}$

backbone

Place backbone disks horizontally

Place leaf disks at leftmost position

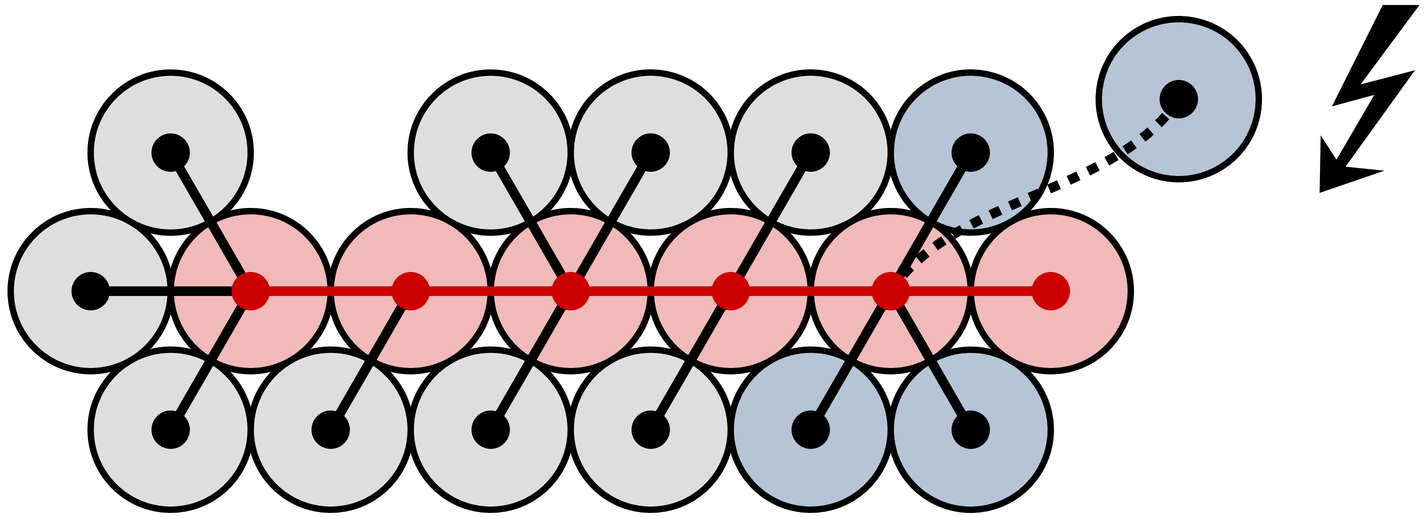


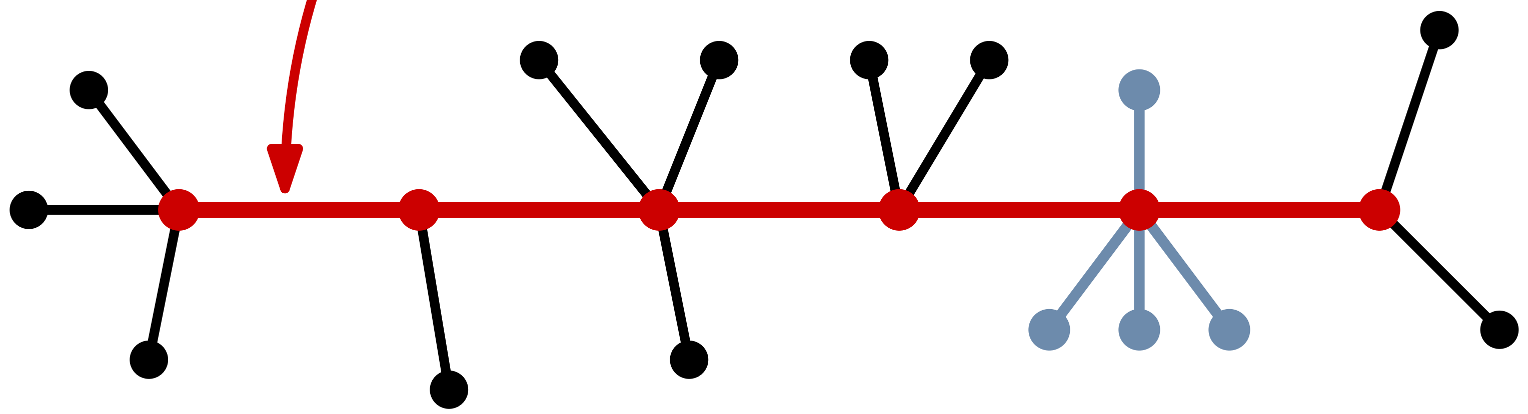


Algorithm: Find longest path $v_0, \underbrace{v_1, \ldots, v_k}_{\text{backbone}}, v_{k+1}$

Place backbone disks horizontally

Place leaf disks at leftmost position





NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "$\exists$ **weak UDCR for** T**?**" is NP-hard.

NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "$\exists$ **weak UDCR for** T**?**" is NP-hard.

Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "$\exists$ **weak UDCR for** T**?**" is NP-hard.

Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "$\exists$ **weak UDCR for** T**?**" is NP-hard.

Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "$\exists$ **weak UDCR for** T**?**" is NP-hard.

Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

Each clause: $\{0, 0, 1\}$ or $\{0, 1, 1\}$

NP-hardness of Recognizing Trees

Freie Universität Berlin

Theorem: Given a tree T, answering "$\exists$ **weak UDCR for** T**?**" is NP-hard.

Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

Each clause: $\{0,0,1\}$ or $\{0,1,1\}$

Example:

$$c_1 = (x_1, x_2, x_3)$$
$$c_2 = (x_1, \overline{x_2}, x_4)$$
$$c_3 = (x_1, x_3, \overline{x_4})$$

NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "$\exists$ **weak UDCR for** T**?**" is NP-hard.

Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

Each clause: $\{0, 0, 1\}$ or $\{0, 1, 1\}$

Example:

$c_1 = (x_1, x_2, x_3)$

$c_2 = (x_1, \overline{x_2}, x_4)$

$c_3 = (x_1, x_3, \overline{x_4})$

Logic engine

NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "∃ **weak UDCR for** T**?**" is NP-hard.

Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

Each clause: $\{0, 0, 1\}$ or $\{0, 1, 1\}$

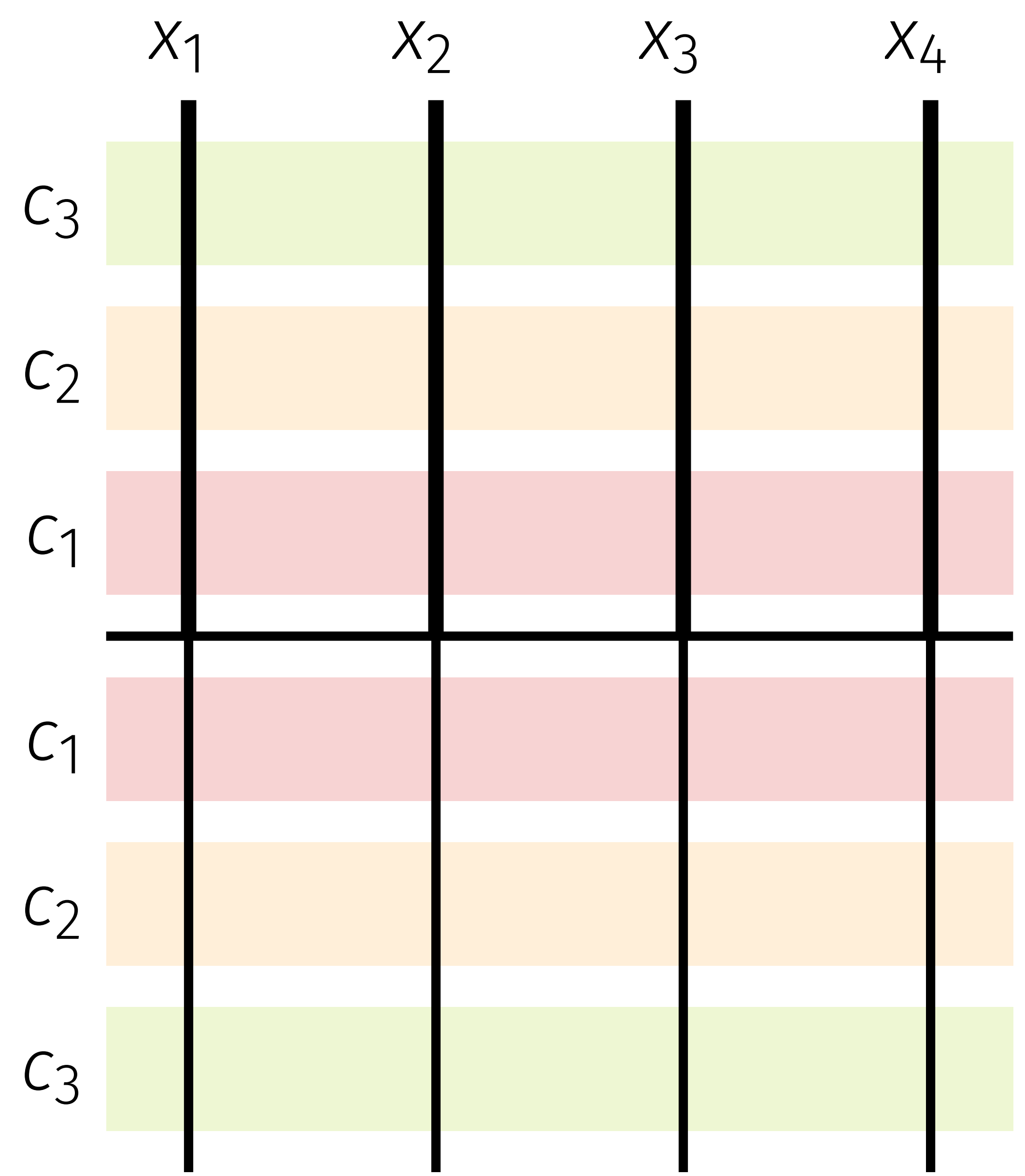
Example: $c_1 = (x_1, x_2, x_3)$

$c_2 = (x_1, \overline{x_2}, x_4)$

$c_3 = (x_1, x_3, \overline{x_4})$

Logic engine

- variable poles & clause levels



NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "∃ **weak UDCR for** T**?**" is NP-hard.

Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

Each clause: $\{0, 0, 1\}$ or $\{0, 1, 1\}$

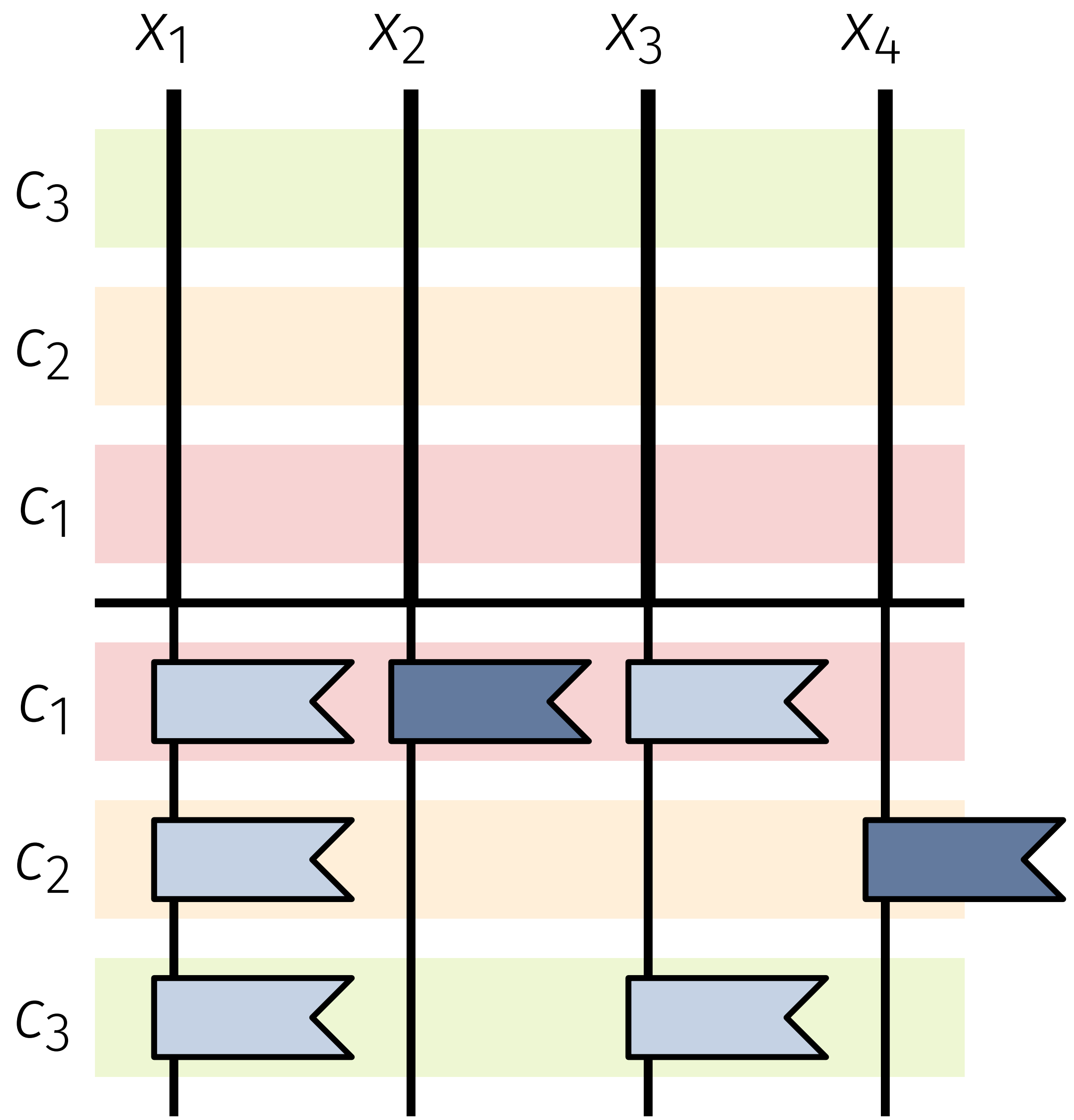
Example: $c_1 = (x_1, x_2, x_3)$

$c_2 = (x_1, \overline{x_2}, x_4)$

$c_3 = (x_1, x_3, \overline{x_4})$

Logic engine

- variable poles & clause levels
- positive literal: flag on bottom



NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "**∃ weak UDCR for T?**" is NP-hard.

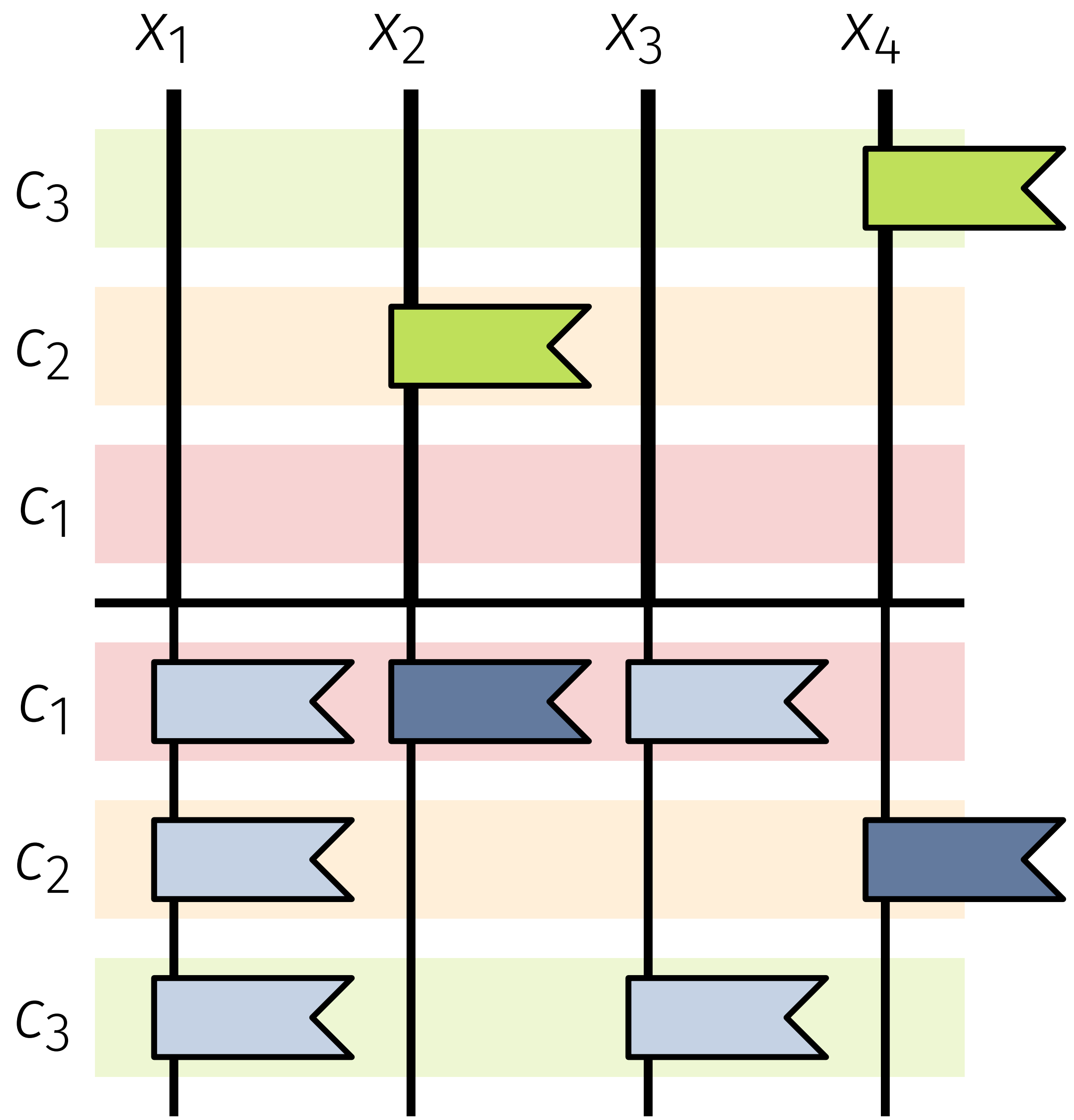
Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

Each clause: $\{0, 0, 1\}$ or $\{0, 1, 1\}$

Example: $c_1 = (x_1, x_2, x_3)$
$c_2 = (x_1, \overline{x_2}, x_4)$
$c_3 = (x_1, x_3, \overline{x_4})$

Logic engine

- variable poles & clause levels
- positive literal: flag on bottom
- negative literal: flag on top



NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "**∃ weak UDCR for** T**?**" is NP-hard.

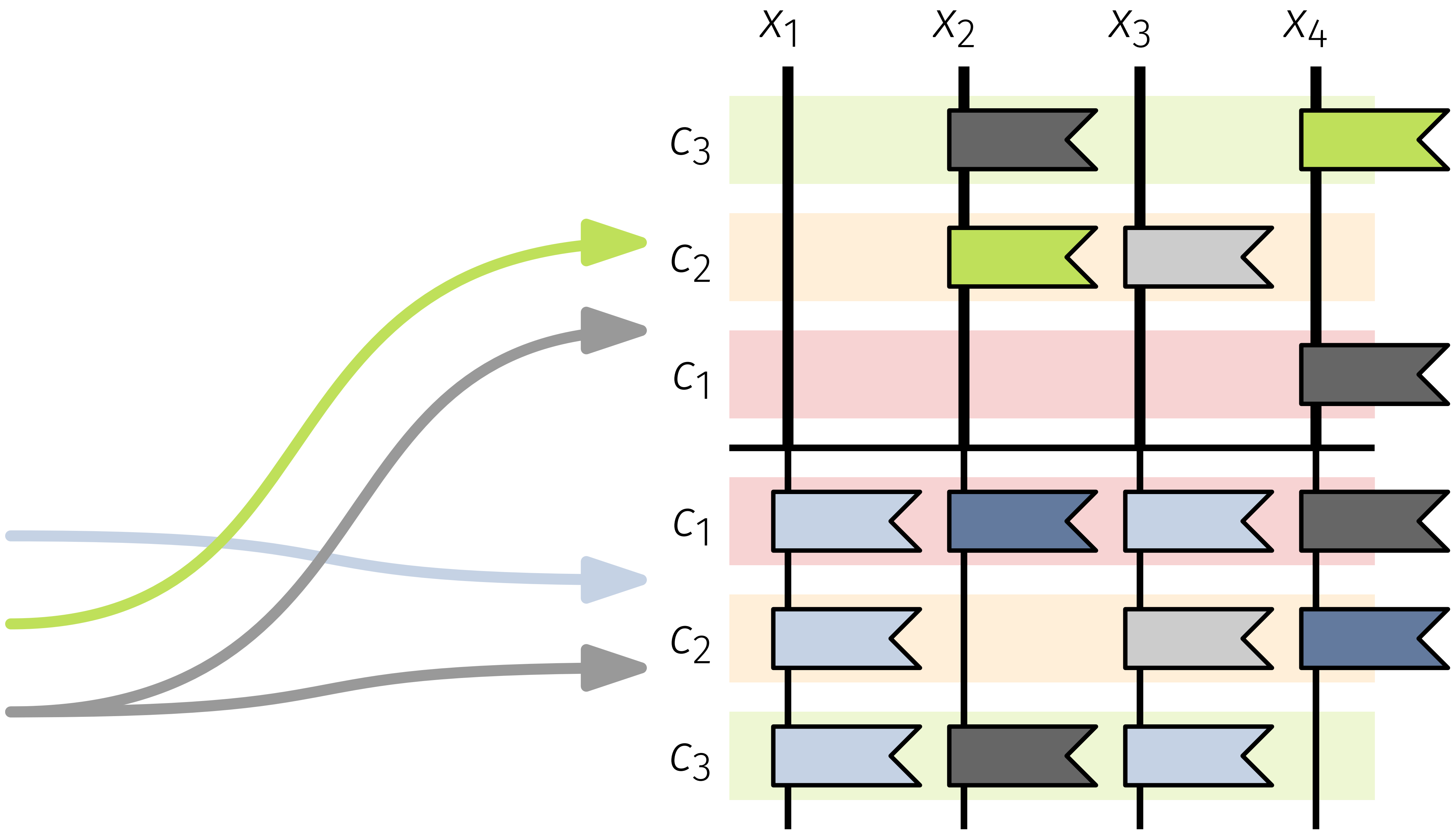
Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

Each clause: $\{0, 0, 1\}$ or $\{0, 1, 1\}$

Example:
$c_1 = (x_1, x_2, x_3)$ $\quad x_4$
$c_2 = (x_1, \overline{x_2}, x_4)$ $\quad x_3$
$c_3 = (x_1, x_3, \overline{x_4})$ $\quad x_2$

Logic engine

- variable poles & clause levels
- positive literal: flag on bottom
- negative literal: flag on top
- no literal: both flags



NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "$\exists$ **weak UDCR for** T**?**" is NP-hard.

Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

Each clause: $\{0, 0, 1\}$ or $\{0, 1, 1\}$

Example:

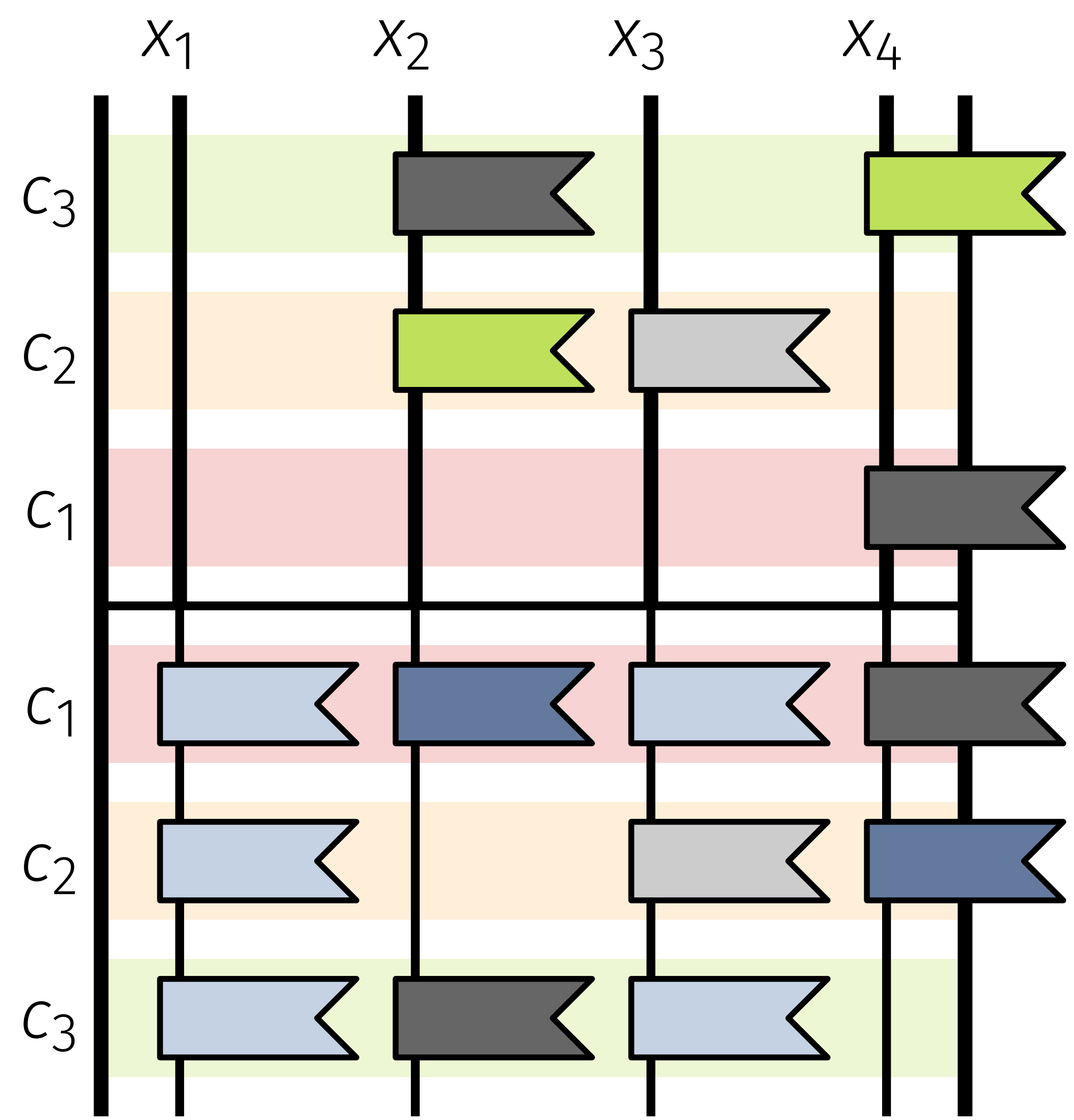
$c_1 = (x_1, x_2, x_3)$ $\quad x_4$

$c_2 = (x_1, \overline{x_2}, x_4)$ $\quad x_3$

$c_3 = (x_1, x_3, \overline{x_4})$ $\quad x_2$

Logic engine

- variable poles & clause levels
- positive literal: flag on bottom
- negative literal: flag on top
- no literal: both flags



NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "**∃ weak UDCR for T?**" is NP-hard.

Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

Each clause: $\{0, 0, 1\}$ or $\{0, 1, 1\}$

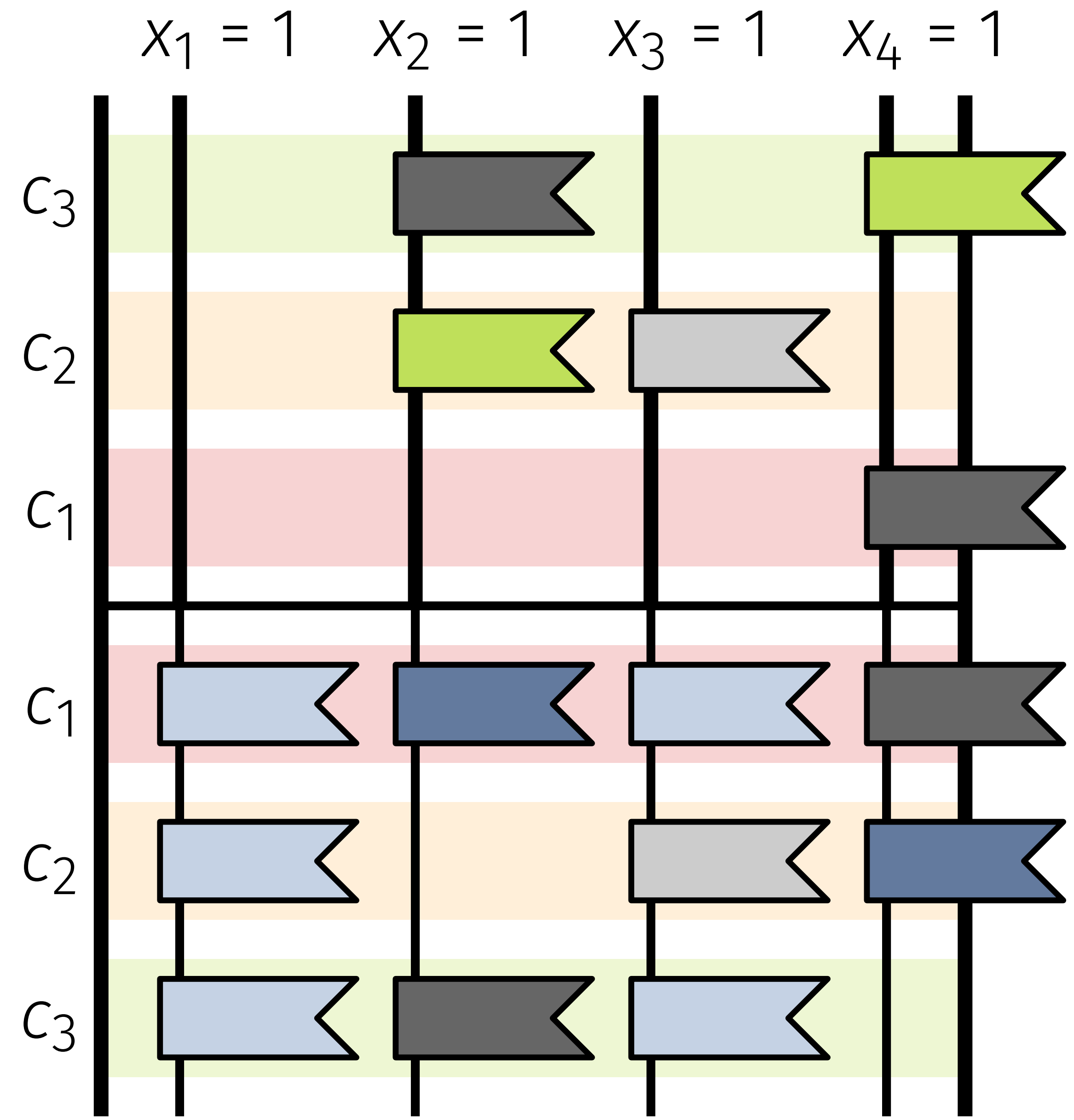
Example: $c_1 = (x_1, x_2, x_3)$ x_4

$c_2 = (x_1, \overline{x_2}, x_4)$ x_3

$c_3 = (x_1, x_3, \overline{x_4})$ x_2

Logic engine

- variable poles & clause levels
- positive literal: flag on bottom
- negative literal: flag on top
- no literal: both flags



NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "$\exists$ **weak UDCR for** T**?**" is NP-hard.

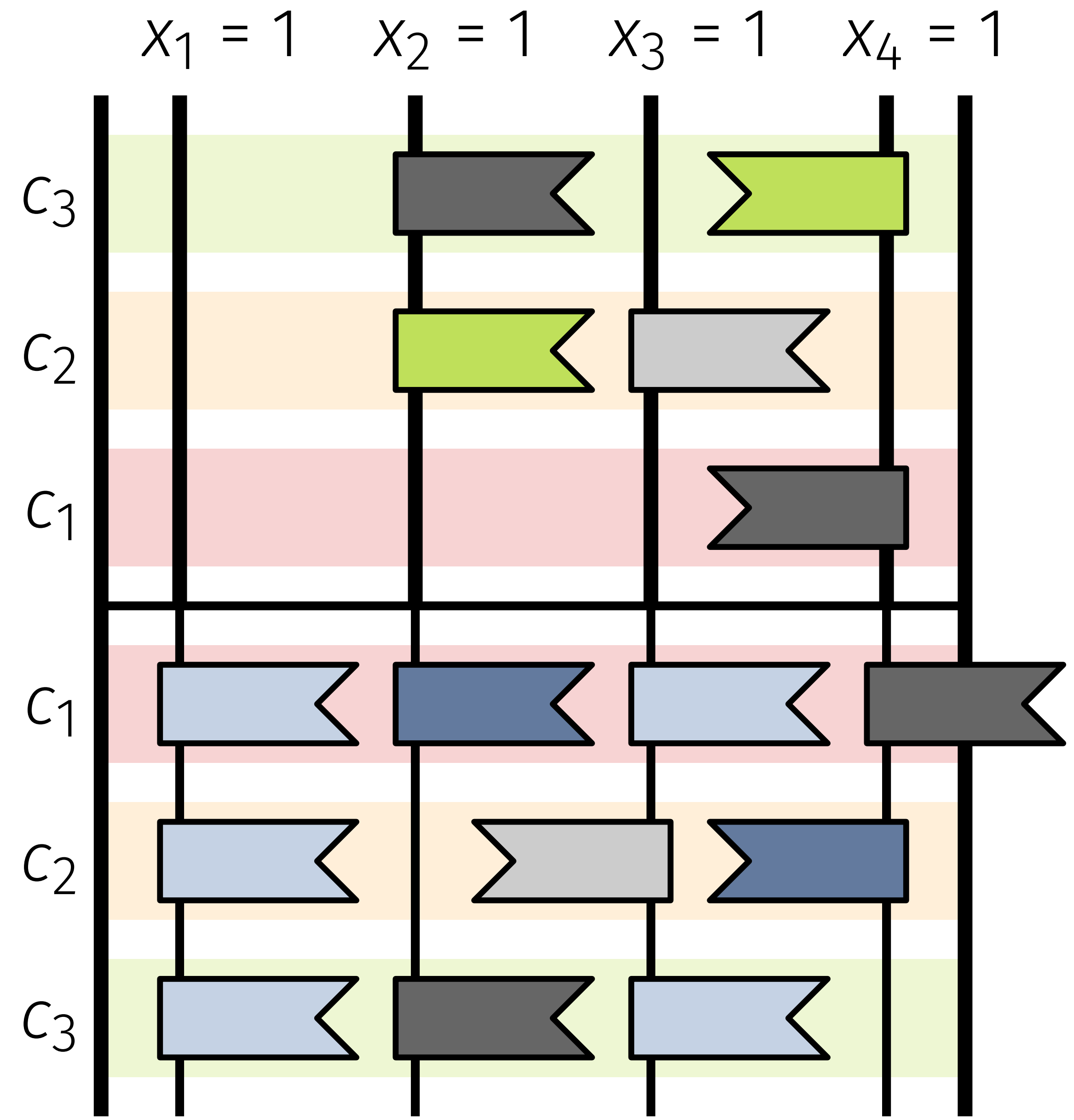
Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

Each clause: $\{0, 0, 1\}$ or $\{0, 1, 1\}$

Example: $c_1 = (x_1, x_2, x_3)$

$c_2 = (x_1, \overline{x_2}, x_4)$

$c_3 = (x_1, x_3, \overline{x_4})$



Logic engine

- variable poles & clause levels
- positive literal: flag on bottom
- negative literal: flag on top
- no literal: both flags

NP-hardness of Recognizing Trees

Theorem: Given a tree T, answering "∃ **weak UDCR for** T**?**" is NP-hard.

Proof: Not-All-Equal-3SAT $\leq_P$ Tree-WUDCR, via *logic engine*.

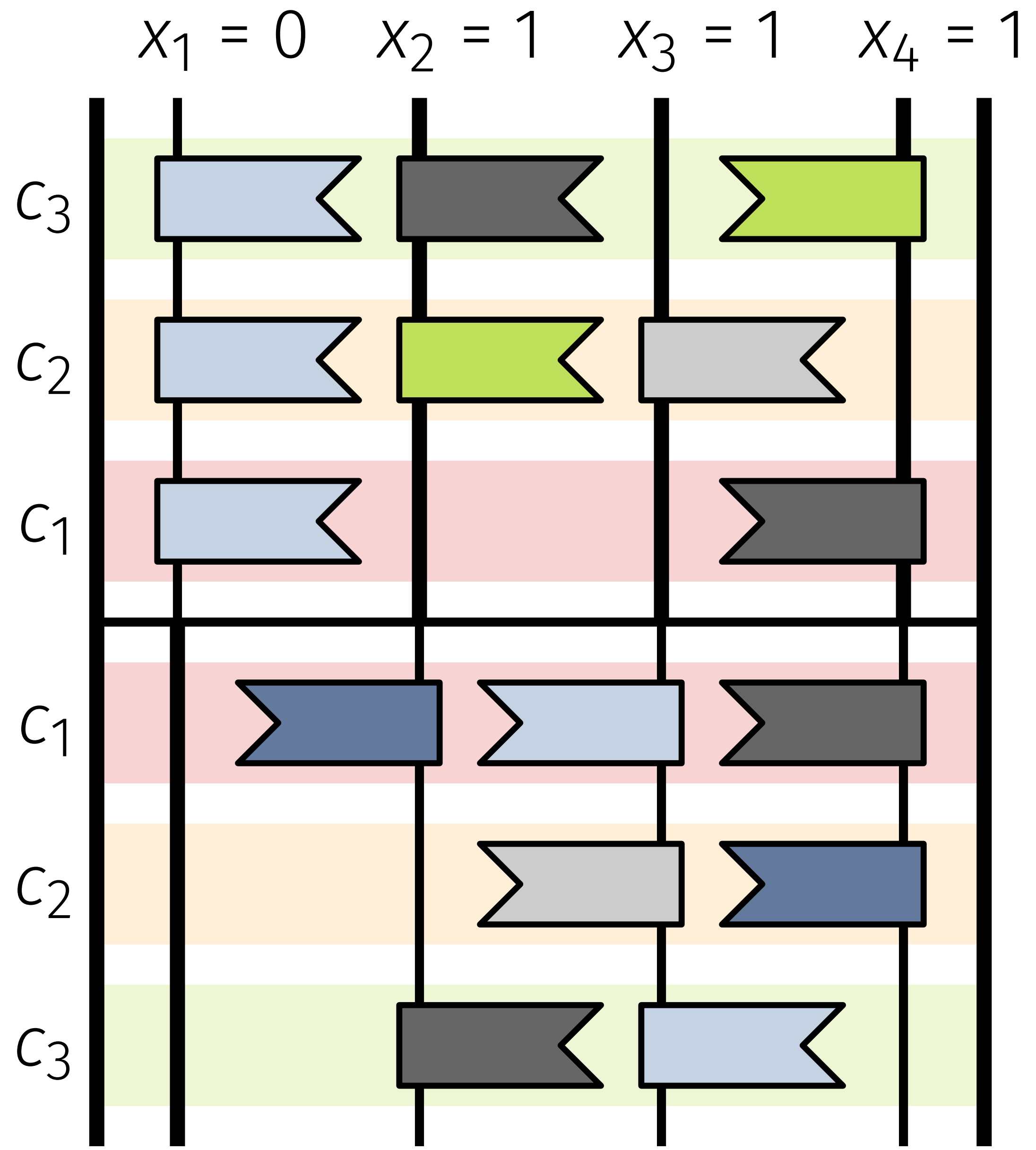
Each clause: $\{0, 0, 1\}$ or $\{0, 1, 1\}$

Example:

$c_1 = (x_1, x_2, x_3)$

$c_2 = (x_1, \overline{x_2}, x_4)$

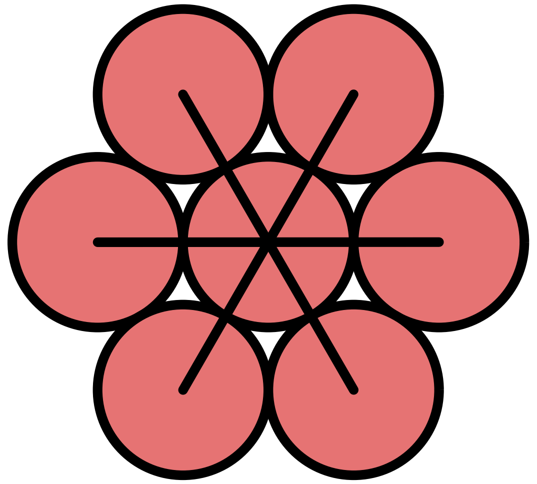
$c_3 = (x_1, x_3, \overline{x_4})$



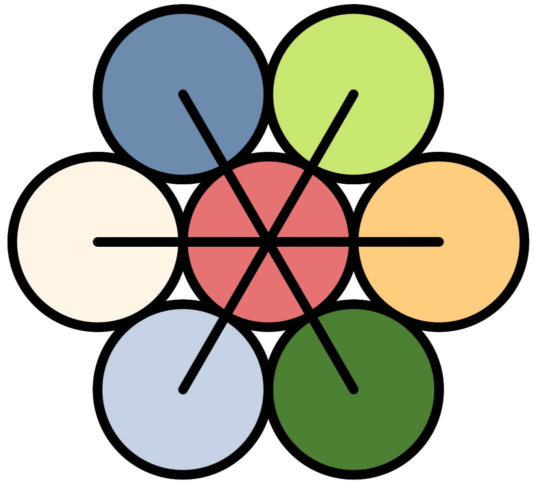
Logic engine

- variable poles & clause levels
- positive literal: flag on bottom
- negative literal: flag on top
- no literal: both flags

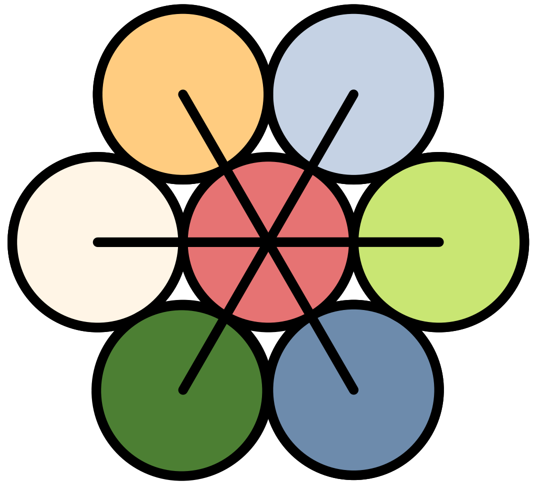
Modeling Line Segments



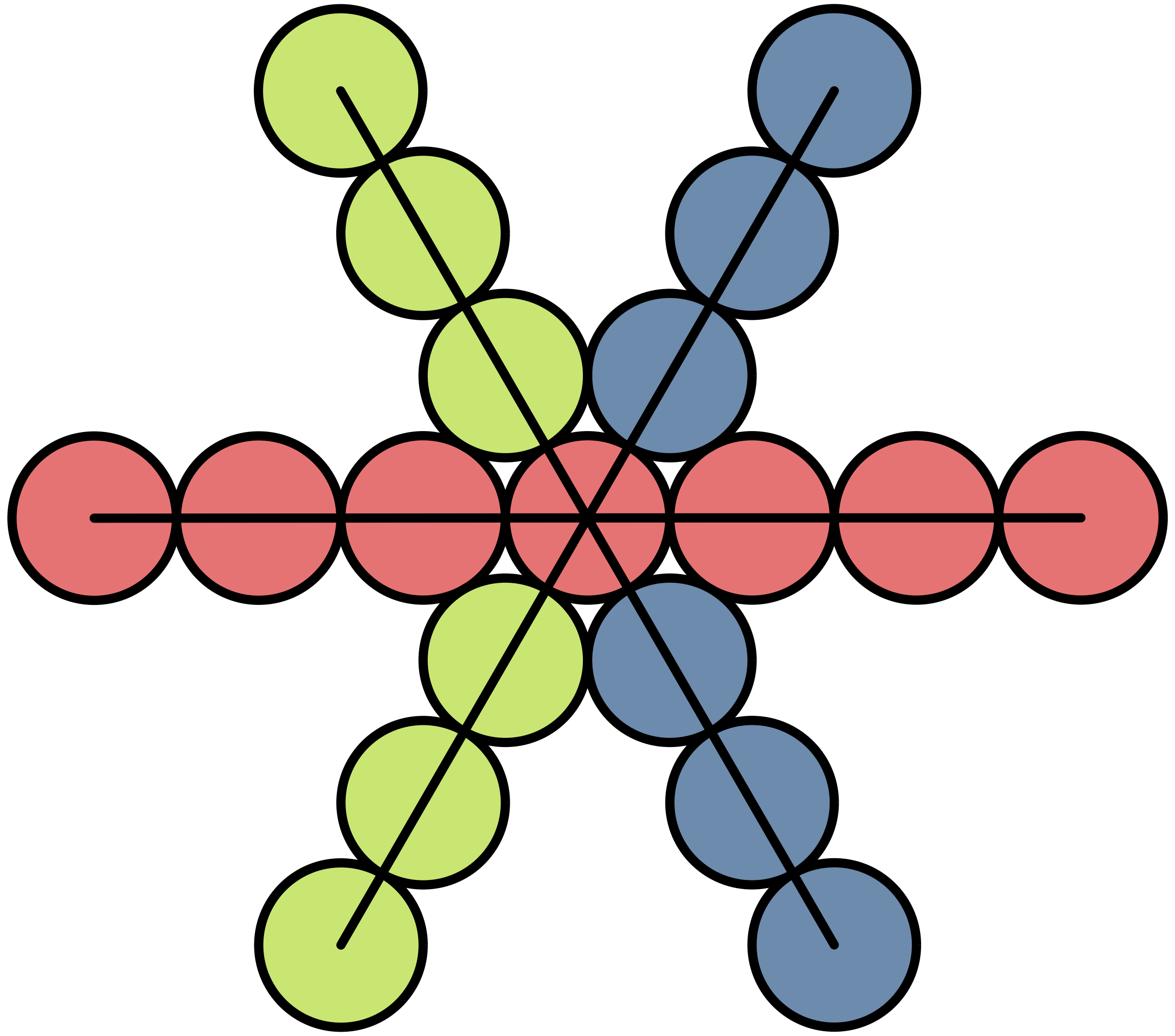
Modeling Line Segments



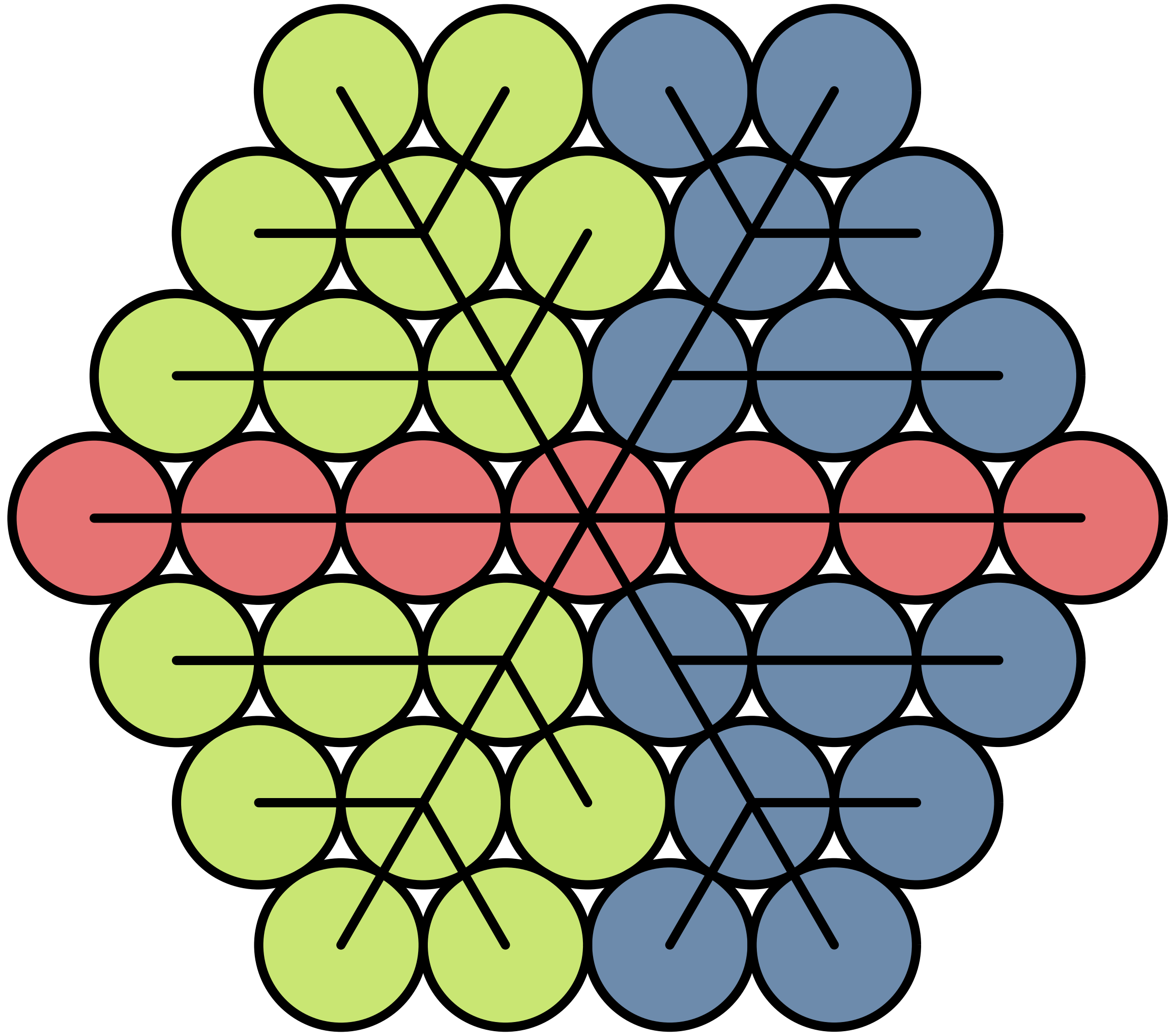
Modeling Line Segments



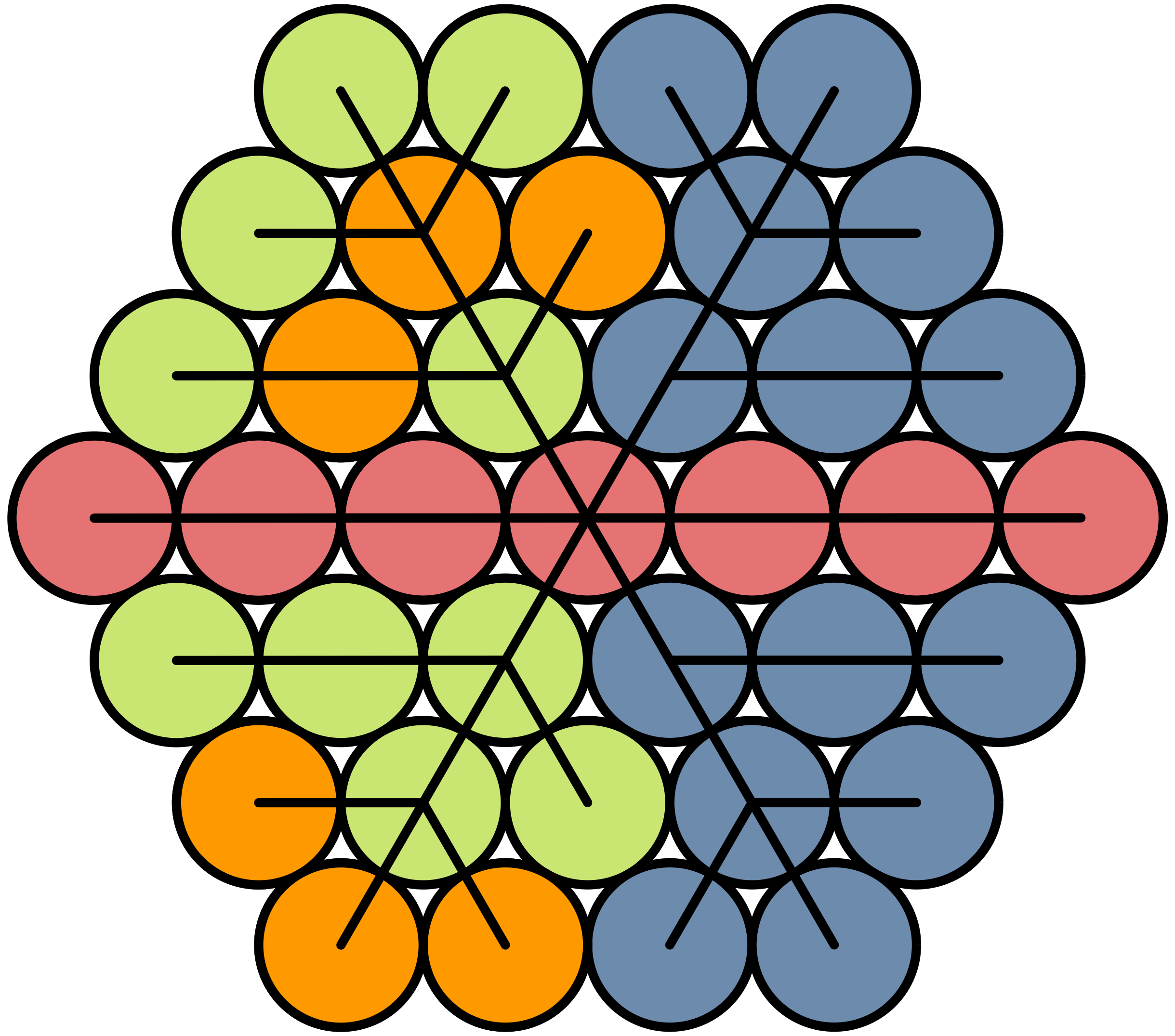
Modeling Line Segments



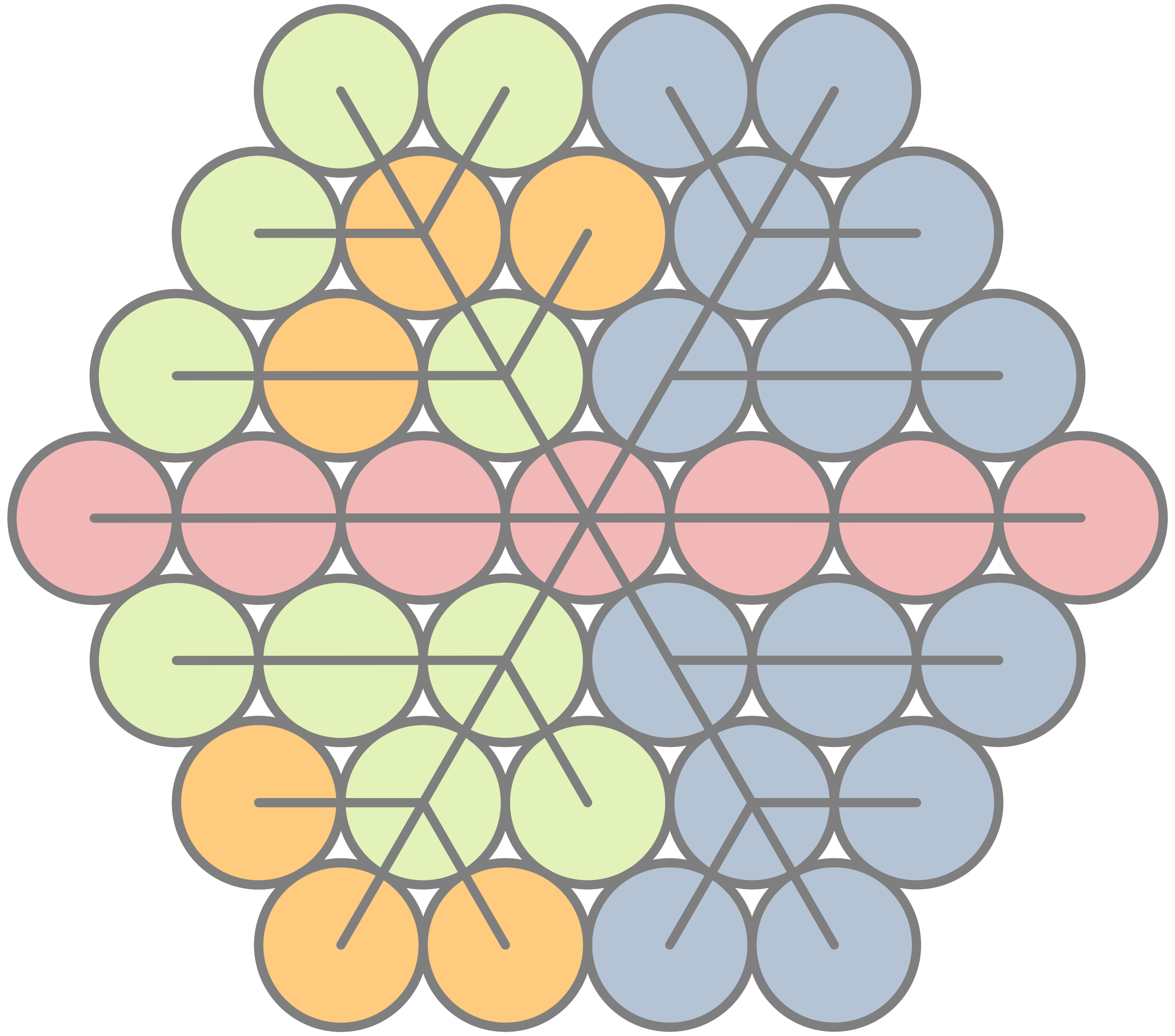
Modeling Line Segments

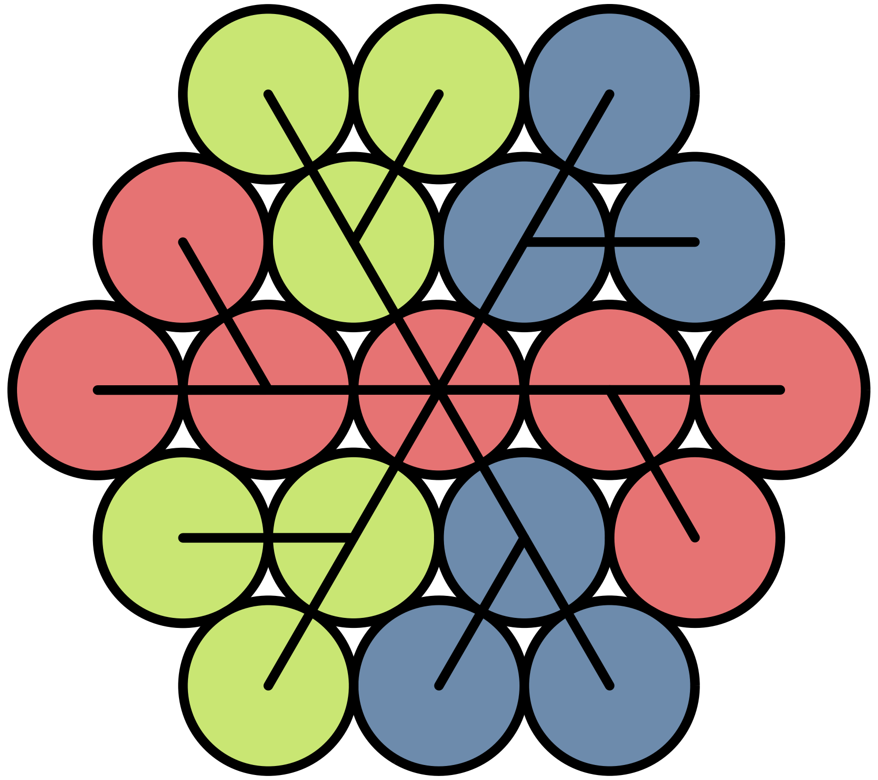


Modeling Line Segments

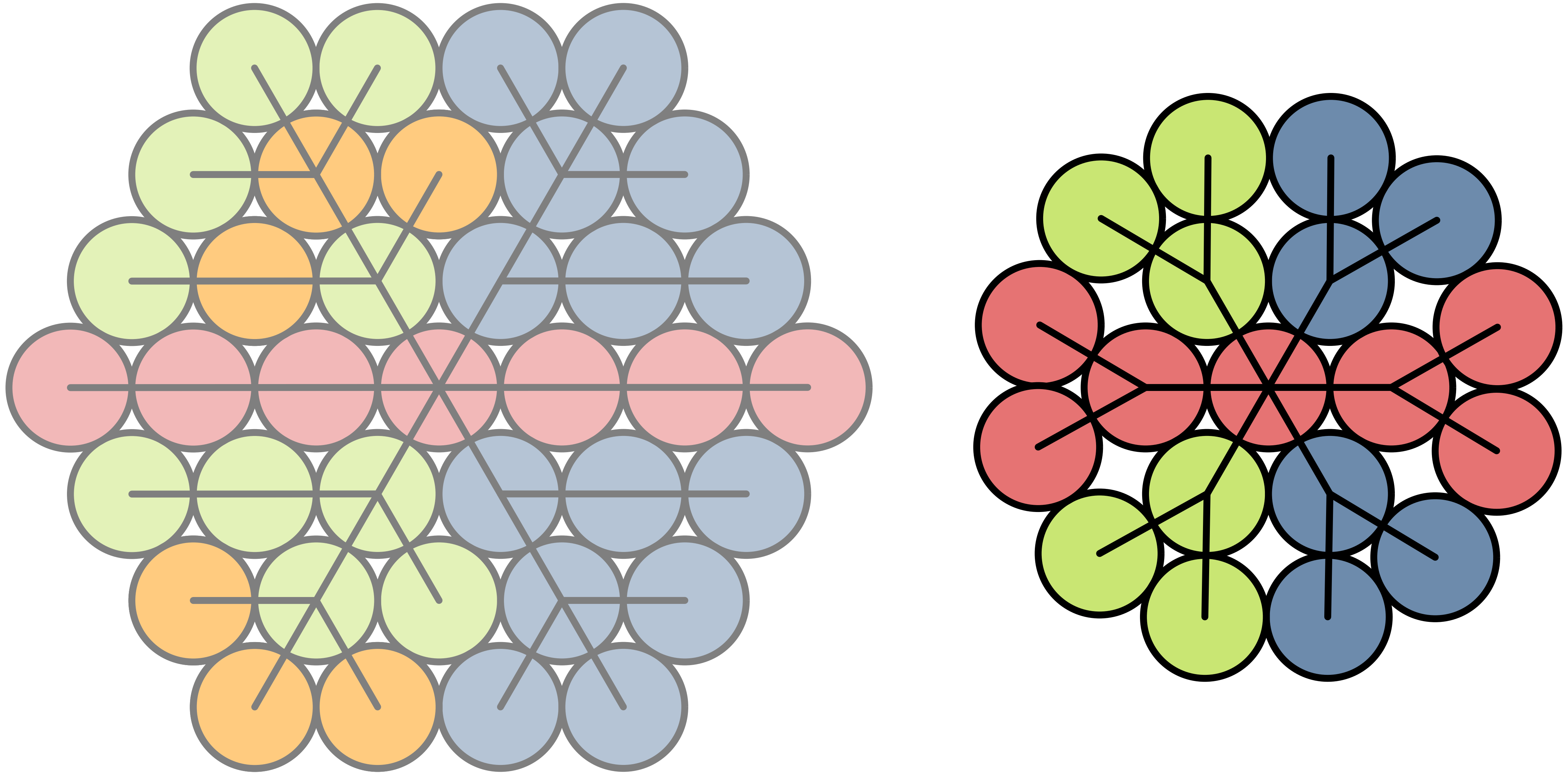


Modeling Line Segments

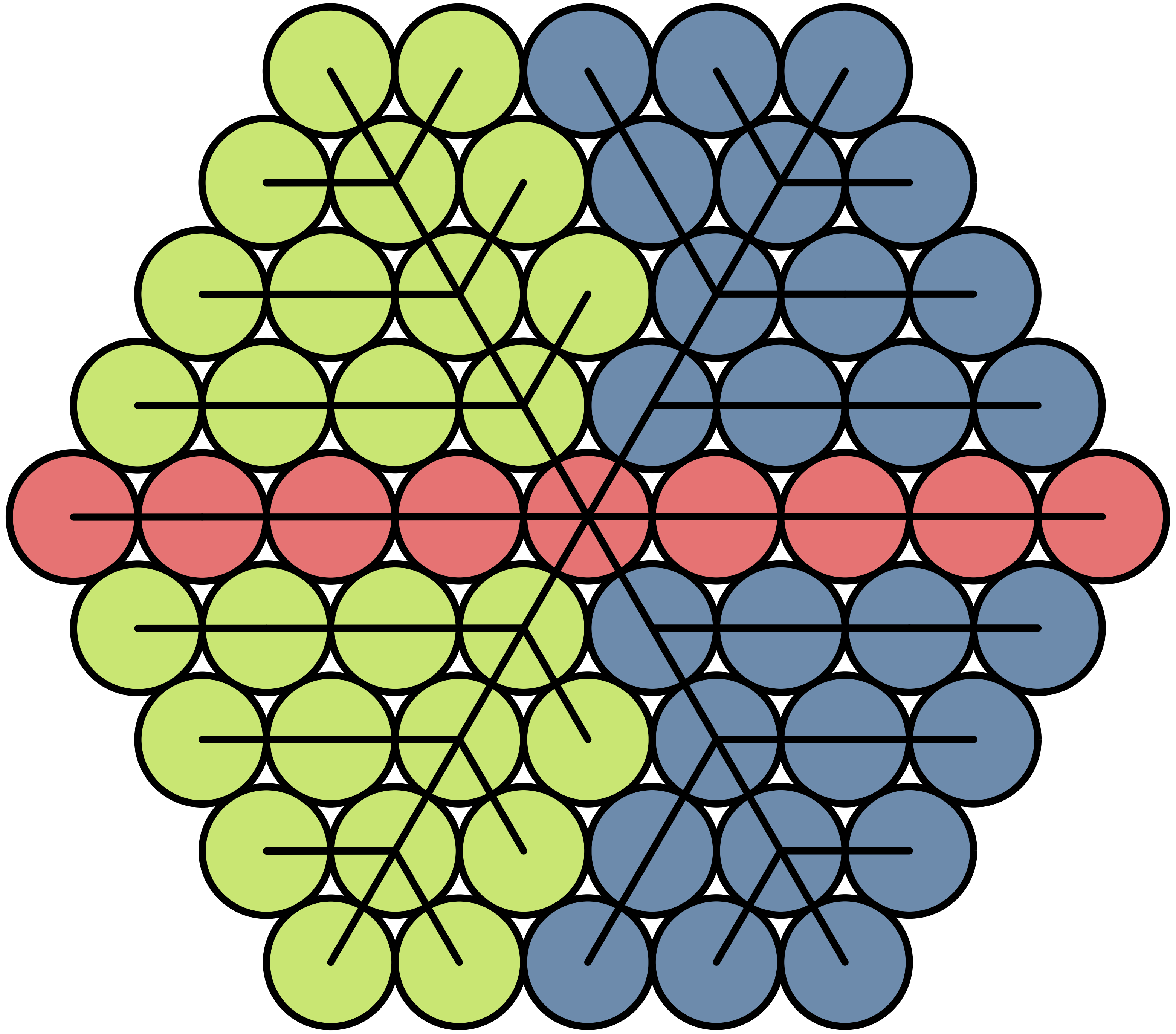




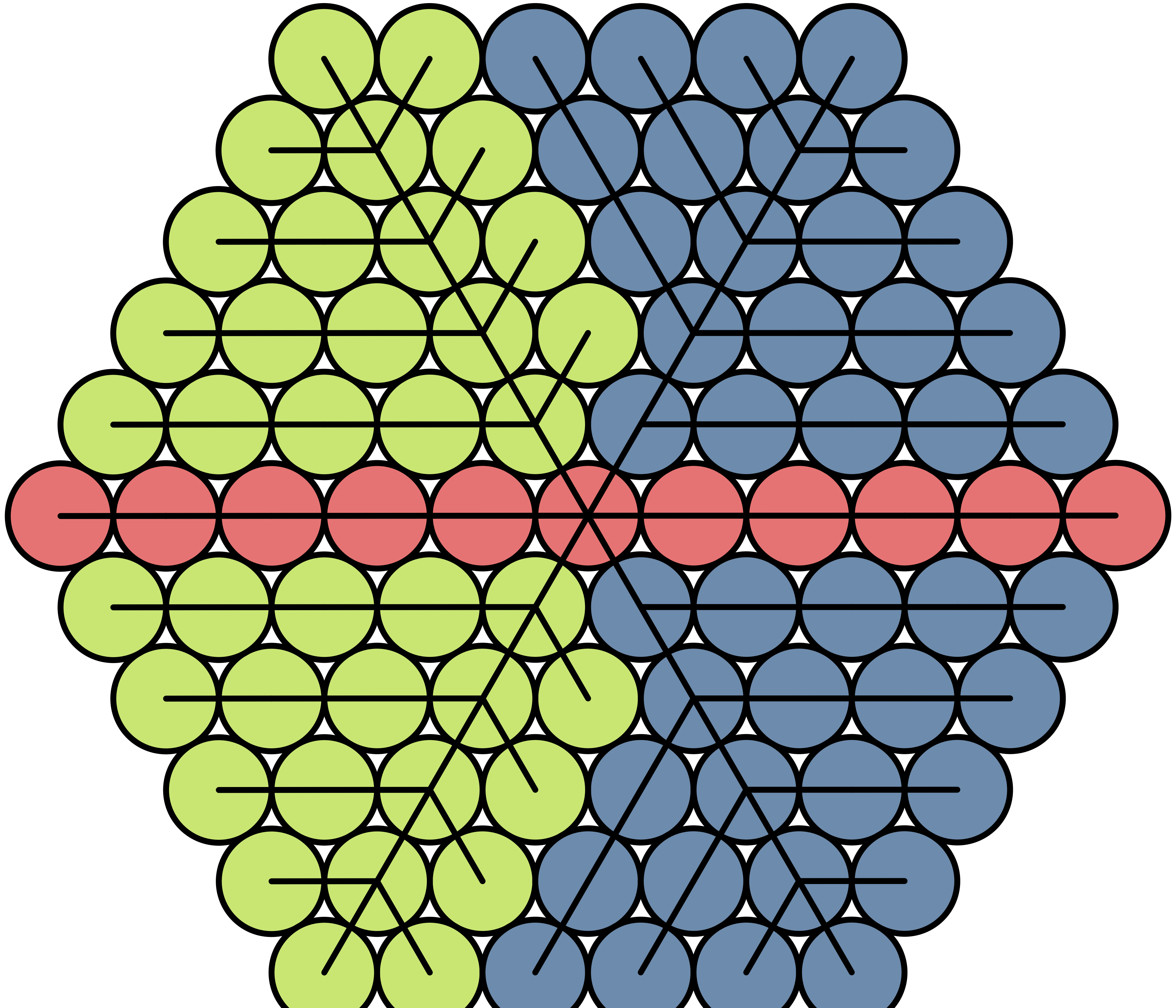
Modeling Line Segments



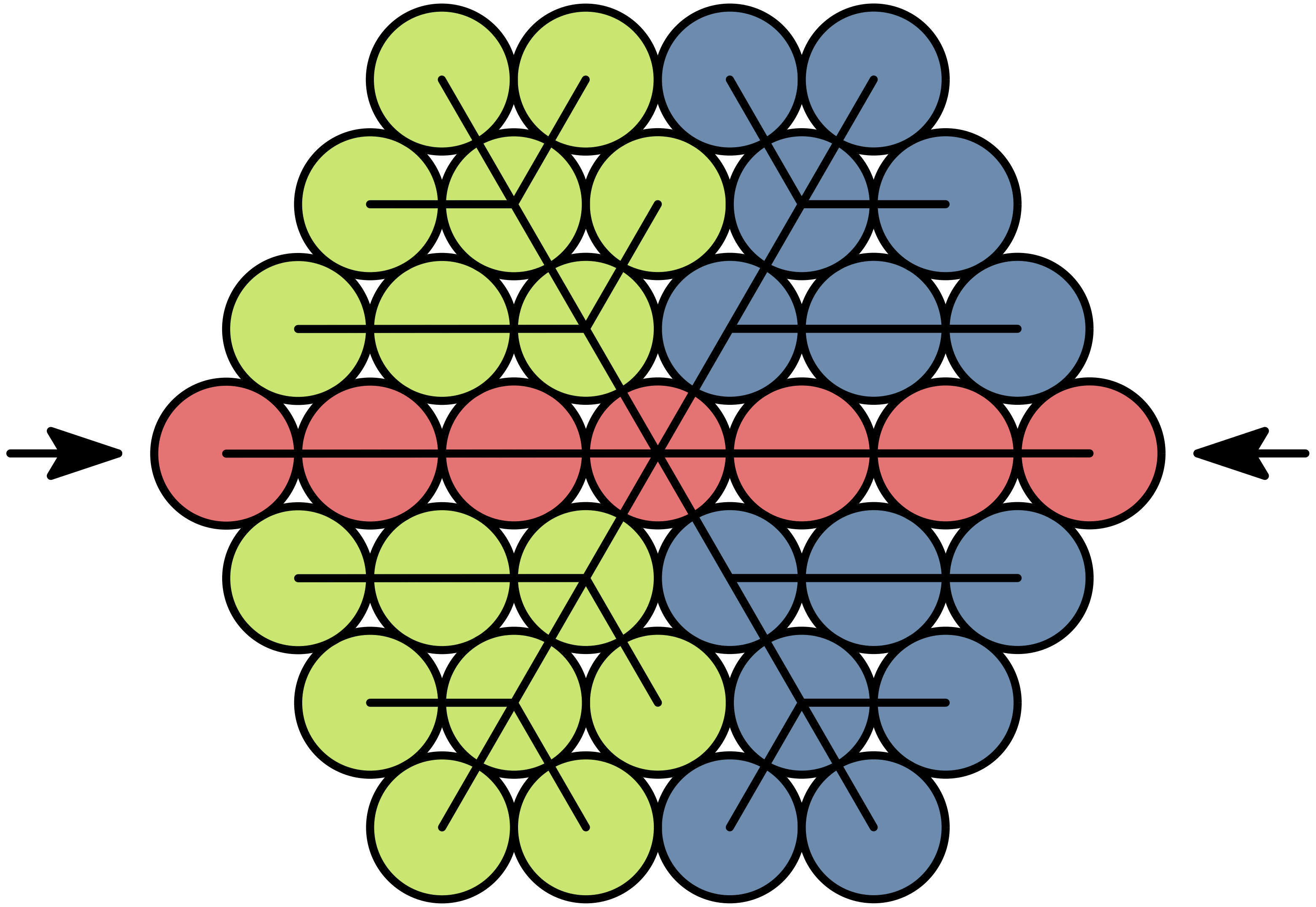
Modeling Line Segments



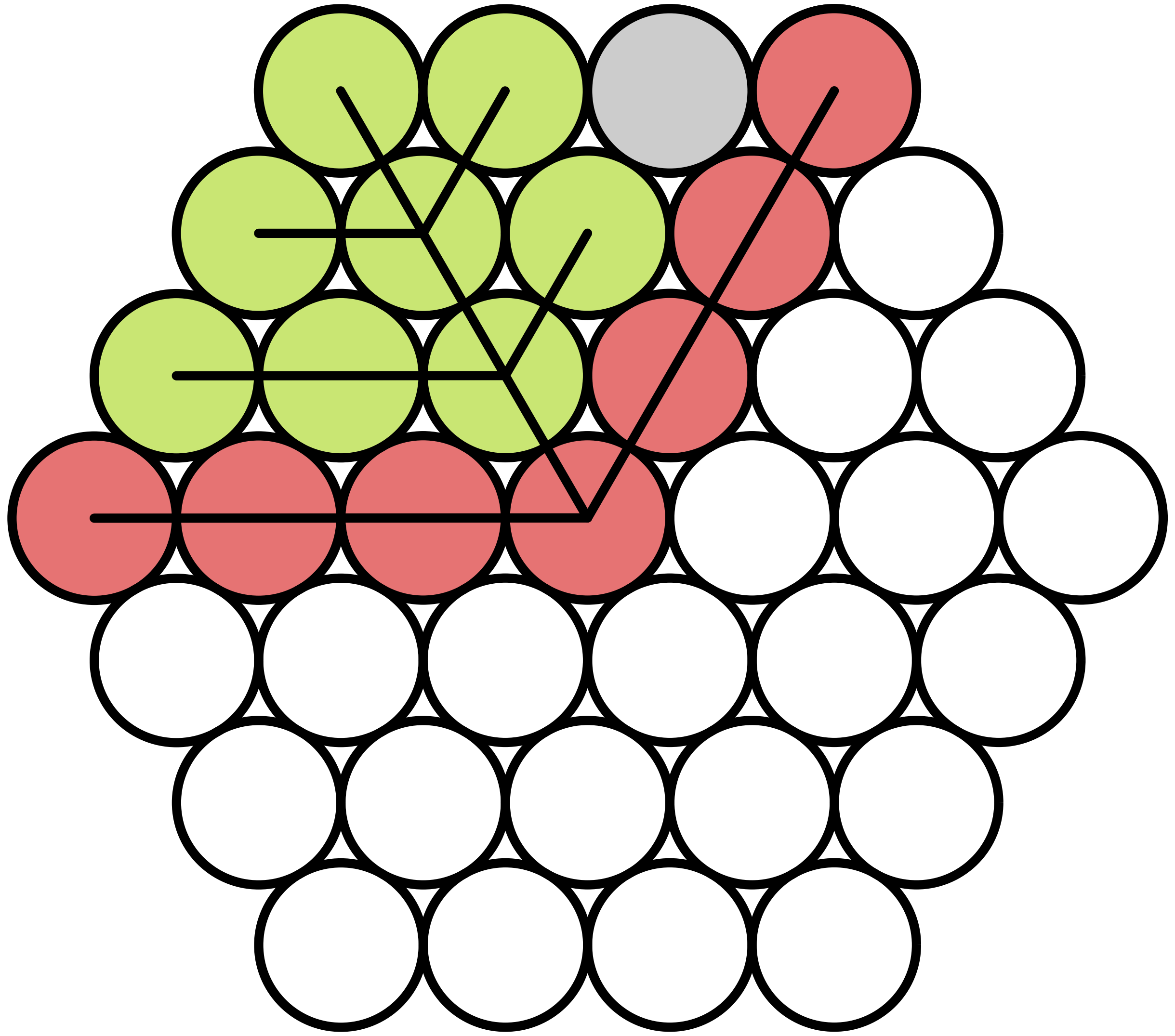
Modeling Line Segments



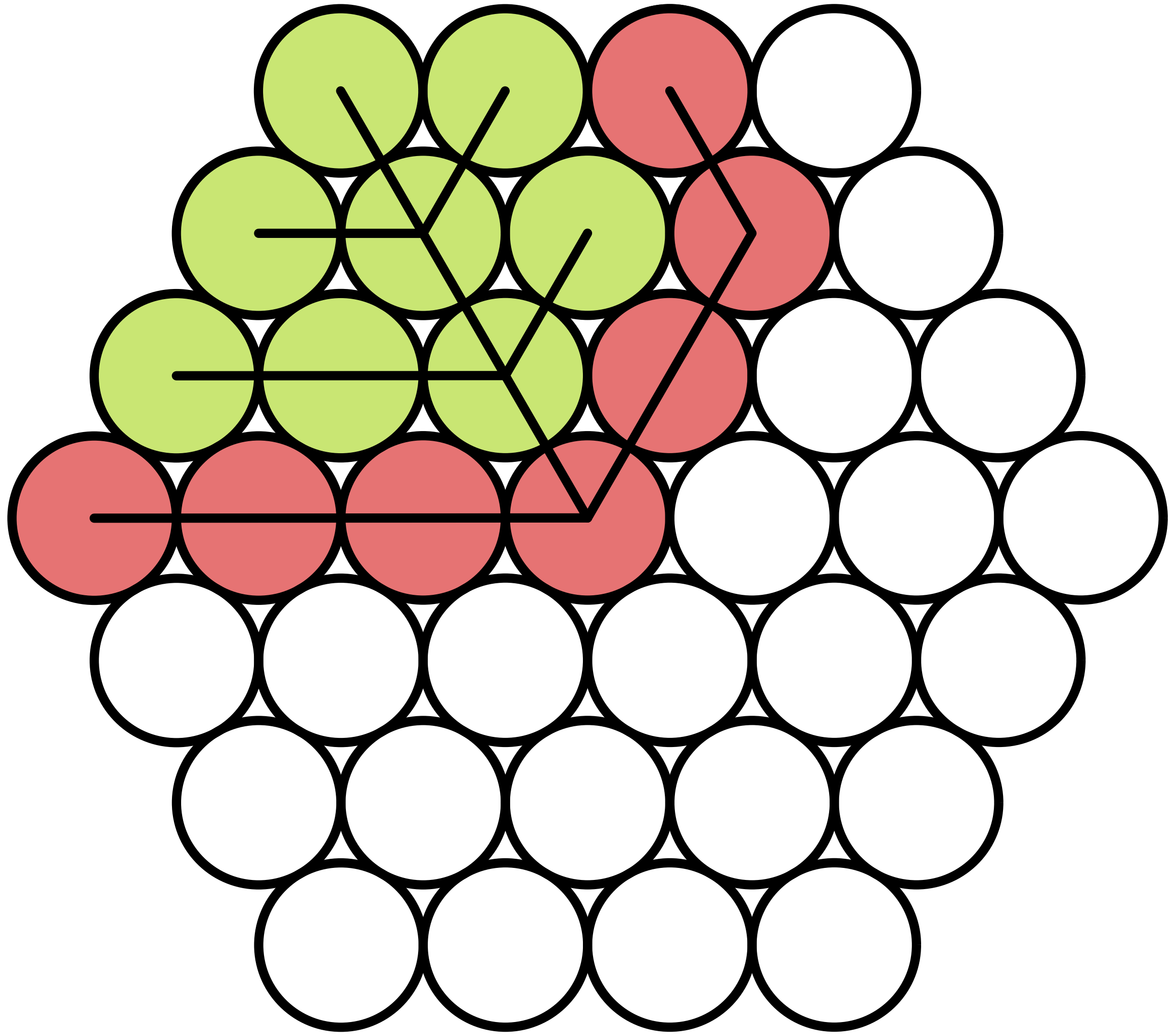
Modeling Line Segments



Modeling Line Segments

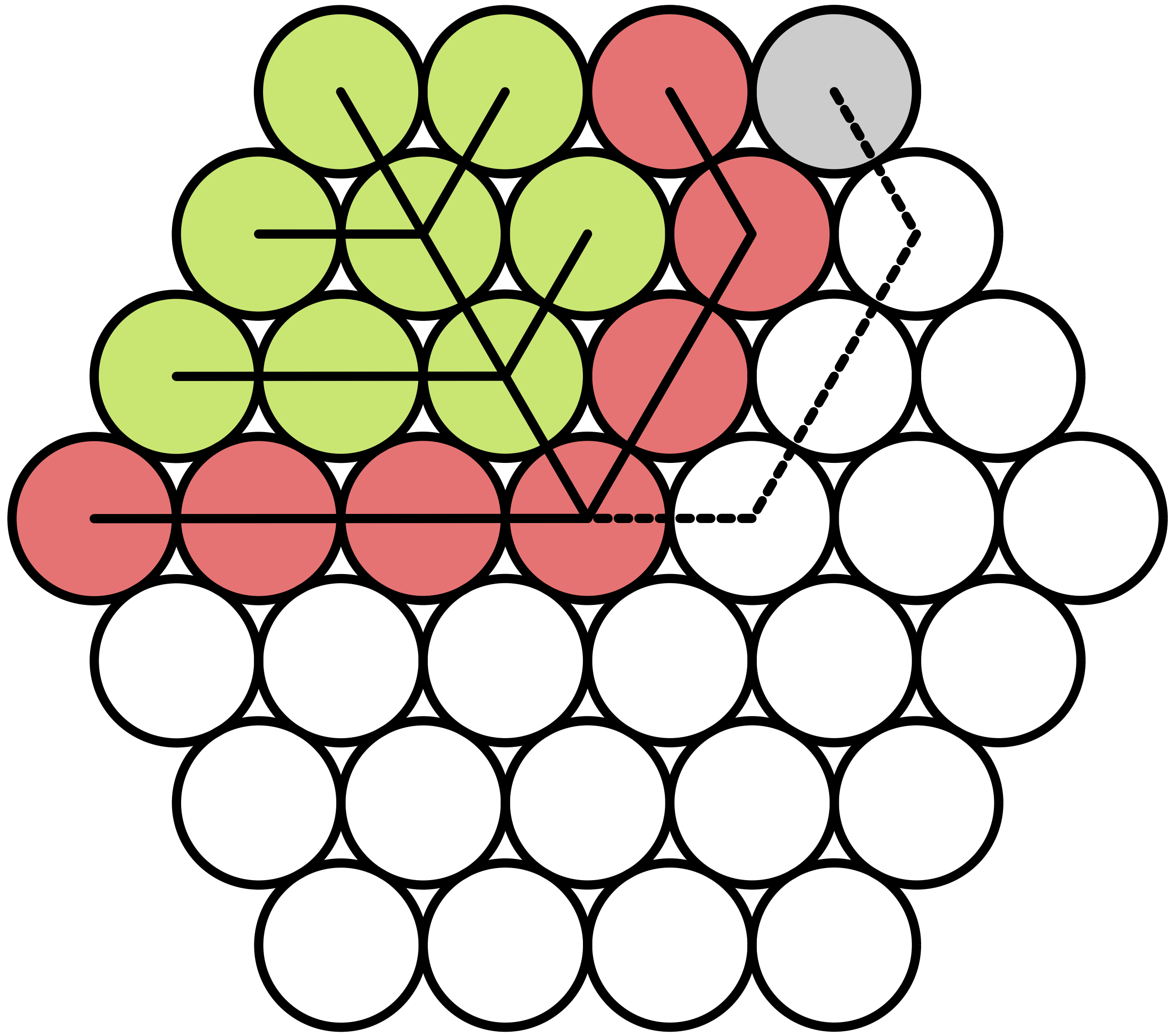


Modeling Line Segments

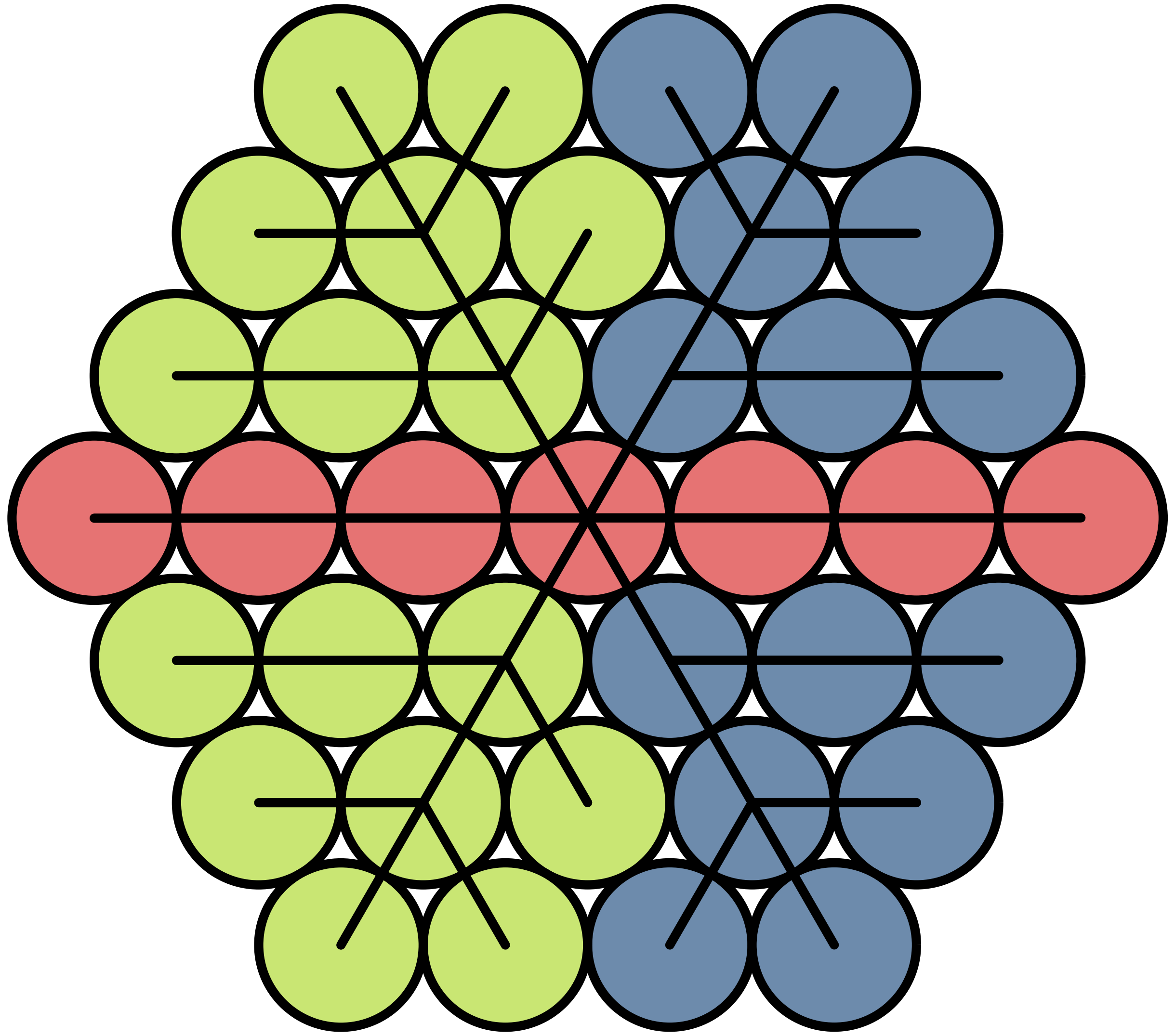


Modeling Line Segments

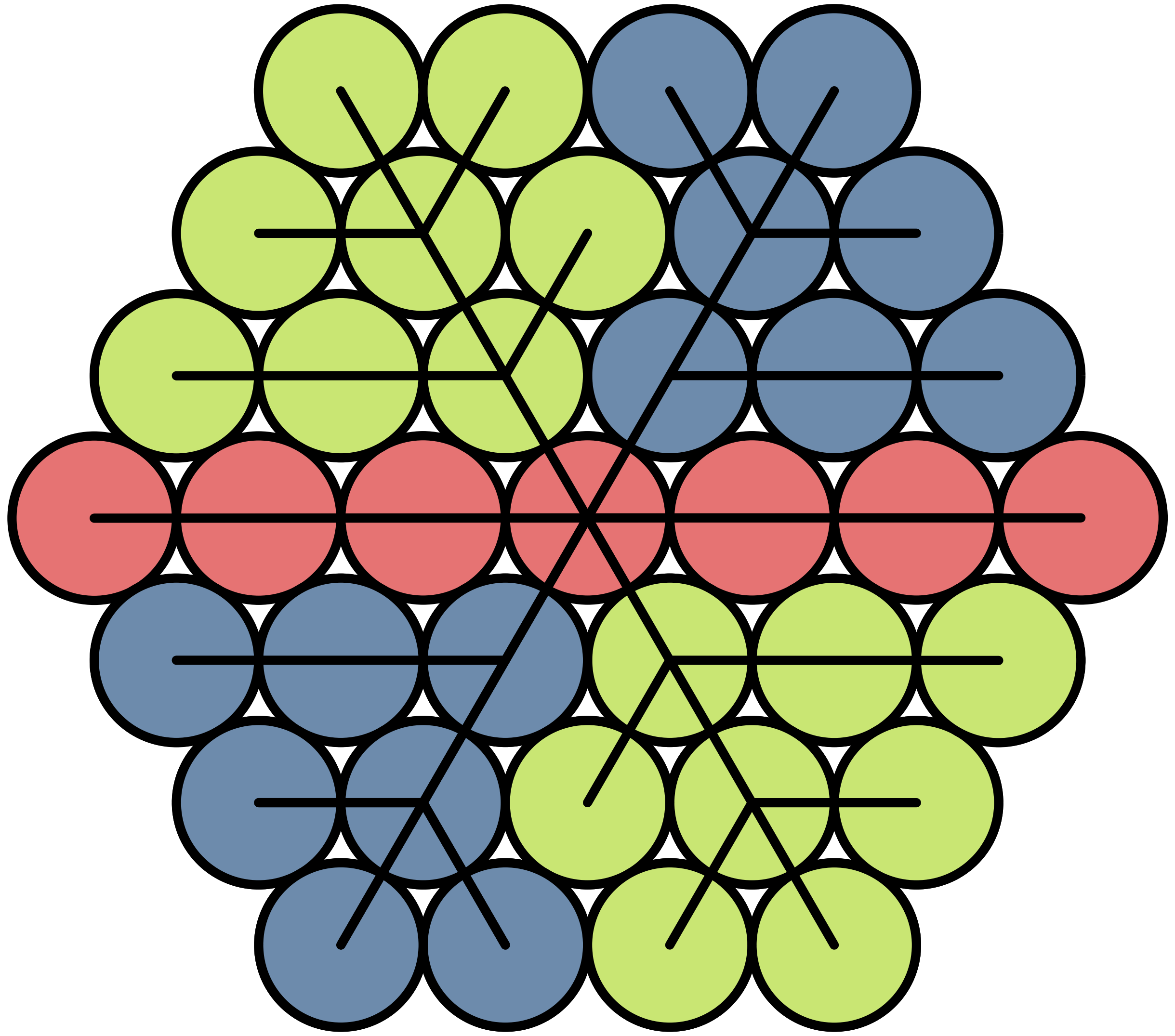
Freie Universität Berlin



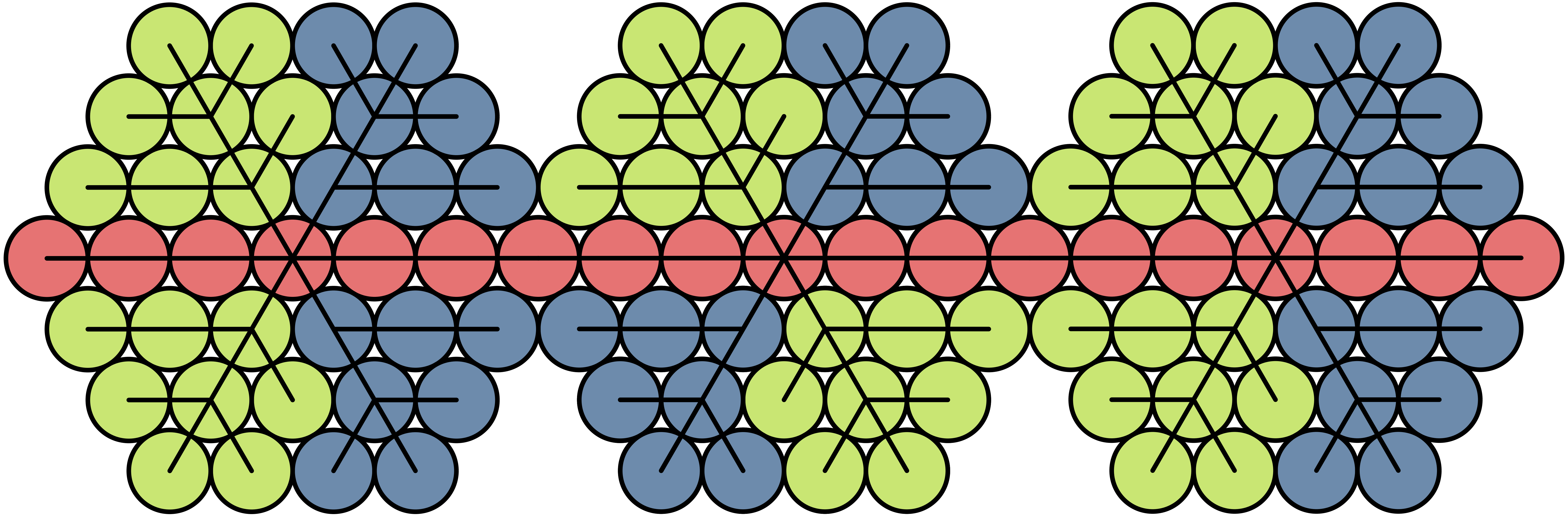
Modeling Line Segments



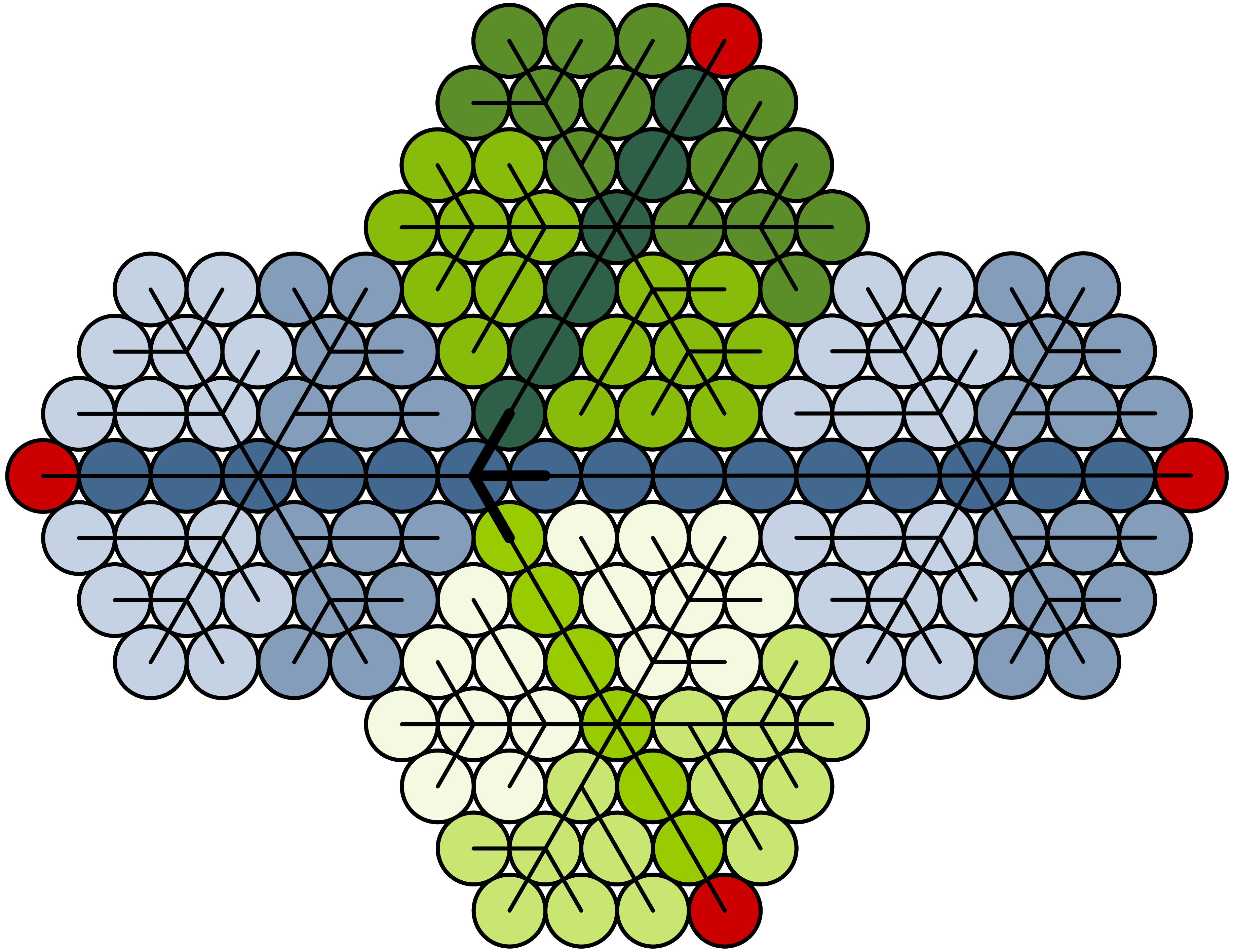
Modeling Line Segments



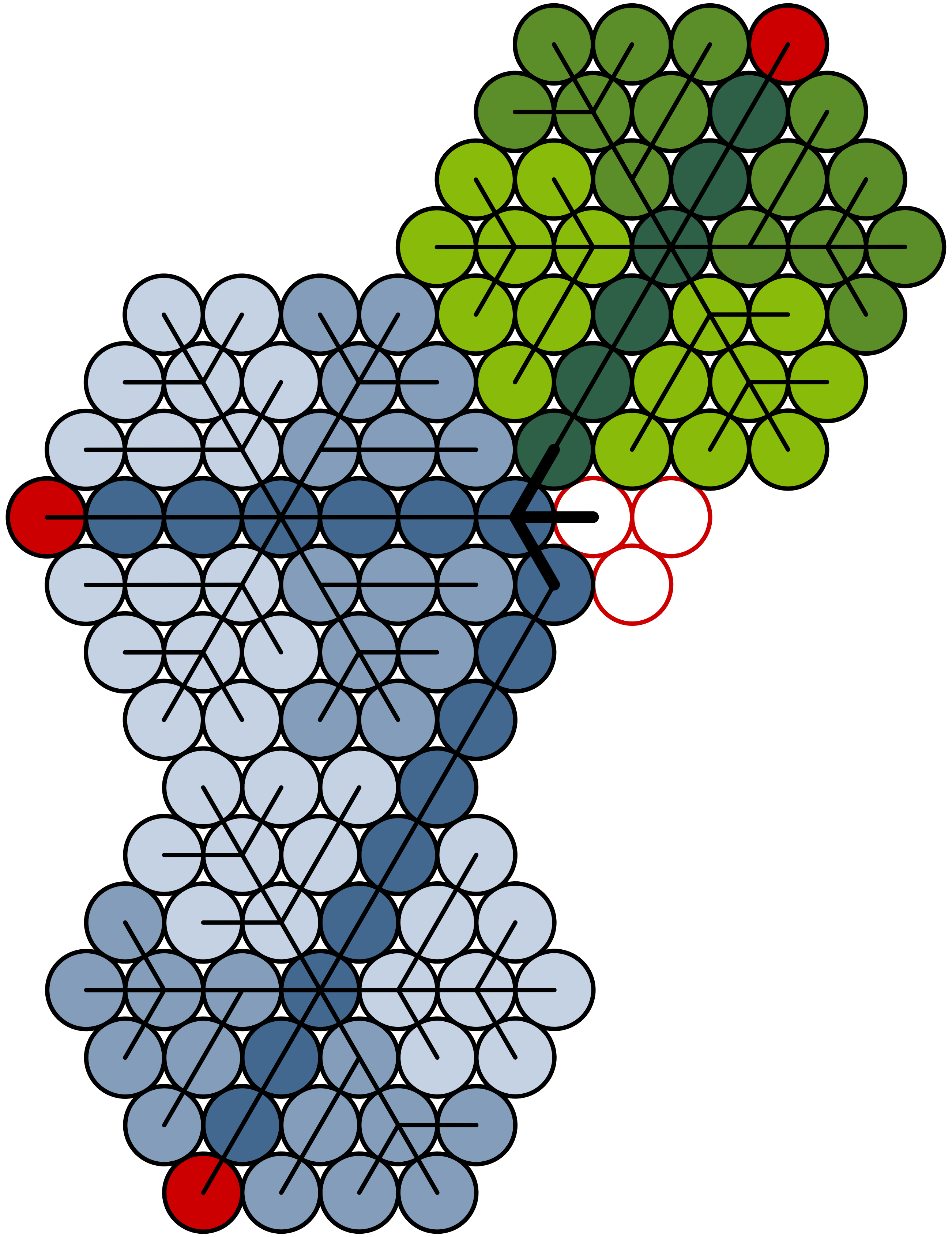
Modeling Line Segments



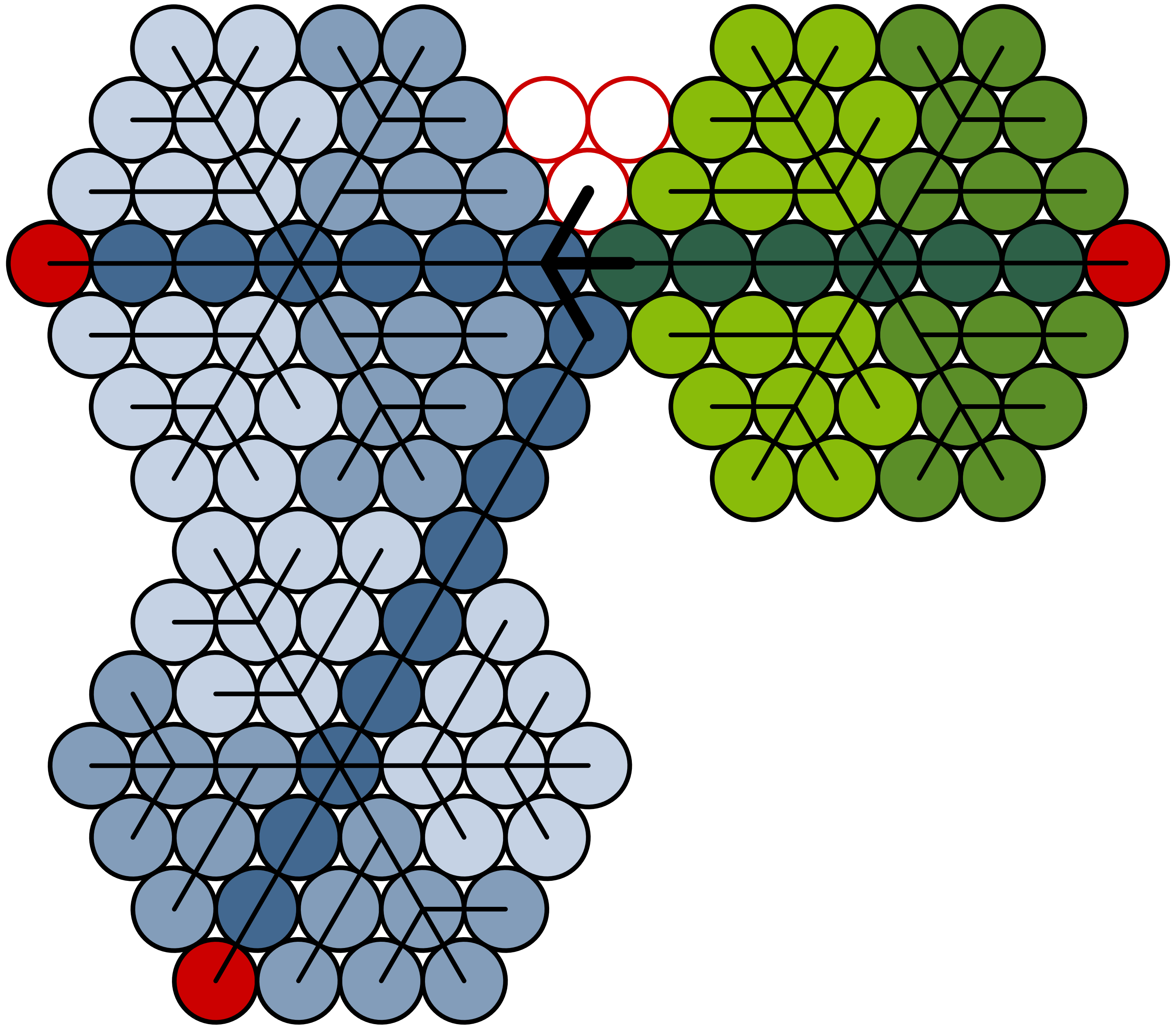
A Branching Gadget



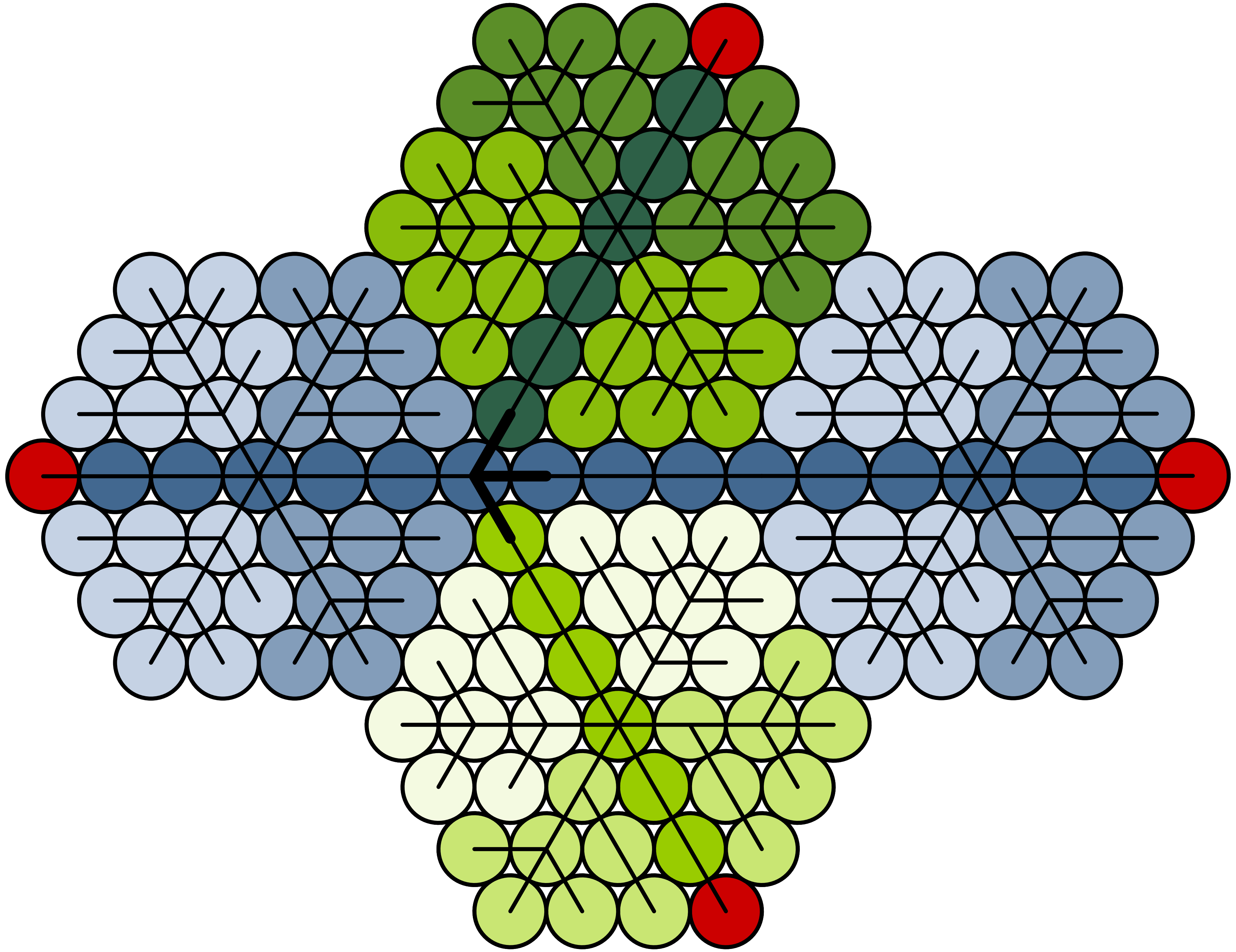
A Branching Gadget



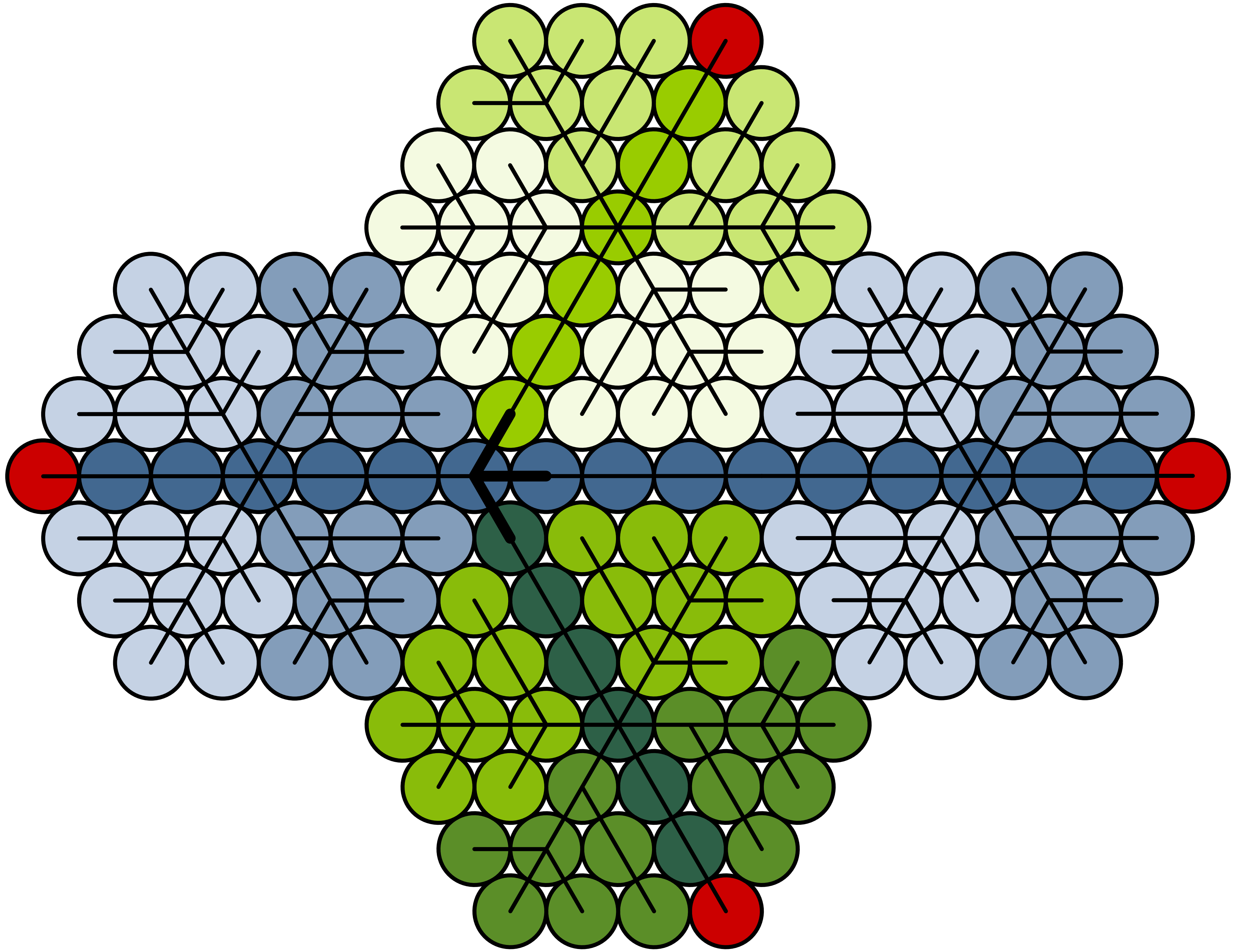
A Branching Gadget



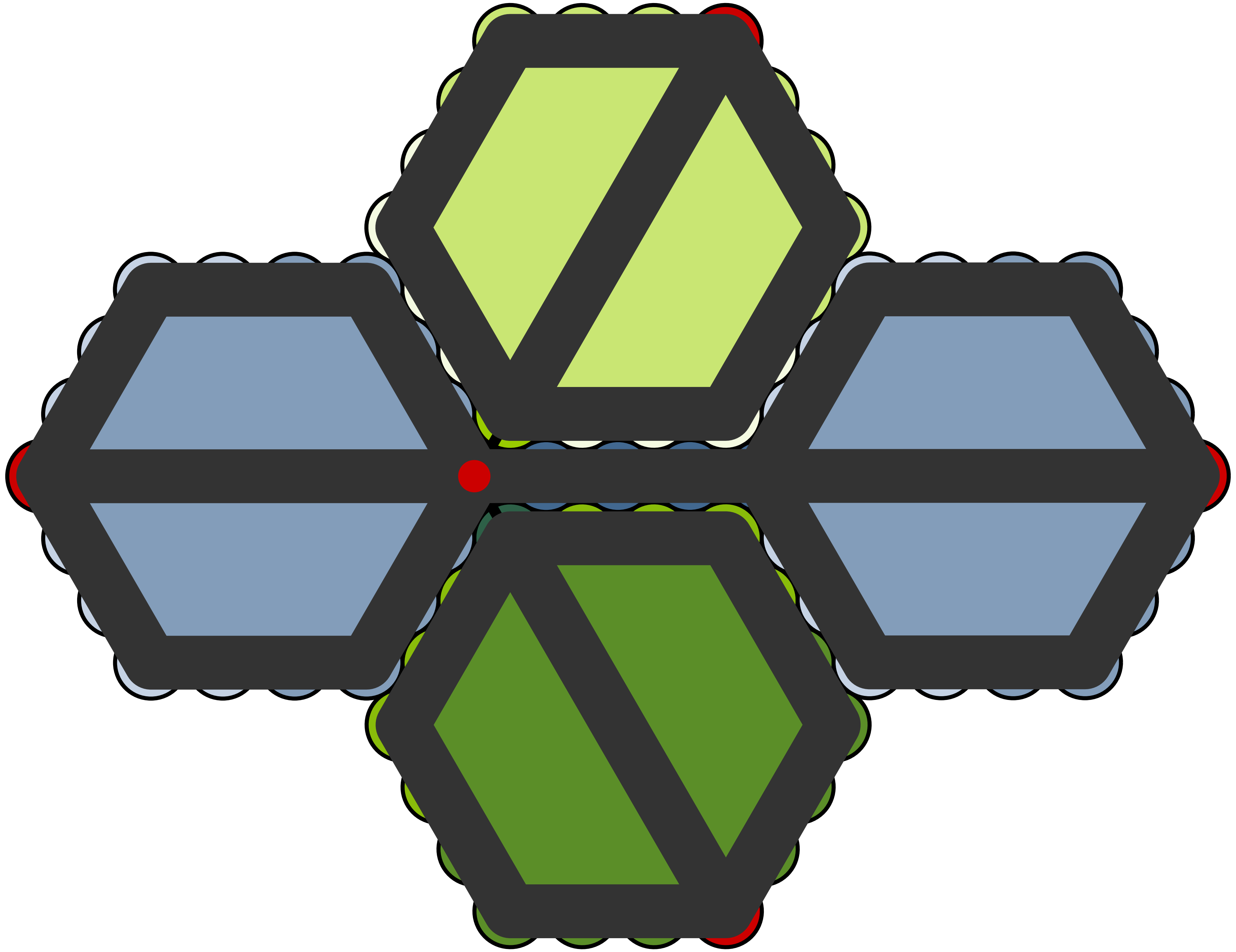
A Branching Gadget



A Branching Gadget

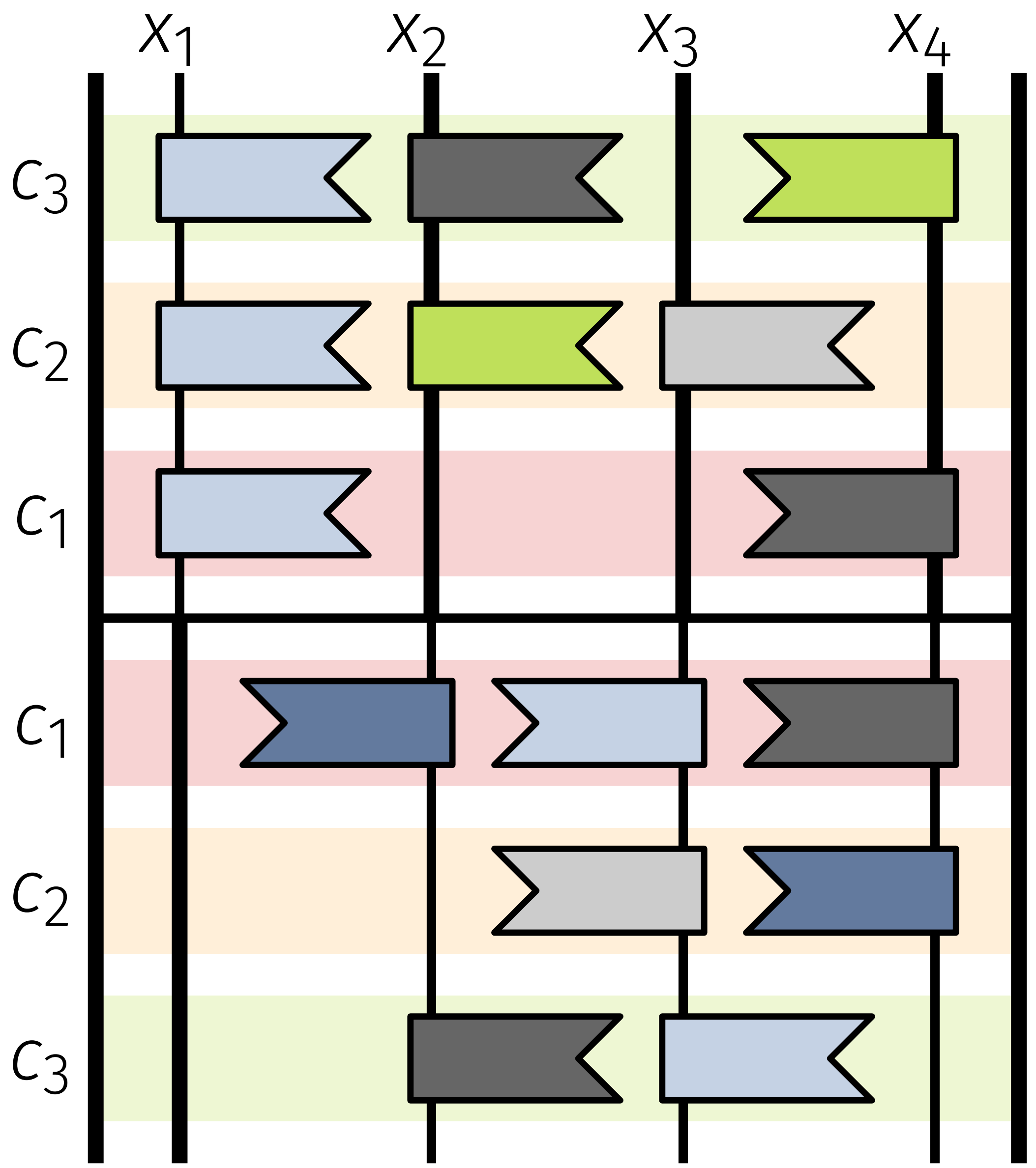


A Branching Gadget

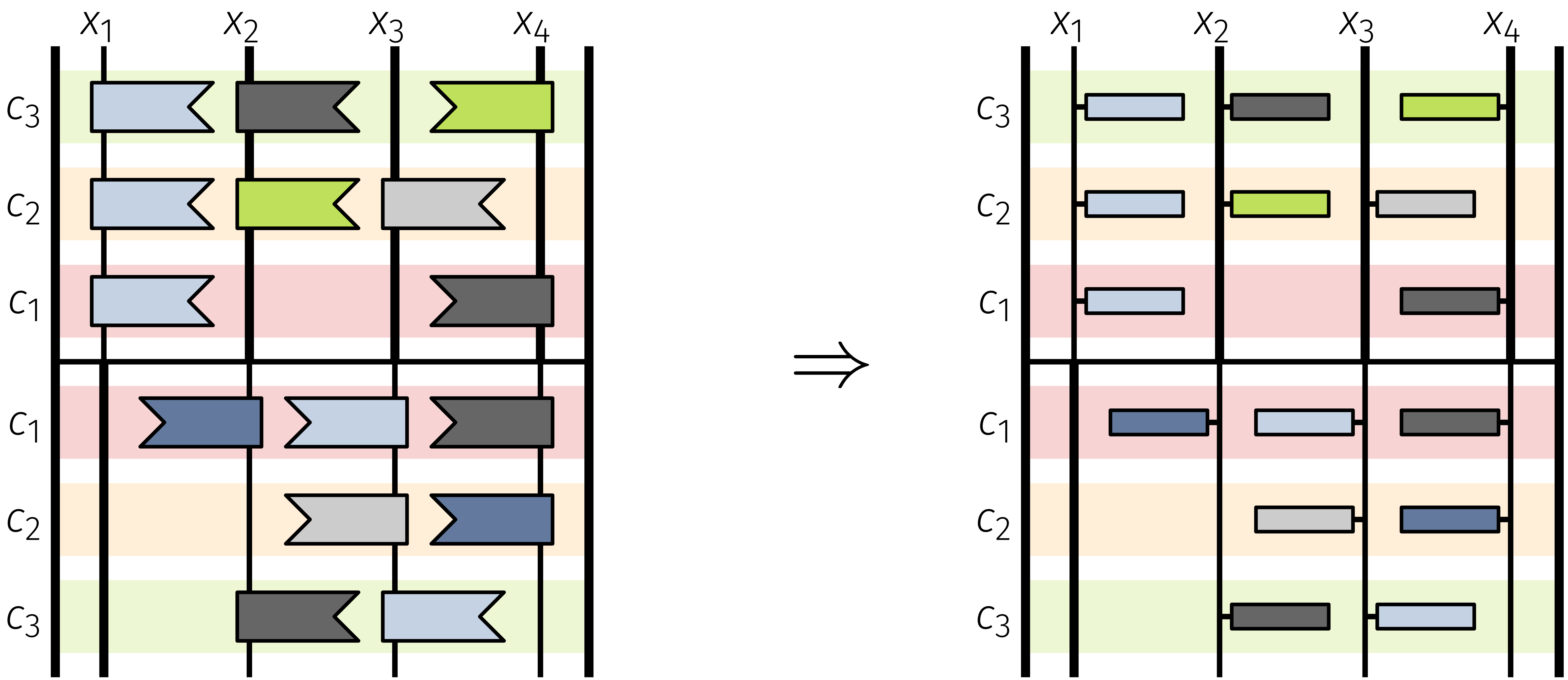


A Branching Gadget

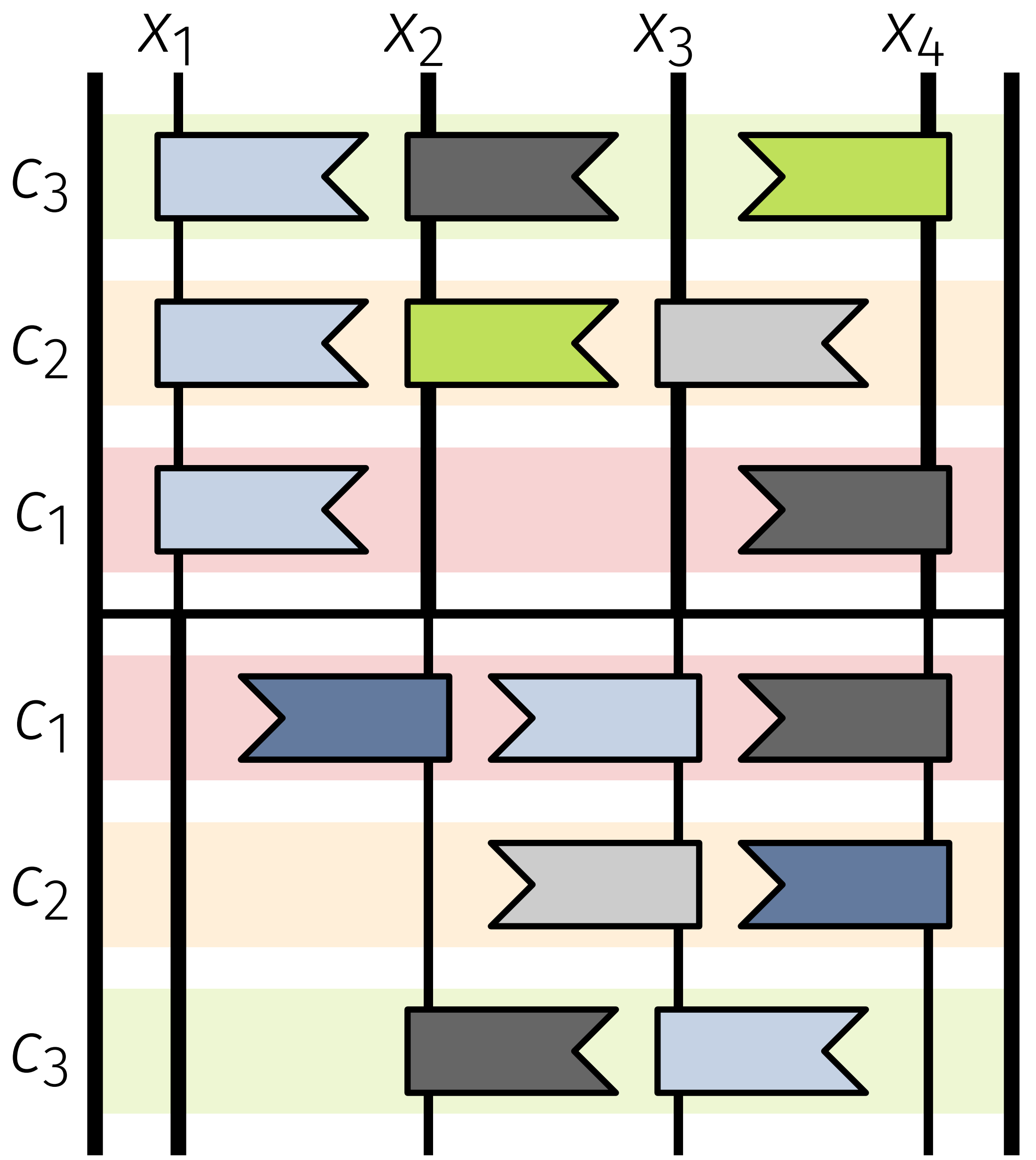
60° Logic Engine



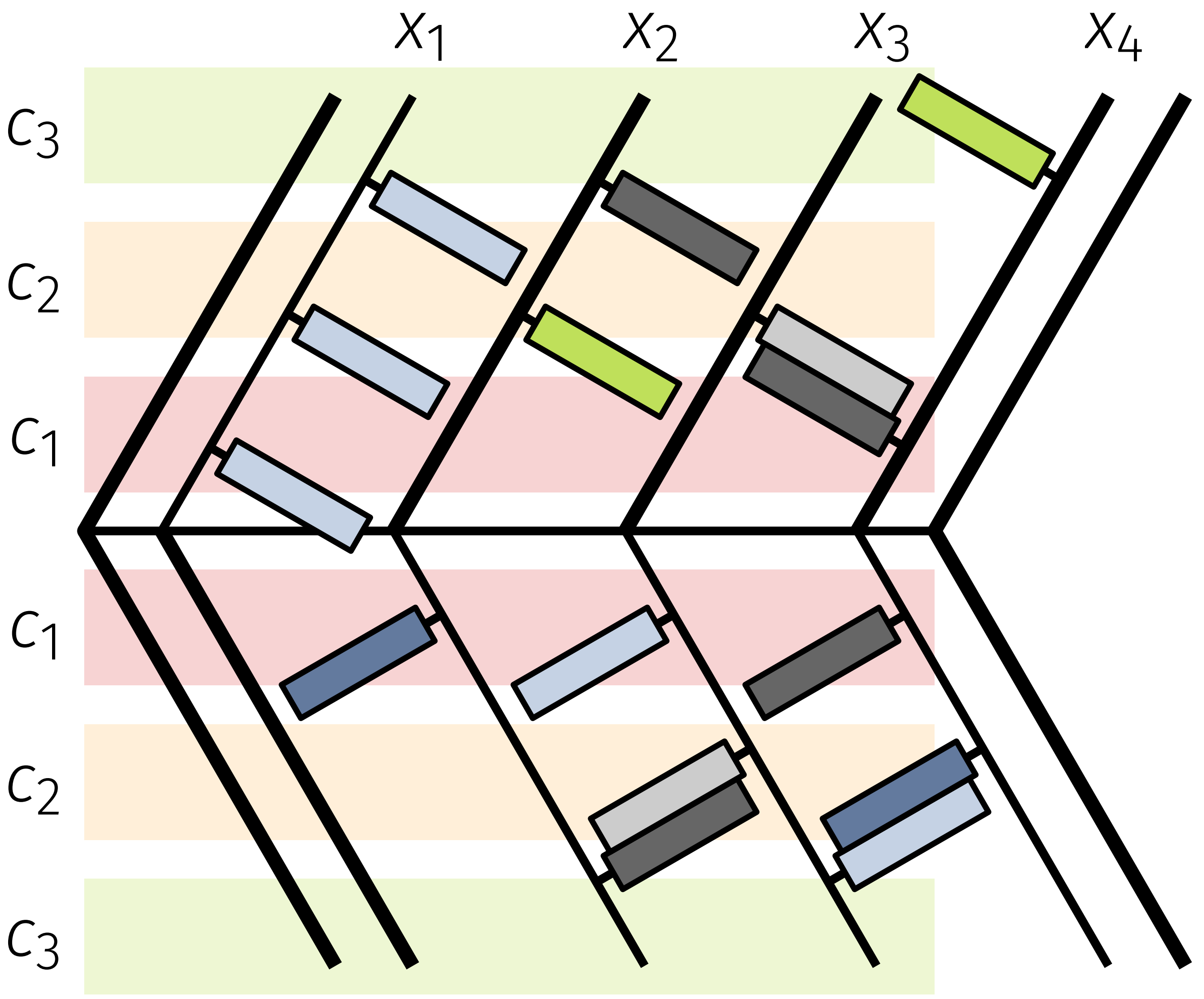
60° Logic Engine



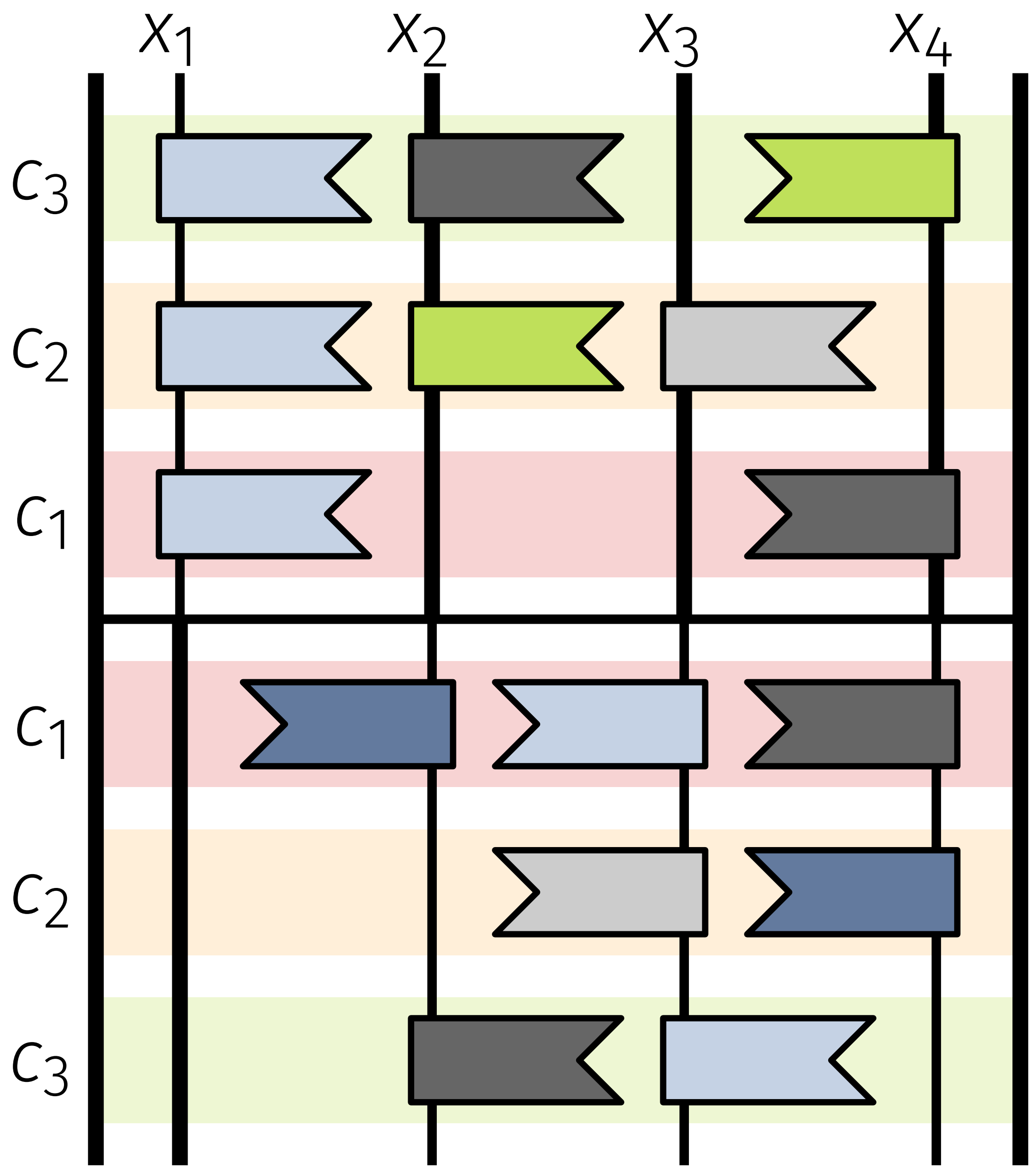
60° Logic Engine



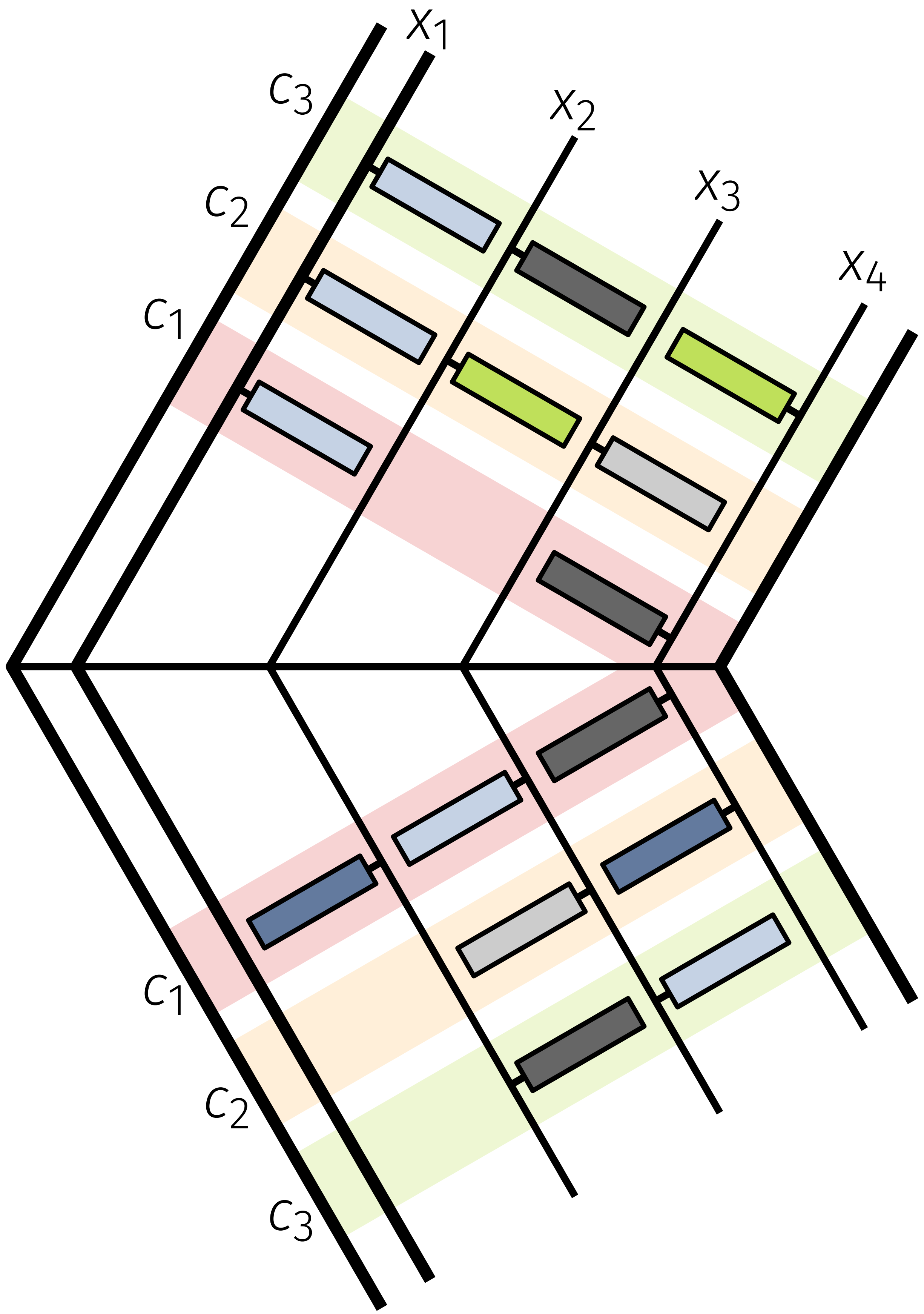
$\Rightarrow$



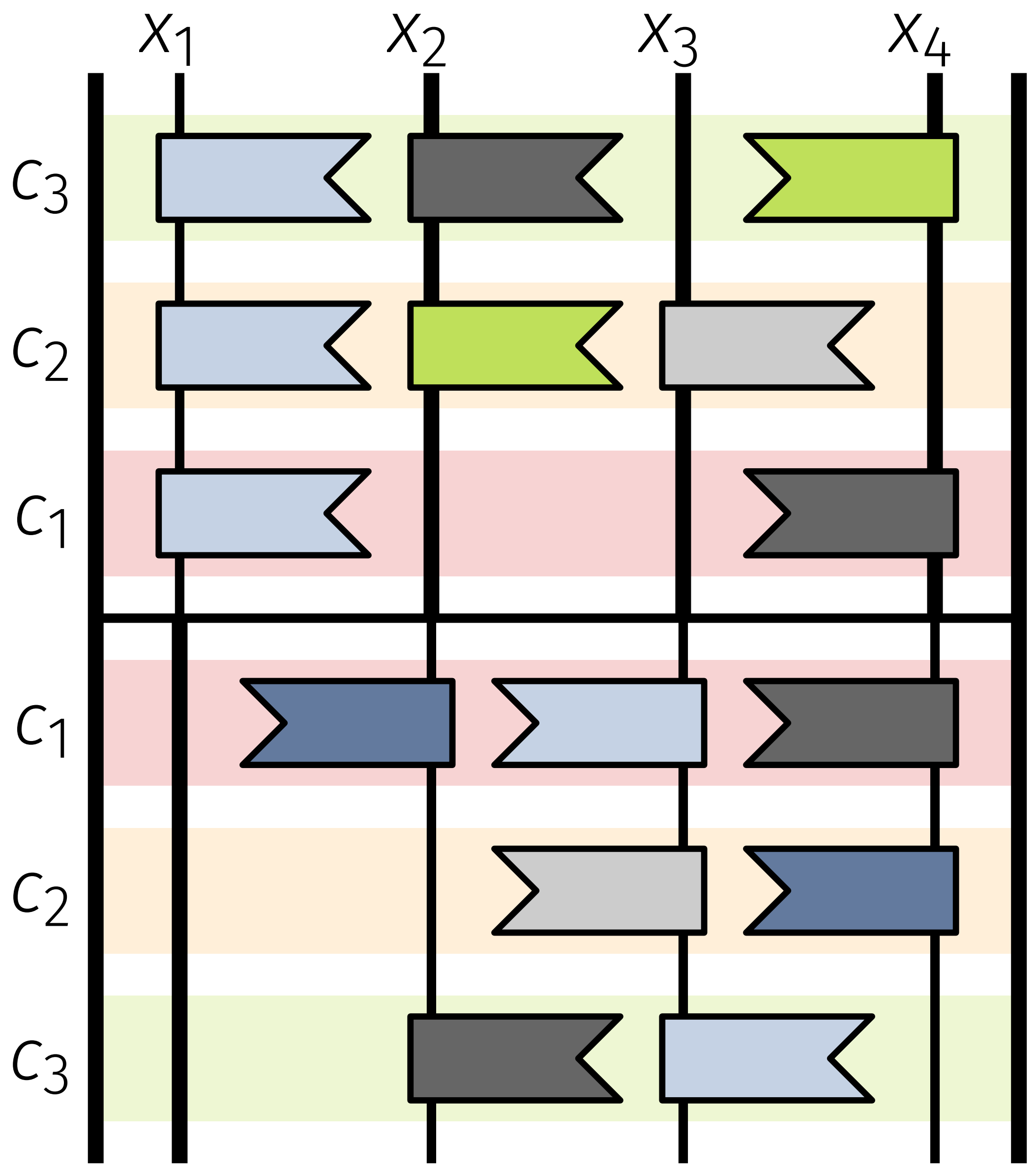
60° Logic Engine



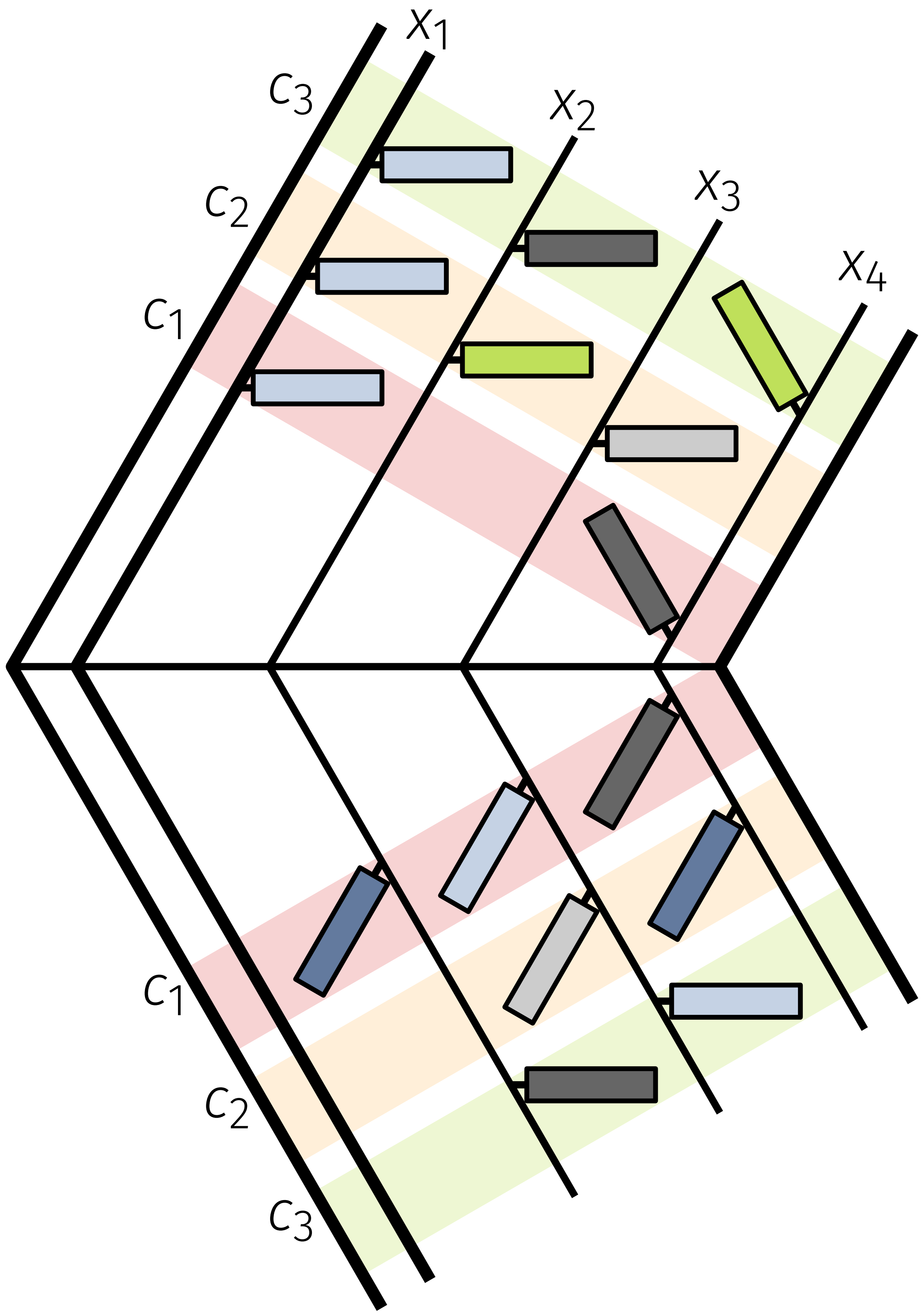
$\Rightarrow$



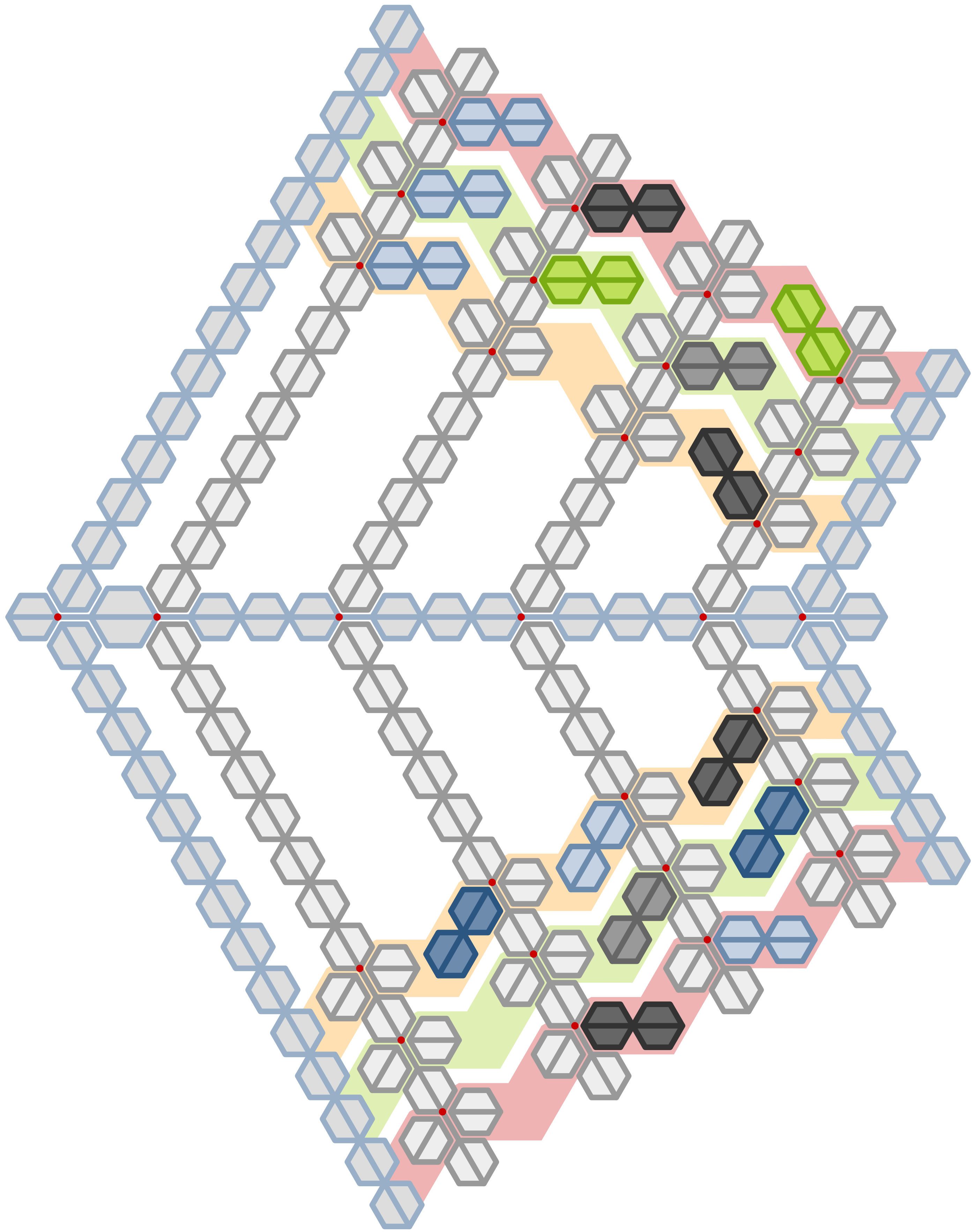
60° Logic Engine



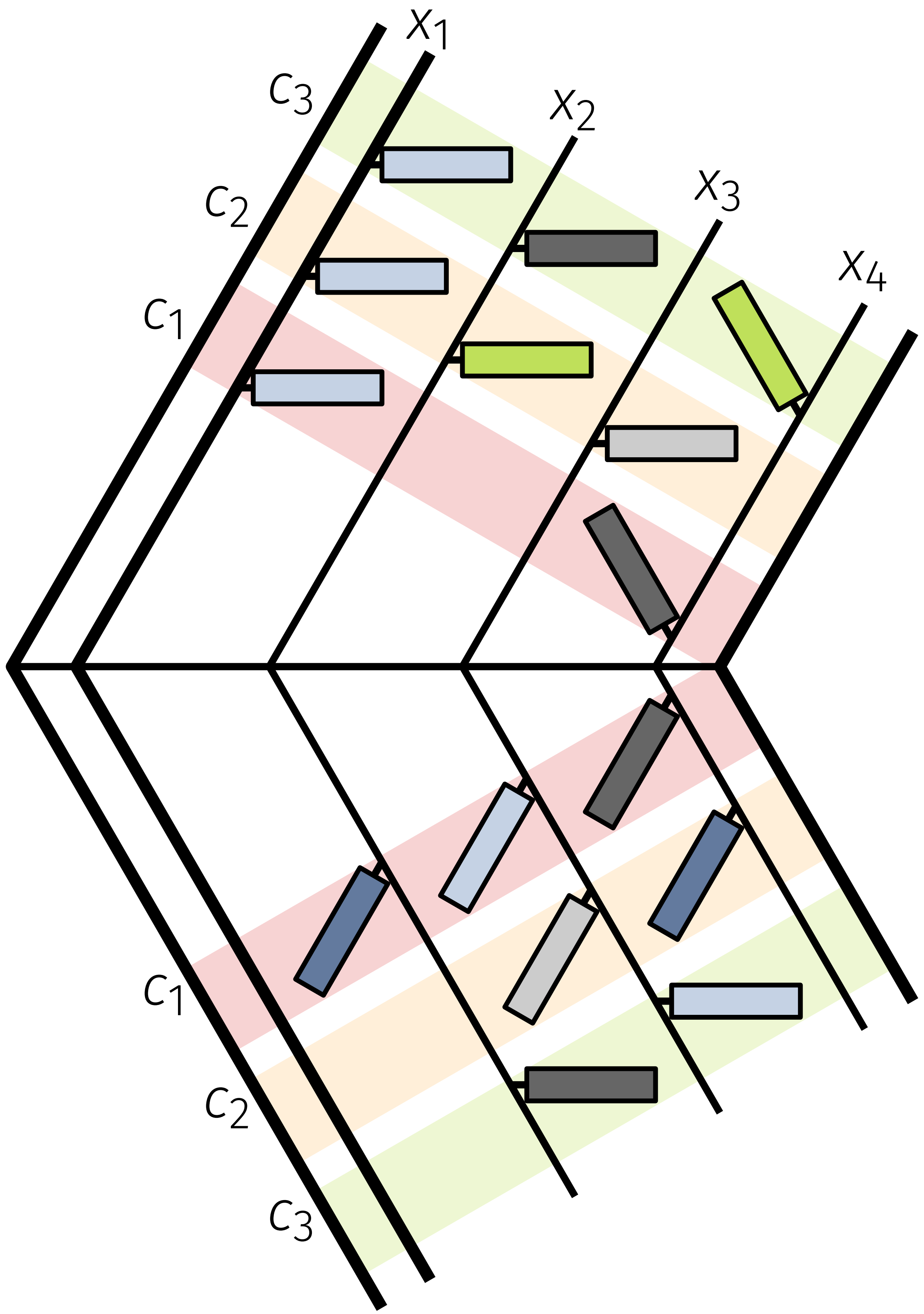
$\Rightarrow$



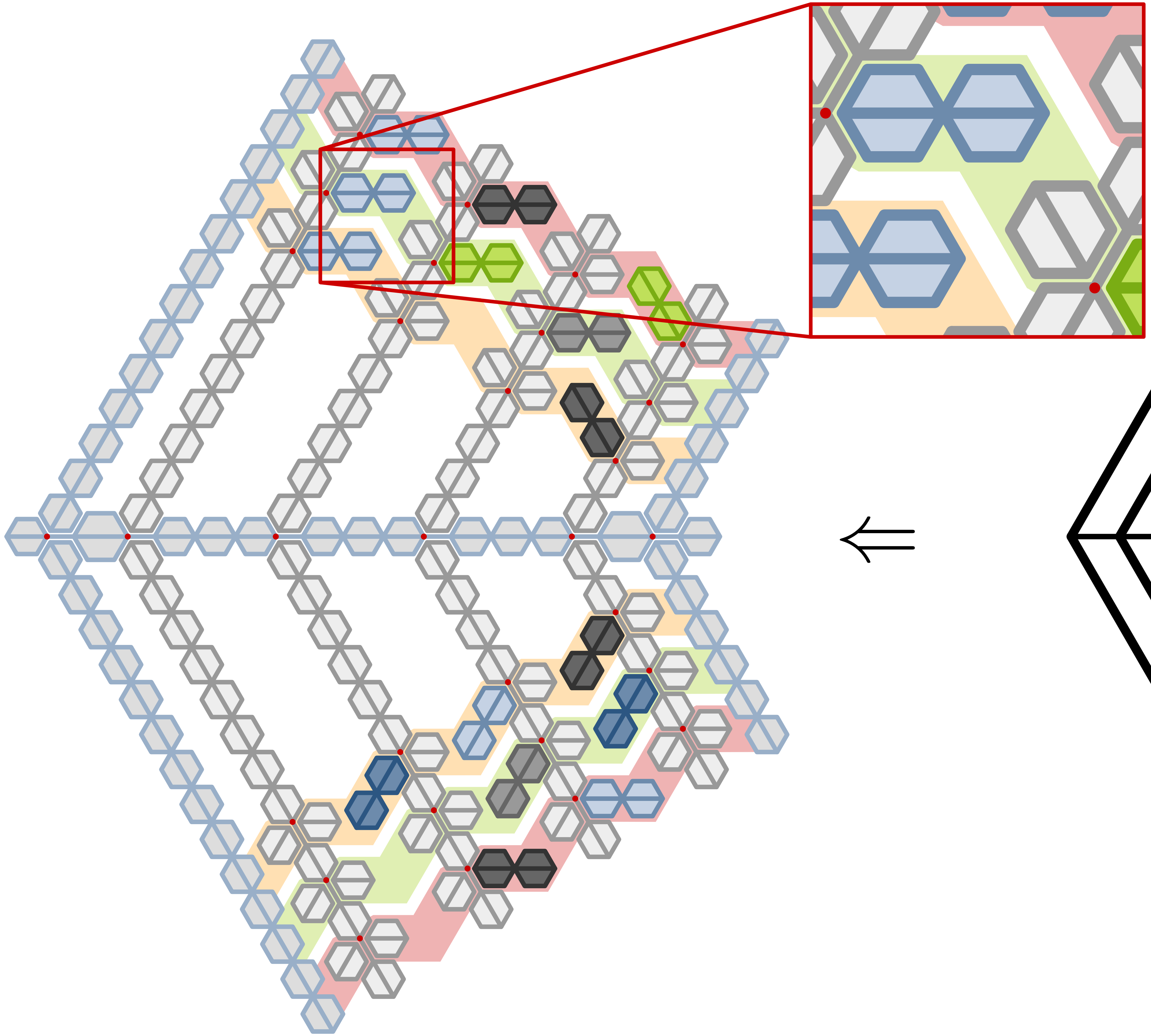
60° Logic Engine

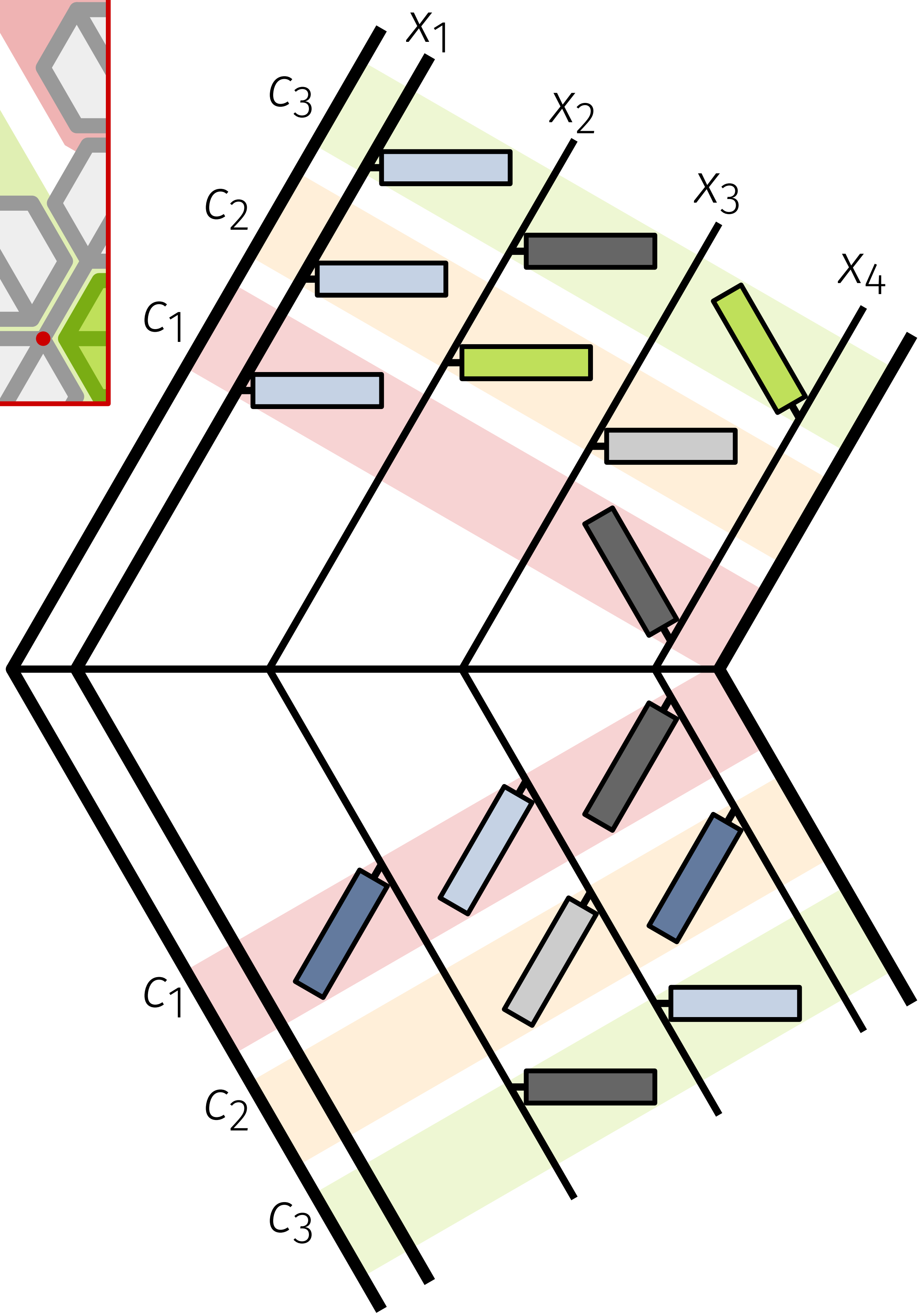


$\Leftarrow$

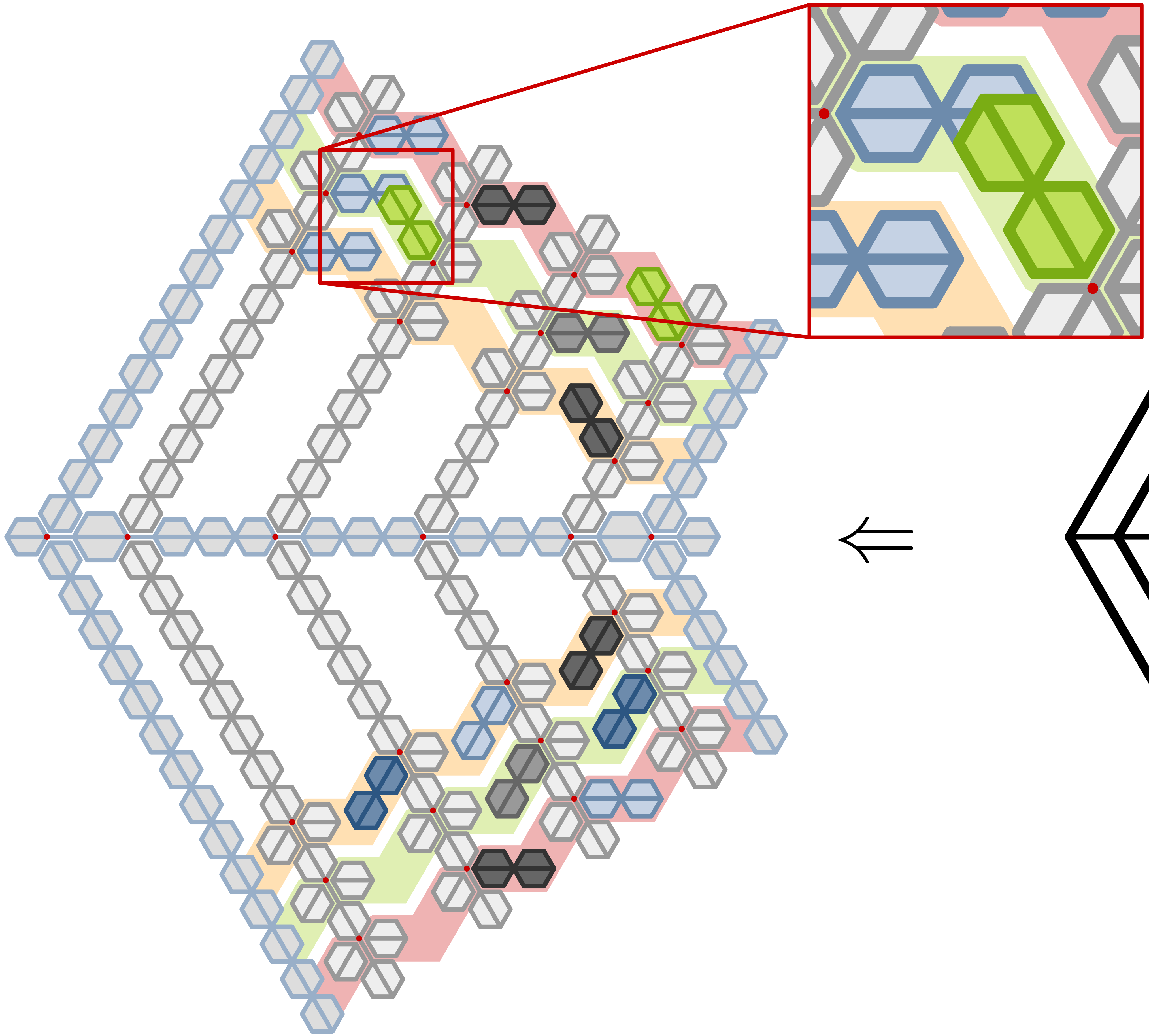


60° Logic Engine

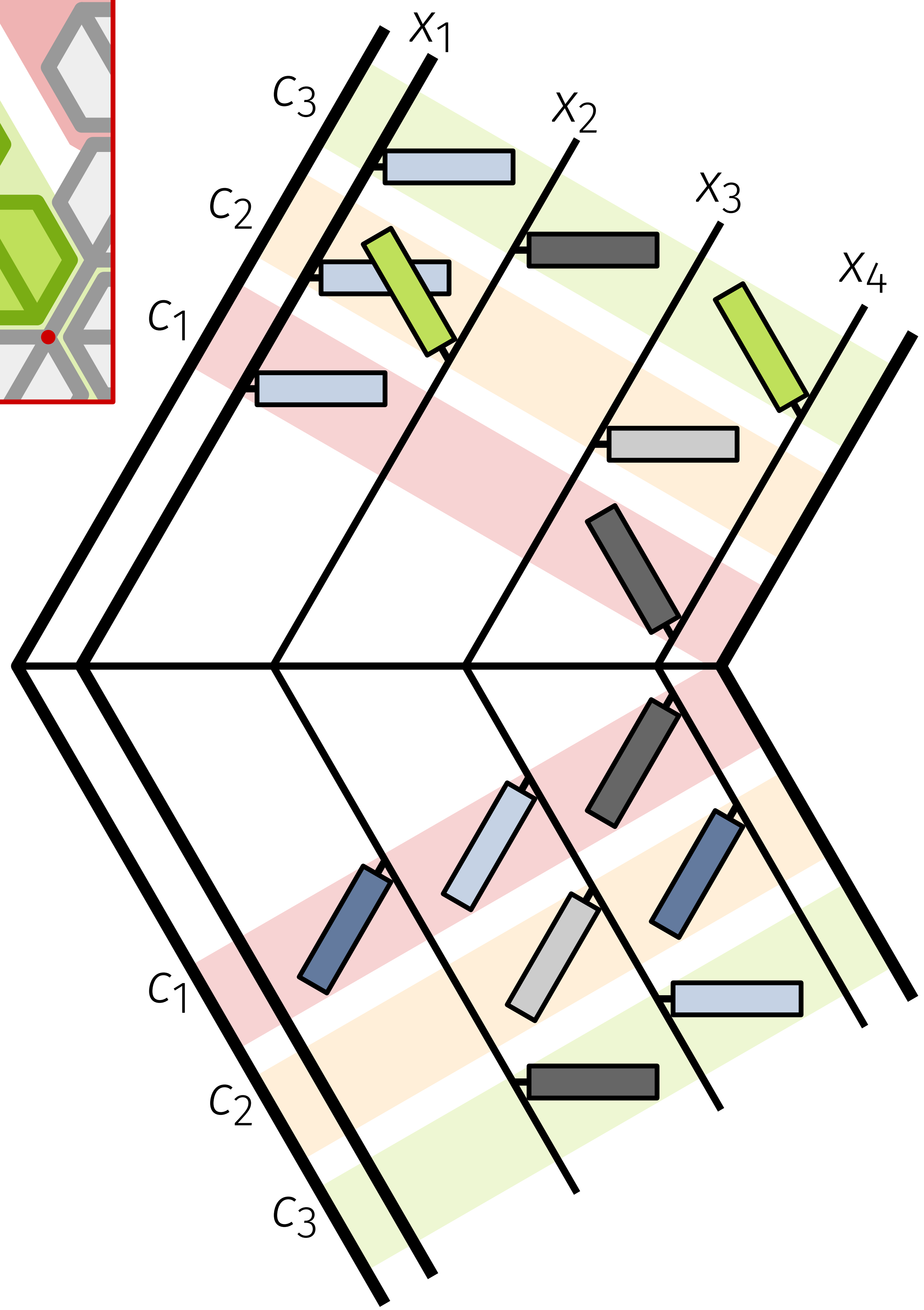




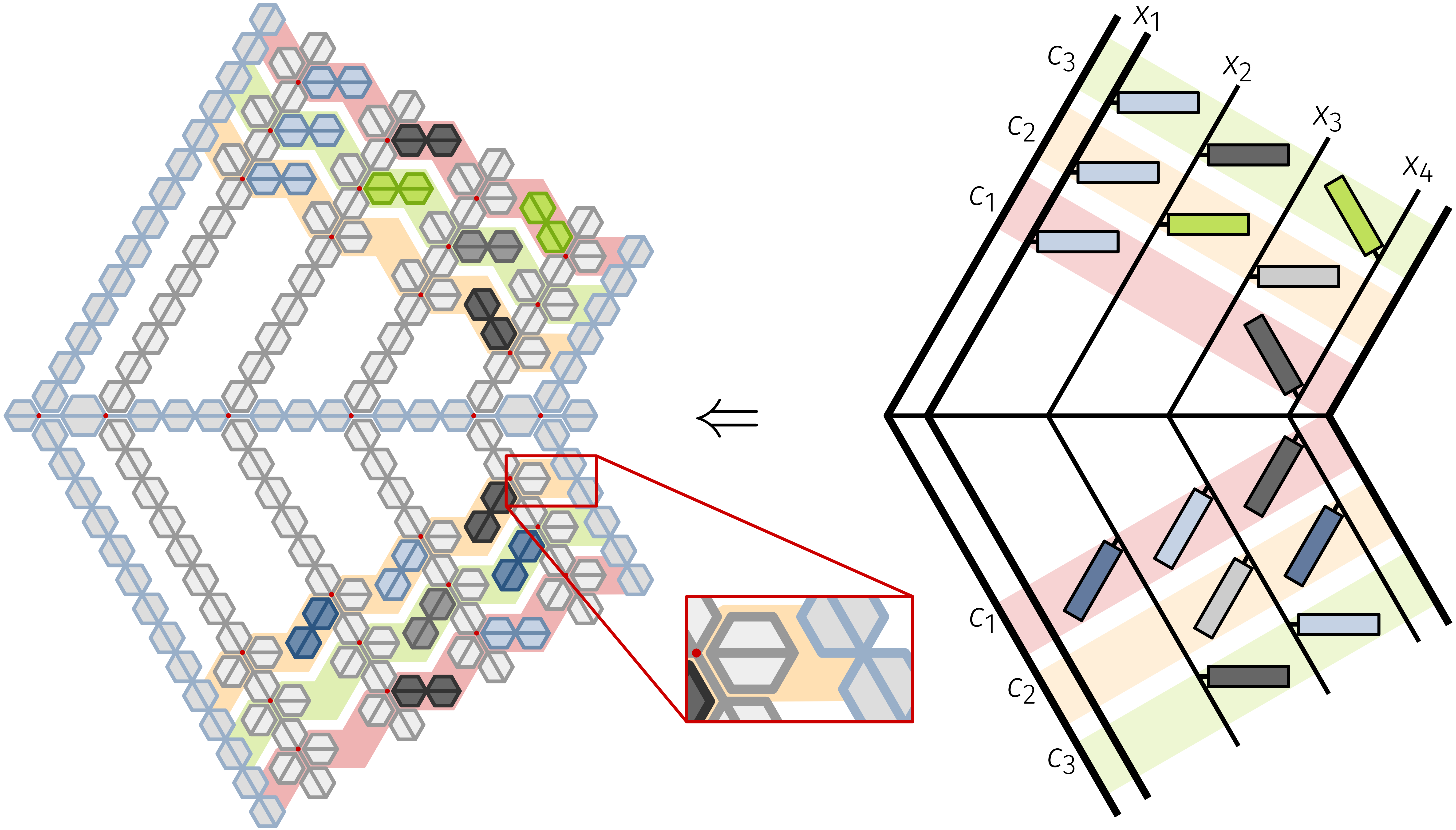
60° Logic Engine



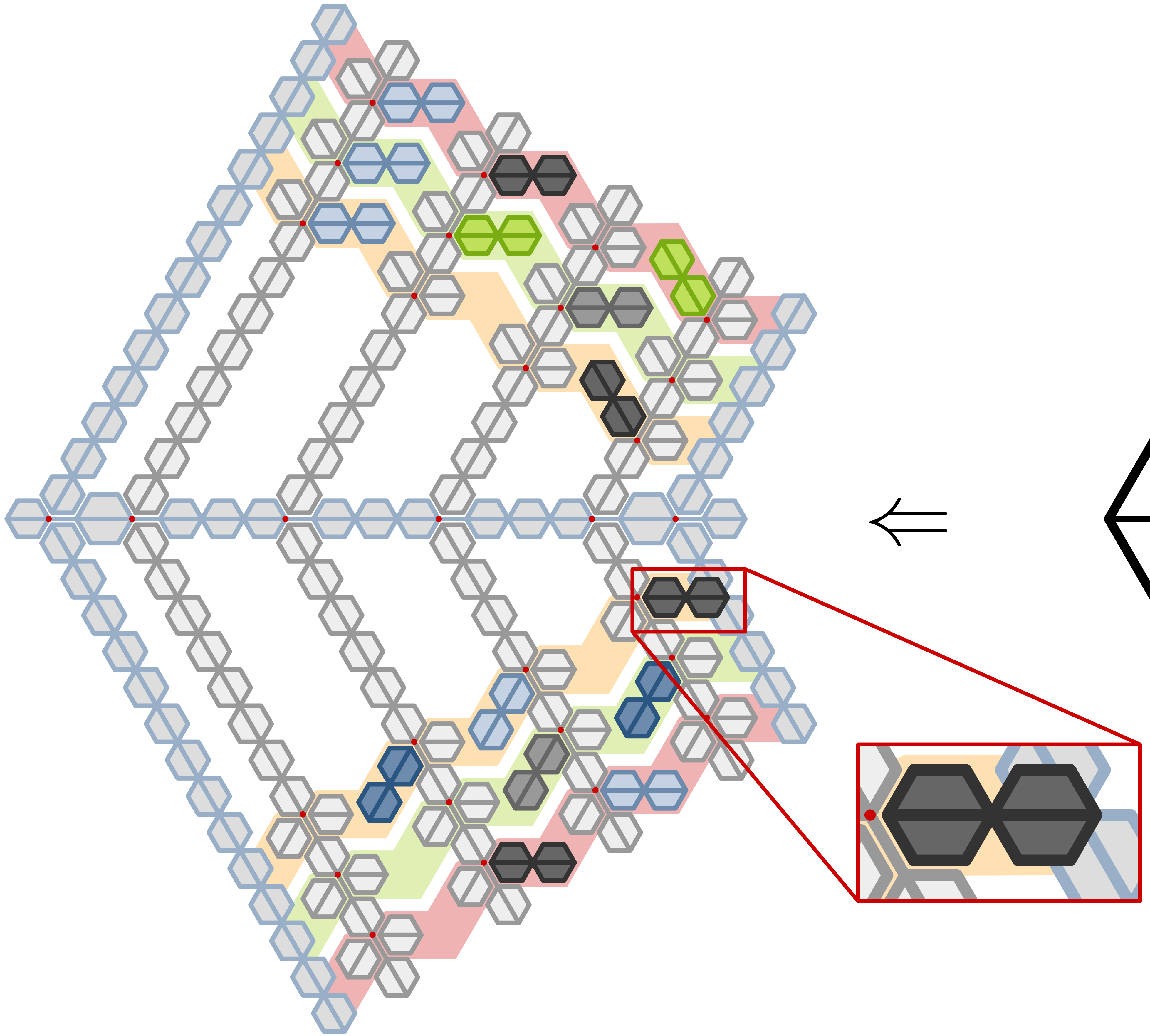
$\Leftarrow$



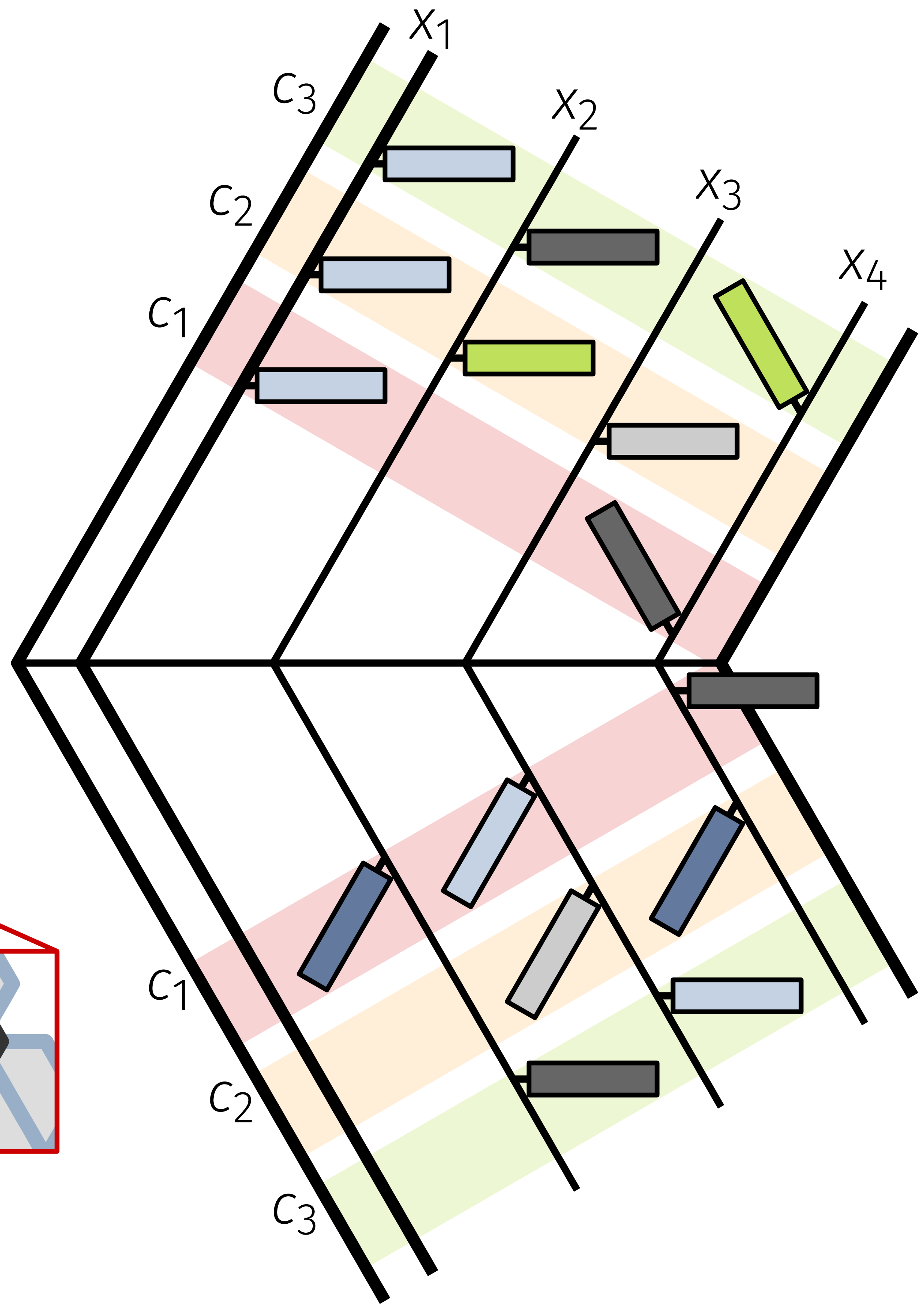
60° Logic Engine



60° Logic Engine



$\Leftarrow$



Conclusion

Freie Universität Berlin

	star	caterpillar	tree
Weak UDCR		Cleve; 2020	Cleve; 2020
Weak Emb. UDCR		Chiu, Cleve, Nöllenburg; 2019	

Conclusion

	star	caterpillar	tree
Weak UDCR		Cleve; 2020	Cleve; 2020
Weak Emb. UDCR		Chiu, Cleve, Nöllenburg; 2019	

- Recognizing *weak* UDCR for caterpillars takes linear time
- Recognizing *weak* UDCR for trees is NP-hard

Conclusion

	star	caterpillar	tree
Weak UDCR		Cleve; 2020	Cleve; 2020
Weak Emb. UDCR		Chiu, Cleve, Nöllenburg; 2019	

- Recognizing *weak* UDCR for caterpillars takes linear time
- Recognizing *weak* UDCR for trees is NP-hard

NP-completeness?

Conclusion

Freie Universität Berlin

	star	caterpillar	lobster	…	tree
Weak UDCR		Cleve; 2020	?		Cleve; 2020
Weak Emb. UDCR		Chiu, Cleve, Nöllenburg; 2019			

- Recognizing *weak* UDCR for caterpillars takes linear time
- Recognizing *weak* UDCR for trees is NP-hard NP-completeness?
- Trees in which all vertices have distance at most k to a central path?
- Lobster graphs (k = 2)?