Orthogonal Graph Drawing

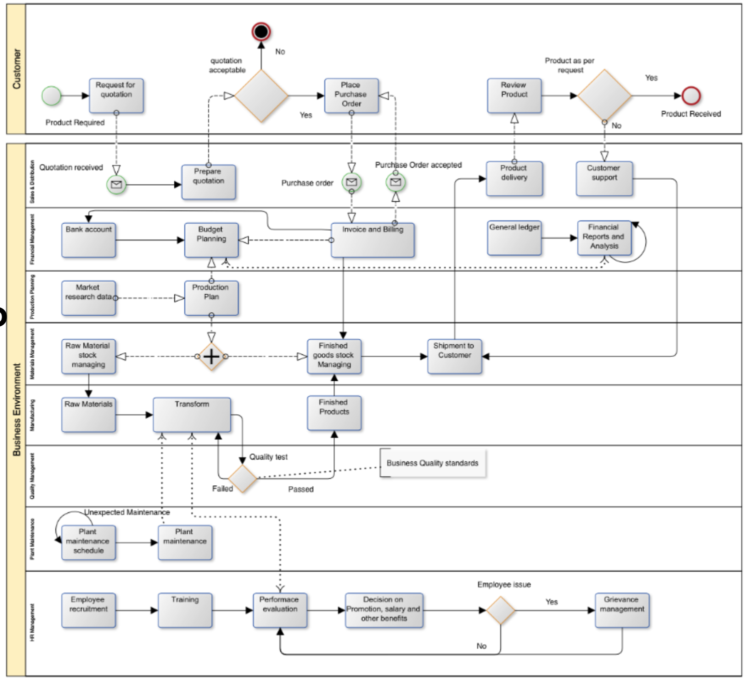
Lecture Graph Drawing Algorithms · 192.053

Martin Nöllenburg

05.06.2018

Part II: 11.06.2018

ALGORITHMS AND COMPLEXITY GROUP

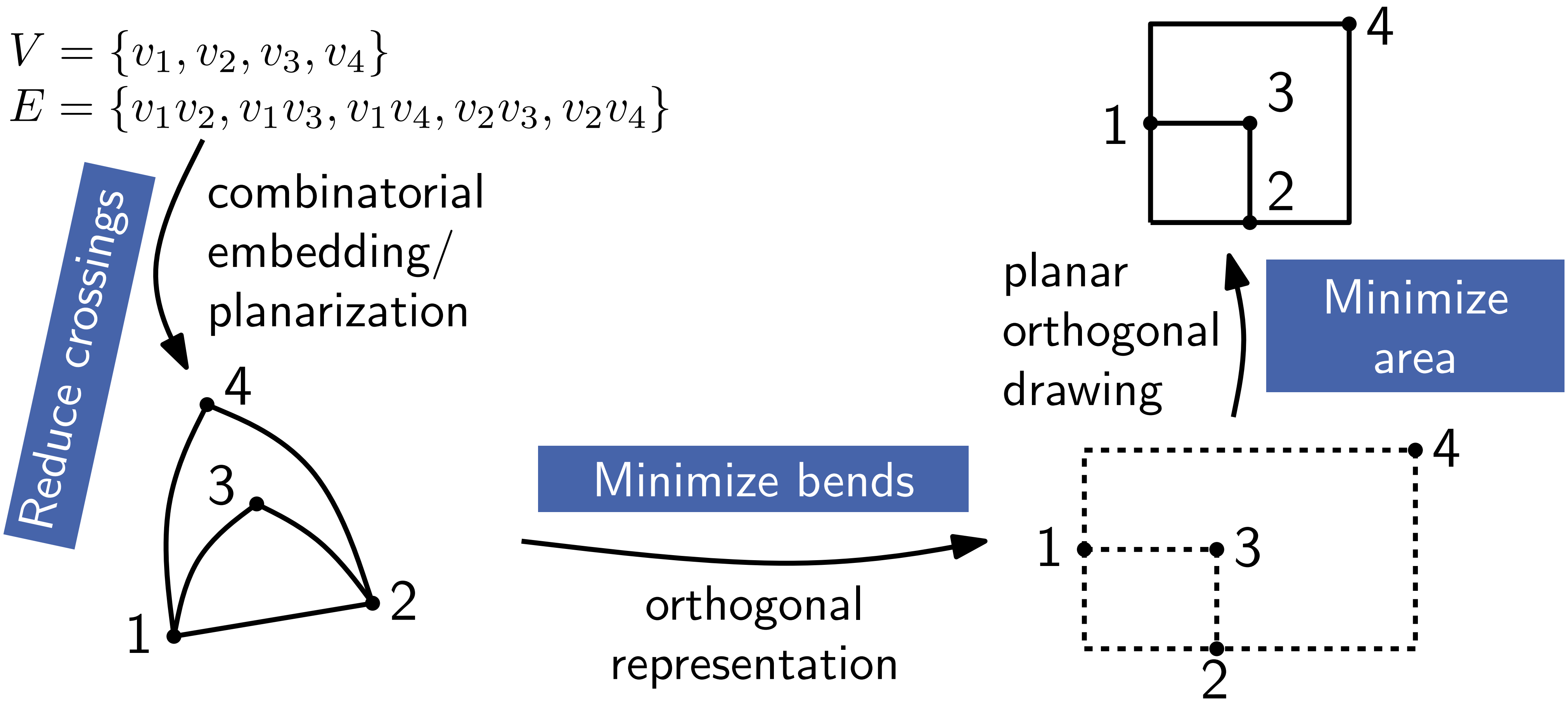


(Planar) Orthogonal Drawings

Three-step approach: *Topology – Shape – Metrics*

[Tamassia SIAM J. Comput. 1987]

$V = \{v_1, v_2, v_3, v_4\}$
$E = \{v_1v_2, v_1v_3, v_1v_4, v_2v_3, v_2v_4\}$

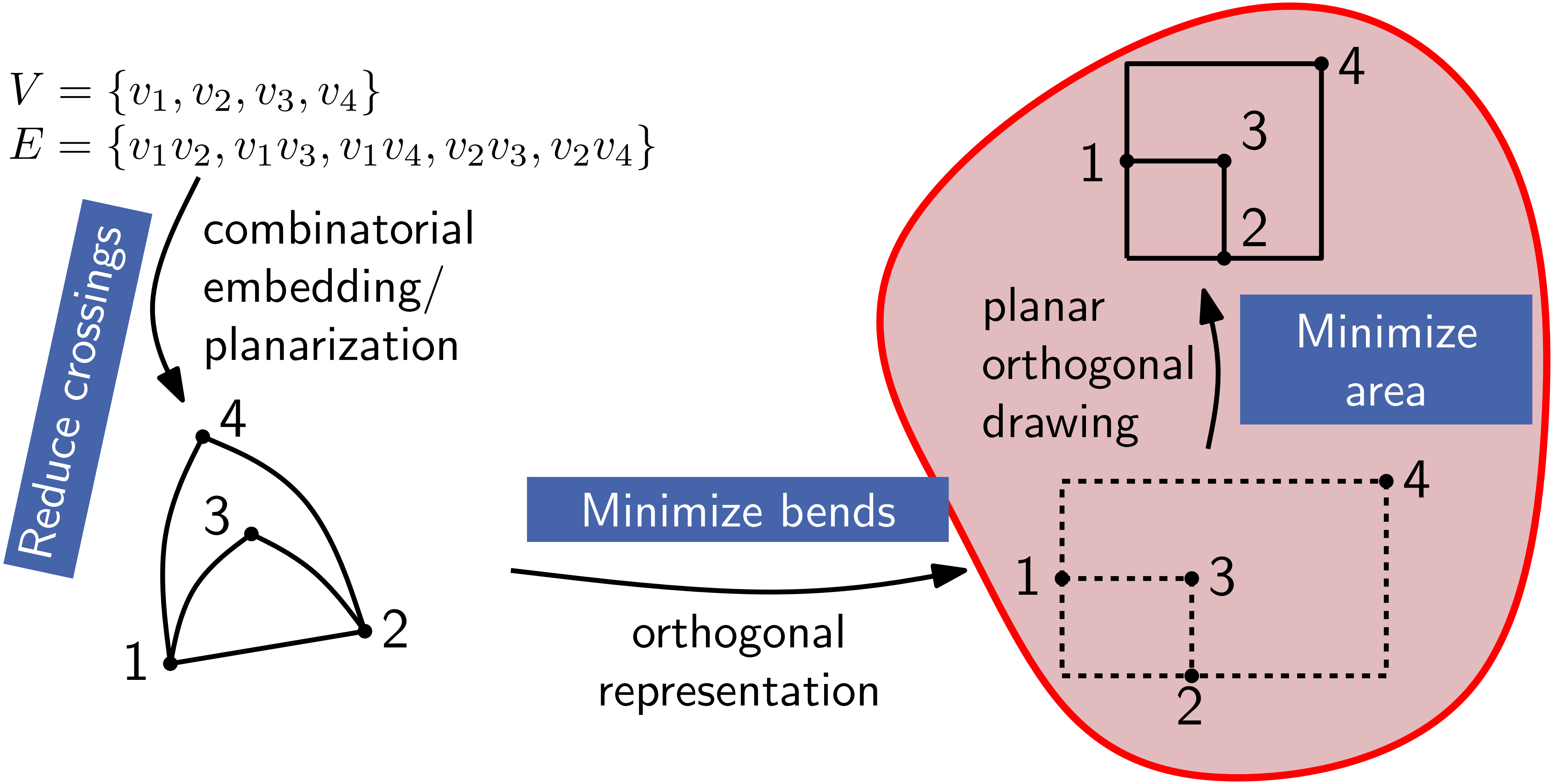


(Planar) Orthogonal Drawings

Three-step approach: *Topology – Shape – Metrics*

[Tamassia SIAM J. Comput. 1987]

$V = \{v_1, v_2, v_3, v_4\}$
$E = \{v_1v_2, v_1v_3, v_1v_4, v_2v_3, v_2v_4\}$



Compaction

Problem Compaction

Given:
- planar graph $G = (V, E)$ with maximum degree 4
- orthogonal representation $H(G)$

Find: compact orthogonal layout of G that realizes $H(G)$

Compaction

Problem Compaction

Given:
- planar graph $G = (V, E)$ with maximum degree 4
- orthogonal representation $H(G)$

Find: compact orthogonal layout of G that realizes $H(G)$

Special case: all faces are rectangles

$\rightarrow$ can show guarantees
- minimum total edge length
- minimum area

Compaction

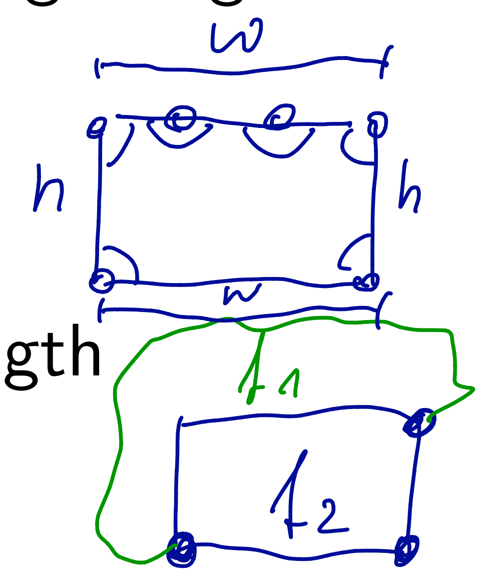
Problem Compaction

Given:
- planar graph $G = (V, E)$ with maximum degree 4
- orthogonal representation $H(G)$

Find: compact orthogonal layout of G that realizes $H(G)$

Special case: all faces are rectangles

$\rightarrow$ can show guarantees
- minimum total edge length
- minimum area



Properties:

- bends only on the outer face
- opposite sides of a face have the same length

Compaction

Problem Compaction

Given:
- planar graph $G = (V, E)$ with maximum degree 4
- orthogonal representation $H(G)$

Find: compact orthogonal layout of G that realizes $H(G)$

Special case: all faces are rectangles

$\rightarrow$ can show guarantees
- minimum total edge length
- minimum area

Properties:

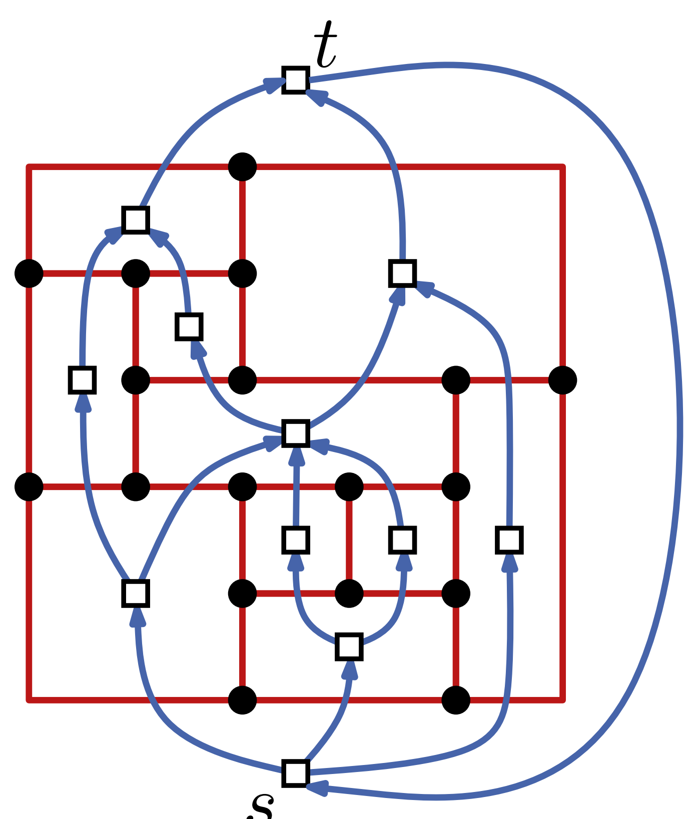
- bends only on the outer face
- opposite sides of a face have the same length

We will create another flow network for compaction.

Flow Network for Edge Length Assignment

Def: flow network $N_{\text{hor}} = ((W_{\text{hor}}, A_{\text{hor}}); \ell; c; b; \text{cost})$

- $W_{\text{hor}} = \mathcal{F} \setminus \{f_0\} \cup \{s, t\}$
- $A_{\text{hor}} = \{(f, g) \mid f, g$ share a horizontal segment and f lies below $g\} \cup \{(t, s)\}$
- $\ell(a) = 1 \quad \forall a \in A_{\text{hor}}$
- $c(a) = \infty \quad \forall a \in A_{\text{hor}}$
- $\text{cost}(a) = 1 \quad \forall a \in A_{\text{hor}}$
- $b(f) = 0 \quad \forall f \in W_{\text{hor}}$

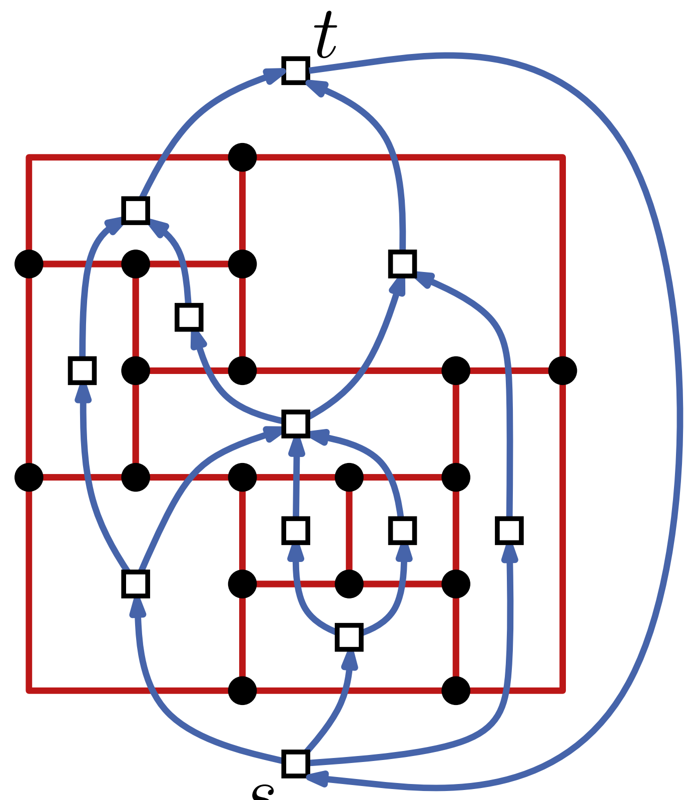


Flow Network for Edge Length Assignment

Def: flow network $N_{\mathsf{hor}} = ((W_{\mathsf{hor}}, A_{\mathsf{hor}}); \ell; c; b; \mathsf{cost})$

- $W_{\mathsf{hor}} = \mathcal{F} \setminus \{f_0\} \cup \{s, t\}$
- $A_{\mathsf{hor}} = \{(f, g) \mid f, g$ share a horizontal segment and f lies below $g\} \cup \{(t, s)\}$
- $\ell(a) = 1 \quad \forall a \in A_{\mathsf{hor}}$
- $c(a) = \infty \quad \forall a \in A_{\mathsf{hor}}$
- $\mathsf{cost}(a) = 1 \quad \forall a \in A_{\mathsf{hor}}$
- $b(f) = 0 \quad \forall f \in W_{\mathsf{hor}}$

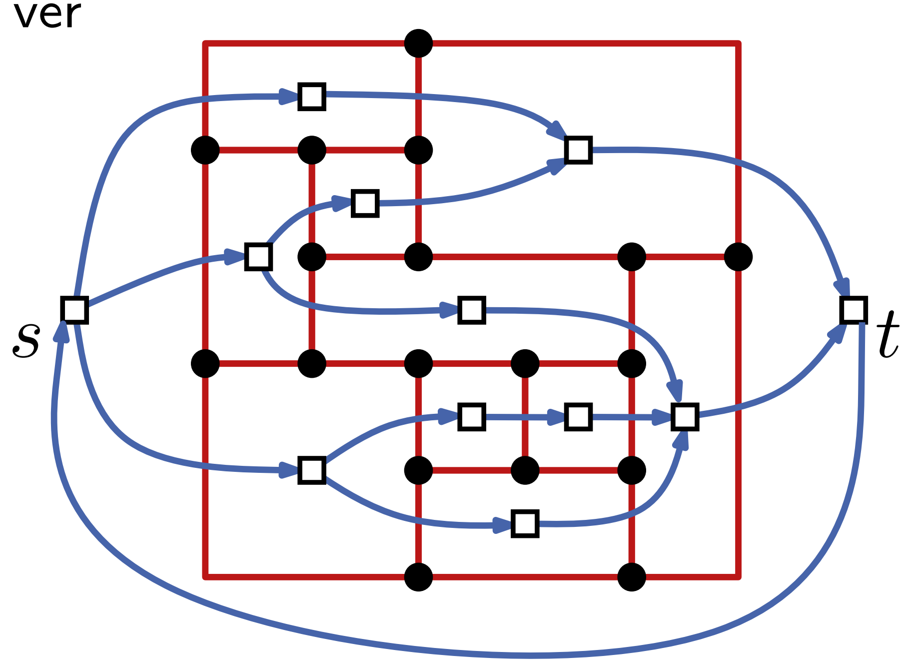
s and t represent lower and upper side of f_0



Flow Network for Edge Length Assignment

Def: flow network $N_{\text{ver}} = ((W_{\text{ver}}, A_{\text{ver}}); \ell; c; b; \text{cost})$

- $W_{\text{ver}} = \mathcal{F} \setminus \{f_0\} \cup \{s, t\}$
- $A_{\text{ver}} = \{(f, g) \mid f, g$ share a vertical segment and f lies to the left of $g\} \cup \{(t, s)\}$
- $\ell(a) = 1 \quad \forall a \in A_{\text{ver}}$
- $c(a) = \infty \quad \forall a \in A_{\text{ver}}$
- $\text{cost}(a) = 1 \quad \forall a \in A_{\text{ver}}$
- $b(f) = 0 \quad \forall f \in W_{\text{ver}}$



Flow Network for Edge Length Assignment

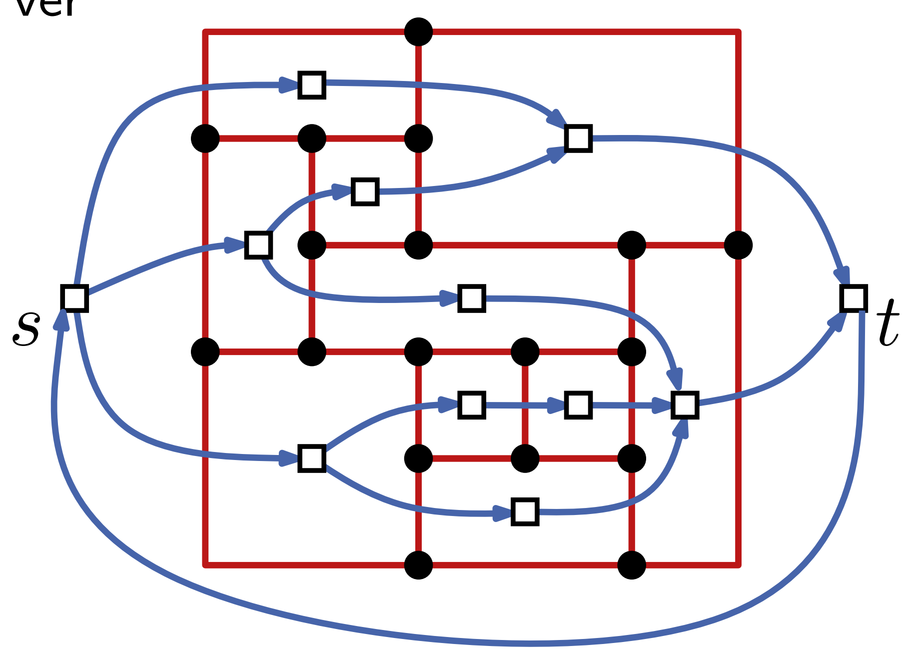
Def: flow network $N_{\text{ver}} = ((W_{\text{ver}}, A_{\text{ver}}); \ell; c; b; \text{cost})$

- $W_{\text{ver}} = \mathcal{F} \setminus \{f_0\} \cup \{s, t\}$
- $A_{\text{ver}} = \{(f, g) \mid f, g$ share a vertical segment and f lies to the left of $g\} \cup \{(t, s)\}$
- $\ell(a) = 1 \quad \forall a \in A_{\text{ver}}$
- $c(a) = \infty \quad \forall a \in A_{\text{ver}}$
- $\text{cost}(a) = 1 \quad \forall a \in A_{\text{ver}}$
- $b(f) = 0 \quad \forall f \in W_{\text{ver}}$

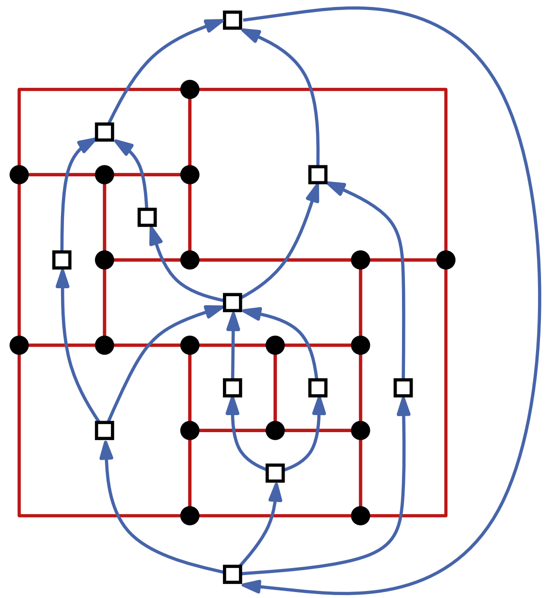
How does a flow represent the total edge length or the area?

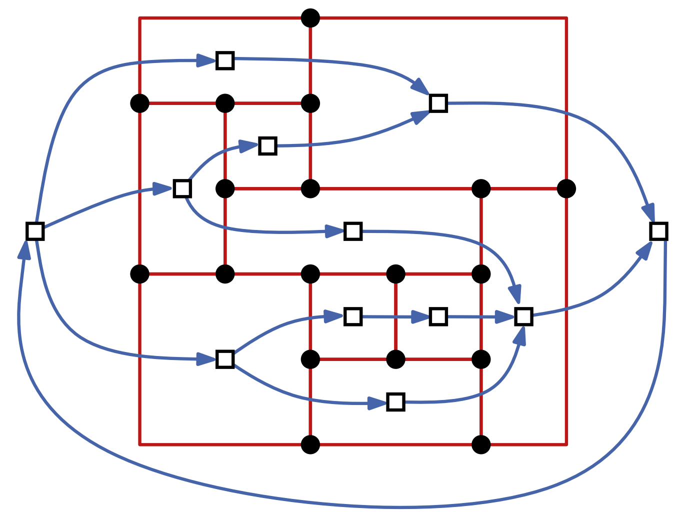
want:

$\sum_{e \in E}$ length(e)



Optimal Layout



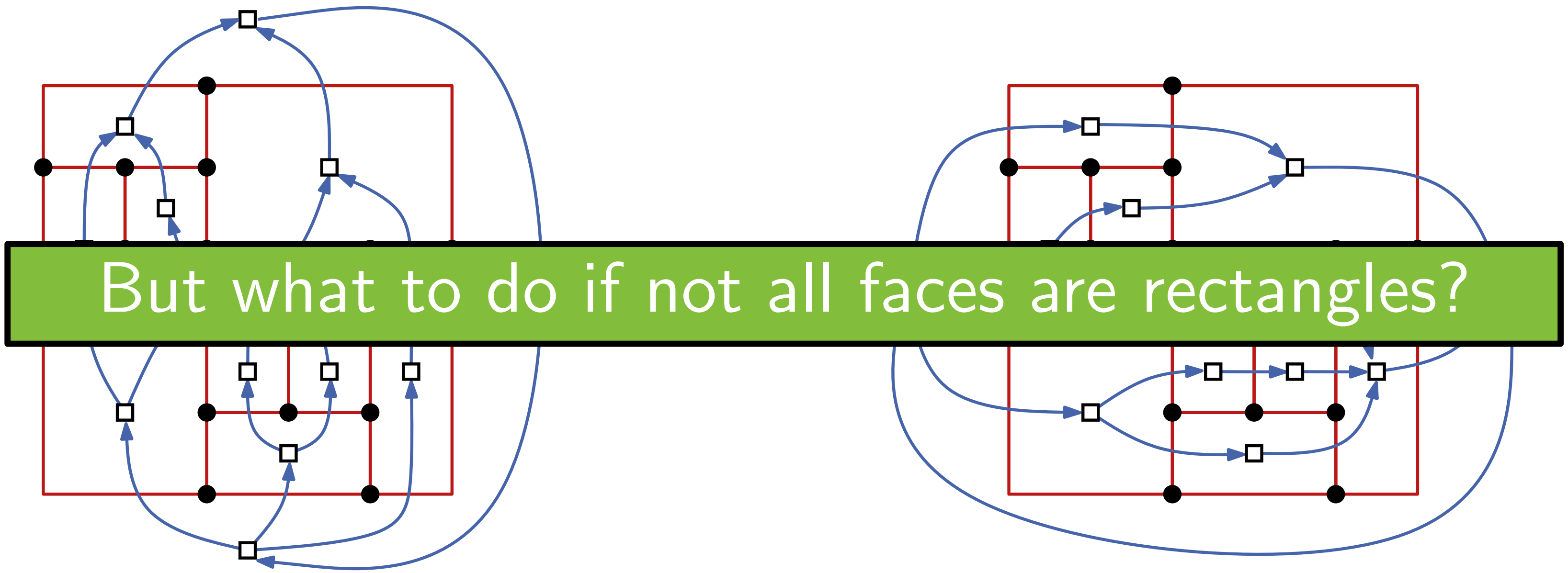


Theorem: Integer flows x_{hor} and x_{ver} in N_{hor} and N_{ver} with minimum cost induce valid orthogonal layout.

- $|x_{\text{hor}}(t, s)|$ und $|x_{\text{ver}}(t, s)|$ correspond to minimum width and height of a layout
- $\sum_{a \in A^{*}_{\text{hor}}} x_{\text{hor}}(e) \cdot \text{cost}(a) + \sum_{e \in A^{*}_{\text{ver}}} x_{\text{ver}}(a) \cdot \text{cost}(a)$ correspond to minimum total edge length

$A_{hor} \setminus \{s,t\}$ $A_{ver} \setminus \{s,t\}$

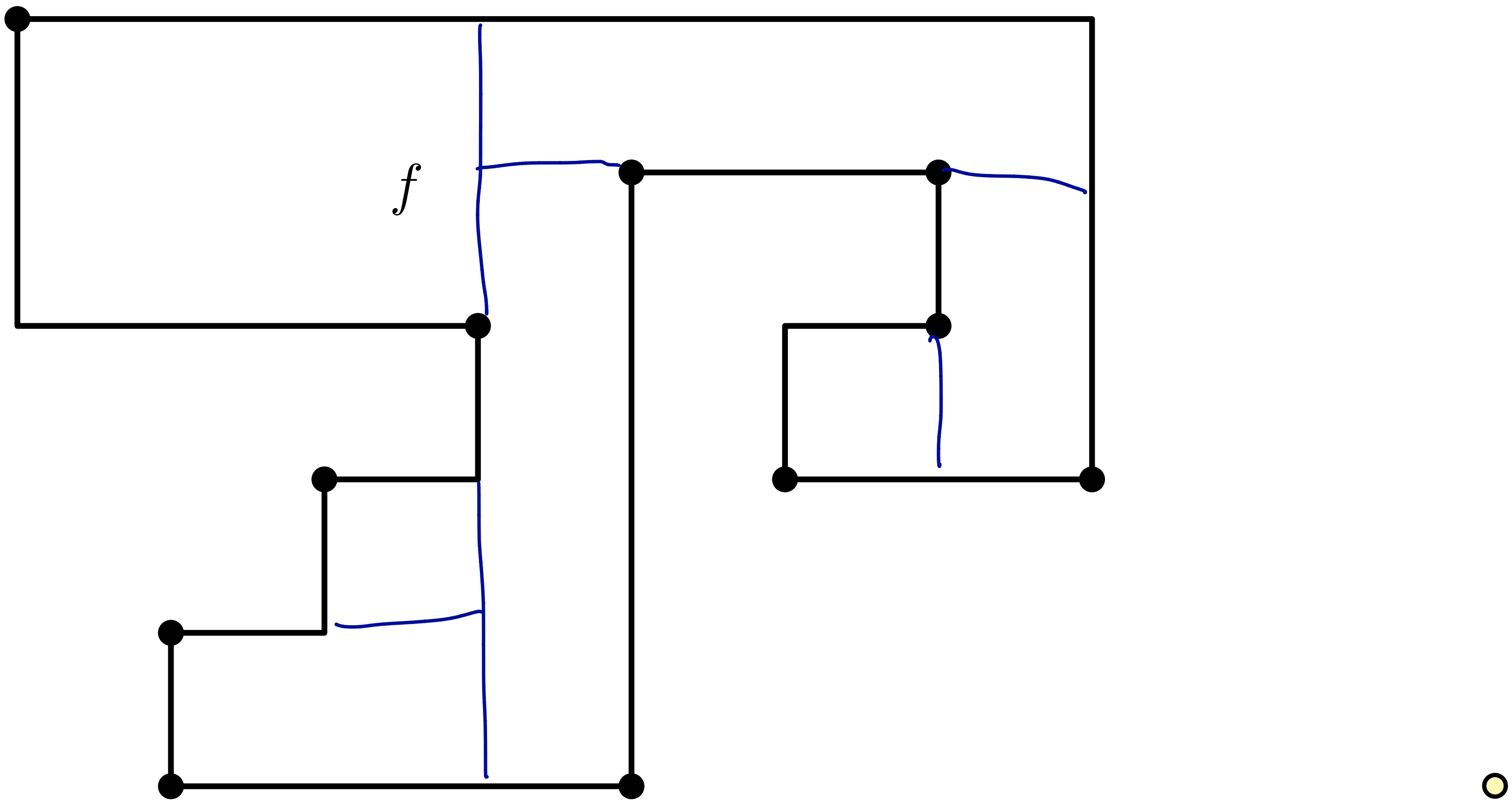
Optimal Layout



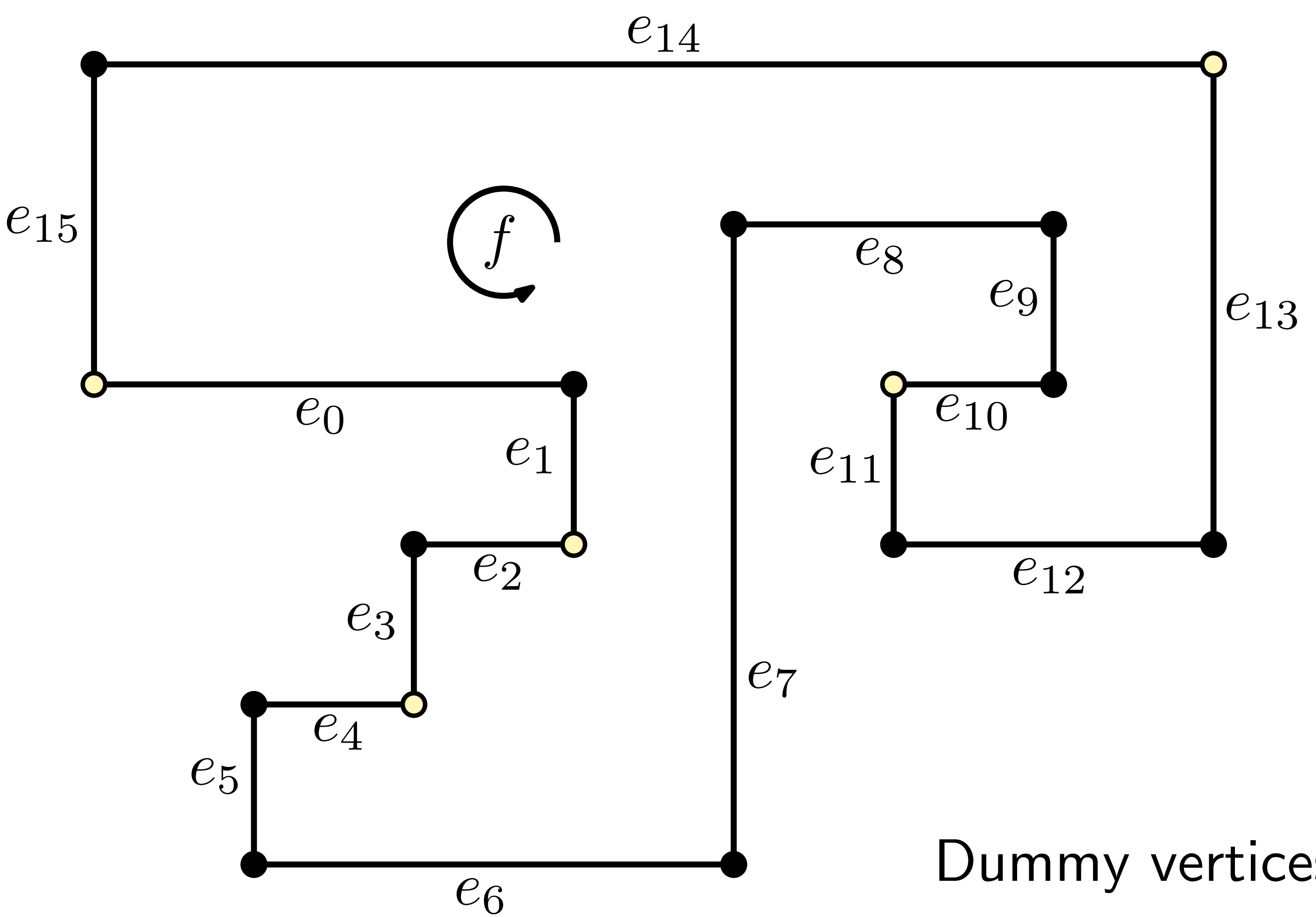
Theorem: Integer flows x_{hor} and x_{ver} in N_{hor} and N_{ver} with minimum cost induce valid orthogonal layout.

- $|x_{\text{hor}}(t,s)|$ und $|x_{\text{ver}}(t,s)|$ correspond to minimum width and height of a layout
- $\sum_{a \in A_{\text{hor}}} x_{\text{hor}}(e) \cdot \text{cost}(a) + \sum_{e \in A_{\text{ver}}} x_{\text{ver}}(a) \cdot \text{cost}(a)$ correspond to minimum total edge length

Refinement of (G, H) – Inner Face

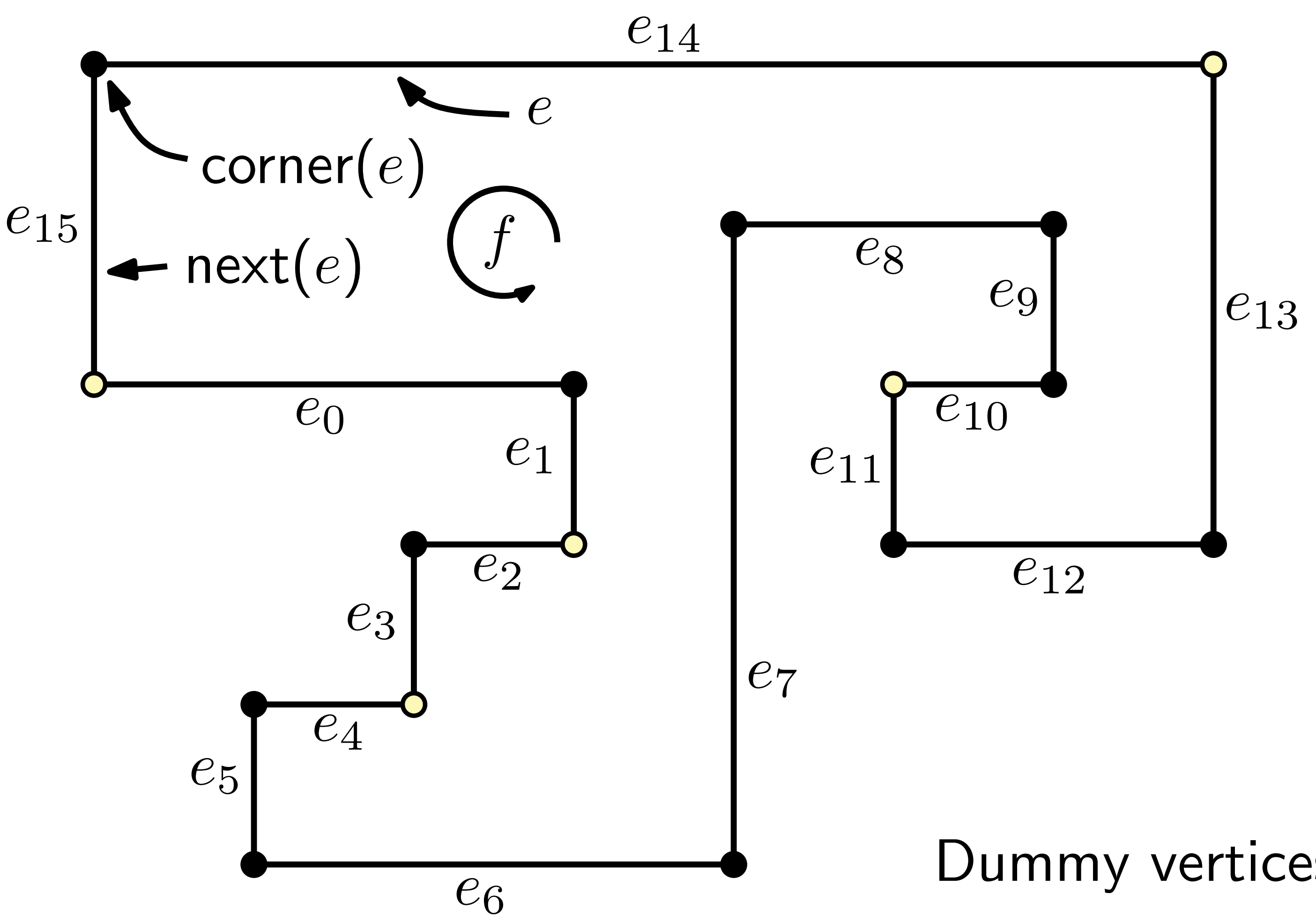


Refinement of (G, H) – Inner Face



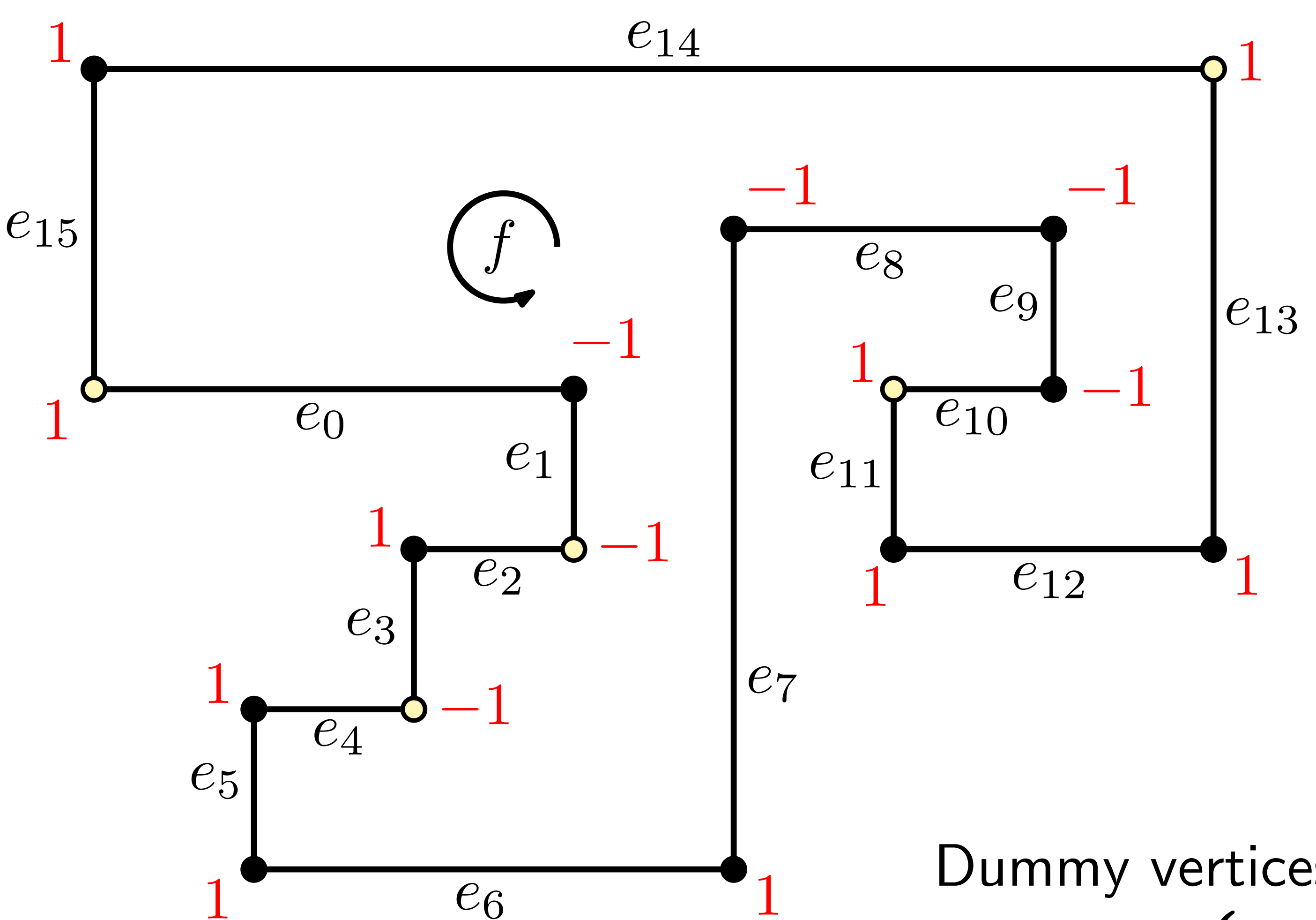
Dummy vertices for bends:

Refinement of (G, H) – Inner Face



Dummy vertices for bends:

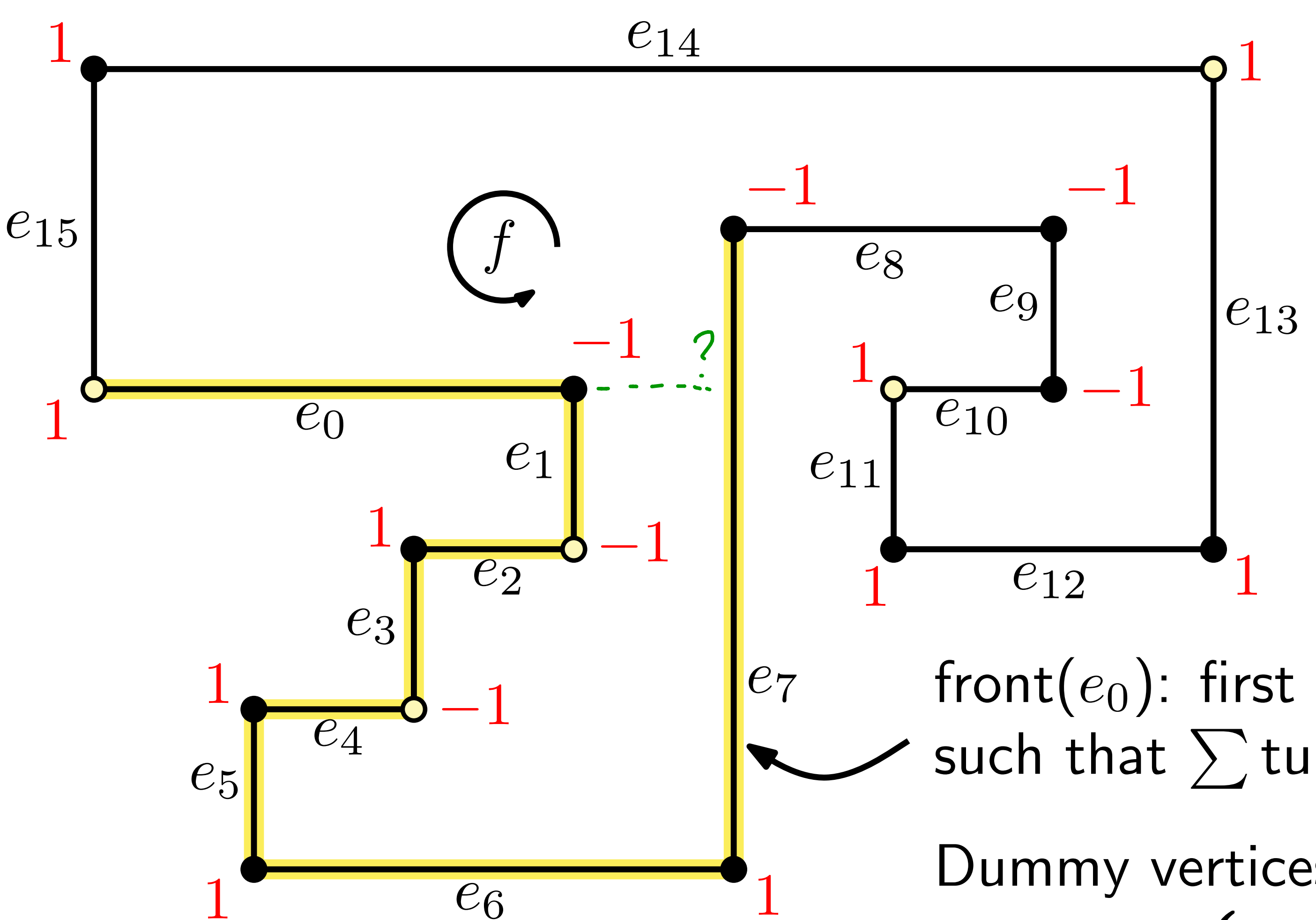
Refinement of (G, H) – Inner Face



Dummy vertices for bends: ○

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

Refinement of (G, H) – Inner Face

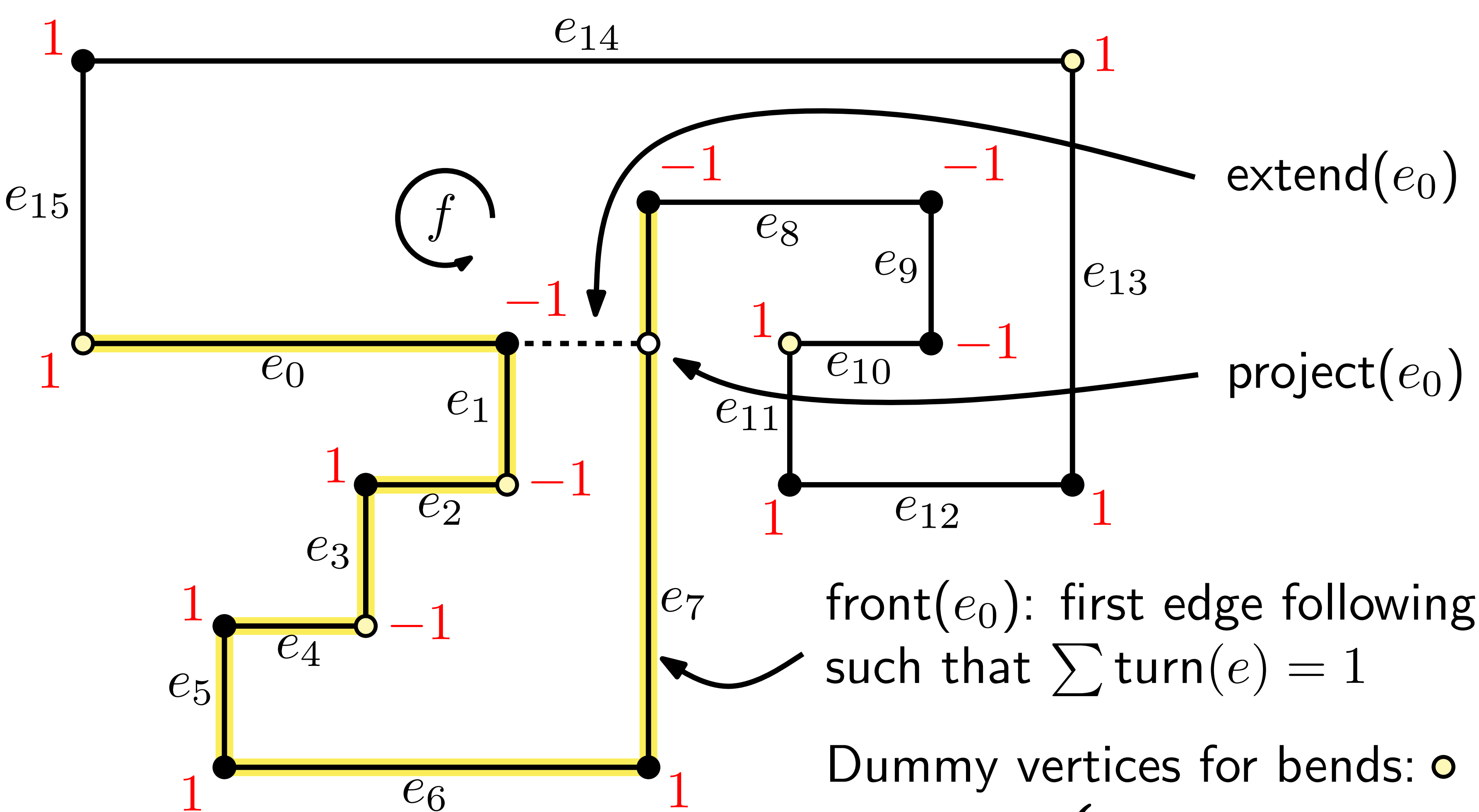


front(e_0): first edge following e_0 such that $\sum$ turn$(e) = 1$

Dummy vertices for bends: ○

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

Refinement of (G, H) – Inner Face

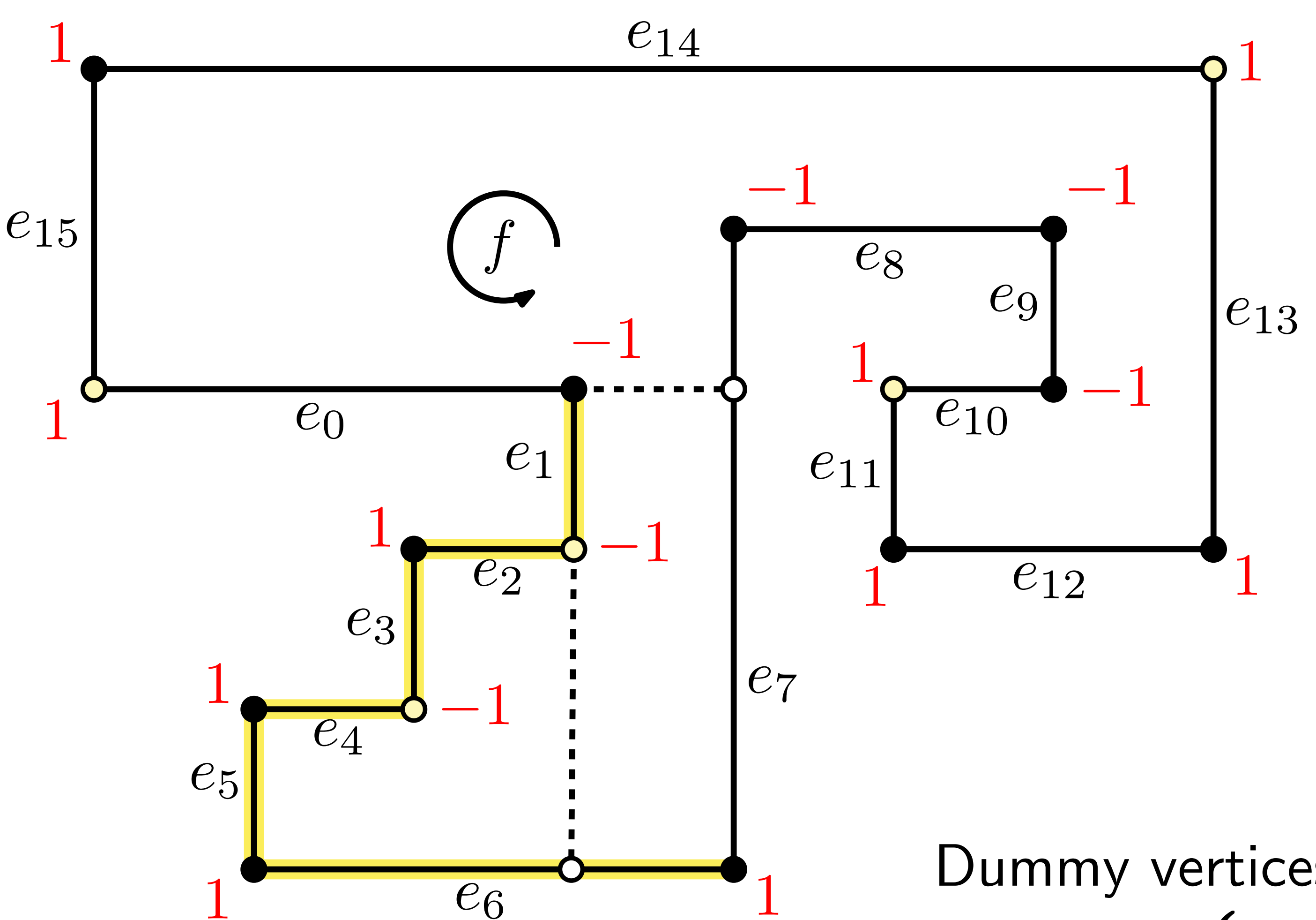


front(e_0): first edge following e_0 such that $\sum$ turn$(e) = 1$

Dummy vertices for bends: ○

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

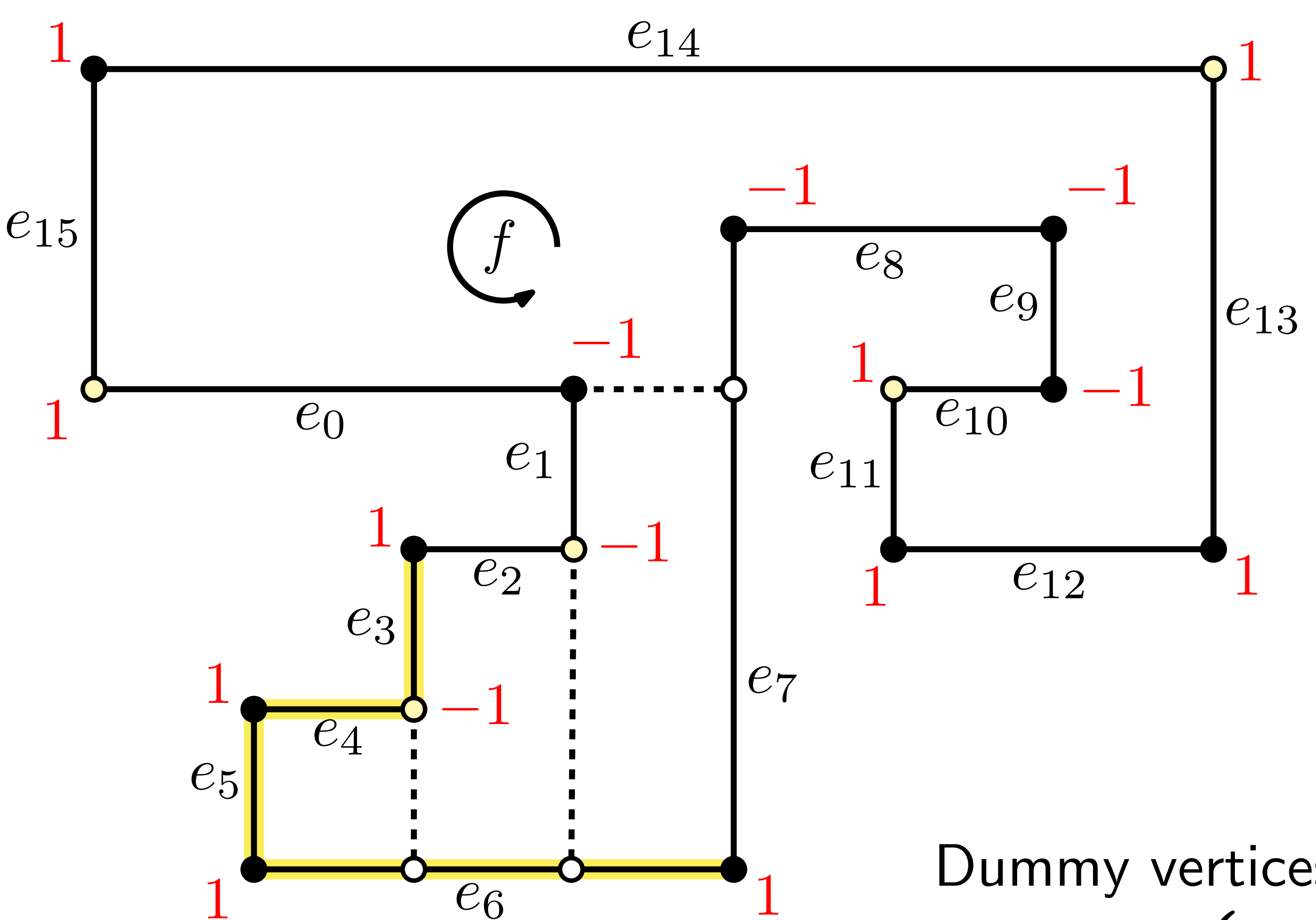
Refinement of (G, H) – Inner Face



Dummy vertices for bends: ○

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

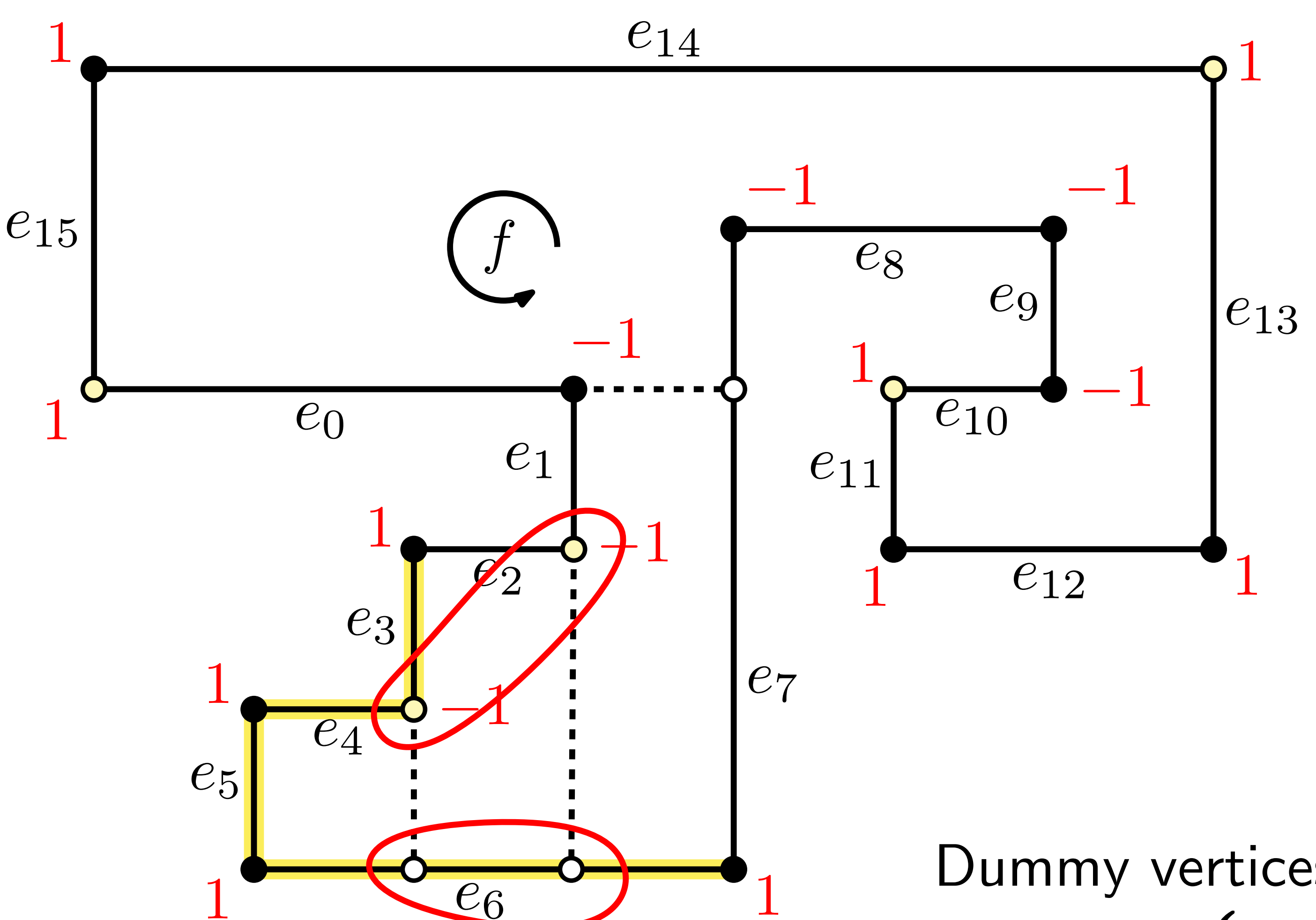
Refinement of (G, H) – Inner Face



Dummy vertices for bends: ○

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

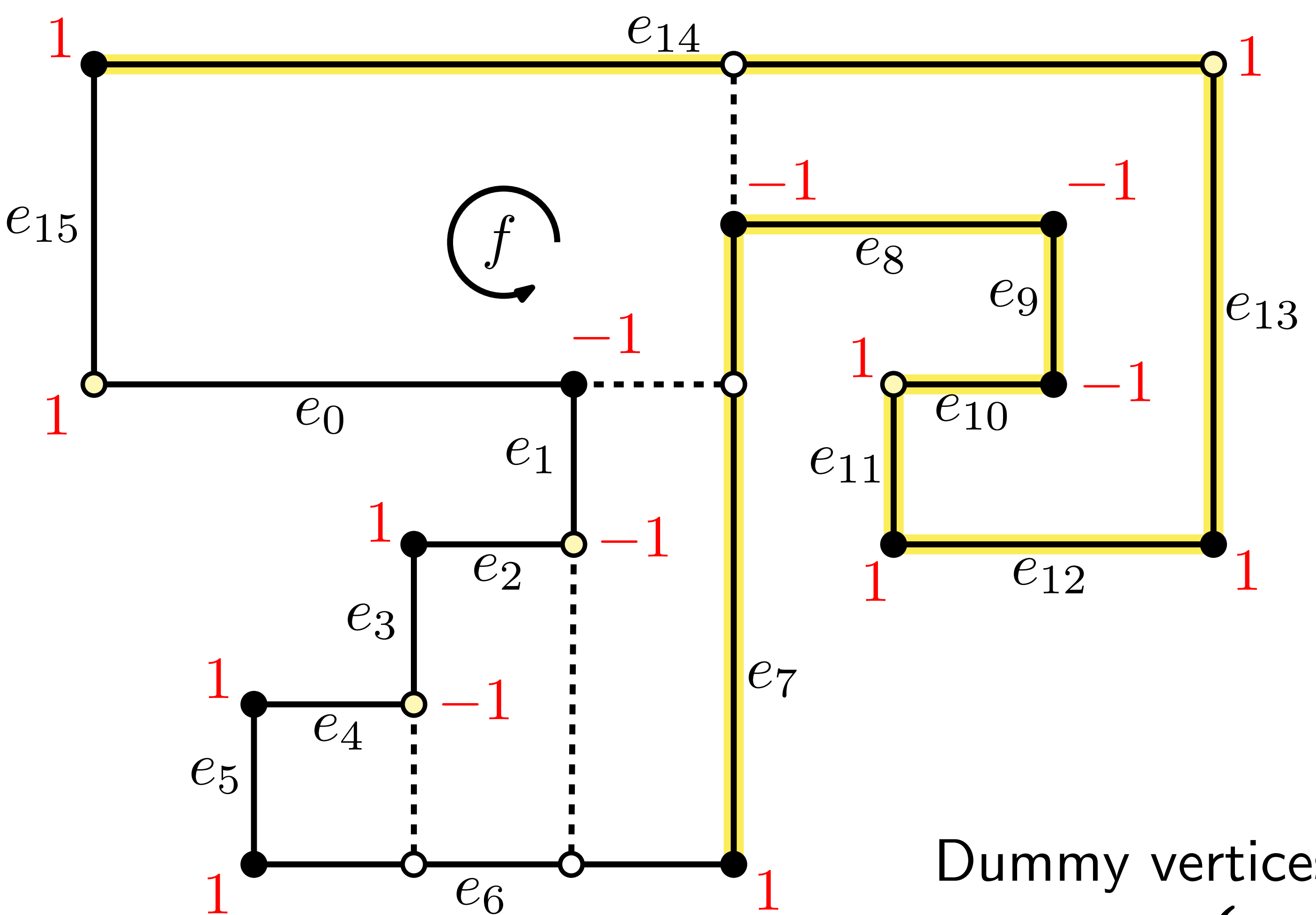
Refinement of (G, H) – Inner Face



Dummy vertices for bends: ○

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

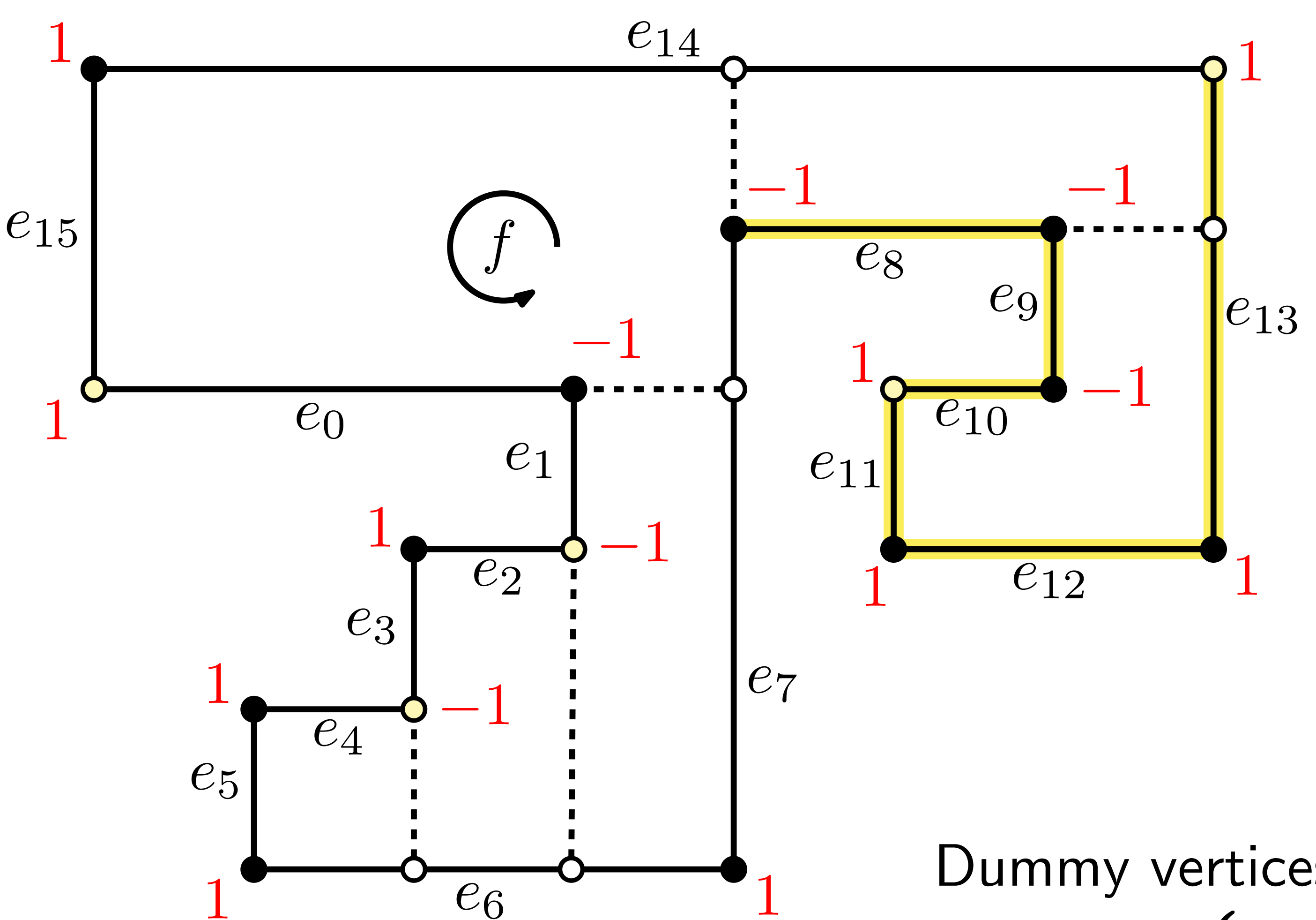
Refinement of (G, H) – Inner Face



Dummy vertices for bends: ○

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

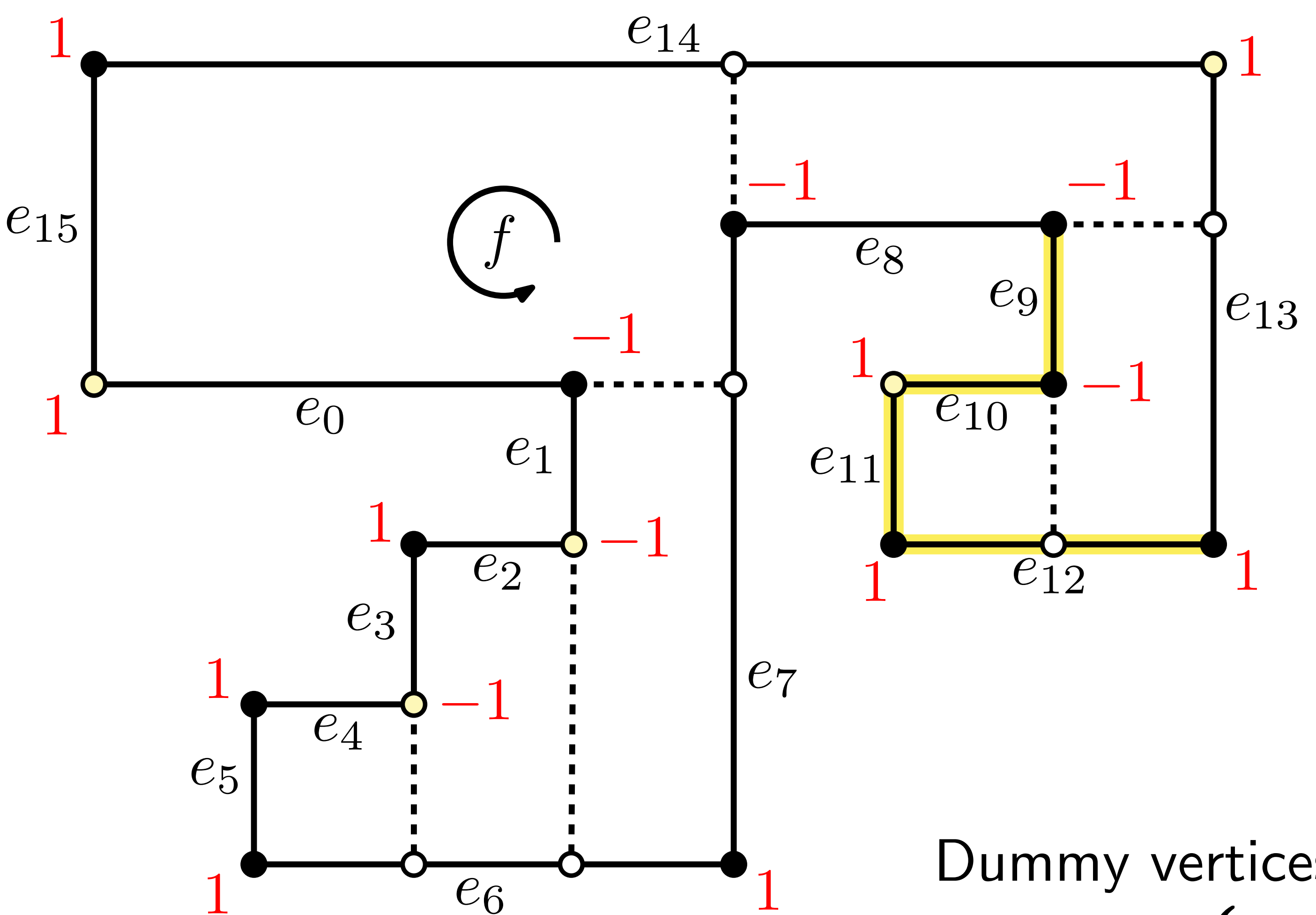
Refinement of (G, H) – Inner Face



Dummy vertices for bends: ○

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

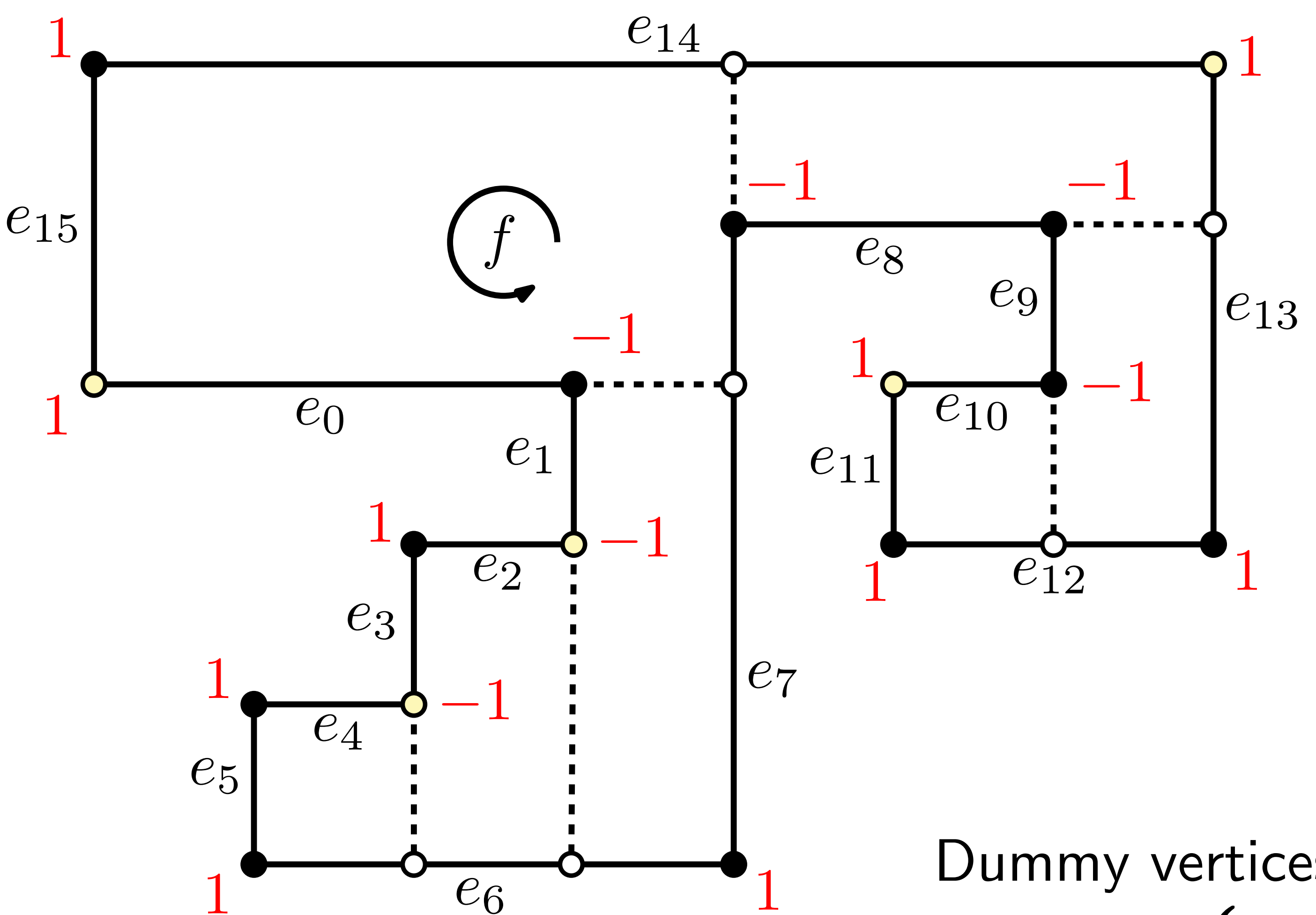
Refinement of (G, H) – Inner Face



Dummy vertices for bends:

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

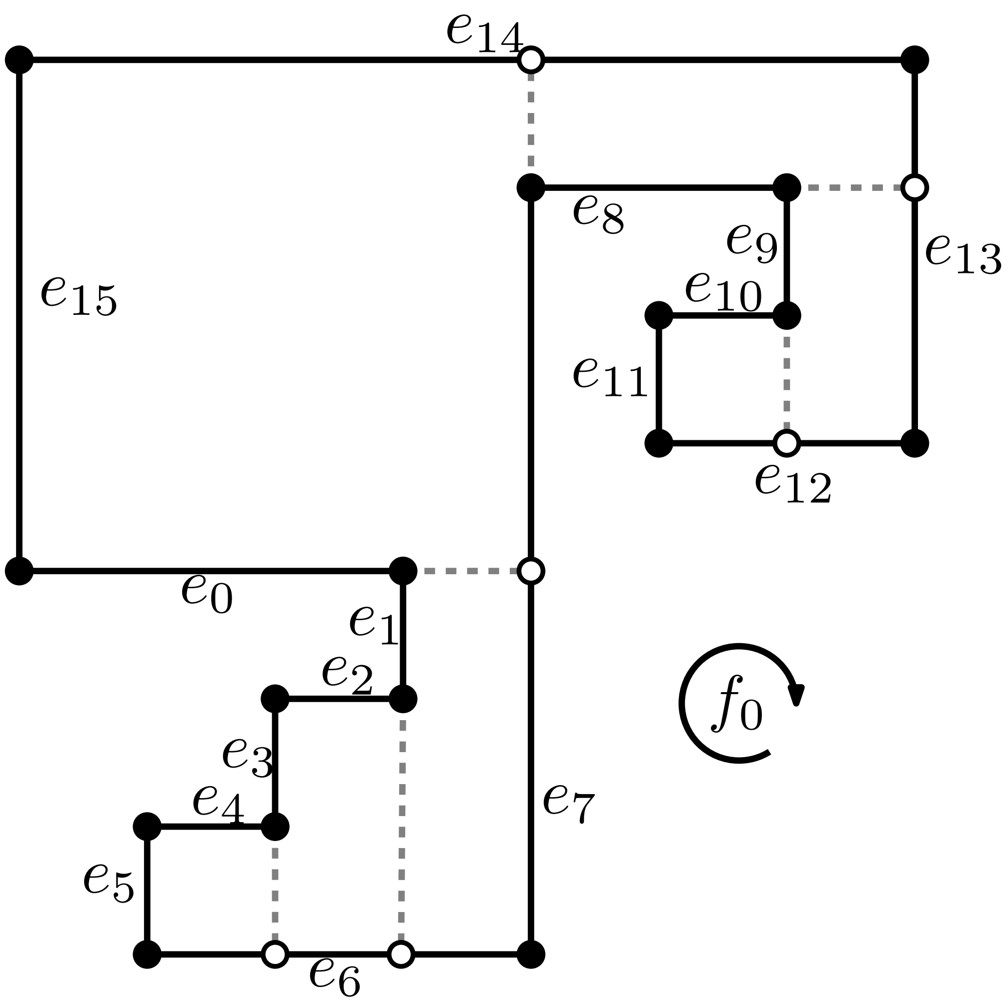
Refinement of (G, H) – Inner Face



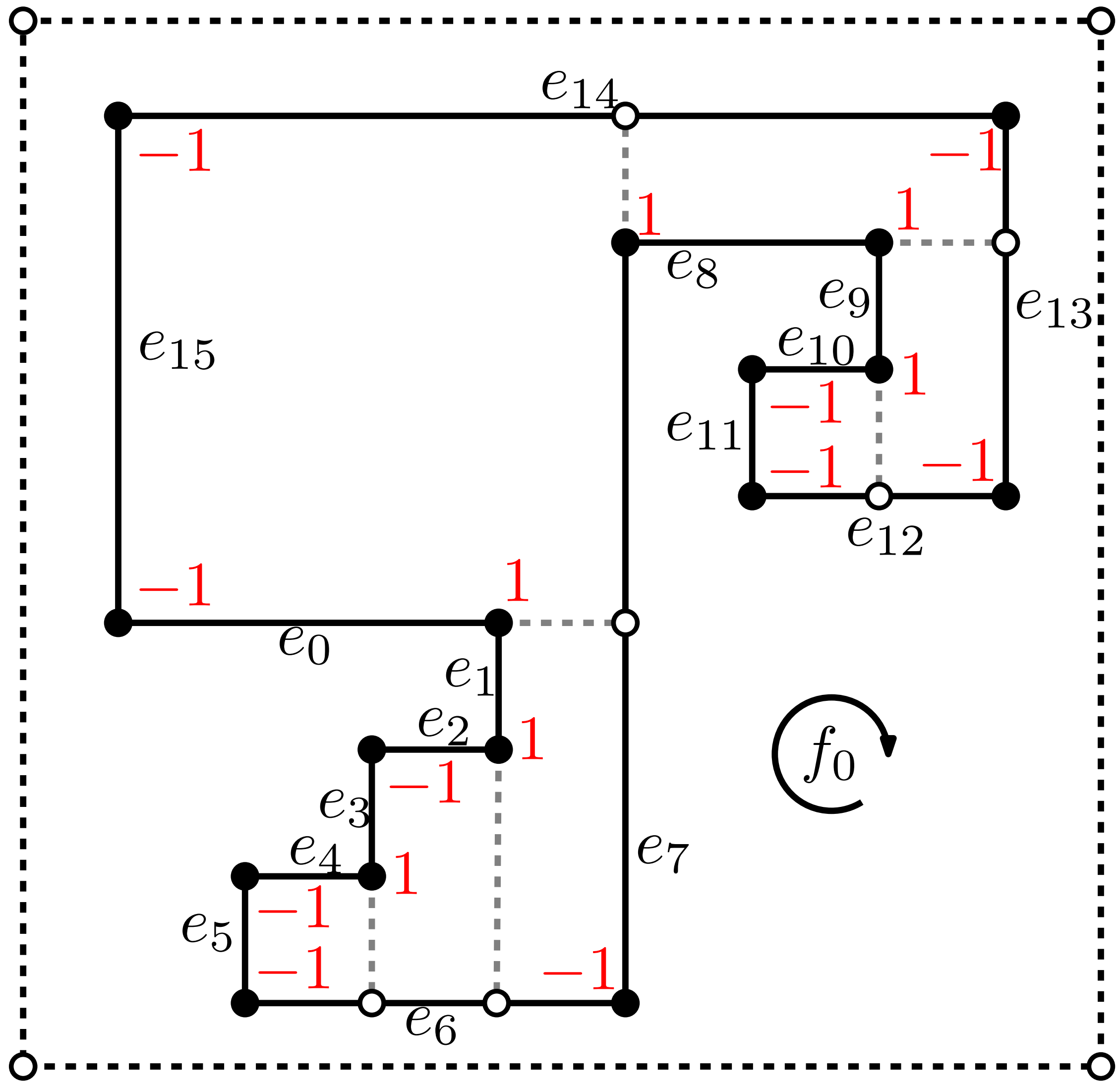
Dummy vertices for bends:

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

Refinement of (G, H) – Outer Face



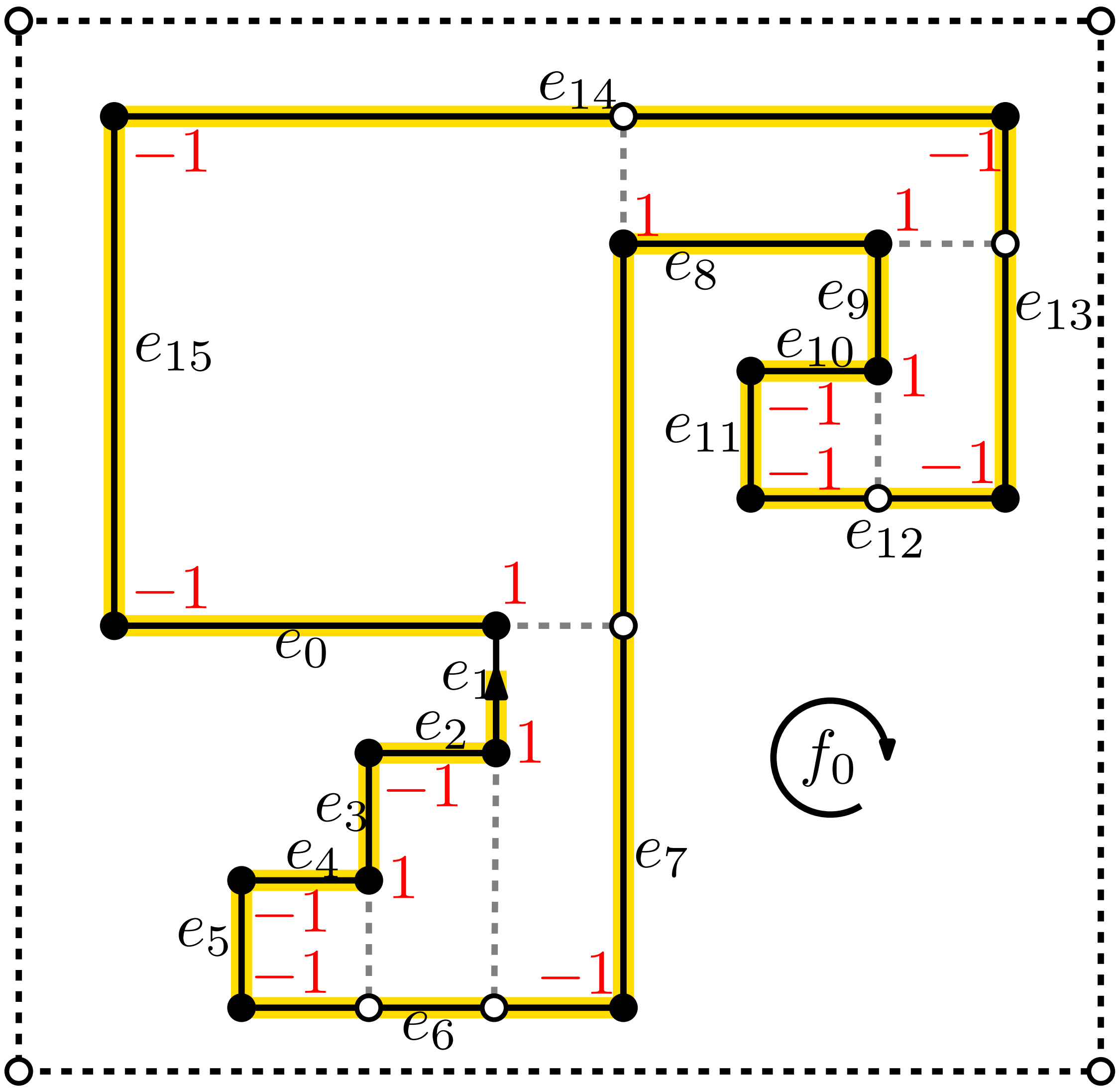
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

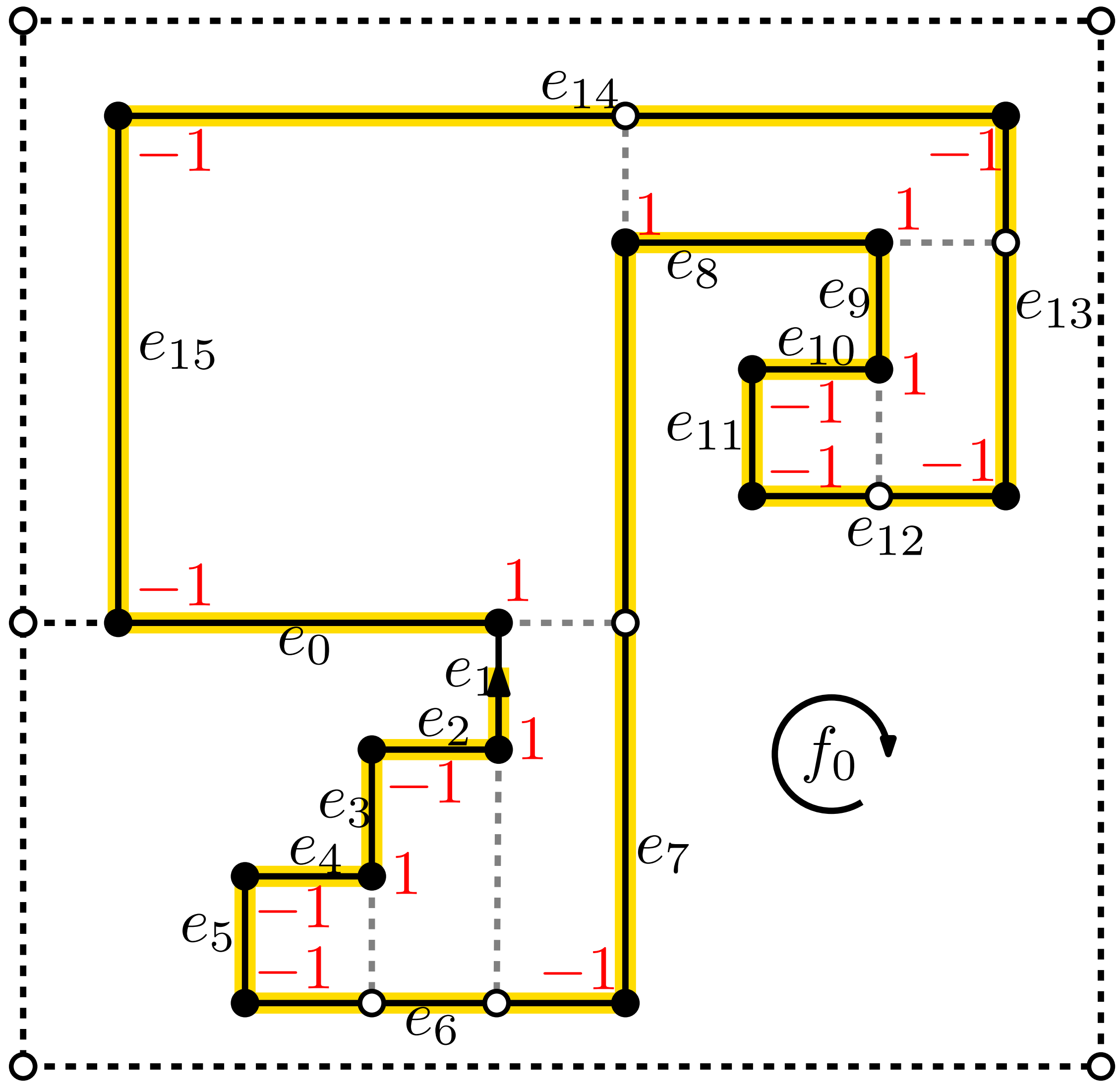
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

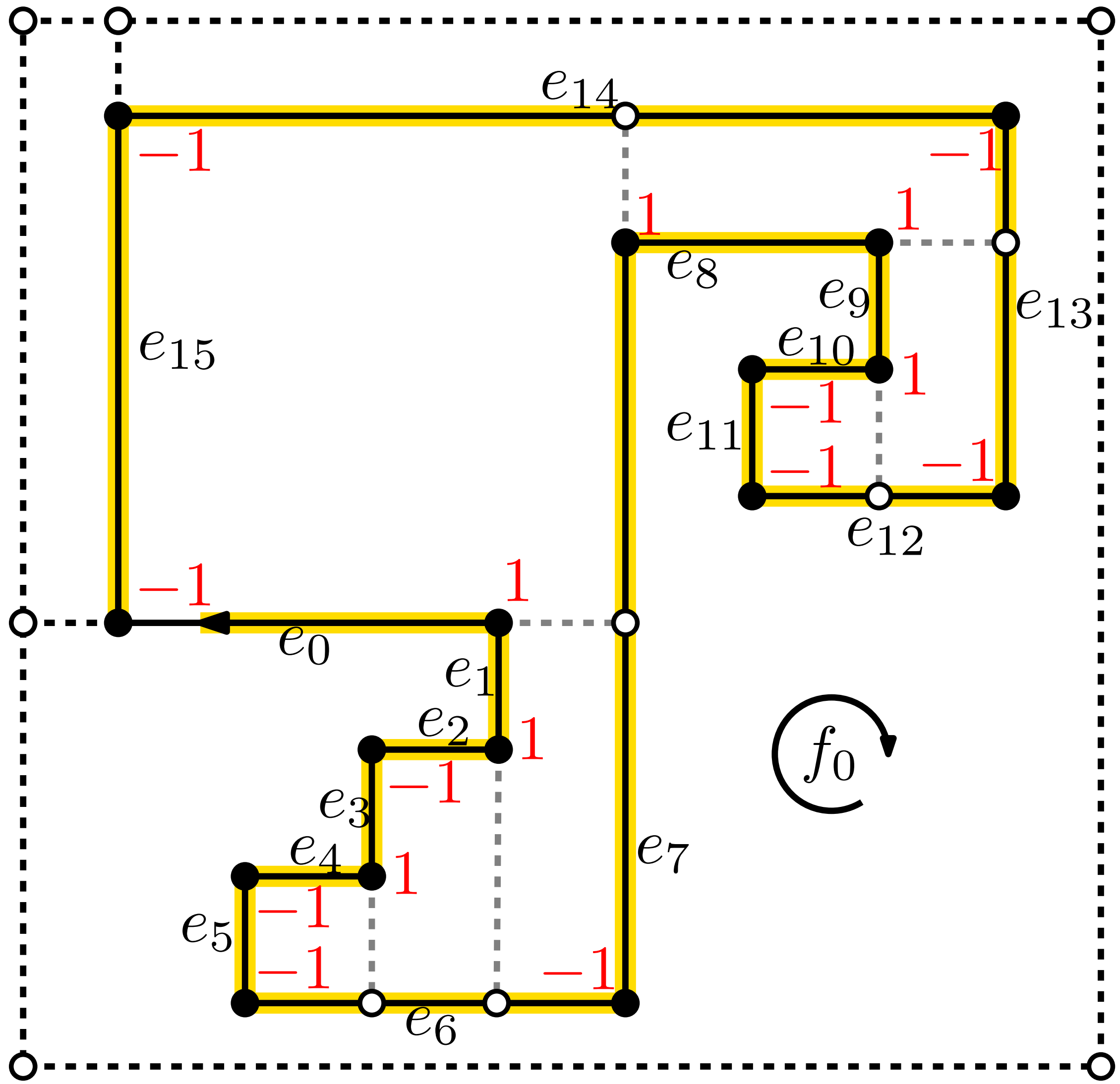
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined
- if $\sum$ turn$(e) < 1$ for the complete boundary of f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

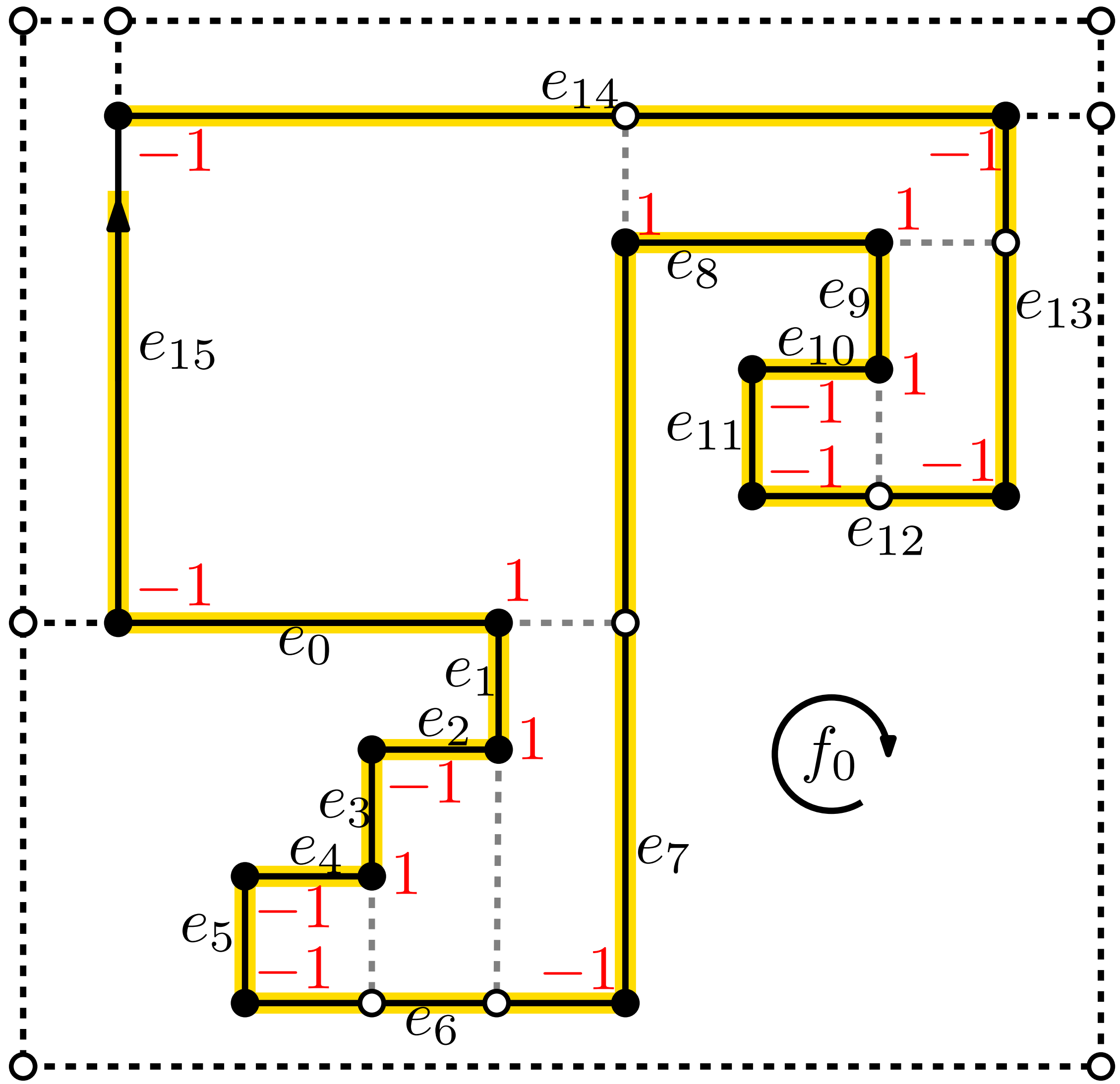
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined
- if $\sum$ turn$(e) < 1$ for the complete boundary of f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

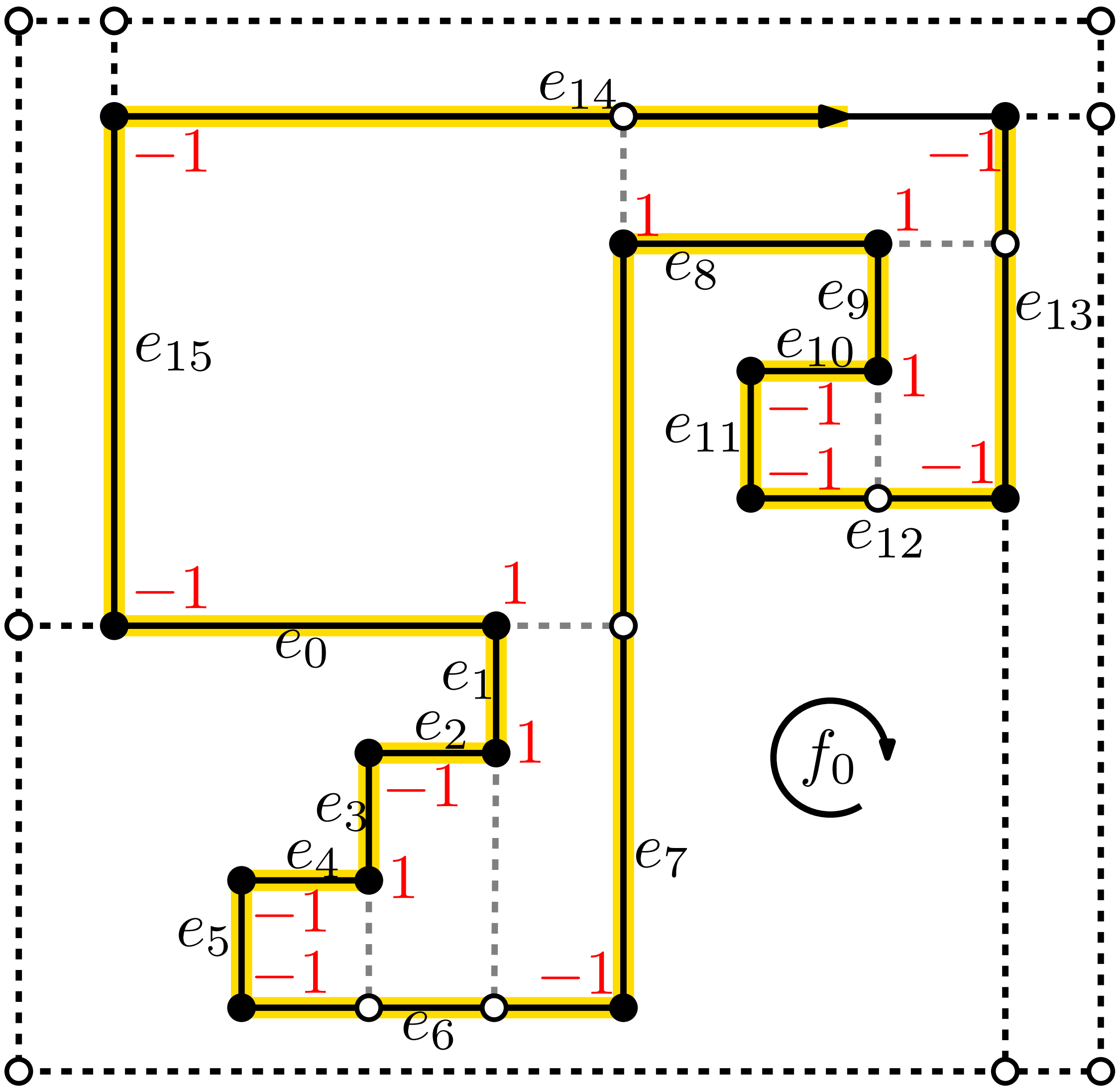
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined
- if $\sum$ turn$(e) < 1$ for the complete boundary of f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

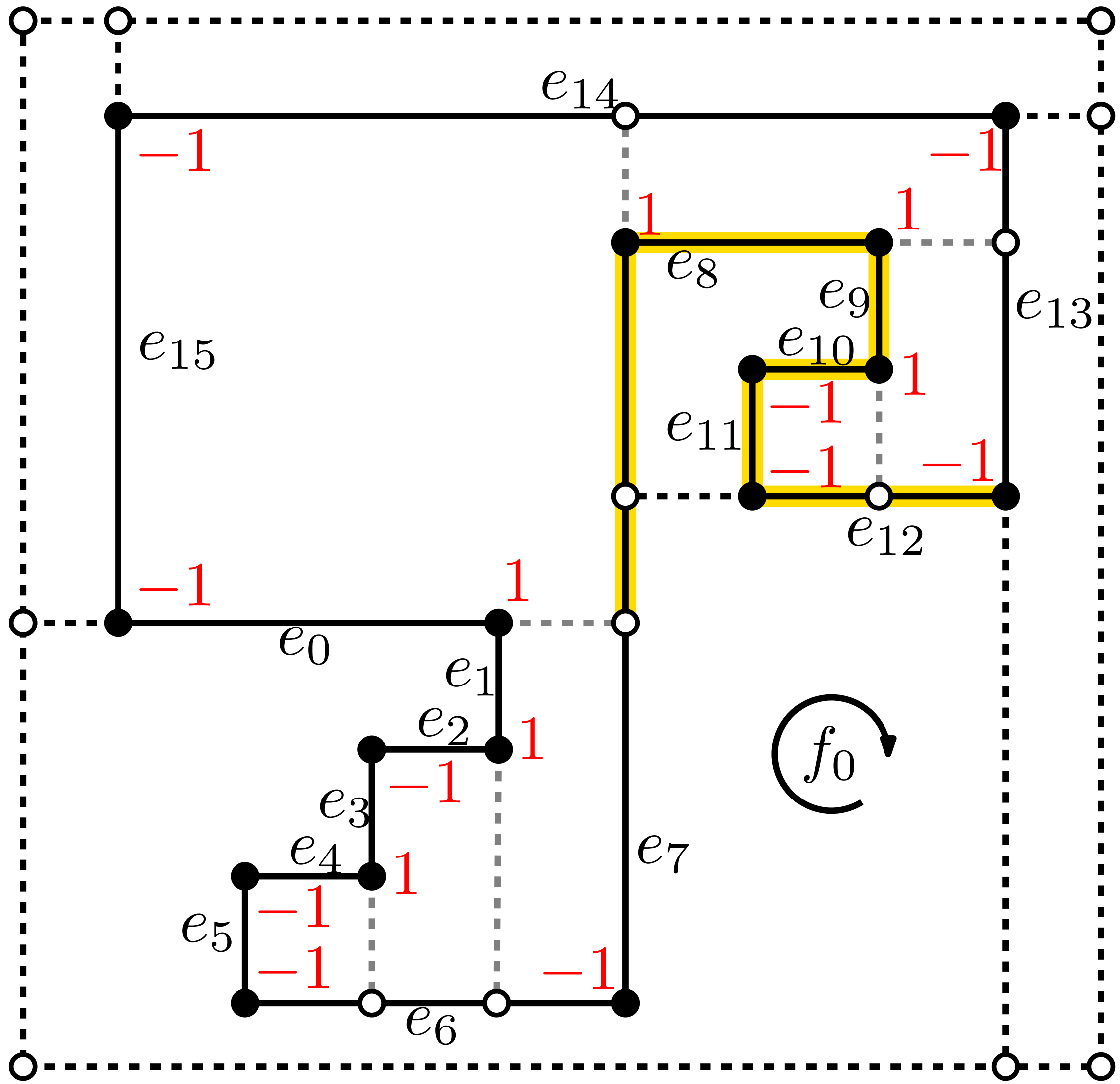
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined
- if $\sum \text{turn}(e) < 1$ for the complete boundary of f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

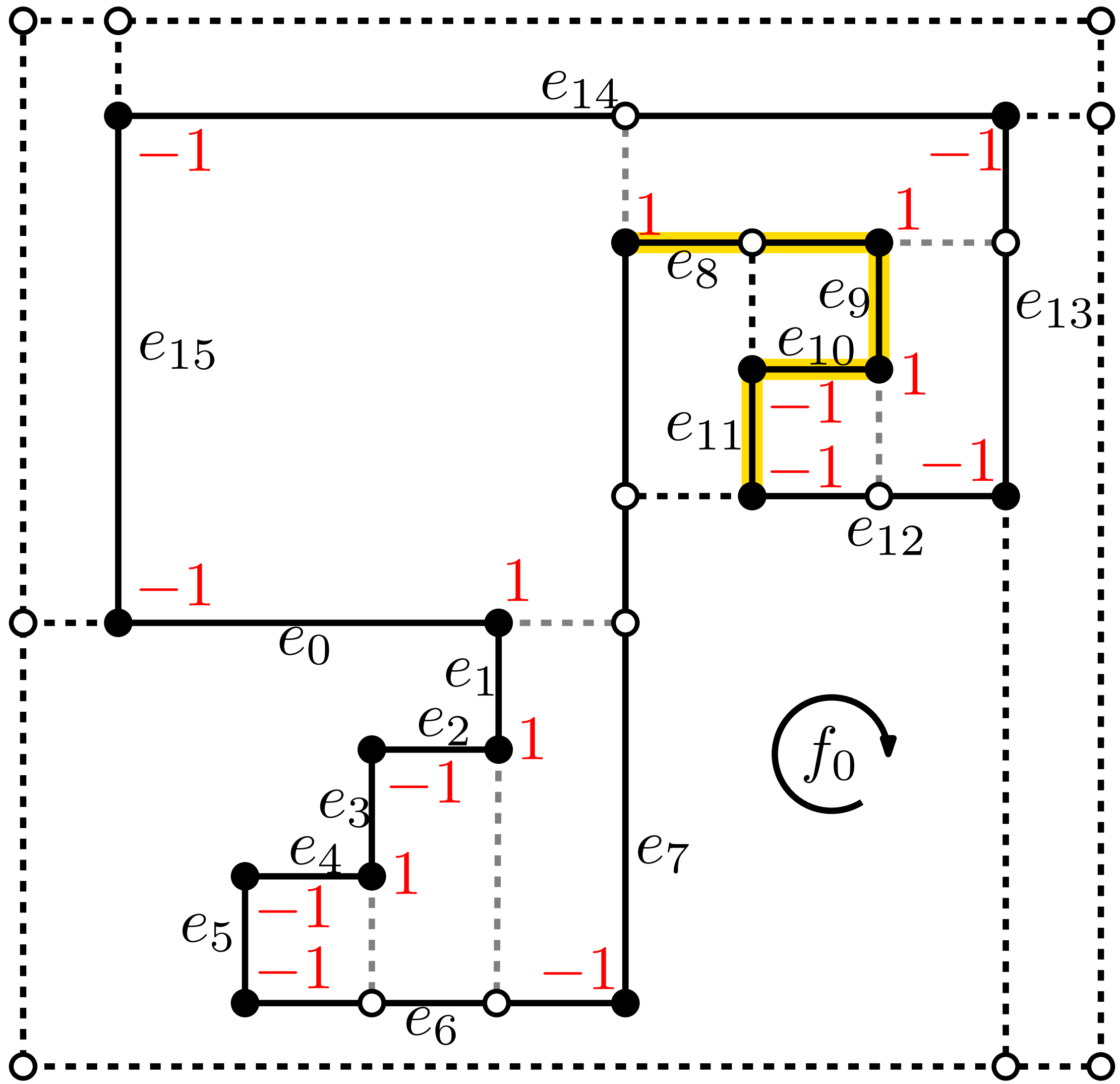
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined
- if $\sum$ turn$(e) < 1$ for the complete boundary of f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

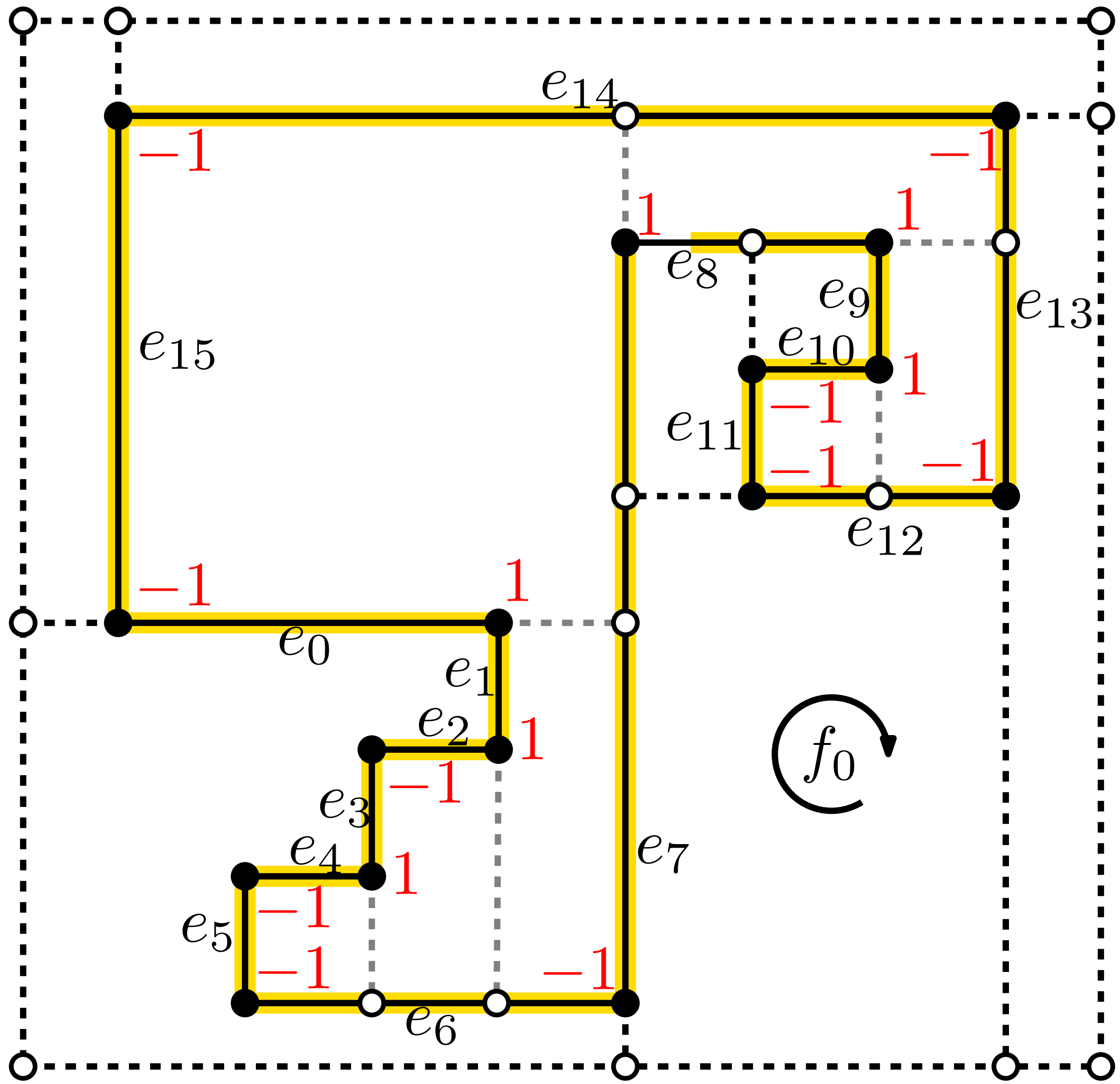
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined
- if $\sum$ turn$(e) < 1$ for the complete boundary of f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

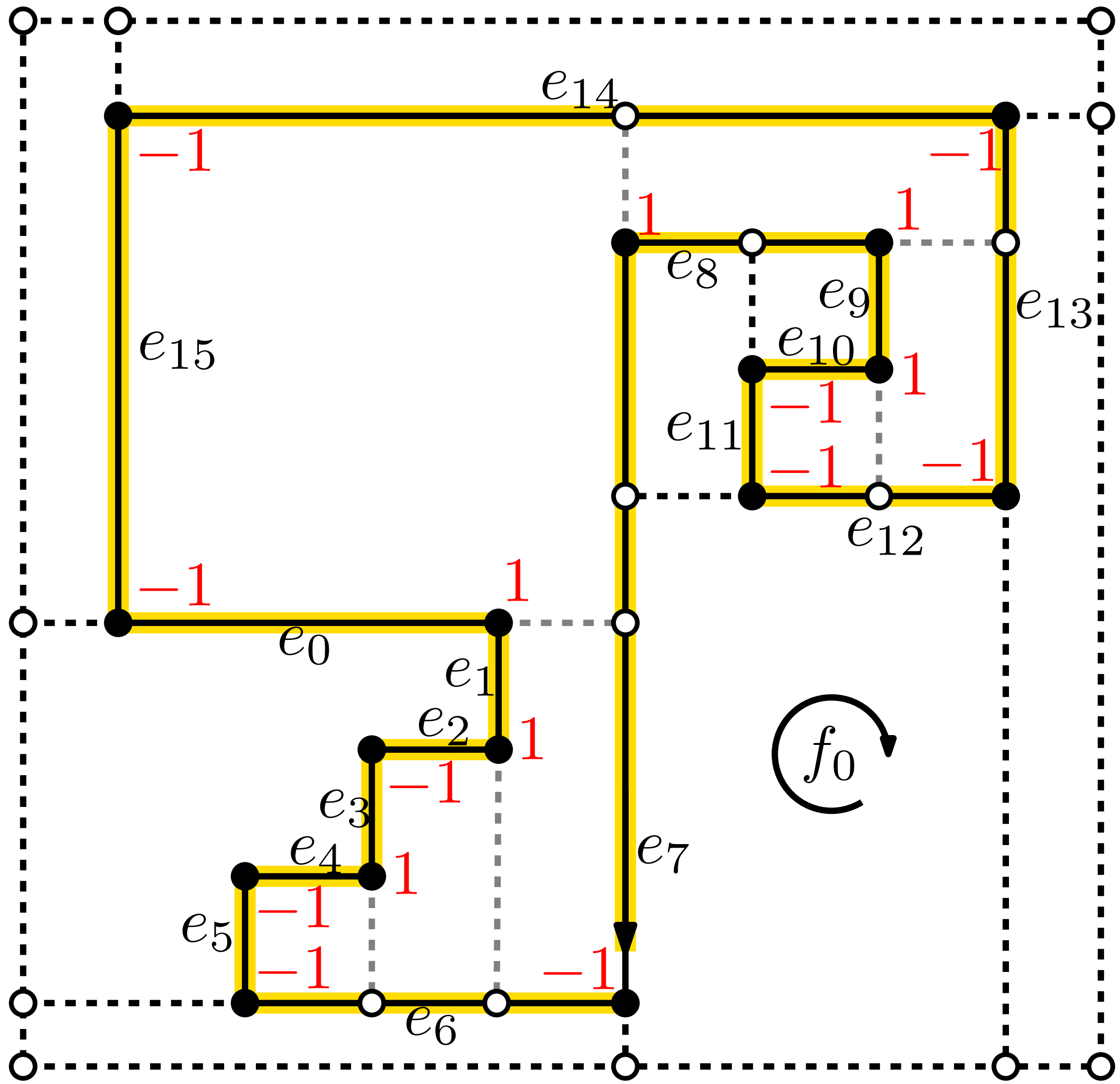
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined
- if $\sum$ turn$(e) < 1$ for the complete boundary of f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

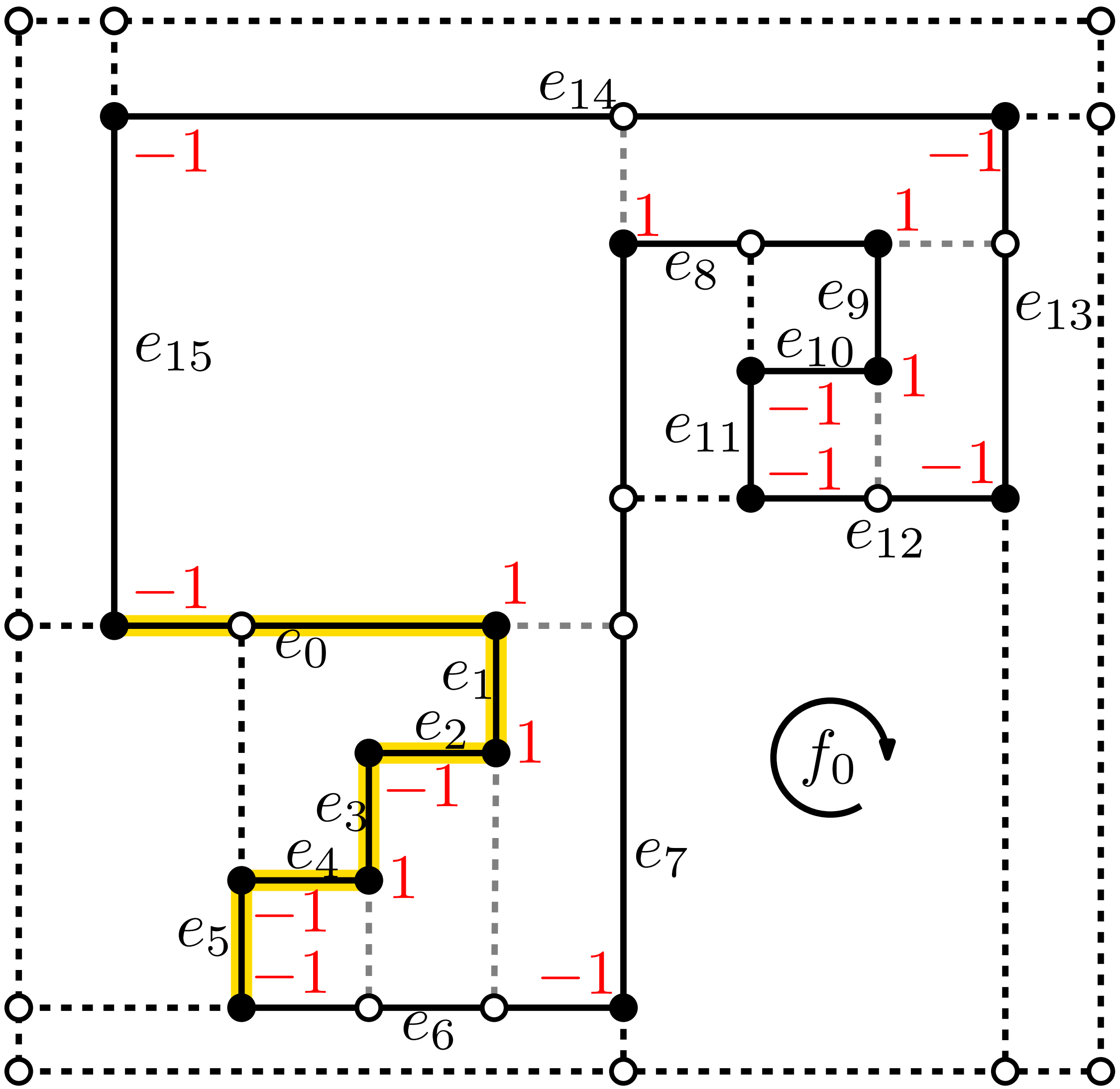
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined
- if $\sum$ turn$(e) < 1$ for the complete boundary of f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

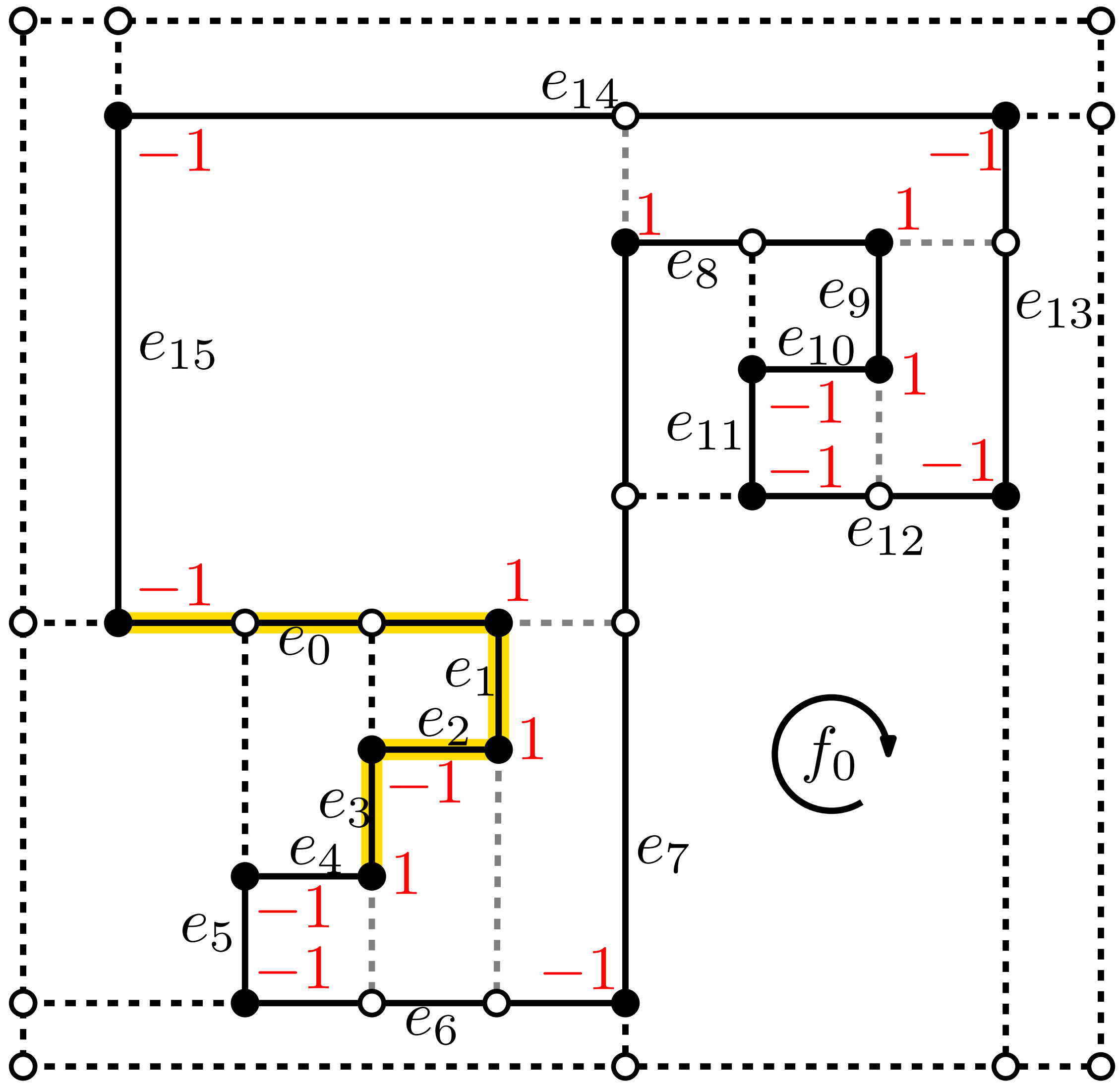
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined
- if $\sum$ turn$(e) < 1$ for the complete boundary of f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

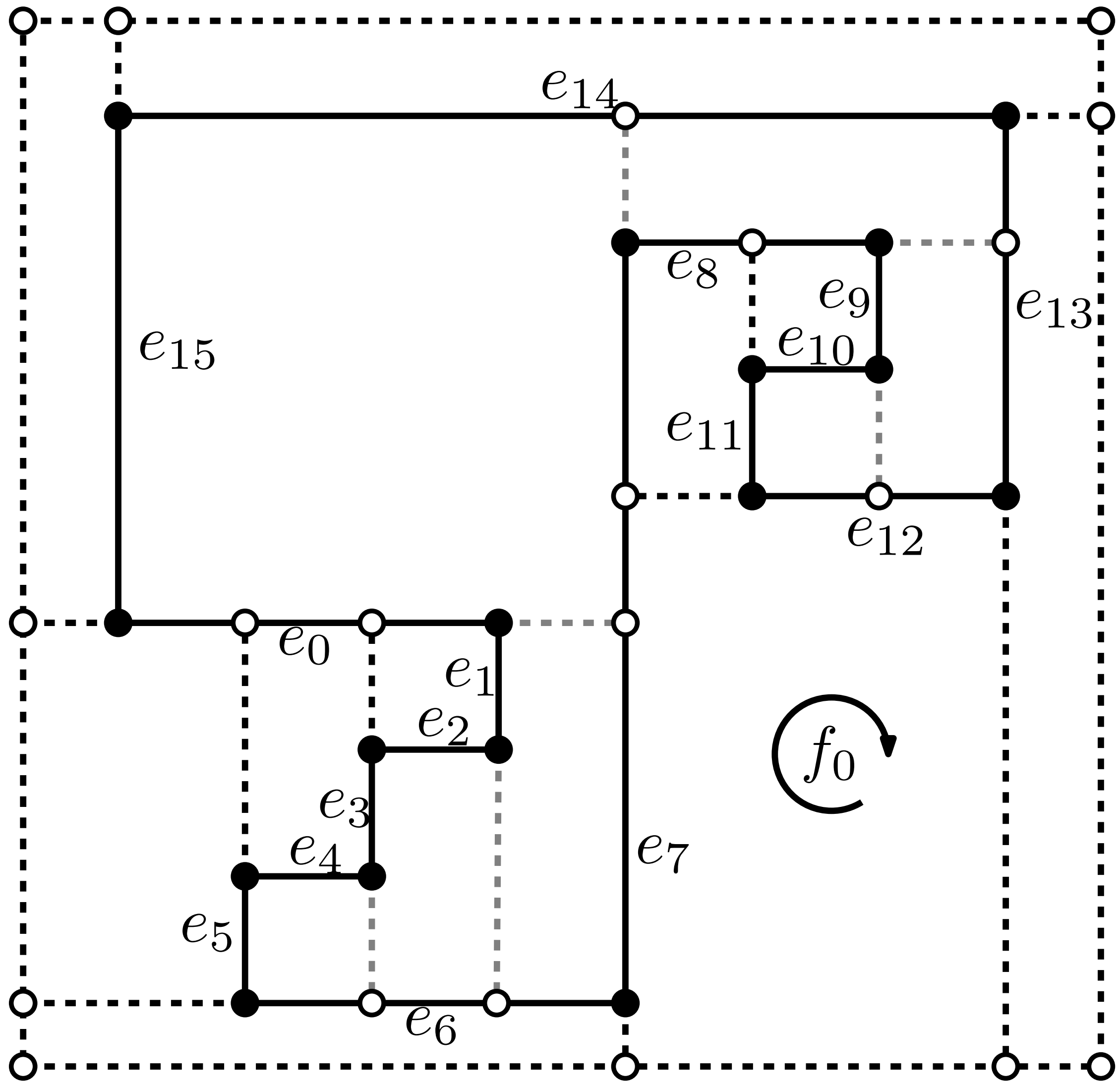
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined
- if $\sum$ turn$(e) < 1$ for the complete boundary of f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

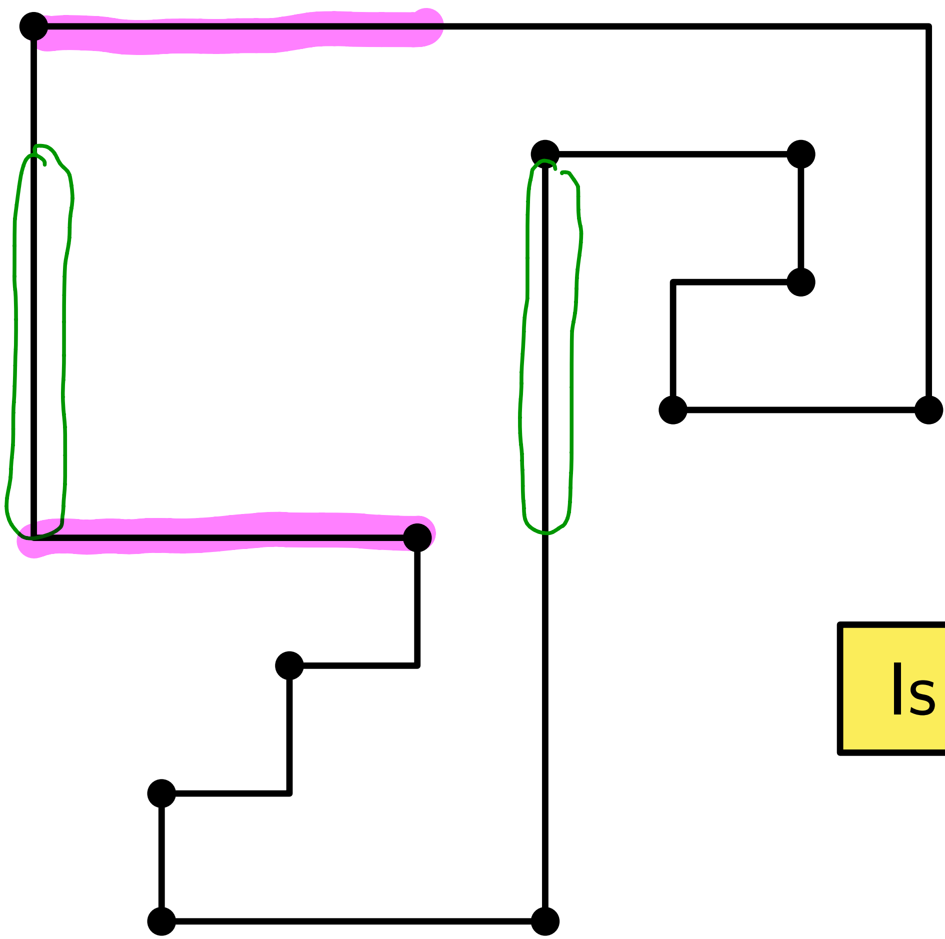
Refinement of (G, H) – Outer Face



- add bounding rectangle R
- front(e) may be undefined
- if $\sum$ turn$(e) < 1$ for the complete boundary of f_0, project on R

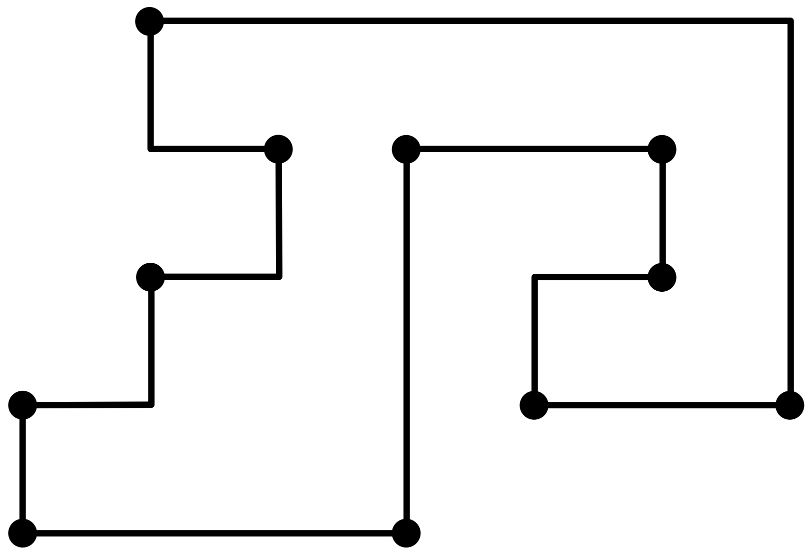
all faces are rectangles $\rightarrow$ can apply flow network for rectangular faces

Refinement of (G, H) – Outer Face



Is the area minimal?

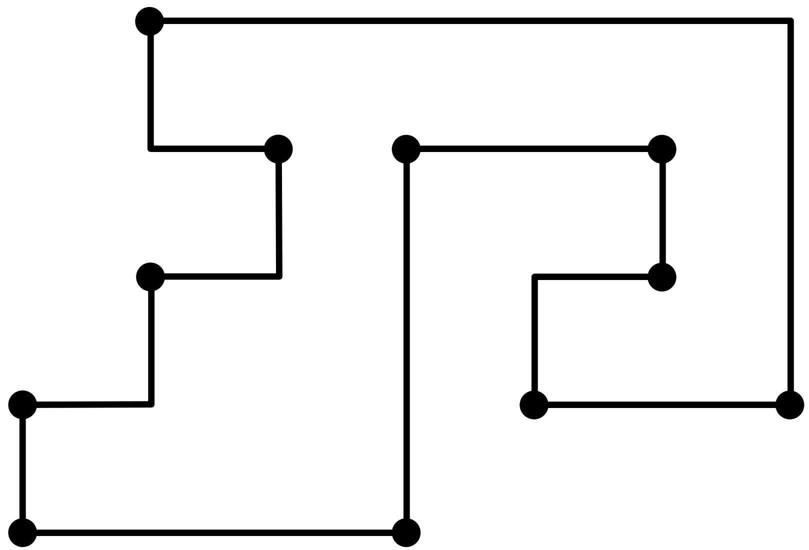
Refinement of (G, H) – Outer Face



Is the area minimal?

NO!

Refinement of (G, H) – Outer Face



Is the area minimal?

NO!

Area Minimization with a given orthogonal representation is an NP-hard problem!

Complexity of Layout Compaction

Theorem: For a graph G with orthogonal representation $H(G)$ and some $K \in \mathbb{N}$ it is NP-complete to decide if $(G, H(G))$ has a grid layout of area at most K.

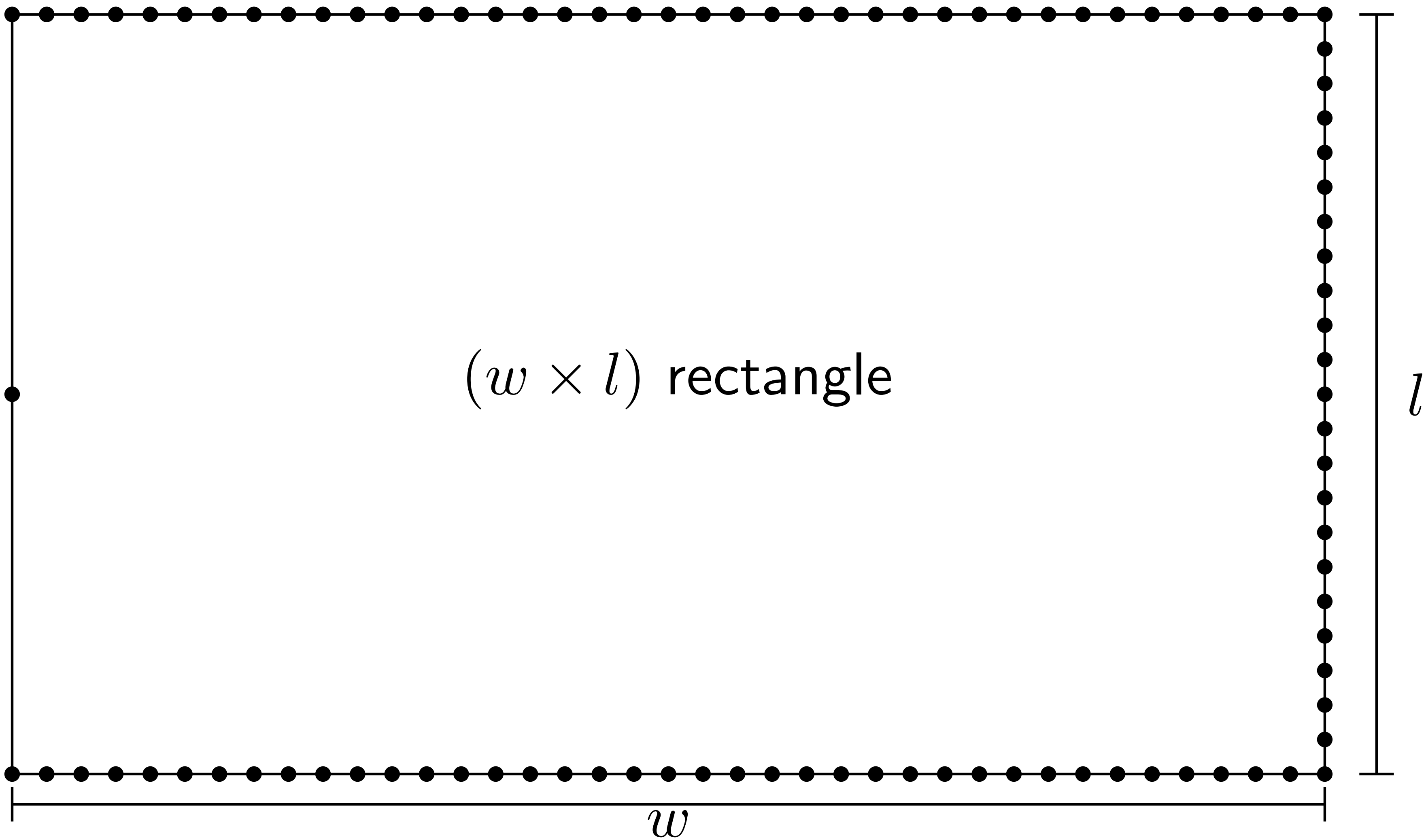
Complexity of Layout Compaction

Theorem: For a graph G with orthogonal representation $H(G)$ and some $K \in \mathbb{N}$ it is NP-complete to decide if $(G, H(G))$ has a grid layout of area at most K.

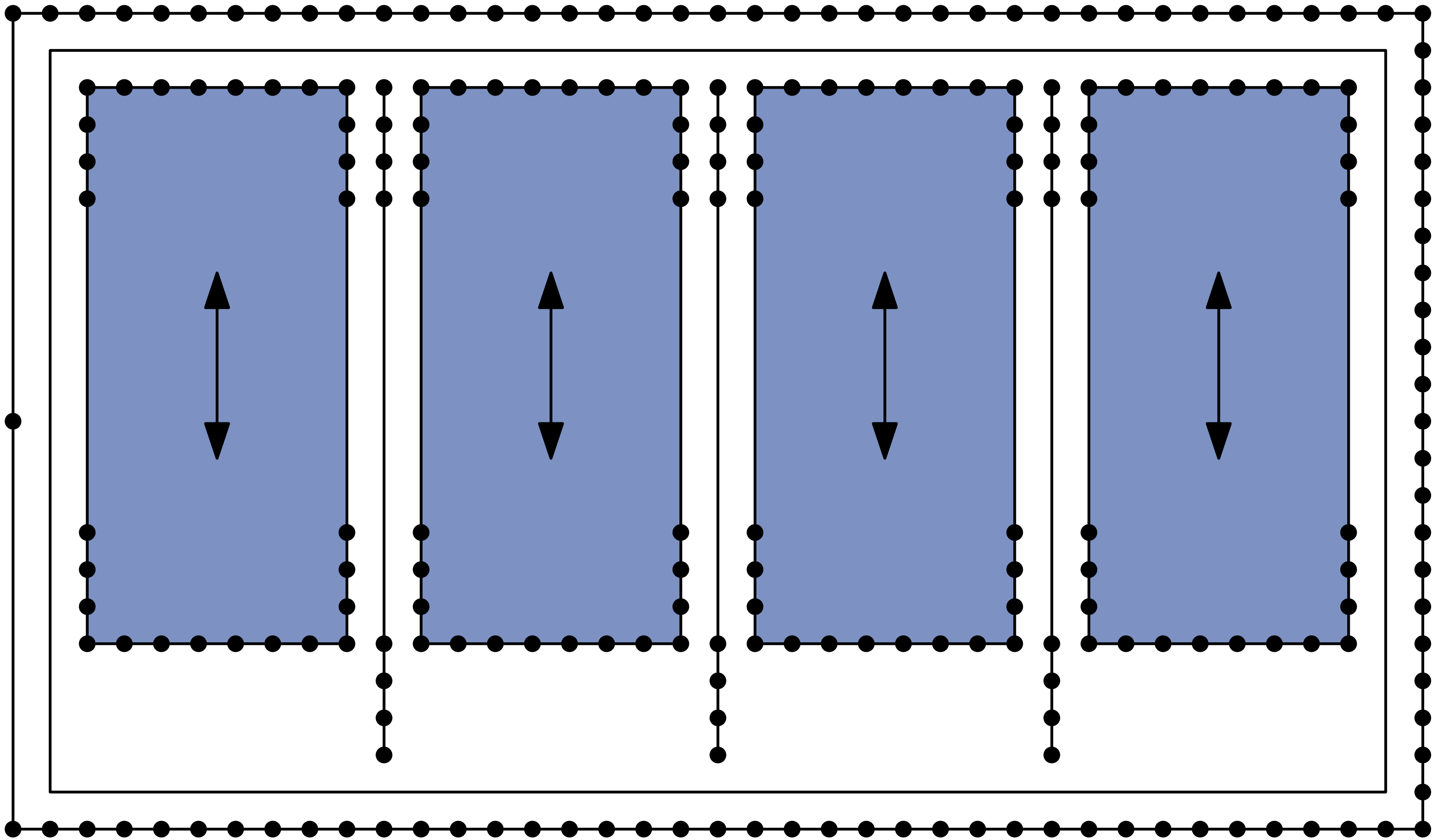
Proof (sketch):

- Reduction from SAT, i.e. satisfiability of a Boolean formula $\Phi = c_1 \wedge \cdots \wedge c_m$ in CNF, where each c_i is a clause on the set of variables $\{x_1, \ldots, x_n\}$
- geometric structure of the reduction
  - clause gadgets
  - variable gadgets
- compute suitable K, such that (G, H) has layout of area at most K iff Φ is satisfiable

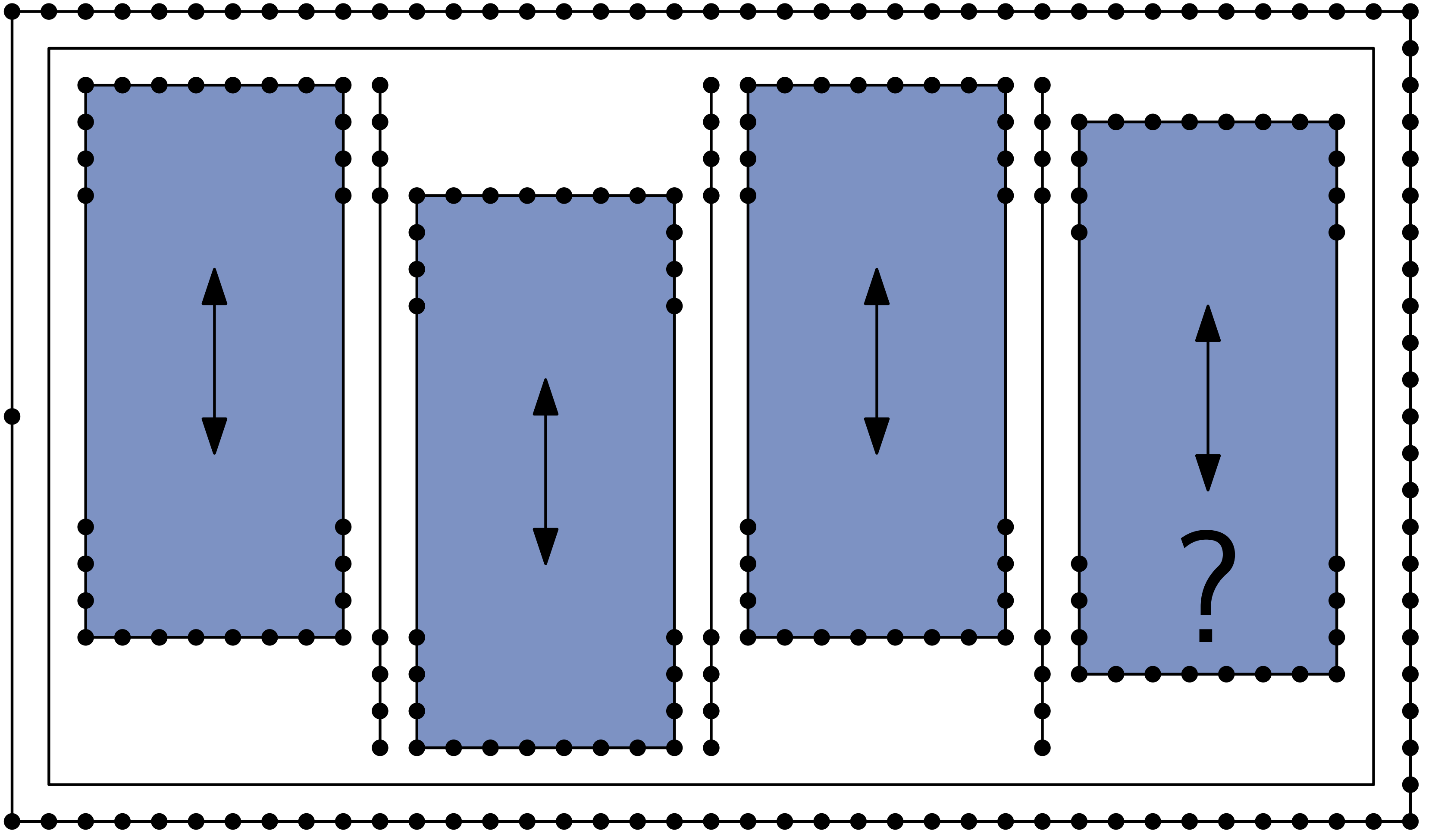
Variable Gadgets



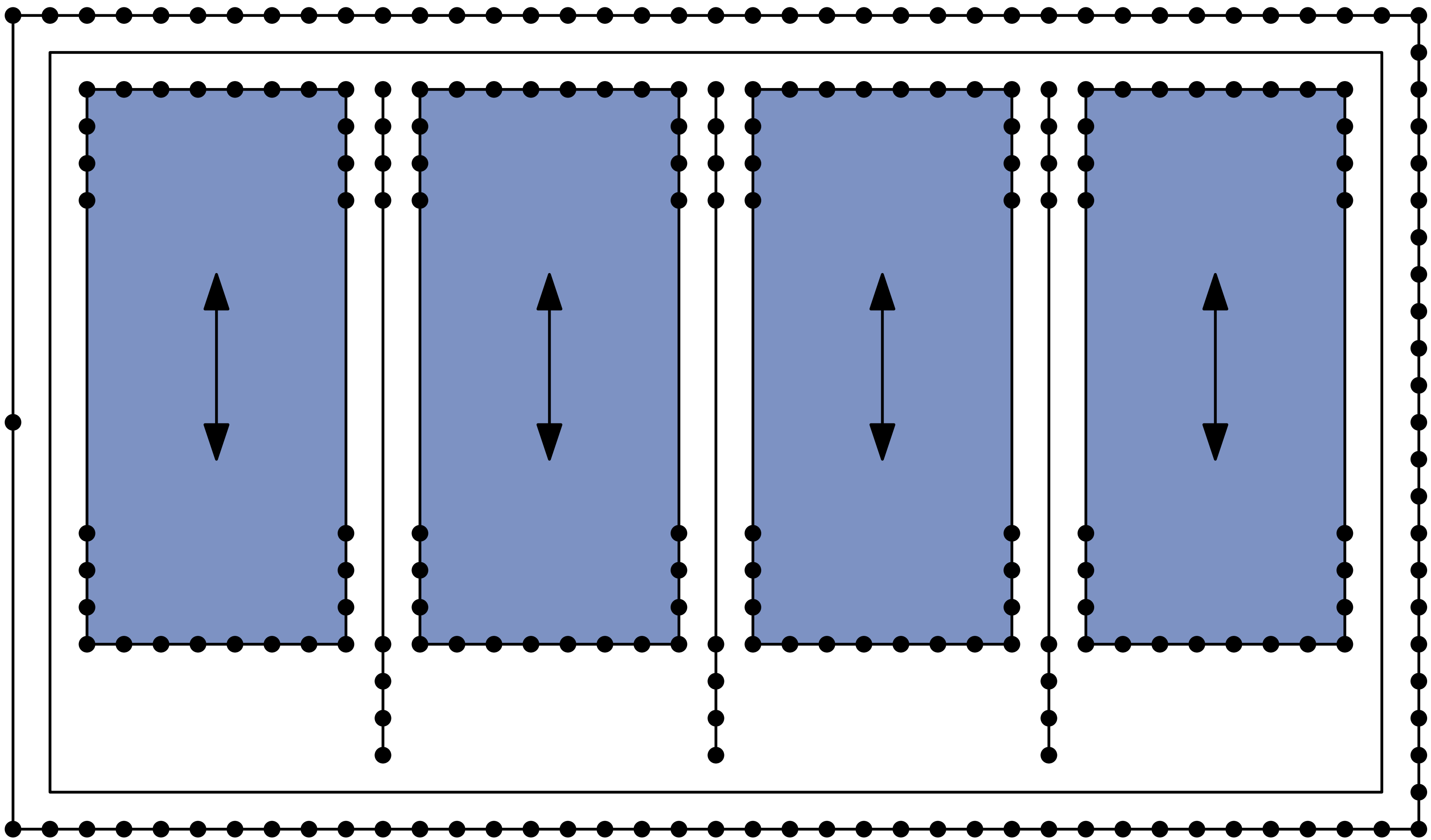
Variable Gadgets



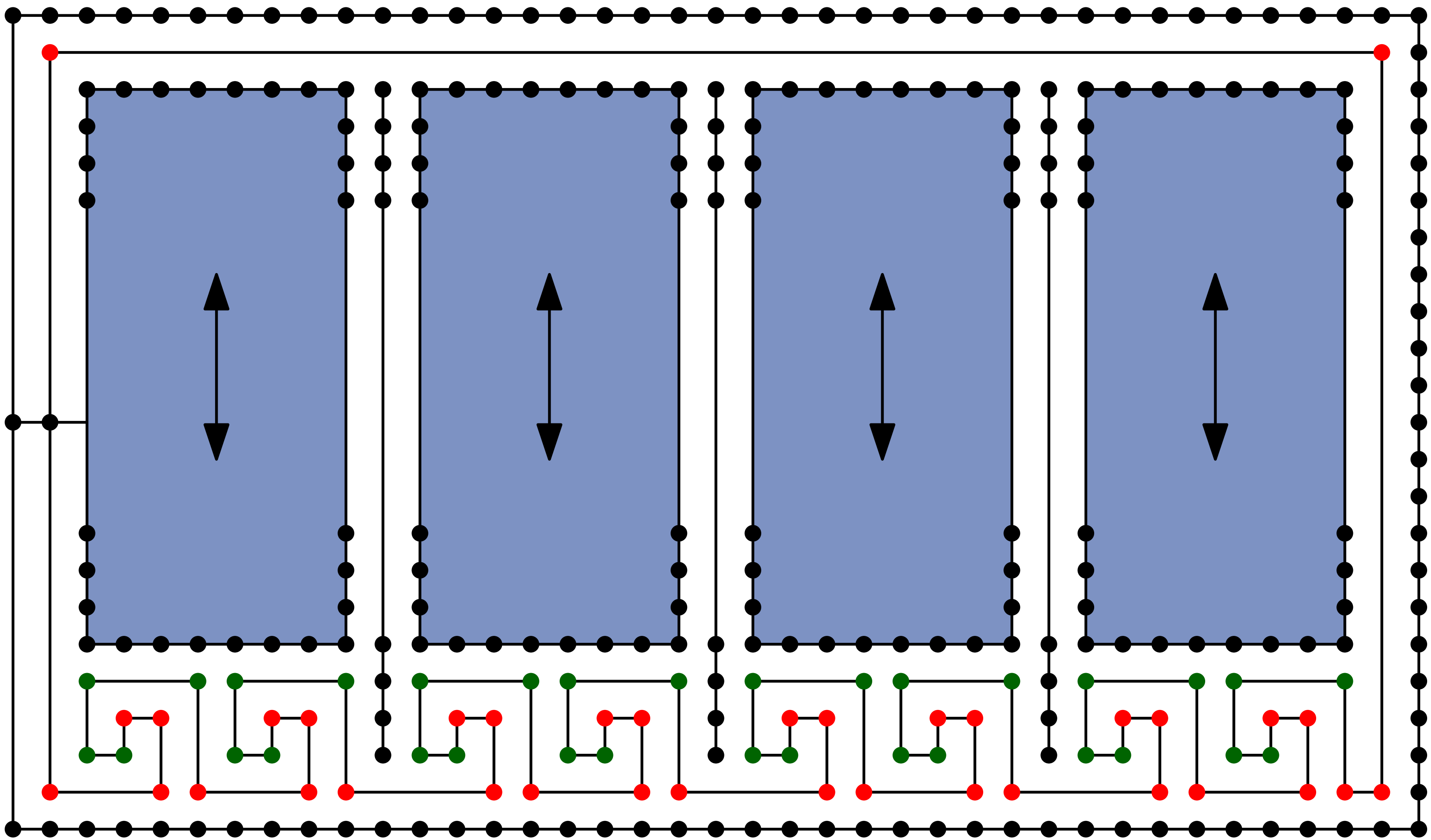
Variable Gadgets



Variable Gadgets



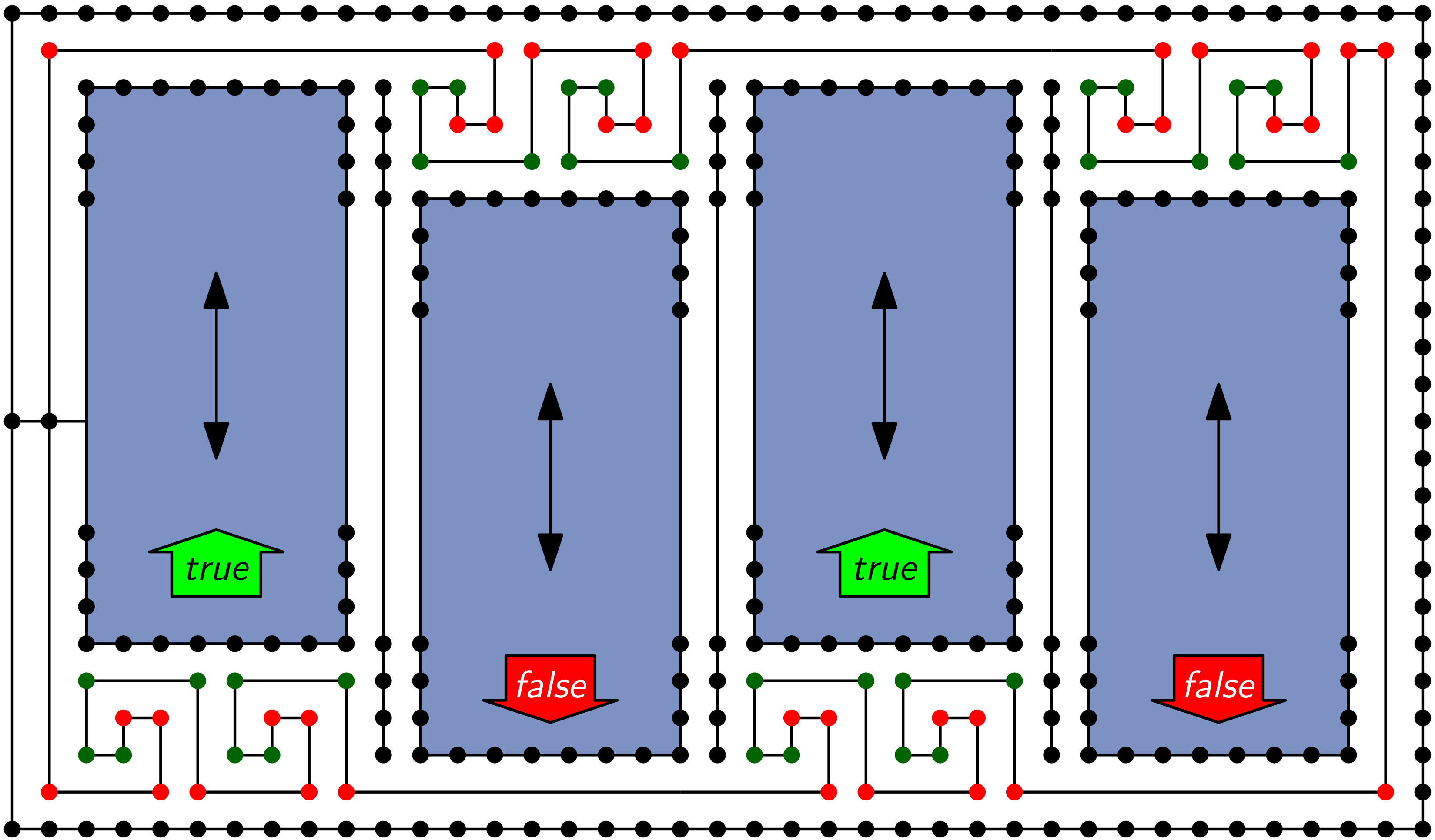
Variable Gadgets



Belt is an edge with orthogonal representation $(r^4 l^4)^{2n} r^4$

- cells of the belt can be shifted within the frame

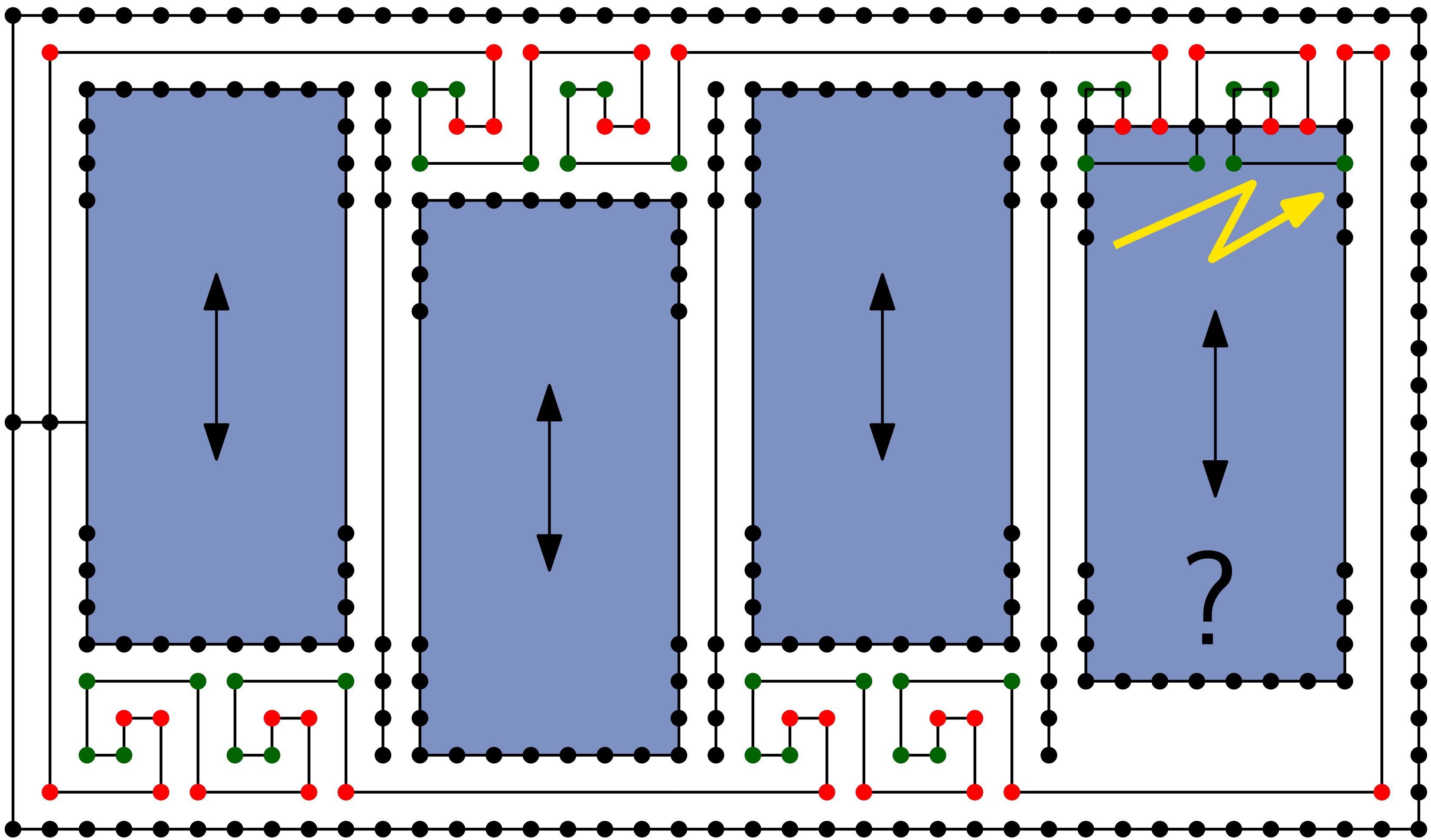
Variable Gadgets



Belt is an edge with orthogonal representation $(r^4 l^4)^{2n} r^4$

- cells of the belt can be shifted within the frame
- each piston represents a variable with two possible states

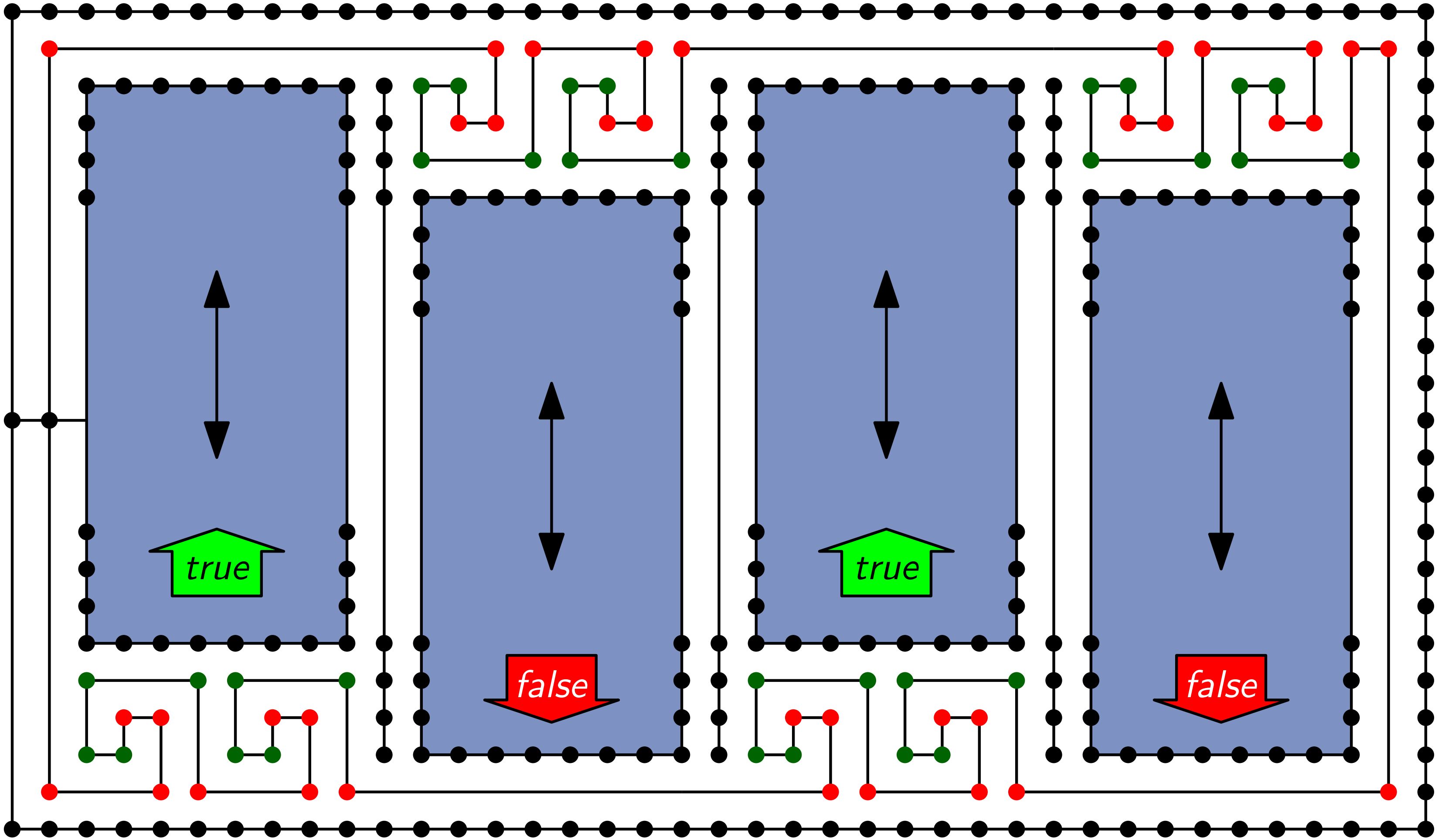
Variable Gadgets



Belt is an edge with orthogonal representation $(r^4 l^4)^{2n} r^4$

- cells of the belt can be shifted within the frame
- each piston represents a variable with two possible states

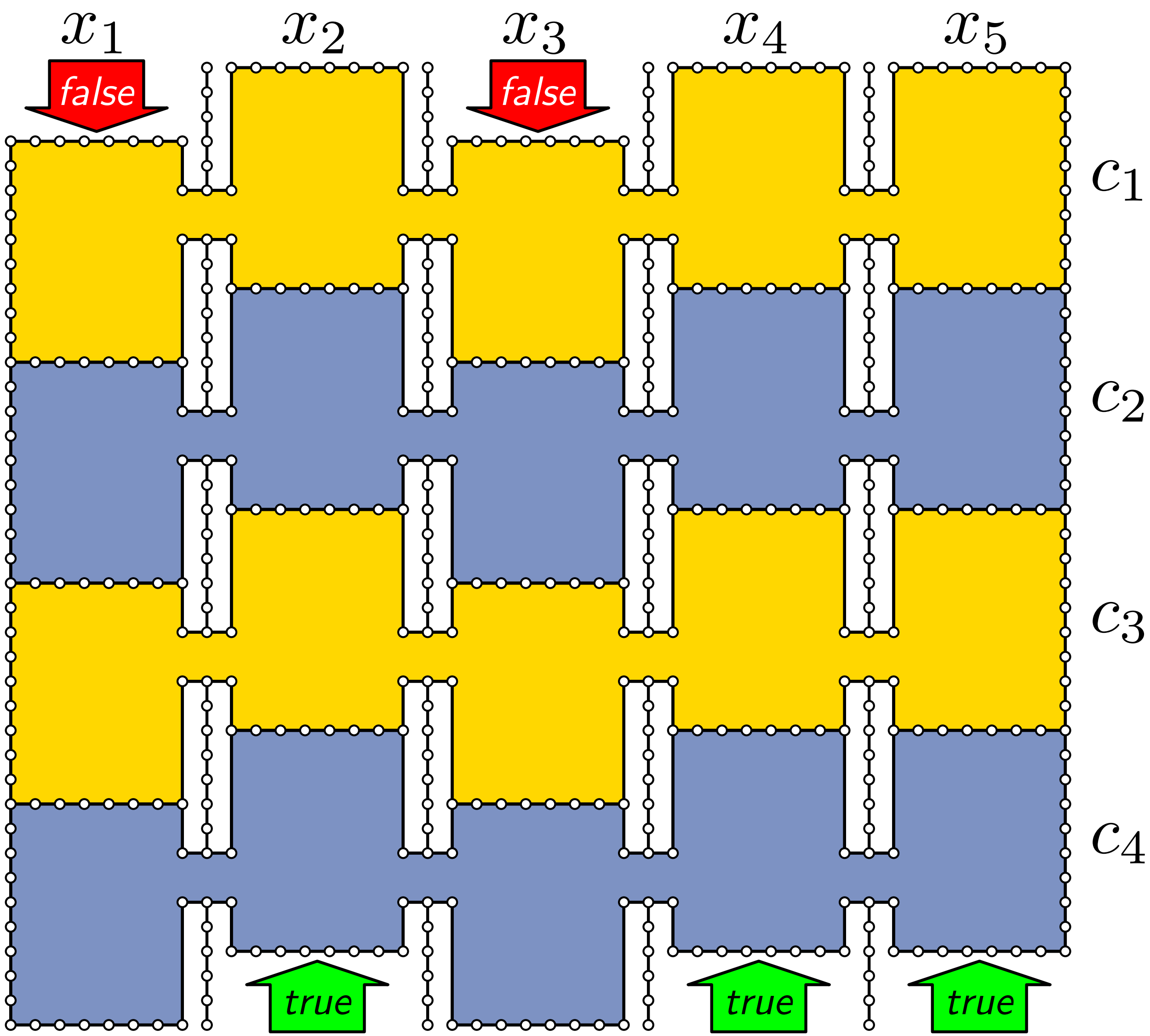
Variable Gadgets



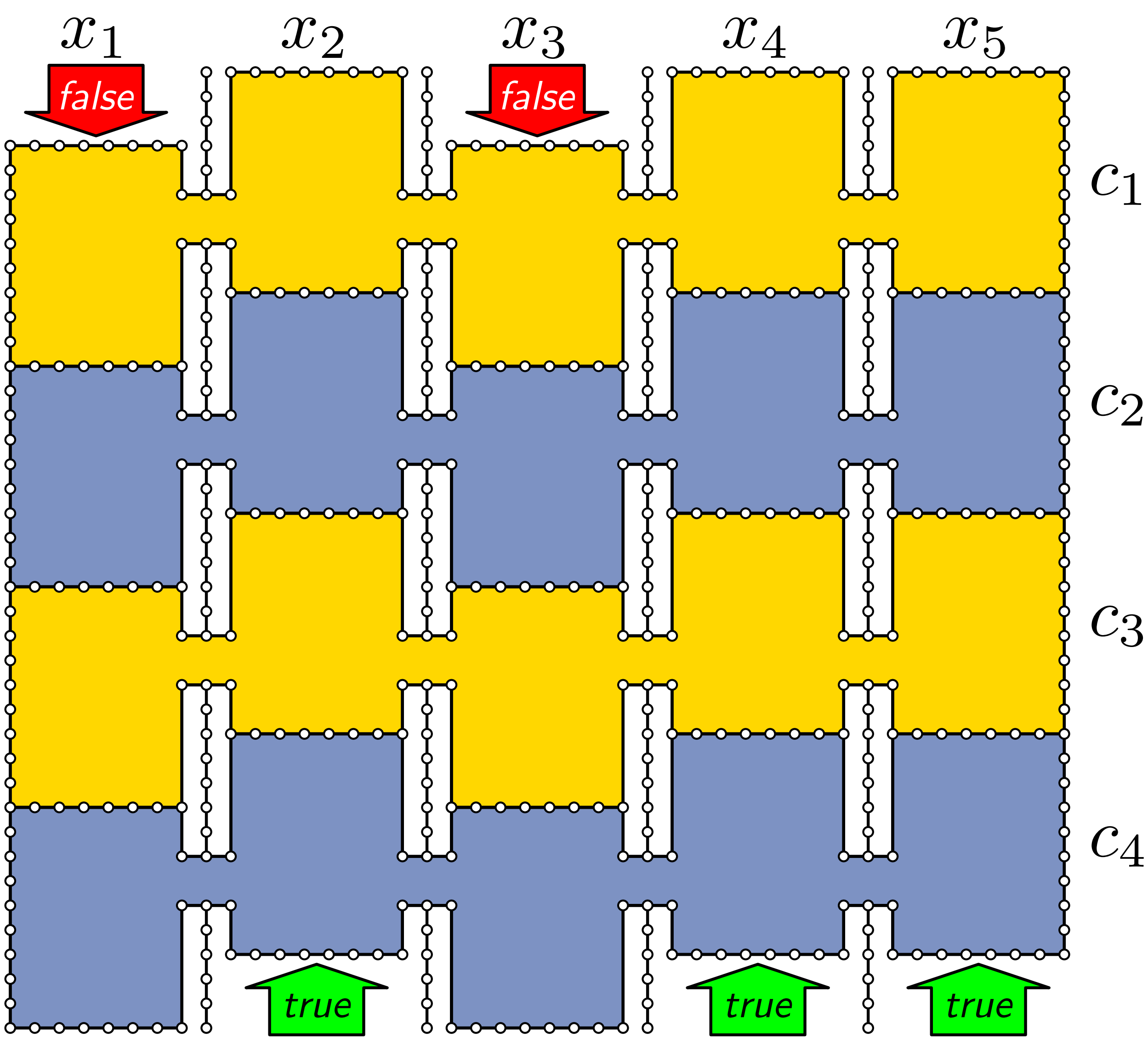
Belt is an edge with orthogonal representation $(r^4 l^4)^{2n} r^4$

- cells of the belt can be shifted within the frame
- each piston represents a variable with two possible states

Clause Gadgets



Clause Gadgets



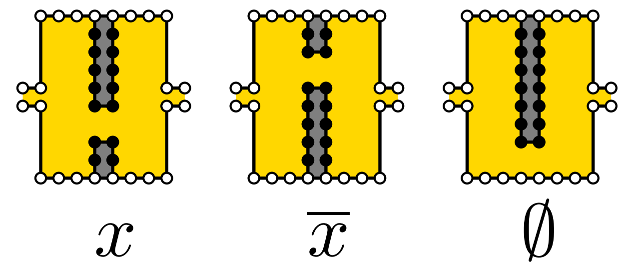
Example:

$c_1 = x_2 \vee \overline{x_4}$

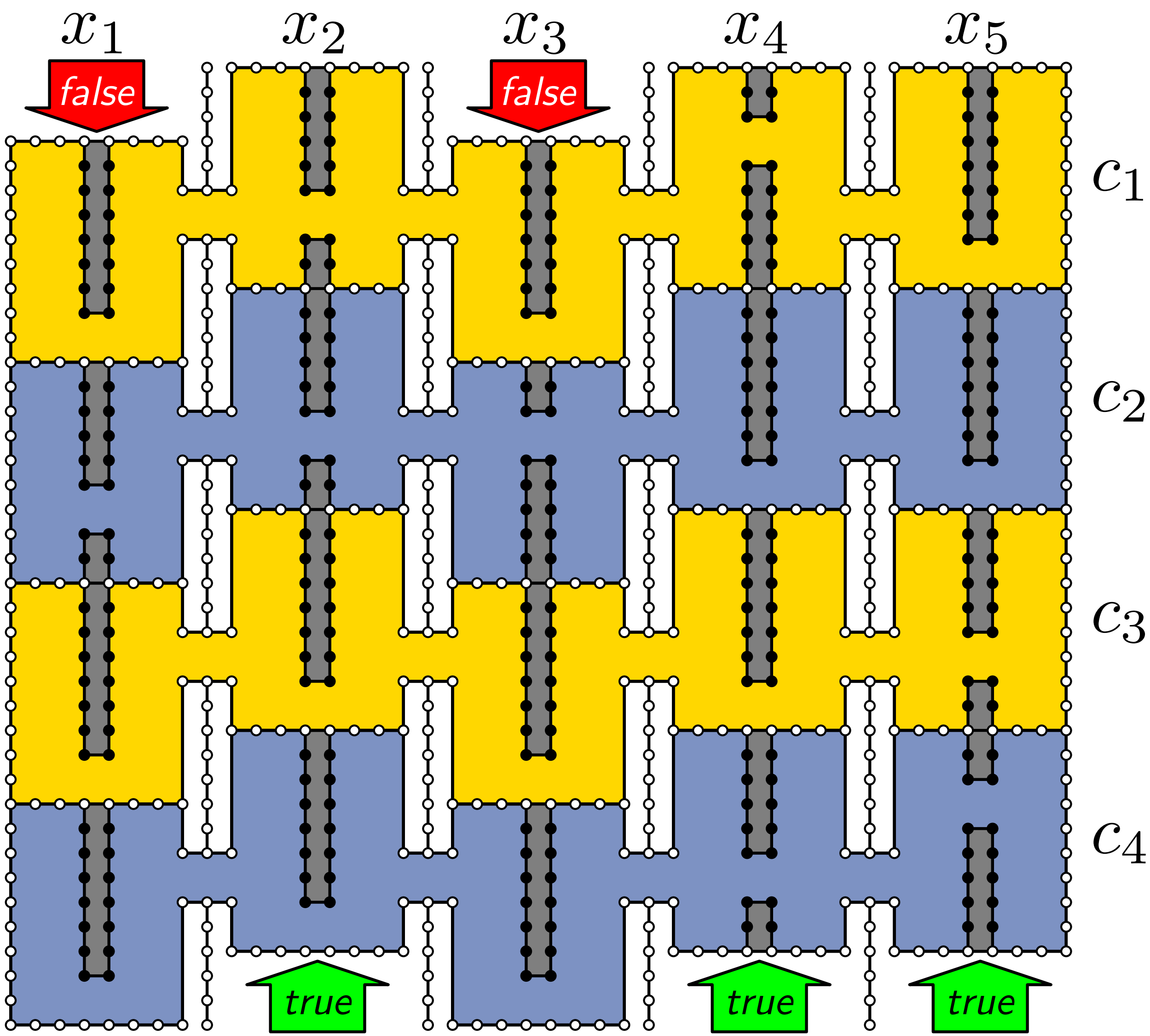
$c_2 = x_1 \vee x_2 \vee \overline{x_3}$

$c_3 = x_5$

$c_4 = x_4 \vee \overline{x_5}$



Clause Gadgets



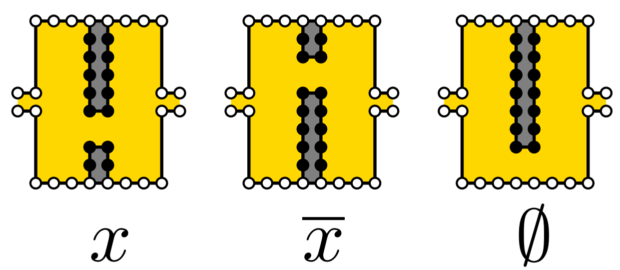
Example:

$c_1 = x_2 \vee \overline{x_4}$

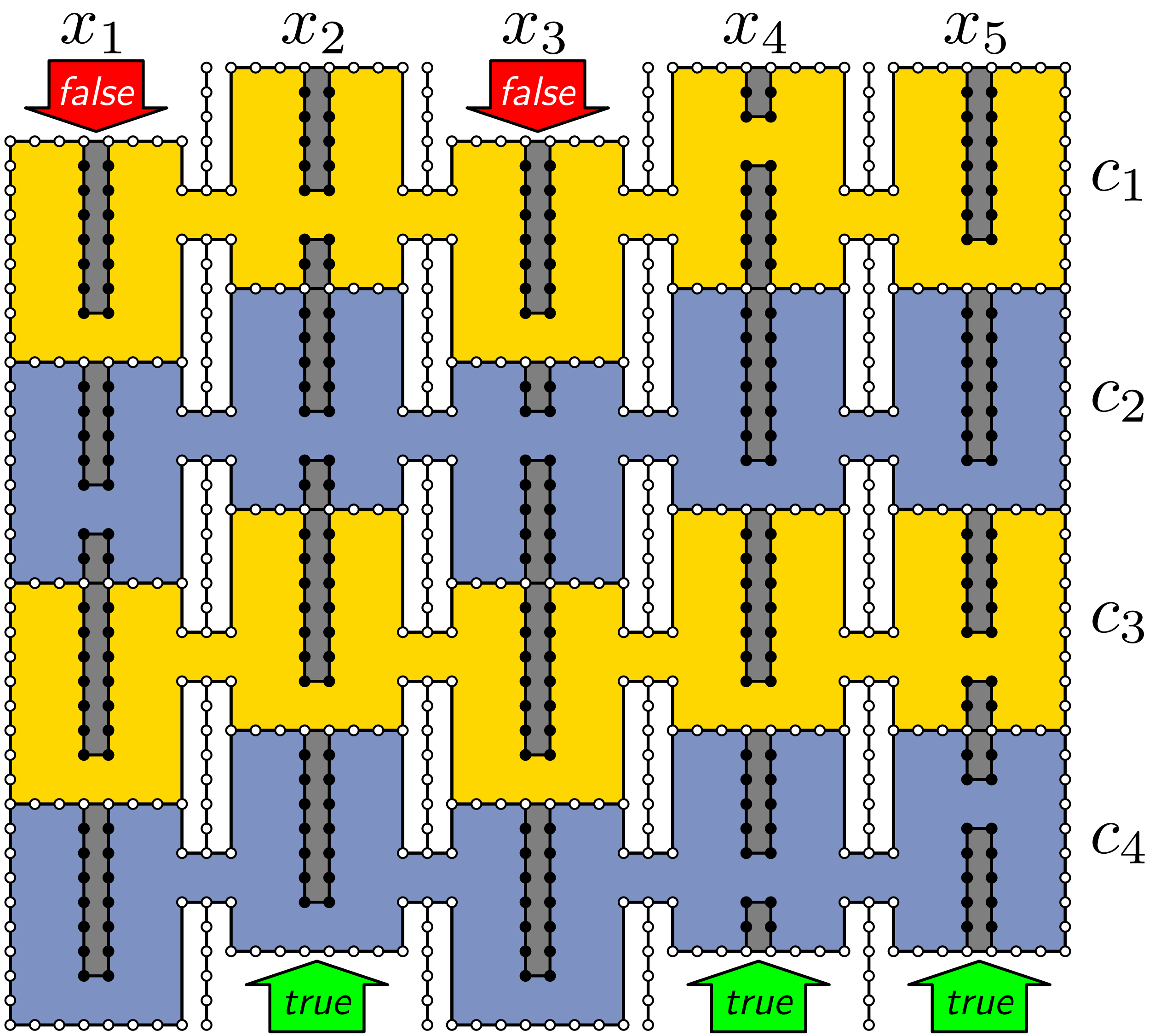
$c_2 = x_1 \vee x_2 \vee \overline{x_3}$

$c_3 = x_5$

$c_4 = x_4 \vee \overline{x_5}$



Clause Gadgets



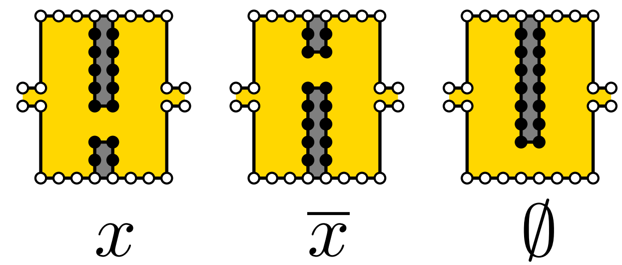
Example:

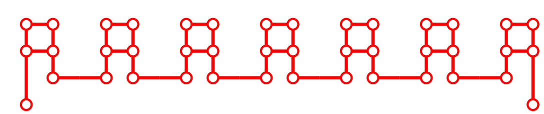
$c_1 = x_2 \vee \overline{x_4}$

$c_2 = x_1 \vee x_2 \vee \overline{x_3}$

$c_3 = x_5$

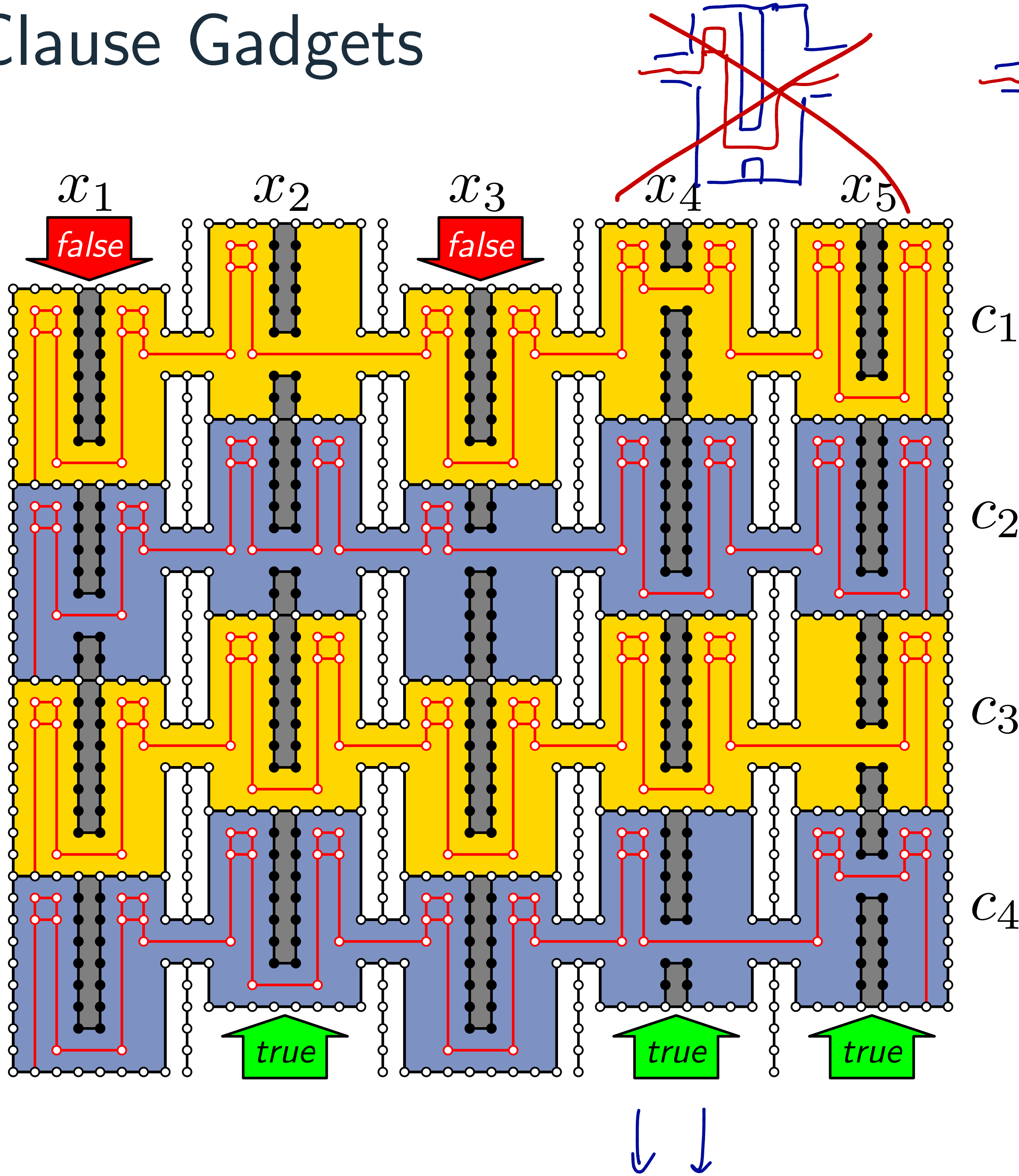
$c_4 = x_4 \vee \overline{x_5}$





route an “A^{2n-1}” chain through each clause

Clause Gadgets



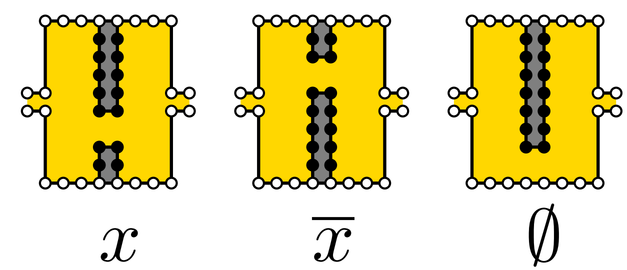
Example:

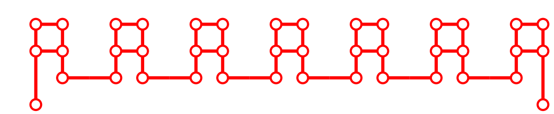
$c_1 = x_2 \vee \overline{x_4}$

$c_2 = x_1 \vee x_2 \vee \overline{x_3}$

$c_3 = x_5$

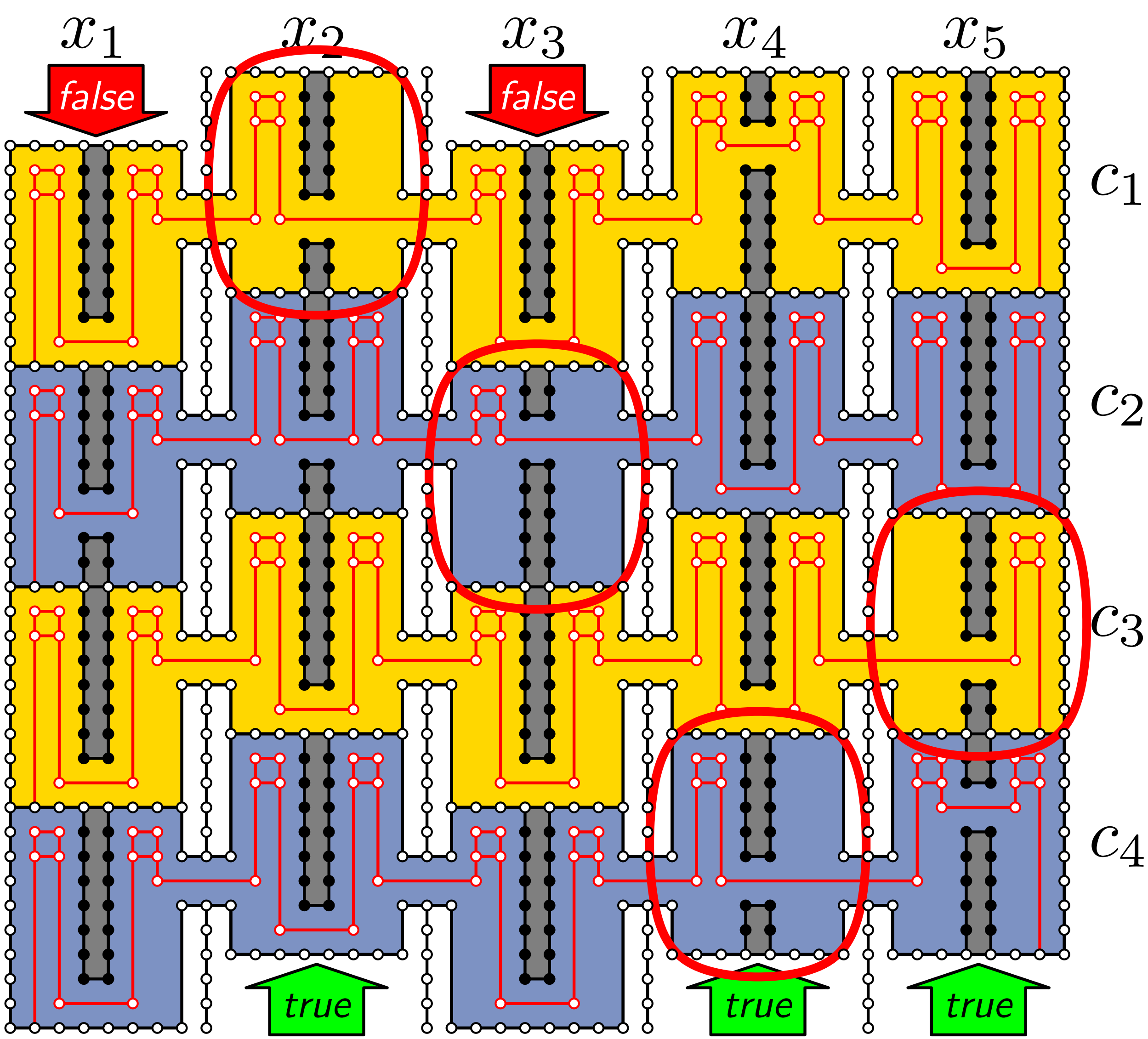
$c_4 = x_4 \vee \overline{x_5}$





route an "A^{2n-1}" chain through each clause

Clause Gadgets



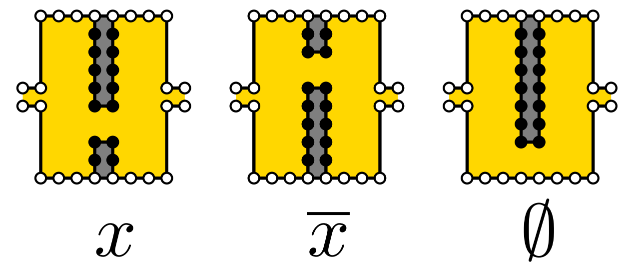
Example:

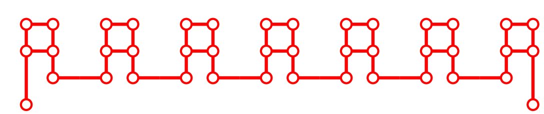
$c_1 = x_2 \vee \overline{x_4}$

$c_2 = x_1 \vee x_2 \vee \overline{x_3}$

$c_3 = x_5$

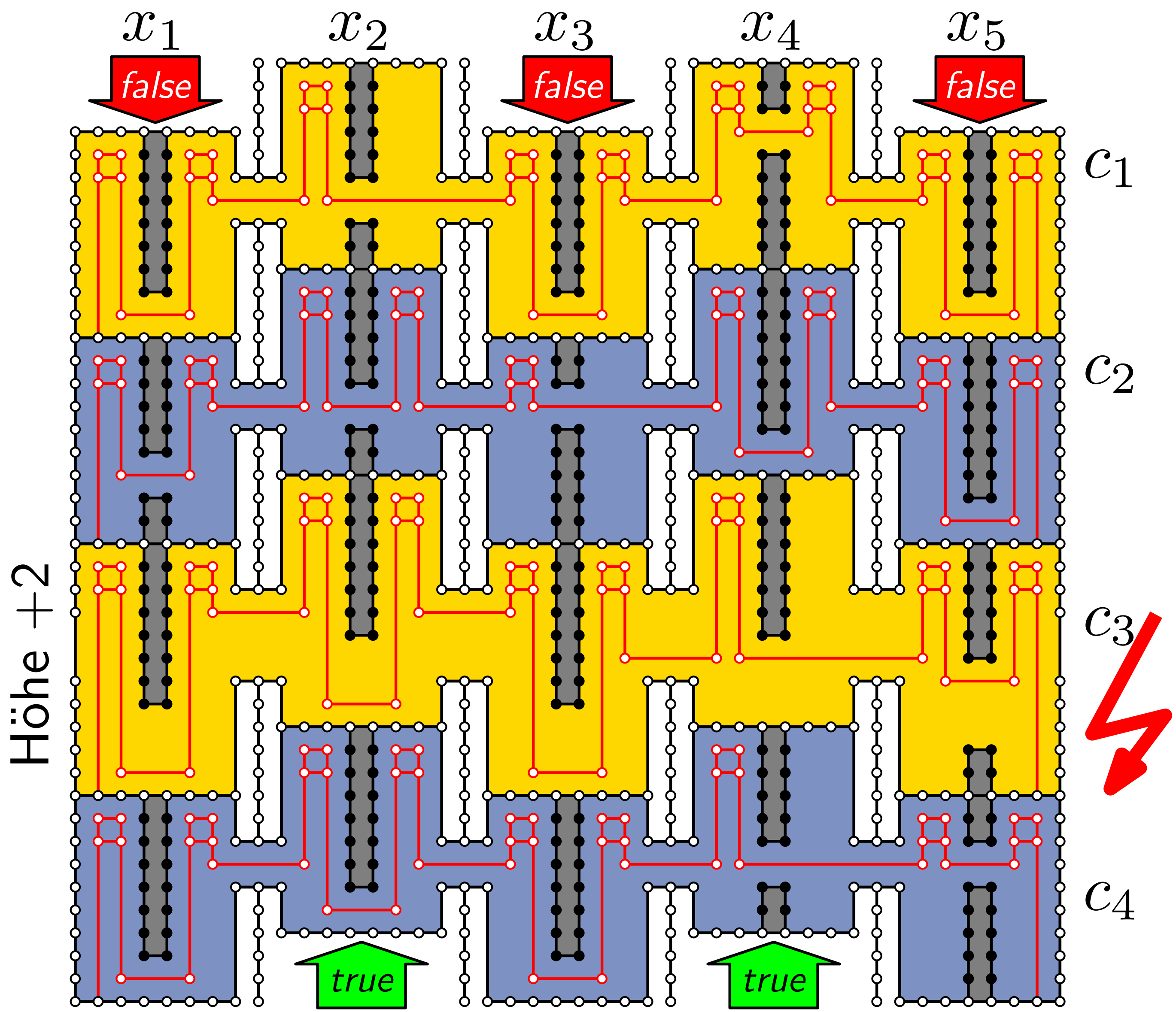
$c_4 = x_4 \vee \overline{x_5}$





route an "A^{2n-1}" chain through each clause

Clause Gadgets



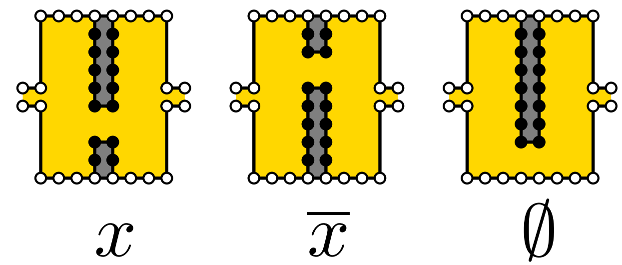
Example:

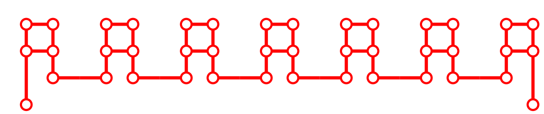
$c_1 = x_2 \vee \overline{x_4}$

$c_2 = x_1 \vee x_2 \vee \overline{x_3}$

$c_3 = x_5$

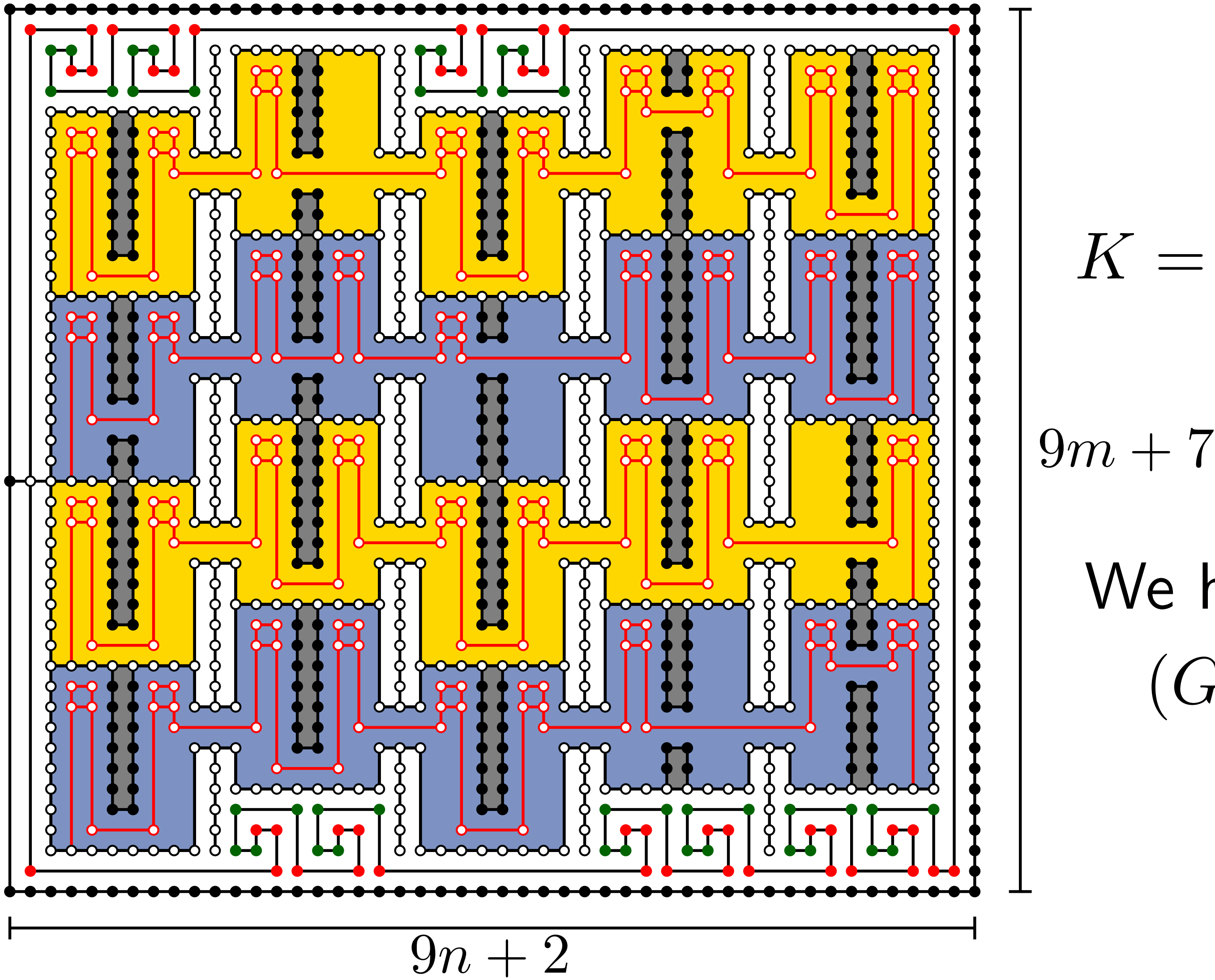
$c_4 = x_4 \vee \overline{x_5}$





route an "A^{2n-1}" chain through each clause

Full Reduction



Define

$$K = (9n+2)\cdot(9m+7)$$

We have:

(G, H) has drawing with area K

$\Leftrightarrow$

Φ is satisfiable

$\square$

Summary

- An orthogonal representation with minimum number of bends can be found in $O(n^{3/2})$ time
- Given an orthogonal representation a layout with minimum area or minimum total edge length can be computed in $O(n^{3/2})$ time for the case of rectangular faces
- In case of non-rectangular faces, reduce the problem to rectangular case. The resulting area is not minimum.

Summary

- An orthogonal representation with minimum number of bends can be found in $O(n^{3/2})$ time
- Given an orthogonal representation a layout with minimum area or minimum total edge length can be computed in $O(n^{3/2})$ time for the case of rectangular faces
- In case of non-rectangular faces, reduce the problem to rectangular case. The resulting area is not minimum.
- Area minimization with a given orthogonal representation is NP-hard. [Patrignani 2001]

Summary

- An orthogonal representation with minimum number of bends can be found in $O(n^{3/2})$ time
- Given an orthogonal representation a layout with minimum area or minimum total edge length can be computed in $O(n^{3/2})$ time for the case of rectangular faces
- In case of non-rectangular faces, reduce the problem to rectangular case. The resulting area is not minimum.
- Area minimization with a given orthogonal representation is NP-hard. [Patrignani 2001]
- Area minimization can be solved with an ILP. [Klau, Mutzel 1999]

Summary

- An orthogonal representation with minimum number of bends can be found in $O(n^{3/2})$ time
- Given an orthogonal representation a layout with minimum area or minimum total edge length can be computed in $O(n^{3/2})$ time for the case of rectangular faces
- In case of non-rectangular faces, reduce the problem to rectangular case. The resulting area is not minimum.
- Area minimization with a given orthogonal representation is NP-hard. [Patrignani 2001]
- Area minimization can be solved with an ILP. [Klau, Mutzel 1999]
- Various heuristics have been implemented and experimentally evaluated for running time and quality. [Klau, Klein, Mutzel 2001]

Summary

- An orthogonal representation with minimum number of bends can be found in $O(n^{3/2})$ time
- Given an orthogonal representation a layout with minimum area or minimum total edge length can be computed in $O(n^{3/2})$ time for the case of rectangular faces
- In case of non-rectangular faces, reduce the problem to rectangular case. The resulting area is not minimum.
- Area minimization with a given orthogonal representation is NP-hard. [Patrignani 2001]
- Area minimization can be solved with an ILP. [Klau, Mutzel 1999]
- Various heuristics have been implemented and experimentally evaluated for running time and quality. [Klau, Klein, Mutzel 2001]
- For non-planar graphs area minimization is hard to approximate. [Bannister, Eppstein, Simons 2012]

(Planar) Orthogonal Drawings

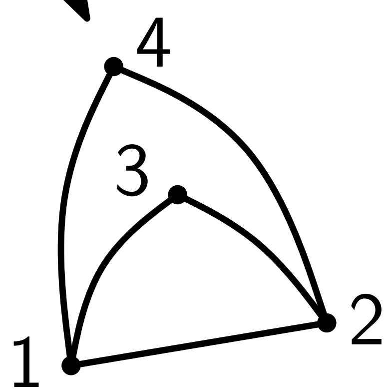
Three-step approach: *Topology – Shape – Metrics*

[Tamassia SIAM J. Comput. 1987]

$V = \{v_1, v_2, v_3, v_4\}$
$E = \{v_1v_2, v_1v_3, v_1v_4, v_2v_3, v_2v_4\}$

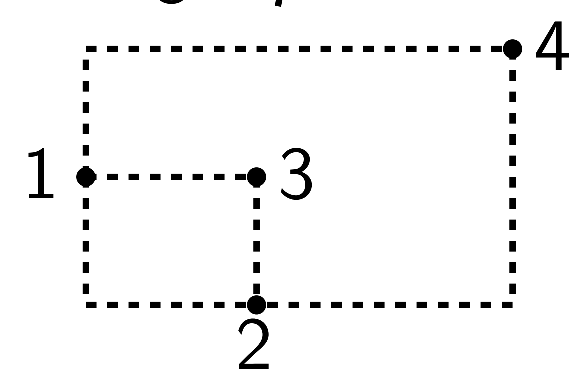
Reduce crossings

combinatorial embedding/ planarization



Minimize bends

orthogonal representation



planar orthogonal drawing

Minimize area