Plane Drawings of Queue and Deque Graphs*

Christopher Auer, Christian Bachmaier, Franz Josef Brandenburg,
Wolfgang Brunner, and Andreas Gleißner

University of Passau, Germany
{auerc,bachmaier,brandenb,brunner,gleissner}@fim.uni-passau.de

Abstract. In stack and queue layouts the vertices of a graph are linearly ordered from left to right, where each edge corresponds to an item and the left and right end vertex of each edge represents the addition and removal of the item to the used data structure. A graph admitting a stack or queue layout is a stack or queue graph, respectively.

Typical stack and queue layouts are rainbows and twists visualizing the LIFO and FIFO principles, respectively. However, in such visualizations, twists cause many crossings, which make the drawings incomprehensible. We introduce linear cylindric layouts as a visualization technique for queue and deque (double-ended queue) graphs. It provides new insights into the characteristics of these fundamental data structures and extends to the visualization of mixed layouts with stacks and queues. Our main result states that a graph is a deque graph if and only if it has a plane linear cylindric drawing.

1 Introduction

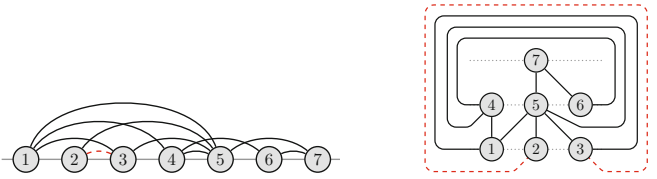
In his pioneering work on the Art of Computer Programming, D. E. Knuth raises the question "How shall we draw a tree?" [10, p. 306], which can be seen as the beginning of Graph Drawing. Knuth also studied elementary data structures, such as stacks, queues and deques, and represented their behavior as train tracks. In this paper we pose the question "How shall we draw a stack, a queue, or a deque?". The purpose of such drawings is to visualize the underlying data structure and the operations applied to it.

Stack and queue layouts have been studied extensively in the past, e.g., in [1, 2, 4–9, 12–14], and are used for 3D drawings of graphs [12, 13], in VLSI design [2] and in other application scenarios (see [9] for a short survey). In these layouts the vertices of a graph are linearly ordered from the left to the right. The vertices are processed in this order and each edge corresponds to an item which is inserted into the data structure at its left endpoint and is removed at its right endpoint. These operations must obey the principles of the underlying data structure, such as "last-in, first-out" for a stack or "first-in, first-out" for a queue.

k-stack (k-queue) layouts are a generalization of stack (queue) layouts: Such layouts consist of a single linear order of the vertices and a partition of the set of

* Supported by the Deutsche Forschungsgemeinschaft (DFG), grant BR835/15-1.

U. Brandes and S. Cornelsen (Eds.): GD 2010, LNCS 6502, pp. 68–79, 2011.



(a) Linear visualization (b) Arched leveled-planar

Fig. 1. Visualizing queue layouts

edges into k subsets, where each subset permits a stack (queue) layout, see, e.g., [5, 9]. k-stack (k-queue) layouts have also been generalized to *mixed* layouts, e.g., two stacks and one queue, and have been studied in [5]. The authors motivate their studies of mixed layouts by their investigations of the deque data structure: A deque can either emulate two stacks or one queue. Conversely, two stacks and one queue can emulate one deque.

Stack layouts are also known as book-embeddings of graphs [1] and the number of pages corresponds to the number of used stacks. These graphs have interesting graph theoretic properties: A graph G is a stack graph if and only if G is outer-planar [1]. Bernhart and Kainen [1] have characterized the class of 2-stack graphs as the subgraphs of planar graphs with a Hamiltonian cycle, and every planar graph has a layout with four stacks [14].

Common visualizations of stack and queue graphs with a given layout place the vertices from left to right on the x-axis according to the given order. The edges are drawn as arches, which are x-monotone curves above the x-axis. Stack layout visualizations show *rainbows* [13] as characteristic structures, which are properly nested arches, whereas the equivalent structure in queue layouts are *twists* [9]. Conversely, in a visualization of queue graphs rainbows are not allowed, while stack graphs forbid twists.

Fig. 1(a) depicts a queue graph. The edges drawn as solid lines constitute a valid queue layout since no two arches nest completely (nesting edges with common end vertices are allowed). A characteristic twist is displayed by edges $\{1, 4\}$ and $\{2, 5\}$, i.e., $\{1, 4\}$ is added to the queue before $\{2, 5\}$ and both are removed in the same order. However, due to the many crossings (one crossing per twist), it is hard to follow the routing of the edges and to validate the queue layout. Introducing the edge $\{2, 3\}$ (dashed) destroys the queue layout since it is completely nested in the arch $\{1, 5\}$, i.e., at first $\{1, 5\}$ and then $\{2, 3\}$ is added to the queue but $\{2, 3\}$ has to be removed before $\{1, 5\}$. Nevertheless, it is hard to recognize immediately that $\{2, 3\}$ destroys the queue layout.

Heath et al. [5, 9] characterize the class of queue graphs as the arched leveled-planar graphs. Such a graph has a planar drawing with vertices placed on levels and inter-level edges only between two adjacent levels or intra-level edges from the left-most vertex to vertices on the right side. Fig. 1(b) shows the queue layout from Fig. 1(a) again, where levels are drawn as dotted lines. Again it is hard to see immediately that edge $\{2, 3\}$ is illegal. Moreover, an invalid drawing of an

arched leveled-planar graph does not necessarily indicate an invalid queue layout since a valid drawing with a different leveling of the vertices is still possible.

In this paper we propose a novel approach for visualizing queue graphs that overcomes the aforementioned drawbacks. We achieve our visualization of a queue graph by winding the arcs at most once around the surface of a 3D cylinder. A 2D representation is achieved by cutting the cylinder and duplicating the vertices. Moreover, our representation is also suitable for deque (double-ended queue) graphs where the deque may even have a restricted set of input or output operations. We can also display graphs with mixed layouts of stacks, queues and deques together in one concise drawing. By applying our drawing technique we immediately arrive at the result that every deque graph is planar.

The remainder of the paper is structured as follows. In Sect. 2 we introduce deque layouts. In Section 3 we define linear cylindric drawings and their equivalent representations. We also characterize the class of graphs permitting a deque layout as the graphs permitting a crossing-free linear cylindric drawing. We describe how linear cylindric drawings help to investigate mixed layouts in Sect. 4. We also revisit queue layouts and show how linear cylindric drawings of queue graphs overcome the drawbacks of the drawings depicted in Fig. 1. Finally, Sect. 5 gives a conclusion and an outlook to future work.

2 Preliminaries

We consider simple undirected graphs $G = (V, E)$ with n vertices and m edges. A linear layout π is a bijective assignment $\pi : V \to \{1, \ldots, n\}$ of the vertices to positions $\{1, \ldots, n\}$. For each edge $e = \{u, v\}$ we denote by $l(e) = \min\{\pi(u), \pi(v)\}$ and $r(e) = \max\{\pi(u), \pi(v)\}$ the position of its left- and right-hand vertex, respectively. Edge $e \in E$ is said to cover the *range* from $l(e)$ to $r(e)$ (both included).

A deque generalizes a stack and a queue: It has two ends, a *head* `h` and a *tail* `t`, to insert and remove items. If insertions or removals are only allowed at the deque's head, it is called *input* or *output restricted*, respectively. Let α and ω be two functions from E to $\{\mathtt{h}, \mathtt{t}\}$ that assign to each edge e the side of its addition and its removal, respectively. α/ω are called *input/output assignments (I/O assignments)*. If $\alpha(e) = \omega(e)$, then e is called a *stack edge*, otherwise a *queue edge*, according to the manner the edges are processed by the deque. We denote by $\Delta(G)$ the tuple (π, α, ω) and call it *linear I/O layout*. $\Delta(G)$ is a *deque layout* iff the vertices can be processed from left to right according to π such that all edges can be processed by the deque according to α and ω.

Definition 1. *A* graph is a deque graph *if and only if G has a deque layout. Accordingly, a graph is an input restricted deque (an output restricted deque, a stack, a queue) graph if it has a respective layout.*

Note that at each vertex v, before any edge can be inserted to a particular side, e.g., head, all edges that end at v and are accessible from the head need to be removed. Also note that at each vertex v, when inserting edges $e \in E$ into the deque pointing to right-hand neighbors, i.e., $l(e) = \pi(v)$, the queue (stack)

edges have to be inserted in (in reverse) order of their respective positions of the right-hand neighbors. This is to ensure that the edges can be removed from the deque when processing the corresponding right-hand neighbor. Furthermore, at first always all queue edges and then all stack edges have to be inserted into the deque since otherwise edges cannot be removed anymore, e.g., consider a stack and a queue edge inserted at the head in that order, then neither of the edges can be removed.

3 Linear Cylindric Drawings of Deque Graphs

In this section we introduce a new type of drawing on the surface of a 3D cylinder (Sect. 3.1) and transform the drawings into equivalent 2D representations. In the case of planar drawings they exactly fit to deque graphs (Sect. 3.2).

3.1 Linear Cylindric Drawings

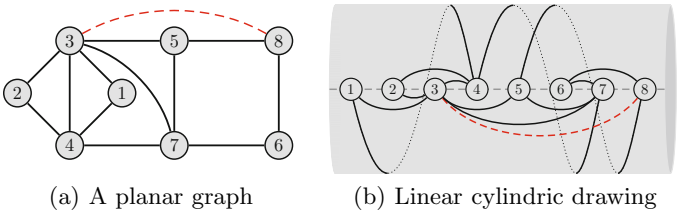
Definition 2. *In a* linear cylindric drawing $\Gamma(G)$ *of a graph* G *the vertices are placed disjointly on a straight line* L*,* the front line*, on the surface of the cylinder parallel to its axis. The edges are drawn as monotone curves in direction of the cylinder's axis and do not cross* L*.*

For convenience, we consider horizontal cylinders where L is parallel to the x-axis. Moreover, we identify the placement of the vertices on L with the permutation $\pi : V \to \{1, \ldots, n\}$.

Obviously, every graph has a linear cylindric drawing. The vertices can be arranged arbitrarily on the front line and the edges are drawn as simple curves on the side surface of the cylinder while not crossing L.

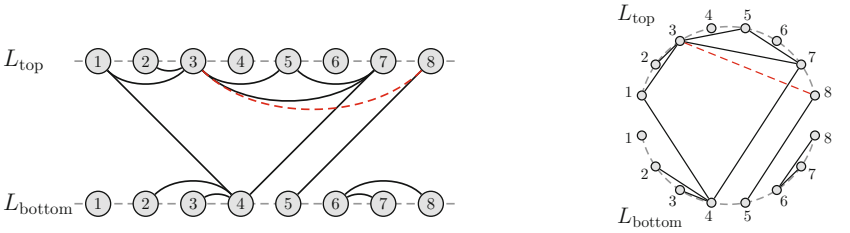
For an example, consider graph $G = (\{1, \ldots, 8\}, E)$ as displayed in Fig. 2(a). Fig. 2(b) visualizes a linear cylindric drawing of G: The vertices are drawn on the horizontal front line (dashed) and edges are either drawn as arches, e.g., $\{2, 4\}$, or wrap at most once around the cylinder, e.g., $\{1, 4\}$. Note that the dashed edge $\{3, 8\}$ causes a crossing with edge $\{4, 7\}$.

A linear cylindric drawing $\Gamma(G)$ imposes a direction onto each edge from the lower to the higher π-value of its vertices. We denote an undirected edge $\{u, v\}$



(a) A planar graph (b) Linear cylindric drawing

Fig. 2. A graph and its linear cylindric drawing



(a) Unrolled cylinder (b) Unrolled cylinder on the circle

Fig. 3. 2D representations of the linear cylindric drawing of Fig. 2(b)

with $\pi(u) < \pi(v)$ by the directed edge (u, v). Moreover, $\Gamma(G)$ partitions the set of edges into four subsets according to their orientation with respect to the front line. Let $E_\cap$ ($E_\cup$) denote the set of edges leaving and entering their vertices above (below) the front line and let $E_/$ ($E_\backslash$) denote the set of edges leaving the front line above (below) and entering at the opposite side. The subscripts of the sets illustrate the shape of the edges. The edges from $E_/$ and $E_\backslash$ wrap around the cylinder once and the edges from $E_\cap$ ($E_\cup$) can be drawn as arches above (below) the front line.

Definition 3. *The tuple* $\mathcal{C}(G) = (\pi, E_\cup, E_\cap, E_\backslash, E_/)$ *is a* linear cylindric embedding of $G = (V, E)$.

We obtain a 2D representation of a linear cylindric drawing, which we call *unrolled cylinder*, by cutting the cylinder along the front line and "bending" the surface of the cylinder until it is plane. The front line with the vertices is duplicated and the two copies constitute the bottom (L_{bottom}) and top (L_{top}) of the so obtained drawing.

For instance, when applied to Fig. 2(b), the result is depicted in Fig. 3(a), where the area that was formerly placed above the front line is now situated at the bottom of the drawing. Then an edge $(u, v) \in E_\backslash$ can be drawn as a straight line from its vertex u on L_{top} to its vertex v on L_{bottom}. Symmetrically, the edges from $E_/$ can be drawn as straight lines L_{bottom} to L_{top}. The edges in $E_\cap$ can be represented as arches above L_{bottom} and, accordingly, the edges from $E_\cup$ can be represented as arches below L_{top}. Note that crossings between each pair of edges $(e_1, e_2) \in E_\cap \times E_\cup$ can always be avoided.

The drawing in Fig. 3(a) can further be continuously transformed to *the unrolled cylinder on the circle* by mapping L_{top} and L_{bottom} to two halves of a circle. The result is depicted in Fig. 3(b). Note that in this drawing all edges are drawn as straight lines and, hence, their routing is uniquely determined.

Without further proof, note that all different types of drawings are topologically equivalent, i.e., all drawings can be transformed into each other by a continuous function without changing the topological structure of the drawing. In particular, planarity is preserved. The equivalence of the drawings has an important

implication: Consider a linear cylindric drawing where non-monotonous edges are allowed. Such a drawing can be continuously transformed to a drawing on the unrolled cylinder on the circle with straight-line edges that are monotonous. This justifies our assumption of monotonous edges in Def. 2.

Definition 4. *A graph G is* linear cylindric planar *if G has a linear cylindric embedding that permits a linear cylindric drawing without crossing edges.*

The classes of graphs that can be drawn without crossings on the plane and on the cylinder are equal. We thus obtain:

Corollary 1. *Every linear cylindric planar graph is a planar graph.*

3.2 Characterization of Deque Graphs

In this section we present the main result of this paper:

Theorem 1. *A graph G is a deque graph iff G is linear cylindric planar.*

The idea of the proof is to construct a one-to-one correspondence between a linear I/O layout $\Delta(G) = (\pi', \alpha, \omega)$ and a linear cylindric embedding $\mathcal{C}(G) = (\pi, E_\cup, E_\cap, E_\backslash, E_/)$ as follows: The positions π of the vertices on L are equal to the linear layout π'. The crucial point of the proof concerns the edges: Consider the edges in $E_\cap$ in the linear cylindric embedding that all leave and enter their end vertices above the front line, e.g., $(2, 4)$ in Fig. 2(b). Edges in $E_\cap$ are drawn as arches in a linear cylindric layout and, hence, they do not cross iff they form rainbows and, consequently, constitute a stack layout. Hence, an edge $(u, v) \in E_\cap$ is interpreted as an edge that is processed by the deque like by a stack, that is, it is inserted at and removed from the same side, e.g., tail. The same holds true for edges in $E_\cup$ with respect to the deque's head. Edges in $E_\backslash$ or $E_/$ which enter and leave at opposite sides of the front line are interpreted as "moving" from one side of the deque to its opposite side. For instance, $(1, 4) \in E_\backslash$, which leaves below and enters above the front line, is interpreted as being inserted at the deque's head and removed from its tail. Consequently, $E_\backslash$ and $E_/$ are queue edges inserted at the head and tail, respectively.

Using this one-to-one correspondence, we are then able to prove that edges cause no crossings in a linear cylindric drawing if and only if they can be processed by the deque. Conversely, any unavoidable crossing in such a drawing can be interpreted as a violation of the deque layout, i.e., crossing edges cannot be processed by the deque by any allowed operation.

First we show that it is sufficient to only consider pairs of edges when investigating a deque layout or a linear cylindric planar embedding. We start with deque layouts:[1]

[1] A similar statement is proven in [3], which is concerned with trains entering and leaving a train station from two sides. Such a train station with n tracks can be modelled by n deques and the trains must be assigned to the deques such that they do not block each other.

Lemma 1. *$\Delta(G) = (\pi, \alpha, \omega)$ is a deque layout if and only if for each pair of edges $e, e' \in E$ with $e \neq e'$, $\Delta_{|e,e'}(G) = (\pi, \alpha_{|e,e'}, \omega_{|e,e'})$ is a deque layout, where $\alpha_{|e,e'}$ and $\omega_{|e,e'}$ are the restrictions of α and ω to $\{e, e'\}$, respectively.*

Proof. "⇒": If all edges can be processed by the deque, in particular any two edges can be processed.

"⇐": For each pair of distinct edges $e, e' \in E$, $\Delta_{|e,e'}$ is a deque layout. We assume for contradiction that $\Delta(G)$ is not a deque layout.

Since an edge can always be inserted into the deque a problem can only occur when removing an edge.

Let $e \in E$ be the first edge that cannot be removed at some vertex v. W. l. o. g. we assume that $\omega(e) = \mathtt{h}$. Let $e_1, \ldots, e_k$ be the elements between the head and e in the deque. $k \geq 1$ since otherwise e could be removed from the deque. If for all edges e_i with $1 \leq i \leq k$, $r(e_i) = \pi(v)$ and $\omega(e_i) = \mathtt{h}$, then all edges e_i could be removed and e could also be removed. Thus, there exists an edge e_j with $1 \leq j \leq k$ which prevents the removal of e, i.e., $r(e_j) > r(e)$ or $\omega(e_j) = \mathtt{t}$.

In the first case, i.e., $r(e_j) > r(e)$, if $l(e_j) = l(e)$ and $\alpha(e_j) = \alpha(e)$, then at the vertex at position $l(e)$ the edges e and e' would have been inserted into the deque such that e would always be accessible from the head (see also Sec. 2), where four cases have to be distinguished: If both are stack edges added to the head then e_j would be have been inserted before e since $r(e_j) > r(e)$. If both are queue edges then e and e_j would have been inserted at the tail in that order. Similarly if e is a stack and e_j a queue edge, then e_j would have been inserted at the head before e is inserted at the head. If e is a queue edge and e_j a stack edge, then e would have been inserted at the tail before e_j is inserted at the tail as a stack edge. In all cases e is accessible from the head. If $l(e_j) = l(e)$ and $\alpha(e_j) \neq \alpha(e)$, then e_j and e can not be processed in $\Delta_{|e,e_j}(G)$ as well independently of their insertion order. In all other cases the relative order in which e_j and e are inserted into the deque is uniquely determined by $l(e)$ and $l(e_j)$. The same order has to be used in $\Delta_{|e,e_j}(G)$ and causes a problem there as well. The second case $\omega(e_j) = \mathtt{t}$ follows analogously. Hence, in each case $\Delta_{|e,e_j}(G)$ is no deque layout, which is a contradiction. □

Lemma 2 is the corresponding version of Lemma 1 for pairs of edges in linear cylindric planar embeddings:

Lemma 2. *$\mathcal{C}(G) = (\pi, E_\cup, E_\cap, E_\backslash, E_/)$ is a linear cylindric planar embedding of a graph $G = (V, E)$ if and only if for each pair of edges $e, e' \in E$ with $e \neq e'$ $\mathcal{C}_{|e,e'}(G) = (\pi, E_\cup \cap \{e, e'\}, E_\cap \cap \{e, e'\}, E_\backslash \cap \{e, e'\}, E_/ \cap \{e, e'\})$ is a linear cylindric planar embedding.*

Proof. "⇒": Take a linear cylindric drawing without crossings. From this drawing a linear cylindric planar embedding for each pair of edges can be obtained.

"⇐": The drawing of the unrolled cylinder on the circle (e.g., Fig. 3(b)) is a drawing with straight lines and is topologically equivalent to a linear cylindric drawing. Note that the routing of the edges is uniquely determined if the edges are drawn as straight lines. In order to construct a plane drawing with all edges,

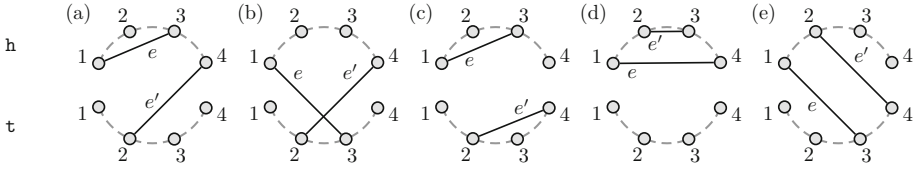


Fig. 4. Different cases in the proof of Lemma 3

simply draw all edges according to their unique straight-line representation. This drawing has no crossings since no pair of edges cross and is conform to embedding $\mathcal{C}(G)$. Hence, $\mathcal{C}(G)$ is linear cylindric planar. □

Lemma 3 is the main step to our theorem. It states that every pair of edges can be processed by a deque if and only if they cause no crossing in the drawing. In order to show this we utilize the aforementioned one-to-one correspondence between a linear cylindric embedding and a linear I/O layout:

Lemma 3. *Given $G = (V, E)$, let $\mathcal{C}(G) = (\pi, E_\cup, E_\cap, E_\setminus, E_/)$ be a linear cylindric embedding and $\Delta(G) = (\pi, \alpha, \omega)$ be a linear I/O layout. If $\forall e \in E$:*

$$\alpha(e) = \mathtt{h} \wedge\ \omega(e) = \mathtt{h} \Leftrightarrow e \in E_\cup\,, \qquad \alpha(e) = \mathtt{h} \wedge\ \omega(e) = \mathtt{t} \Leftrightarrow e \in E_\setminus\,,$$
$$\alpha(e) = \mathtt{t} \wedge\ \omega(e) = \mathtt{h} \Leftrightarrow e \in E_/\,, \qquad \alpha(e) = \mathtt{t} \wedge\ \omega(e) = \mathtt{t} \Leftrightarrow e \in E_\cap\,,$$

then, for every pair of distinct edges $e, e' \in E$, $\mathcal{C}_{|e,e'}(G)$ is a linear cylindric planar embedding if and only if $\Delta_{|e,e'}(G)$ is a deque layout.

Proof. Let $e, e' \in E$ be two distinct edges, therefore $l(e) \neq l(e')$ or $r(e) \neq r(e')$. W. l. o. g., we assume $l(e) < l(e')$.

The proof is a complete differentiation between all cases of how two edges are processed by the deque and routed in the embedding: For each case we show that two distinct edges $e, e' \in E$ do not cross if and only if they can be processed by the deque. We assume that e and e' have non-disjoint ranges, i.e., they overlap. Otherwise, they can be drawn without crossings, and, conversely, they can be processed disjointly by the deque according to their I/O assignments. The same holds true if $r(e) = l(e')$. Note that $l(e) = r(e')$ is not possible since we assume $l(e) < l(e')$. The following five cases remain:

Case (a): $\alpha(e) = \omega(e)$, $\alpha(e') \neq \omega(e')$ (e is a stack edge and e' a queue edge): Without loss of generality, we assume $\alpha(e) = \omega(e) = \mathtt{h}$. If $\alpha(e') = \mathtt{h}$, then e and e' do not cross iff $l(e') \leq l(e)$, which is not possible by assumption, or $l(e') \geq r(e)$, which is not possible by assumption since the edges have to overlap.

If $\alpha(e') = \mathtt{t}$, then e and e' do not cross if and only if $r(e') \leq l(e)$, i.e., the edges do not overlap which contradicts our assumption, or $r(e') \geq r(e)$ (Fig. 4 (a)). In the deque first e is inserted at the head, then e' at the tail. Afterwards, e can be removed from the head and then e'.

Conversely, assume that e and e' can be processed by the deque but, for the sake of contradiction, $r(e') < r(e)$, i.e., e and e' cross. This implies that e' must

be removed from the head before e. However, at first e is inserted at the head and then e' at the tail and, hence, e' cannot be removed from the head because e is blocking its way.

The case where e is a queue and e' is a stack edge follows analogously.

Case (b): $\alpha(e) \neq \omega(e)$, $\alpha(e') \neq \omega(e')$, $\alpha(e) \neq \alpha(e')$ (two queue edges inserted at different sides): Since e and e' overlap this situation always causes a crossing (Fig. 4(b)). Moreover, since e and e' overlap, there is a time instance where both edges are in the deque. However, since both have to be removed from opposite sides they cannot be removed at all.

Case (c): $\alpha(e) = \omega(e) \neq \alpha(e') = \omega(e')$ (two stack edges inserted at different sides): These edges never cross (Fig. 4(c)) and, in the deque, two stack edges inserted at different sides, can always be processed without any problems.

Case (d): $\alpha(e) = \omega(e) = \alpha(e') = \omega(e')$ (two stack edges inserted at the same side): e and e' do not cross if and only if e nests e' and, hence, $r(e') \leq r(e)$ (Fig. 4(d)). In the deque, at first e and then e' can be inserted at the same side of the deque and both are removed from this side in reverse order.

Conversely, since $l(e) < l(e')$, e is inserted before e' into the deque. Since both are stack edges inserted at the same side, e must not be removed before e' and, hence, $r(e') \leq r(e)$. Thus, e properly nests e' and they cause no crossing.

Case (e): $\alpha(e) = \alpha(e') \neq \omega(e) = \omega(e')$ (two queue edges inserted at the same side): Since e and e' do not cross, we have that $r(e) \leq r(e')$ (Fig. 4(e)). Consequently, e is inserted before e' and both are removed in the same order.

Conversely, since e and e' are both queue edges inserted at the same side and e is inserted before e' it follows that e is removed from the deque before e'. Hence, $r(e) \leq r(e')$ and e and e do not cross in the drawing. □

We are now able to prove our main result of Theorem 1:

Proof. Let $\Delta(G)$ be a linear I/O layout and $\mathcal{C}(G)$ a linear cylindric embedding of G. By Lemma 1, G permits a deque layout iff each pair of edges permits a deque layout. By Lemma 3 this holds true if and only if no pair of edges causes a crossing in the linear cylindric embedding. Finally, by Lemma 2 this is true if and only if $\mathcal{C}(G)$ is a plane embedding. □

By Corollary 1 and Theorem 1 we can conclude:

Corollary 2. *A graph $G = (V, E)$ that permits a deque layout is planar.*

Theorem 1 leads to the following interpretation of a linear cylindric drawing: Consider a vertical line drawn in the middle between vertices 2 and 3 from L_{top} to L_{bottom} in Fig. 3(a). This line intersects the edges $(2,3)$, $(1,3)$, $(1,4)$ and $(2,4)$ in that order from top to bottom. This sequence reflects the content of the deque after vertex 2 and before vertex 3 is processed, where $(2,3)$ is situated at the head and $(2,4)$ at the tail. When moving this vertical line like a scan line further to the right, its crossings with edges always correspond to the content of the deque. If the vertical line passes a crossing between two edges e and e', e.g.,

$(3, 8)$ and $(4, 7)$, then this can be interpreted as swapping the positions of e and e' in the deque, which is an invalid deque operation.

Note that the aforementioned interpretation of a linear cylindric drawing is only true if all edges are monotonously drawn from the left to the right, which is another reason why we assume monotonicity in Definition 2.

4 Linear Cylindric Drawings of Queue and Mixed Layouts

In this section we show how linear cylindric drawings can help to investigate layouts of graphs on data structures that allow only a subset of the operations of a deque and layouts with mixtures of data structures like a deque together with a stack or two stacks. Queue, 1- and 2-stack, and input and output restricted deque layouts are special cases of a deque layout:

Corollary 3

- *A graph is a queue (stack) graph iff it is a deque graph where all edges are queue (stack) edges and are inserted either all at the head or all at the tail.*
- *A graph is a 2-stack graph iff it is a deque graph with stack edges only.*
- *A graph is an input (output) restricted deque graph iff it is a deque graph where* $\alpha(e) = \texttt{h}$ *(*$\omega(e) = \texttt{h}$*) for all* $e \in E$*.*

Corollary 4

- *A graph is a queue (stack) graph iff it is linear cylindric planar with the edges either all in* $E_\backslash$ *or all in* $E_/$ *(all in* $E_\cap$ *or all in* $E_\cup$*).*
- *A graph is a 2-stack graph iff it is linear cylindric planar with all edges in* $E_\cup \cup E_\cap$*.*
- *A graph is an input (output) restricted deque graph iff it is linear cylindric planar with all edges in* $E_\cup \cup E_\backslash$ *(*$E_\cup \cup E_/$*).*

4.1 Queue Graphs

In this section we revisit queue graphs and their drawings that have been discussed in Sect. 1. Consider again the two drawings in Fig. 1. In both drawings it is hard to recognize immediately that the edges drawn as solid lines depict a queue layout and that edge $(2, 3)$ destroys this layout with respect to the depicted linear layout.

The same graph can be depicted with the same linear layout on an unrolled cylinder (Fig. 5) where all edges are in $E_\backslash$. It is immediately visible that none of the solid drawn edges cross and, hence, display a valid queue layout. Edge $(2, 3)$ crosses edges $(1, 4)$ and $(1, 5)$ and, consequently, $(2, 3)$ destroys the queue layout. Moreover, it is immediately visible that exactly these three crossing edges destroy the queue layout. Removing edges until the drawing is crossing-free, e.g., edges $(1, 4)$ and $(1, 5)$, reestablishes a valid queue layout.

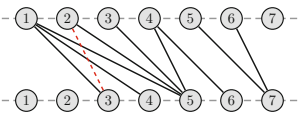
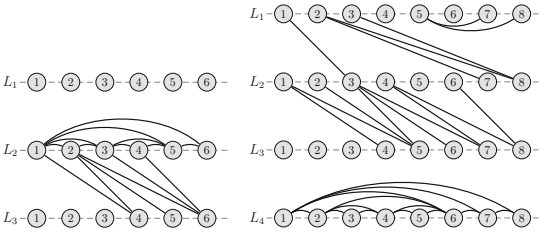


Fig. 5. Unrolled cylinder drawing of a queue graph



(a) K_6 (b) K_8

Fig. 6. Linear cylindric drawings of mixed layouts

4.2 Linear Cylindric Drawings of Mixed Layouts

Stack, queue, and deque layouts can be extended to *mixed layouts*, where k (possibly different) data structures $\{D_1, \ldots, D_k\}$ are given. Such a mixed layout consists of a single linear layout π of the vertices and a partition of the set of edges E consisting of k subsets $E_1, E_2, \ldots, E_k$, where for each $i \in \{1, \ldots, k\}$, $G = (V, E_i)$ has a layout in data structure D_i with linear layout π. Our technique of representing layouts by an unrolled cylinder straightforwardly extends to mixed layouts: Create $k+1$ copies $L_1, \ldots, L_{k+1}$ of the front line, place them one upon the other and display the edges E_i of data structure D_i between the i-th and $(i+1)$-th front line as described in Sect. 3.1.

As an example consider the complete graph K_6 with six vertices. This graph has neither a 2-stack nor a 2-queue layout. Fig. 6(a) shows the K_6 in a linear cylindric drawing with two data structures. Between L_1 and L_2 only stack edges are used. The region between L_2 and L_3 contains only queue edges. Since no edges cross in this drawing we can conclude that the K_6 is a graph with a mixed layout consisting of one stack and one queue.

Figure 6(b) shows a possible mixed layout of the complete graph K_8 with 8 vertices using one input-restricted deque drawn between L_1 and L_2, one queue between L_2 and L_3 and one stack between L_3 and L_4. Note that each edge of the K_8 appears in the representation of exactly one data structure and the same linear layout π is used for all data structures.

5 Conclusion and Future Work

In this paper we introduced the new graph visualization technique by linear cylindric drawings. We proved that the class of graphs that have a plane linear

cylindric drawing are exactly those graphs that permit a layout by the deque data structure. We also showed how our new representation gives deeper insights into the fundamental data structures queue, stack and deque and can even be used to investigate graph layouts with mixed data structures.

The decision problem whether or not a graph permits a stack layout can be solved in linear time [1, 11]. In contrast the corresponding decision problem for a queue layout is $\mathcal{NP}$-hard [9]. The question is open whether or not the decision problem in the case of a deque is solvable in polynomial time. Currently we are in the progress of proving that the class of deque graphs coincides with the class of graphs that are a subgraph of a planar graph with a Hamiltonian path. These are new insights we gained by linear cylindric drawings, which also give new characterizations of, e.g., queue graphs and proper level planar graphs.

References

1. Bernhart, F., Kainen, P.: The book thickness of a graph. J. Combin. Theory, Ser. B 27(3), 320–331 (1979)
2. Chung, F.R.K., Leighton, F.T., Rosenberg, A.L.: Embedding graphs in books: A layout problem with applications to VLSI design. SIAM J. Algebra. Discr. Meth. 8(1), 33–58 (1987)
3. Cornelsen, S., Di Stefano, G.: Track assignment. J. Discrete Algorithms 5(2), 250–261 (2007)
4. Dujmović, V., Wood, D.R.: On linear layouts of graphs. Discrete Math. Theor. Comput. Sci. 6(2), 339–358 (2004)
5. Dujmović, V., Wood, D.R.: Stacks, queues and tracks: Layouts of graph subdivisions. Discrete Math. Theor. Comput. Sci. 7, 155–202 (2005)
6. Heath, L.S., Leighton, F.T., Rosenberg, A.L.: Comparing queues and stacks as mechanisms for laying out graphs. SIAM J. Discret. Math. 5(3), 398–412 (1992)
7. Heath, L.S., Pemmaraju, S.V.: Stack and queue layouts of directed acyclic graphs: Part II. SIAM J. Comput. 28(5), 1588–1626 (1999)
8. Heath, L.S., Pemmaraju, S.V., Trenk, A.N.: Stack and queue layouts of directed acyclic graphs: Part I. SIAM J. Comput. 28(4), 1510–1539 (1999)
9. Heath, L.S., Rosenberg, A.L.: Laying out graphs using queues. SIAM J. Comput. 21(5), 927–958 (1992)
10. Knuth, D.E.: The Art of Computer Programming, 1st edn., vol. 1. Addison-Wesley, Reading (1968)
11. Wiegers, M.: Recognizing outerplanar graphs in linear time. In: Tinhofer, G., Schmidt, G. (eds.) WG 1986. LNCS, vol. 246, pp. 165–176. Springer, Heidelberg (1987)
12. Wood, D.R.: Bounded degree book embeddings and three-dimensional orthogonal graph drawing. In: Mutzel, P., Jünger, M., Leipert, S. (eds.) GD 2001. LNCS, vol. 2265, pp. 312–327. Springer, Heidelberg (2002)
13. Wood, D.R.: Queue layouts, tree-width, and three-dimensional graph drawing. In: Agrawal, M., Seth, A. (eds.) FSTTCS 2002. LNCS, vol. 2556, pp. 348–359. Springer, Heidelberg (2002)
14. Yannakais, M.: Embedding planar graphs in four pages. J. Comput. Syst. Sci. 38(1), 36–67 (1989)