Testing Planarity by Switching Trains

Christopher Auer, Andreas Gleißner, Kathrin Hanauer, and Sebastian Vetter

University of Passau, 94030 Passau, Germany
{auerc,gleissner,hanauer,vetter}@fim.uni-passau.de

We show that planarity testing can be interpreted as a train switching problem. Train switching problems have been studied in the context of permutation networks, i. e., permuting the cars of a train on a given railroad network [5]. The cars enter the network one at a time, some are stored temporarily in the network and the cars leave the network in the prescribed permutation. For the planarity test we use the metaphor of train switching in the context of graph layouts. In a graph layout the vertices are processed according to a total order, i. e., an ordering of the vertices, which is called *linear layout*. The edges are data items that are inserted to and removed from a given data structure. The vertices are processed in the order of the linear layout. At each vertex, at first all edges incident to preceding vertices are removed from the data structure and then all edges incident to succeeding vertices are inserted into the data structure. These operations must obey the principles of the underlying data structure, such as "LIFO" for a stack or "FIFO" for a queue. A graph G is a stack graph, i. e., has a stack layout, if and only if it is outerplanar, and it is a 2-stack graph if and only if it is a subgraph of a planar graph with a Hamiltonian cycle [3].

In [1,2], we have studied graph layouts in the deque: A *deque* has two ends, a head and a tail, to insert and remove edges. A deque can emulate two stacks and additionally allows for "queue edges", i. e., edges inserted and removed at opposite sides. In [1], we introduced *linear cylindric drawings (LCDs)* to visualize deque layouts and showed that plane LCDs characterize deque graphs. An example of an LCD is shown on the poster[1] in "Deque Layout". The vertices are drawn left-to-right in the order of the linear layout. The sides of insertion and removal of the edges in the deque layout are defined by the embedding. For instance, edge $\{1, 4\}$ leaves vertex 1 and enters vertex 4 below the path which means that edge $\{1, 4\}$ is inserted and removed at the tail of the deque. Edge $\{2, 5\}$ leaves vertex 2 above and enters vertex 5 from below and is therefore inserted at the head and removed at the tail.

The metaphor of train switching can readily be applied to deque layouts to obtain a *deque train switching problem*: The linear layout corresponds to a railroad without junctions. The train stations, i. e., the vertices, are positioned along the railroad in the order of the linear layout. If e is an edge inserted into the deque at vertex u and removed at v, then e corresponds to a car that has to be transported from station u to station v. The train that moves the cars corresponds to the deque. At each station v, the cars destined for v have to be removed from the head/tail of the train. All cars with source station v are

[1] http://www.infosun.fim.uni-passau.de/br/publications/gd12-sd-poster.pdf

W. Didimo and M. Patrignani (Eds.): GD 2012, LNCS 7704, pp. 557–558, 2013.

prepended and appended to the train. We obtain that a graph is a deque graph if and only if the corresponding deque train switching problem is solvable.

Deque graphs are characterized as the graphs that are subgraphs of planar graphs with a Hamiltonian path [2]. A single deque is still not able to layout all planar graphs [2]. However, the ability to split a deque into pieces aptly enhances it to characterize planarity. We call this extended deque *splittable deque (SD)*. For the SD, linear layouts are generalized to tree layouts: A tree layout is a DFS tree of a connected graph. The processing starts at the root of the tree and an empty SD. At each vertex v, the SD is split into k pieces, where k is the number of children of v in the DFS tree. These pieces are consecutive in the sense that their concatenation results in the original SD and each piece is assigned to a child of v as the child's input. Each edge to a tree ancestor of v is removed from the head or tail of the corresponding piece. An edge to a descendant of v has to be inserted at the head or tail of the piece belonging to the according subtree. If v is a leaf, the SD is not split and must be emptied. We have the following theorem, where the proof is based on the Fraysseix-Rosenstiehl's planarity criterion [4]:

Theorem 1. *A graph is planar if and only if it is an SD graph.*

SD layouts can be visualized similarly to deque layouts. An example is shown in "Splittable Deque Layout", where the DFS tree edges are drawn bold. Along a path from the root to a leaf, e. g., vertex 7, the SD layout is equal to a deque layout: The path itself is the linear layout and the embedding defines the sides of insertions/removals, e. g., edge b is inserted at the head and removed at the tail. At vertex 3, the SD is split into three pieces and each piece is assigned to one of its children.

The metaphor of train switching can be applied to SDs to obtain the *SD train switching problem* (see "Train Switching Example"). Here, the railroad has a tree structure. At the junctions, i. e., the branches in the DFS tree, the train is split. Hence, a graph is planar if and only if its SD train switching problem is solvable.

References

1. Auer, C., Bachmaier, C., Brandenburg, F.J., Brunner, W., Gleißner, A.: Plane Drawings of Queue and Deque Graphs. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 68–79. Springer, Heidelberg (2011)
2. Auer, C., Gleißner, A.: Characterizations of Deque and Queue Graphs. In: Kolman, P., Kratochvíl, J. (eds.) WG 2011. LNCS, vol. 6986, pp. 35–46. Springer, Heidelberg (2011)
3. Bernhart, F., Kainen, P.: The book thickness of a graph. J. Combin. Theory, Ser. B 27(3), 320–331 (1979)
4. de Fraysseix, H., Rosenstiehl, P.: A depth-first-search characterization of planarity. In: Graph theory (Cambridge, 1981). Ann. Discrete Math., vol. 13, pp. 75–80. North-Holland, Amsterdam (1982)
5. Tarjan, R.: Sorting using networks of queues and stacks. J. ACM 19(2), 341–346 (1972)