Karlsruhe Institute of Technology

Upward and Upward-Planar Drawings with Limited Slopes

Bachelor's Thesis of

Valentin Andreas Quapil

At the Department of Informatics
Institute of Theoretical Informatics

Reviewers:	PD Dr. Torsten Ueckerdt
	Dr. Thomas Bläsius
Advisors:	Paul Jungeblut

Time Period: 3rd May 2021 – 3rd September 2021

Statement of Authorship

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Karlsruhe, August 31, 2021

Abstract

This thesis considers upward and upward-planar drawings with a limited number of distinct slopes. A drawing of a directed graph is called *upward* if all edges are y-monotone ("point up"). We compare the number of distinct slopes needed to draw directed graphs upward and/or planar using only straight line segments.

We present existing algorithms for upward planar drawings of trees and cacti with a limited number of slopes and provide new algorithms for drawing series-parallel graphs upward planar with two or three slopes. We show a lower bound for the space requirements of such drawings: There are series-parallel graphs, cacti, and ordered trees whose upward 3-slope planar drawings require exponential width and height. Here, all vertices must have integer coordinates.

The decision problem whether the number of slopes of an upward planar graph is larger than its maximum in- and outdegree is proven to be $\exists\mathbb{R}$-complete.

Deutsche Zusammenfassung

Diese Bachelorarbeit beschäftigt sich mit upward und upward-planaren Zeichnungen mit begrenzter Anzahl an verschiedenen Steigungen. Eine Zeichnung eines gerichteten Graphen wird *upward* genannt, wenn alle Kanten y-monoton sind („nach oben zeigen"). Wir vergleichen für gerichtete Graphen die mindestens benötigte Anzahl an unterschiedlichen Steigungen für eine geradlinige Zeichnung, die upward und/oder planar ist.

Zum upward planaren Zeichnen von Bäumen und Kakteen mit einer begrenzten Anzahl von Steigungen werden bereits existierende Algorithmen vorgestellt. Für serien-parallele Graphen geben wir neue Algorithmen zum upward planaren Zeichnen mit zwei oder drei Steigungen an.Wir zeigen eine untere Schranke für den Platzbedarf solcher Zeichnungen: Es gibt serien-parallele Graphen, Kakteen und geordnete Bäume, deren upward planare Zeichnungen mit drei Steigungen exponentielle Breite und Höhe benötigen. Hierbei haben alle Knoten ganzzahlige Koordinaten.

Wir zeigen, dass das Entscheidungsproblem, ob die Anzahl der Steigungen eines upward planaren Graphen größer ist als sein maximaler Eingangs- und Ausgangsgrad, $\exists\mathbb{R}$-vollständig ist.

Contents

1. Introduction

Motivation

In Graph Drawing, we want to find algorithms for and limitations of embedding graphs in the euclidian plane. Furthermore, the drawings have to satisfy certain *drawing restraints* and often try optimizing a graphic property (*aesthetics*). There can also be *local constraints*, like fixing absolute or relative positions of several vertices.

This task is motivated in two ways, an applicative and a theoretical.

The applicative motivation is that we want to automate the presentation of graphs to users. The drawing restraints and aesthetics are put in place to improve the readability of graph drawings. Local constraints can be used to ensure similarity between the drawing of, for example, a metro map, and the geographic layout. When studying existing examples of graph drawings (family trees, metro maps, and circuit schematics, and so on) we can identify common drawing restraints:

- edges are straight line segments
- there are few or no crossings
- using a limited set of the slopes for all edges
- upward drawing of directed edges

The last one, upward drawing, often occurs in drawings of family trees, UML-class diagrams, and many other directed graphs (*digrahps*) which contain a hierarchy. Typical aesthetics are, for example:

- minimizing the number of crossings
- minimizing the area of the drawing on the integer grid
- avoiding shallow angles at nodes
- minimizing the number of bends

On the other hand, from a theoretical perspective we want to research the influence of the restraints on the answer to the following central questions: *Which* graphs (or graph classes) can be drawn while satisfying certain drawing restraints? And *how much effort* does it take, computationally, to construct such a drawing? The first question can sometimes not be

answered directly. Then we will focus on the hardness of finding the answer: How difficult is it to find out, if a graph can be drawn while satisfying certain drawing restraints?

In this thesis, we combine upward drawing with a limit on the number of distinct slopes in use. While these restraints arise from practical applications, they are also interesting on the theoretical side. Especially, we are interested in the difference the upward restraint makes to the possibility and complexity of constructing (planar) drawings with a limited number of slopes.

Related Work

The main inspiration of this paper is [KZ21] from Jonathan Klawitter and Johannes Zink. They have shown that it is **NP**-hard to find out if an outerplanar directed graph can be drawn upward straight-line planar with three slopes. This result leaves the challenge to find algorithms for other graph classes. They also find an algorithm for drawing cacti upward planar with three slopes in polynomial time.

[Nö05, Section 6.1.2] is a good example of a constraint similar to the constraint of drawing upward with a limited amount of slopes we use throughout this thesis. In this case, four slopes (eight directions) are used in general, but for every edge, there are only three directions allowed (the ones that correspond the best to the geographical situation). He provides a mixed-integer program for drawing with those constraints among others.

Nadine Krisam [Kri18] has investigated drawings of level planar graphs with two slopes. She uses a flow-based approach and finds an algorithm to draw level planar graphs with two slopes in polynomial time if such a drawing is possible.

[DLM20] shows that every series-parallel graph whose maximum vertex degree is Δ admits an upward planar drawing with at most one bend per edge such that each edge segment has one of Δ distinct slopes. The construction is worst-case optimal in terms of the number of slopes, and it gives rise to drawings with optimal angular resolution π/Δ. This result relates closely to our contribution that for maximum in- and outdegree ≤ 3 we need to use only three slopes in all cases and we can get rid of the bend in some cases.

Contribution

There are three major contributions throughout this thesis.

The first one deals with upward planar drawings of cacti: We find out that series-parallel graphs can be drawn upward planar using two (or three) slopes if and only if the maximum in- and outdegree is 3 and there is no *bad edge*, which is a special type of transitive edge. An algorithm for constructing a corresponding drawing is provided in both cases with and without a given bimodal embedding. Using this result, we also find an upward planar 1-bend drawing using three slopes for any series-parallel graph with maximum in- and outdegree three.

The second contribution is that we find lower bounds on the worst-case area of upward 3-slope planar drawings on the integer grid (all vertices have integer coordinates). Although the 2-slope drawings of series-parallel graphs constructed in the first contribution have linear width and height, for 3-slope drawings we find a series of graphs with exponential width and height. For directed cacti, we also find a series of graphs that need exponential width and height for being drawn upward 3-slope planar. For trees the worst-case area remains an open problem, however, if we fix the embedding of a directed tree, we get exponential width and height.

Thirdly, we show $\exists\mathbb{R}$-completeness for the upward planar slope number problem. Therefore we specify a verification algorithm running on the Real-RAM and, with a few modifications,

apply the construction used to prove $\exists\mathbb{R}$-hardness of the planar slope number problem in [Hof17] to the upward planar slope number problem.

Outline

The thesis is structured the following way: In Chapter 2, we introduce notation, names of graph classes, and drawing restraints used throughout the thesis.

In the first half of the thesis we focus on exploring the possibilities of drawing upward and/or planar with a limited number of slopes: In Chapter 3 we take a broad look at limiting the number of allowed slopes. We first analyze which sets of allowed slopes are interchangeable when drawing upward planar, and then identify relations between upward, upward planar, and planar drawings with limited slopes. In Chapter 4 the algorithms for drawing trees, cacti, and series-parallel graphs are presented.

In the second half of the thesis we take a closer look at the limitations of drawing upward planar: In Chapter 5, for the first time, we take drawing aesthetics into account and find out that the drawings constructed in Chapter 4 need exponential width and height. Finally, in Chapter 6 we find out that even when forcing edges to be oriented upward, the planar slope number problem does not get easier. This is done by showing that the upward planar slope number problem is $\exists\mathbb{R}$-complete.

2. Preliminaries

Notation

notation	definition	meaning
$\binom{V}{k}$, V finite set, k integer	$\{S \subset V : \lvert S\rvert = k\}$	the set of all k-element subsets of V
$[n]$, n integer	$\{1, 2, \ldots, n\}$	the set of the first n positive integers

Definition 2.1. A *drawing* of a graph maps each vertex to a point in the plane ($\mathbb{R}^2$) and each edge to a simple open Jordan curve with corresponding vertices at its endpoints.

A *combinatorial embedding* of a graph is the clockwise order (*cw-order*) of all incident edges for every vertex in one drawing of the graph. Throughout the thesis a combinatorial embedding of a planar graph always includes fixing the outer face.

An edge (u, v) of an acyclic directed graph (*digraph*) is called *transitive* if there exists a directed path from u to v different from (u, v).

2.1 Graph Classes

Throughout the thesis, we will only consider graphs without loops and double edges.

Definition 2.2. A graph is a *cactus*, if it is connected and all edges belong to at most one cycle. Cacti are a subclass of planar graphs.

A digraph is called *cactus digraph*, if its underlying undirected graph is a cactus.

Definition 2.3. A *series-parallel graph* (*SP-graph*) is a directed graph with a fixed source and a fixed sink vertex. SP-graphs are defined by one base case and two composition operations. The base case is a single edge (s, t). s is the source vertex and t is the sink vertex. For composition two arbitrary SP-graphs with disjoint vertices can be used. They are called the (*parallel* or *series*) *components* of the new graph. The composition operations are:

- parallel composition: the sources and sinks of the two components are merged, respectively.

- series composition: the sink of the first component is merged with the source of the second component. The source of the new SP-graph is defined by the source of the first component and the sink of the new SP-graph is defined by the sink of the second component.

One can characterise a series-parallel graph using the *SPQ-tree* (or *decomposition-tree*). This is an ordered, rooted binary tree. It is built analogue to the definition above:

- A single edge is represented by a Q-node.
- A parallel composition is represented by a P-node with the SPQ-trees of the two parallel components as children. If an embedding is given, then the left child must represent the left component and the right child must represent the right component.
- A series composition is represented by a S-node. Again, the SPQ-trees of the components are the children. The left child represents the lower component and the right child represents the higher component.

Note that the SPQ-tree of a series-parallel graph is not always unique.

2.2 Drawing Restraints

Here we provide an overview of all drawing restraints used throughout the thesis.

Definition 2.4 (On the Integer Grid)**.** A drawing is on the *integer grid* if the x- and y-coordinates of all vertices are integers.

Definition 2.5 (Planar)**.** A graph G is considered *planar* if it can be drawn in the plane in such a way that no edges cross each other. Drawings without crossing edges are called *planar* drawings.

Definition 2.6 (Upward)**.** A given digraph $G = (V, E)$ together with a drawing is considered *upward* if the drawing of each edge of G is a y-monotone curve (i.e. when following the curve of an edge in its given direction, the y-coordinate increases on any subsection of the curve).

Observation

Not every graph that is upward and planar is also upward planar (meaning: there is a drawing of the graph that is upward and planar at the same time). An example for this was given in [BETT99, Chapter 6, p. 171] (see Figure 2.1).

Definition 2.7. A *bimodal embedding* of a digraph is a (combinatorial) embedding corresponding to some upward drawing of the graph. It is called bimodal, because for the cw-order at any vertex the incident outgoing and incident incoming edges appear separated in those two groups (we can start in the cw-order at some incoming edge and will first traverse all incoming edges and then all outgoing edges).

Definition 2.8 (k-Slope Drawing)**.** The *slope* of a straight (undirected) edge or line is defined as the angle it forms with the x-Axis. The slope is always in $[0, \pi)$. A horizontal edge has slope 0, a vertical edge has slope $\pi/2$ and so on. Note that for upward-drawing slope 0 is forbidden. We will refer to the steepness of a (non-vertical) line or edge (for example -1 for a line with slope $3/4 \cdot \pi$) as the *gradient*. A non-vertical line or non-vertical straight edge with slope s has a *gradient* of $tan(s)$.

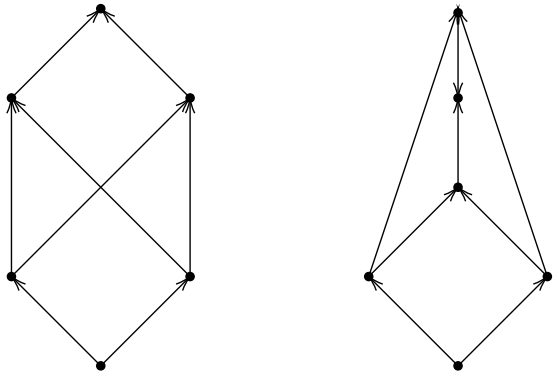

Figure 2.1: A digraph that can be drawn upward (left), planar (right), but not upward planar.

When all the edges of a drawing of a graph or digraph are straight lines we can count the number of distinct slopes. The drawing is a *k-slope drawing*, if it uses at most k different slopes for all edges.

Definition 2.9 (Upward k-Slope Planar)**.** A graph G is called *upward k-slope planar* (up$k$p), if there exists an upward plane k-slope embedding of G.

3. Drawability

Introduction

In this Chapter we are dealing with the following question: Can one draw a given graph with only using k slopes?

First, we take a look at the consequences of choosing a fixed slope set. We find that all slope sets of size two and three are interchangeable when drawing upward planar, respectively. We then prove that this is not the case for slope sets of size four. After that, we compare the effect of changing the drawing restraints on the minimum number of slopes for a fixed graph. We find some bounds on the slope number dependent on the maximum degree.

3.1 Equivalent Slope Sets

When fixing three slopes, we want to only care about some fixed slopes, namely $\pi/4$, $\pi/2$, and $3/4 \cdot \pi$, to get nice drawings.

Before doing so, we first need to analyse which slope sets are interchangeable when drawing upward planar. Formally, we define an equivalence relation for slope sets.

Definition 3.1. A *slope set* is a finite subset of $(0, \pi)$. Zero is not allowed as slope, because we focus on upward drawing. Two slope sets S_1, S_2 are *equivalent* if and only if any digraph that can be drawn upward planar with straight lines using S_1 can also be drawn that way using S_2, and and vice versa.

Lemma 3.2. *Slope sets can only be equivalent to slope sets of the same size.*

Proof. We define star-shaped graphs that are witnesses for any slope sets of different size. Given slope set S_1 of size k_1 and slope set S_2 of size $k_2 < k_1$, then the graph $G = (\{0, 1, \ldots, k_1\}, \{(0, i) \mid i \in \{0, 1, \ldots, k_1\}\}$ can be drawn upward planar with straight lines using S_1 trivially but can not be drawn upward planar with straight lines using slopes from S_2, because vertex 0 has outdegree k_1, but we have less than k_1 available slopes. □

Lemma 3.3. *All slope sets of size three form an equivalence class under the stated equivalence relation.*

Proof. Consider any graph G that is drawable upward planar using any slope set $\{\alpha, \beta, \gamma\}$ $(\alpha > \beta > \gamma)$ of size three. We provide a linear transformation that transforms the drawing to an upward three-slope planar drawing of G that uses the slope set $S \coloneqq \{\pi/4, \pi/2, 3/4 \cdot \pi\}$.

The linear transformation consists of four steps (see Figure 3.1):

- Compress or stretch along the horizontal axis by some factor $M_1 > 0$ so that $\alpha' - \gamma' = \pi/2$. This is always possible as $0 < \alpha - \gamma < \pi$.
- Rotate by γ' clockwise, so that $\gamma'' = 0$. Now $\alpha'' = \pi/2$.
- Compress or stretch along the horizontal axis by some factor $M_2 > 0$ so that $\beta^{(3)} = \pi/4$. This is always possible as $0 < \beta'' < \alpha'' = \pi/2$. Note that the other slopes stay the same.
- Rotate by $\pi/4$ so that the resulting slopes are exactly the slopes in S.

We can use the inverse of this transformation to get to any other slope set of size three. □

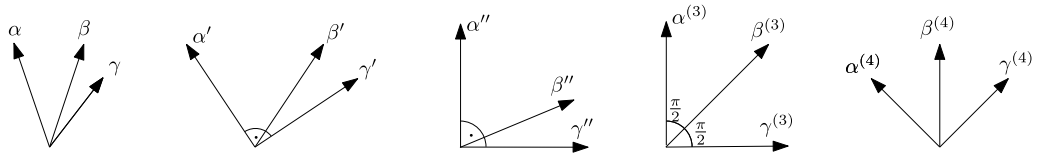

Figure 3.1: Visualisation of the steps of a linear transformation of three arbitrary slopes to some standard fixed slopes. Operations used: stretching, rotating, stretching, rotating.

Theorem 3.4. *There are infinitely many equivalence classes of slope sets of size four.*

Note

Every slope set of size four has infinitely many equivalent slope sets, which can be shown by applying linear transformations. Further, every slope set is equivalent to a slope set of standardized form: Let the given slope set consist of slopes $s_1 > s_2 > s_3 > s_4$. Apply the transformation from the proof of Lemma 3.3 with $\alpha = s_1, \beta = s_3, \gamma = s_4$. The transformation maps s_1, s_3, and s_4 to the same slopes as in the previous Lemma. s_2 is mapped to some slope between the mapping of s_1 and s_3. Therefore, we can say that the resulting slope set is of the following form:

Definition 3.5. A *standard slope set* is a set of the form $S_\alpha = \{\pi/4, \pi/2, \alpha, 3/4 \cdot \pi\}$ with $\pi/2 < \alpha < 3/4 \cdot \pi$.

Because of this equivalence of any slope set to a standard slope set we only focus on standard slope sets. It remains to show that there are infinitely many pairwise non-equivalent standard slope sets.

Definition 3.6. The square gadget is a directed graph with vertices [11] and edges as shown in the two upward 4-slope planar drawings in Figure 3.2.

Lemma 3.7. *When drawing the square gadget upward planar using only the slopes of a standard slope set ($\{\pi/4, \pi/2, \alpha, 3/4 \cdot \pi\}$), the outer face is always bounded by four edges forming a square or a rectangle of width w and height $\tan(\alpha - \pi/4) \cdot w$ (when up is north east, as in Figure 3.2). The four edges on the outer face have slopes $\pi/4$ and $3/4 \cdot \pi$.*

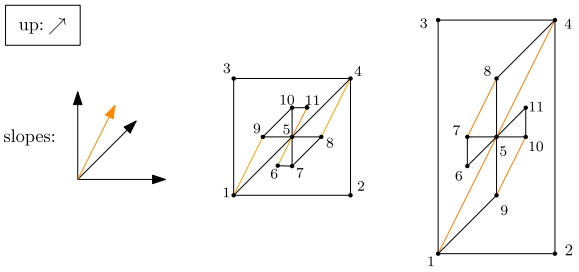

Figure 3.2: Two upward 4-slope drawings of the square gadget. The left drawing has same width and height. The gadget is closed. The right drawing has width 1 (length of edge $(1, 2)$) and height $\tan(\alpha - \pi/4)$ (length of edge $(2, 4)$). The gadget is open.

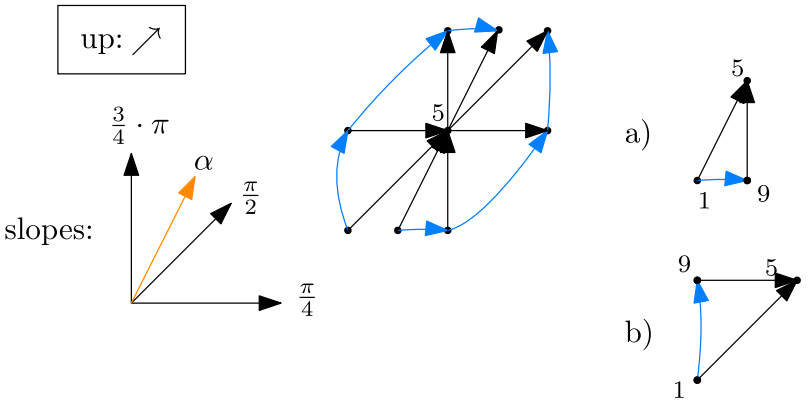

Figure 3.3: Sketch of the proof of Lemma 3.7. There are only two ways to draw the edges incident to vertex 5 of the square gadget.

Proof. Look at vertex 5. It has full degree. The adjacent vertices are connected by two paths, $(1, 9, 10, 11)$ and $(6, 7, 8, 4)$. These paths fix the slopes of incident edges of vertex 5 to two possible solutions (see Figure 3.3). In both cases we prove that any upward planar drawing using a standard slope set must follow the conditions formulated in the lemma.

a) The path $(1, 9, 10, 11)$ is on the right side of vertex 5. The edge $(1, 9)$ must have slope $\pi/4$ or $\pi/2$. Because of the directed edge $(5, 4)$ the vertices $2, 3$, and 4 must be outside of the triangle formed by vertices $1, 9$, and 5. This fixes slope $\pi/2$ for edge $(1, 9)$. The same argument can be made for edge $(8, 4)$. The edges between vertices $1, 2, 3$, and 4 must use the remaining slopes ($\pi/4$ and $3/4 \cdot \pi$). Because these are the highest and lowest slopes, 1 is the single source, and 4 is the single sink of the graph, the edges $(1, 2)$, $(1, 3)$,$(2, 4)$, and $(3, 4)$ bound the outer face.

   The slope of the edges $(1, 5)$ and $(5, 4)$ is α. The height of the drawing (the length of the edge with slope $3/4 \cdot \pi$ starting at vertex 1) must be $\tan(\alpha - \pi/4)$ times the width of the drawing (the length of the edge with slope $3/4 \cdot \pi$ starting at vertex 1).

b) If the path $(1, 9, 10, 11)$ is on the left side of vertex 5, the edge $(1, 9)$ must have slope α or $3/4 \cdot \pi$. With the same argumentation as above we find that it must have slope α. The edges between vertices $1, 2, 3$ and 4 must use slopes $\pi/4$ and $3/4 \cdot \pi$ and bound the outer face.

   The slope of the edges $(1, 5)$ and $(5, 4)$ is $\pi/2$. The height and width of the drawing must be the same and the edges bounding the outer face form a square.

□

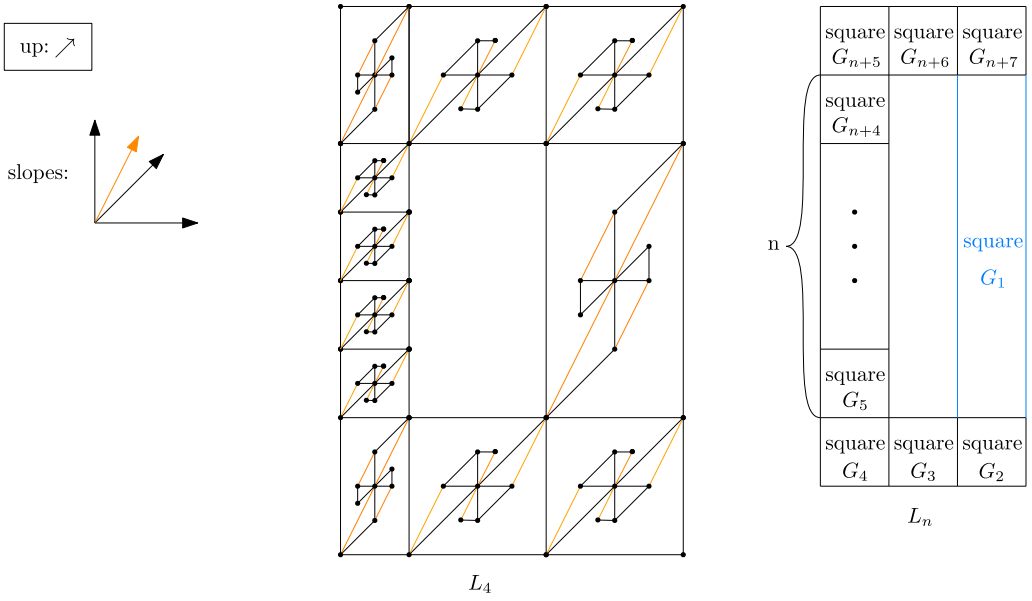

Figure 3.4: A Series of digraphs, so that for every pair of them there exists a standard slope set that can only be used for drawing one of them upward planar.

Proof of Theorem 3.4. In this proof up will again be north-east. We combine many square gadgets to obtain the graph L_i ($i \geq 2$) as shown in Figure 3.4. The graph consists of copies of the square gadget with some merged vertices. The copies of the gadget are labelled $G_i, i \in \mathbb{N}$.

We already know that the boundary of an up4p drawing of the square gadget is always rectangular and either both sides have the same length. Then we call the gadget *closed*. Or, depending on the standard slope set S_α used, its height is $s \coloneqq \tan(\alpha - \pi/4)$ times its width. Then the gadget is *open*.

The gadget L_{n^2} ($n > 1$) can be drawn upward planar using the slope set $S_{\arctan(n)+\pi/4}$: Only open G_1, G_4, and G_{n^2+5}. All copies $G_k, 5 \leq k \leq n^2 + 4$ have width and height 1. The bottom row has height n, G_2 has width n, and G_1 has width n and height n^2. An example for $n = 2$ can be seen in Figure 3.4.

We now show that the gadget L_{n^2} ($n > 1$)can not be drawn upward planar using a slope set S_α, if $s \coloneqq \tan(\alpha - \pi/4) < n$ (s is the height of an open widget with width 1).

Look at an upward planar drawing of L_{n^2} using a slope set, so that s is the height of an open widget with width 1. If the row of n^2 gadget copies is horizontal, we first apply a linear transformation. This process is illustrated in Figure 3.5: Flip the x-axis (north) and y-axis (east) and scale the x-axis of the drawing by $1/s$.

The row of n^2 gadget copies is now vertical. We scale the drawing without changing the slopes, so that G_4 has width 1. As shown in Figure 3.6, we measure the lengths h, l_1 and l_2. If G_4 is open, then $l_1 = s$. If it is closed, then $l_1 = 1$. l_1 is also the height of G_2. Depending on its shape (open or closed), G_2 has width $l_2 \in \{1/s, 1, s\}$. Depending on the shape of G_1, its height is $h \in \{1/s, 1, s, s^2\}$. Therefore, we now have an upper bound on the height:

$$h \leq s^2$$

On the other hand, the row of n^2 gadget copies give us a lower bound on h: The height is minimized if all of these gadgets are closed. Then they all have width and height 1. Therefore we get:

$$h \geq n^2$$

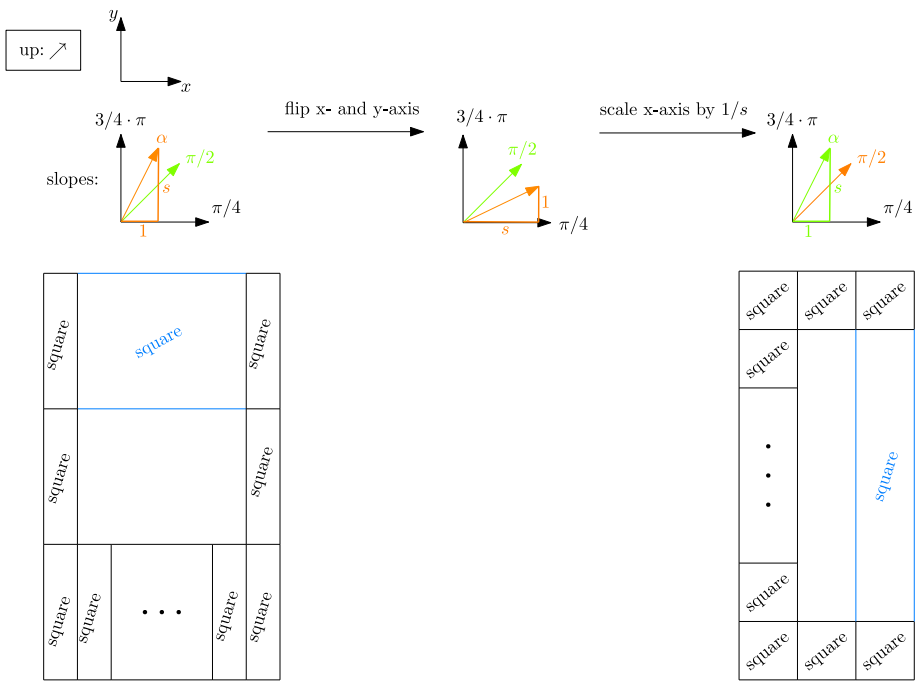

Figure 3.5: Illustration of how L_n can be transformed if it is "laying" instead of "standing". The set of used slopes remains the same.

If we combine both inequalities, we finally get:

$$s \geq n$$

This shows that the graph L_{n^2} can not be drawn upward planar using a slope set S_α, if $\tan(\alpha - \pi/4) = s < n \iff \alpha < \arctan(n) + \pi/4$ and it can be drawn upward planar if $\alpha = \arctan(n) + \pi/4$. Note that the transformation applied to the inequality is only possible because $\pi/4 < \alpha - \pi/4 < \pi/2$ and $\tan(x)$ is strictly monotone in $(\pi/4, \pi/2)$.

To conclude the proof we now know that the slope sets

$$\{S_\alpha \mid \alpha = \arctan(i) + \pi/4, 1 < i \in \mathbb{N}\}$$

are pairwise not equivalent. □

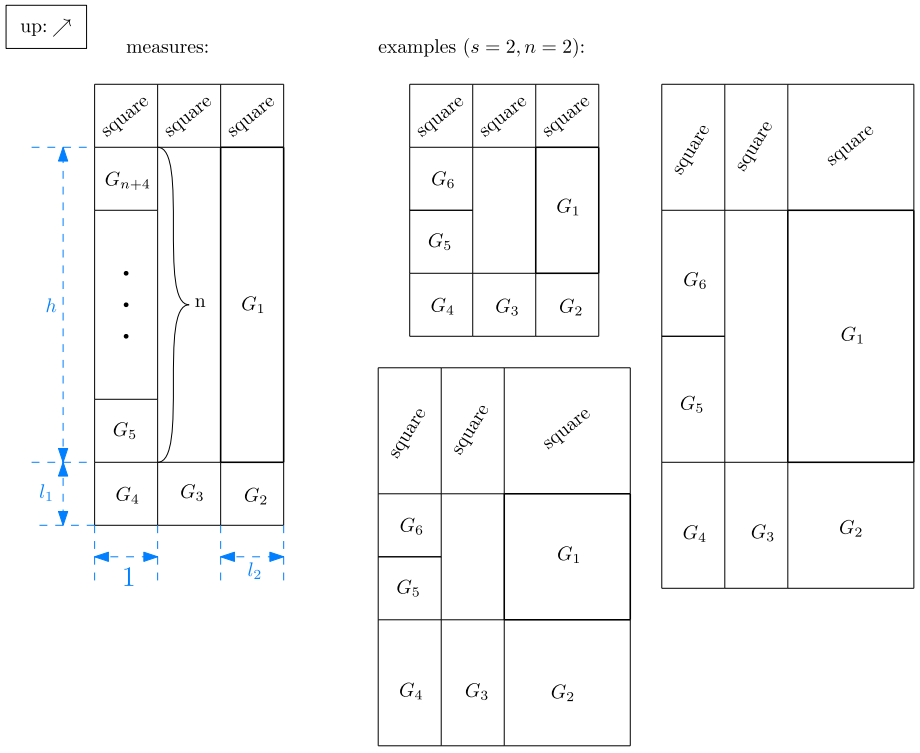

Figure 3.6: left: A drawing of L_n with variables for the length of certain edges. right: possible drawings of L_2 with different combinations of open and closed gadget copies with $s = 2$. In any case, the row of square gadgets must have the same height as G_1.

3.2 Slope Number Comparisons

There is a clear relationship between the upward slope number of a digraph and the slope number of the underlying graph:

Lemma 3.8. *For the upward slope number* $\mathrm{usn}(D)$ *of any acyclic digraph* D *the slope number* $\mathrm{sn}(G)$ *of the underlying undirected graph* G *is a lower bound, or in short:*

$$\mathrm{usn}(D) \geq \mathrm{sn}(G)$$

Equivalently, for the upward planar slope number $\mathrm{upsn}(D)$ *and the planar slope number* $\mathrm{psn}(G)$ *we get*

$$\mathrm{upsn}(D) \geq \mathrm{psn}(G)$$

Proof. Let D be some directed graph and let G be its underlying undirected graph. If D has upward slope number $\mathrm{usn}(D) = k$, then we can find an embedding using k slopes. The same embedding is also valid for G, thus $\mathrm{sn}(G) \leq k$.

The same argument also yields the result in the planar case. □

3.2.1 Small Maximum Degree

We have seen that adding the restraint of drawing upward can increase the number of slopes needed for a drawing. To find out how big this increase is, we look at the minimum and maximum slope numbers when fixing the maximum degree.

Lower Bound

There is a relation between the slope number $\text{sn}(G)$ of a (undirected) graph G and its maximum degree $\Delta(G)$. The trivial lower bound is that

$$\text{sn}(G) \geq \frac{\Delta(G)}{2}$$

because at some vertex with maximum degree, every slope can be used at most two times. This lower bound of course also holds for the planar case, if the graph is planar.

For upward drawings we can improve this bound by differentiating between the maximum indegree $\Delta^+(G)$ and maximum outdegree $\Delta^-(G)$:

$$\text{usn}(G) \geq \max\{\Delta^+(G), \Delta^-(G)\} \geq \frac{\Delta(G)}{2}$$

Every slope can be used at most one time for all incoming edges at a vertex. The edges going into the vertex with maximum indegree $\Delta^+(G)$ therefore need $\Delta^+(G)$ pairwise different slopes. The same argument holds true for outgoing edges and $\Delta^-(G)$.

Again, this lower bound also holds for the upward planar case, if the graph is upward planar.

Upper Bound

The upper bound is more interesting. We know the following **tight** upper bounds:

max Deg. Δ	slope number (sn)	upward sn	planar sn	upward planar sn
1	1	1	1	1
2	3	3	3	4
3	4 [MS09]	?	?	?
4	?	?	?	?
≥ 5	unbounded	unbounded	bounded	?

The cases for maximum degree 1 are trivial. We always have single edges that can all be placed vertically.

With $\Delta = 2$ we allow paths and circles. In the non-upward cases we can draw the cycles as subdivisions of triangles. In the upward planar case, the worst case is a graph $\{[4], \{(1,3),(2,3),(1,4),(2,4)\}\}$. For drawing upward (upward planar) one can easily find algorithms to draw with three (four) slopes (look at the cases with alternating upward and downward edges on the circle and two consecutive upward edges, illustrated in Figure 3.7).

For $\Delta \geq 3$ there is not much known. [MS09] takes a lot of effort to show that for degree three, any graph can be drawn with straight lines using only four slopes. The slopes used are $\{0, \pi/4, \pi/2, 3/4 \cdot \pi\}$. [DESW07] showed that planar graphs with maximum degree four can be drawn with only three slopes, if three bends are allowed on the outer face. Finding a bound for the slope number for $\Delta = 4$ is a well-known open problem.

For maximal degree Δ we can find an upper bound and a lower bound on the worst-case planar slope number:

$$\text{psn}(\Delta) \in \mathcal{O}(c^\Delta) \cap \Omega(3\Delta - 6), c \in \mathbb{R}$$

The upper bound was found by [KPP10, Theorem 1]. More precisely, it is $\text{psn}(\Delta) \in \mathcal{O}(\Delta^2(3+2\sqrt{3})^{12\Delta})$. The lower bound is stated in [KPP10, Theorem 13].

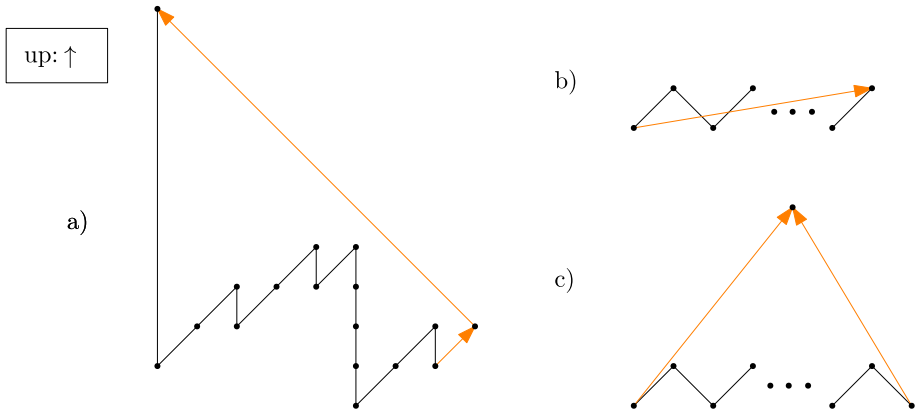

Figure 3.7: Sketch of upward drawing algorithms for digraphs of degree 2. left (a): two consecutive up's (or down's) exist, right: alternating up's and down's, (b: non-planar, c: planar)

3.2.2 Slope Number of Complete Graphs

[DSW07] show that a complete graph on n vertices has slope number n. We take a look at the directed complete graph on n vertices and show that it also has slope number n.

Lemma 3.9. *Let D_n be the acyclic directed graph $D_n := (\{1, 2, \ldots, n\}, \{(i, j) \mid 0 < i < j \leq n\})$ and K_n the complete graph on $n > 2$ vertices. We get*

$$\mathrm{usn}(K_n) = \mathrm{sn}(D_n) = n$$

Note

D_n is not some random acyclic directed graphs with underlying undirected graph K_n. In fact, all such graphs are isomorph (they are the same, except that vertices are named differently). We proof by induction that they are all isomorph to D_n: For $n = 3$ this is trivially the case. For $n > 3$ there must be a vertex v with outdegree $d^-(v) = 0$ (otherwise D has a cycle). Rename v to n and apply induction to the graph without v to rename all other vertices.

Proof of Lemma 3.9. Assume some embedding of K_n using $\mathrm{sn}(K_n)$ slopes. Then we can rotate that embedding so that no two vertices have the same y-coordinate. Now rename the vertices: Numerate from bottom to top from 1 to n and we can see that this is an embedding of D_n using $\mathrm{sn}(K_n)$ slopes. Furthermore for $0 < i < j \leq n$ we have $y(i) < y(j)$ inside the embedding by construction. Therefore we constructed an upward drawing of D_n using $\mathrm{sn}(K_n)$ slopes. □ □

4. up$k$p Drawing Algorithms

[KZ21, Theorem 5] state and prove that deciding whether an upward outerplanar digraph admits an up3p drawing is NP-hard with and without a given embedding. We want to look at other classes where the decision problem is probably easier. At the same time we also look at the corresponding drawing algorithms. We start with subclasses of outerplanar digraphs (trees and cacti) with maximum in- and outdegree three. We then provide a new algorithm to draw series-parallel graphs up3p.

4.1 Trees

Theorem 4.1 ([KZ21])**.** *A directed tree with maximum indegree and outdegree k always admits an* up$k$p *drawing.*

Proof. First pick a root node. We add new vertices and edges to get $T_{k,h}$ (the balanced tree of height h where all inner (non-leaf) vertices have indegree and outdegree k). This tree can be drawn up$k$p by a recursive strategy (for example see Figure 4.1). Each vertex v has some bounding box containing v and its subtree (all vertices that appear in depth-first between the first and the second visit at v). Now we can simply remove the vertices and edges we added at the beginning to obtain a drawing of the original tree. □

[KZ21] also shows that a tree with a given bimodal embedding can be drawn up$k$p in certain cases. The minimal k (the upward planar slope number) can be determined in linear time. This is done by assigning slopes with a greedy algorithm: The slope of an edge (v, w) must be bigger than the slope of the previous outgoing edge of v and the next incoming edge of w in counterclockwise order (ccw-order), if such edge(s) exist. An example can be seen in Figure 4.2.

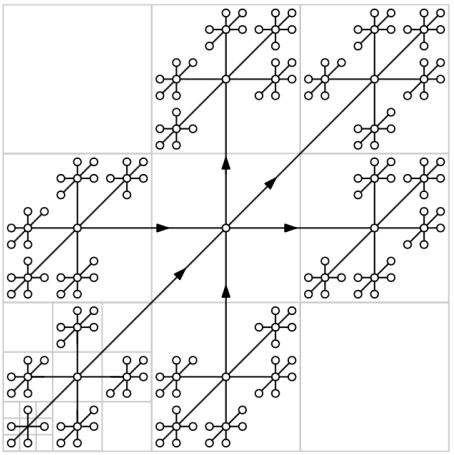

Figure 4.1: The balanced tree $T_{3,3}$ with bounding boxes for some vertices.

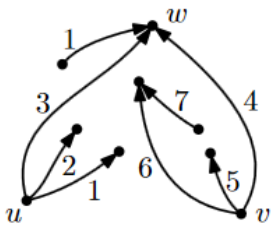

Figure 4.2: A directed tree with given embedding can lead to arbitrary many slopes. Possible slope labels are assigned with a greedy strategy (higher number means higher slope).

4.2 Cacti

In contrast to trees (as in [KZ21, Theorem 1]), not every cactus-digraph with indegree and outdegree at most k admits an up$k$p drawing. A counterexample for $k = 3$ is shown in Figure 4.3.

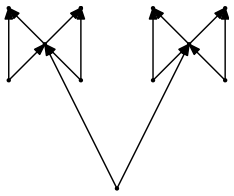

Figure 4.3: A cactus digraph that does not admit an up3p drawing.

Definition 4.2 ((Consistent) Slope Assignment)**.** Given a graph $G = (V, E)$, a *slope assignment* is a function $s : E \to [0, \pi)$. A straight-line drawing of G is *using* that slope assignment if and only if for any edge e the slope of e is $s(e)$. The slope assignment is also called k-slope assignment, where k is the number of distinct slopes that are used.

A slope assignment is *consistent* for a graph G and a set of drawing restraints, if there is a straight-line drawing of G using the slope assignment while also satisfying the drawing restraints.

[KZ21] discusses drawing cacti with uniform angles (slopes are all multiples of a single angle, a fraction of π. However, one could argue that the consistent slope-assignment found can also be used to draw on an (exponentially big) integer grid.

Theorem 4.3 ([KZ21]). *One can test in $\mathcal{O}(k^4n^2)$ whether an upward planar cactus (with or without a bimodal embedding) can be drawn* upkp. *If yes, the test can also return a consistent slope-assignment.*

We do not go into greater detail. The result was found by Klawitter and Zink very recently.

4.3 Series-Parallel Graphs

The class of series-parallel graphs overlaps with the outerplanar graphs (but are neither a super- nor a subclass). We find algorithms for drawing up2p and up3p, but cannot generalise the algorithm that is used to draw series-parallel graphs upward planar (using an unlimited number of distinct slopes) or generalise our own approach for drawing

Note that in contrast to [KM21, Section 4.3] we orient the edges from the source vertex to the sink vertex and do not allow arbitrary direction for the edges.

4.3.1 Unrestricted Number of Slopes

The algorithm presented in [BETT99, Section 3.2.2] achieves linear height and width on the integer grid by a divide and conquer strategy as illustrated in Figure 4.4: The drawing is contained inside an isosceles right-angled triangle. The base is vertical and the other sides of the triangle are to the left of the base. By induction SPQ-tree, the series-parallel graph is drawn with linear width and height: A Q-node is drawn as a single edge of length 1. An S-node is drawn by stacking the drawings of the two series components on top of each other. A P-node is drawn by putting the drawings of the parallel components next to each other and pulling down the source vertices to merge them. The sink vertices are pulled up to merge them as well. To preserve planarity a few invariants have to be taken

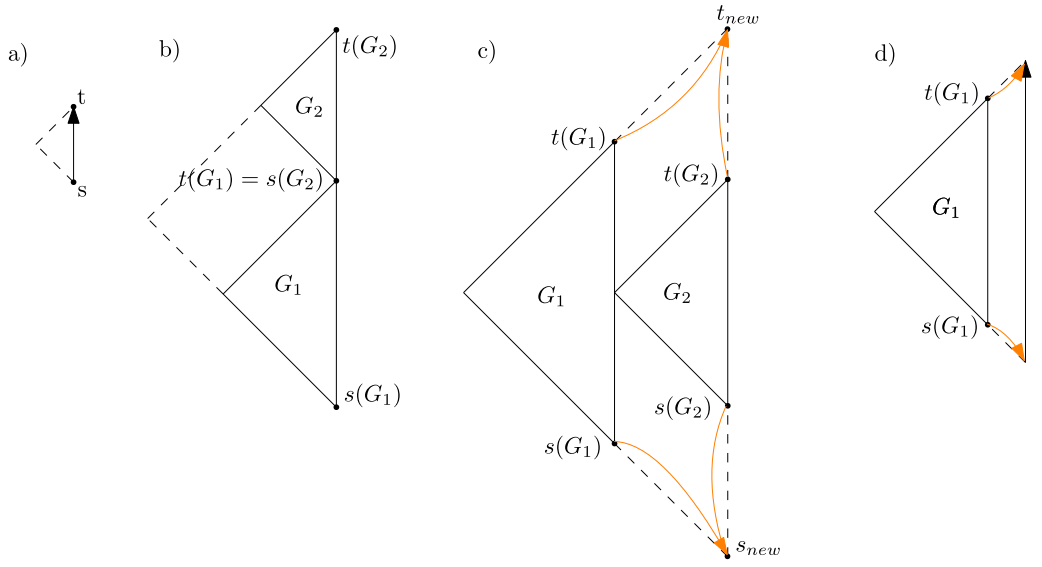

Figure 4.4: Illustration of the divide-and-conquer approach with unlimited slopes. a) a Q-node, b) a S-node and c), d) P-node composition in the resulting drawing. The orange arrows mark how vertices are moved around to obtain a drawing of parallel composite graphs.

into account. We leave them out of this short description.

The important part is that during the P-node merging step vertices are moved around. The slopes of incident edges may change, and there seems to be no easy solution on how to control the slopes. This is why we cannot use this algorithm for drawing with limited number of slopes. We have to figure out another way to merge source and sink vertices in case of parallel composition: We do not draw the parallel components directly but remove certain vertices and edges that are then used to connect the components and can be adjusted in length.

4.3.2 Two Slopes

Theorem 4.4. *A series-parallel digraph is drawable* up2p *(with and without given bimodal embedding) if and only if it has maximum in- and outdegree two and contains no transitive edge.*

Note that a transitive edge of a series-parallel graph can be found in linear time when the SPQ-tree is already constructed: A transitive edge is represented as a Q-node that is a child of a P-node.

Proof. Assume the decomposition tree has a transitive edge. As illustrated in Figure 4.5, no matter what slope is assigned to the transitive edge, no directed path graph can be drawn in parallel without crossing the edge: the other allowed slope enters the sink vertex from the opposite side it leaves the source vertex.

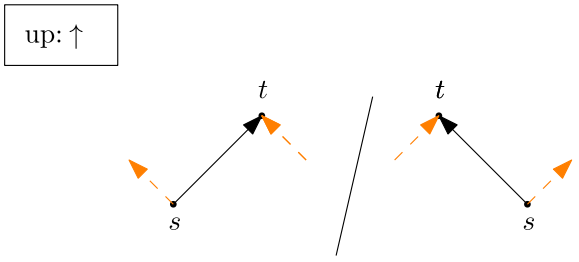

Figure 4.5: A parallel composition with a single edge cannot be drawn upward planar with two slopes.

We use induction on the SPQ-tree to prove the existence of a drawing if the conditions of the theorem are true. The cases are also illustrated in Figure 4.6, an example can be seen in Figure 4.7.

Each drawing is on the integer grid. However, we use north-east as up, so that the two allowed slopes (we use $\pi/4$ and $3/4 \cdot \pi$) are horizontal and vertical. Each up2p drawing D has a bounding box (a box that contains all vertices and edges of the drawing) spanned by the source and the sink vertex. The width and height of the bounding box is also the width $w(D)$ and height $h(D)$ of D, respectively.

Let G be a series-parallel graph with maximum in- and outdegree two that has no transitive edges. We distinguish by the root node of the decomposition tree:

S: Let G_1 and G_2 be the parallel components of G. G_1 and G_2 are subgraphs of G, thus they also have a maximum in- and outdegree of 2. Also, their decompositon trees are subtrees of the decomposition tree of G, so these also can not have a Q-node as child of a P-node (which is equivalent to a transitive edge in the corresponding series-parallel graph). By induction we get up2p drawings D_1 and D_2 of the two

graphs. Translate D_2 so that its source vertex is at the same place as the sink vertex of D_1 and merge those two vertices. We obtain a up2p drawing of G with width $w(D_1) + w(D_2)$ and height $h(D_1) + h(D_2)$.

P: The source vertex has exactly outdegree two, the sink vertex has exactly indegree two. Because the source and sink vertex of G are not directly connected by a single edge (that edge would be transitive) we get exactly two connected components, G' and G'', when removing the source and sink vertex from G. Both of them are either a single vertex or a series-parallel graph that is again satisfying the conditions of the theorem and can be drawn by induction. The single vertex can be drawn within a bounding box of height and width 0. If a bimodal embedding was given, let G' be a subgraph of the right parallel component of G. Now we translate the drawing of G'' so that the lower right corner of its bounding box is exactly one unit above and one unit left of the upper left corner of the boundingg box of G''. Then we add back the source and sink vertex and connect them to the drawings as illustrated in Figure 4.6. Altogether the new drawing has width $w(G') + w(G'') + 1$ and height $h(G') + h(G'') + 1$.

Q: G is a single edge. Draw a horizontal edge of length 1. The drawing has height 0 and width 1.

□

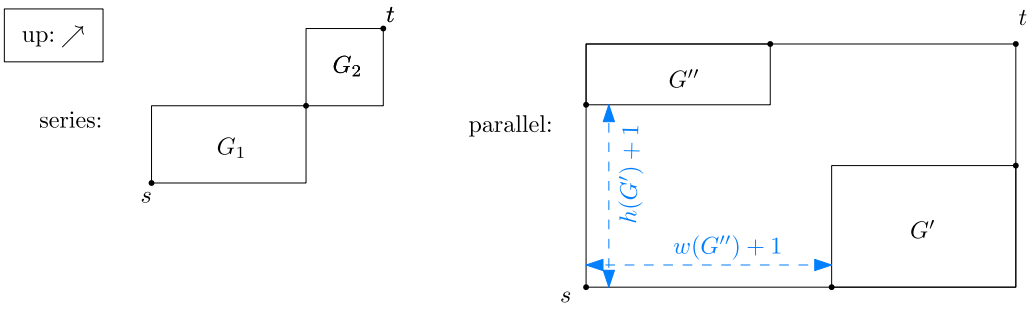

Figure 4.6: Illustration of how to combine up2p drawings according to the proof of Theorem 4.4.

4.3.3 Three Slopes

Now we want to modify the algorithm presented for two slopes. The cases Q and S essentially work the same way. For the P-node we now have a special case, in which one or two of the subgraphs must be drawn in a special way: If a graph with given embedding has for example degree three at the source vertex and consists of only two parallel components, then the component using up two edges of the source vertex must use different slopes for those, depending on where the other component is located in the bimodal embedding. If the other component is a single edge, then drawing up3p is even impossible. In this case, among others, we call the edge a "bad edge". We will later prove that any series-parallel graph with maximum in- and outdegree three that does not contain any bad edge can be drawn up3p. All bad edges are transitive edges.

Remember that if a bimodal embedding is given, the ordering of the children of a P-node corresponds to the ordering of parallel components in the graph (left child corresponds to left component in embedding).

Definition 4.5. A *bad edge* of a series-parallel digraph can be found using the decomposition tree of the digraph: If a P-node contains an S-node and a Q-node as its children, and

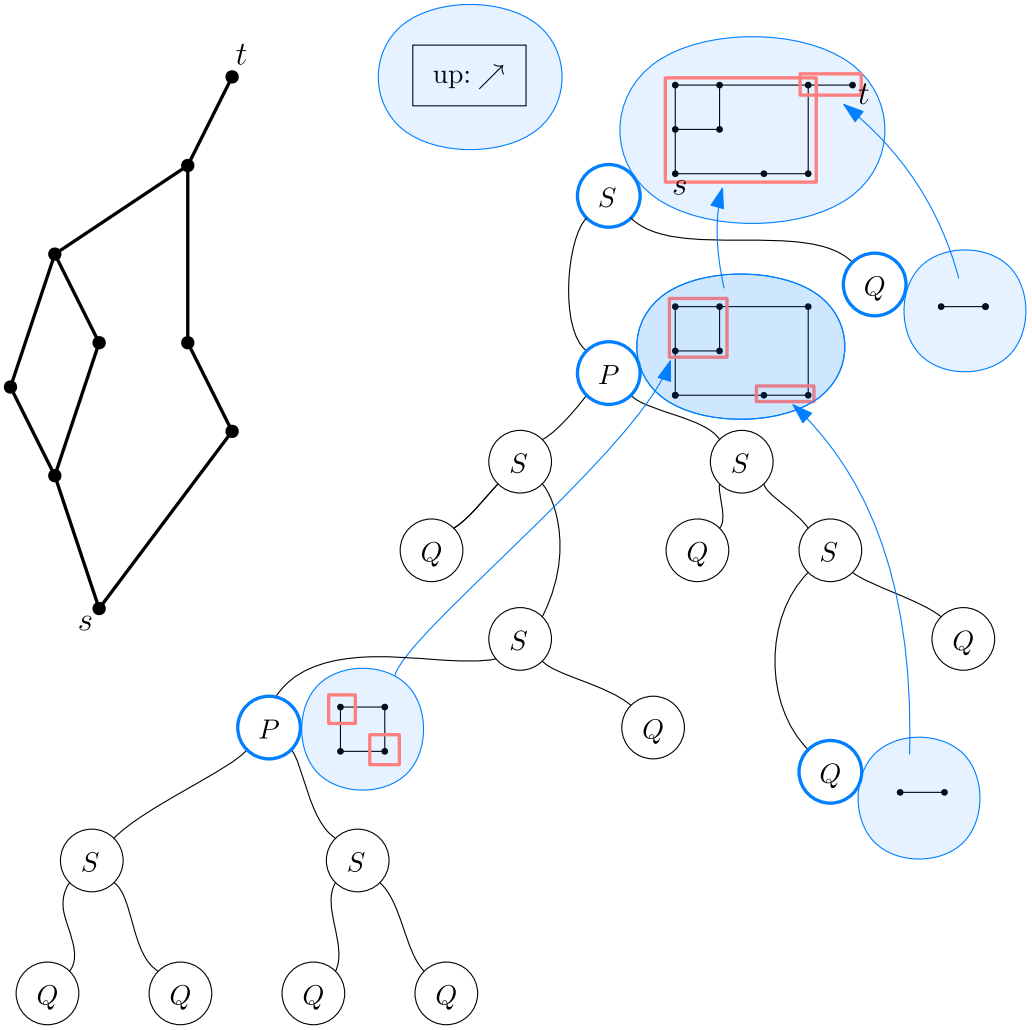

Figure 4.7: An example how to draw a series-parallel graph up2p. On the left is the original series parallel graph, in the middle there is the SPQ-tree of the graph. The up2p drawings used in the algorithm are added next to the corresponding vertices of the SPQ-tree. When combining drawings, the bounding boxes are marked red in the new drawing.

for the series-parallel graph corresponding to the S-node the source vertex or sink vertex has degree 2, then the edge corresponding to the Q-node is called a *type-1 bad edge* (see Figure 4.8).

If a bimodal embedding is given, there are additional bad edges: We call a transitive edge a *type-2 bad edge* if it is the first or last incoming edge at a vertex with indegree 3 or the first or outcoming edge at a vertex with outdegree 3.

Lemma 4.6. *In any* up3p *drawing a transitive edge must have the middle slope.*

Let D be a graph containing some transitive edge $e = (u, v)$. There exists a directed path P from u to v different from (u, v). In any up3p embedding, P must be drawn either completely to the left or to the right of e. To the left (right) of e it must use a higher (lower) slope than e for its first edge and a lower (higher) slope than e for its last edge. In either case, a higher and a lower slope than the slope of e must be used for drawing P. Because we only have three slopes, e must use the middle slope.

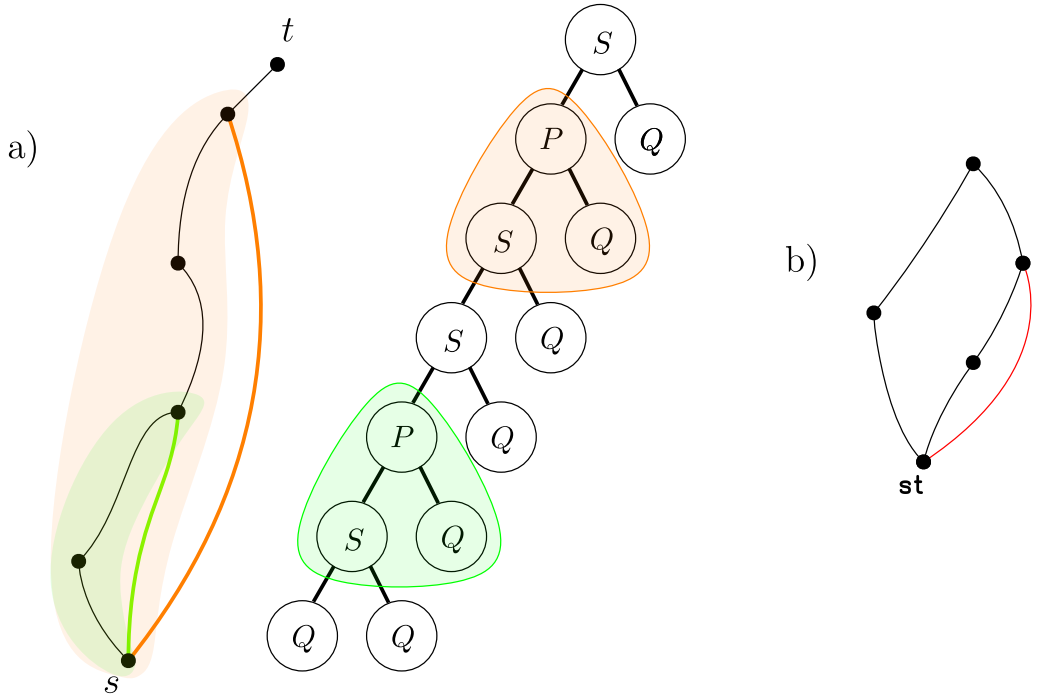

Figure 4.8: a) A SP-graph that cannot be drawn up3p because of a type-1 bad edge (marked orange). The subgraph marked by the orange area has degree two at its source vertex. Next to it is the SPQ-tree of the graph. The green edge is not a bad edge, because the source and sink vertices of the subgraph marked by the green area have degree one. b) A transitive edge that is a type-2 bad edge if the embedding is taken into account.

Theorem 4.7. *A series-parallel graph with maximum in- and outdegree 3 with or without a given bimodal embedding is drawable* up3p *if and only if it contains no bad edge.*

Proof. First we show that a series-parallel graph containing a bad edge can never be drawn up3p:

Assume some series-parallel graph G contains a type-1 bad edge $e = (u, v)$ and can be drawn up3p. The SPQ-tree of G contains the Q-node corresponding to e. Its parent is a P-node. The second child of the P-node is an S-node. Let G_S be the graph corresponding to the S-node. Because e is a transitive edge, it must use the middle slope (by Lemma 4.6).

Because G_S is a composition of two series components, there exists some middle vertex m that is the sink of its lower component and the source of its upper component. Further, P has a source-vertex (or sink-vertex) of degree 2. The only two slopes left for those edges are the lowest and the highest. But then the incident vertices are located on different sides of e. They cannot both be connected with an upward (or downward) path to m without crossing e. Contradiction.

Assume some series-parallel graph with given embedding contains a type-2 bad edge $e = (u, v)$. The edge is transitive and by Lemma 4.6 e can only be drawn using the middle slope. The two other edges starting at u (or ending at v) must use the remaining slopes. Then one of them is left and one is right of e. But a type-2 bad edge is left (or right) of both of these edges. Contradiction.

Now it is left to prove that any series-parallel graph with or without given bimodal embedding containing no bad edge with maximum in- and outdegree 3 can be drawn

up3p on the plane. We will prove the case with a given bimodal embedding below by construction.

In the case without an embedding we construct an embedding before applying the same construction. The resulting embedding has no type-2 bad edges:

Let G be a series-parallel graph without an embedding that contains no (type-1) bad edge and has maximum in- and outdegree 3. Let T be its SPQ-tree, children of P-nodes are ordered arbitrarily. Now take the bimodal embedding of G that is described by the SPQ-tree. For every type-2 bad edge e, identify the other parallel component G_P. If this component is again a parallel composition, identify its two parallel components and rearrange them so that one is left and one is right of e. In the other case G_P is a series composition. Because e is no type-1 bad edge, the source and sink vertex of G_P must have degree one. Swap e and G_P in the embedding.

This operation have no effect on other possible type-2 bad edges, because there can be no edge parallel to e (double edges are generally forbidden) and at no other place the incoming or outgoing order of edges is changed. After eliminating all type-2 bad edges, the bimodal embedding contains no more bad edges and we can apply the construction that is used for series-parallel graphs with an embedding. This construction follows down below.

Construction

We construct an up3p drawing of a series-parallel graph with maximum in- and outdegree 3 with an embedding, that does not contain any bad edges. We use rational coordinates. This makes the description of the algorithm easier, because we can scale graphs by arbitrary factors (> 0).

We again use a divide-and-conquer approach as can be seen in the example in Figure 4.13. For our construction, we use the slope set $\{\pi/4, \pi/2, 3/4 \cdot \pi\}$. Again, we use north-east as up, so that we can easily make use of bounding boxes spanned by the source and sink vertex of a series-parallel graph.

Of course, multiple ways of drawing the same SP-graph up3p can be possible. We are interested in two special cases, the left-leaning and right-leaning case. We prove that such drawings exist for every series-parallel graph that can be drawn up3p (with the given embedding).

Definition 4.8. An up3p drawing of a series-parallel graph G with source vertex s and sink vertex t is *left-leaning* (*right-leaning*) if the following conditions are satisfied (see Figure 4.9 for an example):

- If s has degree < 3 and no transitive edge starts at s then the smallest (biggest) slope is not used incident to s and
- if t has degree < 3 and no transitive edge ends at t then the biggest (smallest) slope is not used incident to t.

Let G be a series-parallel graph with a bimodal embedding that contains no (type-1 or type-2) bad egde and has maximum in- and outdegree 3. Let T be its SPQ-tree, s the source vertex and t the sink vertex. We construct a left-leaning and a right-leaning up3p drawing of G, each with the same bimodal embedding as G.

We do this construction by induction, as we did earlier in the case of up2p drawing of series-parallel graphs. A general outline of the proof structure can be seen in Figure 4.10.

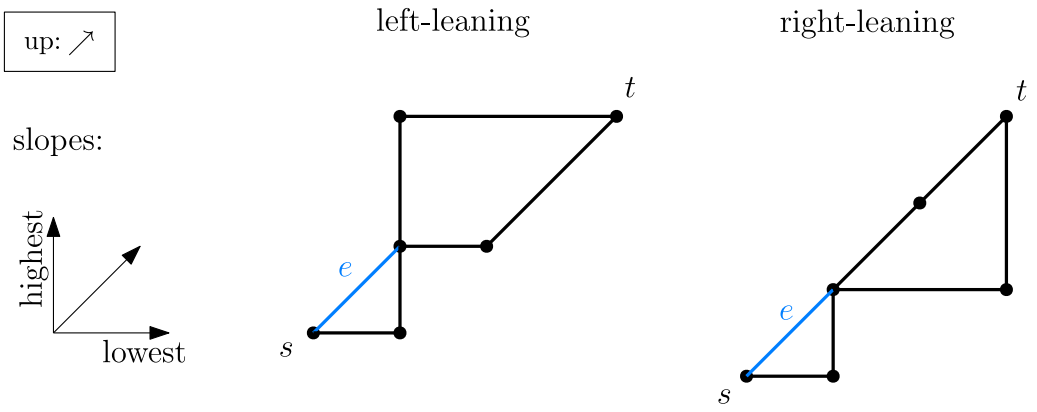

Figure 4.9: Two drawings of the same SP-graph with bimodal embedding, one left-leaning and one right-leaning. The condition at s is not satisfied because of e, so even in the left-leaning case, the lowest slope can be used.

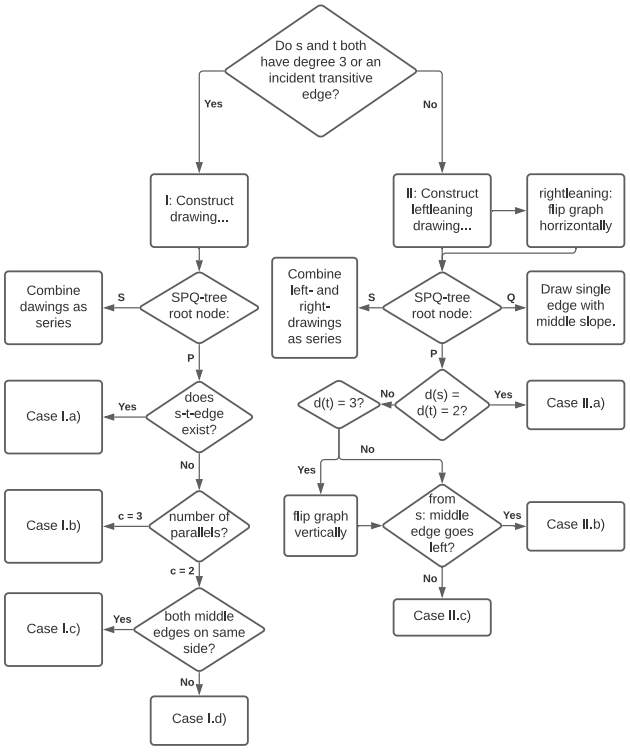

Figure 4.10: Outline of the proof structure in the construction proving Theorem 4.7.

I. First the cases where both s and t have degree three or an incident transitive edge, respectively. In this case, any up3p drawing is left- and right-leaning and we only have to construct a single drawing. We distinguish by the root node of the decomposition tree:

S: Similar to up2p drawing algorithm: Let G_1 and G_2 be the series components of G. G_1 and G_2 are subgraphs of G, thus they also have a maximum in- and outdegree of 3. Also, their decompositon trees are subtrees of the decomposition tree of G, so these also can not have a bad edge. By induction we get (both left-leaning and right-leaning) up3p drawings D_1 and D_2 of the two graphs. Translate D_2 so that

its source vertex is at the same place as the sink vertex of D_1 and merge those two vertices. We obtain an up3p drawing of G.

P: We must make a few case distinctions here:

a) s-t-edge exists: construct up3p drawings D_1 (,D_2) of the one or two connected components left when removing s and t and then join the drawings as sketched in Figure 4.11 a), ordering the subgraphs according to the given embedding. D_1 (and D_2) can be either a SP-graph or a single vertex.

Let c be the number of connected components of G after removing s and t.

b) no s-t-edge, $c = 3$: construct up3p drawings $D_i, i \in [3]$ of the three connected components left when removing s and t and join the drawings as sketched in in Figure 4.11 b), ordering the subgraphs according to the given embedding. $D_i, i \in [3]$ can be either a SP-graph or a single vertex.

If a) and b) do not apply, there is no s-t-edge and $c = 2$. s and t still must have either degree three or an incident transitive edge, respectively. A transitive edge (that cannot be an s-t-edge) would imply that it belongs to one of the two parallel components of G. But then this component has two edges connecting to s (or t) and we again get degree 3 at s (or t). Let G_l be the left and G_r be the right parallel component of G. It is important whether the middle edges incident to s and t belong to the same graph of G_l, G_r or not.

c) same graph (without loss of generality G_l): Let D_l be a left-leaning up3p drawing of G_l. It must have the shape as in Figure 4.11 c): The SPQ-tree of G_l must have an S-node as parent node, because it cannot be a Q-node (degree 2 at s) and it cannot be a P-node (then there would be three connected components and we would be in case b). Let v be the sink of the lower component of G_l. We look at s, t works the same way. The degree of s inside G_l is 2. Either no transitive edge starts at s. Then, because we use the left-leaning drawing, the smallest slope is not used at s. Or if there is such a transitive edge, the transitive edge must be to the right of the other edge, because otherwise (when taking G into account, where s has degree 3) it would be a type-2 bad edge. Then, if the transitive edge is the rightmost edge that is outgoing from s (inside G_l!), it also has the lowest slope, and according to Lemma 4.6 it can not be the smallest of the three available slopes. In either case, the smallest slope can be used for drawing the right parallel component. G_r has a source and sink vertex of degree 1. Remove them to obtain a SP-graph or single vertex, G'_r. We shrink the up3p drawing of G'_r so that it fits inside the rectangle at the bottom right of v.

d) different graph (without loss of generality s has two incident edges to G_l): Let G'_l be the SP-graph obtained from G_l by deleting the sink vertex of G_l (which has degree 1 inside of G_l) and let G'_r be the SP-graph obtained from G_r by deleting the source vertex of G_r. Now draw according to Figure 4.11 d). G'_l is drawn left-leaning, G'_r is drawn right-leaning. Both graphs must be first scaled to the same height. Using the same argument as above we can prevent any collisions.

Q: this case is not possible: A single edge is not transitive and s and t have degree 1.

II. For the other cases, at least one of s and t has degree < 3 and no incident transitive edge. We need to construct separate left-leaning and right-leaning drawings. Without loss of generality we only construct left-leaning drawings. The right-leaning drawing can be constructed by flipping the bimodal embedding horizontally (reverse the order of edges in

the bimodal embedding of each vertex), applying the same algorithm as for left-leaning and flipping the drawing back.

Again, we distinguish by the root node of the decomposition tree:

S: Let G_1 and G_2 be the series components of G. Use the left-leaning drawings of G_1 and G_2 and translate the drawing of G_2 so that its source vertex is at the same place as the sink vertex of G_1. We get a left-leaning drawing, because the edges incident to the source of G_1 and sink of G_2 and their slopes have not changed.

P: There cannot be an edge from s to t: it would be a transitive edge at both s and t.

a) If both s and t have degree 2, remove s and t from G, draw the two connected components up3p (they must be either a SP-graph or a single vertex) and connect them as shown in Figure 4.12 a). The connecting edges are making sure the drawing is left-leaning.

Assume that without loss of generality s has degree 2 and t has degree 3 (and not the other way around, which can be solved by flipping the graph vertically). Let G_l be the left and G_r be the right parallel component of G. It is important whether the middle edge at t is part of the right or left component:

b) left component: Let the SP-graph G'_l be obtained by deleting the source vertex of G_l (which has degree 1 inside of G_l) and let G'_r be obtained from G_r by deleting the source and the sink vertex of G_r. G'_r is either a SP-graph or a single vertex. For G'_l use a left-leaning drawing and combine the drawings according to Figure 4.12 b).

c) right component: Let the SP-graph G'_r be obtained from G_r by deleting the source vertex of G_r and let G'_l be obtained from G_l by deleting the source and the sink vertex of G_l. G'_l is either a SP-graph or a single vertex. For G'_r use a right-leaning drawing and combine the drawings according to Figure 4.12 c).

Note that in cases b), c) we have to adjust the sizes of G'_l or G'_r to make sure s can be placed properly and edges starting at s are not intersecting with any other edges or vertices.

Q: G is a single edge. The left- and right-leaning drawings are constructed the same way: Draw a diagonal edge going one right and one up. Note that we can use this drawing as a left-leaning and right-leaning up3p drawing at the same time.

□

Now we want to generalise this result on any series-parallel graph with maximum in- and outdegree three. As we have seen, bad edges can prevent up3p drawings with straight lines.

Definition 4.9. A *1-bend drawing* is a drawing where each edge is drawn as one or two line segments.

Accordingly, a *1-bend* up$k$p *drawing* of a graph is an upward planar drawing where edges are drawn as one or two line segments and all of those line segments have only k distinct slopes.

Lemma 4.10. *A series-parallel graph with maximum in- and outdegree three with or without a given bimodal embedding admits a 1-bend up3p drawing.*

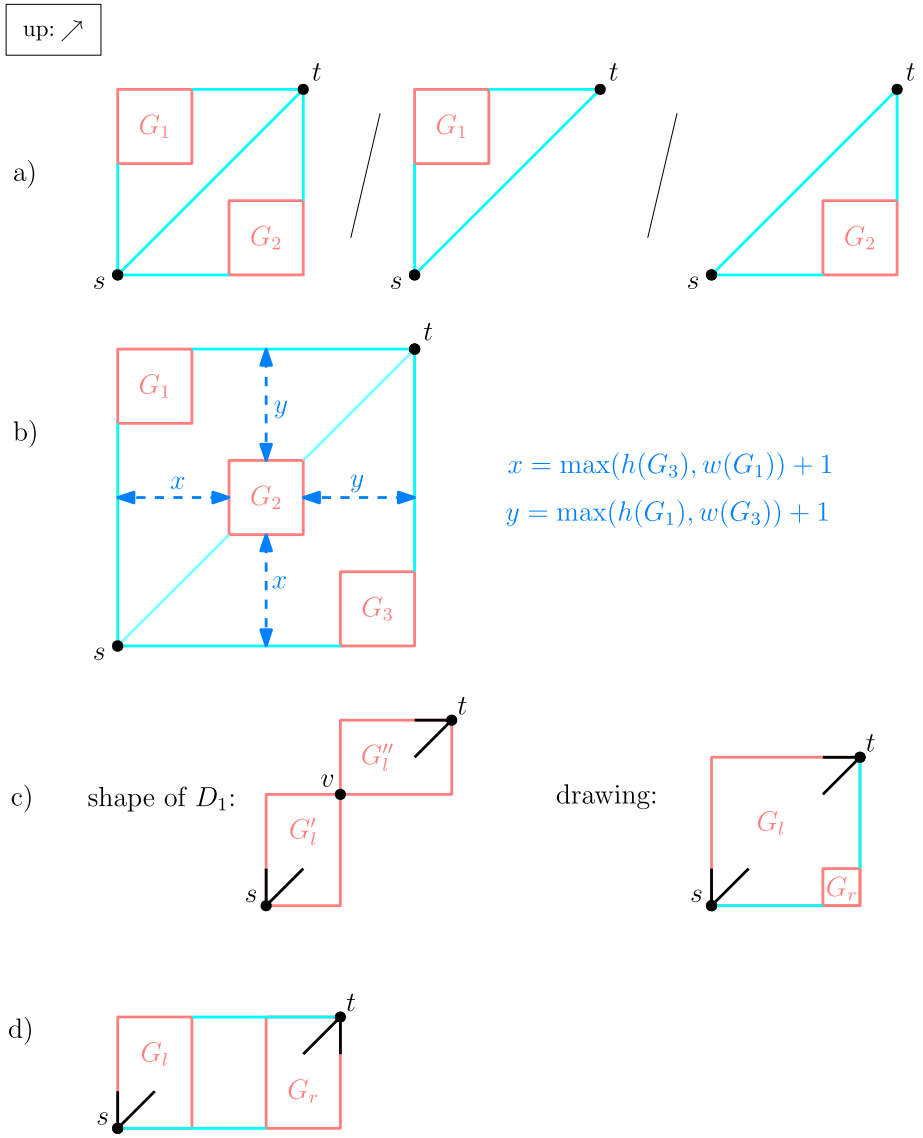

Figure 4.11: How to combine drawings to draw a parallel composition in case I. The resulting drawing is left-leaning and right-leaning at the same time.

Proof. Let G be a series-parallel graph with maximum in- and outdegree three with or without a bimodal embedding. If G comes without an embedding, choose any bimodal embedding.

Obtain G' by intersecting each transitive edge once. For an edge (u, v) this is done by removing the edge and adding a new vertex x and edges $(u, x), (x, v)$ to the graph. The new edges replace the old edge in the cw-orders of u and v.

Notice that both type-1 and type-2 bad edges are transitive edges and, by definition, G' has no transitive edges. The in- and outdegree of existing vertices stayed the same. Newly added vertices have in- and outdegree 2. Therefore, G' has no bad edges and the maximum in- and outdegree is at most 3. We apply Theorem 4.7 to obtain an up3p drawing of G'. Remove the vertices added by intersecting transitive edges. Join the incident edges to obtain edges with a bend. This concludes the proof. □

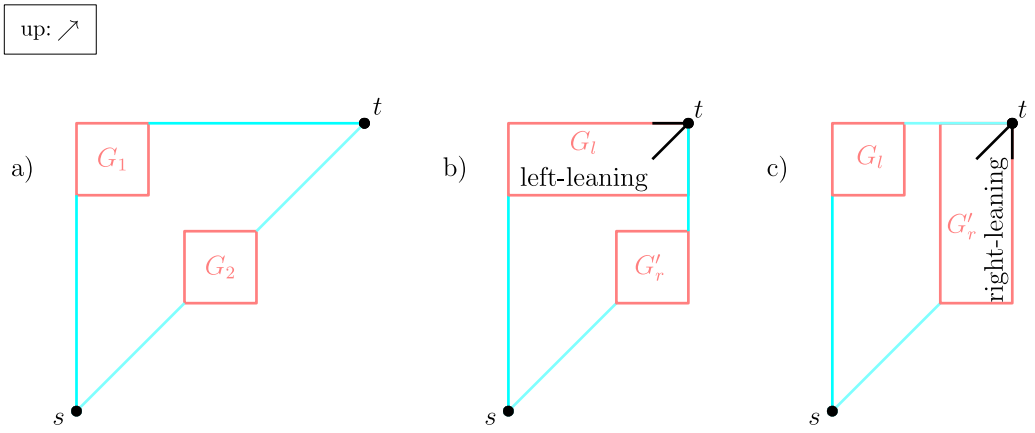

Figure 4.12: How to combine drawings to draw a parallel composition in case II. The resulting drawing is left-leaning. Right-leaning drawings can be constructed similarly.

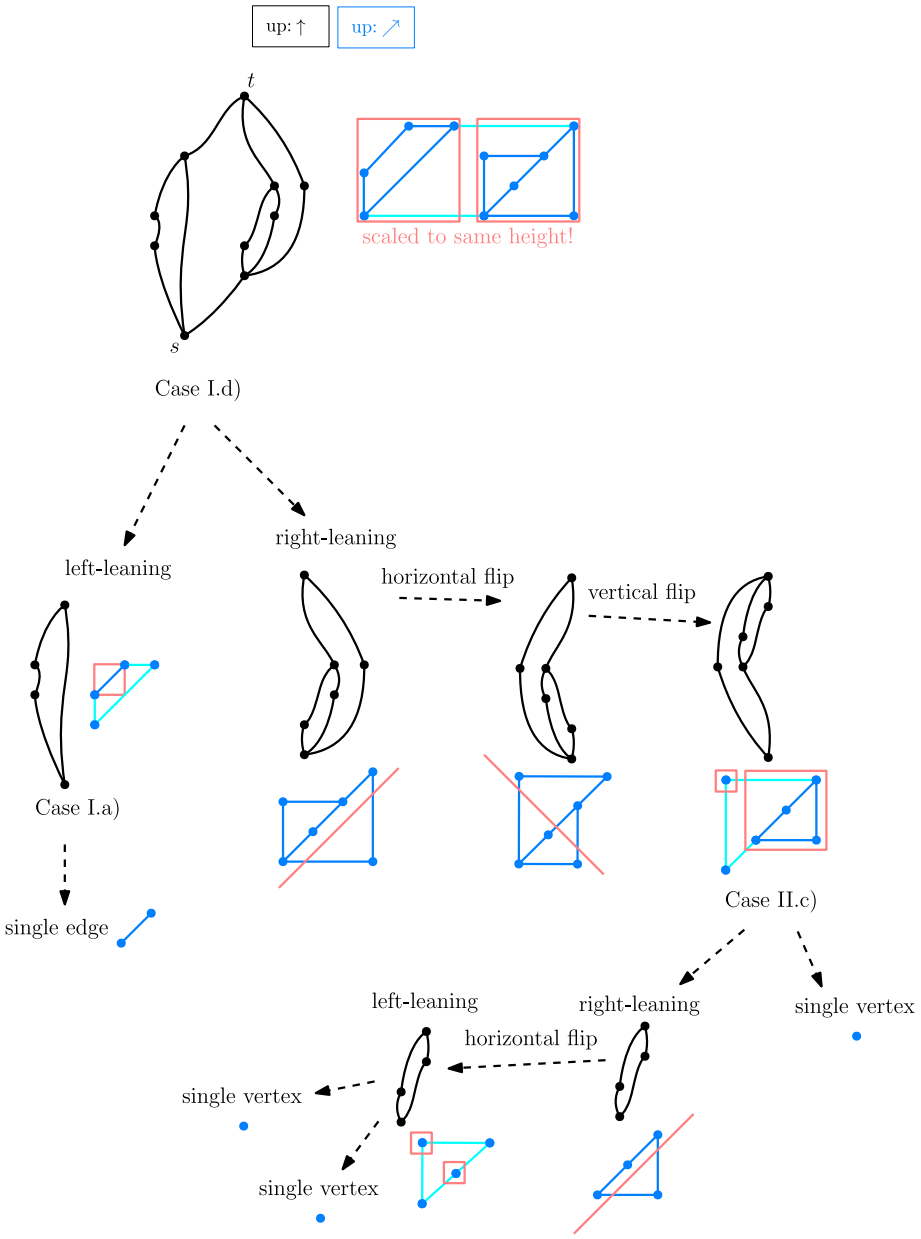

Figure 4.13: An example of drawing a series-parallel graph up3p. The graph is given with a bimobal embedding. At first, the graph is split into two parallel components. The left component has to be drawn left-leaning, the right component has to be drawn right-leaning. Because we only specified how to draw left-leaning, we can flip the right component horizontally and draw it left-leaning. Because the source vertex has degree three and the sink vertex has degree two we also have to flip it vertically. We apply case II.c). The left component is a single vertex. The right component has to be drawn right leaning again. After finishing the drawings, we flip back every time we used flipping. This is visualised by the red axis. When two drawings are combined, we visualise the drawings by red bounding boxes. The edges used to connect the components are drawn in light blue.

5. Area of up3p Drawings

We find lower bounds on the worst-case area of up3p drawings on the integer grid.

Although up2p drawings of series-parallel graphs have linear width and height, for up3p drawings we find a series of graphs with exponential width and height. For directed cacti and directed trees with fixed embedding, we also find graphs so that any up3p drawing needs exponential width and height.

5.1 Trees

[KZ21] have shown that trees can always be embedded upward three-slope planar using exponential space. Now we want to find a lower bound as well. We can actually find an exponential lower bound for trees given a bimodal embedding and for cacti without a bimodal embedding. It remains an open problem, if trees without a bimodal embedding also have an exponential worst-case scenario, although it seems unlikely.

Definition 5.1. Recall that a *slope assignment* is a function $s : E \to [0, \pi)$. For paths with a fixed slope assignment for each edge, we introduce a *shorthand notation* of only writing down the slopes (as if walking along the path). If the arrow points upwards, the edge is directed towards the second vertex, otherwise the edge is directed towards the first vertex. The vertices are labeled as v_i incrementally, starting with v_0. As seen in 5.1, the slope of the edge corresponds directly to the slope of the arrow.

Theorem 5.2. *Any planar drawing of the path*

$$P_n := (\nwarrow\nearrow\searrow\swarrow\uparrow \\ \searrow\swarrow\nwarrow\nearrow\downarrow)^n$$

with the assigned slopes on the integer grid has width and height of $\Omega(c^{|V|}), c > 1$.

Let us prove this theorem by first looking at two lemmas.

Lemma 5.3. *Let* w_n, h_n *be the minimum width and height of an* up3p *drawing of* P_n *on the integer grid where the vertex* v_{10n} *is the lowest vertex.*

Then $w_n, h_n \in \Omega(4^n)$.

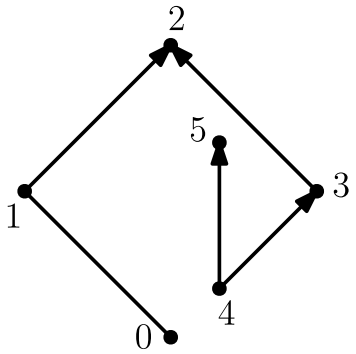

Figure 5.1: upward three-slope planar drawing of the graph denoted by $\nwarrow\nearrow\searrow\swarrow\uparrow$. Note that the direction of the arrows used in the shorthand notation is reversed when travelling an edge in the opposite direction.

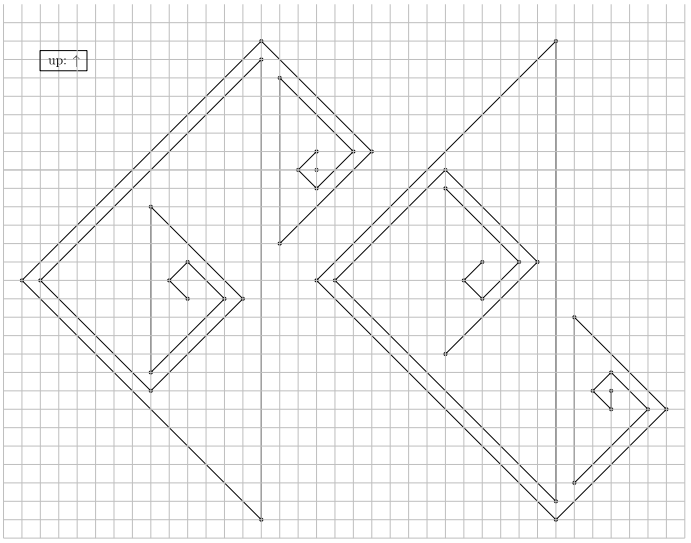

Figure 5.2: The presumably most compact up3p drawings of P_2 on the integer grid. One can see that P_n is some kind of spiral.

Proof. Consider some up3p embedding of P_n, $n \geq 2$ on the integer grid where the vertex v_{10n} is the lowest vertex. If we traverse P_n, starting at vertex v_{10n}, we always rotate counterclockwise and must spiral inward, as vertex v_{10n} is the lowest vertex. As illustrated in Figure 5.3 a), we can find $2n-1$ specific isosceles triangles (triangle 1 to triangle $2n-1$) inside of the drawing: Triangle T_i is made of the vertices $u_1 \coloneqq v_{5i-1}$ and $u_2 \coloneqq v_{5i-2}$ and the point p on the crossing of edge u_1, v_{5i} and the extension of edge u_2, v_{5i-3}. All those triangles have the same interior angles ($\pi/2$ at u_2 and $\pi/4$ at u_1 and p). The edge u_1, p is vertical, and the orientation alternates between left and right. Most importantly, T_i is always contained inside of T_{i+1}. The height and width of T_1 must be at least 2. The height and width of the last triangle $2n-1$ is smaller than the height and width of the drawing. As we can see in Figure 5.3, the height and width of triangle $i+1$ must be at least the height and width of triangle i times 2, respectively. Thus we can conclude that the height and width of this drawing of P_n must be at least 2^{2n-1}, completing the proof. □

Lemma 5.4. *Let $n \geq 2$ be even. Assume some* up3p *drawing of P_n. The "middle vertex" v_{5n} is lower than all vertices $0 \leq v < v_{5n}$ or lower than all vertices $v_{5n} < v \leq v_{10n}$ (see Figure 5.4):*

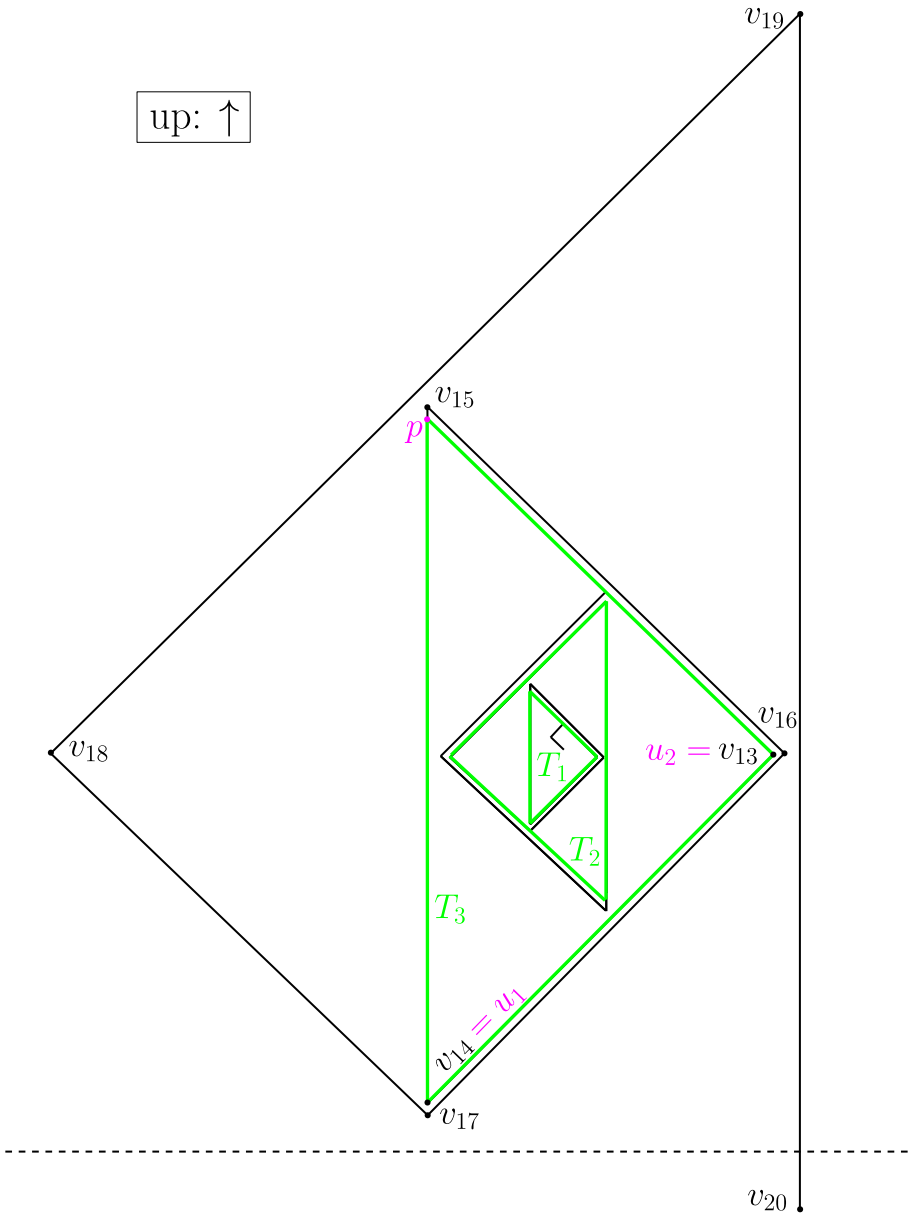

Figure 5.3: Drawing of P_2 with triangles T_1, T_2, T_3.

Proof. Let $v_w, 5n < w \leq 10n$ be some vertex that is below (or on the same level as) v_{5n} ($y(v_w) \leq y(v_{5n})$). Traversing the incremental path $P_{inc} := (v_{5n-1}, v_{5n}, v_{5n+1}, \ldots, v_w)$ we are always rotating clockwise. Thus, as long as we are left of v_{5n}, we are restricted to staying above v_{5n} as well. The only possibility to get below (or on the same level as) v_{5n} is to cross the x-coordinate $x(v_{5n})$ above v_{5n-1}. Some vertex or point on an edge of the incrementing path, p now is located at $x = x(v_{5n}), y > y(v_{5n-1})$.

We can see that $(v_{5n-1}, v_{5n}, v_{5n+1}, \ldots, p)$ is a "cage" for the decrementing path $P_{dec} := (v_{5n-1}, v_{5n-2}, \ldots, v_0)$: When we traverse this path, we always rotate counterclockwise and P_{dec} forms an inward spiral. But then, no vertex of P_{dec} can be below v_{5n}, completing the proof. □

Proof of Theorem 5.2. Let $n \geq 2$ be equal. Consider some up3p embedding of P_n with minimal height or width, respectively.

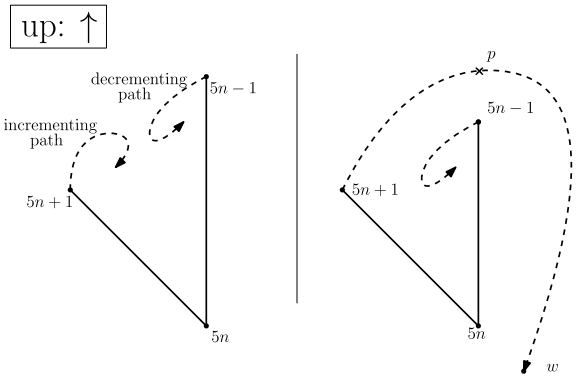

Figure 5.4: Illustration of Proof that v_{5n} is lower than all vertices of first or of second half.

- Case 1: v_{5n} is lower than all vertices $v_i, 0 \leq i < 5n$. Then we can directly apply the first lemma to the subgraph $P_{n/2}$.
- Case 2: vertex $5n$ is lower than all vertices $v_i, 5n < i \leq 10n$. Then we only look at the induced subgraph (and embedding) of vertices $v_{5n+4}, \ldots, v_{10n-6}$ (where v_{5n+4} has the lowest y-coordinate). We flip the embedding horizontally and by relabelling $(v_{10n-6} \to v_0, v_{10n-5} \to v_1, \ldots)$ we get a drawing of $P_{n/2-1}$ with corresponding slopes, and v_{5n-10} has the lowest y-coordinate.

Thus, P_n always contains a drawing of $P_{n/2-1}$ or $P_{n/2}$, where we can apply the lemma. P_n must have a minimum width and height of:

$$\Omega(4^{\frac{n}{2}}) = \Omega(4^{\frac{n}{2}} - 1) = \Omega(2^n) = [...] \approx \Omega(1.07^{|V|}) \text{ (with } |V| = 10 \cdot n + 1)$$

□

Corollary

There exist tree digraphs with fixed bimodal embeddings that need exponential space to be drawn up3p.

More formally: There exists a sequence of tree digraphs with increasing number of vertices so that any up3p drawing of such a tree $T = (V, E)$ on the integer grid has width and height of $\Omega(c^{|V|}), c > 1$.

5.2 Cacti

For cacti we want to generalise this result on digraphs without a bimodal embedding. We use additional nodes and edges to make sure the general structure of the graph is still a spiral.

Definition 5.5. We define another shorthand notation for some cactus digraphs as a chain of gadget graphs: A gadget graph (G, v) is a graph G with a home vertex h. We use the following gadgets as set $\mathbb{G}$:

$$0 \coloneqq ((\{h\}, \emptyset), h) \text{ - "no gadget"}$$
$$A, B : \text{See Figure 5.5}$$

The shorthand string $S \in L((\mathbb{G} \cdot \{\uparrow, \downarrow, \Uparrow, \Downarrow\})^* \cdot \mathbb{G})$ alternates gadget graphs and directions or "conditioned directions". Every gadget in S is a copy of the gadget, the home vertices are labelled v_i, starting with v_0. The direction or conditioned direction between the gadget with home vertex v_i and v_{i+1} is translated to further paths of the graph (see Figure 5.5):

- ↑: edge (v_i, v_{i+1})
- ↓: edge (v_{i+1}, v_i)
- ⇑: new vertex $\tilde{v}_i$, edges $(v_i, v_{i+1}), (v_i, \tilde{v}_i), (\tilde{v}_i, v_{i+1})$
- ⇓: new vertex $\tilde{v}_i$, edges $(v_{i+1}, v_i), (v_{i+1}, \tilde{v}_i), (\tilde{v}_i, v_i)$

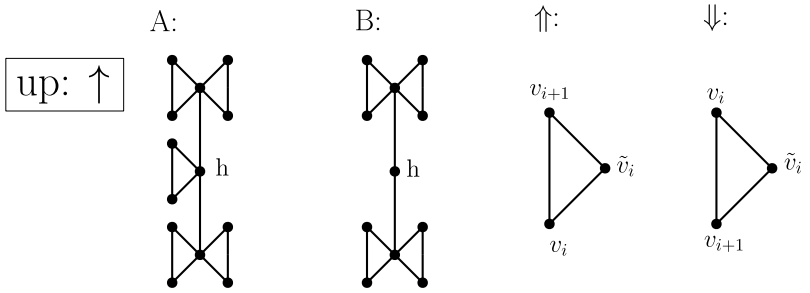

Figure 5.5: Gadget graphs that imply some constraints on new edges. A: "edges on same side", B: "edges not vertical", '⇑' and '⇓' are the symbols representing "conditioned directions". They are used to "block one side".

The graph now consists of the specified gadget graphs, that are connected by single edges or by triangles ("conditioned directions") in the specified direction.

Example

The gadgets used are already giving us many opportunities to force a specific up3p embedding. 5.6 shows a graph that cannot be drawn up3p (but could be drawn with four slopes) as an example of how to use the shorthand notation.

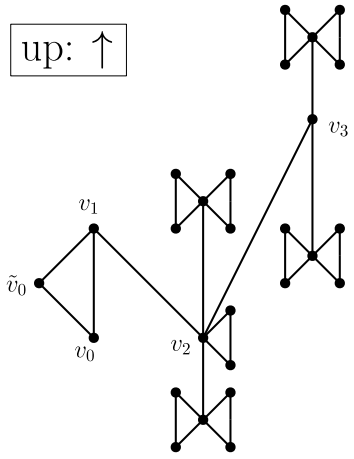

Figure 5.6: A directed cactus denoted by $0 \Uparrow 0 \downarrow A \uparrow B$ using the introduced shorthand notation, drawn upward 4-slope planar.

Theorem 5.6. *There exist cacti digraphs that need exponential space to be drawn* up3p*.*

More formally: There exists a sequence of cacti digraphs with increasing number of vertices so that any up3p *drawing of such a cactus $C = (V, E)$ on the integer grid has width and height of $\Omega(c^{|V|}), c > 1$.*

Proof. We define the sequence (C_n) of cacti digraphs:

$$C_n \coloneqq 0\ (\uparrow A \uparrow B \downarrow A \downarrow 0 \Uparrow 0 \downarrow A \downarrow B \uparrow A \uparrow 0 \Downarrow 0)^n$$

When taking a look at the complete up3p-drawing of C_1 in Figure 5.7, it must be clear that the slopes of any up3p drawing of C_n are defined when fixing the slope of the first edge ($\nwarrow$ or $\nearrow$). Without loss of generality let it be $\nwarrow$. Look at the induced drawing of the subgraph induced by the home vertices. Clearly, it has the same slopes as P_n had earlier, thus we can directly apply the result to get exponential width and height in n. On the other hand, C_n has $2n \cdot (13 + 11 + 13 + 3) + 1 \in \Theta(n)$ vertices, so the height and width of the drawing is also exponential in $|V|$. □

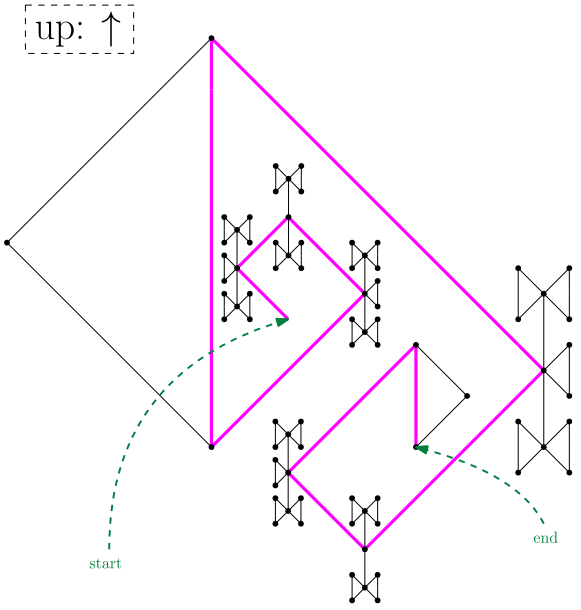

Figure 5.7: A single part of an cactus that needs exponential space for being drawn up3p on an integer grid. The induced drawing of the subgraph induced by the home vertices is marked purple.

5.3 Series-Parallel Graphs

In Chapter 4 we found an algorithm to draw SP-graphs up2p with linear width and height: One can see that from the source vertex upward and to the right we use every (integer) x-coordinate and every (integer) y-coordinate until the sink vertex at least once. This means we have $O(n)$ width and height.

For drawing up3p we again find an exponentially-sized example:

Theorem 5.7. *There exist series-parallel graphs that need exponential space to be drawn* up3p.

More formally: There exists a sequence of series-parallel graphs with increasing number of vertices so that any up3p *drawing of such a cactus $C = (V, E)$ on the integer grid has width and height of $\Omega(c^{|V|}), c > 1$.*

Proof. Look at the series-parallel graph G_n (an example can be seen in Figure 5.8): $G_1 = ([3], \{(1,2), (1,3), (1,4), (2,3), (2,4)\})$. G_{n+1} is defined by taking a copy of G_n with source vertex s and sink vertex t and adding three vertices u, v, w and the edges (u, s), (u, v), (u, w), (v, w), and (t, w).

First, G_1 has some constant size if drawn on the integer grid, at least width and height 2. Now look at an up3p drawing of G_{n+1}: It consists of a up3p drawing of G_n and one of

two options for the drawing of the three new vertices (source u, sink w and third vertex v) and new edges. There are only two possible options how they can be drawn for a fixed drawing of G_n: on the left or on the right side. In either case, the width and height at least double compared to the used drawing of G_n, as can be seen in Figure 5.8, proving the theorem. □

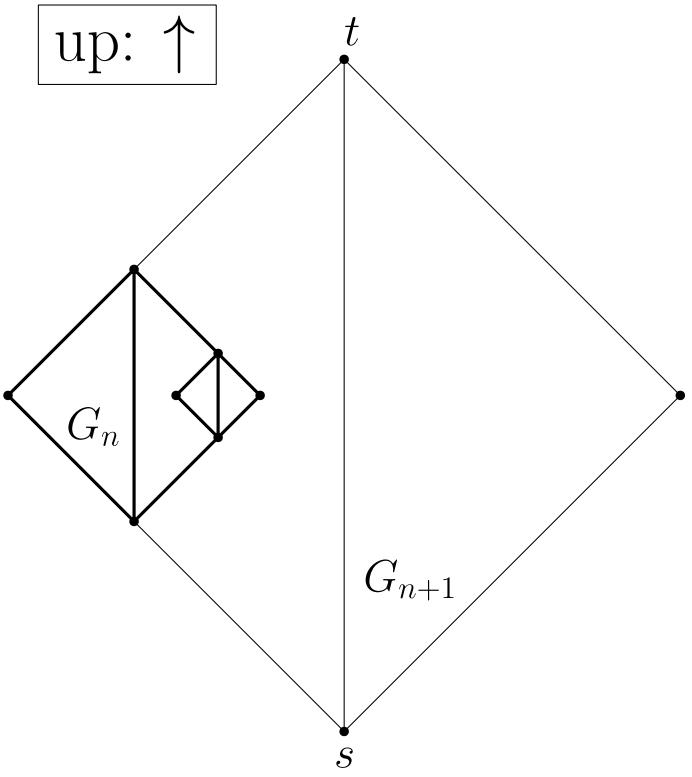

Figure 5.8: An exponentially sized series-parallel graph.

6. $\exists\mathbb{R}$-Completeness of the upward planar slope number

In this chapter we take a look at the decision problem $\Delta/2$-SlopePlanar, which is defined for planar drawing. [Hof17] proved that it is $\exists\mathbb{R}$-complete. We formulate a similar decision problem, $\Delta^{\pm}$-SlopeUpwardPlanar for upward planar drawing. At the first glance, drawing graphs with limited slopes could be easier (or harder) when the direction of edges is already restricted, because we have less choices on how to draw a graph. But nevertheless we are able to prove that $\Delta^{\pm}$-SlopeUpwardPlanar is also $\exists\mathbb{R}$-complete.

6.1 Complexity Zoo

We introduce a new complexity class, the class $\exists\mathbb{R}$. It can be thought of as an analogue of **NP**, but dealing with reals instead of booleans.

Definition 6.1. *Desicion Problem ETR*:

Given a quantifier-free formula of polynomial equations and inequalities $\phi(x_1, \ldots, x_n)$ over the alphabet $\Sigma = \{0, 1, x_1, \ldots, x_n, +, \cdot, =, \leq, <, \wedge, \vee, \neg\}$, are there real numbers x_i that satisfy ϕ? ETR is **NP**-hard and inside **PSPACE** (see for example [Mat14]).

The Complexity Class $\exists\mathbb{R}$

The complexity class $\exists\mathbb{R}$ consists of the computational decision problems that reduce to the problem ETR in polynomial time. However there is an alternative, more intuitive characterisation of $\exists\mathbb{R}$ similar to the definition of **NP**. It is using a new machine model, the Real-RAM.

Definition 6.2 (Real-RAM)**.** The *Real-Register-Address-Machine (real-RAM)* is a machine model that is able to perform algorithms on real numbers (see Figure 6.1). Its input is defined as $(a, b) \in \mathbb{R}^n \times \mathbb{Z}^m$, with unknown $n, m \in \mathbb{N}$, suitably encoded into corresponding integer and real registers before the algorithm begins. The word size used for the integer registers is some $w \in \Omega(\log(n + m))$, allowing constant-time access to the input data. The machine supports operations like addition, subtraction, multiplication, and division for reals and integers. For a complete overview of operations look at [EvdHM20, Subsection 6.1]. Every operation costs one time step. The machine can halt by accepting or rejecting.

Definition 6.3. A *real verification algorithm* for some decision problem is a real-RAM algorithm A that decides in polynomial time if some witness (also called certificate) $(a, b) \in (\mathbb{Z}^n, \mathbb{R}^m)$ is a valid solution to the given instance I encoded as an integer. a, b and I are given as input to the real RAM. For any YES-instance there must be a witness so that A accepts, for any NO-instance A must always reject.

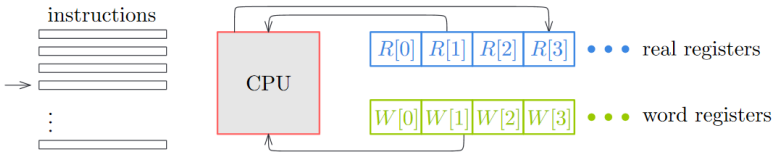

Figure 6.1: Model of the real RAM (from [EvdHM20, Section 1]).

Alternative Characterisation of ∃ℝ

[EvdHM20, Theorem 2.1] proofs that a desicion problem is in ∃ℝ if and only if there is a real-RAM verification algorithm for the problem.

6.2 Prelimilarities

Definition 6.4. A *pseudoline* is a x-monotone curve that extends infinitely in positive and negative x-direction. We use integers to label and identify pseudolines in an arrangement or intersection pattern.

Definition 6.5. A *pseudoline arrangement* is a finite set of pseudolines, so that every pair of pseudolines intersect exactly once. For a *line arrangement* we use lines instead of pseudolines. Of course, a line arrangement is also a pseudoline arrangement.

Definition 6.6. An *intersection pattern* p of some pseudoline arrangement S with pseudoline labels A is the partial order of all crossings
$\{C \subseteq A \mid \text{exactly the pseudolines } C \text{ intersect at one point}\}$ with:

$$\{a, b\,(, \dots)\} \leq \{b, c\,(, \dots)\} \iff x(\text{Crossing of } a \text{ and } b) \leq x(\text{Crossing of } b \text{ and } c)\ (a, b, c \in A)$$

All other relations are induced by transitivity. We also say that S *realises* p. An example is visualised in Figure 6.2.

Definition 6.7. The *arrangement-graph* and *arrangement-graph-drawing* of a pseudoline arrangement is defined as follows (for the graph just dismiss the mapping onto the plane):

Place a vertex on each intersection and an edge pointing towards positive x on each section of a pseudoline. The vertex is also referred to as a *crossing*, the edge is also referred to as a part of the corresponding pseudoline. For each pseudoline i truncate the two infinitely long sections to sections starting at the intersection and ending at some (very short) distance. Place a vertex start_i at the new ending with lower x-coordinate and end_i at the new ending with higher x-coordinate.

Definition 6.8. *Desicion Problem* STRETCHABILITY:

Given an intersection pattern that can be realised by some pseudoline arrangement, is there a line arrangement with the same intersection pattern as the pseudoline arrangement?

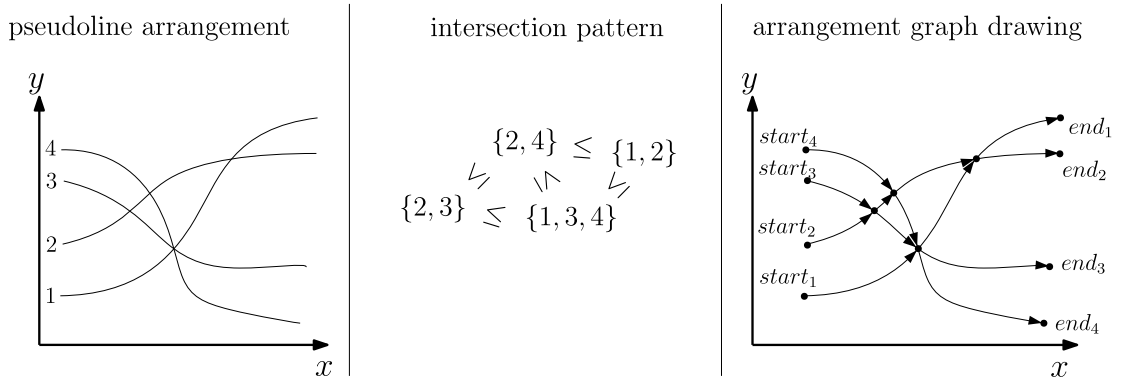

Figure 6.2: A pseudoline arrangement, its intersection pattern and its arrangement graph drawing.

Alternative Formulation

The problem Stretchability can be expressed in several different ways. Before showing that all formulations refer to the same problem, we treat them as different problems.

Definition 6.9. *Desicion Problem Stretchability*:*

Given an intersection pattern that can be realised by some pseudoline arrangement, is there a homeomorphism of the plane that maps the curves onto lines?

In this formulation, the "direction" of the lines remains open, where as in the first formulation the "direction" of the lines must be the same as in the pseudoline arrangement (crossings of the intersection pattern are ordered from negative to positive x).

After showing the equivalence, we will take a look at problems related to (upward) planar drawings:

Definition 6.10. *Desicion Problem* $\Delta/2$*-SlopePlanar:*

For some undirected graph G, is it possible to draw that graph planar using straight line segments as edges and only $\Delta/2$ slopes (where Δ is the maximum degree of G)?

Definition 6.11. Desicion Problem $\Delta^{\pm}$-SlopeUpwardPlanar:

For some directed graph G, is it possible to draw that graph planar using straight line segments as edges and only $\Delta^{\pm}$ slopes (where $\Delta^{\pm} := \max(\max \text{indegree}, \max \text{outdegree})$)?

6.3 Idea

Now we can state the main theorem of this chapter:

Theorem 6.12. $\Delta^{\pm}$*-SlopeUpwardPlanar is* $\exists\mathbb{R}$*-complete.*

We know Stretchability is $\exists\mathbb{R}$-complete (see [Mat14, Theorem 4.1]). First we show that Stretchability* is the same problem as Stretchability phrased in a different way. We only do this for clarification, but we use the lemmas we develop later. [Hof17] uses a formulation similar to Stretchability* in his proof of $\exists\mathbb{R}$-hardness of $\Delta/2$-SlopePlanar. However, he uses the pseudoline arrangement as input of the problem and does not clarify further how the pseudoline arrangement is represented.

We then want to prove $\exists\mathbb{R}$-completeness for $\Delta^{\pm}$-SlopeUpwardPlanar. To prove $\exists\mathbb{R}$-membership, we use a verification algorithm that could be run on the Real-RAM. $\exists\mathbb{R}$-hardness of $\Delta/2$-SlopePlanar was proven by [Hof17]. We will follow this proof strategy to show $\exists\mathbb{R}$-hardness of $\Delta^{\pm}$-SlopeUpwardPlanar, but we reduce from Stretchability.

6.4 Equivalence of Stretchability and Stretchability*

Definition 6.13. The *ascending (or descending) gradient-ordering* at a crossing c of a pseudoline arrangement is obtained by taking all pseudolines that go through c and ordering them by the gradient they have at that crossing in ascending (or descending) order.

In the following lemma we show that the pseudoline arrangement does not play a major role for the gradient-ordering. It is fixed by the intersection pattern.

Lemma 6.14. *For any intersection pattern with more than one crossing and some fixed crossing c we can find the ascending or descending gradient-ordering of the pseudolines going through c in any realizing pseudoline arrangement at c. This can be done in polynomial time.*

Proof. Let P be the intersection pattern with n pseudolines and $L = \{l_1, \dots, l_k\}$ be the pseudolines involved in the given crossing c. As illustrated in Figure 6.3, there must exist a pseudoline l not involved (otherwise, all pseudolines would cross at the same point), crossing m of the pseudolines before c and $k - m$ pseudolines after c. Because all pseudolines cross exactly once pairwise, the crossings $c_i \coloneqq cross(l, l_i)$ between the pseudolines L and l are pairwise at different locations.

All crossings involving l are ordered totally by the intersection pattern P and all crossings c_i must also occur before or after c in p (they share the pseudoline l_i). Therefore, P orders $c_i (i \in [k])$ and c in a total order: $O = c_{\sigma(1)} < \dots < c_{\sigma(m)} < c < c_{\sigma(m+1)} < \dots c_{\sigma(k)}$ ($\sigma : [k] \to [k]$ is some permutation of $[k]$). Now we claim that the order of descending (or ascending, we do not know which one) gradients must be $\sigma(m+1) < \dots < \sigma(n) < \sigma(1) < \dots < \sigma(m)$. Whether the order we found is ascending or descending depends on the position of l relative to c: If l passes below c, we ordered the gradients ascending (as in Figure 6.3), and descending, otherwise. Because we only got the intersection pattern, we cannot tell, which one it is.

To prove the claim, assume some two pseudolines that both cross l after c occur in the wrong order. Then, as sketched in Figure 6.3, those must cross a second time between c and their crossings with l. Contradiction.

The same argument holds if both pseudolines cross l before c. If one crosses l before c and one c before l, the gradient-order of the two pseudolines is obviously fixed: The pseudoline that crosses l before c has a higher gradient at c than the other one if and only if l passes below c).

To wrap it up, we found that $\sigma(m+1) < \dots < \sigma(n) < \sigma(1) < \dots < \sigma(m)$ is the gradient-order we searched for.

The algorithm to get the gradient-ordering at one crossing for an intersection pattern of n lines uses only quadratic time: Finding a pseudoline label not belonging to c can be done in linear time. The order in which the pseudolines pass the corresponding pseudoline can also be found in $\mathcal{O}(n \log n)$ (we need to sort the labels). The rearrangement at the end can be done in linear time. □

Definition 6.15. For a partial order P a *linear extension* Q is a total order so that all relations given in P are preserved ($x \leq_P y \Rightarrow x \leq_Q y$).

The *start-order* of pseudolines in some pseudoline arrangement is the order of the pseudolines when sorting them with ascending y-coordinate at some x-coordinate to the left of all crossings. Two pseudolines are *neighbors* in some pseudoline arrangement if they occur direcly after each other in the start-order.

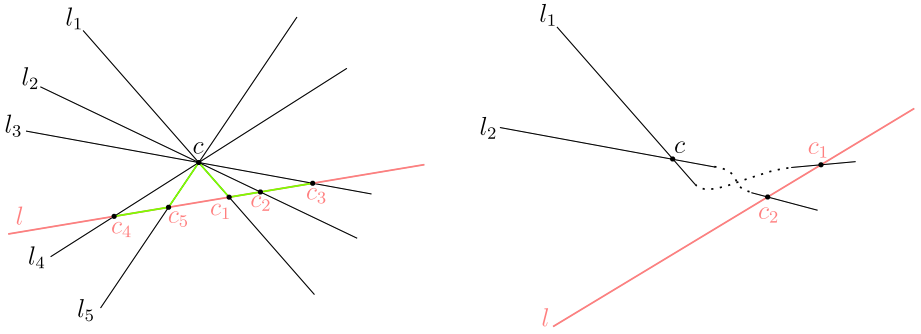

Figure 6.3: Left: A single crossing c of a pseudoline arrangement and a line l not involved. The Order (corresponding edges highlighted in green) of crossings of l and the involved pseudolines tells us the order of gradients of the involved pseudolines at c: $l_1 < l_2 < l_3 < l_4 < l_5$. Right: two hypothetic pseudolines where the gradients at the crossing do not correspond to the order of crossings with l. They must cross a second time, which is not allowed.

Note

In any pseudoline-arrangement there are two pseudolines having only one neighbor and all other pseudolines have exactly two neighbors.

Lemma 6.16. *For any intersection pattern with more than one intersection, the start-order is fixed (except for reversing). It can be calculated in polynomial time when the intersection pattern is given.*

Again, In this lemma we do not care if we find the exact order or the reversed order.

Proof of Lemma 6.16. Let P be some intersection pattern with n labels $\{1, 2, \ldots, n\}$. We claim that in any pseudoline arrangement A realizing P the neighbors of any pseudoline are fixed. This will be proven in the following two paragraphs. If the neighbors of each pseudoline are fixed, we can select one of the two pseudolines occuring first or last in the order of pseudolines (exactly the two pseudolines with only one neighbor). Then we can always add the (other) neighbor of the newest added pseudoline as new maximum of the order until we arrive at the last pseudoline (that again has only one neighbor).

Construction

The mechanism used to identifying those neighbors can be understood as taking any realising pseudoline arrangement and labelling the "layers" of the arrangement, see Figure 6.4. Then we can look which layers are touching. For the proof however, we do not use any pseudoline arrangement to underline that the construction is completely independent of the order of pseudolines. Fix some linear extension $L = c_1 < c_2 < \cdots < c_n$ of the crossings in the intersection pattern. In increasing order, starting at c_1, for each crossing c_i calculate the gradient-ordering $l_1, l_2, \ldots, l_k$ of the involved k pseudolines at the crossing as described in the previous lemma. Now for all crossings c to the right of c_i ($c_i <_L c$) substitute l_1 with l_k, l_2 with l_{k-1} and so on. At the same time note that lines l_i and l_{i+1}, $i \in [k-1]$ must be neighbors.

Correctness

Take a look at any pseudoline arrangement realizing the given intersection pattern P. At each x-coordinate, re-label the pseudolines with *layer labels*: The pseudoline with the lowest y-coordinate gets the label 1, and we continue ascending with ascending y-coordinate.

The layer labelling is equivalent to the renaming process in the construction above: The layer labelling can be achieved by reversing the order of labels of pseudolines of each crossing to the right of it, going from left to right, as illustrated in Figure 6.4. Every such step is equivalent to the relabelling process in the construction above: We use the order of gradients in ascending or descending order in both cases.

Further, the layer labels incident to the crossings make clear which pseudolines must be neighbors: Any two neighbors $i, i+1$ in the arrangement have a crossing with incident layer labels $i, i+1$, at which no other pseudoline with intermediate gradient intersects. Thus, i and $i+1$ are also identified as neighbors in the algorithm.

If two pseudolines are identified as neighbors in the algorithm, the responsible crossing is also the witness that they really must be neighbors, because we can follow the corresponding layers to the left. If they were no neighbors there must be an intermediate layer and the corresponding label would always be in between them in the gradient-ordering at the crossing. Therefore the algorithm would not mark them as neighbors.

The algorithm runs in polynomial time: We first need to calculate the gradient-orderings for all ($\leq n^2$) crossings, each in polynomial time. Then we need to go through all crossings and for all crossings right of each crossing (we can for example perform a depth-first-search on the graph representing the intersection pattern, if it is given in that form) switch labels. This can be done in $\mathcal{O}(n^4)$ (depth-first-search on $\leq n^2$ nodes, repeated $\leq n^2$ times) in total. At the end we only have to go through all crossings again and for each crossing find neighboring labels, which can be done in at most $\mathcal{O}(n^3)$ ($\leq n^2$ crossings with $\leq n$ already sorted labels each). □

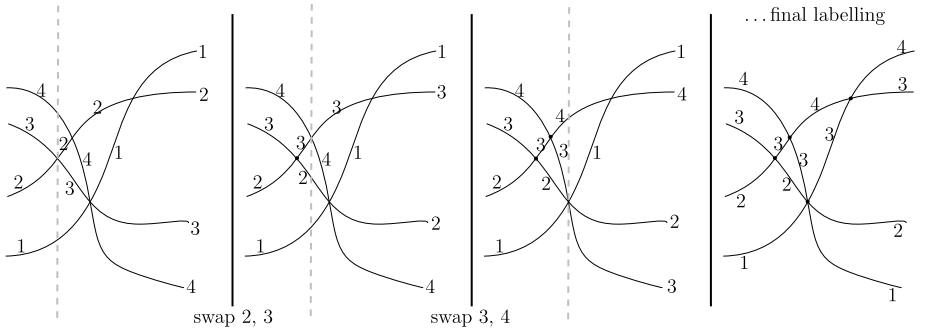

Figure 6.4: The process of going from pseudoline labels to layer labels.

Theorem 6.17. *The problems* Stretchability *and* Stretchability* *are equivalent.*

Proof. We ignore the trivial case of only one intersection. Instances with only one intersection are always YES-instances.

⇒ If we have a YES-instance of Stretchability, then for the intersection pattern P and any realising pseudoline arrangement A, there also is a line arrangement L with the same intersection pattern. By the last lemma we know that the order of the pseudolines in A and in L is either the same or exactly reversed. Here, the order of the pseudolines is defined by comparing y-coordinates left of all crossings. If the order is exactly reverse, flip A vertically to obtain A'. Now the two pseudoline arrangements are clearly homeomorph: Because they have the same intersection pattern and the peudolines are exactly in the same order (left of all crossings), the corresponding arrangement-graph-drawings of A' and L must have the same combinatorial embedding (with same, fixed

outer face). Thus, there exists a homeomorphism between the arrangement-graph drawings.

The same homeomorphism also maps A' onto L (except left and right of all crossings, where finding an homeomorphism from peudolines to straight lines is trivial). Thus, the pseudoline and line arrangement are homeomorph and we have a YES-instance of StretchAbility*.

$\Leftarrow$ If we have a YES-instance of StretchAbility*, there is a homeomorphism of the plane mapping the curves of some pseudoline arrangement realising the given intersection pattern onto lines. We fix those lines and prove that we can use them to show that the given intersection pattern has a valid line arrangement.

For each of the resulting lines there is one direction we can traverse the line so that the intersections occur in the same order as originally. We refer to this as the direction of the line. Yet, this direction does not have to point to positive x (in the sense that when travelling in that direction the x-coordinate always increases). We prove that if we fix the directed lines, we can rotate the plane so that when travelling along a directed line, the x-coordinate always increases. Then the resulting (rotated) lines form a valid line arrangement of the given intersection pattern.

Assume we cannot rotate the palne so taht all directed lines point towards positive x. We contradict this case by finding a circular dependency (three intersection points that induce a cycle on the arrangement-graph, although the arrangement-graph must be acyclic). As visualised in Figure 6.5, there exist three lines, so that for example two point towards positive x and one points towards negative x. In case 1 these three lines intersect in three different points and these intersection points are a circular dependency. In the other case, they intersect in a single point. Then (because at the beginning we said that there should be at least two intersections) there must exist another line. Where ever this line is pointing (case 2a: positive x, case 2b: negative x), in any case we get some circular dependency, as sketched in 6.5. The cases where the fourth line intersects the third line at lower x-coordinate than the intersection of lines 1, 2 and 3 are equivalent by applying point reflection. The cases where the fourth line intersects the third line in the other direction that illustrated are equivalent by flipping along the y-axis and swapping lines 1 and 2.

□

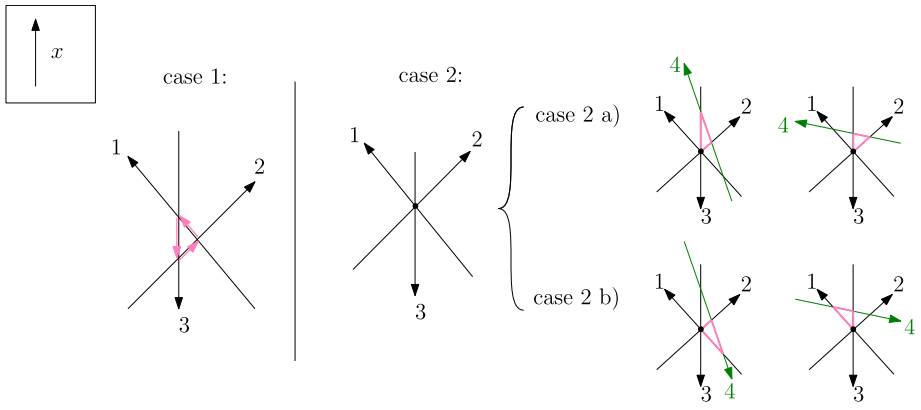

Figure 6.5: A visualisation that any line arrangement can be rotated so that all lines point towards the positive x.

6.5 ∃ℝ-Membership

We prove that $\Delta^{\pm}$-SlopeUpwardPlanar is in ∃ℝ:

We provide an algorithm for the Real-RAM. The input encodes the Graph $G = (V, E), |V| = n$ as an integer vector (incidence list). The certificate will be interpreted as $2n$ reals for the position of the vertices.

Algorithm 6.1: Verification for $\Delta^{\pm}$-SlopeUpwardPlanar

```
verify((V, E), certificate x, y) //x, y : V ⟶ ℝ
for (u, v) ∈ E do
    if x(u) ≥ x(v) then
        REJECT //edge not upward
    end
end
d_max ← Δ±(G) //calculate allowed number of slopes
d ← numSlopes(G) //the number of slopes used for the edges (calculate the slope
  of each edge and count number of unique slopes)
if d > d_max then
    REJECT
end
for e1, e2 ∈ E, e1 ≠ e2 do
    if intersect(e1, e2) then
        REJECT //not plane
    end
end
ACCEPT
```

If a digraph is a YES-instance, this induces that there exist coordinates, so that the algorithm accepts those as certificate. On the other hand if the algorithm accepts, the input certificate is also a witness that the graph can be drawn according to the given restrictions (it was checked that the edges are pointing upward, use only $\Delta^{\pm}$ slopes and do not intersect). The algorithm clearly runs in polynomial time in n.

6.6 ∃ℝ-Hardness

Now we want to reduce from Stretchability to $\Delta^{\pm}$-SlopeUpwardPlanar. As stated earlier, the reduction follows the proof from [Hof17]. However, we have to make some adjustments, because in our case the produced graph needs to be directed and also needs to be drawable upward planar.

Transformation

Let some intersection pattern of an unknown, valid pseudoline arrangement A of n pseudolines be given. Again, we exclude the trivial case of only one crossing (it is always a YES-instance). First we rename the pseudolines: Using Lemma 6.16, we label the pseudolines from 1 to n in the start-order determined by the intersection pattern.

We will now construct a digraph by modifying the arrangement-graph corresponding to the intersection pattern. To make the construction better understandable, we also use an arrangement-graph-drawing D of A and assume that the pseudolines were labeled. However, the whole transformation can be computed without constructing any drawing. Instead, one

would use a bimodal embedding that is using the start-order of corresponding pseudolines to determine the cw-order of the edges: The edge with pseudoline, that appears first in the start-order, also is the first incoming and outgoing edge and so on.

Rotate D by $\pi/2$ counterclockwise. Now all edges in D are y-monotone (upward). Let V_c be the vertices that are not a start- or end-vertex (i.e. the *crossings*). They can be partitioned into $V_c = V_{tc} \cup V_{sc} \cup V_{ec}$:

- The crossings V_{tc} where there is at least one outgoing and one incoming edge where the adjacent vertex is a crossing (*transit crossings*),
- the crossings V_{sc} where there is no incoming edge connecting to another crossing (*start crossing*), and
- the crossings V_{ec} where there is no outgoing edge connecting to another crossing (*end crossing*).

Ensure In- and Outdegree $2n$ at All Crossings

For each vertex v in V_c and each pseudoline not already passing through v, place a *pseudoline stub* (two vertices and two short edges) so that when starting at the incoming edge belonging to pseudoline 1 at v and rotating clockwise around v, we first come across all incoming edges, belonging to pseudolines $1, 2, \ldots, n$ in that order. After that, we come across all outgoing edges in the same order. Now v has indegree and outdegree n. After every incoming or outgoing edge in cw-order add another short incoming or outgoing edge and vertex (*in-between stub*), respectively. The last incoming and outgoing edge is called the *horizontal stub*. For an example see Figure 6.6.

Notice that if we only look at v and the $4n$ incident edges, fix the clockwise order of the edges and try to draw them with $2n$ slopes, then each pair of edges belonging to the same pseudoline or pseudoline stub (or in general the ith incoming and ith outgoing edge in clockwise order) must have the same slope.

After we have added all these vertices and edges for every crossing, assure yourself that we still have a planar drawing D' (See 6.7).

Fixing the Order at Transit Crossings

At the end we want to output a graph, not a drawing or embedding. Therefore we need to fix the order of edges at each transit crossing. We will do this by connecting the vertices of degree 1 inside each inner face with a cycle c so that the drawing stays planar. We also need to add some edges in a similar way to the outer face. Because we deal with a digraph and want to make it upward drawable we also need to specify the orientation of those newly added edges.

We do this by labelling path-source (s) and path-sink (t) vertices and building paths from path-source to path-sink vertices. An example result of this procedure be seen in Figure 6.7.

Outer Face

First label path-sources and path-sinks on the outer face: The vertices of degree 1 of horizontal stubs of start-crossings are labelled with s. The vertices of horizontal stubs of end-crossings are labelled with t. Crossings on the boundary of the outer face with no adjacent start- or end-vertex occur either in between the end-crossings or in between the start-crossings when traversing the boundary of the outer face. For each such crossing v pick any adjacent vertex w with degree 1 that is lying on the outer face. If w has indegree one, label it s, otherwise label it t. For an example look at the highest of the vertices labelled s in Figure 6.7.

Inner Face

For every inner face f we label one path-source and one path-sink:

Because pseudolines are x-monotone, and D' is using the same layout, each inner face of D' can have only one source and one sink. Identify the only crossing on the boundary of f that has two incoming edges that are also on the boundary of f (the sink of the face). Pick any adjacent vertex of degree 1 on the boundary of f and label it t. Equivalently, identify the only crossing on the boundary of f that has two outgoing edges that are also on the boundary of f (the source of the face) and pick any adjacent vertex of degree 1 on the boundary of f. Label it s.

Connecting the Vertices of Degree One

Now we connect the path-sources to the path-sinks. For every vertex v_1 labelled s, traverse the boundary of the (only) face he lies once in clockwise and once in counterclockwise direction, both times until hitting a vertex v_2 labelled t or s. Build a new directed path from v_1 to v_2 along all vertices of degree 1 that were found while traversing the boundary until connecting to t. If another vertex labelled s occurs before t, dismiss that specific case do not build such a path. This occurs next to start- and end-crossings on the outer face when traversing in the wrong direction.

Modification to Inner Faces: Adjustment Vertex

For guaranteeing that the added edges can be drawn properly, even if we only use limited slopes, we need to add a small modification: For each inner face f of D' (without the newly added paths) again identify the source of f. From the adjacent vertices of degree 1 on the boundary of f pick the first vertex v in counterclockwise order (the rightmost). Subdivide its rightmost outgoing edge (v, w) (if it was picked as path-source it has another outgoing edge). This means, a new vertex a is added and the edge (v, w) is replaced with the edges $(v, a), (a, w)$.

The newly added vertex is also called adjustment vertex a. We call the modified Graph D''.

Finally, the output of our transformation is the graph D'' (without any embedding). Without further proof we state that the whole construction could also be done completely in a purely combinatorial way.

proof of Theorem 6.12. In the previous subsection we have already shown that $\Delta^{\pm}$-SlopeUpwardPlanar is in $\exists\mathbb{R}$. Now we complete the proof of Theorem 6.12 by proving that the transformation presented in this subsection reduces Stretchability to $\Delta^{\pm}$-SlopeUpwardPlanar.

The given transformation needs only polynomial time (if done the combinatorial way). We proof for an instance I of Stretchability, consisting of n pseudolines (or, more specific, the intersection pattern of n pseudolines) and the transformation result $f(I)$, an instance of $\Delta^{\pm}$-SlopeUpwardPlanar: If and only if I is a YES-instance of Stretchability then $f(I)$ is also a YES-instance.

- $\Rightarrow$ Assume the pseudoline arrangement I is stretchable. Then take the resulting line arrangement L. From L construct the drawing D'' as in the transformation. Note that D'' is also a drawing of $f(I)$. The edges that are part of lines already use only n different slopes, but we didn't specify how the other edges should be drawn. We are allowed to use $2n$ slopes. The pseudoline stubs get the slopes of the corresponding pseudoline. The in-between stubs get slopes so that they are bisecting the angle. The

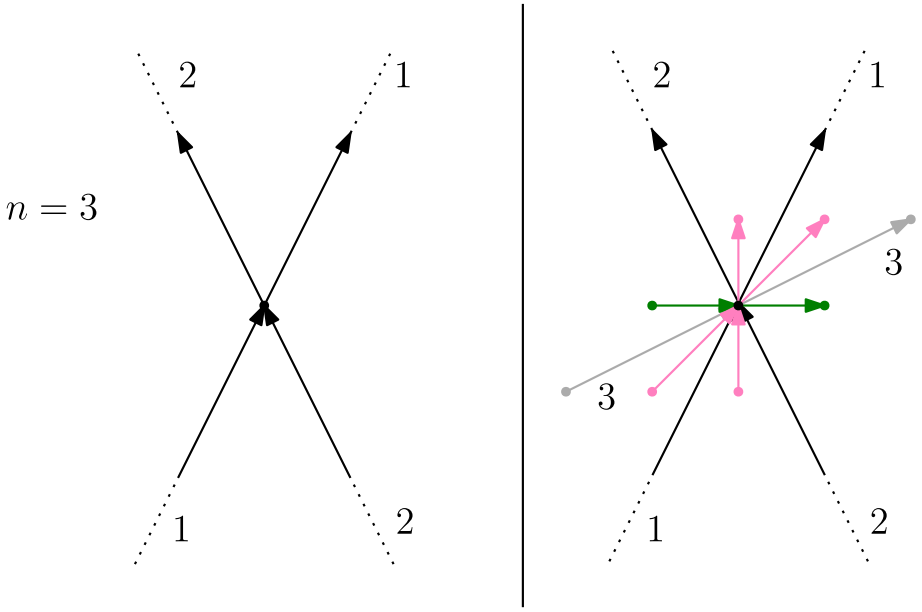

Figure 6.6: All edges and vertices that are added to a crossing - gray: pseudoline stub, pink: in-between stubs, green: horizontal stub.

horizontal stubs are drawn exactly horizontal. This is okay as the drawing could be rotated a very small degree counterclockwise so that the horizontal stubs point upward. We now used all $2n$ slopes.

The only remaining task is to draw the paths we added from vertices marked s to vertices marked t. We draw them in parallel to the lines they are next to, starting at s. For the outer face, we end at t without any problems. The length of the incident edges can be adjusted. For an inner face, before we get to the adjustment vertex, we calculate its position using the distance to the line we are currently parallel to and the two possible slopes for the last edge: horizontal or the same slope as the edge between path-source and the adjacent crossing (See Figure 6.8). Now we have completed the drawing, thus we have a YES-instance of $\Delta^{\pm}$-SlopeUpwardPlanar.

Note that all problems of wrongly intersecting edges occuring during the drawing of stubs and paths can be solved by starting with a lower distance between path-source and adjacent crossing.

$\Leftarrow$ Assume the graph $f(I)$ can be drawn upward planar with $2n$ slopes. Let D be the drawing. Look at any transit crossing c. Its adjacent vertices all lie on a cycle that surrounds c (there are additional vertices subdividing the cycle that we can contract). This fixes the bimodal embedding of c. c has indegree and outdegree exactly $2n$. As already observed, the ith incoming and ith outgoing edge must have the same slope. This means that the incoming and outgoing sections belonging to the same pseudoline have the same slope. We can then draw the lines along these edges and rotate the drawing by $\pi/2$ clockwise to obtain a valid line arrangement. The lines are also directed in the right direction, because the drawing was upward. Because all lines cross exactly once (by the definition of valid pseudoline arrangements), no other crossings can occur further to the left or furthe to the right. This completes the proof.

□

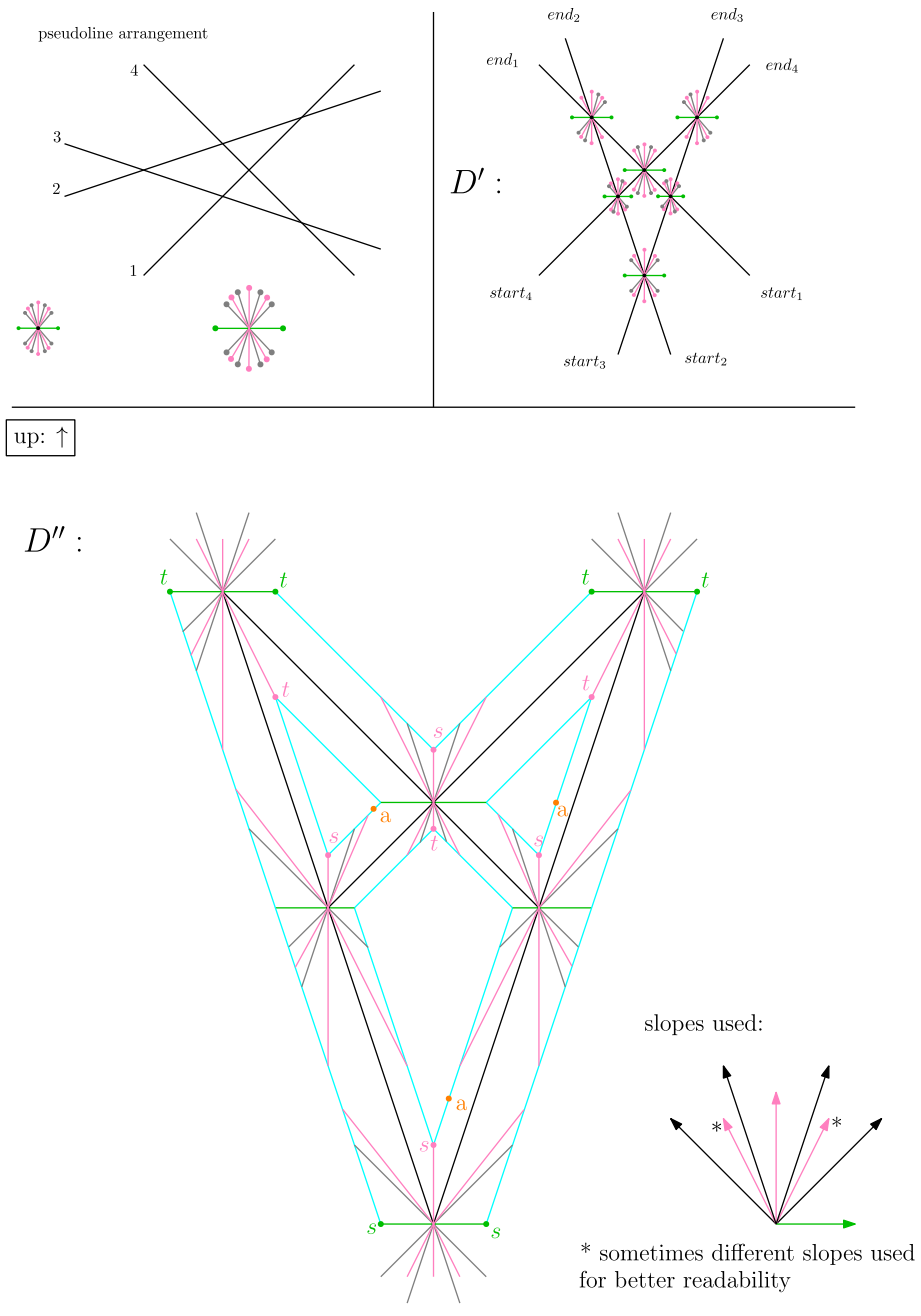

Figure 6.7: A pseupoline arrangement and the different steps of the transformation.

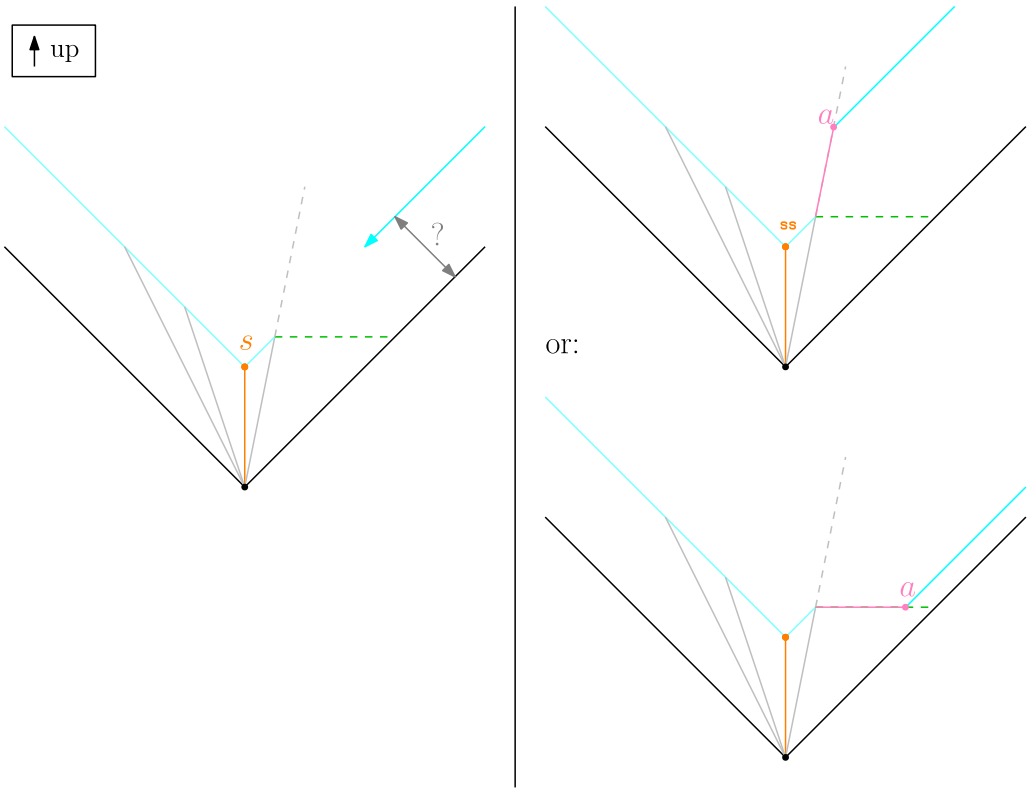

Figure 6.8: The strategy to *pull down* the adjustment vertex until hitting one of the dashed lines to complete the drawing.

7. Conclusion

The goal of this thesis was to take a closer look at algorithms and limitations for upward (planar) drawing with a limited amount of different slopes and comparing the results to drawings without the *upward* restraint.

We first looked at slope sets and found out that for two and three slopes it does not make a difference which specific slopes to choose. For four slopes we found that, regarding upward planar drawing, not all slope sets are equivalent. We found out that generally, the slope number increases when adding the constraint of upward drawing.

We then looked at upward k-slope planar drawing problems for specific graph classes. Although there are polynomial algorithms for drawing trees, cacti, and series-parallel graphs, we found that on one hand, we get exponentially big drawings in the worst-case, when drawing on the integer grid. On the other hand, the algorithms are still very limited to small graph classes and/or amounts of slopes. In Chapter 6 we have shown that the upward planar slope number problem is $\exists\mathbb{R}$-complete. Thus, restricting the direction of edges by adding the *upward* restraint, in general, gives us no computational benefit in trying to calculate a planar drawing with as few slopes as possible.

Open Problems

There are a number of open problems arising specifically from the results of this thesis. When drawing upward planar with four slopes, we found out that the choice of slopes makes a difference for which graphs can be drawn. This leaves open the possibility that there is some universal set of slopes that can always be used, if a graph can be drawn up4p. This question can be compared to [MS09]. They show that every connected cubic graph can be drawn in the plane with straight-line edges using only four distinct slopes. These four slopes must be equivalent (by the definition of equivalent slope sets in this thesis) to the slopes $\{0, \pi/4, \pi/2, 3/4 \cdot \pi\}$. However, we suspect that such a slope set does not exist regarding upward planar drawing with four slopes. An idea how to prove this can be seen in .1.

Secondly, it is still open, if the upward planar slope number is bounded by the maximum degree. We know that the planar slope number is bounded. On the other hand we know that there exist digraphs with bigger upward planar slope number than planar slope number. It would be interesting to find out if the result for the planar slope number can still be transferred to the upward planar slope number.

Thirdly, we investigated up2p and up3p drawings of series-parallel graphs. We have seen that the construction gets a lot more complicated when going from two to three slopes. Now it would be interesting to find out what happens when we allow four different slopes. We believe it is likely that the strategy used for three slopes can be recycled, but the number of different cases will increase drastically.

Last but not least, studying the worst-case area of directed trees is an interesting topic: We found cacti that need exponential size when drawn up3p on the integer grid. But it is unlikely that we need exponential size for drawing any (unordered) directed tree. Is there a sub-exponential upper bound on the minimum width and height of up3p drawings of unordered directed trees with maximum in- and outdegree three on the integer grid?

Appendix

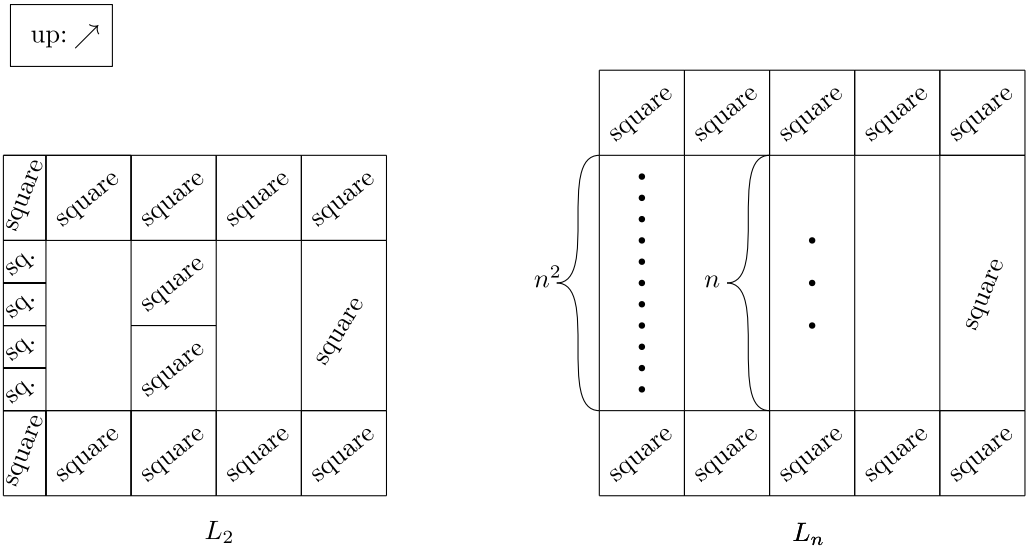

Figure .1: Graphs that can presumably only be drawn upward 4-slope planar with exactly one standard slope set.

Bibliography

[BETT99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

[DESW07] Vida Dujmović, David Eppstein, Matthew Suderman, and David R. Wood. Drawings of planar graphs with few slopes and segments. *Computational Geometry*, 38(3):194–212, Oct 2007.

[DLM20] Emilio Di Giacomo, Giuseppe Liotta, and Fabrizio Montecchiani. 1-bend upward planar slope number of sp-digraphs. *Computational Geometry*, 90:101628, 2020.

[DSW07] Vida Dujmović, Matthew Suderman, and David R. Wood. Graph drawings with few slopes. *Computational Geometry*, 38(3):181–193, 2007.

[EvdHM20] Jeff Erickson, Ivor van der Hoog, and Tillmann Miltzow. Smoothing the gap between np and er, 2020.

[Hof17] Udo Hoffmann. On the complexity of the planar slope number problem. *Journal of Graph Algorithms and Applications*, 21(2):183–193, 2017.

[KM21] Jonathan Klawitter and Tamara Mchedlidze. Upward planar drawings with two slopes. *CoRR*, abs/2106.02839, 2021.

[KPP10] Balázs Keszegh, János Pach, and Dömötör Pálvölgyi. Drawing planar graphs of bounded degree with few slopes, 2010.

[Kri18] Nadine Krisam. Drawing of level planar graphs with fixed slopes, 2018.

[KZ21] Jonathan Klawitter and Johannes Zink. Upward planar drawings with three and more slopes, 2021.

[Mat14] Jiri Matousek. Intersection graphs of segments and $\exists\mathbb{R}$, 2014.

[MS09] Padmini Mukkamala and Mario Szegedy. Geometric representation of cubic graphs with four directions. *Computational Geometry*, 42(9):842–851, 2009.

[Nö05] Martin Nöllenburg. Automated drawings of metro maps, 2005.