Constructions, Removals and How to Certify 3-Connectivity in Linear Time

Jens M. Schmidt∗
Freie Universität Berlin, Germany

Abstract

It is well-known as an existence result that every 3-connected graph \( G = (V, E) \) on more than 4 vertices admits a sequence of contractions and a sequence of removal operations to \( K_4 \) such that every intermediate graph is 3-connected. We show that both sequences can be computed in optimal time, improving the previously best known running times of \( O(|V|^2) \) to \( O(|V| + |E|) \). This settles also the open question of finding a linear time 3-connectivity test that is certifying and extends to a certifying 3-edge-connectivity test in the same time. The certificates used are easy to verify in time \( O(|E|) \).

1 Introduction

The class of 3-connected graphs has been studied intensively for many reasons in the past 50 years. One algorithmic reason is that graph problems can often be reduced to handle 3-connected graphs; applications include problems in graph drawing (see [18] for a survey), problems related to planarity [6, 12] and online problems on planar graphs (see [5] for a survey). From a complexity point of view, 3-connectivity is in particular important for problems dealing with longest paths, because it lies, somewhat surprisingly, on the borderline of NP-hardness: Finding a Hamiltonian cycle is NP-hard for 3-connected planar graphs [11] but becomes solvable in linear running time [8] for higher connectivities, as 4-connected planar graphs have been proven to be Hamiltonian [23].

We want to design efficient algorithms from inductively defined constructions of graph classes. In general, such constructions start with a set of base graphs and apply iteratively operations from a fixed set of operations such that precisely the members of the graph class of interest are constructed. This way we obtain a (not necessarily unique) sequence of graphs for each member \( G \) of the graph class, which we call a construction sequence of \( G \). The construction does not only give a computational approach to test membership in these classes, it can also be exploited to prove properties of the graph class using only the fixed set of operations applied in every step. Fortunately, graph theory provides inductively defined constructions for many graph classes, including planar graphs, triangulations, \( k \)-connected graphs for \( k \leq 4 \), regular graphs and various intersections of these classes [3, 4, 14]. However, most of these constructions have not been exploited computationally.

For the class of 3-connected graphs, one of the most noted constructions is due to Tutte [24], based on the following fact: Every 3-connected graph \( G \) on more than 4 vertices contains a contractible edge, i.e., an edge that preserves the graph to be 3-connected upon contraction. Contracting iteratively this edge yields a sequence of 3-connected graphs top-down from \( G \) to a

∗This research was supported by the Deutsche Forschungsgemeinschaft within the research training group “Methods for Discrete Structures” (GRK 1408). Email: jens.schmidt@inf.fu-berlin.de.
$K_4$-multigraph. Unfortunately, also non-3-connected graphs can contain contractible edges, but adding a side condition establishes a full characterization: A graph $G$ on more than 4 vertices is 3-connected if and only if there is a sequence of contractions from $G$ to a $K_4$-multigraph on edges $e$ with both end vertices having at least 3 neighbors [9]. Every contracted edge in this sequence is then contractible. It is also possible to describe this sequence bottom-up from $K_4$ to $G$ by using the inverse operations edge addition and vertex splitting; in fact this is the original form as stated in Tutte’s famous wheel theorem [24].

Barnette and Grünbaum [2] and Tutte [25] give a different construction of 3-connected graphs that is based on the following argument: Every 3-connected graph $G \neq K_4$ contains a removable edge. Removing this edge leads, similar as in the sequence of contractions, to a top-down construction sequence from $G$ to $K_4$. Adding a side condition then fully characterizes 3-connected graphs. We will define removals and removable edges in Section 2. Again, the original proposed construction was given bottom-up from $K_4$ to $G$, using three operations.

Although both existence theorems on contractible and removable edges are used frequently in graph theory [21, 22, 25], the first non-trivial computational results to create the corresponding construction sequences were published more than 45 years afterwards: In 2006, Albroscheit [1] gave an algorithm that computes a construction sequence for 3-connected graphs in $O(|V|^2)$ time in which contractions and removals are allowed to intermix. In 2010, an algorithm was given [20] that constructs the (pure) sequences of contractions and removals, respectively, in the same running time. One of the building blocks of this algorithm is a straight-forward transformation from the sequence of removals to the sequence of contractions in time $O(|E|)$. This shows that the sequence of Barnette and Grünbaum is algorithmically at least as powerful as the sequence of contractions. It is important to note that all algorithms do not rely on the 3-connectivity test of Hopcroft and Tarjan [13], which runs in linear time but is rather involved. It was also shown that all previously mentioned construction sequences can be stored in linear space $O(n + m)$ [20]. Nevertheless, we are not aware of any algorithm that computes any of these sequences in subquadratic time up to now.

The main contribution of this paper is an optimal algorithm that computes the construction sequence of Barnette and Grünbaum bottom-up in time and space $O(|V| + |E|)$. This has a number of consequences.

Top-down and bottom-up variants of both constructions. One can immediately obtain the sequence of removals out of Barnette and Grünbaum’s construction sequence by replacing every operation with its inverse removal operation. Applying the transformation of [20] implies optimal time and space algorithms for the sequence of contractions and its bottom-up variant as well.

Certifying 3-connectivity in linear time. Blum and Kannan [7] initiated the concept of programs that check their work. Mehlhorn and Näher [15, 16, 17] developed this idea further and introduced the concept of certifying algorithms, which give a small and easy-to-verify certificate of correctness along with their output. Achieving such algorithms is a major goal for problems where the fastest solutions known are complicated and difficult to implement. Testing a graph on 3-connectivity is such a problem, but surprisingly few work has been devoted to certify 3-connectivity, although sophisticated linear-time recognition algorithms (not giving an easy-to-verify certificate) are known for over 35 years [13, 26, 27]. The currently fastest algorithms that certify 3-connectivity need $O(|V|^2)$ time and use construction sequences as certificates [1, 20]. Recently, a linear time certifying algorithm for 3-connectivity has been given for the subclass of Hamiltonian graphs, when the Hamiltonian cycle is part of the input [9]. In general, finding a certifying algorithm for 3-connectivity in subquadratic time is an open problem [9].
We give a linear-time certifying algorithm for 3-connectivity by using Barnette and Grünbaum’s construction sequence as certificate. The certificate can be easily verified in time $O(|E|)$, as shown in [20]. This implies also a new, simple-to-implement and certifying test on 3-connectivity in linear time and space that is path-based and neither relies on the algorithm of Hopcroft and Tarjan nor uses low-points.

Certifying 3-edge-connectivity in linear time. We are not aware of any test for 3-edge-connectivity that is certifying and runs in linear time. Galil and Italiano [10] show that testing $k$-edge-connectivity of a graph $G$ can be reduced to test $k$-vertex-connectivity on a slightly modified graph $G'$, blowing up the number of vertices and edges only by a factor $O(k)$. For $k = 3$, the reduction blows up each vertex $v \in V(G)$ to a wheel graph with as many spokes as $v$ has neighbors. Constructing $G'$ and applying the certifying 3-vertex-connectivity test to $G'$ yields a certifying 3-edge-connectivity test in linear time and space. However, we have to augment the certificate by the mapping $\phi$ that maps each vertex of $G$ to the vertices and edges being contained in the corresponding wheel graph in $G'$. This ensures that the construction of $G'$ can be verified while preserving linear time and space.

2 Construction Sequences

Let $G = (V,E)$ be a finite graph with $n$ vertices and $m$ edges. For $k \geq 1$, a graph $G$ is $k$-\textit{connected} if $n > k$ and deleting every set of $k - 1$ vertices leaves a connected graph. A vertex (a pair of vertices) that leaves a disconnected graph upon deletion is called a \textit{cut vertex} (a \textit{separation pair}). Let $v \rightarrow_G w$ denote a path $P$ from vertex $v$ to vertex $w$ in $G$ and let $s(P) := v$ and $t(P) := w$. For a vertex $v$ in $G$, let $N(v) = \{w \mid vw \in E\}$ denote its set of neighbors and $\text{deg}(v)$ its degree. Let $\delta(G)$ be the minimum degree in $G$.

Let $K_n$ be the complete graph on $n$ vertices and let $K_n^m$ be the complete graph on $n$ vertices with $m$ edges between each pair of vertices. For a rooted tree $T$ and $x \in V(T)$, let $T(x)$ be the maximal subtree of $T$ rooted at $x$. We assume for convenience that the input graph $G$ is simple for the rest of the paper, although all results extend to multigraphs. A \textit{subdivision} of a graph $G$ replaces each edge of $G$ by a path of length at least one. Conversely, we want a notation to get back to the graph without subdivided edges. If $\text{deg}(v) = 2$ and $|N(v) \setminus \{v\}| = 2$, let \textit{smoothing} $v$ delete $v$ followed by adding an edge between its neighbors. If one of the conditions is violated, let \textit{smoothing} $v$ not change the graph.

\textit{Removing} an edge $e = xy$ of a graph deletes $e$ followed by smoothing $x$ and $y$. An edge of $G$ is called \textit{removable}, if removing it results in a 3-connected graph. Iteratively removing removable edges in a 3-connected graph $G$ leads to a \textit{sequence of removals} from $G$ to $K_4$, the existence of which characterizes 3-connected graphs when adding a side condition similar as in the sequence of contractions. We describe the equivalent bottom-up construction of $G$ due to Barnette and Grünbaum. The construction starts with $K_4$ and applies iteratively one of the three following operations, which are called \textit{BG-operations} (see Figure 1):

1. Add an edge $xy$ (possibly a parallel edge).
2. Subdivide an edge $ab$ by a vertex $x$ and add the edge $xy$ for a vertex $y \notin \{a,b\}$.
3. Subdivide two non-parallel edges by vertices $x$ and $y$, respectively, and add the edge $xy$.

Let $G_4, G_5, \ldots, G_z$ with $G_4 = K_4$ and $G_z = G$ be a construction sequence of $G$ using BG-operations. As $K_4$ is 3-connected and BG-operations preserve 3-connectivity, every $G_l$ with $4 < l \leq z$ is also 3-connected. We represent the construction sequence in a different, but equivalent way, as shown in [2, 20]: Each graph $G_l$ corresponds to a unique $G_l$-subdivision $S_l$.
in $G$, which can be readily seen by iteratively deleting the removable edges in the top-down variant. In this representation, the vertices of $G_l$ correspond to the vertices in $S_l$ of degree at least 3; we call the latter real vertices. We define the operations on $S_l$ that correspond to BG-operations. Let $V_{\text{real}}(S_l)$ be the set of real vertices in $S_l$. The links of $S_l$ are the unique paths in $S_l$ that contain real end vertices but no other real vertex. Let two links be parallel if they share the same end vertices.

Definition 1. A BG-path for a subgraph $S_l \subset G$ is a path $P = x \rightarrow y$ with the properties:

1. $S_l \setminus P = \{x, y\}$
2. Every link of $S_l$ that contains $x$ and $y$, contains them as end vertices.
3. If $x$ and $y$ are inner vertices of links $L_x$ and $L_y$ of $S_l$, respectively, and $|V_{\text{real}}(S_l)| \geq 4$, then $L_x$ and $L_y$ are not parallel.

It is easy to see that every BG-path for $S_l$ corresponds to a BG-operation on $G_l$ and vice versa. The choice of the $K_4$-subdivision $S_4$ is not crucial [20]: At the expense of having additional parallel edges in intermediate graphs $G_l$, there exists a construction sequence to $G$ from each prescribed $K_4$-subdivision in $G$. This provides an efficient computational approach to construction sequences, since we can start with an arbitrary $K_4$-subdivision $S_4$ in $G$. The representation with subdivisions allows then to search the next BG-path in the neighborhood of the current subdivision in $G$. We summarize the results.

Theorem 2. The following statements are equivalent:

1. A simple graph $G$ is 3-connected
2. $\exists$ sequence of BG-operations from $K_4$ to $G$ (see [2, 25])
3. $\exists$ sequence of BG-paths from a $K_4$-subdivision in $G$ to $G$ and $\delta(G) \geq 3$ (see [2, 20])
4. $\exists$ sequence of BG-paths from each $K_4$-subdivision in $G$ to $G$ and $\delta(G) \geq 3$ (see [20])
5. $\exists$ sequence of removals from $G$ to $K_4$ on removable edges $e = xy$
   with $|N(x)| \geq 3$, $|N(y)| \geq 3$ and $|N(x) \cup N(y)| \geq 5$ (see [20])
6. $\exists$ sequence of contractions from $G$ to $K_4$ on edges $e = xy$ with $|N(x)| \geq 3$
   and $|N(y)| \geq 3$ (see [24])

The following Lemma allows to focus only on computing sequence 2.(3).

Lemma 3 ([20, Proof of Theorem 2 and Lemma 4.1]). There is an algorithm that transforms a given sequence 2.(2), 2.(3) or 2.(5) to each of the sequences 2.(2)-2.(6) in linear time. If the transformation yields one of the sequences 2.(2)-2.(5), the number of operations is preserved.

Throughout the rest of the paper, a construction sequence will therefore refer to sequence 2.(3), unless stated otherwise. To construct such a sequence, we will use the following Lemma.
Lemma 4 ([20]). Let \( G \) be a 3-connected graph and \( H \subset G \) with \( H \) being a subdivision of a 3-connected graph. Then there is a BG-path for \( H \) in \( G \). Moreover, every link of \( H \) of length at least 2 contains an inner vertex on which a BG-path for \( H \) starts.

Every contraction sequence 2.6 contains exactly \( n - 4 \) contractions, implying that the \( K_4 \)-multigraph contains exactly \( m - n - \) 2 parallel edges. The number of removals is also fixed.

Lemma 5. Every sequence 2.2-2.5 contains exactly \( m - n - 2 \) operations, i.e., \( z = m - n + 2 \).

Proof. It suffices to show the claim for each sequence 2.2 with Lemma 3. Let \( a, b \) and \( c \) denote the number of BG-operations in the sequence that create zero, one and two new vertices, respectively. Then \( b + 2c = n - 4 \) and \( a + 2b + 3c = m - 6 \) hold, since \( K_4 \) consists of four vertices and six edges. Subtracting the equations gives that \( a + b + c = m - n - 2 \).

### 3 Chain Decompositions

Let \( G \) be a 3-connected graph. According to Lemma 4, it suffices to add iteratively BG-paths to an arbitrary \( K_4 \)-subdivision in \( G \) to get a construction sequence. Note that we cannot make wrong decisions when choosing a BG-path, since Lemma 4 can always be applied on the new subdivision and therefore ensures a completion of the sequence. Instead of starting with a \( K_4 \)-subdivision, we will w.l.o.g. start with a \( K_3^3 \)-subdivision \( S_3 \) and find a BG-path for \( S_3 \) that results in a \( K_4 \)-subdivision. We first show how \( S_3 \) is computed and then describe a decomposition of \( G \) into special paths that allows us to find the BG-paths efficiently.

A Depth First Search (DFS) is performed on \( G \), assigning a Depth First Index (DFI) to every vertex. Let \( T \) be the corresponding DFS-tree, \( r \) be the root of \( T \) and \( u \) be the vertex that is visited second. Both, \( r \) and \( u \), have exactly one child in \( T \), as otherwise they would form a separation pair in \( G \). For two vertices \( v \) and \( w \) in \( T \), let \( v = (\text{proper}) \text{ ancestor of } w \) and \( w \) be a \( (\text{proper}) \text{ descendant of } v \) if \( v \in V(r \rightarrow_T w) \) (and \( v \neq w \)). A backedge is an edge \( vw \in E(G) \setminus E(T) \) oriented from \( v \) to \( w \) with \( v \) being an ancestor of \( w \) (note that this differs from standard notation). A backedge \( vw \) enters a subtree \( T' \) of a tree if \( v \notin V(T') \) but \( w \in V(T') \).

To compute \( S_3 \), we choose two backedges \( ra \) and \( rb \) and denote the least common ancestor of \( a \) and \( b \) in \( T \) with \( x \). The paths \( x \rightarrow_T r \), \( ra \cup a \rightarrow_T x \) and \( rb \cup b \rightarrow_T x \) are the three subdivided edges of \( S_3 \) in \( G \) with real vertices \( r \) and \( x \). Now, \( G \) is decomposed into special paths \( \{C_0, C_1, \ldots, C_{m-n+1}\} =: C \), called chains, whose edge sets partition \( E(G) \). The decomposition imposes a total order \(<\) on \( C \) with \( C_0 < C_1 < \ldots < C_{m-n+1} \) that is identical to the order in which the chains were computed. We set \( C_0 := x \rightarrow_T r \), \( C_1 := ra \cup a \rightarrow_T x \) and \( C_2 := rb \cup b \rightarrow_T x \). The remaining chains are then computed by applying the following procedure subsequently for each vertex \( v \) in increasing DFI-order: For every backedge \( vw \) not in a chain, we traverse the path \( w \rightarrow_T r \) until a vertex \( x \) is found that is already contained in a chain. The traversed path \( v \rightarrow_G x \) including \( vw \) forms the new chain.

Note that every chain \( C_i \neq C_0 \) contains exactly one backedge and thus \( |C| = m - n + 2 \). Also, \( s(C_i) \) is always a proper ancestor of \( t(C_i) \). Chains admit the following tree structure.

**Definition 6.** Let the **parent** of a chain \( C_i \neq C_0 \) be the chain that contains the edge from \( t(C_i) \) to the parent of \( t(C_i) \) in \( T \).

**Lemma 7.** The parent relation defines a tree \( U \) with \( V(U) = C \) and root \( C_0 \).

Proof. Let \( D_0 \neq C_0 \) be a chain in \( C \) and let \( D_1, \ldots, D_k \) be the sequence of chains containing the edges of \( t(D_0) \rightarrow_T r \) in that order, omitting double occurrences. By definition of the parent
relation, each $D_i$, $0 \leq i < k$, has parent $D_{i+1}$. It follows with $D_k = C_0$ that $U$ is connected. Moreover, $U$ is acyclic, as parent chains are always smaller in $<$ than their children by definition of the decomposition.

### 3.1 Classifying Chains and Restrictions

We extend the chain decomposition to assign one of the types $1$, $2a$, $2b$, $3a$ and $3b$ to each chain in $C \setminus \{C_0\}$. The motivation for this classification is that chains of certain types are, under some conditions, BG-paths and therefore allow to compute the next step of the construction sequence. The types are defined by Algorithm 1: E.g., a chain $C_i$ with parent $C_k$ is of type $1$ if $t(C_i) \to_T s(C_i) \subseteq C_k$ and of type $2$ if it is not of type $1$ and $s(C_i) = t(C_k)$ holds (see Figures 2 and 5(b)). All chains are unmarked at the beginning of Algorithm 1. It is not difficult to show that the decomposition and classification can be carried out in linear time. We omit a proof.

**Algorithm 1**

```plaintext
classify($C_i \in C \setminus \{C_0\}$, DFS-tree $T$)
1: $C_k := parent(C_i)$ → the parent of $C_i$ in $U$: $C_k < C_i$
2: if $t(C_i) \rightarrow_T s(C_i)$ is contained in $C_k$ then → type 1
3: assign type 1 to $C_i$
4: else if $s(C_i) = s(C_k)$ then → type 2: $C_k \neq C_0$, $t(C_i)$ is inner vertex of $C_k$
5: if $C_i$ is a backedge then → type 2a
6: assign type 2a to $C_i$
7: else → type 2b
8: assign type 2b to $C_i$; mark $C_i$
9: else → type 3: $s(C_i) \neq s(C_k)$, $C_k \neq C_0$, $t(C_i)$ is inner vertex of $C_k$
10: if $C_k$ is not marked then → type 3a
11: assign type 3a to $C_i$
12: else → $C_k$ is marked
13: assign type 3b to $C_i$; create a list $L_i = \{C_i\}$; $C_j := C_k$ → type 3b
14: while $C_j$ is marked do → $L_i$ is called a caterpillar
15: unmark $C_j$; append $C_j$ to $L_i$; $C_j := parent(C_j)$
```

**Lemma 8.** Computing a chain decomposition of a $3$-connected graph and classifying each chain with Algorithm 1 takes running time $O(n + m)$.

**Definition 9.** Let a subdivision $S_t \subseteq G$ be upwards-closed if for each vertex in $S_t$ the edge to its parent is in $E(S_t)$. Let $S_t$ be modular if $S_t$ is the union of chains.

In order to find BG-paths efficiently, we want to restrict every subdivision $S_t$ to be upwards-closed and modular. However, configurations exist where no BG-path for a subdivision $S_t$ is a chain, e.g., the subdivision $S_3 = \{C_0, C_1, C_2\}$ in Figure 2. This violates the modularity of $S_{t+1}$ and we have to weaken the restriction: We will allow intermediate subdivisions that are neither upwards-closed nor modular but demand in these cases that we can find a set of $t$ BG-paths in advance that restores these properties after $t$ steps.

We impose the additional restriction that each link of $S_t$ that consists only of tree edges has no parallel link, except $C_0$ in $S_3$. This prevents BG-path candidates from violating property 1.3 due to the DFS-structure. We summarize the restrictions.

(R1) For each upwards-closed and modular subdivision $S_t$, BG-paths are only added as single chains of type $1$, $2a$ or $3a$, with $S_{t+1}$ being upwards-closed and modular or as sets of $t > 1$ subsequent BG-paths constructing an upwards-closed modular subdivision $S_{t+t}$ that differs from $S_t$ in exactly $t$ chains of types $2b$ and $3b$.  

(R2) For each upwards-closed and modular subdivision $S_I$, the links of $S_I$ that consist only of tree edges of $T$ have no parallel links, except $C_0$ in $S_3$.

We refer to the current upwards-closed and modular subdivision in a construction sequence that is restricted by (R1) and (R2) as $S_I^R$. The existence of a restricted sequence is shown in Section 3.2. We show that chains of type 3a help to find BG-paths efficiently (proof omitted).

**Lemma 10.** Let $C_i$ be a chain of type 3a and $C_k$ the parent of $C_i$ such that $C_k$ but not $C_i$ is contained in $S_I^R$. Then $C_i$ is a BG-path for $S_I^R$ preserving (R1) and (R2).

### 3.2 Caterpillars and Existence of the Restricted Sequence

While chains of type 3a form BG-operations under the conditions of Lemma 10, chains of types 1 and 2 in general do not. For every chain $C_i$ of type 3b, Algorithm 1 collects a list $L_i$ of chains that contains only $C_i$ and chains of type 2b (see line 15). We call each list $L_i$ a *caterpillar*.

**Definition 11.** Let the parent of a caterpillar $L_j$ be the parent of the chain in $L_j$ that is minimal with respect to $<$. Let a caterpillar $L_j$ with parent $C_k$ be *bad* for a subdivision $S_I$ if $s(C_j)$ is a descendant of $t(C_k)$ and $s(C_j) \rightarrow C_k$; $s(C_j)$ contains no inner real vertex (see Figure 3(a)). Otherwise, $L_j$ is called a *good* caterpillar (see Figure 3(b)).

Caterpillars bundle the single chains of type 2b, which cannot immediately be added as BG-paths. They also offer a simple decomposition into successive BG-paths.

**Lemma 12.** Let $L_j$ be a caterpillar that consists of $t$ chains and has parent $C_k$. Let $C_k$ but no chain in $L_j$ be contained in $S_I^R$. If $L_j$ is good, $L_j$ can be efficiently decomposed into $t$ successive BG-paths satisfying (R1) and (R2).

**Proof.** (sketch) Let $y$ be the last vertex of the minimal chain in $L_j$ and let $C_h$ be the parent of $C_j$. We add either the path $P := C_j \cup (t(C_j) \rightarrow_T y)$, followed by $C_i \setminus P$ for all chains $C_i \in L_j \{C_j\}$ (see Figure 3(b)) or the path $(C_j \cup C_h) \setminus (t(C_j) \rightarrow_T y)$, followed by $t(C_j) \rightarrow_T y$ and $C_i \setminus (t(C_j) \rightarrow_T y)$ for every remaining chain $C_i \in L_j \{C_j, C_h\}$.

**Definition 13.** We define the equivalence relation $\sim$ on $E(G) \setminus E(S_I)$ with $e \sim e$ for all $e \in E(G) \setminus E(S_I)$ and with $e \sim f$ for all $e, f \in E(G) \setminus E(S_I)$ if there is a path $e \rightarrow_G f$ without an inner vertex in $S_I$. Let the *segments* of $S_I$ be the subgraphs of $G$ that are induced by the equivalence classes of $\sim$. Let $H \cap S_I$ be the *attachment points* of $H$.

**Definition 14.** For a chain $C_i$ and a subdivision $S_I^R$, let $\text{Children}_{12}(C_i)$ be the set of children of $C_i$ of types 1 and 2 that are not contained in $S_I^R$ and let $\text{Type}_3(C_i)$ be the set of chains of type 3 that start at a vertex in $C_i$ and are not contained in $S_I^R$.

The following theorem is a key result of this paper and leads not only to an existence proof of the restricted construction sequence but also to an efficient algorithm for computing it.
Theorem 15. For a subdivision $S^R$, let $C_i$ be a chain such that $\text{Children}_{12}(C_j) = \text{Type}_3(C_j) = \emptyset$ holds for every proper ancestor $C_j$ of $C_i$. Then all chains in $\text{Children}_{12}(C_i) \cup \text{Type}_3(C_i)$ and their proper ancestors that are not already contained in $S^R$ can be successively added as BG-paths (possibly being part of caterpillars) such that $(R_1)$ and $(R_2)$ is preserved. Moreover, the chains in $\text{Type}_3(C_i)$ that are contained in segments in which the minimal chain is not contained in $\text{Children}_{12}(C_i)$ can be added at any point in time in arbitrary order (together with their proper ancestors that are not contained in $S^R$).

The precondition of Theorem 15 is met in every subdivision: For $S^R$, $C_0$ is the desired chain and applying the Theorem on $C_0$ allows to take the descendants of $C_0$ in $U$ in subsequent subdivisions. This ensures the existence of the restricted construction sequence.

Corollary 16. Let $G$ be a 3-connected graph with a chain decomposition $C = \{C_0, \ldots, C_{m-n+1}\}$. Then there is a construction sequence of $G$ restricted by $(R_1)$ and $(R_2)$ that starts with $S^R_3 = \{C_0 \cup C_1 \cup C_2\}$.

4 A Linear-Time Algorithm

With Lemma 8, a chain decomposition, a subdivision $S^R_3$ and the classification of chains can be computed in time $O(n+m)$. Theorem 15 provides an algorithmic method to find iteratively BG-paths building the restricted construction sequence: Iteratively for each chain $C_i$, $0 \leq i \leq m-n$, we add all chains in $\text{Children}_{12}(C_i) \cup \text{Type}_3(C_i)$ (we say that $C_i$ is processed). Note that $C_i$ meets the precondition of Theorem 15 and that $\text{Children}_{12}(C_i)$ and $\text{Type}_3(C_i)$ can be built in time $O(|C_i| + \text{Children}_{12}(C_i)| + |\text{Type}_3(C_i)|)$ by storing lists of type 3 chains at every vertex. We partition the chains in $\text{Type}_3(C_i)$ into the different segments of $S^R$ containing them by storing a pointer on each $C_j \in \text{Type}_3(C_i)$ to the minimal chain $D$ of the segment containing $C_j$. The chain $D$ is computed by traversing $T$ from $t(C_i)$ to the root until the next vertex is contained in $S^R_i$. The current vertex is then an inner vertex of $D$ (each inner vertex has a pointer to its chain) and we mark each vertex of the traversed path with $D$. Further traversals get $D$ by stopping at the first marker that points to a chain not in $S^R_i$. Since all traversed chains will be added, the running time amortizes to a total of $O(n+m)$.

First, we add all chains in $\text{Type}_3(C_i)$ that are contained in segments in which the minimal chain is not contained in $\text{Children}_{12}(C_i)$ (this can be checked in constant time per chain). According to Theorem 15, the chains can be added in arbitrary order, as long as their proper ancestors that are not in $S^R$ are added before. We want to add the remaining chains in $\text{Children}_{12}(C_i) \cup \text{Type}_3(C_i)$. However, Theorem 15 does not specify in which order the chains have to be added, so we need to compute a valid order on them.

Let $C_j$ be a remaining chain in $\text{Type}_3(C_i)$ and let $H$ be the segment containing it. Then the minimal chain $D$ in $H$ is of type 1 or 2, as it is contained in $\text{Children}_{12}(C_i)$. If $D$ is of type 1 or 2a, $s(D)$, $t(D)$, $s(C_j)$ and all other attachment points of $H$ must be contained in $C_i$. The same holds for the remaining case of $D$ being of type 2b, as the type 3b-chain in the caterpillar containing $D$ cannot start in a proper ancestor of $C_i$ by assumption. Let the dependent path of $H$ be the maximal path in $C_i$ connecting two attachment points of $H$, e.g., for $D$ being of type 1 or 2a, the dependent path is just $s(D) \rightarrow C_i \rightarrow t(D)$. We can compute all attachment points of $H$ and therefore the dependent path of $H$ efficiently, as the previous computation provides $D$ and the set of chains $H \cap \text{Type}_3(C_i)$; we just have to add $s(D)$ and $t(D)$ to the start vertices of the latter chains.

If $D$ is a chain of type 2a (thus, $H = D$) and $t(D)$ is real, we can add $D$. Otherwise, restriction $(R_2)$ implies that every segment $H$ that has a dependent path $P$ without inner real vertices does neither contain chains nor caterpillars forming BG-paths while preserving $(R_1)$.
and \((R_2)\). Conversely, if \(P\) contains an inner real vertex, all chains in \(H \cap (\text{Children}_{12}(C_i) \cup \text{Type}_3(C_i))\) can be added: If \(D\) is of type 1, \(D\) does not violate \((R_2)\) and can be added and if \(D\) is of type \(2b\), the caterpillar containing \(D\) is good and can be added with Lemma 12. As the minimal chain in \(H \setminus \{D\}\) is not contained in \(\text{Children}_{12}(C_i)\), the remaining chains in \(H \cap \text{Type}_3(C_i)\), if exist, can be added as well using Theorem 15.

Finding a valid order on the remaining chains in \(\text{Children}_{12}(C_i) \cup \text{Type}_3(C_i)\) thus reduces to finding an order on their segments such that the dependent paths of the segments contain inner real vertices. Having this sequence would allow to add a \(BG\)-path due to Theorem 2 when processing some chain, say \(C_i\). Having this order on their segments such that the dependent paths of the segments contain inner real vertices. Having this sequence would allow to add a \(BG\)-path due to Theorem 2 when processing some chain, say \(C_i\).

The construction sequences \(2.1\), \(2.2\), \(2.3\) and \(2.6\) of a 3-connected graph can be computed in time \(O(n + m)\).

A New Certifying 3-Connectivity Test. It remains to deal with the case when the input graph \(G\) is not 3-connected. For simplicity, we assume \(G\) to be 2-connected, although the chain decomposition can check this fact. If \(G\) is not 3-connected, the described algorithm fails to add a \(BG\)-path to \(H\) when processing some chain, say \(C_i\). Therefore, after the processing phase for \(C_i\), \(\text{Children}_{12}(C_i)\) must still contain a chain \(C_j\). Let \(H\) be the segment containing \(C_j\) and let \(H' \supseteq H\) be the set of segments that map to the connected component of the interval overlap graph containing \(I(H)\). Then the union of dependent paths of the segments in \(H'\) is a path \(P \subseteq C_i\) and the two extremal attachment points on \(P\) of segments in \(H'\) build
A 3-connected input graph $G$ with $n = 18$ and $m = 34$. Straight lines depict the edges of the DFS-tree $T$.

The decomposition of $G$ into $m - n + 2 = 18$ chains. Light solid chains are of type 1, dashed ones of type 2 and black solid ones of type 3. The only chain of type 2a is $C_3$. The only chains of type 3b are $C_{14}$ and $C_{16}$, which create the caterpillars $L_{14} = \{C_{14}, C_6, C_5\}$ and $L_{16} = \{C_{16}, C_4\}$, respectively.

The subdivision $S_9^R = \{C_0, C_1, C_2\}$ (thick edges). We start with processing $C_0$. Since $\text{Children}_{12}(C_0) = \emptyset$, we can add all chains in Type$_3(C_0) = \{C_7, C_8, C_9\}$. The first two have parents that are already contained in $S_9^R$. We thus add one of them as BG-path, say $C_7$.

The $K_4$-subdivision $S_{10}^R$. Its real vertices are depicted in black. Note that choosing $C_8$ instead of $C_7$ would have led also to a $K_4$-subdivision. After adding the remaining chain $C_8$ as BG-path, the parent of $C_9$ is contained in $S_{10}^R$ and can therefore be added as well.

We process $C_1$ next, but have to continue to process $C_2$, as there is nothing to add. According to Theorem 15, $C_3$, $C_{10}$, $C_{11}$, $C_{12}$, $C_{13}$, $C_{15}$ and the caterpillar $L_{14}$ can be added.

We first add $C_{11}$, as its segment does not contain a chain in $\text{Children}_{12}(C_2)$. To obtain the right order of the remaining chains, we group them by segments.

The subdivision $S_{14}^R$. The next non-trivial chain to process is $C_5$. The interval $[v_{11}, v_1] \subset I(L_{16})$ contains the inner real vertex $v_{12}$ and overlaps with $[v_{10}, v_{12}] \subset I(C_5)$. This implies that $L_{16}$ can be added, forming the two BG-paths $v_{11} \rightarrow_{G \setminus E(S_{15})} v_1$ and $v_{12} \rightarrow_{T} v_{17}$.

The subdivision $S_{17}^R$. It remains to process the chain $C_6$, where $C_{17}$ is added as the last BG-path of the construction sequence. This results in the subdivision $S_{18}^R$, which is identical to $G$.

Figure 5: A running example of the algorithm.
a separation pair. This pair certifies that $G$ is not 3-connected and can be computed in linear time.
A Omitted Proofs

We give the omitted proofs and the preparatory lemmas that lead to them.

**Lemma 18** (aka Lemma 8). Computing a chain decomposition of a 3-connected graph and classifying each chain with Algorithm 1 takes running time $O(n + m)$.

**Proof.** The DFS tree $T$ can be obtained in time $O(n + m)$. The subdivision $S_3$ can be found in time linearly dependent on $E(S_3)$ by taking two arbitrary back edges $ra$ and $rb$ with $r$ being the root of $T$ and finding the lowest common ancestor of $a$ and $b$ by traversing $T$ upwards. The computation of each remaining chain $C_i$, $i > 2$, takes time linearly dependent on its length, too, which gives a running time of $O(n + m)$ for the chain decomposition.

In order to obtain a fast classification in Algorithm 1, we store the following information on each chain $C_i$: A pointer to its parent $C_k$ (for $C_i \neq C_0$), pointers to $s(C_i)$ and $t(C_i)$ and the information whether $C_i$ is a backedge. In addition, for each inner vertex of $C_i$ a pointer to $C_i$ is stored. That allows us to check vertices on being contained as inner vertices or end vertices in arbitrary chains in $O(1)$. If $C_k = C_0$, we can check the condition on $C_i$ being of type 1 in constant time by testing whether $s(C_i)$ and $t(C_i)$ are contained in $C_0$. If $C_k \neq C_0$, we check in constant time whether $s(C_i)$ and $t(C_i)$ are contained in $C_k \setminus \{s(C_k)\}$. The condition for type 2 needs constant time as well. Every chain is marked at most once, therefore unmarked as most once in line 15 of Algorithm 1, which gives a total running time of $O(n + m)$. 

**Lemma 19.** Let $S_l$ be upwards-closed and modular. Then a BG-path $P$ for $S_l$ is a chain if and only if $S_{l+1}$ is upwards-closed and modular.

**Proof.** If $P$ is a chain, $t(P)$ is contained in $S_l$ and $S_{l+1}$ must be upwards-closed and modular due to the DFS structure. If $P$ is not a chain, we assume to the contrary that $S_{l+1}$ is upwards-closed and modular. Then $P$ must be the union of $t > 1$ chains; let $C_i$ be the first chain in $P$. Now $P$ cannot start with $t(C_i)$, since $s(C_i)$ is in $S_l$ and property 1.1 contradicts $t > 1$. Thus, $P$ starts with $s(C_i)$, which contradicts $t > 1$ as well, as $S_{l+1}$ is upwards-closed and a second chain in $P$ would include another backedge in $P$ at a vertex that is already incident to two DFS tree edges.

Lemma 19 shows that this restriction implies every BG-path to be a chain.

**Lemma 20.** Each path $P$ in $S_l^R$ having properties 1.1 and 1.2 is a BG-path. If $P$ is additionally a chain of type 2a or 3a, $(R_1)$ and $(R_2)$ are preserved.

**Proof.** For the first claim, assume to the contrary that $P$ violates property 1.3. Then $|V_{real}(S_l)| \geq 4$ must hold and $S_l \neq S_3$ follows. Let $R$ and $Q$ be the parallel links of $S_l$ that contain the end vertices of $P$ as inner vertices, respectively. At least one of them, say $R$, contains a backedge, since otherwise $T$ would contain a cycle. Let $C_i \neq C_0$ be the chain in $S_l$ that contains $R$. Since $C_i$ contains exactly one backedge, $s(C_i)$ is an end vertex of $R$. If $R \subset C_i$, $Q$ must contain a backedge, as $t(C_i)$ is an inner real vertex of $t(R) \rightarrow T s(R)$. In that case, all inner vertices of $Q$ lie in a subtree of $T$ that cannot be reached by $P$ due to property 1.1 and $S_l$ being upwards-closed. Thus, $R = C_i$ and with the same argument $Q = t(C_i) \rightarrow T s(C_i)$ holds. With $(R_2)$, $S_l$ must be $S_3$ and $Q = C_0$, which contradicts our assumption.

For the second claim, each chain $C_i$ of type 2 or 3 has by definition an inner real vertex in $t(C_i) \rightarrow T s(C_i)$ and therefore preserves $(R_2)$. If $C_i$ is of type 2a or 3a, $(R_1)$ is preserved as well, as $S_{l+1}$ is upwards-closed and modular with Lemma 19.

We show that chains of type 3a help to find BG-paths efficiently (Lemma 10) as part of the following Lemma.
Figure 5: The effect of restriction \((R_2)\) on the dashed BG-path.

Figure 6: A chain \(C_i \not\subseteq S_l\) of type 3.

**Lemma 21** (aka **Lemma 10**). Let \(C_k\) be the parent of a chain \(C_i \neq C_0\).

- If \(C_i\) is not of type 1, \(C_k \neq C_0\) and \(t(C_i)\) is an inner vertex of \(C_k\).

- Let \(C_k\) but not \(C_i\) be contained in \(S_l^R\). If \(C_i\) is either of type 1 with an inner real vertex in \(t(C_i) \rightarrow_T s(C_i)\) or of type 3a, \(C_i\) is a BG-path for \(S_l^R\) preserving \((R_1)\) and \((R_2)\).

**Proof.** Assume to the contrary that \(C_i\) is not of type 1 and \(C_k = C_0\). Because \(t(C_i)\) is contained in \(C_0\), \(s(C_i)\) must be in \(C_0\) as well. But then \(C_i\) would be of type 1, since \(t(C_i) \rightarrow_T s(C_i) \subseteq C_0\). Therefore, if \(C_i\) is not of type 1, \(C_k \neq C_0\) holds and \(C_k\) must start with a backedge. Then the definition of the parent relation implies that \(t(C_i)\) is an inner vertex of \(C_k\).

For the second claim, let \(C_i\) first be of type 3a. Since \(S_l\) is upwards-closed, modular and contains \(C_k\), \(C_i\) satisfies the property 1.1 of BG-paths. In addition, \(s(C_i) \neq s(C_k)\) holds by definition and with \(C_k < C_i\), \(s(C_i)\) must be an inner vertex of the path \(t(C_k) \rightarrow_T s(C_k)\) (see Figure 6). Therefore, the only chains \(C_j\) that contain \(s(C_i)\) and \(t(C_i)\) are different from \(C_0\) and fulfill \(C_i \cap C_j = \{s(C_i), t(C_i)\} = \{s(C_j), t(C_j)\}\). This implies \(C_i\) having property 1.2. Using Lemma 20, \(C_i\) is a BG-path for \(S_l\) that preserves \((R_1)\) and \((R_2)\).

If \(C_i\) is of type 1, property 1.1 follows from the same argument as before. Additionally, the inner real vertex in \(t(C_i) \rightarrow_T s(C_i)\) prevents any link containing \(s(C_i)\) and \(t(C_i)\) from having \(s(C_i)\) or \(t(C_i)\) as an inner vertex and therefore ensures property 1.2. Lemma 20 implies that \(C_i\) is a BG-path for \(S_l\) and \(C_i\) must preserve \((R_1)\) and \((R_2)\), the latter due to the inner real vertex in \(t(C_i) \rightarrow_T s(C_i)\).

**Proposition 22.** Every caterpillar \(L_j\) consists of exactly one chain of type 3b, namely the chain \(C_j\), and one or more chains of type 2b.
Lemma 23. $C \setminus \{C_0\}$ is partitioned into single chains of types 1, 2a, 3a and the chains being contained in caterpillars. Moreover, no chain is contained in two caterpillars.

Proof. With Proposition 22, it remains to show that every chain $C_i$ of type 2b or 3b is contained in exactly one caterpillar. If $C_i$ is of type 3b, $C_i$ is part of the caterpillar $L_i$ (see Algorithm 1, line 13) and will not be assigned to a second caterpillar afterwards, as it is not marked. Otherwise, $C_i$ is of type 2b and was therefore marked. We show that, after all chains in $C$ have been classified, $C_i$ is not marked anymore. This forces $C_i$ to be contained in exactly one caterpillar, as the only way to unmark chains is to assign them to a caterpillar (see Algorithm 1, line 15) and no chain is marked twice.

Let $C_k$ be the parent of $C_i$. Because $C_i$ is of type 2b, $s(C_i) = s(C_k)$ holds and $C_i$ is not a backbone, implying that the last edge $e$ of $C_i$ is in $T$. Let $x$ be the end vertex of $e$ different from $t(C_i)$. Using Lemma 21, $C_k \neq C_0$ holds and $t(C_i)$ is an inner vertex of $C_k$. Then at least one backbone $vw$ with $v \notin \{s(C_i), t(C_i)\}$ must enter $T(x)$, since otherwise $s(C_i)$ and $t(C_i)$ would be a separation pair of $G$. Let $C_j$ be the minimal chain with respect to $< $ that contains such a backbone.

As $C_j > C_i$ holds, the vertex $v$ is an inner vertex of $t(C_i) \rightarrow_T s(C_i)$, implying that $C_j$ is not of type 2. In addition, $C_j$ is not of type 1, since $t(C_j) \rightarrow_T v$ contains edges from $C_i$ and $C_k$. At the time $C_j$ is found in the chain decomposition, every chain that already ends at a vertex in $T(x)$ starts at $s(C_i)$ and is therefore of type 2a or 2b. Since chains that are backedges cannot have children, the parent of $C_j$ is marked and $C_j$ is of type 3b. Moreover, every chain corresponding to an inner vertex of the path $C_j \rightarrow_U C_i$ is marked. This concludes $C_i$ to become unmarked due to line 15 of Algorithm 1.

The following gives the detailed proof of Lemma 12.

Lemma 24 (aka Lemma 12). Let $L_j$ be a caterpillar that consists of $t$ chains and has parent $C_k$. Let $C_k$ but no chain in $L_j$ be contained in $S_i^R$. Then, if and only if $L_j$ is good, the chains in $L_j$ can be efficiently decomposed into $t$ successive BG-paths satisfying $(R_1)$ and creating subdivisions $S_{i+1}, S_{i+2}, \ldots, S_{i+t-1}, S_i^R$, each of which satisfies $(R_2)$.

Proof. Let $L_j$ be good and let $y$ be the last vertex of the minimal chain in $L_j$, thus $y \in V(C_k)$. We assume at first that $s(C_j)$ is a proper ancestor of $t(C_k)$ (see Figure 3(b)). Then the path $P = C_j \cup (t(C_j) \rightarrow_T y)$ fulfills properties 1.1 and 1.2 and is a BG-path for $S_j$ with Lemma 20. Adding $P$ preserves $S_j$ to be upwards-closed but not modular. The restriction $(R_2)$ is also preserved, as $t(C_k)$ is real and, for $S_i = S_3$, $C_k$ must be either $C_1$ or $C_2$, implying that $s(P)$ becomes an inner real vertex of $C_0$. Successively, for each chain $C_i$ of the $t-1$ chains in $L_i \setminus \{C_j\}$, we now add $C_i \setminus P$, which is a BG-path yielding an upwards-closed subdivision for analogue reasons.

Now assume that $s(C_j)$ is a descendant of $t(C_k)$ (see Figure 3(a)). Then $s(C_j) \in V(C_k)$ and since $L_j$ is good, there is a real vertex $a$ strictly between $s(C_j)$ and $s(C_k)$ in $C_k$. We first show that $t(C_k) \rightarrow_T s(C_k)$ contains an inner real vertex as well. Assume the contrary. Then $C_k$ must be of type 1 and has been added before, contradicting restriction $(R_2)$ unless $S_i = S_3$. But $S_i$ must be different from $S_3$, since $a$ exists, and it follows that $t(C_k) \rightarrow_T s(C_k)$ contains an inner real vertex $b$. Let $C_b$ be the parent of $C_j$. Then $(C_j \cup C_b) \setminus (t(C_j) \rightarrow_T y) \setminus \{t(C_j)\})$ is a BG-path due to the real vertices $a$ and $b$ and we add it, although it neither preserves $S_{i+1}$ to be upwards-closed nor modular. We next add $t(C_j) \rightarrow_T y$, which restores upwards-closedness. The resulting subdivisions $S_{i+1}$ and $S_{i+2}$ both satisfy $(R_2)$, as $b$ is real in $S_{i+1}$ and $S_{i+2}$ and $y$ is real in $S_{i+2}$. We proceed with adding successively paths, namely for each chain $C_i$ of the $t-2$ remaining chains in $L_i \setminus \{C_j, C_b\}$ the path $C_i \setminus (t(C_j) \rightarrow_T y)$. With the same line of argument, these paths obtain upwards-closed subdivisions $S_{i+3}, \ldots, S_{i+t}$, each of which satisfies $(R_2)$. 

14
to properties 1.1 and 1.2, must hold and $C_s$ follows. Then $L$ in $L$ closed. All paths can be computed in time linearly dependent on the total number of edges in $L$.

For the only if part, let $P_1$ and $P_2$ be the first two BG-paths in a decomposition of the chains in $L_j$; these exist, since $t > 1$ holds in every caterpillar. Let $L_j$ be bad, as otherwise the claim follows. Then $s(C_j) \in V(C_k)$. We show that $L_j$ cannot be bad, as $S_t$ contains a real vertex in $C_k$ strictly between $s(C_j)$ and $s(C_k)$. Because of properties 1.1 and 1.2, $P_1 \cap S_t = \{s(C_k), s(C_j)\}$ must hold and $P_1$ is a link of $S_{t+1}$ being parallel to $s(C_j) \rightarrow C_k s(C_k)$. Since only the chain of type 3b in $L_j$ starts at $s(C_j)$, both end vertices of $P_2$ must be different from $s(C_j)$. Then, due to properties 1.1 and 1.2, $P_2$ joins inner vertices of the parallel links $P_1$ and $s(C_j) \rightarrow C_k s(C_k)$ in $S_{t+1}$, contradicting property 1.3, as $|V_{\text{real}}(S_{t+1})| \geq 4$.

**Lemma 25.** Let $C_i$ be a chain of type 3 such that $s(C_i) \in V(S^R_t)$, $C_i \not\subset S^R_t$ and $C_i$ is minimal among the chains of type 3 in its segment $H$. Let $D_1 > \ldots > D_k$ be all ancestors of $C_i$ in $H$ with $D_1 = C_i$ and $D_k$ being the minimal chain in $H$. Then all chains $D_k, \ldots, D_1$ can be successively added as BG-paths preserving $(R_1)$ and $(R_2)$ (possibly being part of caterpillars), unless one of the following exceptions holds:

1. $C_i$ is of type $3a$, $k = 2$, $D_k$ is of type 1, $s(C_i)$ is an inner vertex of $t(D_k) \rightarrow T s(D_k)$ and there is no inner real vertex in $t(D_k) \rightarrow T s(D_k)$ (Figure 7(a)),

2. $C_i$ is of type $3b$, $D_k$ is of type $2b$ and $L_i = \{D_1, \ldots, D_k\}$ with $L_i$ being bad (Figure 7(b)),

3. $C_i$ is of type $3b$, $L_i = \{D_1, \ldots, D_{k-1}\}$, $D_k$ is of type 1, $s(C_i)$ is an inner vertex of $t(D_k) \rightarrow T s(D_k)$ and there is no inner real vertex in $t(D_k) \rightarrow T s(D_k)$ (Figure 7(c)).

**Proof.** Let $D \in \{D_2, \ldots, D_k\}$. Then $D$ is not of type 3 by assumption and not of type 2a, as chains of that type cannot have children. Assume that $D$ is of type $2b$ and let $L_j$ be the caterpillar containing $D$ due to Lemma 23. If $C_j \neq C_i$, $C_j < C_i$ holds, as otherwise $C_j$ would
Figure 8: After \( C_i \) was added, the next minimal chain \( C_j \) is in no exception.

not be the chain of type 3b in \( L_j \). But then \( C_j \) contradicts the minimality of \( C_i \), since \( C_j \) is not contained in \( S_l \) and of type 3b. We conclude that every chain in \( \{D_2, \ldots, D_k\} \) of type 2b is contained in \( L_i \) and forces \( C_i \) to be of type 3b. This is used in the following case distinction.

Let \( C_i \) be of type 3a. If \( k = 1 \), \( C_i \) is a BG-path for \( S_l \) with Lemma 21 and the claim follows. Otherwise, \( k > 1 \) and all chains in \( \{D_2, \ldots, D_k\} \) are of type 1. Then \( s(D_2) \) is a proper ancestor of \( s(C_i) \), since \( D_2 < C_i \) and \( C_i \) is not of type 2. Moreover, \( s(C_i) \) is a proper ancestor of \( t(D_2) \), because otherwise \( H \cap S_l = \{s(D_2), t(D_2)\} \) is a separation pair of \( G \) due to the minimality of \( C_i \). It follows that \( s(C_i) \) is an inner vertex of \( t(D_2) \rightarrow_T s(D_2) \). If \( k > 2 \), \( D_3 \) must contain \( t(D_2) \rightarrow_T s(D_2) \), because \( D_2 \) is of type 1 and a child of \( D_3 \). Therefore, the edge \( e \) joining \( s(C_i) \) with the parent of \( s(C_i) \) in \( T \) is contained in \( D_3 \). But since \( S_l \) is upwards-closed, \( e \) is also contained in \( S_l \), contradicting that \( D_3 \not\subseteq S_l \). Thus, \( k = 2 \). If \( t(D_2) \rightarrow_T s(D_2) \) contains an inner real vertex, \( D_2 \) and \( C_i \) can be subsequently added as BG-paths with Lemma 21, otherwise 25.1 is satisfied.

Let \( C_i \) be of type 3b. Then all chains in \( \{D_2, \ldots, D_k\} \) that are of type 2b must be contained in \( L_i \). Since every caterpillar \( L_j \) contains the parent of the chain \( C_j \) and since \( S_l \) contains no child in \( L_i \) due to \( (R_1) \), \( k > 1 \) holds and \( D_2 \) is of type 2b with \( D_2 \in L_i \). Let \( D_t \) with \( 1 < t < k \) be the minimal chain in \( L_i \). If \( t = k \) and \( L_t \) is good, all chains in \( L_i \) can be decomposed to BG-paths according to Lemma 12. If \( t = k \) and \( L_t \) is bad, 25.2 is satisfied. Only the case \( k > t \) remains. Then \( D_{t+1} \) is of type 1 and, using the same arguments as in the case for type 3a, \( s(C_i) \) is an inner vertex of \( t(D_k) \rightarrow_T s(D_k) \) and \( k = t + 1 \). If \( t(D_k) \rightarrow_T s(D_k) \) contains an inner real vertex, Lemmas 21 and 12 imply that \( D_k \) and \( L_i \) can be iteratively added as set of successive BG-paths, preserving \( (R_1) \) and \( (R_2) \). Otherwise, 25.3 is satisfied.

We extend Lemma 25 to non-minimal chains of type 3.

**Lemma 26.** Let the preconditions of Lemma 25 hold. If \( C_i \) is not contained in one of the exceptions 25.1-25.3 (as \( C_i \)), the chains of type 3 in \( H \) that start in \( S_l^T \) and their ancestors in \( H \) can be successively added as BG-paths (in reversed order), preserving \( (R_1) \) and \( (R_2) \).

**Proof.** Using Lemma 25, we add the chains \( C_i, D_2, \ldots, D_k \) in \( H \) as BG-paths. This partitions \( H \) into new segments; let \( H' \subseteq H \setminus \{C_i, D_2, \ldots, D_k\} \) be such a new segment. If \( H' \) does not
contain chains of type 3 that start in $S_l$, the claim follows for such chains in $H'$. Otherwise, let $C_j$ be the minimal chain of type 3 in $H'$ that starts in $S_l$ and let $C_j \supset D'_2 > \ldots > D'_k$ be its ancestors in $H'$. We show that $C_j$ is not contained in one of the exceptions 25.1-25.3 and can therefore be added as BG-path with Lemma 25, along with its proper ancestors in $H'$. First, assume to the contrary that $C_j$ is contained in exception 25.1 or 25.3 (see Figure 8(a)). Because $D'_k$ is a proper descendant of $D_k$ and $D'_k$ is of type 1, $s(C_j) \in V(S_l)$ cannot be an inner vertex of $t(D'_k) \rightarrow_T s(D'_k)$, contradicting the assumption. Now assume to the contrary that $C_j$ is contained in exception 25.2 (see Figure 8(b)). Then $C_j$ is of type $3b$ and part of a bad caterpillar $L_j$, whose parent $D$ is not contained in $H'$. Because $L_j$ contains only chains in $H'$, $D$ must be a descendant of $D_k$ and is therefore contained in $H \setminus H'$. Since $L_j$ is bad, $s(C_j)$ is contained in $S_l \cap D$ and it follows with $s(C_j) \neq s(D)$ that $D$ must end in $S_l$ at the vertex $s(C_j)$. As $D_k$ is the only chain in $H$ that ends in $S_l$, $D = D_k$ must hold. But this contradicts $L_j$ being bad, as $D$ contains the inner real vertex $s(D_{k-1})$. Thus, $C_j$ and its ancestors in $H'$ can be added, partitioning $H'$ into smaller segments. Iterating the same argument for these segments establishes the claim for all chains of type 3 in $H$ that start in $S_l$. 

The next lemma shows that the only chains of type 1 that cannot be added are either backedges or are contained as $D_k$ in exceptions 25.1-25.3.

**Lemma 27.** Let $C_j$ be a chain in $S_i^R$ and let $D_k$ be a child of $C_j$ that is of type 1 and not in $S_i^R$. If $D_k$ is not a backedge, there is a chain of type 3 in the segment containing $D_k$ that starts in $t(D_k) \rightarrow_T s(D_k) \subset C_j$. If $D_k$ is neither a backedge nor contained (as $D_k$) in the exceptions 25.1 and 25.3, $D_k$ can be added as BG-path.

**Proof.** Let $H$ be the segment of $D_k$ and assume that $D_k$ is not a backedge. We first show that $H$ contains a chain of type 3 that starts in $t(D_k) \rightarrow_T s(D_k)$. We can assume that $D_k$ is not contained in the exceptions 25.1 and 25.3, as then $H$ would contain such a chain by definition. Let $x$ be the last but one vertex in $D_k$. Since $G$ is 3-connected, there is a minimal chain $C_i$ entering $T(x)$ such that $s(C_i)$ is an inner vertex of $t(D_k) \rightarrow_T s(D_k)$, as otherwise the inner vertices of $D_k$ would be separated by $\{s(D_k), t(D_k)\}$. By definition of the chain decomposition, $C_i$ must be of type $3a$ or $3b$. Because $D_k$ is not contained in exceptions 25.1 and 25.3 and $H$ cannot contain exception 25.2, Lemma 25 can be applied on $C_i$, obtaining the last claim.

Here we give the proof of the key theorem (**Theorem 15**) of the paper.

**Theorem 28 (aka Theorem 15).** For a subdivision $S_l^R$, let $C_i$ be a chain such that $\text{Children}_{12}(C_i) = \text{Type}_3(C_j) = \emptyset$ holds for every proper ancestor $C_j$ of $C_i$. Then all chains in $\text{Children}_{12}(C_i) \cup \text{Type}_3(C_i)$ and their proper ancestors that are not already contained in $S_l^R$ can be successively added as BG-paths (possibly being part of caterpillars) such that $(R_1)$ and $(R_2)$ is preserved. Moreover, the chains in $\text{Type}_3(C_i)$ that are contained in segments in which the minimal chain is not contained in $\text{Children}_{12}(C_i)$ can be added at any point in time in arbitrary order (together with their proper ancestors that are not contained in $S_l^R$).

**Proof.** By assumption, $C_i$ is contained in $S_l$. Let $D \not\subset S_l$ be a child of $C_i$. If $C_i = C_0$, $D$ must be of type 1. Let $C_i \neq C_0$. Then $D$ cannot be of type $3b$, as otherwise it would be contained in $S_l$ due to $(R_1)$ and $C_i \subset S_l$. It is neither of type $3a$, since in that case $s(D)$ is contained in a proper ancestor of $C_i$, implying $D \subset S_l$ by assumption. We conclude that $D$ is of type 1 or 2 and focus on the cases where $D$ cannot be added. Let $P$ be the path on which $D$ depends on. If $D$ is of type 1, $P$ does not contain an inner real vertex, as otherwise $D$ can be added as BG-path due to Lemma 21. With Lemma 27, $D$ must be either a backedge or be contained as the minimal chain in exception 25.1 or 25.3. If $D$ is of type $2a$, $s(C_i)$ is real and neither $t(D)$ nor an inner
vertex in $P$ can be real, since otherwise $D$ can be added as BG-path, preserving $(R_1)$ and $(R_2)$. If $D$ is of type 2b, $D$ is the minimal chain of a caterpillar $L_a$ with parent $C_i$. According to Lemma 12, $L_a$ is bad and, thus, corresponds to exception 25.2. The following is a list of the possible cases for which a child $D$ of $C_i$ is not added.

1. $D$ is of type 1 without an inner real vertex in $P$ and either a backedge or the minimal chain in exception 25.1 or 25.3

2. $C_i \neq C_0$ and $D$ is of type 2a without a real vertex in $P \setminus \{s(D)\}$

3. $C_i \neq C_0$ and $D$ is of type 2b without an inner real vertex in $P$ ($D$ is the minimal chain in exception 25.2)

We iteratively add all chains $D$ in $X' \cup Y'$ that do not satisfy one of the above three cases 15.1-15.3 for $D \in X'$ and whose segments do not contain one of the exceptions 25.1-25.3 for $D \in Y'$ (the latter followed by adding the proper ancestors in the segment of $D$ according to Lemma 26). Let $X$ be the set of remaining chains in $X'$ and let $Y$ be the set of remaining chains in $Y'$. If $X = \emptyset$, $Y = \emptyset$ holds as well, as otherwise the minimal chain in the segment containing one of the exceptions 25.1-25.3 is a child of $C_i$, contradicting $X = \emptyset$. This implies the claim for $X = \emptyset$.

We prove the theorem by showing that $X = \emptyset$ must hold. Assume to the contrary that $X \neq \emptyset$ and let $S_t$ be the current subdivision (all segments will be dependent on $S_t$). Then $C_i$ must contain a link $L$ of length at least two, because the dependent path $P$ in each of the cases 15.1-15.3 is in $C_i$ and contains a non-real vertex due to the 3-connectivity and simpleness of $G$. According to Lemma 4, $L$ contains an inner vertex $v$ on which a BG-path $B$ starts (not necessarily being a chain and not necessarily preserving $(R_1)$ or $(R_2)$). Let $e$ be the first edge of $B$. Then $e$ is not contained in the segment of any $x \in X$, as otherwise $B$ would not have property 1.2, because $v$ is non-real and all start vertices of the chains in the segment of $x$ that are in $S_t$ are contained in $L$. Thus, $C(e)$ cannot be a child of $C_i$ and it follows that $s(C(e)) = v$. In particular, $C(e)$ is not of type 1.

The segment of $e$ cannot contain a chain of type 3 that starts in $C_i$, as it otherwise contains a chain $x \in X$ of type 1 or 2b due to exceptions 25.1-25.3, contradicting the previous argument. In particular, $C(e)$ is not of type 3 and the only remaining case is that $C(e)$ is of type 2.

Let $C_k$ be the maximal ancestor of $C(e)$ that is not of type 2. Then $s(C_k) = v$ holds by construction of the chain decomposition and $C_k$ must be contained in the segment of $C(e)$ due to $(R_1)$, $(R_2)$ and $v$ being non-real. Since the segment of $e$ cannot contain a chain of type 3 that starts in $C_i$, $C_k$ must be of type 1. But then, as $v$ is an inner vertex of $L$, $C_k$ must be a child of $C_i$, contradicting that $e$ is not contained in the segment of any $x \in X$. This is a contradiction to the existence of $B$ and it follows that $X = \emptyset$, which implies the claim.

\[\Box\]

References

[1] S. Albroscheit. Ein Algorithmus zur Konstruktion gegebener 3-zusammenhängender Graphen (in German). Diploma thesis, Freie Universität Berlin, 2006.

[2] D. W. Barnette and B. Grünbaum. On Steinitz’s theorem concerning convex 3-polytopes and on some properties of 3-connected graphs. Many Facets of Graph Theory, Lecture Notes in Mathematics, 110:27–40, 1969.

[3] V. Batagelj. Inductive classes of graphs. In Proceedings of the 6th Yugoslav Seminar on Graph Theory, pages 43–56, Dubrovnik, 1986.
[4] V. Batagelj. An improved inductive definition of two restricted classes of triangulations of the plane. In Combinatorics and Graph Theory, Banach Center Publications, volume 25, pages 11–18. PWN Warsaw, 1989.

[5] G. D. Battista and R. Tamassia. On-line graph algorithms with SPQR-trees. In Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP’90), pages 598–611, 1990.

[6] P. Bertolazzi, G. D. Battista, C. Mannino, and R. Tamassia. Optimal upward planarity testing of single-source digraphs. SIAM J. Comput., 27(1):132–169, 1998.

[7] M. Blum and S. Kannan. Designing programs that check their work. In Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC’89), pages 86–97, New York, 1989.

[8] N. Chiba and T. Nishizeki. The Hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs. J. Algorithms, 10(2):187–211, 1989.

[9] A. Elmasry, K. Mehlhorn, and J. M. Schmidt. A linear-time certifying triconnectivity algorithm for Hamiltonian graphs, preprint available at http://www.mpi-inf.mpg.de/~mehlhorn/ftp/Hamilton.pdf.

[10] Z. Galil and G. F. Italiano. Reducing edge connectivity to vertex connectivity. SIGACT News, 22(1):57–61, 1991.

[11] M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar Hamiltonian circuit problem is NP-complete. SIAM J. Comput., 5(4):704–714, 1976.

[12] C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. In Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’01), pages 246–255, 2001.

[13] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. SIAM J. Comput., 2(3):135–158, 1973.

[14] E. L. Johnson. A proof of the four-coloring of the edges of a regular three-degree graph. Technical report, O. R. C. 63-28 (R. R.) Min. Report, Univ. of California, Operations Research Center, 1963.

[15] D. Kratsch, R. M. McConnell, K. Mehlhorn, and J. P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. SIAM J. Comput., 36(2):326–353, 2006. Preliminary version in SODA 2003, pp. 158–167.

[16] K. Mehlhorn and S. Näher. From algorithms to working programs: On the use of program checking in LEDA. In Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science (MFCS’98), pages 84–93, 1998.

[17] K. Mehlhorn, S. Näher, M. Seel, R. Seidel, T. Schilz, S. Schirra, and C. Uhrig. Checking geometric programs or verification of geometric structures. Comput. Geom. Theory Appl., 12(1-2):85–103, 1999.

[18] P. Mutzel. The SPQR-tree data structure in graph drawing. In Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP’03), pages 34–46, 2003.

[19] S. Olariu and A. Y. Zomaya. A time- and cost-optimal algorithm for interlocking sets – With applications. IEEE Trans. Parallel Distrib. Syst., 7(10):1009–1025, 1996.

[20] J. M. Schmidt. Construction sequences and certifying 3-connectedness. In Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS’10), Nancy, France, pages 633–644, 2010.

[21] C. Thomassen. Kuratowski’s theorem. Journal of Graph Theory, 5(3):225–241, 1981.

[22] C. Thomassen. Reflections on graph theory. Journal of Graph Theory, 10(3):309–324, 2006.

[23] W. T. Tutte. A theorem on planar graphs. Trans. Amer. Math. Soc., 82:99–116, 1956.

[24] W. T. Tutte. A theory of 3-connected graphs. Indag. Math., 23:441–455, 1961.
[25] W. T. Tutte. Connectivity in graphs. In *Mathematical Expositions*, volume 15. University of Toronto Press, 1966.

[26] K.-P. Vo. Finding triconnected components of graphs. *Linear and Multilinear Algebra*, 13:143–165, 1983.

[27] K.-P. Vo. Segment graphs, depth-first cycle bases, 3-connectivity, and planarity of graphs. *Linear and Multilinear Algebra*, 13:119–141, 1983.