Constructive Geometric Constraint Solving

Antoni Soto i Riera

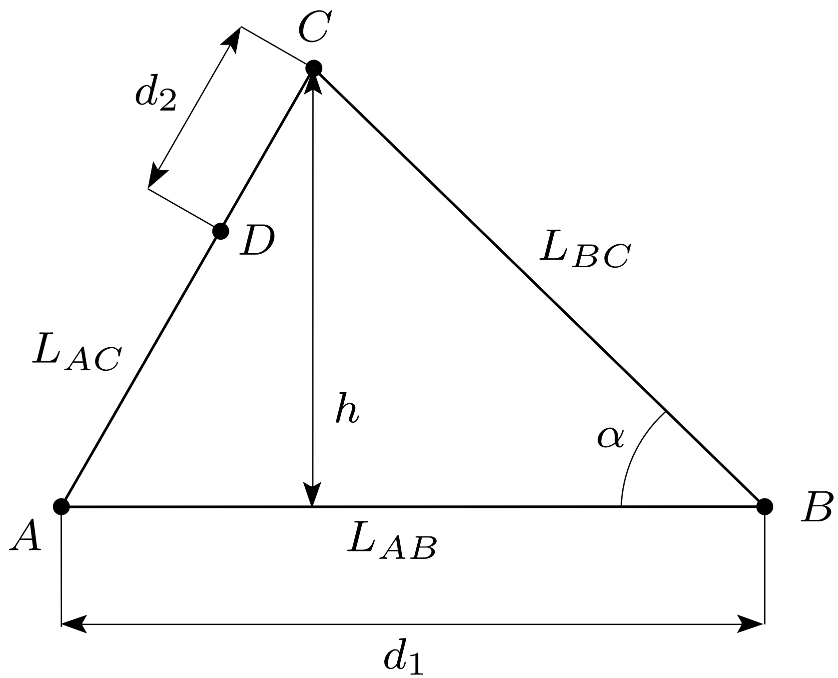
Departament de Llenguatges i Sistemes Informàtics

Universitat Politècnica de Catalunya

Barcelona, September 2002

Preliminaries

Geometric constraint problem



A *geometric constraint problem* consists of

- a set of geometric elements, $\{A, B, C, D, L_{AB}, L_{AC}, L_{BC}\}$,
- a set of geometric constraints defined between them, and
- a set of parameters, $\{d_1, d_2, \alpha, h\}$.

Geometric constraint solving

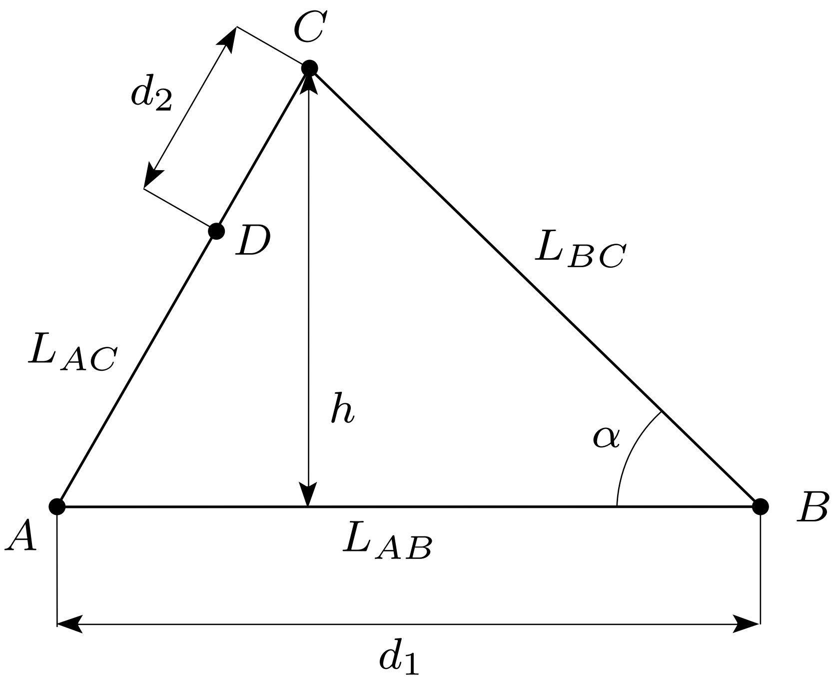
A geometric constraint problem can be represented by a predicate φ in first order logic.

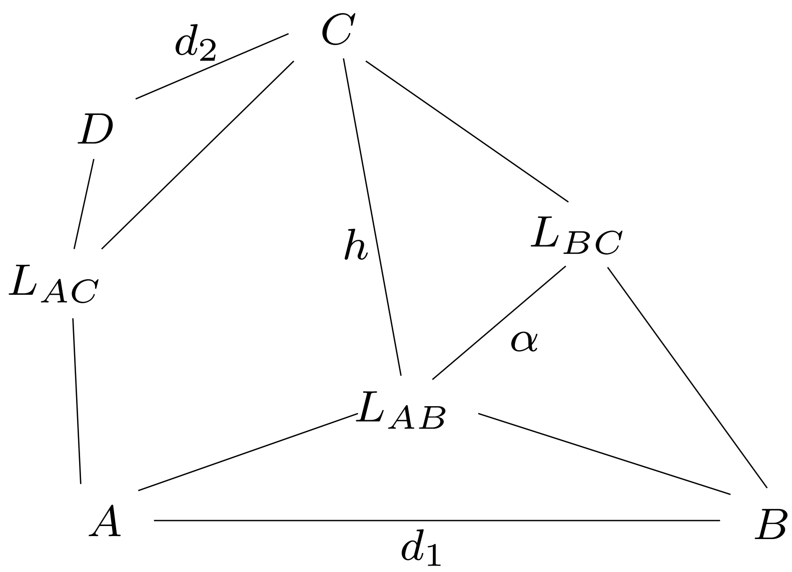
$$\begin{aligned} &\varphi(A, B, C, D, L_{AB}, L_{AC}, L_{BC}) \\ &\quad \equiv \quad \mathrm{d}(A, B) = d_1 \wedge \mathrm{on}(A, L_{AB}) \wedge \mathrm{on}(B, L_{AB}) \wedge \mathrm{on}(A, L_{AC}) \wedge \\ &\qquad\quad \mathrm{on}(C, L_{AC}) \wedge \mathrm{on}(D, L_{AC}) \wedge \mathrm{on}(B, L_{BC}) \wedge \mathrm{on}(C, L_{BC}) \wedge \\ &\qquad\quad \mathrm{h}(C, L_{AB}) = h \wedge \mathrm{a}(L_{AB}, L_{BC}) = \alpha \wedge \mathrm{d}(C, D) = d_2 \end{aligned}$$

Geometric constraint solving consists in proving the truth of the existentially quantified predicate φ that represents the geometric constraint problem.

$$\exists A \; \exists B \; \exists C \; \exists D \; \exists L_{AB} \; \exists L_{AC} \; \exists L_{BC} \quad \varphi(A, B, C, D, L_{AB}, L_{AC}, L_{BC})$$

Geometric Constraint Graph



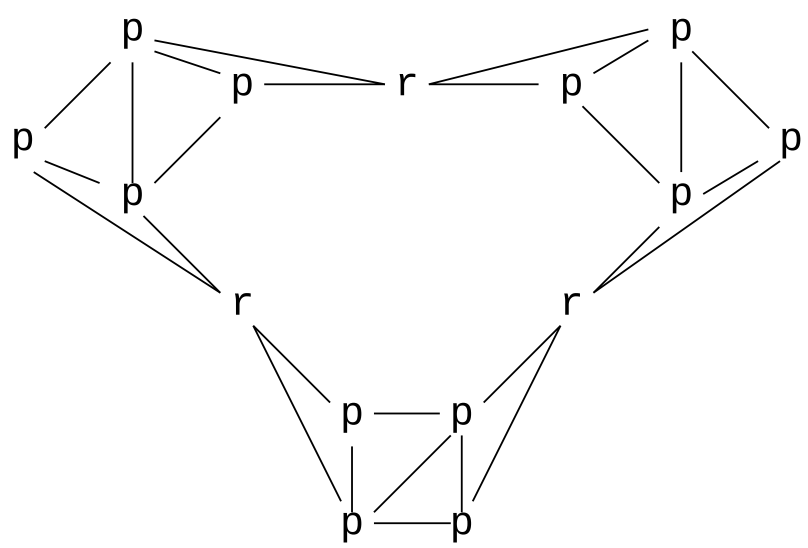


A geometric constraint problem can also be represented by means of a *geometric constraint graph* $G = (V, E)$ where the nodes in V are geometric elements with two degrees of freedom and the edges in $E \subseteq V \times V$ are geometric constraints such that each of them cancels one degree of freedom.

Well-constrained graphs

Theorem 1 (Laman, 1970) *Let $G = (P, D)$ be a geometric constraint graph such that the vertices in P are points in the two-dimensional Euclidean space and the edges in $D \subseteq P \times P$ are distance constraints. G is* generically well-constrained *if and only if for all $G' = (P', D')$, induced subgraph of G by the set of vertices $P' \subseteq P$,*

1. $|D'| \leq 2\,|P'| - 3$, and
2. $|D| = 2\,|P| - 3$.



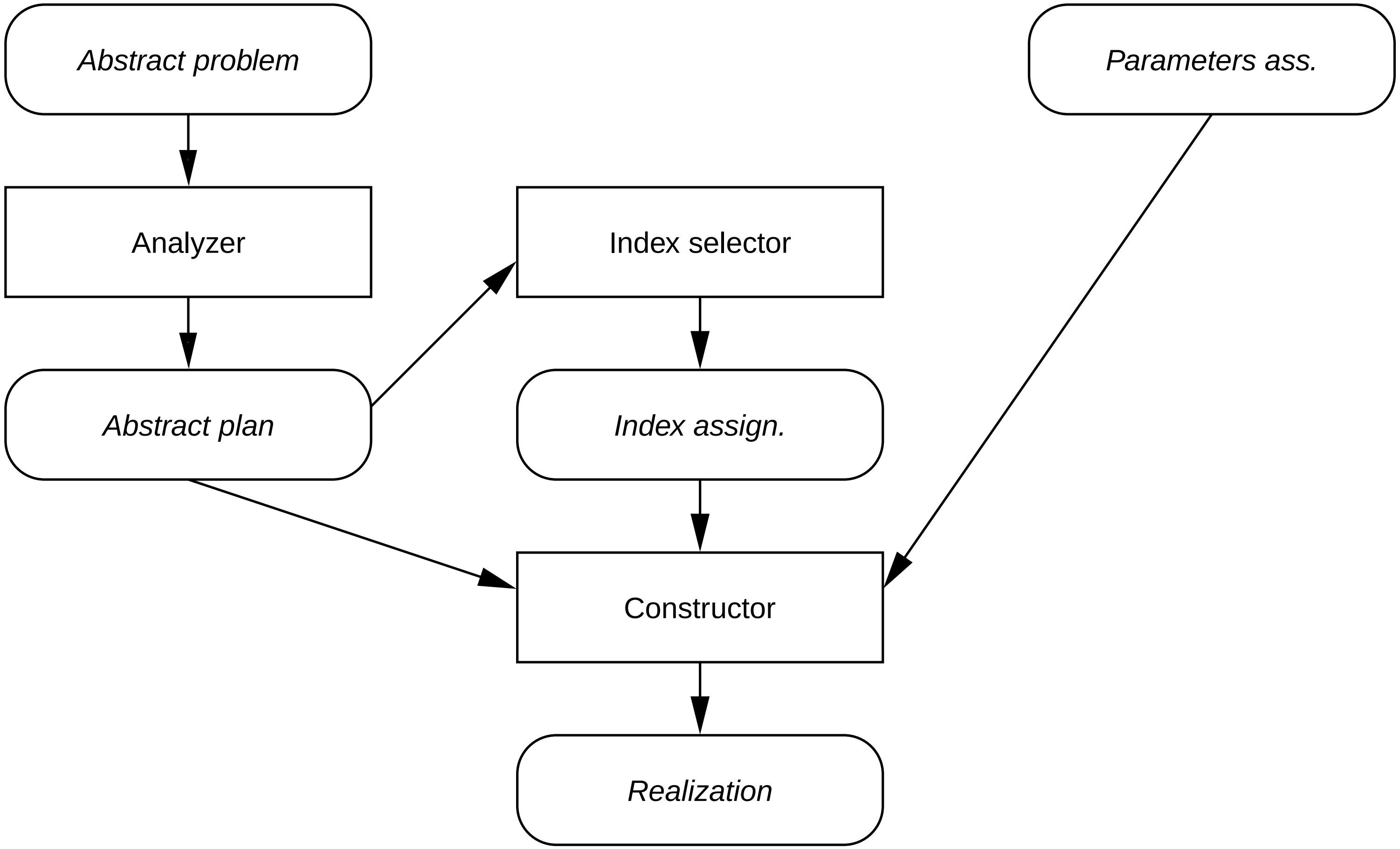
Structurally well-constrained graphs

A necessary condition for a geometric constraint problem to be solvable is that the associated constraint graph must be structurally well-constrained. Let $G = (V, E)$ be a geometric constraint graph.

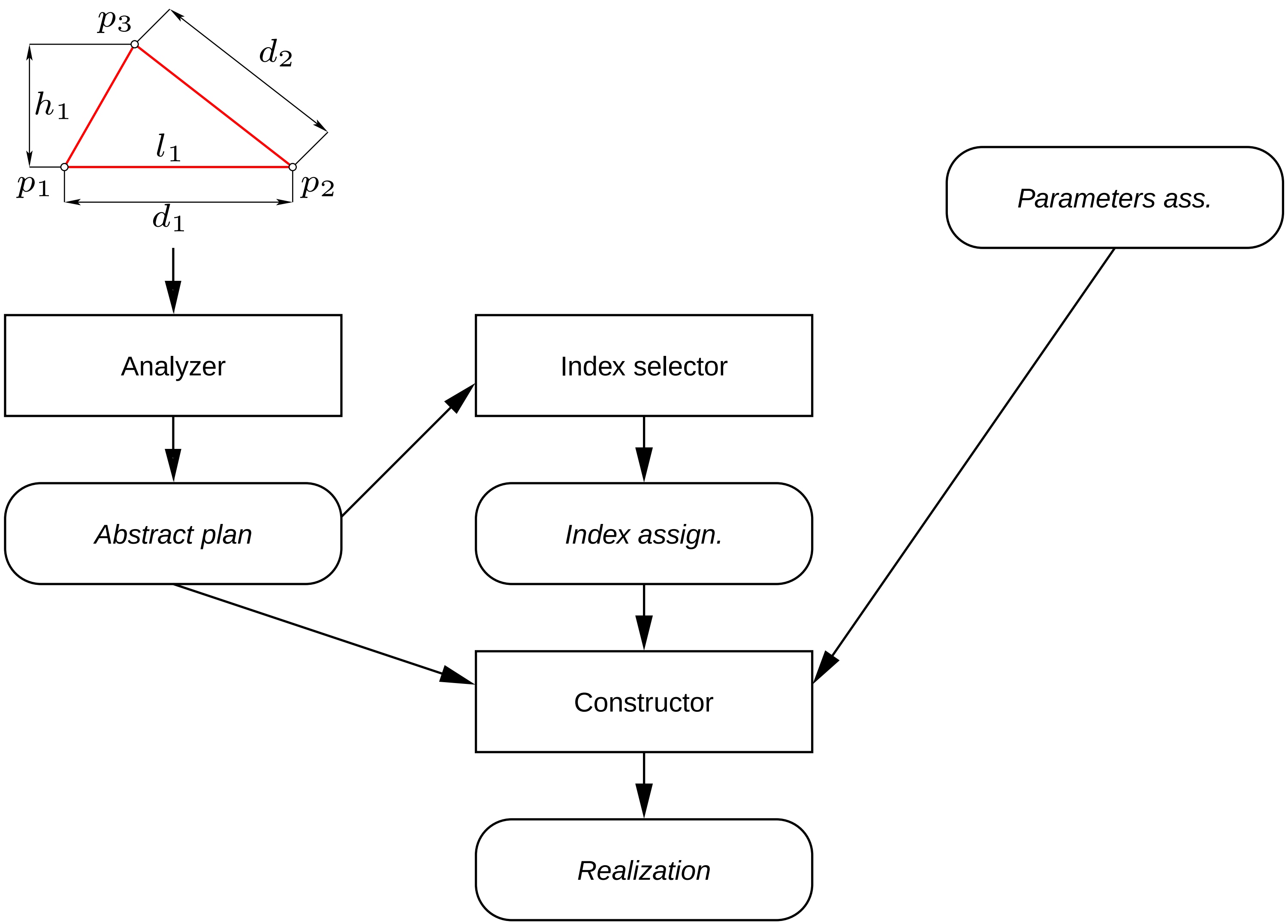
1. G is *structurally over-constrained* if there is an induced subgraph with $m \leq |V|$ nodes and more than $2m - 3$ edges.
2. G is *structurally under-constrained* if it is not structurally over-constrained and $|E| < 2\,|V| - 3$.
3. G is *structurally well-constrained* if it is not structurally over-constrained and $|E| = 2\,|V| - 3$.

Constructive Geometric Constraint Solvers

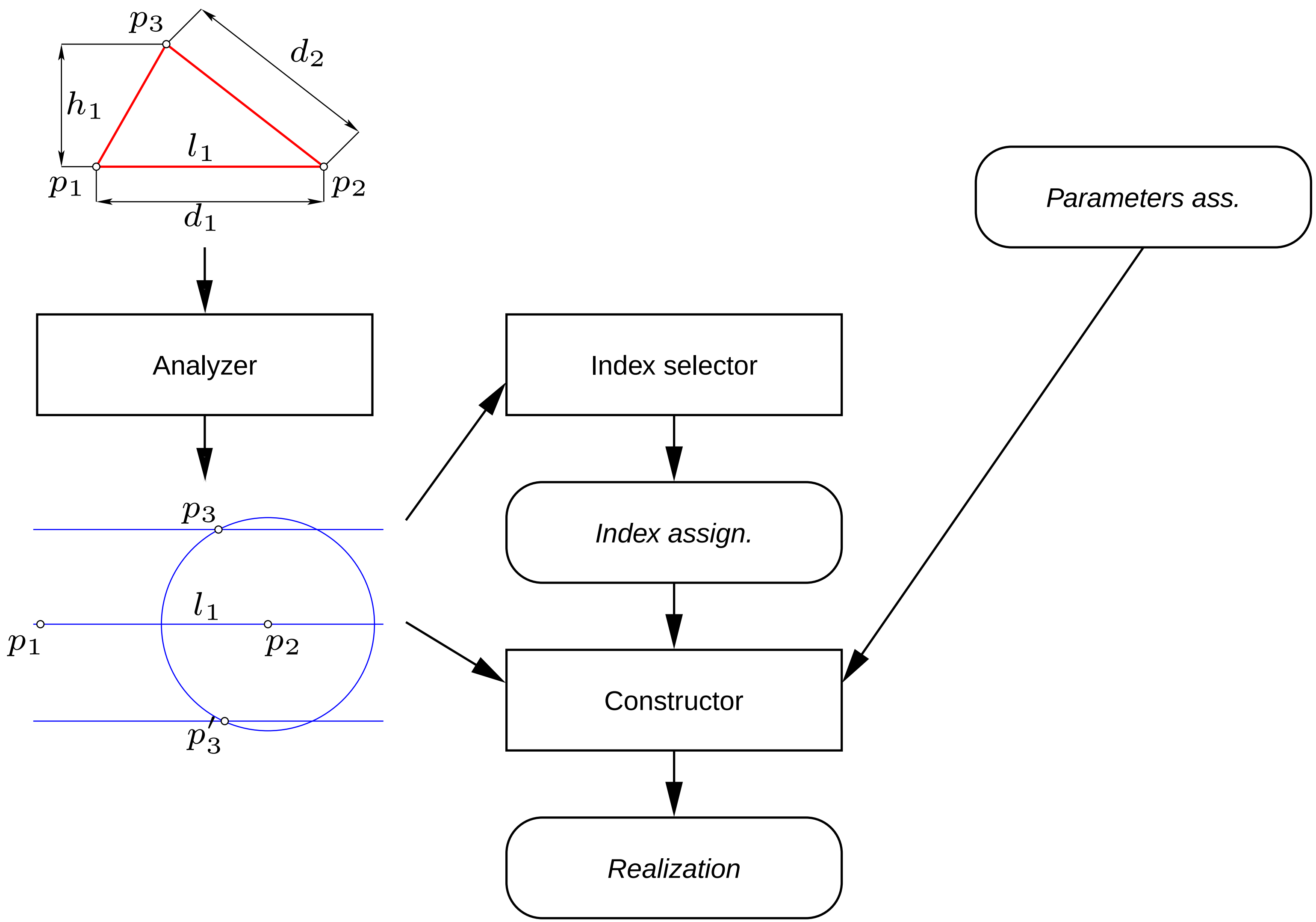
Architecture for Constructive Geometric Constraint Solvers



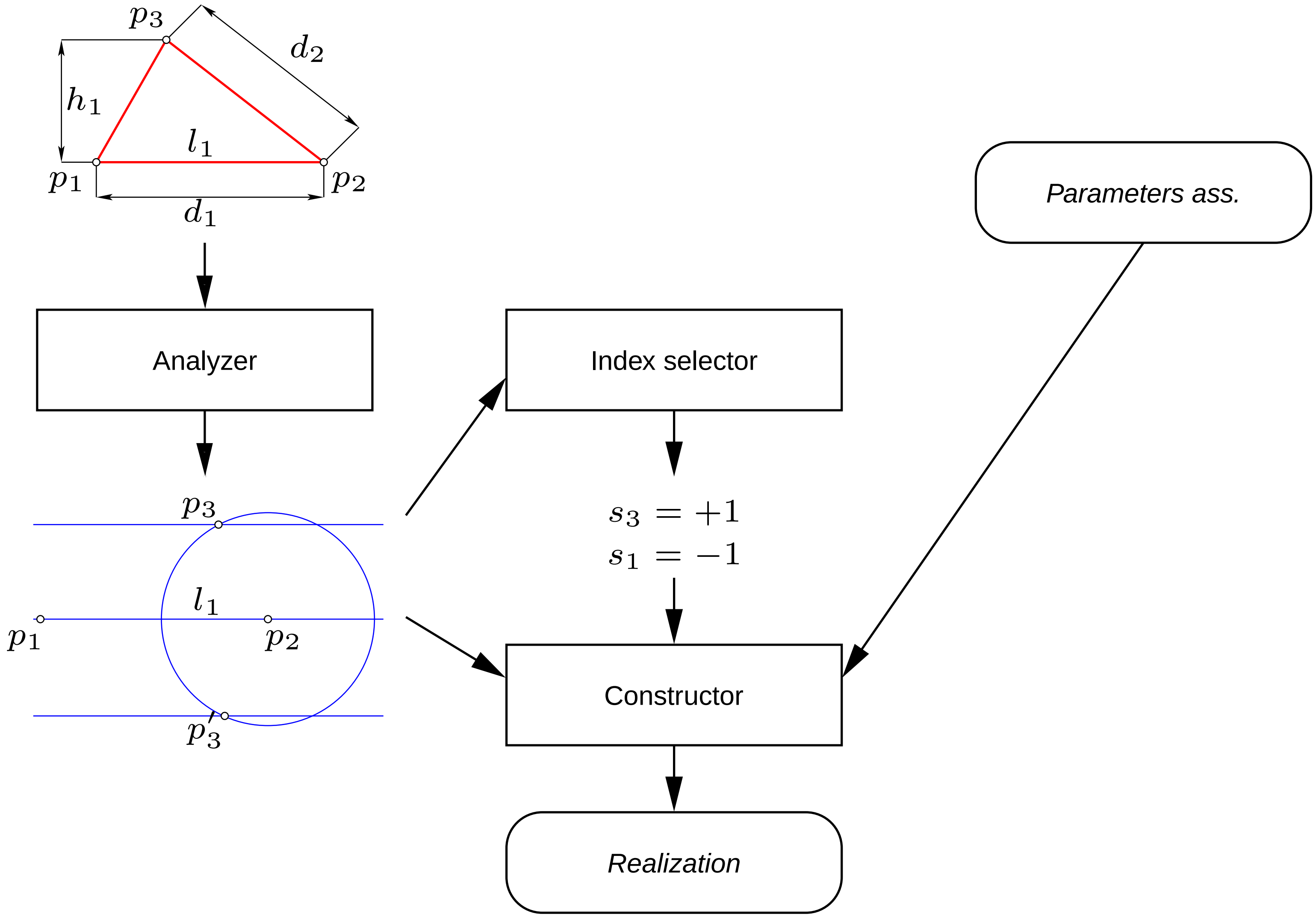
Architecture for Constructive Geometric Constraint Solvers



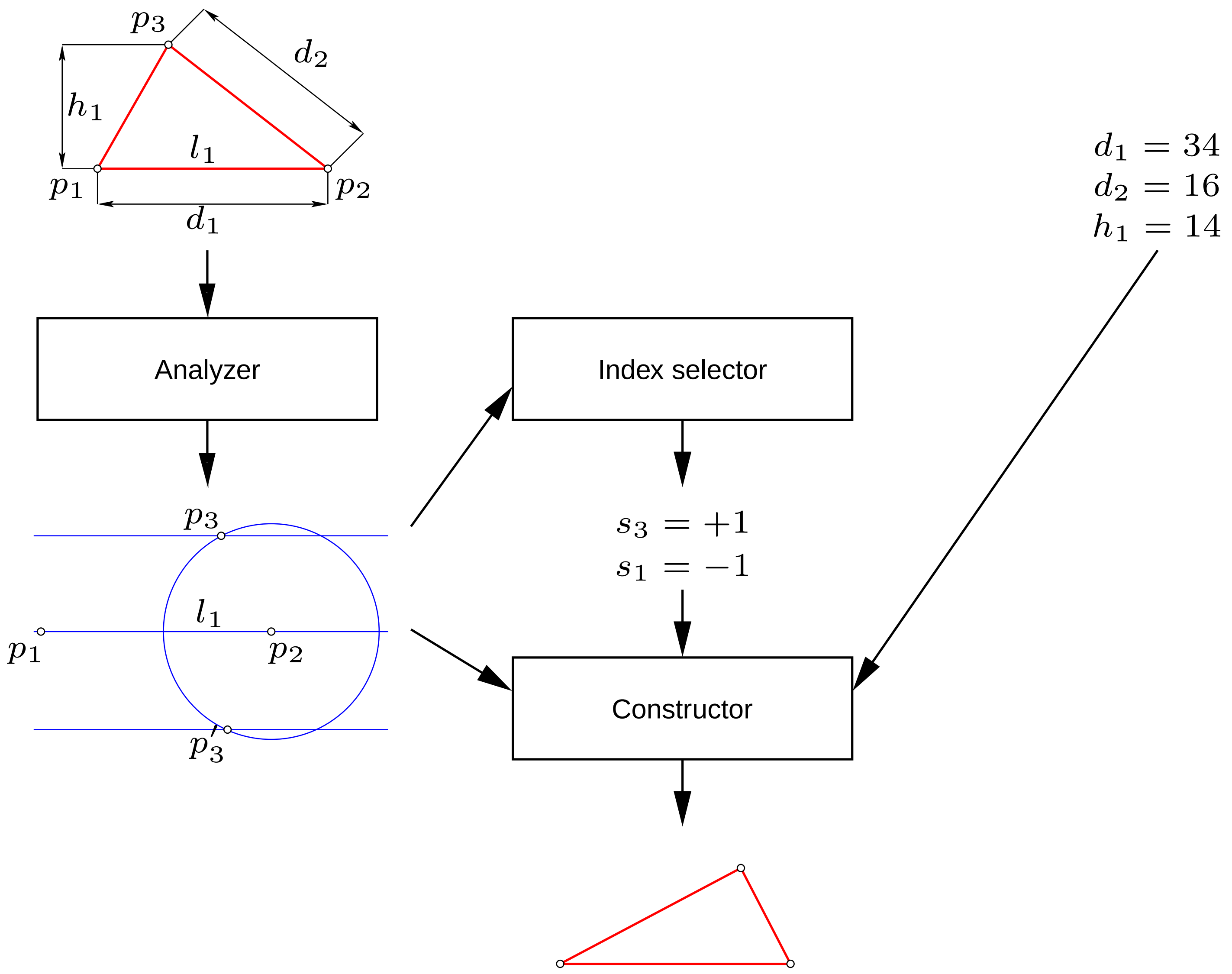
Architecture for Constructive Geometric Constraint Solvers



Architecture for Constructive Geometric Constraint Solvers

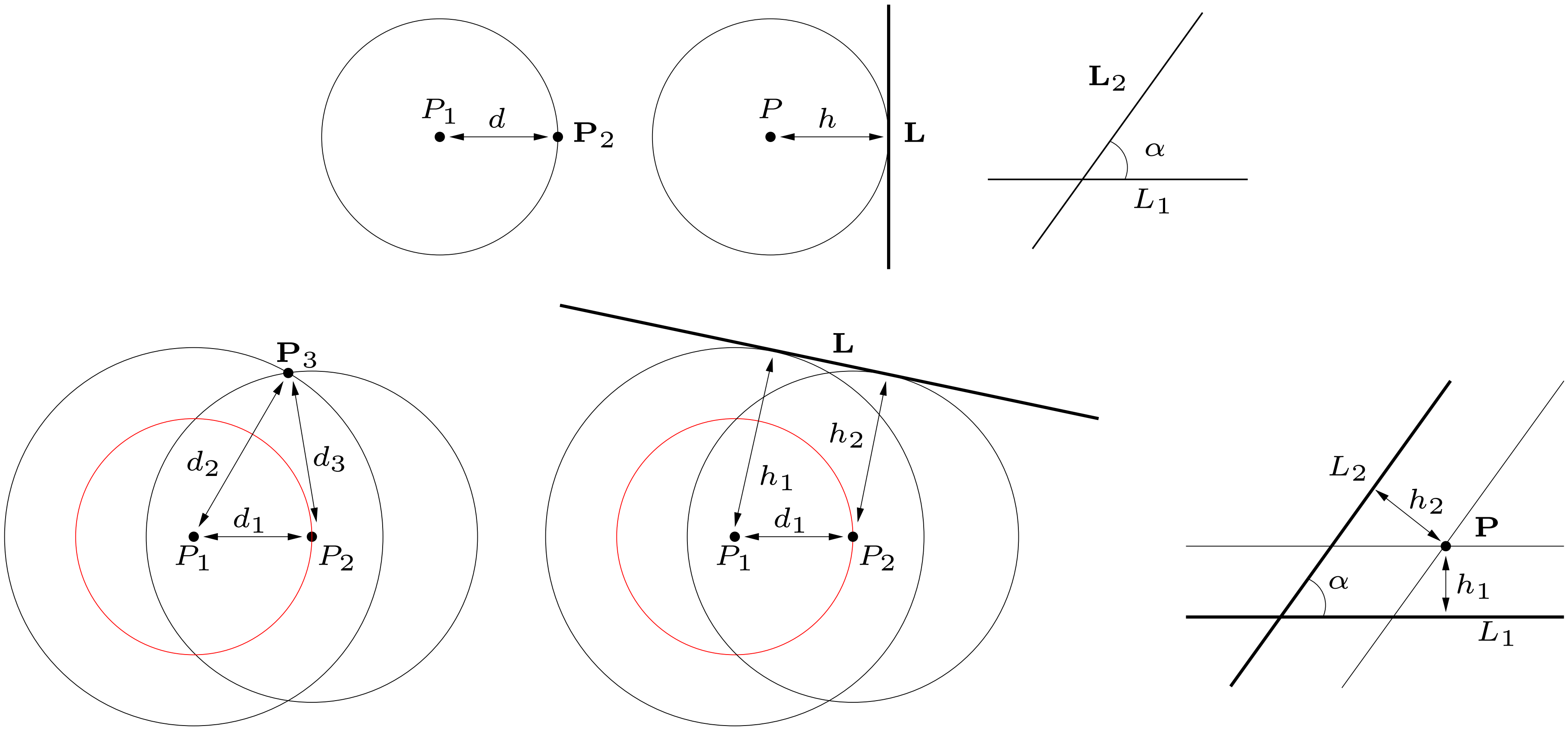


Architecture for Constructive Geometric Constraint Solvers



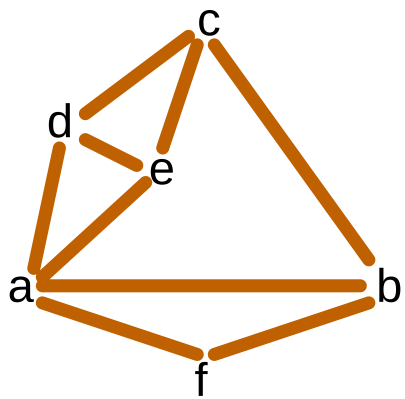
Clusters

A *cluster* is a set of two dimensional geometric elements with known positions with respect to a local coordinate system.



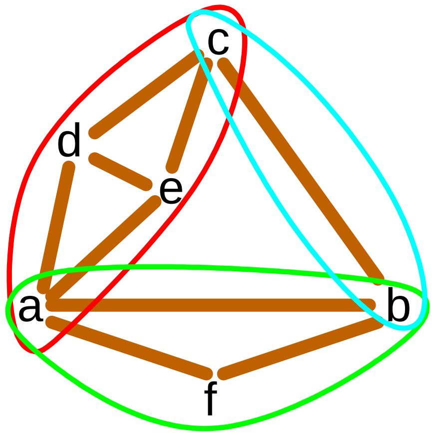
Tree decomposition

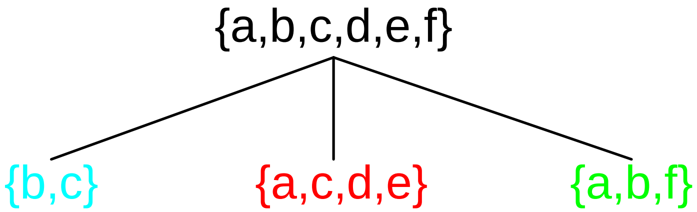
There are graphs that can be tree decomposed



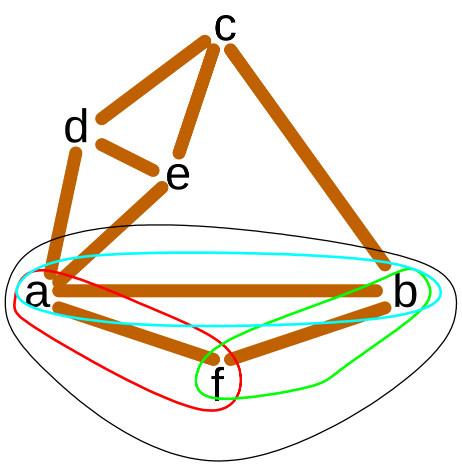
{a,b,c,d,e,f}

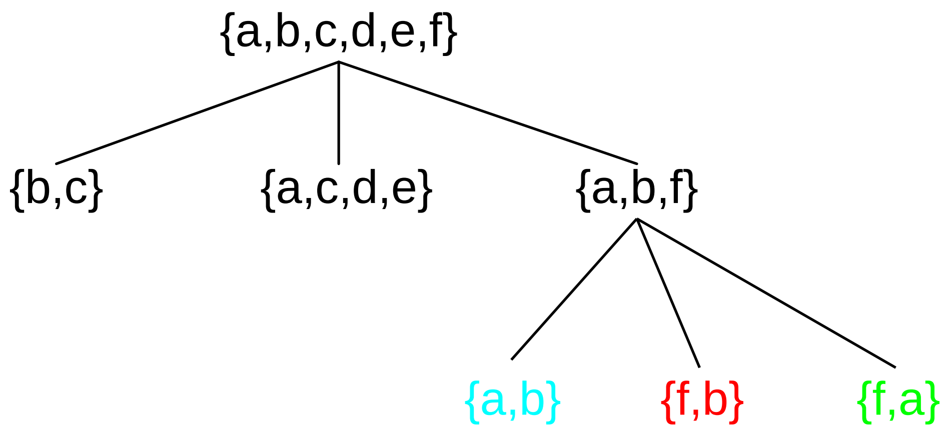
There are graphs that can be tree decomposed



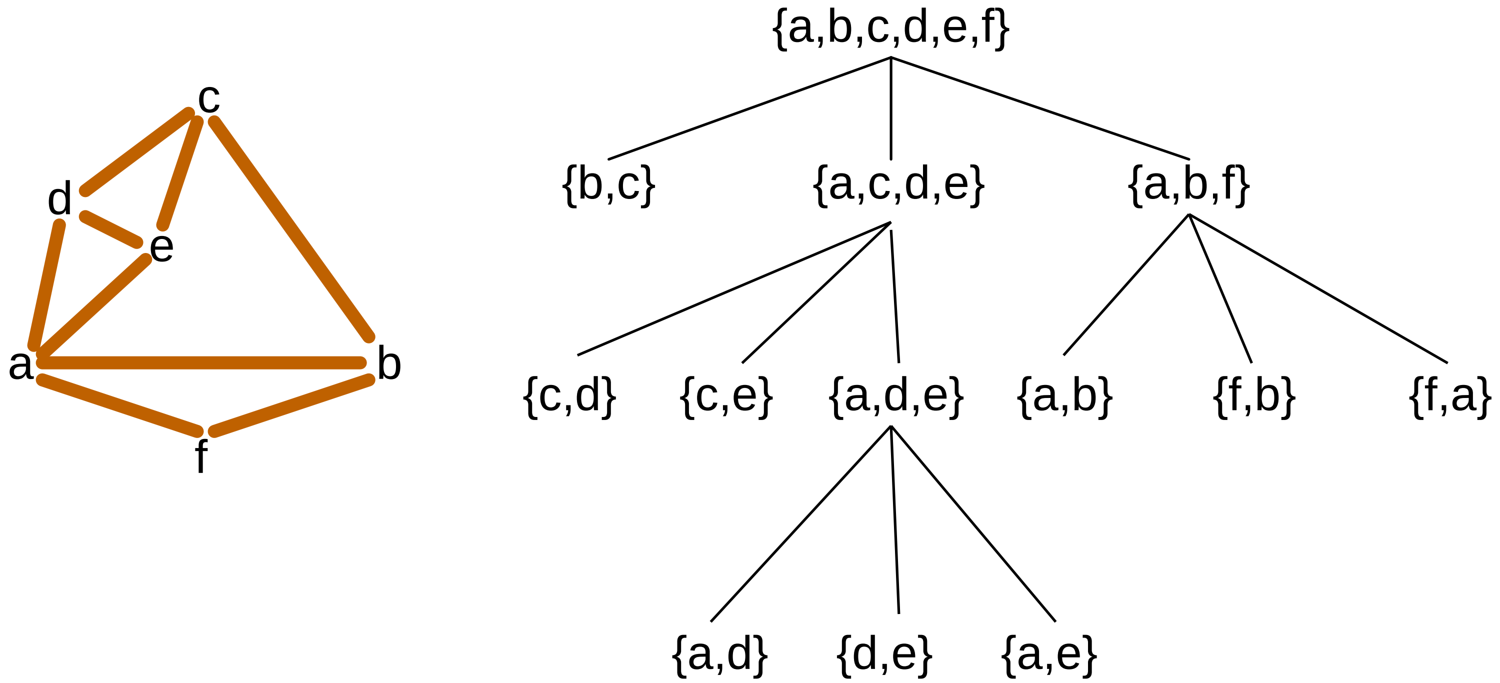


There are graphs that can be tree decomposed

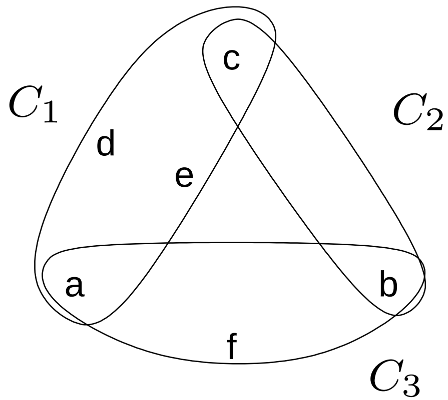




There are graphs that can be tree decomposed

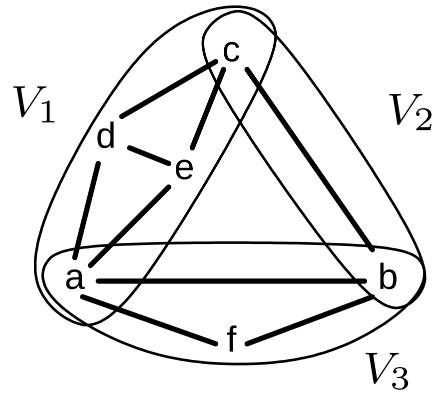


Set decompositions



Let C be a set with, at least, three different members, say a, b, c. Let $\{C_1, C_2, C_3\}$ be three subsets of C. We say that $\{C_1, C_2, C_3\}$ is a *set decomposition* of C if

1. $C_1 \cup C_2 \cup C_3 = C$,
2. $C_1 \cap C_2 = \{a\}$,
3. $C_2 \cap C_3 = \{b\}$ and
4. $C_1 \cap C_3 = \{c\}$



Let $G = (V, E)$ be a graph and let $\{V_1, V_2, V_3\}$ be three subsets of V. $\{V_1, V_2, V_3\}$ is a *set decomposition* of G if it is a set decomposition of V and for every edge e in E, $V(e) \subseteq V_i$ for some i, $1 \leq i \leq 3$.

Tree decomposition

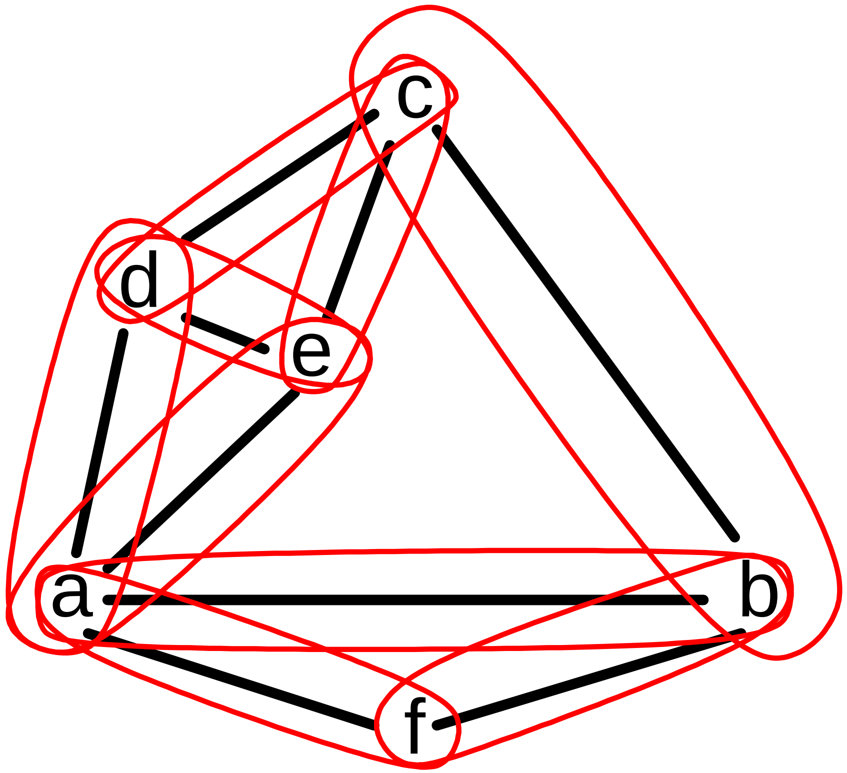
Let $G = (V, E)$ be a graph. A 3-ary tree T is a *tree decomposition* of G if

1. V is the root of T,
2. Each internal node $V' \subset V$ of T is the father of exactly three nodes, say $\{V_1', V_2', V_3'\}$, which are a set decomposition of the subgraph induced by V', and
3. Each leaf node contains exactly two vertices of V.

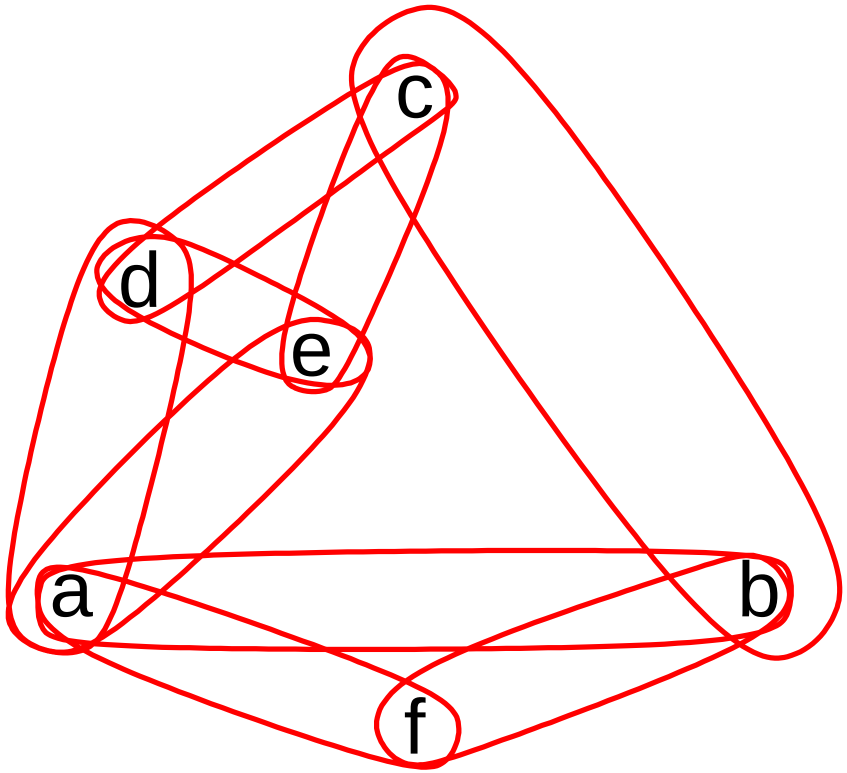
A graph G is *tree decomposable* if there is a tree decomposition of G.

Reduction analysis

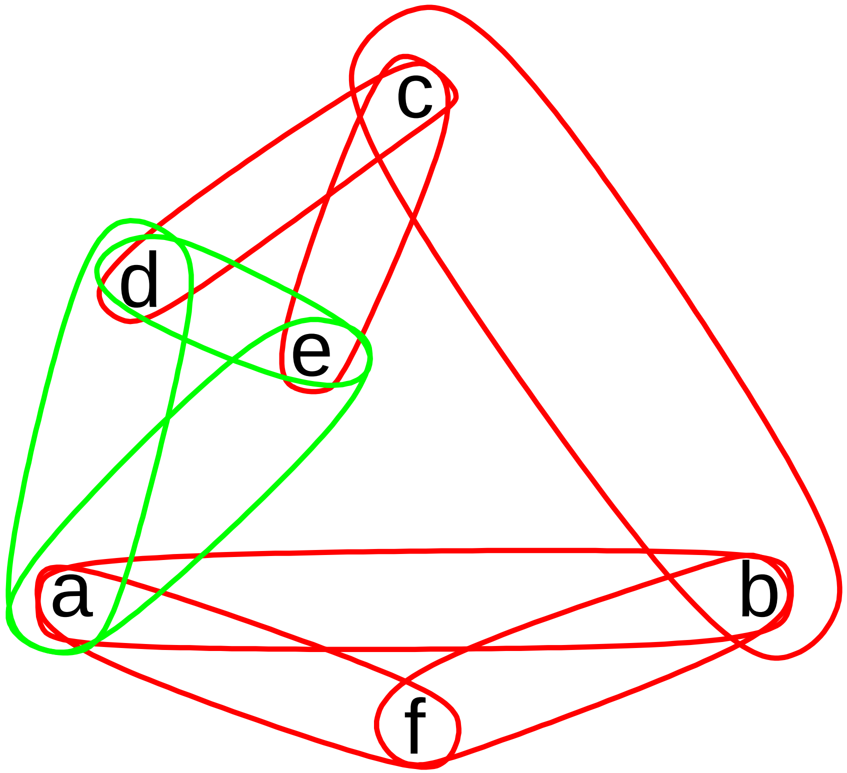
There are graphs that can be reduced



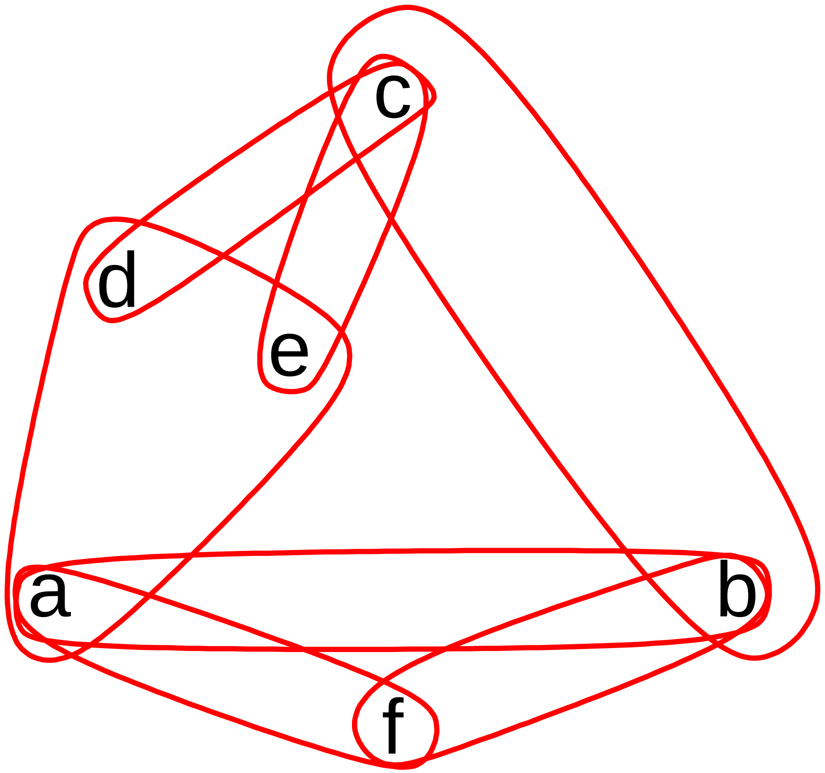
There are graphs that can be reduced



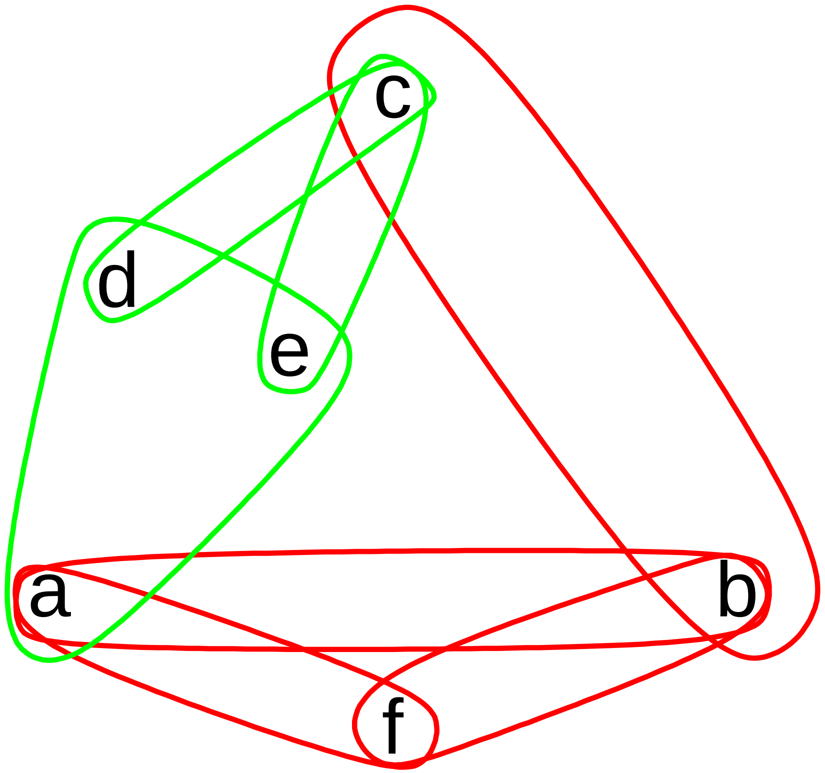
There are graphs that can be reduced



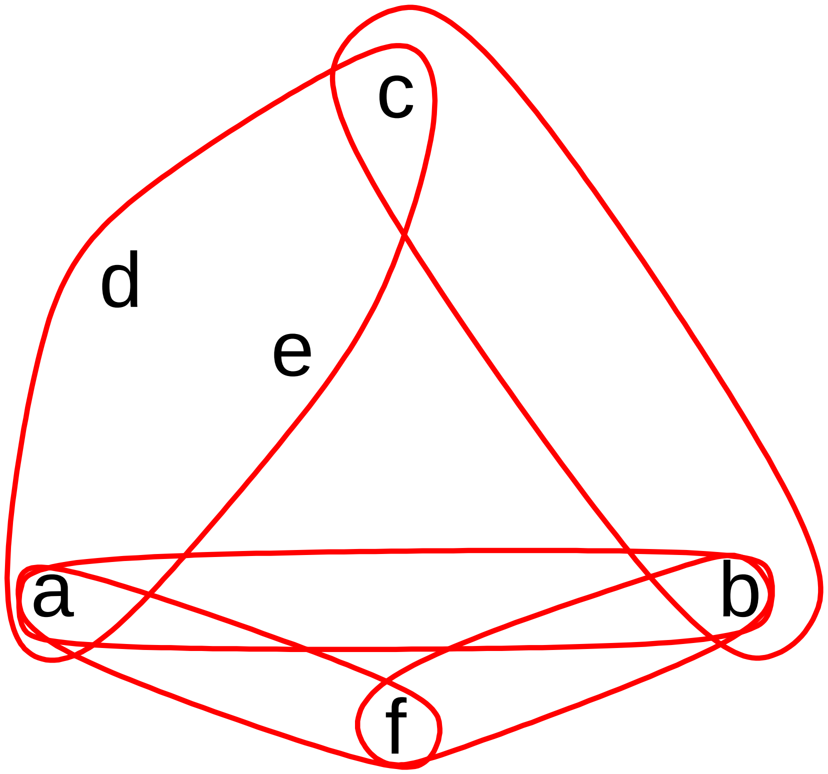
There are graphs that can be reduced



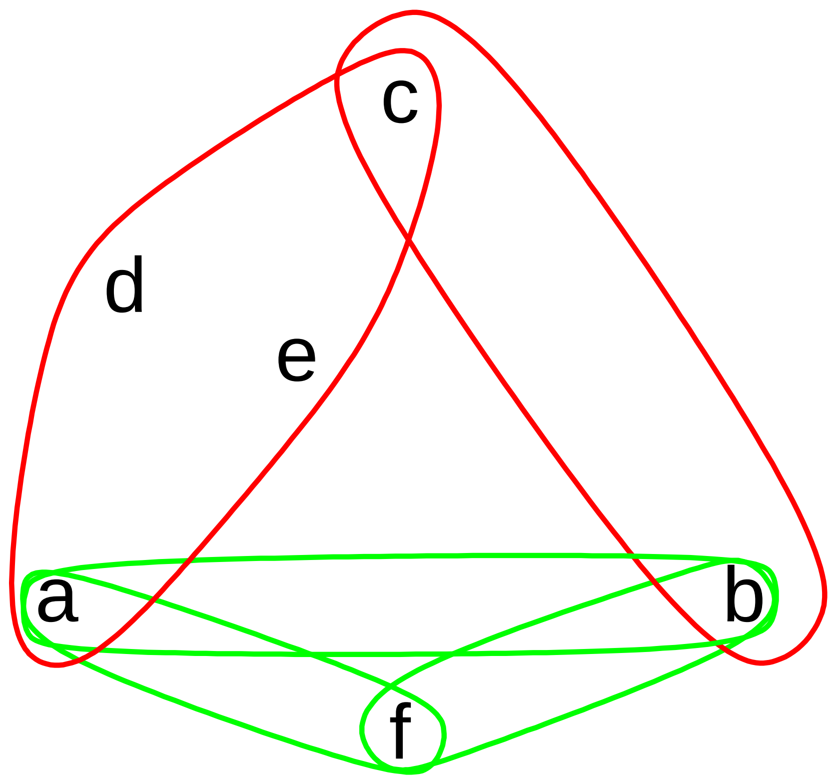
There are graphs that can be reduced



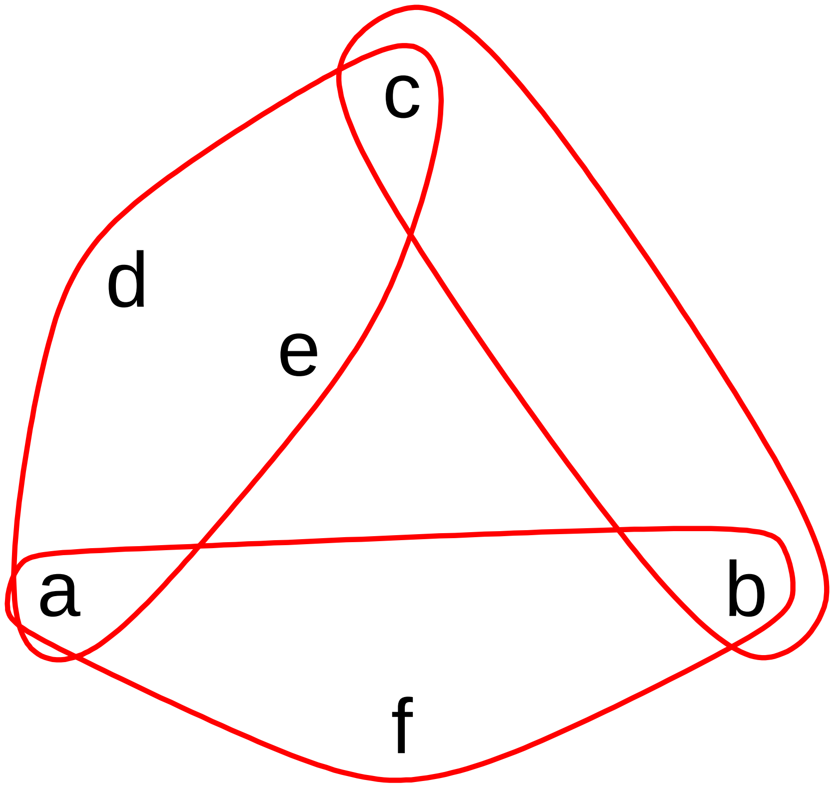
There are graphs that can be reduced



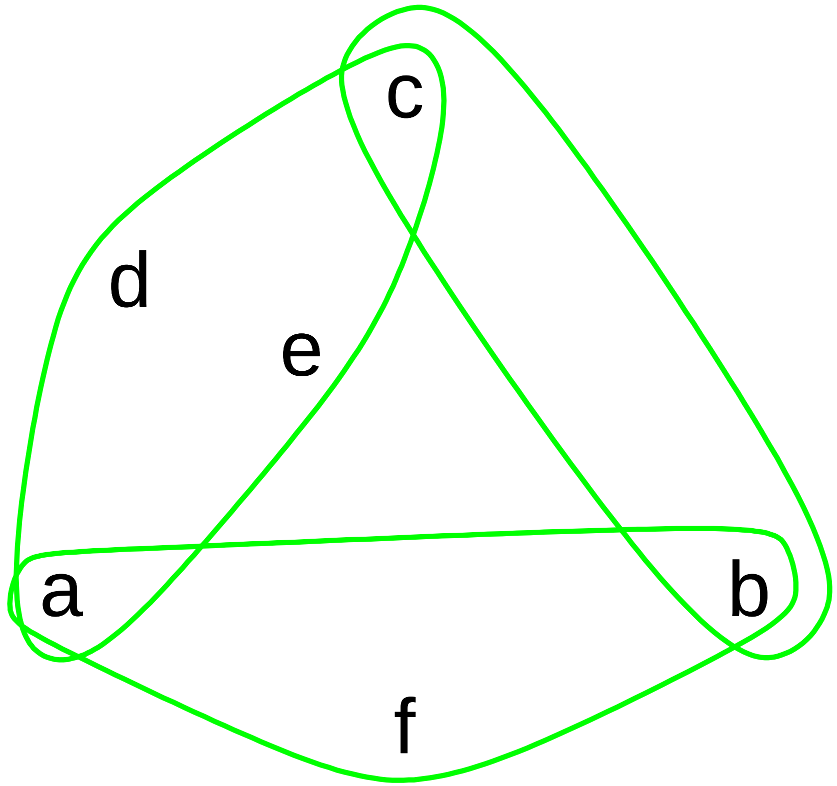
There are graphs that can be reduced



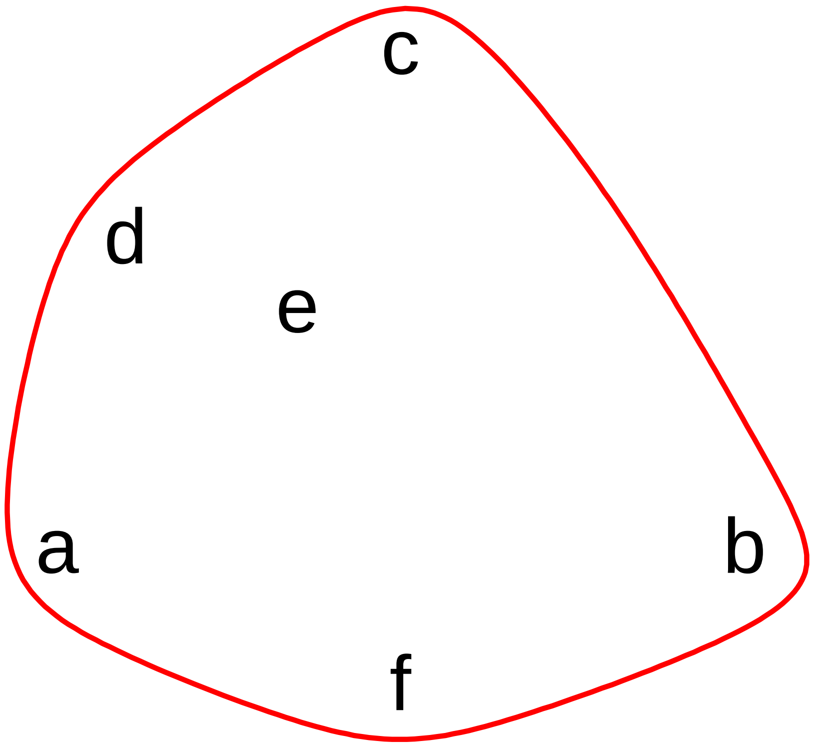
There are graphs that can be reduced



There are graphs that can be reduced



There are graphs that can be reduced



Reduction analysis

Let $G = (V, E)$ be a geometric constraint graph.

We define the *initial set of clusters* $\mathbf{S}_G = \{\{u, v\} \mid (u, v) \in E\}$.

Let $\mathbf{S}$ be a set of clusters in which there are three clusters C_1, C_2, C_3 such that $\{C_1, C_2, C_3\}$ is a set decomposition of C.

$\mathbf{S} \longrightarrow_r \mathbf{S}'$ is a reduction rule where $\mathbf{S}' = (\mathbf{S} - \{C_1, C_2, C_3\}) \cup C$.

The geometric constraint problem represented by the geometric constraint graph G is *solvable by reduction analysis* if $\mathbf{S}_G$ reduces to the singleton $\{V\}$.

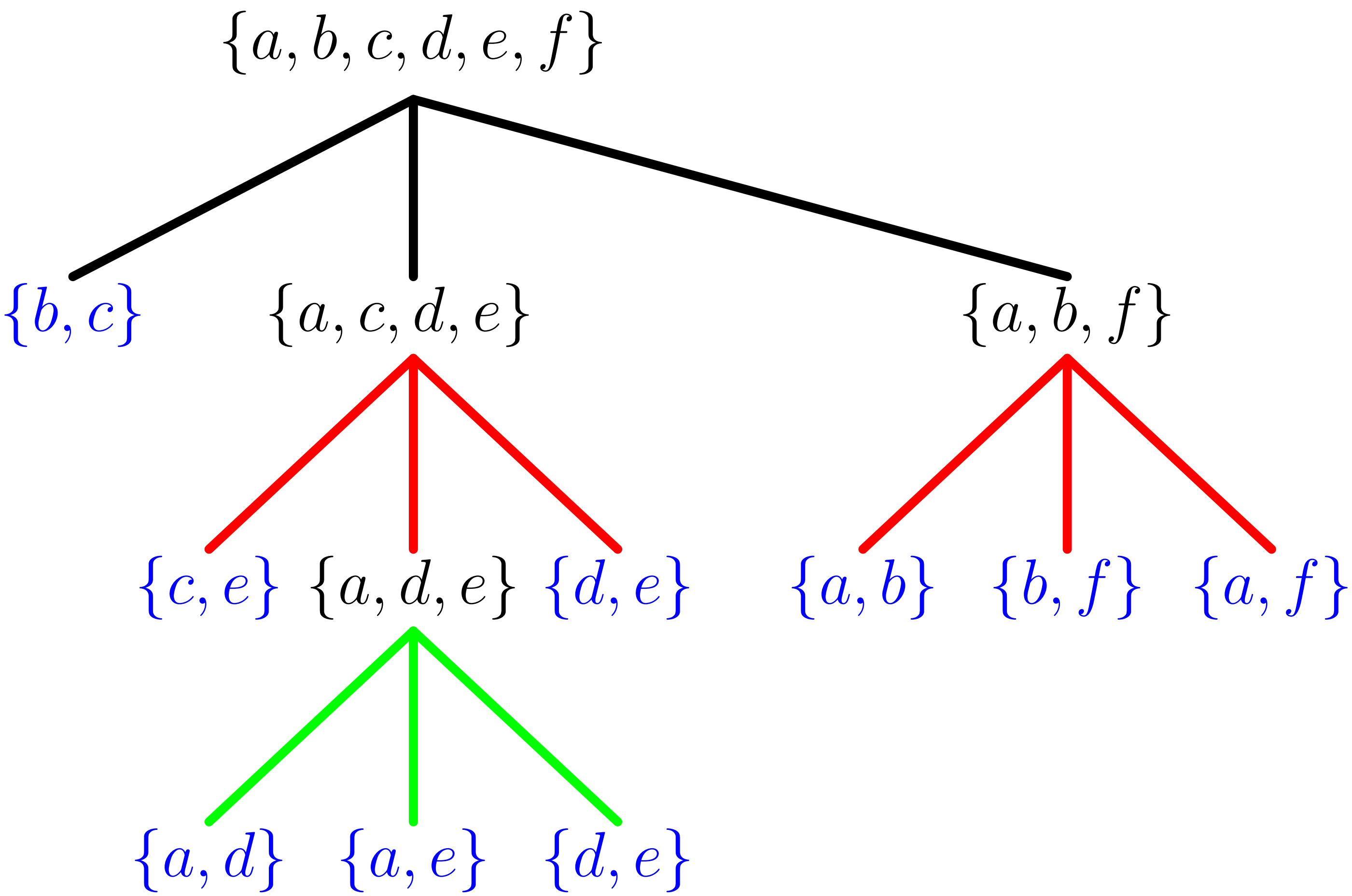
If G is not structurally over-constrained, the abstract reduction system induced by the reduction rule $\longrightarrow_r$ is terminating and confluent which implies the unique normal form property and canonicity.

The domain of solvable graphs by reduction analysis

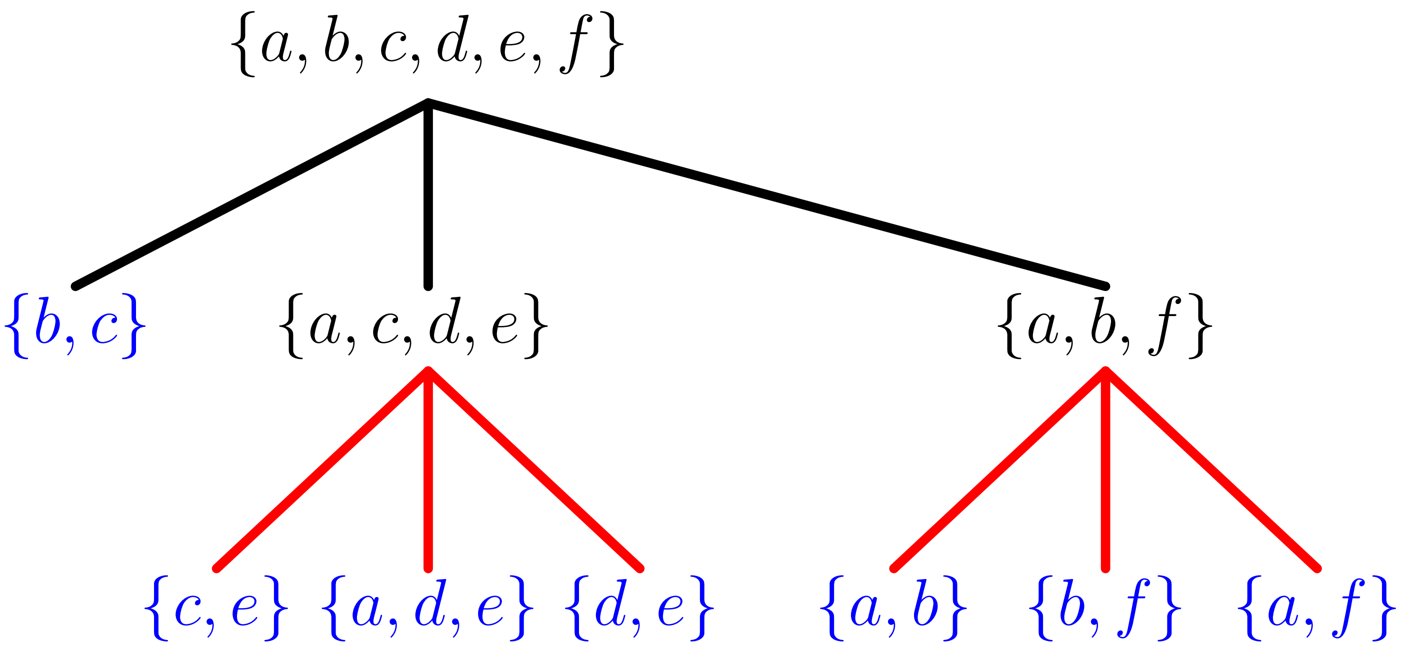
Let $G = (V, E)$ be a well-constrained geometric constraint graph. The following assertions are equivalent:

1. G is tree decomposable.
2. G is solvable by reduction analysis.

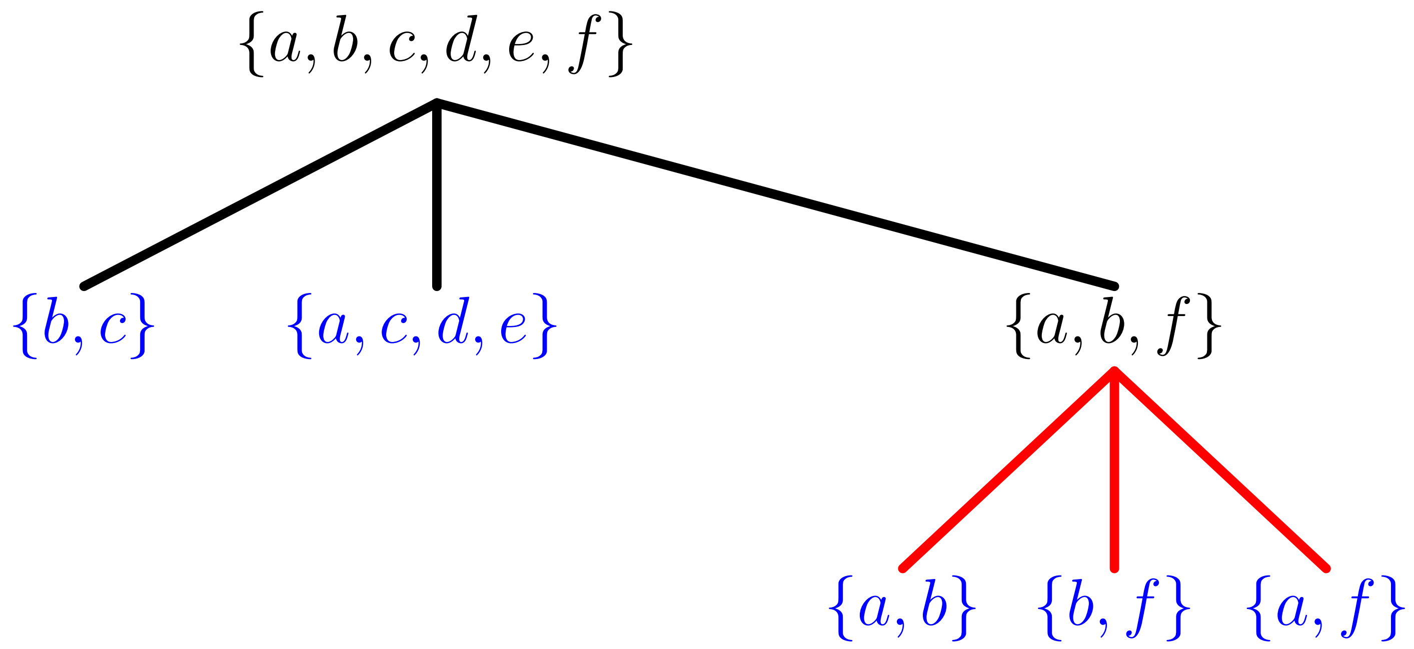
The domain of solvable graphs by reduction analysis



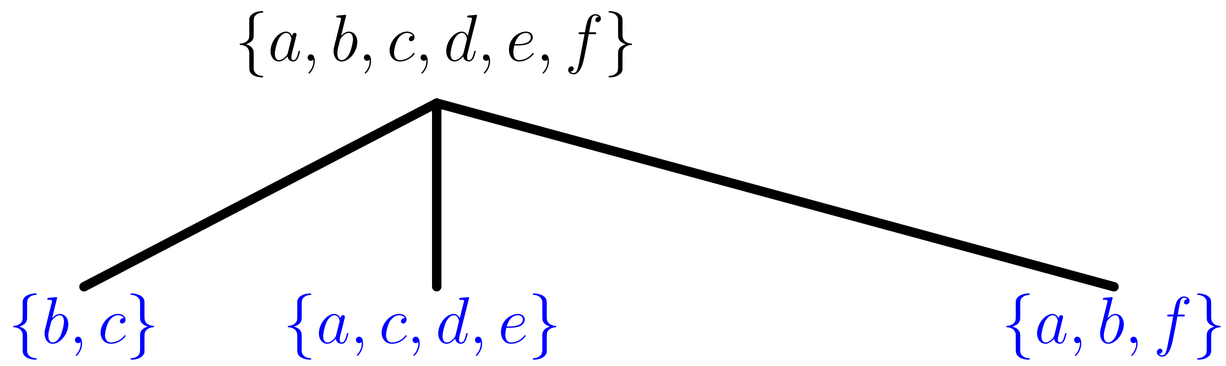
The domain of solvable graphs by reduction analysis



The domain of solvable graphs by reduction analysis



The domain of solvable graphs by reduction analysis

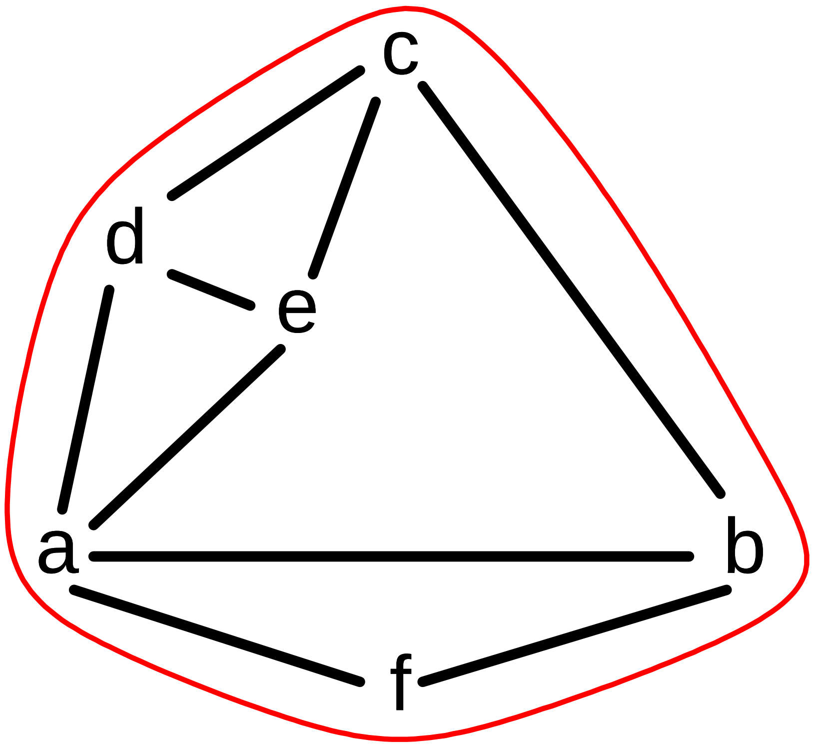


The domain of solvable graphs by reduction analysis

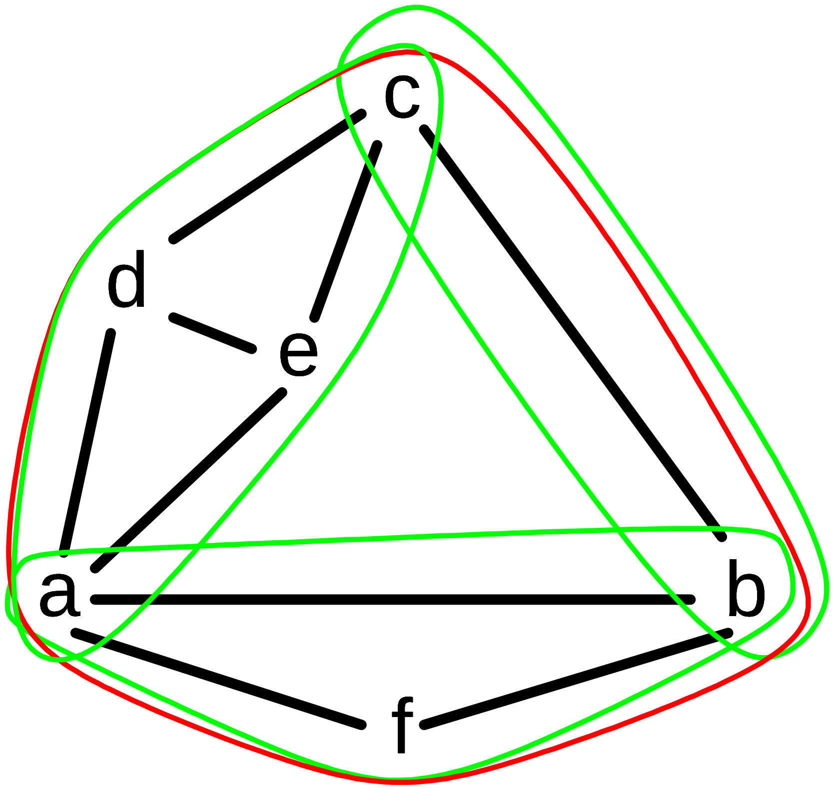
$\{a, b, c, d, e, f\}$

Decomposition analysis

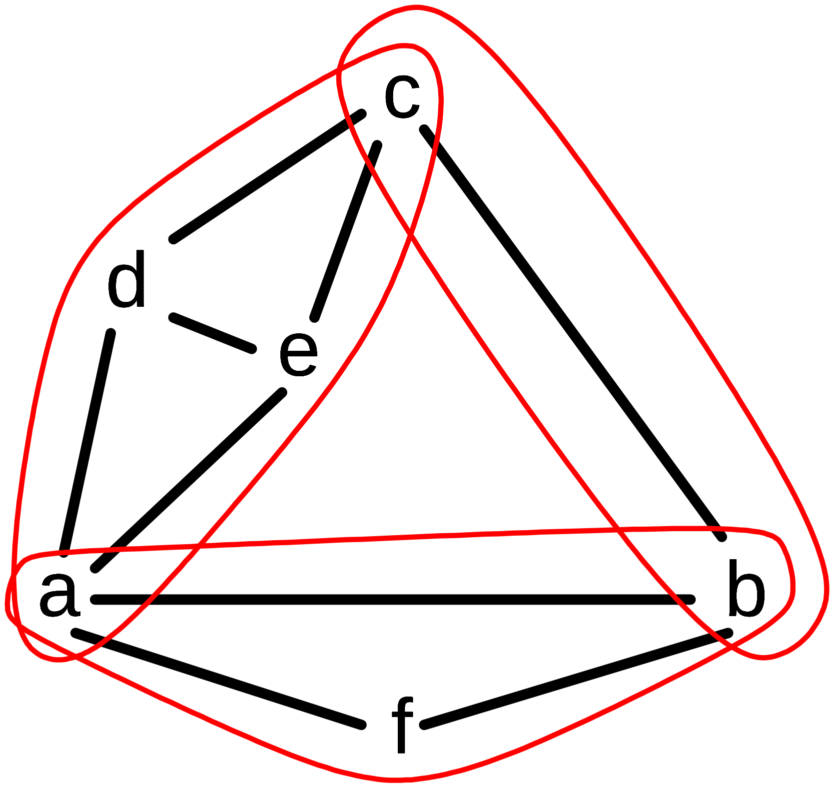
There are graphs that can be decomposed



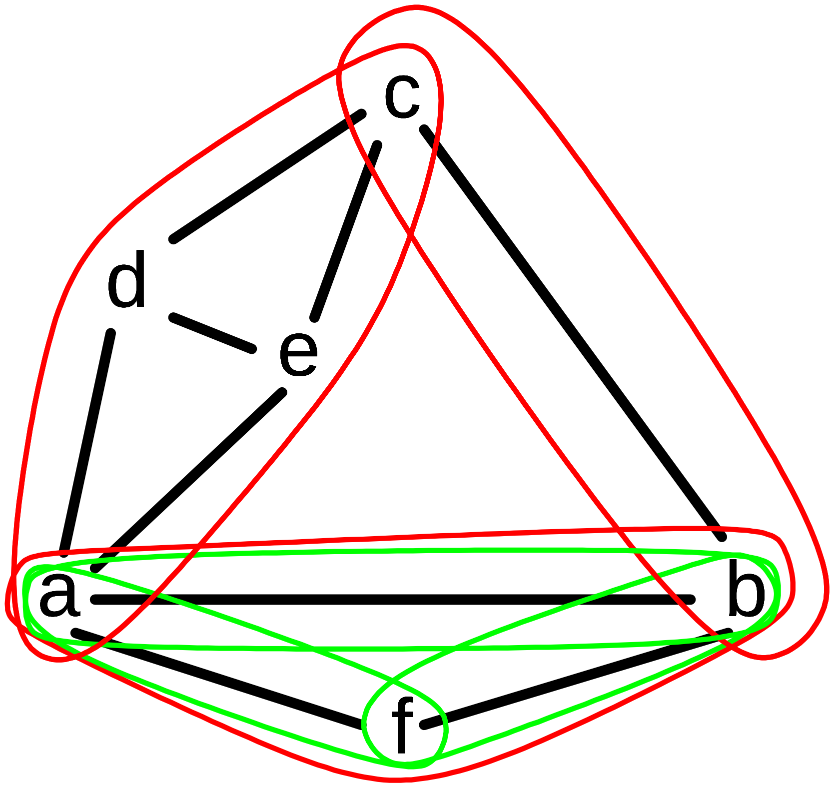
There are graphs that can be decomposed



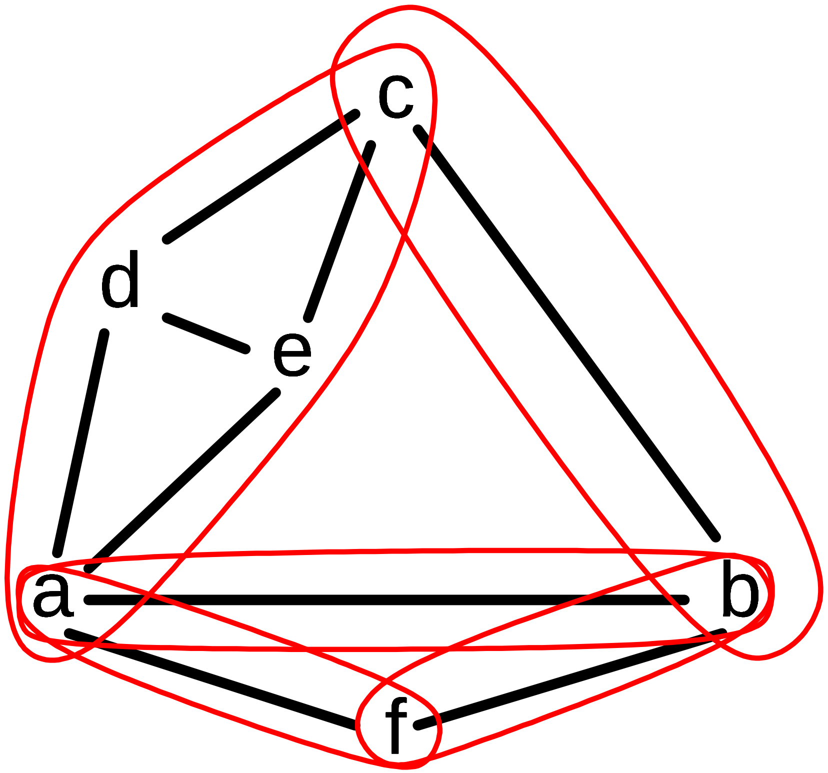
There are graphs that can be decomposed



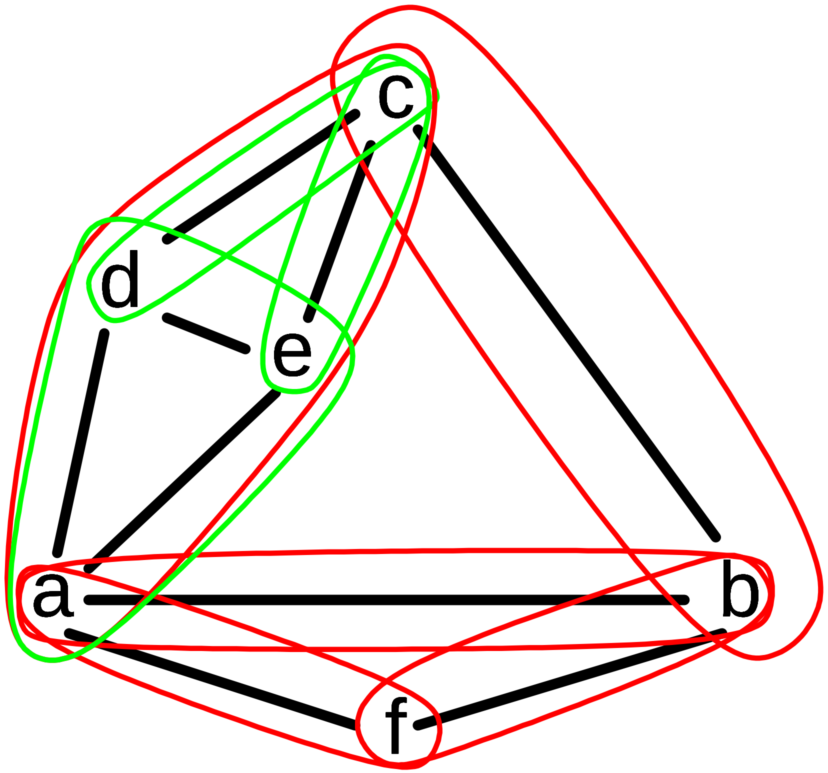
There are graphs that can be decomposed



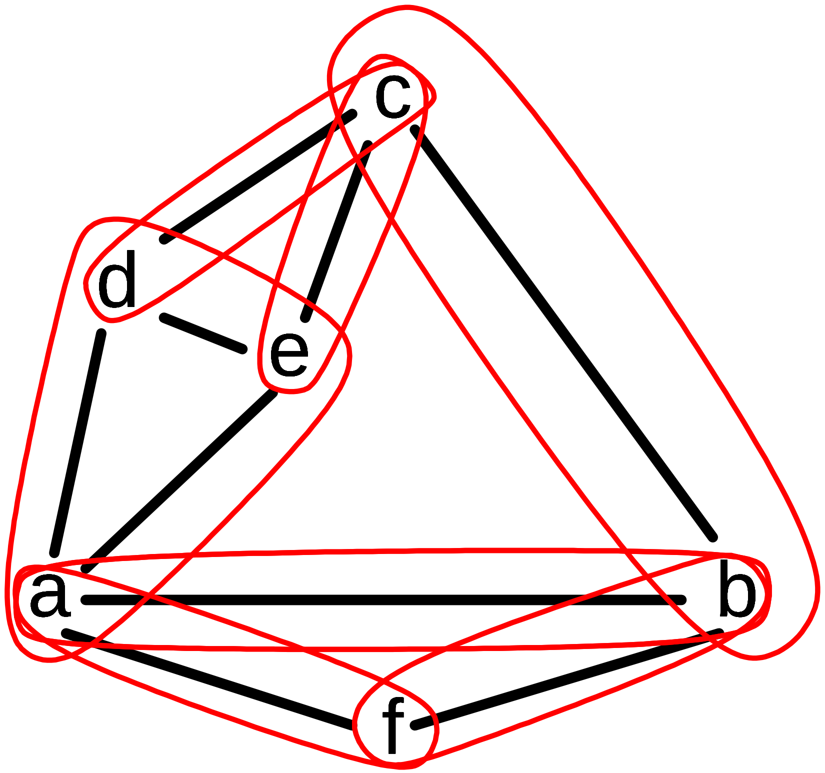
There are graphs that can be decomposed



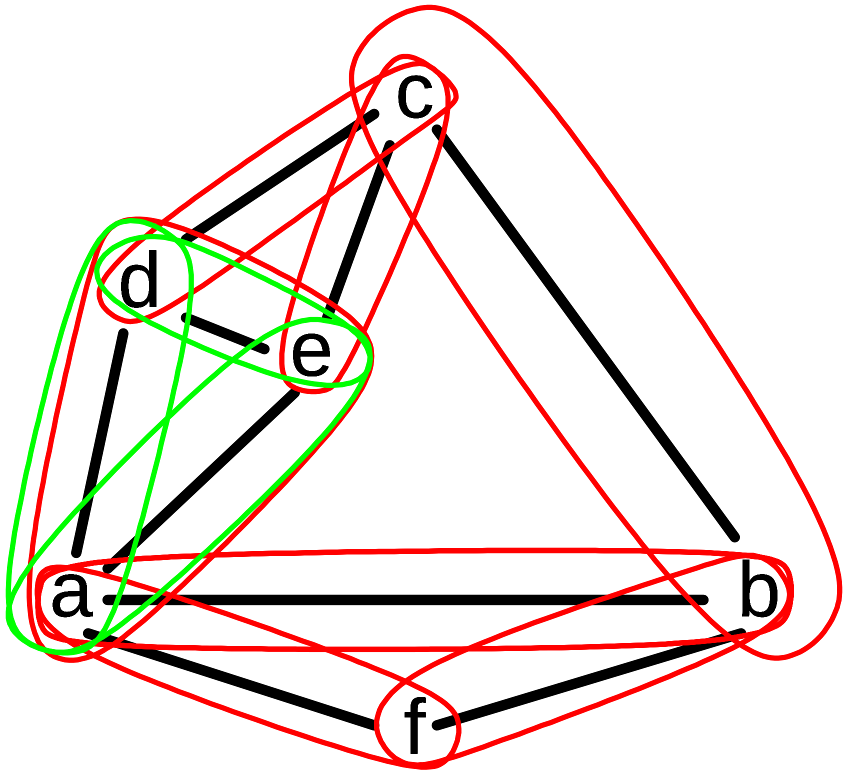
There are graphs that can be decomposed



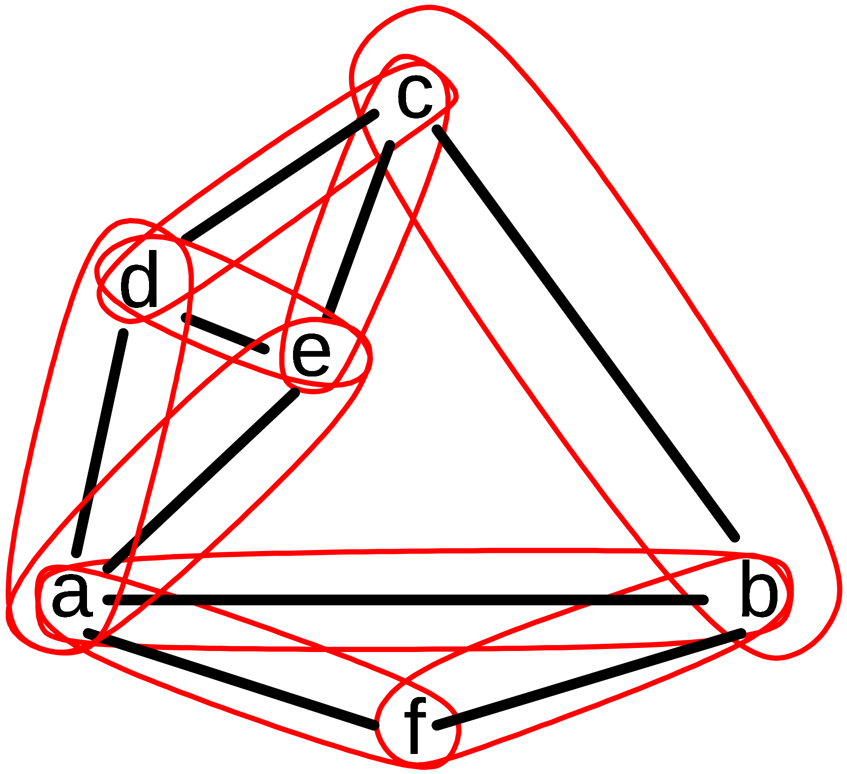
There are graphs that can be decomposed



There are graphs that can be decomposed



There are graphs that can be decomposed



Decomposition analysis

Let $G = (V, E)$ be a geometric constraint graph.

We define the *initial set of clusters* $\mathbf{O}_G = \{V\}$.

Let $\mathbf{O}$ be a set of clusters in which there is a cluster C such that $\{C_1, C_2, C_3\}$ is a set decomposition of the subgraph of G induced by C.

$\mathbf{O} \longrightarrow_o \mathbf{O}'$ is a reduction rule where $\mathbf{O}' = (\mathbf{O} - C) \cup \{C_1, C_2, C_3\}$.

The geometric constraint problem represented by the geometric constraint graph G is *solvable by decomposition analysis* if $\mathbf{O}_G$ reduces to $\mathbf{S}_G$.

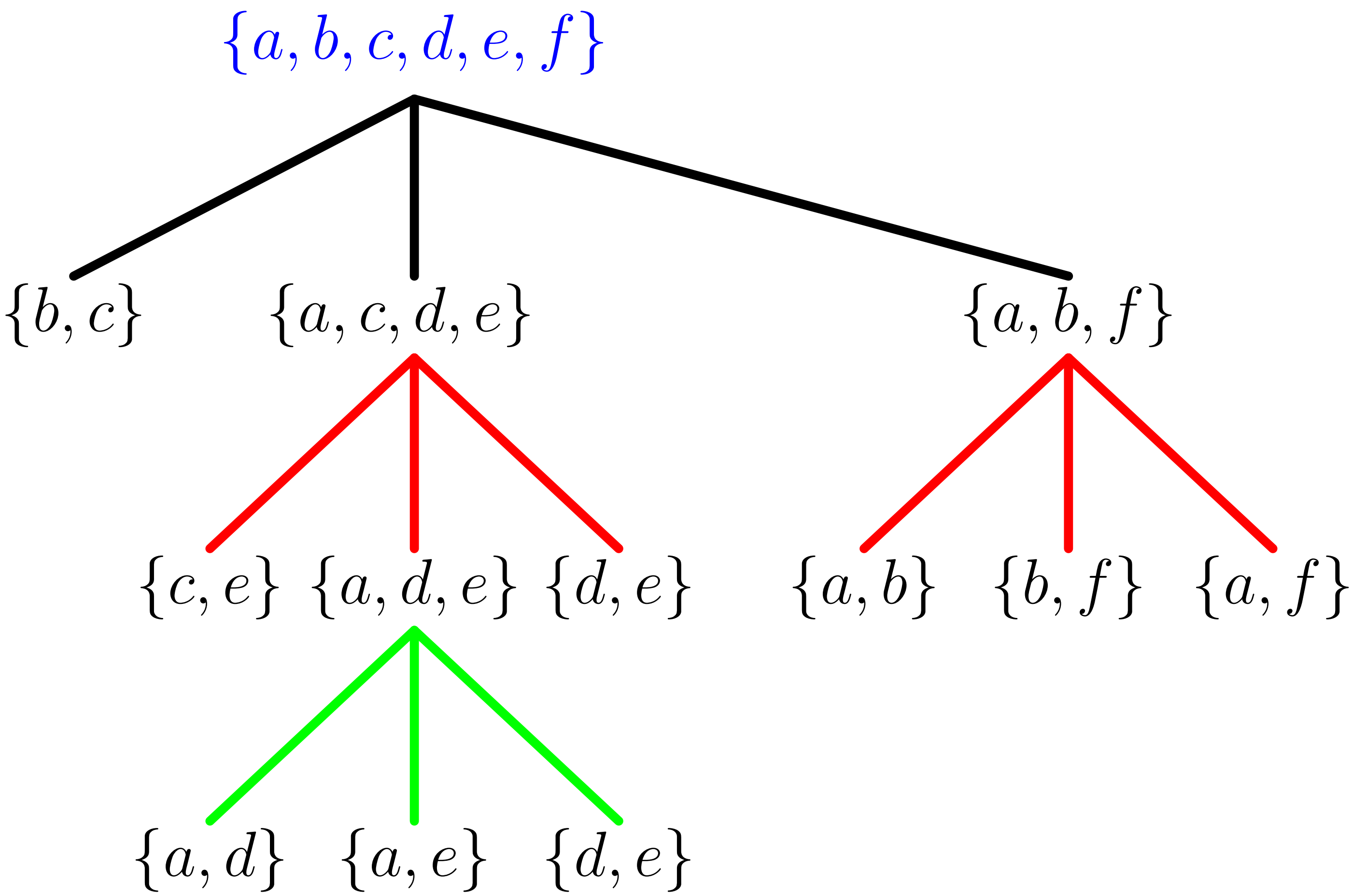
The reduction relation $\longrightarrow_o$ induces an abstract reduction system.

The domain of solvable graphs by decomposition analysis

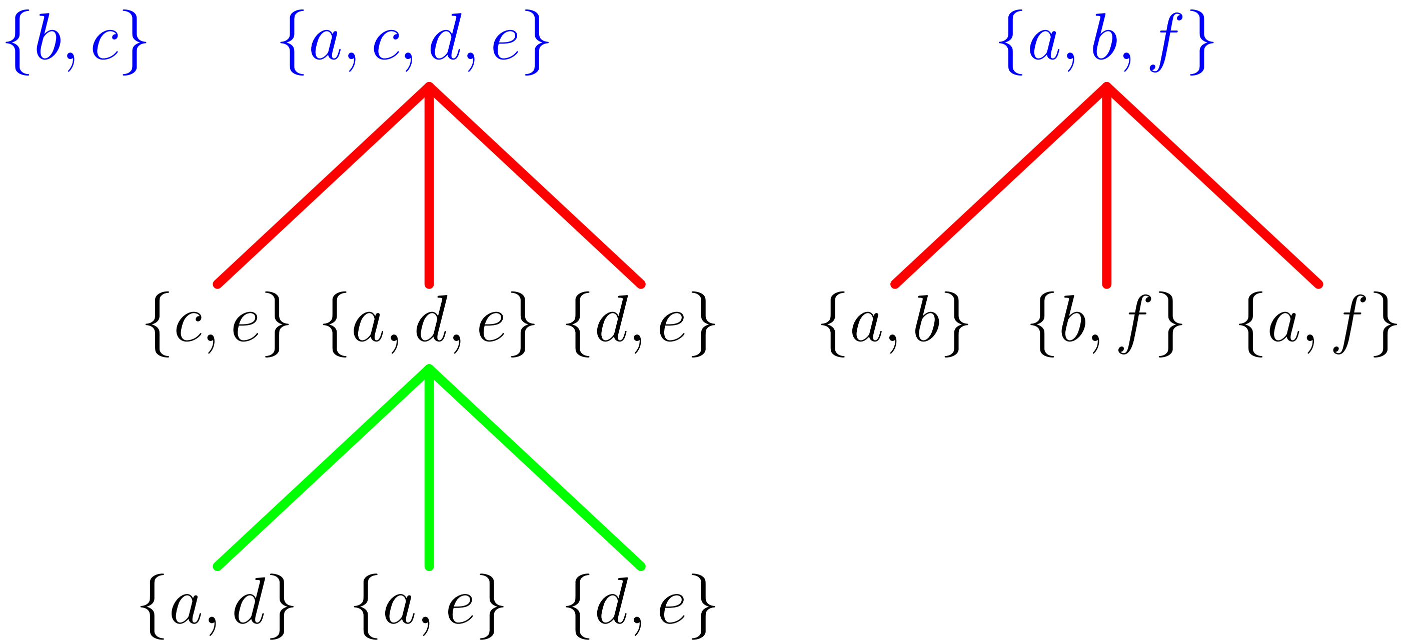
Let $G = (V, E)$ be a well-constrained geometric constraint graph. The following assertions are equivalent:

1. G is tree decomposable.
2. G is solvable by decomposition analysis.

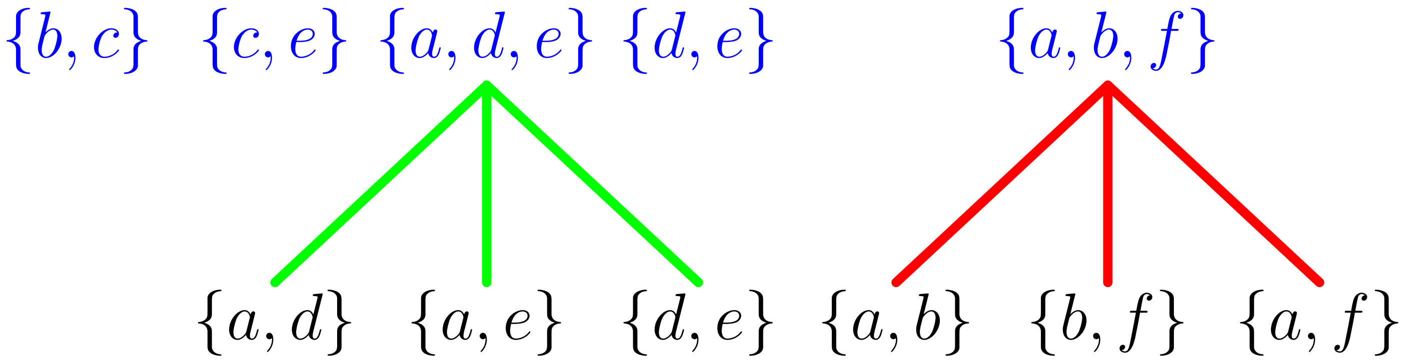
The domain of solvable graphs by decomposition analysis



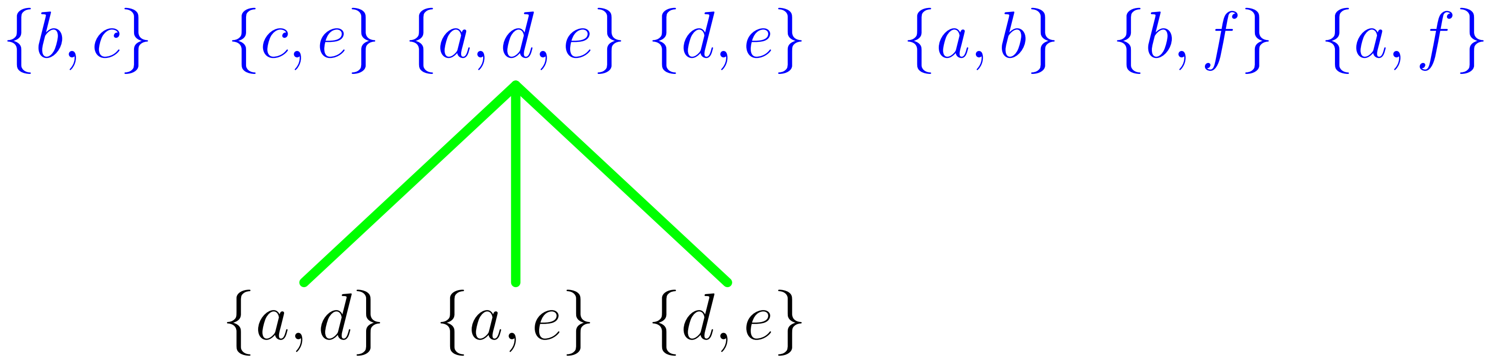
The domain of solvable graphs by decomposition analysis



The domain of solvable graphs by decomposition analysis



The domain of solvable graphs by decomposition analysis

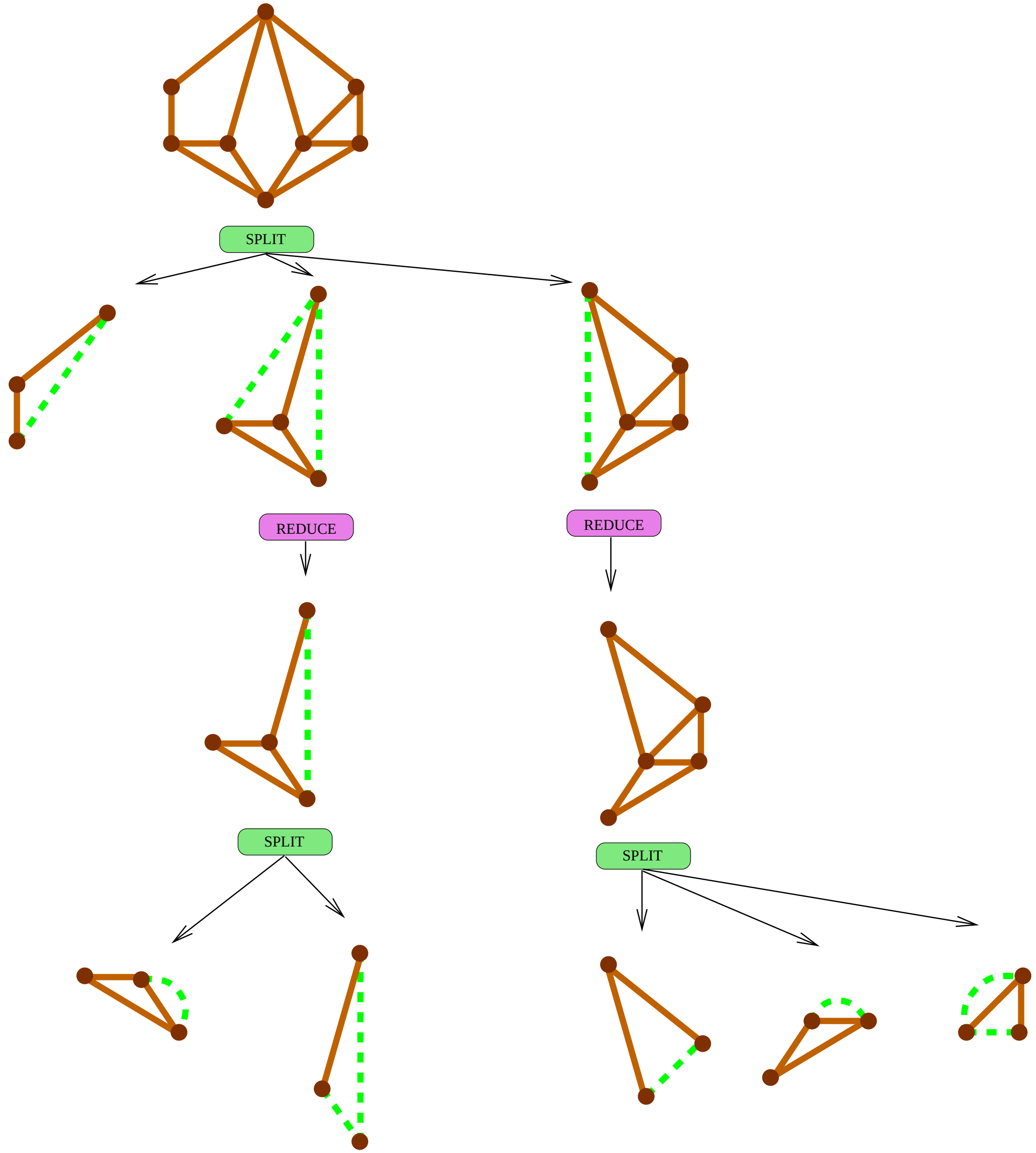


The domain of solvable graphs by decomposition analysis

$$\{b, c\} \quad \{c, e\} \quad \{a, d\} \quad \{a, e\} \quad \{d, e\} \quad \{d, e\} \quad \{a, b\} \quad \{b, f\} \quad \{a, f\}$$

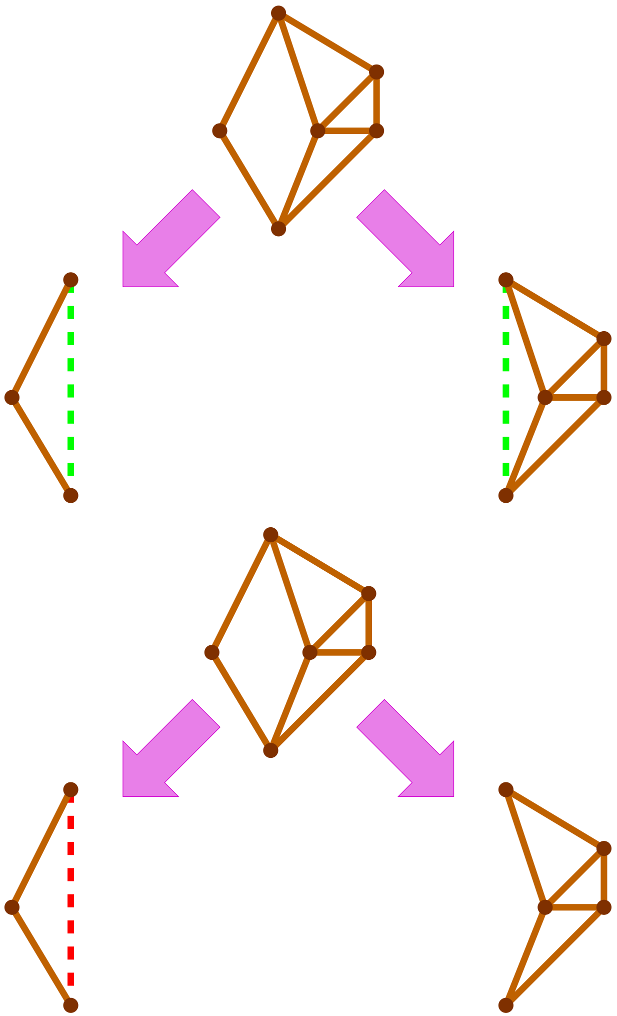
Reformulating Owen's algorithm

Owen's algorithm relies on computing triconnected components . . .



- . . . but after each split some well chosen edges should be removed to continue the process.
- It is difficult to understand which edges should be removed and the reason why they should be removed.

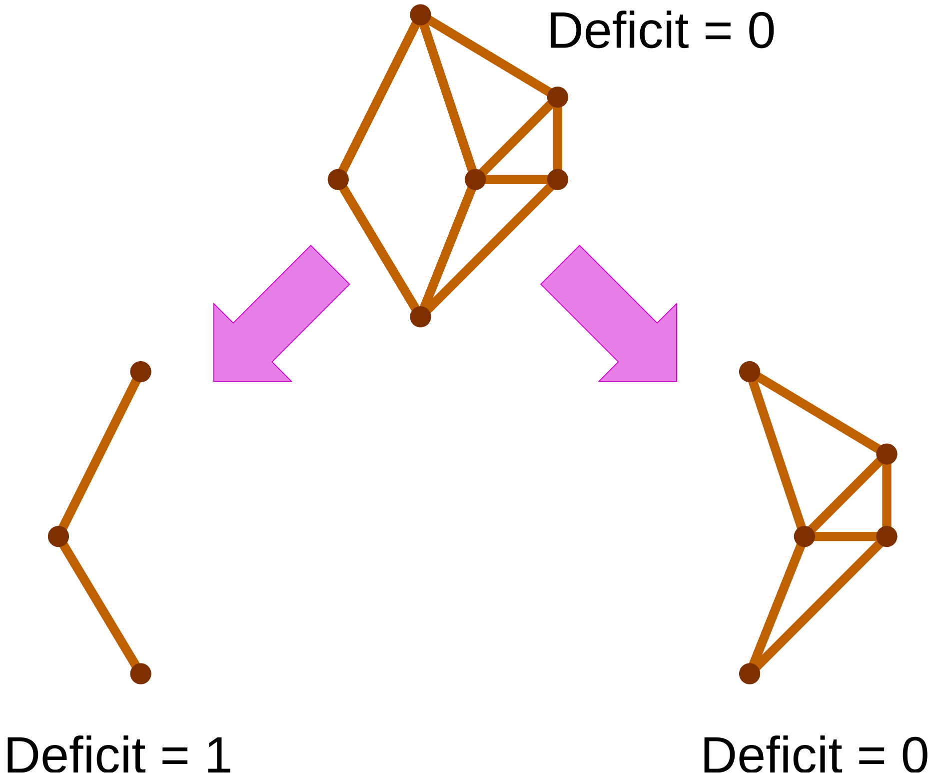
Which edges and why should they be removed?



- The triconnected components algorithm subdivides the graph and adds virtual edges to preserve connectivity properties.
- To further subdivide, Owen's algorithm removes virtual edges *"at any articulation pair with no single edge and exactly one more complex subgraph"*.

The property to be preserved in decomposition algorithms is the deficit

- What is essential to preserve in the graph subdivision process is *rigidity properties*, not connectivity properties.



- Deficit function of a graph $G = (V, E)$ is defined as

$$\text{Deficit}(G) = (2|V| - 3) - |E|$$

- At every graph split, deficit value should be maintained. Thus new edges must be added to fulfill this requirement.

Two results show how deficit can be maintained

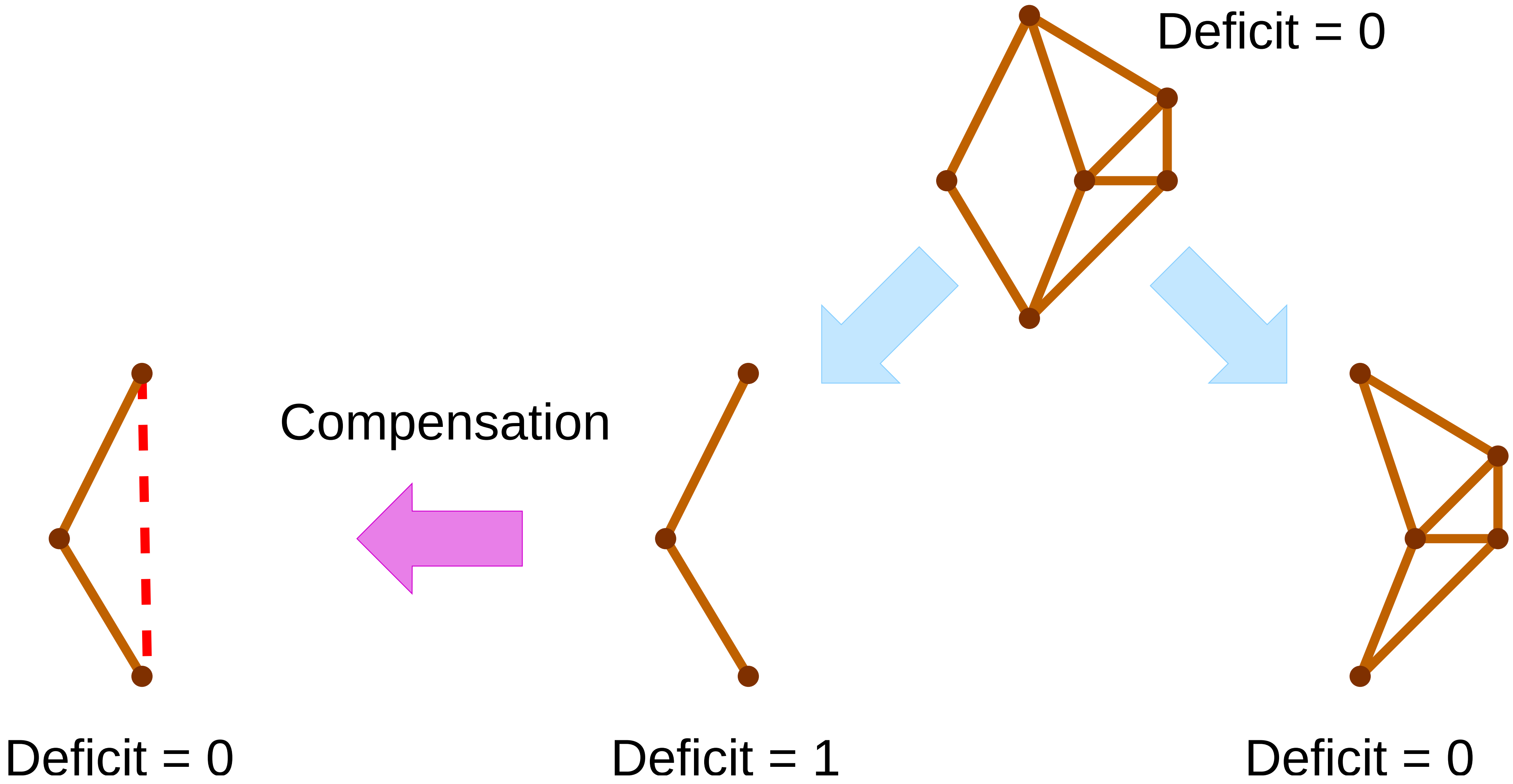
Let G be a well-constrained constraint graph and G' and G'' separating graphs of G. Then

- Deficit$(G) =$ Deficit$(G') +$ Deficit$(G'') - 1$
- If Deficit(G') $>$ Deficit(G''), G' is under-constrained and G'' is well-constrained.

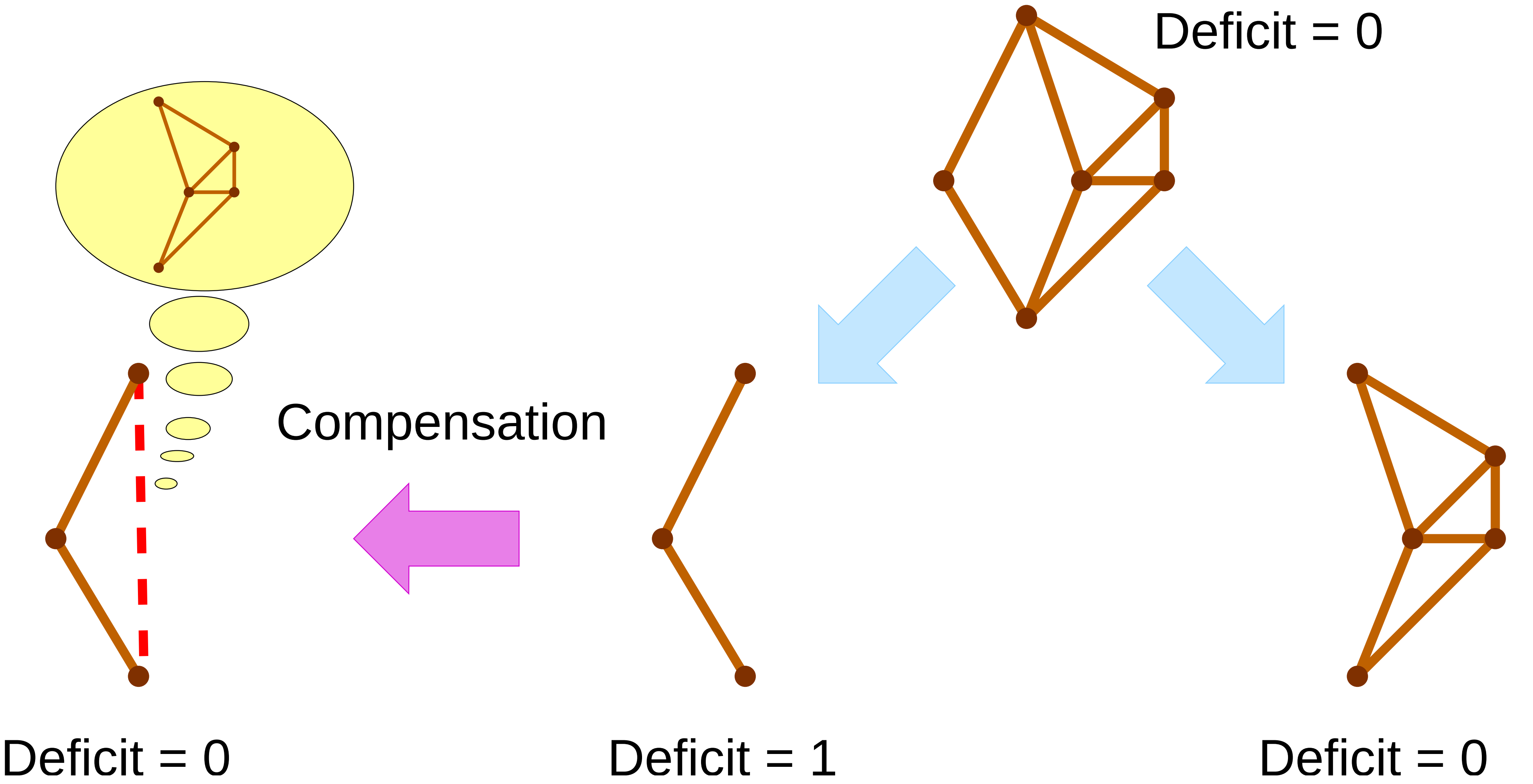
Therefore

> To maintain well-constraintness one virtual edge must be added to the separating graph G'.
>
> The virtual edge subsumes the rigidity properties due to the separating graph G''

Example of deficit compensation



Example of deficit compensation



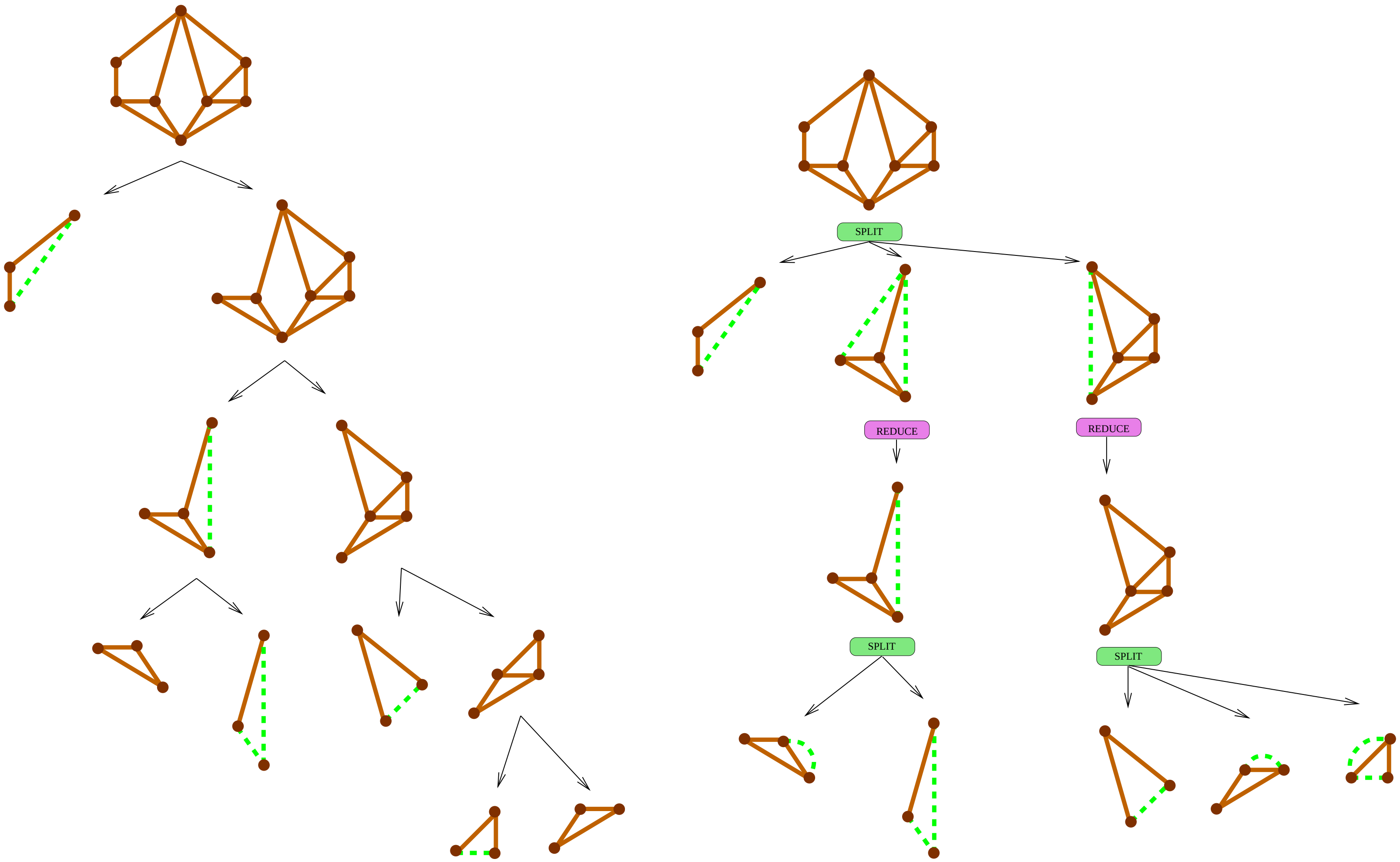
A new formulation of Owen's decomposition algorithm

- A clear and simple application of divide-and-conquer.
- Uses separating pairs to subdivide the graph.
- Applies deficit compensation to maintain rigidity structure.

func Analysis(G)
  if Triconnected(G) **then**
    S := BinaryTree(G, nullTree, nullTree)
  else
    G_1,G_2 := SeparatingGraphs(G)
    if Deficit(G_1) $>$ Deficit(G_2) **then**
      G_1 := AddVirtualEdge(G_1)
    else
      G_2 := AddVirtualEdge(G_2)
    fi
    S := BinaryTree(G, Analysis(G_1),
                Analysis(G_2))
  fi
  return S
end

The result of the new formulation is an s-tree

- The new algorithm yields a binary form of the Owen's tree. We name it a *s-tree*.

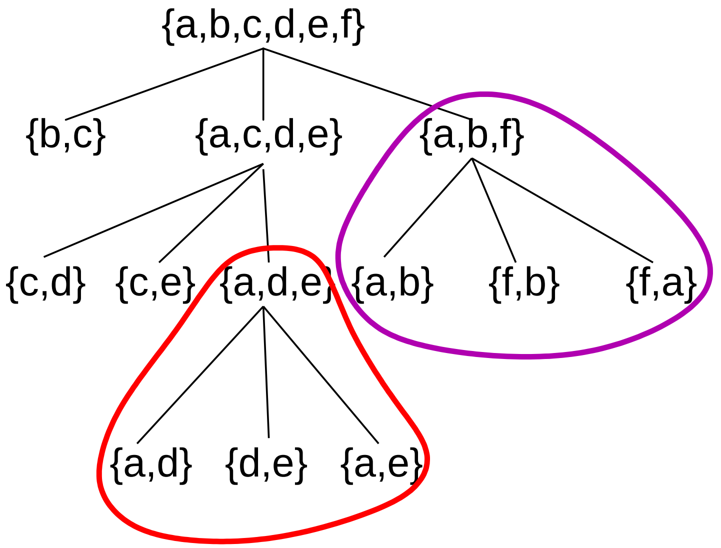


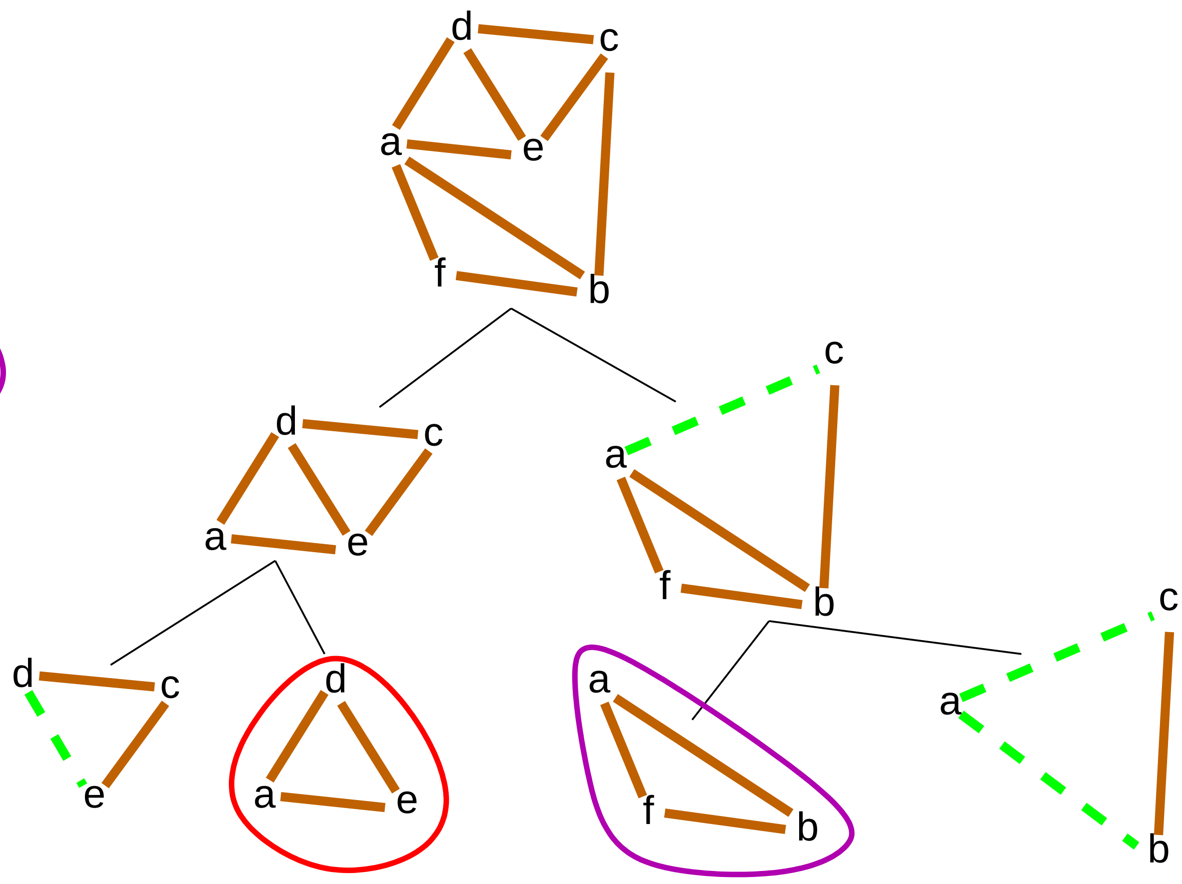
The domain of Owen's method

Let $G = (V, E)$ be a well-constrained geometric constraint graph. The following assertions are equivalent:

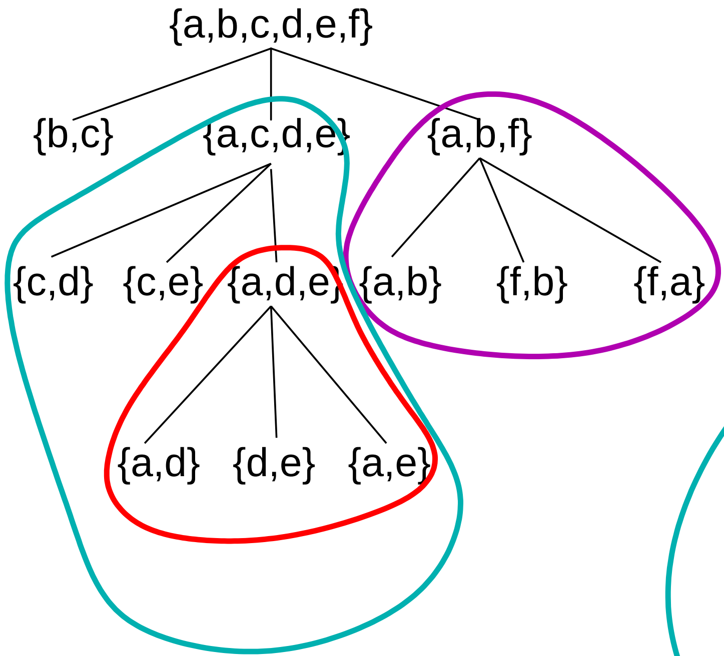
1. G is tree decomposable.
2. G is s-tree decomposable.

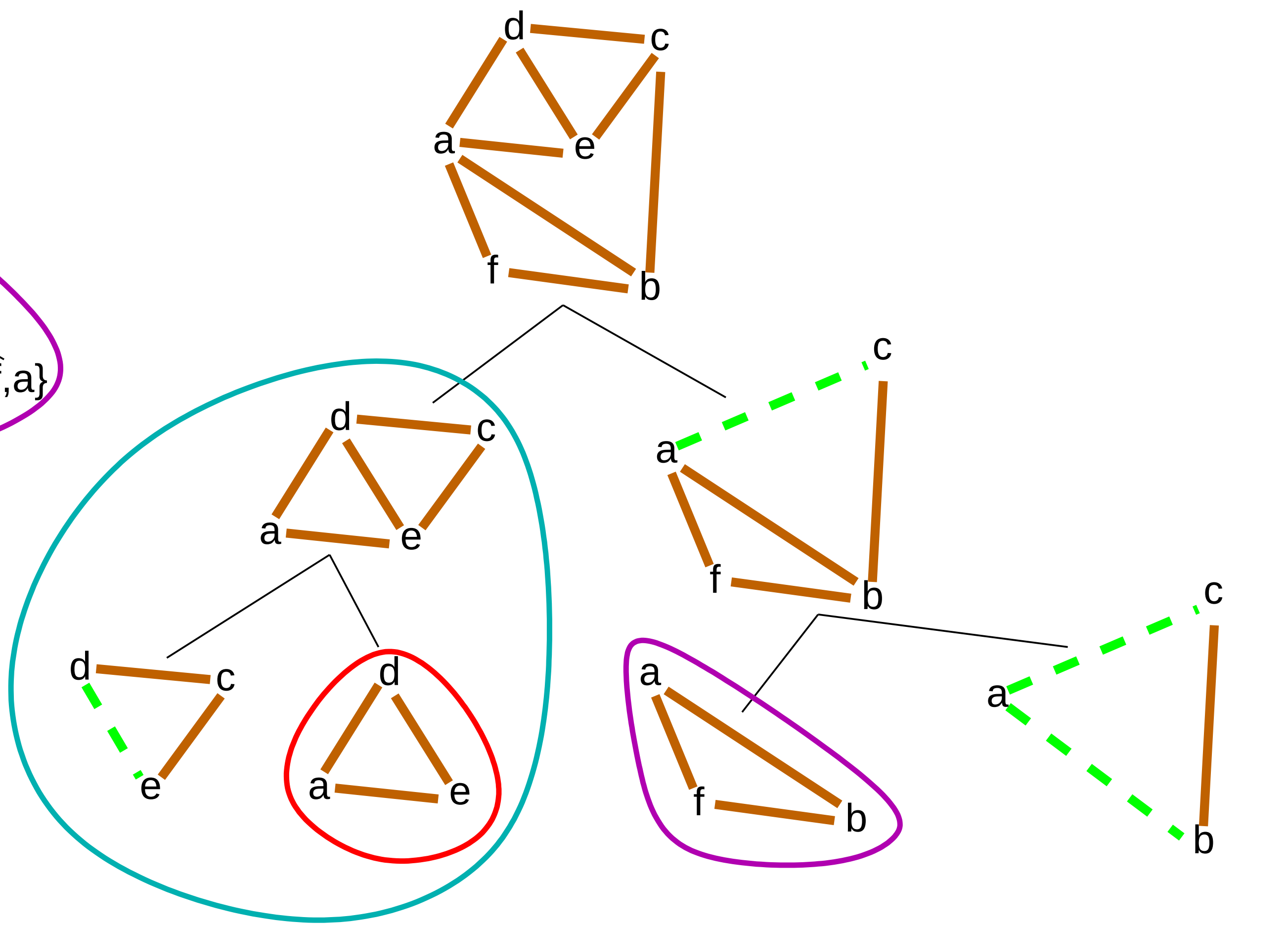
The domain of Owen's method



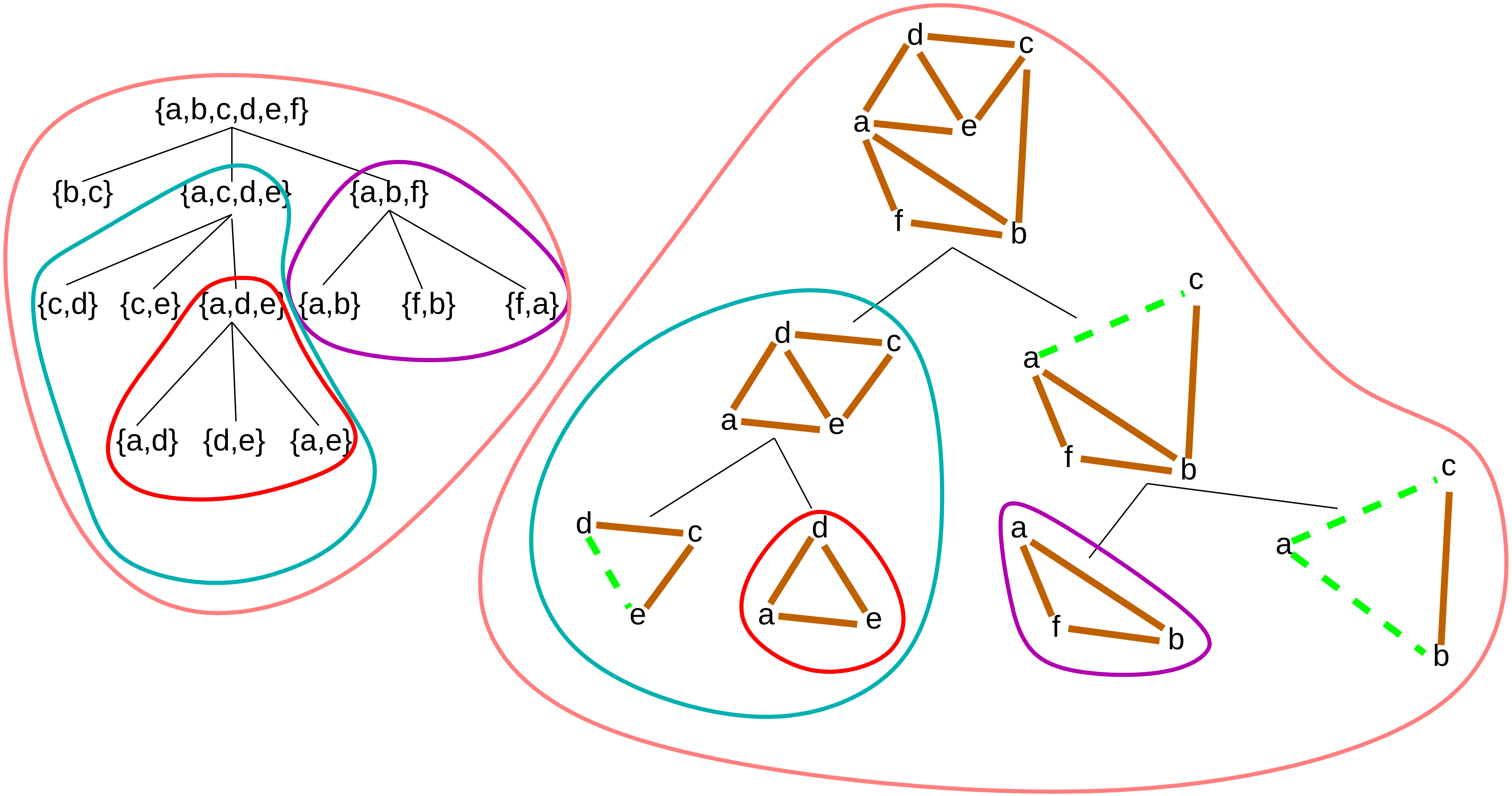


The domain of Owen's method





The domain of Owen's method



Domain equivalence of constructive methods

Constructive methods have the same domain

Let $G = (V, E)$ be a well-constrained geometric constraint graph.
The following assertions are equivalent:

1. G is tree decomposable.
2. G is s-tree decomposable.
3. G is solvable by reduction analysis.
4. G is solvable by decomposition analysis.

The class of graphs fullfiling the above properties is named the *constructively solvable graphs class*.

Summary

- We have introduced the tree decomposition of a graph.
- Tree decomposable graphs characterize the domain of reduction analysis, decomposition analysis and Owen’s method.
- The domains of constructive methods are the same.
- We have clarified and reformulated Owen’s algorithm.
- The reformulated algorithm applies a divide-and-conquer schema and it is conceptually simpler.
- The output of this algorithm is an s-tree.