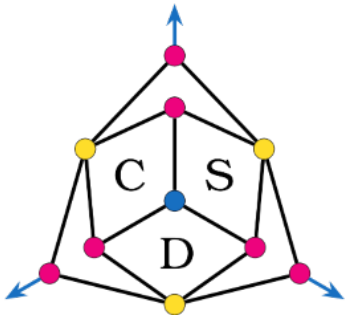
graphotaxy: a graph classification system

Anthony Labarre

Université Gustave Eiffel / LIGM

May 5, 2026



Université Gustave Eiffel

What?

I only work with undirected, unweighted, unlabelled, simple graphs.

What?

I only work with undirected, unweighted, unlabelled, simple graphs.

A **graph class** is a family of graphs that satisfy a given property (bipartite, planar, chordal, . . .).

What?

I only work with undirected, unweighted, unlabelled, simple graphs.

A **graph class** is a family of graphs that satisfy a given property (bipartite, planar, chordal, . . .).

I focus on two problems:

1. **recognition:** given a graph G and a graph class $\mathcal{C}$, decide whether $G \in \mathcal{C}$;

What?

I only work with undirected, unweighted, unlabelled, simple graphs.

A **graph class** is a family of graphs that satisfy a given property (bipartite, planar, chordal, . . .).

I focus on two problems:

1. **recognition:** given a graph G and a graph class $\mathcal{C}$, decide whether $G \in \mathcal{C}$;
2. **classification:** given a graph G, determine all classes to which G belongs.

Why?

Additional knowledge on the structure of graphs under consideration is valuable in various areas (see e.g. all previous talks). Most notably:

- **algorithms:** "hard" problems may become "easy" or "easier" on particular graph classes (NPC $\rightarrow$ P, W $\rightarrow$ FPT, ...);

Why?

Additional knowledge on the structure of graphs under consideration is valuable in various areas (see e.g. all previous talks). Most notably:

- **algorithms:** "hard" problems may become "easy" or "easier" on particular graph classes (NPC $\to$ P, W $\to$ FPT, ...);
- **"pure" graph theory:** how does this new class relate to those famous classes (inclusion, separation, ...)?

Why?

Additional knowledge on the structure of graphs under consideration is valuable in various areas (see e.g. all previous talks). Most notably:

- **algorithms:** "hard" problems may become "easy" or "easier" on particular graph classes (NPC $\rightarrow$ P, W $\rightarrow$ FPT, ...);
- **"pure" graph theory:** how does this new class relate to those famous classes (inclusion, separation, ...)?
- **extremal graph theory:** what is / are the graph class(es) that optimise this invariant's value?

Scope

`graphotaxy` only deals with classes that appear in the ISGCI (*Information System on Graph Classes and their Inclusions*).

https://www.graphclasses.org/

Scope

`graphotaxy` only deals with classes that appear in the ISGCI (*Information System on Graph Classes and their Inclusions*).

https://www.graphclasses.org/

This is "an encyclopaedia of graph classes", which features a wealth of information for each class. Most importantly for me:

1. its recognition status (in P, NP-complete, or unknown);
2. its minimal superclasses and its maximal subclasses in ISGCI;

Challenges

There are quite a few hurdles to obtaining a comprehensive classification. I want to:

Challenges

There are quite a few hurdles to obtaining a comprehensive classification. I want to:

1. **maximise coverage:**

Challenges

There are quite a few hurdles to obtaining a comprehensive classification. I want to:

1. **maximise coverage:**

2. **minimise running time:**

Challenges

There are quite a few hurdles to obtaining a comprehensive classification. I want to:

1. **maximise coverage:** but:
   - × ISGCI contains 1 690 classes (and counting);

2. **minimise running time:**

Challenges

There are quite a few hurdles to obtaining a comprehensive classification. I want to:

1. **maximise coverage:** but:
   - × ISGCI contains 1 690 classes (and counting);
   - × many of them are hard to recognise;
2. **minimise running time:**

Challenges

There are quite a few hurdles to obtaining a comprehensive classification. I want to:

1. **maximise coverage:** but:
   - × ISGCI contains 1 690 classes (and counting);
   - × many of them are hard to recognise;
   - × many algorithms exist only on paper;
2. **minimise running time:**

Challenges

There are quite a few hurdles to obtaining a comprehensive classification. I want to:

1. **maximise coverage:** but:
   - × ISGCI contains 1 690 classes (and counting);
   - × many of them are hard to recognise;
   - × many algorithms exist only on paper;
2. **minimise running time:** but:
   - × polynomial does not mean "affordable" (there are $O(|V|^{15})$ recognition algorithms);

Challenges

There are quite a few hurdles to obtaining a comprehensive classification. I want to:

1. **maximise coverage:** but:
   - × ISGCI contains 1 690 classes (and counting);
   - × many of them are hard to recognise;
   - × many algorithms exist only on paper;
2. **minimise running time:** but:
   - × polynomial does not mean "affordable" (there are $O(|V|^{15})$ recognition algorithms);
   - × I don't want to run *all* recognition algorithms on *all* input graphs;

Challenges

There are quite a few hurdles to obtaining a comprehensive classification. I want to:

1. **maximise coverage:** but:
   - × ISGCI contains 1 690 classes (and counting);
   - × many of them are hard to recognise;
   - × many algorithms exist only on paper;
2. **minimise running time:** but:
   - × polynomial does not mean "affordable" (there are $O(|V|^{15})$ recognition algorithms);
   - × I don't want to run *all* recognition algorithms on *all* input graphs;

But let's not give up just yet.

Coverage

Good news: some classes:

- are equivalent (e.g., bipartite $\equiv$ odd-cycle-free), so I "only" need to handle 1 219 classes;

Coverage

Good news: some classes:

- are equivalent (e.g., bipartite $\equiv$ odd-cycle-free), so I "only" need to handle 1 219 classes;
- admit a **finite** forbidden induced subgraph characterisation, which allows for a naïve recognition algorithm;

Coverage

Good news: some classes:

- are equivalent (e.g., bipartite $\equiv$ odd-cycle-free), so I "only" need to handle 1 219 classes;
- admit a **finite** forbidden induced subgraph characterisation, which allows for a naïve recognition algorithm;
- depend on others (e.g., chordal $\cap$ planar);

Coverage

Good news: some classes:

- are equivalent (e.g., bipartite $\equiv$ odd-cycle-free), so I "only" need to handle 1 219 classes;
- admit a **finite** forbidden induced subgraph characterisation, which allows for a naïve recognition algorithm;
- depend on others (e.g., chordal $\cap$ planar);
- are complementary (e.g., G is co-bipartite $\Leftrightarrow$ $\overline{G}$ is bipartite);

Coverage

Good news: some classes:

- are equivalent (e.g., bipartite $\equiv$ odd-cycle-free), so I "only" need to handle 1 219 classes;
- admit a **finite** forbidden induced subgraph characterisation, which allows for a naïve recognition algorithm;
- depend on others (e.g., chordal $\cap$ planar);
- are complementary (e.g., G is co-bipartite $\Leftrightarrow$ $\overline{G}$ is bipartite);

As of today, 728 recognisers are currently implemented, covering 1069 classes out of 1690 (63.25 % coverage).

Coverage

`graphotaxy` provides two options to communicate additional knowledge:

- `--negative ids`: classes to which no input graph belongs;
- `--positive ids`: classes to which all input graphs belong;

Coverage

`graphotaxy` provides two options to communicate additional knowledge:

- `--negative ids`: classes to which no input graph belongs;
- `--positive ids`: classes to which all input graphs belong;

You can also restrict the scope of `graphotaxy` using these two options:

- `--only ids`: classes to which the classification must be restricted;
- `--skip ids`: classes whose recognition should be skipped;

(use ISGCI ids: "bipartite" = gc_69, "perfect" = gc_56, ...)

Efficiency

Critical ingredients allow me to speed up the classification process:

1. ordering recognisers by increasing complexity;

Efficiency

Critical ingredients allow me to speed up the classification process:

1. ordering recognisers by increasing complexity;
2. extensive use of caching;

Efficiency

Critical ingredients allow me to speed up the classification process:

1. ordering recognisers by increasing complexity;
2. extensive use of caching;
3. inclusion relationships between classes;

Efficiency

Critical ingredients allow me to speed up the classification process:

1. ordering recognisers by increasing complexity;
2. extensive use of caching;
3. inclusion relationships between classes;
4. exclusion relationships between classes;

Efficiency

Critical ingredients allow me to speed up the classification process:

1. ordering recognisers by increasing complexity;
2. extensive use of caching;
3. inclusion relationships between classes;
4. exclusion relationships between classes;
5. forbidden induced subgraph characterisations;

Efficiency

Critical ingredients allow me to speed up the classification process:

1. ordering recognisers by increasing complexity;
2. extensive use of caching;
3. inclusion relationships between classes;
4. exclusion relationships between classes;
5. forbidden induced subgraph characterisations;

What's better than an efficient recognition algorithm?

Efficiency

Critical ingredients allow me to speed up the classification process:

1. ordering recognisers by increasing complexity;
2. extensive use of caching;
3. inclusion relationships between classes;
4. exclusion relationships between classes;
5. forbidden induced subgraph characterisations;

What's better than an efficient recognition algorithm?
An algorithm you don't run.

Inclusion relationships

I build the **ISGCI inclusion digraph** $\mathscr{G} = (V, A)$, with:

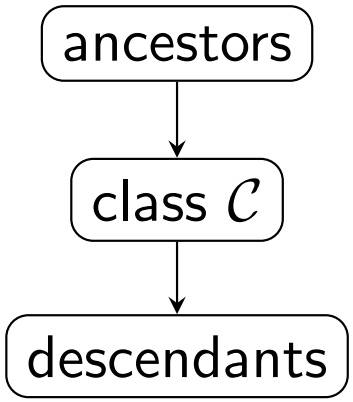
- $V =$ all graph classes in ISGCI;
- A contains an arc $(\mathcal{C}, \mathcal{D})$ iff $\mathcal{D}$ is a maximal subclass of $\mathcal{C}$.

Inclusion relationships

I build the **ISGCI inclusion digraph** $\mathscr{G} = (V, A)$, with:

- V = all graph classes in ISGCI;
- A contains an arc $(\mathcal{C}, \mathcal{D})$ iff $\mathcal{D}$ is a maximal subclass of $\mathcal{C}$.

Every answer helps:

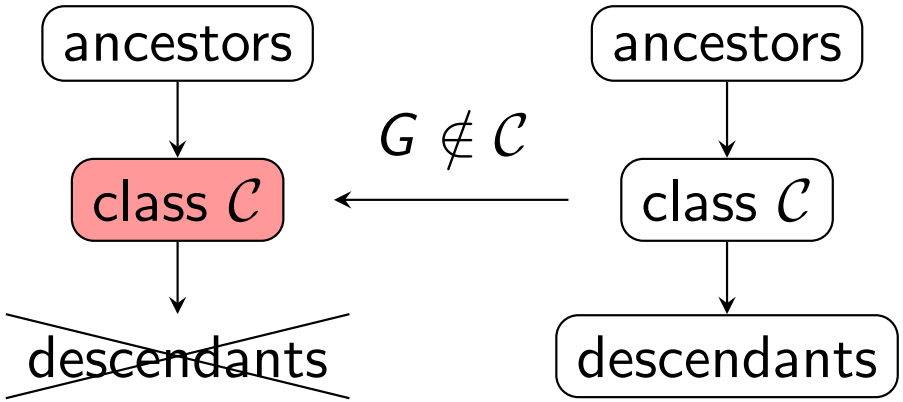


Inclusion relationships

I build the **ISGCI inclusion digraph** $\mathscr{G} = (V, A)$, with:

- V = all graph classes in ISGCI;
- A contains an arc $(\mathcal{C}, \mathcal{D})$ iff $\mathcal{D}$ is a maximal subclass of $\mathcal{C}$.

Every answer helps:

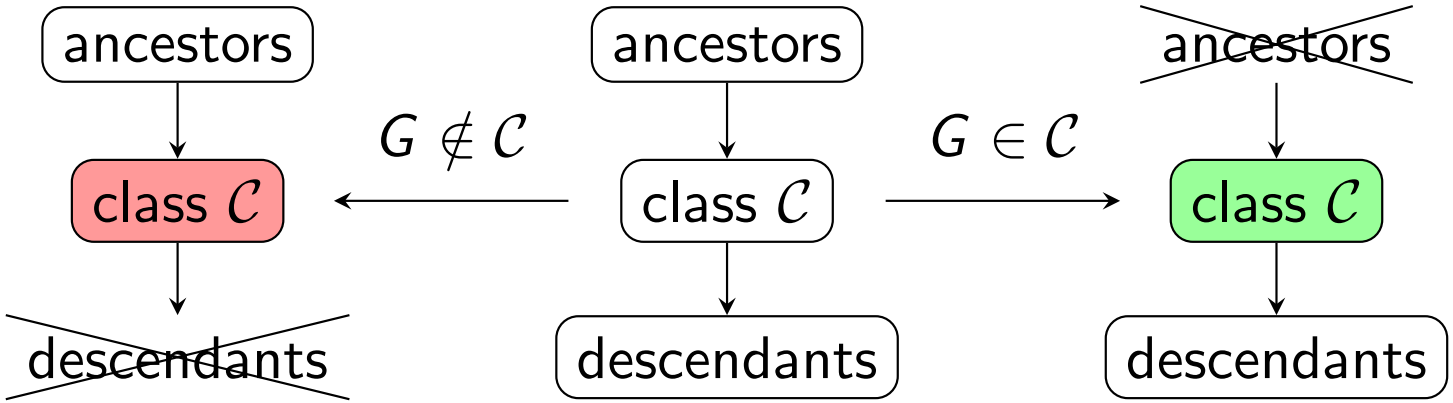


Inclusion relationships

I build the **ISGCI inclusion digraph** $\mathscr{G} = (V, A)$, with:

- V = all graph classes in ISGCI;
- A contains an arc $(\mathcal{C}, \mathcal{D})$ iff $\mathcal{D}$ is a maximal subclass of $\mathcal{C}$.

Every answer helps:

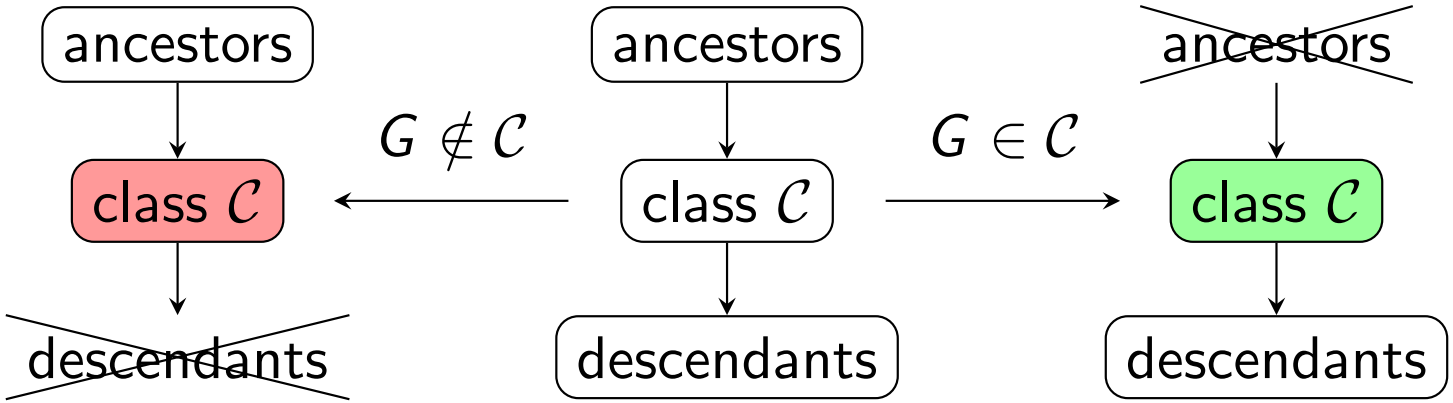


Inclusion relationships

I build the **ISGCI inclusion digraph** $\mathscr{G} = (V, A)$, with:

- V = all graph classes in ISGCI;
- A contains an arc $(\mathcal{C}, \mathcal{D})$ iff $\mathcal{D}$ is a maximal subclass of $\mathcal{C}$.

Every answer helps:



These **propagations** allow me to avoid running many recognition algorithms at all.

Exclusion relationships

Besides inclusion relationships (i.e., "$G \in \mathcal{C} \Rightarrow G \in \mathcal{D}$"), I also take advantage of **exclusion relationships** (i.e., "$G \in \mathcal{C} \Rightarrow G \notin \mathcal{D}$").

Exclusion relationships

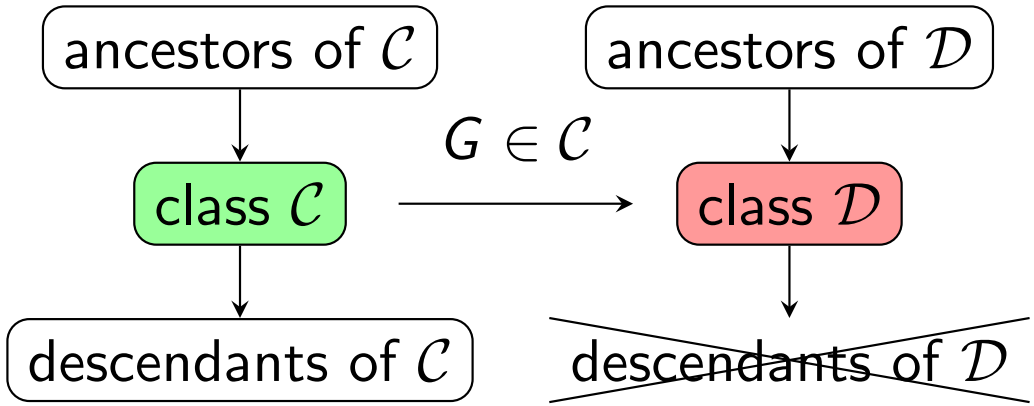
Besides inclusion relationships (i.e., "$G \in \mathcal{C} \Rightarrow G \in \mathcal{D}$"), I also take advantage of **exclusion relationships** (i.e., "$G \in \mathcal{C} \Rightarrow G \notin \mathcal{D}$"). For instance:

- k-regular for $k > 1 \Rightarrow \neg$ tree (trivial);
- unbreakable $\Rightarrow \neg$ P_4-free (less trivial);
- ⋮

ISGCI doesn't provide these, so collecting them takes some time (**please send me yours!**).

Exclusion relationships

The idea is similar to what we've seen regarding inclusion relationships: if $G \in \mathcal{C} \Rightarrow G \notin \mathcal{D}$, then:



And so I can skip even more classes in addition to the earlier propagations.

Forbidden induced subgraphs

Given a set S of graphs, a graph G is said to be S-**free** if it contains no induced subgraph isomorphic to an element of S.

Forbidden induced subgraphs

Given a set S of graphs, a graph G is said to be S-**free** if it contains no induced subgraph isomorphic to an element of S.

A graph class that can be completely characterised by such a set of "forbidden" induced subgraphs is said to admit a **FISC (forbidden induced subgraph characterisation)**; e.g.:

- cograph $\equiv P_4$-free;
- split $\equiv (2K_2, C_4, C_5)$-free;

⋮

Forbidden induced subgraphs

Given a set S of graphs, a graph G is said to be S-**free** if it contains no induced subgraph isomorphic to an element of S.

A graph class that can be completely characterised by such a set of "forbidden" induced subgraphs is said to admit a **FISC (forbidden induced subgraph characterisation)**; e.g.:

- cograph $\equiv P_4$-free;
- split $\equiv (2K_2, C_4, C_5)$-free;

⋮

$\Rightarrow$ naïve recognition algorithm: for every H in S, check whether any $|V_H|$-subset of V_G induces a subgraph isomorphic to H;

✓ good for coverage (about 400 such classes);

✗ slow – but let's work on that.

Exploiting FISCs

- FISCs in ISGCI are based on "smallgraphs" shared by many classes;

Exploiting FISCs

- FISCs in ISGCI are based on "smallgraphs" shared by many classes;
- `graphotaxy` keeps track of the status of each smallgraph in each input graph (found, missing, or unknown);

Exploiting FISCs

- FISCs in ISGCI are based on "smallgraphs" shared by many classes;
- `graphotaxy` keeps track of the status of each smallgraph in each input graph (found, missing, or unknown);
- (Even partial) FISCs can be associated to each recogniser, so I can:

Exploiting FISCs

- FISCs in ISGCI are based on "smallgraphs" shared by many classes;
- `graphotaxy` keeps track of the status of each smallgraph in each input graph (found, missing, or unknown);
- (Even partial) FISCs can be associated to each recogniser, so I can:
  1. gather information on smallgraphs without searching for them (e.g., P_4-free graphs can be recognised in linear time);

Exploiting FISCs

- FISCs in ISGCI are based on "smallgraphs" shared by many classes;
- `graphotaxy` keeps track of the status of each smallgraph in each input graph (found, missing, or unknown);
- (Even partial) FISCs can be associated to each recogniser, so I can:
  1. gather information on smallgraphs without searching for them (e.g., P_4-free graphs can be recognised in linear time);
  2. propagate results in a fashion similar to what we've seen for graph classes;

Exploiting FISCs

- FISCs in ISGCI are based on "smallgraphs" shared by many classes;
- `graphotaxy` keeps track of the status of each smallgraph in each input graph (found, missing, or unknown);
- (Even partial) FISCs can be associated to each recogniser, so I can:
  1. gather information on smallgraphs without searching for them (e.g., P_4-free graphs can be recognised in linear time);
  2. propagate results in a fashion similar to what we've seen for graph classes;
  3. avoid running other recognisers entirely, by first inspecting their FISC;

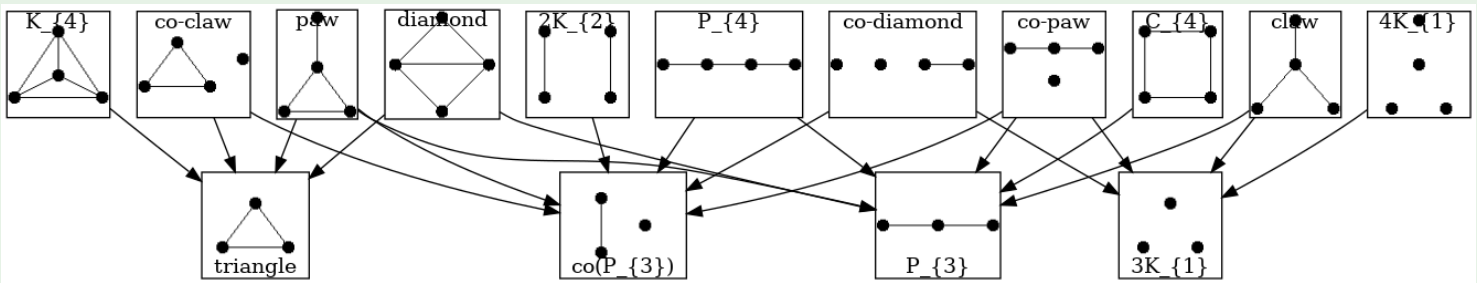
Propagations for patterns

I build a **smallgraph inclusion digraph** $\mathscr{S} = (V, A)$, where:

- $V \supseteq$ all smallgraphs in ISGCI ($|V| = 611$);
- A contains an arc (G, H) iff H is an induced subgraph of G.

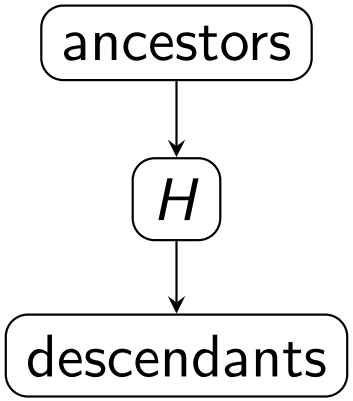
Example

The subgraph of $\mathscr{S}$ induced by all smallgraphs of sizes 3 and 4:



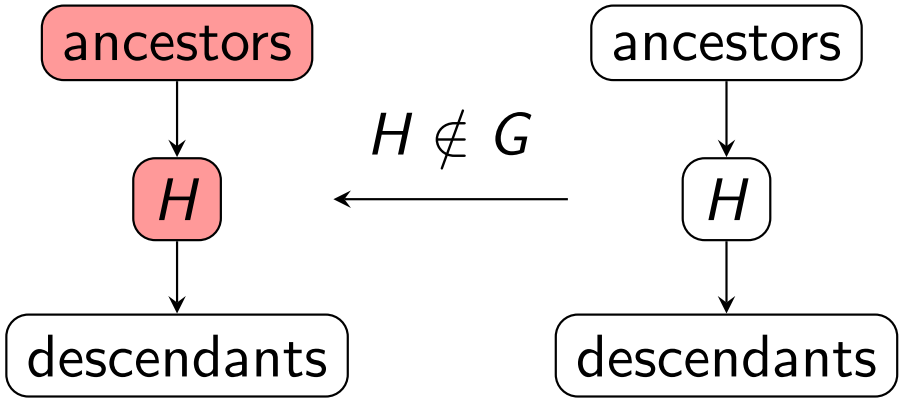
Propagations for patterns

Again, every answer helps:



Propagations for patterns

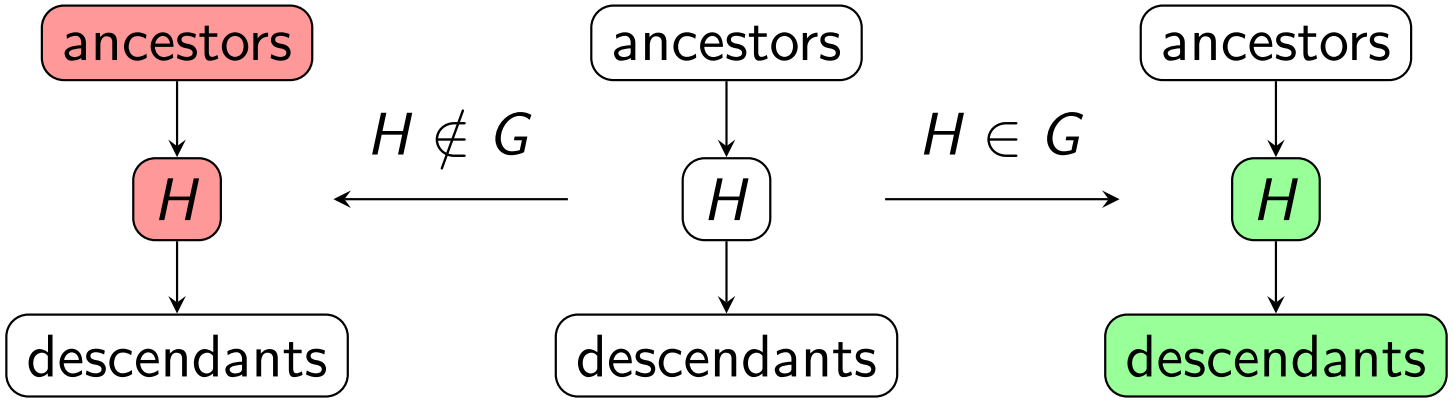
Again, every answer helps:



1. if $H \notin G$, then neither does any of its supergraphs;

Propagations for patterns

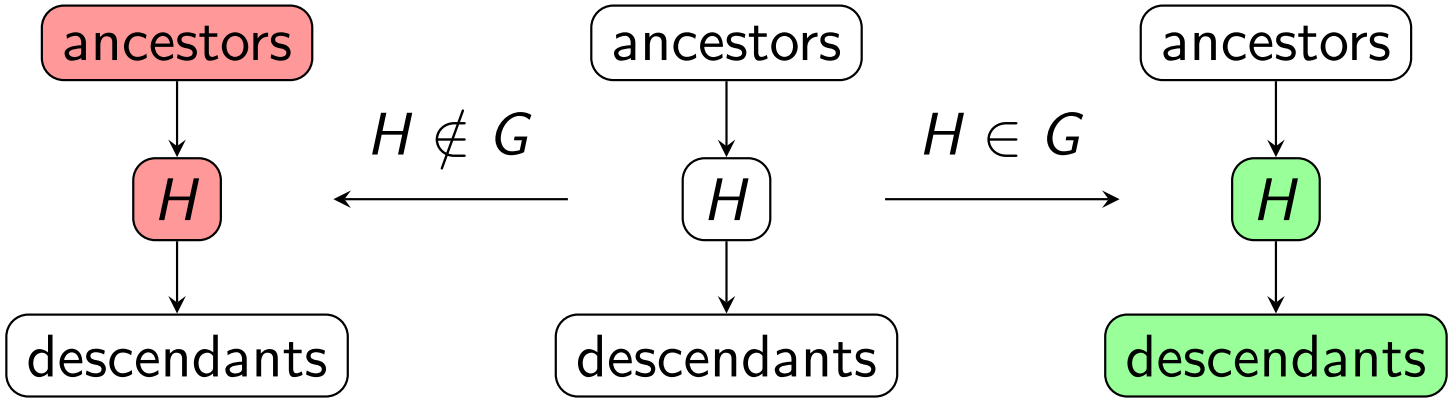
Again, every answer helps:



1. if $H \notin G$, then neither does any of its supergraphs;
2. if $H \in G$, then so do all of H's induced subgraphs;

Propagations for patterns

Again, every answer helps:



1. if $H \notin G$, then neither does any of its supergraphs;
2. if $H \in G$, then so do all of H's induced subgraphs;

When there's really no way around using the naïve algorithm, I launch a better tool: the Glasgow Subgraph Solver.

https://github.com/ciaranm/glasgow-subgraph-solver

Demos

Let's have a look at:

1. **single graph classification:** the Cayley graph of $\mathfrak{S}_5$ generated by prefix reversals ("pancake flipping");
2. **multiple graph classification:** all self-complementary graphs (i.e., graphs isomorphic to their complement) on 9 vertices;
3. **visualising a single graph classification:** (using external tools);

Thanks for your attention!

You can check out `graphotaxy` here:

https://github.com/alabarre/graphotaxy

Feedback is welcome and highly valued.