6. Treewidth

Goal: Algorithmic verification of concurrent programs with recursion

Problem: Model is Turing complete, 2 stacks = 1 tape of a TM.
↳ Everything is undecidable.

Approach: Consider under-approximations of the system behavior
↳ Good for bug hunting
- if there is a bug in the under-approximation, there is one in the full system
- if there is no bug in the under-approximation, we cannot conclude safety of the full system.

Paradigm shift in the analysis: From configurations to computations

Before:
- Focused on the set of reachable configurations
- Developed techniques for finite representation and efficient approximation (linear algebra). (particles slope, e.g. upward-closed)

Now:
- Focus on the set of feasible computations
- Show that they (or an under-approximation of this set) have a simple shape
  ↳ Bounded treewidth
- Can be processed efficiently by means of automata.

Outline:
- Treewidth
  ↳ Brambles for lower bounds
  ↳ Cops & robbers for upper bounds

- Courcelle's theorem
  - ↳ MSO interpretations
  - ↳ MSO is decidable on graphs of bounded treewidth
  - ↳ Converse (Seese):
    If MSO is decidable,
    then the graphs have bounded treewidth.
- From concurrent pushdowns to Courcelle.
  - ↳ Bounded context switching yields bounded treewidth
  - ↳ Reduction
  - ↳ Extensions.

6.1 Treewidth

Goal: - Introduce the notion of treewidth
- Develop techniques to show lower and upper bounds for classes of graphs.

Idea: - Say <u>how tree-like</u> a graph is.
- Generalize techniques from trees to graphs.

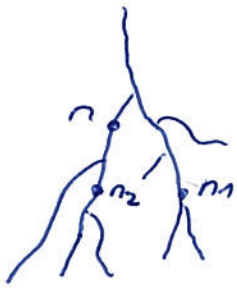
<u>Definition (Tree decomposition)</u>:
- A <u>tree decomposition</u> of a graph $G=(V,E)$ is a node-labelled tree $T=(N,\to,B)$ with $B: N \to \mathbb{P}(V)$ (called <u>bags</u>) so that
  (1) $\bigcup_{n\in N} B(n) = V$
  (2) For every edge $\{x,y\}\in E$ there is an $n\in N$ with $\{x,y\}\subseteq B(n)$.

(3) Interpolation property:

Let n be a node on the unique path from n_1 to n_2.

Then $B(n_1) \cap B(n_2) \subseteq B(n)$.

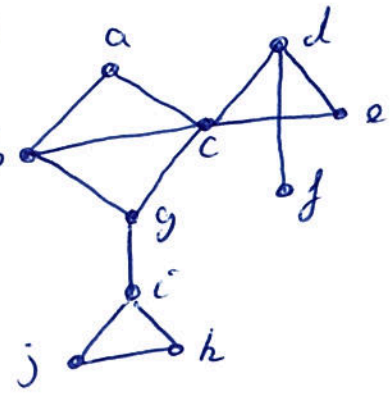
Illustration:



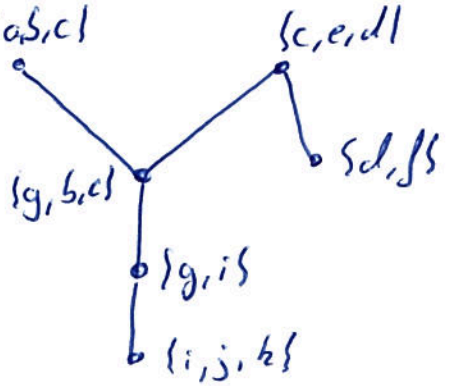
- The width of the tree decomposition is $\max\{|B(n)| \mid n \in N\} - 1$.
- Graph G has treewidth $k \in \mathbb{N}$, denoted $tw(G) = k$, if
  - there is a tree decomposition of width k and
  - there is no tree decomposition of width $k-1$.
- A class of graphs K has bounded treewidth if $\exists k \in \mathbb{N}\ \forall G \in K: tw(G) \leq k$.

Examples:

(1) $G =$



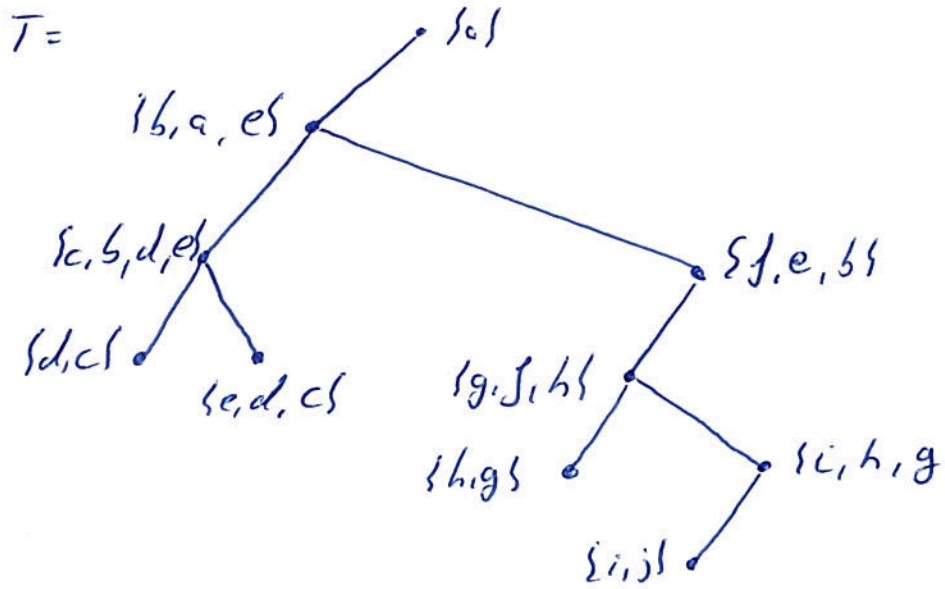
$T =$



We have
- $width(T) = 2$
- hence $tw(G) \leq 2$.
- Actually, $tw(G) = 2$.

(2) $G =$

$T =$



We have
- $width(T) = 3$,
- hence $tw(G) \leq 3$.
- Actually, $tw(G) = 2$.

Problem:
- Determining the treewidth for a class of graphs is challenging
  ↳ Have to find a strategy how to decompose each graph in the class.

Approach: Provide techniques for proving lower and upper bounds on the treewidth.

6.2 Brambles

Goal: Prove lower bounds on the treewidth.

Definition (Bramble):
- A bramble for $G = (V, E)$ is
  a family of connected subgraphs $(G_i)_{i \in I}$ with $G_i = (V_i, E_i)$
  that all (pairwise) touch each other
  ↳ overlap or
  ↳ connected by an edge.

- The order of a bramble is defined to be the size of the smallest hitting set:

$H \subseteq V$: $H \cap V_i \neq \emptyset$ for all $1 \leq i \leq n$.

Theorem (Seymour & Thomas '93):

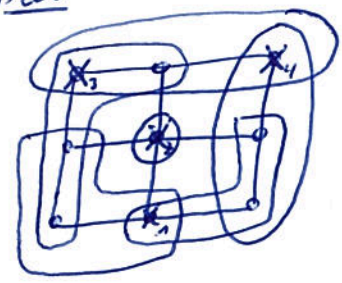
A graph G has a bramble of order k iff $tw(G) \geq k-1$.

Intuitively:

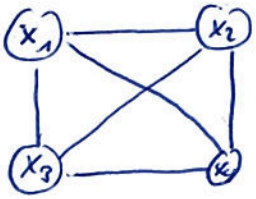
If a graph G has a bramble of order k_1 then G "embeds" a clique of order k. Cliques of order k have treewidth $k-1$. (more precisely K_k)

Examples:

(1)



order $(B) = 4$



(2) Higher-order pushdown computations have unbounded treewidth, already of order 2.

A higher-order pushdown system has order 2 = stacks of stacks
order 3 = stacks of (stacks of stacks)
order 4 = stacks of order-3 stacks.
?

Focus on order 2.

Operations:

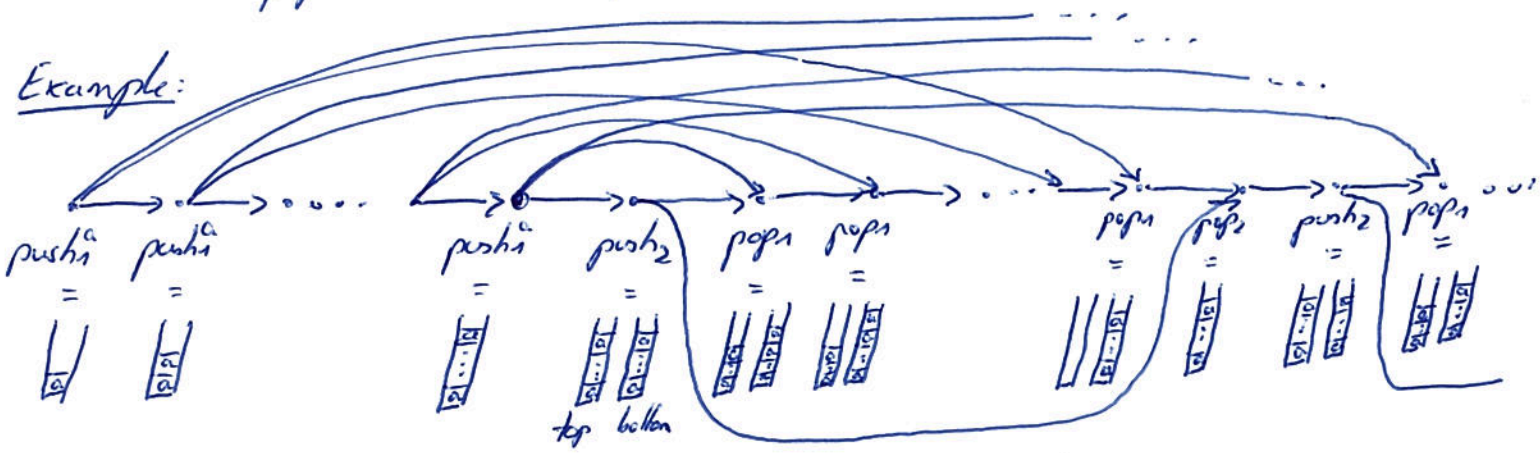
$push_1^a$ = push a on topmost stack

pop_1 = pop topmost stack

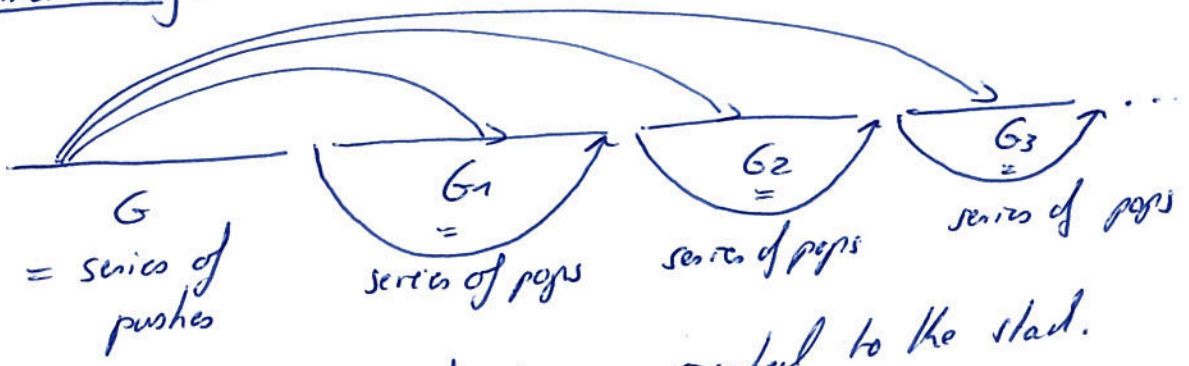
$push_2$ = create a copy of the topmost stack

pop_2 = remove topmost stack.

Example:



Schematically:



↳ In G_1, a number of a's is pushed to the stack.

↳ Then in G_1, the stack is copied and the a's are consumed until the stack is empty.

↳ The now empty stack is popped and a new copy is created

In G_2, again the a's are consumed.

↳ When the stack is empty, we again pop and create a new copy.

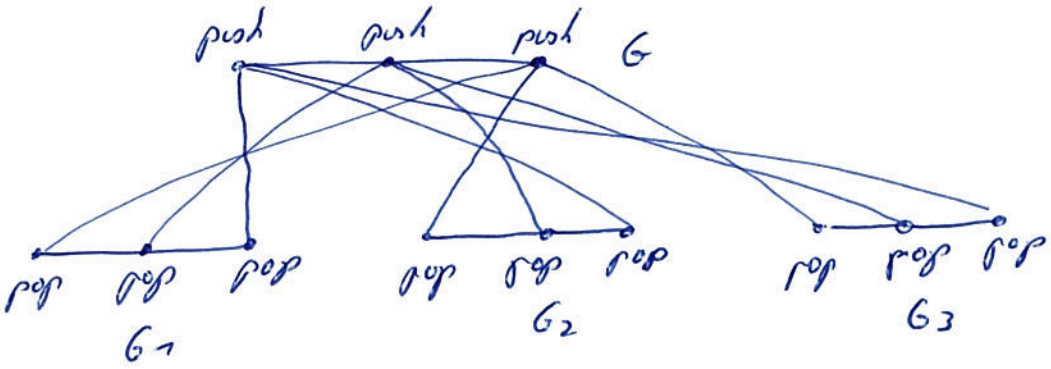
Theorem (A., Ong, Tsukada, unpublished):

The graphs of higher-order computations do not have bounded treewidth, already at order 2.

Proof:

Consider the above computation and assume we have as many copies G_i as G has pushes.

Represent the graph as follows (3 pushes, for simplicity):

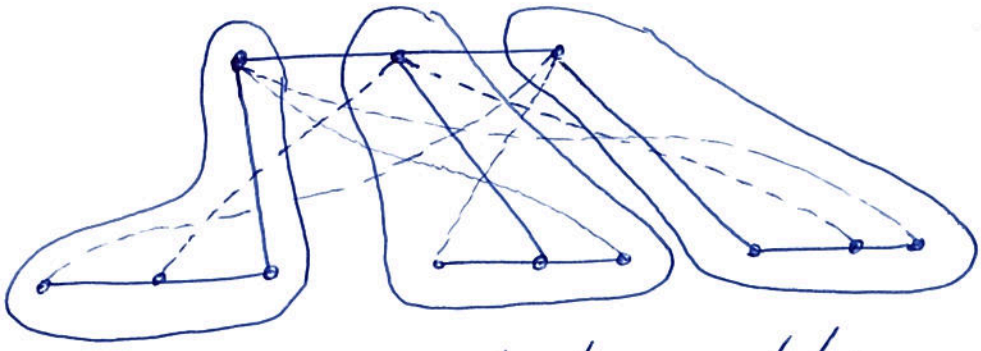


We associate a distinct push in G with each of the G_i.

↳ This yields a connected subgraph

↳ Moreover it works as we assume as many G_is as we have pushes.

The result is a <u>bramble</u>:



The dotted edges indicate connectedness of the resulting subgraphs.

⇒ The smallest hitting set has to hit every subgraph.

⇒ Since the number of subgraphs grows with the number of pushes, and since there is no bound on the pushes, by Seymour and Thomas there is no bound on the treewidth. □

Open:

↳ Do we obtain bounded treewidth
if we assume an upper bound $k \in \mathbb{N}$
on how often an element may be popped?

↳ Can we afford $k \in \mathbb{N}$ elements that are popped
an unbounded number of times
while the remaining elements are popped at most k times?

6.3 Cops and Robbers

Goal: Prove upper bounds on the treewidth

Rules: Cops and robber placed on the nodes of a graph

^{r}↳ Robber can, at any time, run at great speed
along the paths of the graph

^{r}↳ Robber cannot run through cops

^{c}↳ Cops either stand on a node
or fly by helicopter to another node

^{c}↳ Goal is to land on a node where robber is.

Definition (Cops and robbers game):

Let $G = (V, E)$ and $k \in \mathbb{N}$.

↳ A configuration in the cops and robbers game
is a pair
(C, r) with $C \subseteq V$ and $|C| = k$,
and $r \in V \setminus C$.

↳ The initial configuration is arbitrary.

↳ To move from configuration (C_i, r_i) to (C_{i+1}, r_{i+1})

the cops choose

$C_{i+1} \subseteq V$ with $|C_{i+1}| = k$

and the robber chooses

$r_{i+1} \in V \setminus C_{i+1}$ so that

there is a path from r_i to r_{i+1} in $V \setminus (C_i \cap C_{i+1})$.

↳ The cops <u>win</u> if, after some rounds, the robber has no vertex to choose.

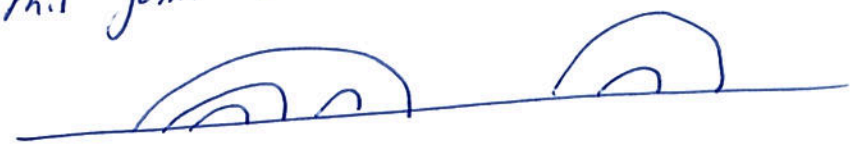
<u>Theorem (Seymour & Thomas '93)</u>:

Graph G has treewidth $\leq k$

iff $k+1$ cops can catch the robber.

<u>Example</u>:

- Graphs of pushdown computations have treewidth 2.
- Show that 3 cops can catch the robber on a graph of this form:



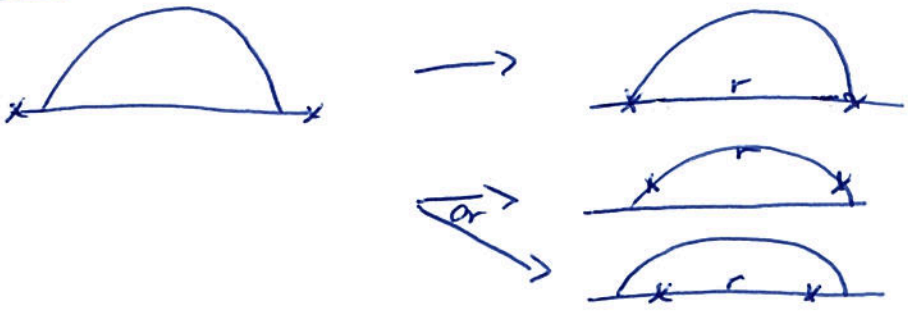
<u>Case 1:</u>

↳ Halfen the interval

↳ Continue searching in the correct half

=> The cops are also called <u>omniscient</u>.

<u>Case 2:</u>



Continue searching on top or bottom, depending on where root is.

Question:

Do graphs corresponding to computations of automata always admit such positional strategies?