Courcelle's Theorem Made Dynamic

Patricia Bouyer-Decitre[1], Vincent Jugé[1] & Nicolas Markey[1,2]

1: LSV, CNRS & ENS Paris-Saclay 2: IRISA, CNRS & Inria & Université Rennes 1

Dynamic decision problems

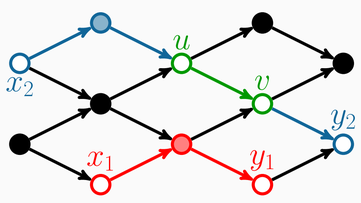
Context: Given a decision problem, at what cost can we update our decision when one bit of the problem input is modified?

Dynamic complexity class: If precomputing auxiliary data in $\mathcal{C}$ helps us treating input updates in $\mathcal{C}'$, we say that the dynamic problem is in $\mathsf{Dyn}(\mathcal{C},\mathcal{C}')$.

Example: Reachability in acyclic graphs is in Dyn(NL,FO) [5]

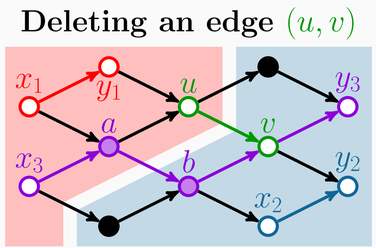
Decision problem: Given two vertices s, t of an acyclic graph $G = (V, E)$, does there exist a path from s to t in G?
Input updates: Edge deletion or insertion (without creating cycles)
Auxiliary predicate: $\mathbf{R}(x, y)$ = "There exists a path from x to y".

Inserting an edge (u, v)



$\mathbf{R}(x, y) \leftarrow \mathbf{R}(x, y) \vee (\mathbf{R}(x, u) \wedge \mathbf{R}(v, y))$

Deleting an edge (u, v)



$\mathbf{R}(x, y) \leftarrow (\mathbf{R}(x, y) \wedge \mathbf{R}(y, u)) \vee (\mathbf{R}(x, y) \wedge \neg\mathbf{R}(x, u)) \vee (\exists (a, b) \neq (u, v)$ s.t. $\mathbf{R}(x, a) \wedge \mathbf{R}(b, y) \wedge \mathbf{E}(a, b) \wedge \mathbf{R}(a, u) \wedge \neg\mathbf{R}(b, u))$

Courcelle's theorem

Ingredients: A graph $G = (V, E)$, a tree-decomposition $\mathcal{D}$ of width κ of G, a succinct encoding **enc** of $\mathcal{D}$ and an MSO formula φ

Tree-decomposition of width κ of G: Pair $\mathcal{D} = \langle \mathcal{T}, \mathbf{bag}\rangle$, where $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ is an ordered binary tree and **bag** is a mapping $\mathcal{N} \mapsto 2^V$ such that:

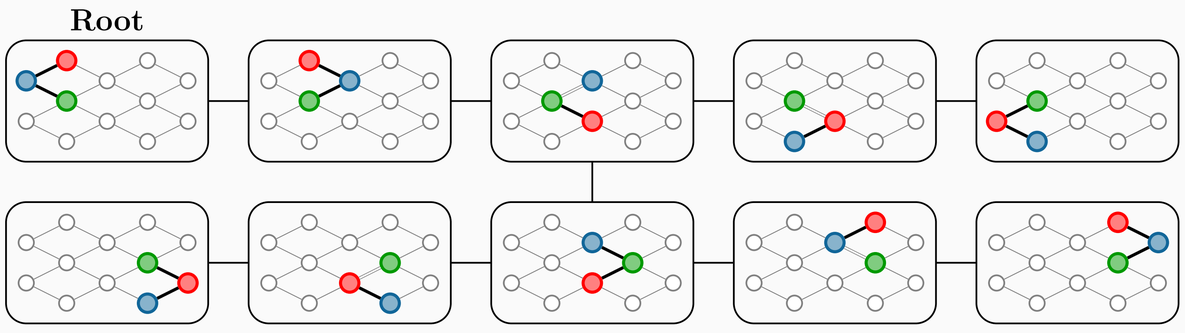
1. for each vertex $v \in V$, the set $\{n \in \mathcal{N} : v \in \mathbf{bag}(n)\}$ is connected and non-empty;
2. for each edge $e = (v_1, v_2) \in E$, the set $\{n \in \mathcal{N} : \{v_1, v_2\} \subseteq \mathbf{bag}(n)\}$ is non-empty;
3. for each node $n \in \mathcal{N}$, the set $\mathbf{bag}(n)$ is of cardinality at most $\kappa + 1$.

Succinct encoding of $\mathcal{D}$: Triple $\mathbf{enc} = \langle \chi, \lambda^v, \lambda^e\rangle$, where $\chi : V \mapsto \{0, \ldots, \kappa\}$, $\lambda^v : \mathcal{N} \mapsto 2^{\{0,\ldots,\kappa\}}$ and $\lambda^e : \mathcal{N} \mapsto 2^{\{0,\ldots,\kappa\}^2}$ are mappings such that, for each node $n \in \mathcal{N}$:

1. the restriction of χ to $\mathbf{bag}(n)$ is injective (hence χ is a proper coloring of G);
2. $\lambda^v(n) = \{\chi(v) : v \in \mathbf{bag}^\star(n)\}$, where $\mathbf{bag}^\star(n) = \mathbf{bag}(n) \setminus \mathbf{bag}(m)$ if m is n's parent; $= \mathbf{bag}(n)$ if n has no parent;
3. $\lambda^e(n) = \{(\chi(v_1), \chi(v_2)) : (v_1, v_2) \in E \cap \mathbf{bag}^\star(n)^2\}$.

Labeling every node $n \in \mathcal{T}$ with the pair $(\lambda^v(n), \lambda^e(n))$ gives a **succinctly encoded tree-decomposition of G**.

Example: Succinctly encoded tree-decomposition of width 2



MSO formula: Formula over graphs with quantification on (sets of) edges and vertices

Example: The graph G is strongly connected iff G satisfies the formula

$$\varphi \equiv \forall X \subseteq V. \forall x, y \in V. x \notin X \vee y \in X \vee (\exists u, v \in V \text{ s.t. } \mathbf{E}(u, v) \wedge u \in X \wedge v \notin X)$$

Theorem statement [3]

Given an integer κ and an MSO formula φ, there exists a tree automaton $\mathcal{A}_{\kappa,\varphi}$ such that, for all graphs G and all succinctly encoded tree-decompositions $\mathcal{T}^{\mathbf{succinct}}$ of width κ of G:

G satisfies φ iff $\mathcal{A}_{\kappa,\varphi}$ accepts $\mathcal{T}^{\mathrm{succinct}}$.

Sequentially simulating runs of tree automata

Context: Bottom-up, deterministic automata perform computations in a distributed way. How can we simulate them on a single (sequential) computation thread?

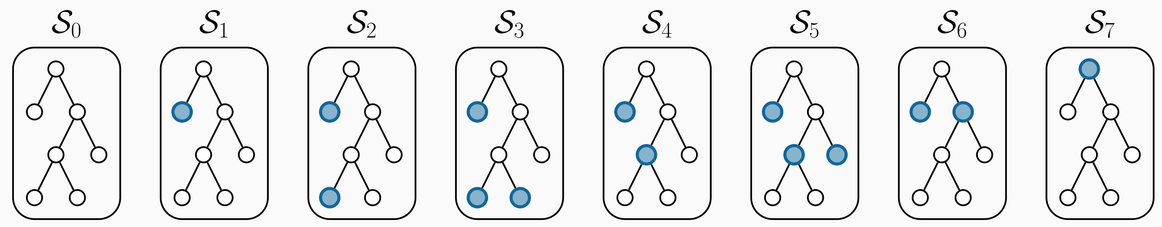
Tree automata and distributed computation: The run of the tree automaton $\mathcal{A} = \langle \Sigma, Q, \delta, \iota, F\rangle$ on a labeled tree $\mathcal{T} = (\mathcal{N}, \mathcal{E}, \Sigma)$ is the mapping $\rho : \mathcal{N} \mapsto Q$ such that:

1. $\rho(n) = \delta(\iota, \lambda(n), \iota)$ for all leaves n with label $\lambda(n) \in \Sigma$;
2. $\rho(n) = \delta(\rho(m_1), \lambda(n), \rho(m_2))$ for all nodes n with label $\lambda(n)$ and children m_1 and m_2.

The automaton $\mathcal{A}$ accepts the tree $\mathcal{T}$, with root τ, iff $\rho(\tau) \in F$.

Slicing $\mathcal{T}$: Choose subsets $\mathcal{S}_0, \ldots, \mathcal{S}_\ell$ of $\mathcal{N}$ such that $\mathcal{S}_0 = \emptyset$, $\mathcal{S}_\ell = \{\tau\}$ and, for $k \geqslant 1$:

1. there is a unique node n_k in $\mathcal{S}_k \setminus \mathcal{S}_{k-1}$; 2. its children (if any) belong to $\mathcal{S}_{k-1}$.



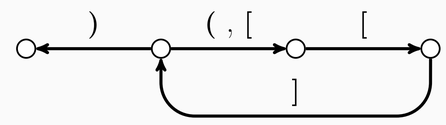
Sequential simulation: Compute the run restrictions $\rho\restriction_{\mathcal{S}_k}$ for $0 \leqslant k \leqslant \ell$:

1. the initial restriction $\rho\restriction_{\mathcal{S}_0}$ is fixed; 2. $\rho\restriction_{\mathcal{S}_\ell}$ determines whether $\mathcal{A}$ accepts $\mathcal{T}$;
3. $\rho\restriction_{\mathcal{S}_{k+1}}$ depends on $\rho\restriction_{\mathcal{S}_k}$ and $\lambda(n_{k+1})$ only: we set $\rho\restriction_{\mathcal{S}_{k+1}} = \Pi_k(\rho\restriction_{\mathcal{S}_k}, \lambda(n_{k+1}))$.

Sequential computations vs Dyck-path reachability

Dyck words: Well-parenthesized words (with multiple kinds of parentheses)
Dyck paths in a labeled graph: Paths whose labels are Dyck words

Example: There are 7 Dyck paths in this graph. Will you find them all?



Simulating a successful run with paths: Create a graph Γ with:

1. vertices (k, π), where $\pi : \mathcal{S}_k \mapsto Q$ for $0 \leqslant k \leqslant \ell$;
2. edges $(k, \pi) \mapsto (k + 1, \pi')$, where $\pi' = \Pi_k(\pi, \lambda(n_{k+1}))$.

$\mathcal{A}$ accepts $\mathcal{T}$ iff there is a path from $(0, \rho\restriction_{\mathcal{S}_0})$ to some vertex (ℓ, π) where $\pi(\tau) \in F$.

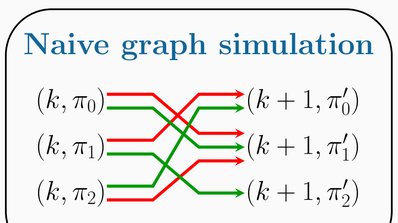
⚠ Issue: Changing one label of $\mathcal{T}$ may cause many changes in Γ!

Simulating a successful run with Dyck paths: Insert **gadgets** into Γ, i.e. add:

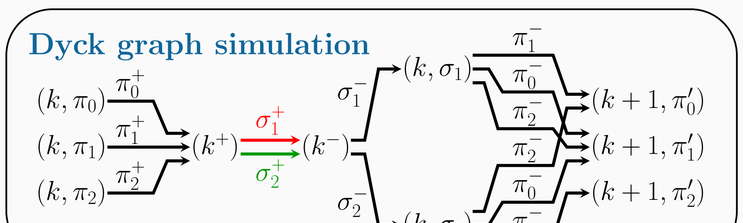
1. vertices (k^+), (k^-) and (k, σ) ;
2. egdes $(k, \pi) \xrightarrow{\pi^+} (k^+)$, $(k^+) \xrightarrow{\lambda(n_{k+1})^+} (k^-)$, $(k^-) \xrightarrow{\sigma^-} (k, \sigma)$ and $(k, \sigma) \xrightarrow{\pi^-} (k + 1, \pi')$

for $0 \leqslant k \leqslant \ell$, $\sigma \in \Sigma$, $\pi : \mathcal{S}_k \mapsto Q$ and $\pi' = \Pi_k(\pi, \lambda(n_{k+1}))$.
$\mathcal{A}$ accepts $\mathcal{T}$ iff there is a **Dyck** path from $(0, \rho\restriction_{\mathcal{S}_0})$ to some vertex (ℓ, π) where $\pi(\tau) \in F$.

Naive graph simulation



Dyck graph simulation



→ **edge present if $\lambda(n_{k+1}) = \sigma_1$** → **edge present if $\lambda(n_{k+1}) = \sigma_2$** → **edge present at all times**

Making Courcelle's theorem dynamic

Using two more ingredients in addition to the above constructions**:**

1. Computing **logarithmic-depth** tree-decompositions of width $4\kappa + 3$ in L [2, 4];
2. Solving Dyck-path reachability problems in acyclic graphs in $\mathsf{Dyn}(\mathsf{LogCFL},\mathsf{FO})$ [6].

Dynamic Courcelle's theorem statement [1]

Let κ and φ be fixed. Given a maximal graph $G^\star = (V, E^\star)$ of tree-width κ, an initial subgraph $G = (V, E)$ with $E \subseteq E^\star$, and updating G by adding/deleting edges $e \in E^\star$:

checking whether G satisfies φ is feasible in Dyn(L,FO).

References

1. P. Bouyer, V. Jugé, and N. Markey. Courcelle's theorem made dynamic. coRR/1702.05183, 2017.
2. H. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11(1-2):1–21, 1993.
3. B. Courcelle. The monadic second-order logic of graphs. I: Recognizable sets of finite graphs. *Inform. and Comput.*, 85(1):12–75, 1990.
4. M. Elberfeld, A. Jakoby, and T. Tantau. Logspace versions of the theorems of Bodlaender and Courcelle. In *FOCS'10*, pages 143–152. IEEE Comp. Soc. Press, 2010.
5. S. Patnaik and N. Immerman. Dyn-FO: A parallel, dynamic complexity class. *J. Comput. System Sci.*, 55(2):199–209, 1997.
6. V. Weber and T. Schwentick. Dynamic complexity theory revisited. *Theory Comput. Syst.*, 40(4):355–377, 2007.