7.3 MSO Interpretation of Graphs of Bounded Treewidth in Trees

Goal: Define an MSO interpretation of graphs of bounded treewidth in trees.

Recall: • Graphs are undirected and both vertex and edge labelled:

$$G = (V, E, \lambda_V, \lambda_E).$$

• In this setting, a tree decomposition

$$T = (N, \to, B)$$

also adds edges $e \in E$ to bags $B(n)$ with the following requirements:

↳ if $e = \{x, y\} \in B(n)$, then $x \in B(n)$ and $y \in B(n)$.

↳ Moreover, each edge is in at most one bag.

The following lemma will be helpful to identify nodes in the tree that represent the same vertex in the graph.

Recall that in a tree decomposition of width $m \in \mathbb{N}$, bags contain at most $m+1$ vertices/edges.

Lemma:

Let $G = (V, E, \lambda_V, \lambda_E)$ be a graph and $T = (N, \to, B)$ be a tree decomposition of G with $\text{width}(T) = m \in \mathbb{N}$.

There is a coloring

$$\lambda : V \longrightarrow C = \{c_1, \dots, c_{2m+2}\}$$

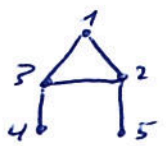
of the vertices of G with $2m+2$ colors so that $\forall n_1 \to n_2$ in $T : \forall x \neq y \in B(n_1) \cup B(n_2) : \lambda(x) \neq \lambda(y)$.

Actually, a coloring as in the previous lemma can be computed in linear time – provided the tree decomposition is given.

<u>Idea</u>:
- Assign colors to the vertices in the graph starting from those in the root node of the tree decomposition and processing the tree top down.
- A color from level n can be reused to color vertices of a child at level $n+2$, provided the color has not been used at level $n+1$.

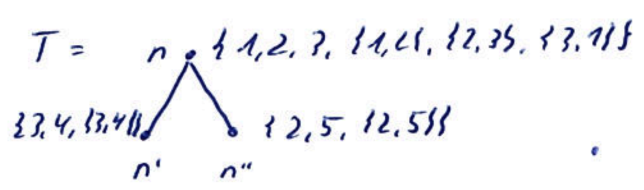
<u>Construction of a tree structure $S_{T'}$ from a graph structure S_G</u>:

<u>Idea</u>: ◦ Consider $G =$



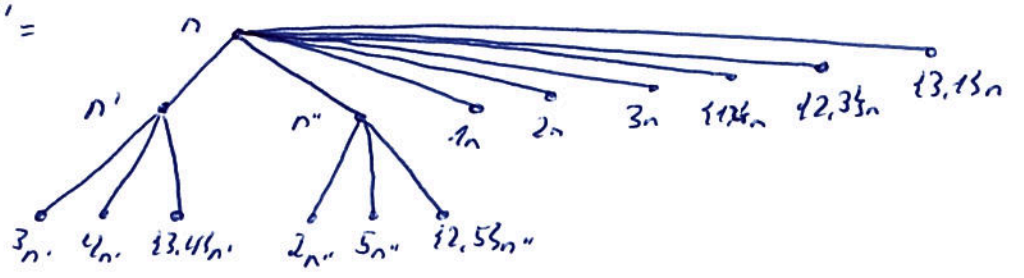
and its tree decomposition

$T =$



◦ Introduce a new node to represent every element in a bag:

$T' =$



<u>Formally</u>:
- Consider a graph $G = (V, E, \lambda_V, \lambda_E)$ and a tree decomposition $T = (N, \to, B)$ of width$(T) = m$.
- With the previous lemma, there is a labelling

$$\lambda : V \longrightarrow C = \{c_1, \dots, c_{2m+2}\}$$

for the vertices of G.
- Before we can turn T into a logical structure, we modify the tree as follows, yielding T':

For each node $n \in N$ of T

and each vertex or edge $x \in V \cup E$ in $B(n)$:

define a new node x_n in T'

and connect it to n with a new edge.

Now T' gives rise to the tree structure

$$S_{T'} := (\underbrace{A'}_{N' \cup \to'}, \underbrace{P_{N'}', P_{\to}'}_{\text{nodes and edges in } T'}, P_V', P_E', (P_a')_{a \in \Sigma_V}, (Q_a')_{a \in \Sigma_E}, (P_c')_{c \in C}, (R_i')_{1 \le i \le \binom{m+1}{2}})$$

The relations are as follows:

$P_V' := \{x_n \mid x \in P_V\}$ // x is a vertex of G

$P_E' := \{e_n \mid e \in P_E\}$ // e is an edge of G

$P_a' := \{x_n \mid x \in P_a\}$ // x is labelled by a in G

$Q_a' := \{e_n \mid e \in Q_a\}$ // e is labelled by a in G

$P_c' := \{x_n \mid \lambda(x) = c\}$ // x is colored by c

To represent the edges of G,

note that each node $n \in N$ of T

has at most $m+1$ vertices of G in its bag $B(n)$,

where $m = \text{width}(T)$.

Hence, there are at most

$\binom{m+1}{2}$ unordered pairs $\{x,y\}$ of such vertices.

For each pair $\{x,y\}$, indicated by a number $1 \le i \le \binom{m+1}{2}$,

define a unary predicate

$R_i' := \{x_n, y_n, e_n \mid R(x,e) \text{ and } R(y,e)\}$.

Definition (Courcelle's MSO Interpretation):

Define

$$\tau := (\alpha(x), \mathcal{E}(x,y), \gamma_{P_V}(x), \gamma_{P_E}(e), (\gamma_{P_a}(x))_{a\in\Sigma_V}, (\gamma_{Q_a}(e))_{a\in\Sigma_E}, \gamma_R(x,e))$$

as follows:

$$\alpha(x) := P_V'(x) \vee P_E'(x)$$

For the equivalence, define an auxiliary predicate

$$\mathcal{E}_1(x,y) := \bigvee_{c\in C} P_c'(x) \wedge P_c'(y)$$

$\wedge$ "there is an x-y-path whose internal vertices all have neighbors z satisfying $P_c'(z)$".

Then

$$\mathcal{E}(x,y) := x=y \vee (P_V'(x) \wedge P_V'(y) \wedge \mathcal{E}_1(x,y))$$

// Recall that edges are in precisely one bag.

Moreover:

$$\gamma_{P_V}(x) := P_V'(x)$$

$$\gamma_{P_E}(e) := P_E'(e)$$

$$\gamma_{P_a}(x) := P_a'(x)$$

$$\gamma_{Q_a}(e) := Q_a'(e)$$

$$\gamma_R(x,e) := P_V'(x) \wedge P_E'(e) \wedge \bigvee_{1\le i\le \binom{m+1}{2}} (R_i'(x) \wedge R_i'(e))$$

$\wedge$ "there is a node incident with x and e"

8. Courcelle's Theorem

<u>So far:</u> $S_G \models \varphi$ iff $S_{T'} \models \tau^{-1}(\varphi)$.

<u>Now:</u> • Reduce satisfaction $S_T \models \psi$ of an MSO formula ψ by a tree structure S_T to a membership query $T' \in L(A_\psi)$.
• Here, A_ψ is a <u>tree automaton</u> accepting the models of ψ.

8.1 Tree Automata

• A <u>bottom-up tree automaton (BUTA)</u> is a tuple $A = (\Sigma, Q, \delta, F)$ with
  • Σ = finite <u>ranked</u> alphabet, i.e. letters $a \in$ have an arity, also called rank $rk(a) \in \mathbb{N}$
  • Q = finite set of states, with final states $F \subseteq Q$.
  • $\delta = \bigcup_{a \in \Sigma} \delta_a$ with $\delta_a \subseteq \underbrace{Q \times \ldots \times Q}_{rank(a)} \times Q$

• A <u>run</u> of A on a Σ-labelled tree $T = (N, \to, \lambda)$ is a function

$$r : N \to Q$$

that labels the nodes of T by states of A so that we have for all nodes $n \in N$

$$((r(n.c_1), \ldots, r(n.c_k)), r(n)) \in \delta_a,$$

where $a = \lambda(n)$ and $k = rank(a)$.

• A run is <u>accepting</u> if $r(root) \in F$.

and $L(A) := \{ T \text{ over } \Sigma \mid \text{there is an accepting run of } A \text{ on } T \}$.

• A language of finite trees L is <u>regular</u>, if there is a tree automaton A with $L = L(A)$.

Example:

The language of propositional formulas that evaluate to true is regular:

$$\Sigma = \{t/_0, f/_0, \neg/_1, \wedge/_2, \vee/_2\}$$

and $A = (\Sigma, \{q_0, q_1\}, \rightarrow, \{q_1\})$

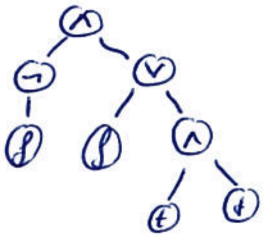
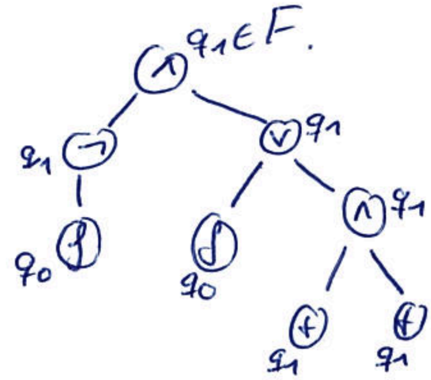
with

$$\rightarrow_f q_0 \qquad q_0 \rightarrow_\neg q_1 \qquad (q_0,q_0) \rightarrow_\wedge q_0 \qquad (q_0,q_0) \rightarrow_\vee q_0$$

$$\rightarrow_t q_1 \qquad q_1 \rightarrow_\neg q_0 \qquad (q_0,q_1) \rightarrow_\wedge q_0 \qquad (q_0,q_1) \rightarrow_\vee q_1$$

$$(q_1,q_0) \rightarrow_\wedge q_0 \qquad (q_1,q_0) \rightarrow_\vee q_1$$

$$(q_1,q_1) \rightarrow_\wedge q_1 \qquad (q_1,q_1) \rightarrow_\vee q_1.$$

A run of A on  is the following: 

Lemma:

Regular tree languages are closed under intersection and under complementation.

8.2 From MSO to Tree Automata (and Back):

Goal: • Given an MSO formula φ construct a BUTA A_φ so that

$$S_T \models \varphi \quad \text{iff} \quad T' \in L(A_\varphi).$$

• Predicates: $P_k(x,y)$ = x is kth child of y

$P_a(x)$ = label of x is a

Key technique: Alphabet extension

↳ Encode unary predicates and free variables into the node label:

$$\Sigma_V := \{0,1\}^{|\text{Unary Predicates}|+|\text{Free Variables}|}$$

↳ Technically: Identify

S_T, I with T_I, T extended by labels for the interpretation (of predicates and variables).

<u>Construction</u>: Illustrated on words, see additional material.

<u>Lemma</u>:

Consider an MSO property φ on labelled trees.
Given a labelled tree, it can be decided in <u>linear time</u>
whether $S_T \models \varphi$ holds.

<u>Proof</u>:

Precompute a deterministic BUTA A_φ
with

$$S_T \models \varphi \quad \text{iff} \quad T' \in L(A_\varphi).$$

Here, T' is the labelled variant of T
that makes the interpretation of predicates explicit.

Check $T' \in L(A_\varphi)$. □

<u>Theorem (Courcelle)</u>:

Consider an MSO property φ on graphs of bounded treewidth.
Given a graph G together with a tree decomposition T,
it can be checked in linear time whether $G \models \varphi$ holds.

<u>Proof</u>:

Precompute the deterministic BUTA $A_{\tau^{-1}(\varphi)}$.
Modify the tree decomposition as explained in Section 7.3
to obtain $S_{T'}$.
Turn the predicates in $S_{T'}$ into labels,
resulting in T''.
Check $T'' \in L(A_{\tau^{-1}(\varphi)})$. □

8.3 Applications

Goal: Define an automaton model on graphs
that is good for modelling concurrent pushdown systems

Idea: Reduce model checking to validity in MSO

Lemma (Corollary of Courcelle's theorem):
Consider an MSO formula on (undirected and labelled) graphs,
and a number $k \in \mathbb{N}$.
It is decidable whether there is a graph G of treewidth at most k
so that
$$S_G \models \varphi.$$

Proof:
With Courcelle's result,
we obtain a tree automaton $A_{\tau^{-1}(\varphi)}$
acting on trees that encode graphs.
We have
$$S_{T'} \models \tau^{-1}(\varphi) \quad \text{iff} \quad T'' \in L(A_{\tau^{-1}(\varphi)}),$$
where again T'' turns predicates into labels.
We now define another automaton A_k
that accepts all tree encodings of graphs of treewidth $\leq k$
↳ predicates P_a' only at leaves,
↳ every leaf is a vertex or an edge etc.
Then the problem boils down to
$$L(A_k) \cap L(A_{\tau^{-1}(\varphi)}) \neq \emptyset.$$
□

Note:
Automaton A_k is also needed when dealing with negation
in the proof of Courcelle's result itself.
↳ cf. automaton A_x in Büchi's construction for words.

<u>Summary:</u>

$$S_G \models \varphi \quad \overset{(1)}{\iff} \quad S_{T'} \models \tau^{-1}(\varphi)$$
$$\overset{(2)}{\iff} \quad T'' \in L(\mathcal{A}_{\tau^{-1}(\varphi)}),$$

with

(1) <u>MSO Interpretation</u>

↳ Introduced the general concept in Section 7.2

↳ Gave a concrete interpretation of graphs as trees following Courcelle in Section 7.3

(2) <u>Büchi's Construction (lifted to trees)</u>

↳ Induction on structure of formulas, alphabet extension to handle free variables, see additional material and Section 8.2

↳ Definition of tree automata in Section 8.1.

<u>Application:</u>
Let $G\mathcal{A}$ be a graph automaton (not defined so far)

$G\mathcal{A} \models \varphi$ on graphs from class φ_K of treewidth bounded by $n \in \mathbb{N}$.

$\iff \varphi_{G\mathcal{A}} \wedge \varphi_K \rightarrow \varphi$ valid for all graphs of treewidth bounded by $n \in \mathbb{N}$

$\iff \underbrace{\neg(\varphi_{G\mathcal{A}} \wedge \varphi_K \rightarrow \varphi)}_{\psi}$ unsatisfiable on graphs of treewidth bounded by $n \in \mathbb{N}$

$\iff \mathcal{L}(\mathcal{A}_n) \cap \mathcal{L}(\mathcal{A}_{\tau^{-1}(\psi)}) = \emptyset$.

Here, $\varphi_{G\mathcal{A}}$ is an MSO formula capturing the behavior of $G\mathcal{A}$. See construction of $\varphi_{\mathcal{A}}$ for an NFA $\mathcal{A}$ in additional material (Theorem Büchi I).

□