Graph Automata

Jan Leike

July 2nd, 2012
Motivation

We want an automata model that
Motivation

We want an automata model that

- operates on graphs,
- generalizes nested words and tree automata, and
- has some nice properties.
Outline

1. Introduce graph automata and related concepts
2. Proof that their emptiness is decidable\(^1\)
3. Show applications

\(^1\)side conditions apply
Definition of Graph Automata

Let $\Sigma$ be a finite alphabet and $\mathcal{C}$ be a class of $\Sigma$-labeled graphs. A graph automaton on $\mathcal{C}$, $GA = (Q, (T_a)_{a \in \Sigma}, \text{type})$, where

- $Q$ is a finite set of states,
- $T_a \subseteq Q \times Q$ is a tiling relation for every $a \in \Sigma$, and
- $\text{type} : Q \rightarrow 2^\Sigma \times 2^\Sigma$ is the type-relation.
Definition of Graph Automata

Let $\Sigma$ be a finite alphabet and $\mathcal{C}$ be a class of $\Sigma$-labeled graphs. A graph automaton on $\mathcal{C}$, $GA = (Q, (T_a)_{a \in \Sigma}, \text{type})$, where

- $Q$ is a finite set of states,
- $T_a \subseteq Q \times Q$ is a tiling relation for every $a \in \Sigma$, and
- $\text{type} : Q \to 2^\Sigma \times 2^\Sigma$ is the type-relation.

$GA$ accepts a graph $G = (V, (E_a)_{a \in \Sigma})$ iff there is a map $\rho : V \to Q$ such that

- for every $(u, v) \in E_a$, $(\rho(u), \rho(v)) \in T_a$ and
- for every $v \in V$, $\text{type}(\rho(v)) = (\text{In}, \text{Out})$, where
  $\text{In} = \{ a \mid \exists u. (u, v) \in E_a \}$ and $\text{Out} = \{ a \mid \exists u. (v, u) \in E_a \}$. 

We restrict ourselves to graphs with at most one incoming and at most one outgoing $a$-labeled edge for each $a \in \Sigma$ at any vertex.
Definition of Graph Automata

Let $\Sigma$ be a finite alphabet and $C$ be a class of $\Sigma$-labeled graphs. A graph automaton on $C$, $GA = (Q, (T_a)_{a \in \Sigma}, type)$, where

- $Q$ is a finite set of states,
- $T_a \subset Q \times Q$ is a tiling relation for every $a \in \Sigma$, and
- $type : Q \rightarrow 2^\Sigma \times 2^\Sigma$ is the type-relation.

$GA$ accepts a graph $G = (V, (E_a)_{a \in \Sigma})$ iff there is a map $\rho : V \rightarrow Q$ such that

- for every $(u, v) \in E_a$, $(\rho(u), \rho(v)) \in T_a$ and
- for every $v \in V$, $type(\rho(v)) = (In, Out)$, where $In = \{a \mid \exists u. (u, v) \in E_a\}$ and $Out = \{a \mid \exists u. (v, u) \in E_a\}$.

We restrict ourselves to graphs with at most one incoming and at most one outgoing $a$-labeled edge for each $a \in \Sigma$ at any vertex.
Check a graph for 3-colorability.

\[ GA_3 = (Q, (T_a)_{a \in \Sigma}) \] where

\[ \Sigma = \{a\}, \quad Q = \{q_1, q_2, q_3\} \] and \[ T_a = \{(q_i, q_j) \mid i \neq j\}. \]

Example: 3-color the Petersen graph
Example (proper)

Check a graph for 3-colorability.

$$GA_3 = (Q, (T_a)_{a \in \Sigma}, type)$$ where

$$\Sigma = \{a_1, \ldots, a_n\},$$

$$Q = \{q_1, q_2, q_3\} \times (2^{\Sigma} \times 2^{\Sigma}),$$

$$T_a = \{((q_i, t), (q_j, t')) | i \neq j\}$$ for $$a \in \Sigma$$ and

$$type((q, t)) = t$$ for $$(q, t) \in Q.$$

This restricts the graph to at most $$n$$ incoming and outgoing edges at every vertex.
Goal

Theorem (Madhusudan and Parlato 2011 [2])
Let \( \mathcal{C} \) be a class of MSO-definable \( \Sigma \)-labeled graphs. The problem of checking, given \( k \in \mathbb{N} \) and a graph automaton \( GA \), whether there is some \( G \in \mathcal{C} \) of tree-width at most \( k \) that is accepted by \( GA \), is decidable, and decidable in time \(|GA|^{O(k)}\).
Monadic second order logic

We use the following syntax for MSO, where $x$, $y$ are variables, $X$ is a set of vertices and $E_a$ is an $a$-labeled edge for $a \in \Sigma$.

$$\varphi ::= x = y \mid E_a(x, y) \mid x \in X \mid \varphi \lor \varphi \mid \neg \varphi \mid \exists x. \varphi \mid \exists X. \varphi$$
Definition of tree-width

The *tree-decomposition* of a graph $G = (V, E)$ is a tuple $(T, (B_t)_{t \in T})$, where $T = (T, F)$ is a tree and for every node $t \in T$, $B_t \subseteq V$ is a bag of vertices of $G$ such that

- for every $v \in V$, there is a node $t \in T$ such that $v \in B_t$,
- for every edge $(u, v) \in E$, there is a node $t \in T$ such that $u, v \in B_t$, and
- if $v \in B_t$ and $v \in B_{t'}$, for nodes $t, t' \in T$, then for every $t''$ that lies on the unique path connecting $t$ and $t'$, $v \in B_{t''}$.

The width of a tree decomposition is the size of the largest bag in it, minus one; i.e., $\max \{ |B_t| | t \in T \} - 1$. The tree-width of a graph is the smallest of the widths of any of its tree decompositions.
The *tree-decomposition* of a graph $G = (V, E)$ is a tuple $(T, (B_t)_{t \in T})$, where $T = (T, F)$ is a tree and for every node $t \in T$, $B_t \subseteq V$ is a bag of vertices of $G$ such that

- for every $v \in V$, there is a node $t \in T$ such that $v \in B_t$,
- for every edge $(u, v) \in E$, there is a node $t \in T$ such that $u, v \in B_t$, and
- if $v \in B_t$ and $v \in B_{t'}$, for nodes $t, t' \in T$, then for every $t''$ that lies on the unique path connecting $t$ and $t'$, $v \in B_{t''}$.

The width of a tree decomposition is the size of the largest bag in it, minus one; i.e. $\max\{|B_t| \mid t \in T\} - 1$.

The *tree-width* of a graph is the smallest of the widths of any of its tree decompositions.
Example

stack one

stack two

Input word for a 2-NWA.
Example: Tree decomposition

Canonical tree decomposition of the graph. (→ Formal definition)
Some facts about tree-width

- A graph without edges has tree-width 0.
- A tree has tree-width of at most 1.
- A graph with a $k$-clique has a tree-width of at least $k - 1$.
- A graph with $n$ vertices has a minimal tree decomposition using at most $n$ nodes.
- Many NP-complete problems become tractable on graphs of bounded tree-width.
- Computing tree-widths is NP-hard.
Decidable emptiness

Theorem (Madhusudan and Parlato 2011 [2])
Let $\mathcal{C}$ be a class of MSO-definable $\Sigma$-labeled graphs. The problem of checking, given $k \in \mathbb{N}$ and a graph automaton $GA$, whether there is some $G \in \mathcal{C}$ of tree-width at most $k$ that is accepted by $GA$, is decidable, and decidable in time $|GA|^{O(k)}$. 
Decidable emptiness (proof sketch)

Let $G$ be an input graph and $(\mathcal{T}, (B_t)_{t \in \mathcal{T}})$ its tree decomposition.
Decidable emptiness (proof sketch)

Let $G$ be an input graph and $(\mathcal{T}, (B_t)_{t \in \mathcal{T}})$ its tree decomposition.

- The node labels of $\mathcal{T}$ contain information on the structure of the subgraph contained in the bag and which vertex also occurs at the parent node.
Decidable emptiness (proof sketch)

Let $G$ be an input graph and $(\mathcal{T}, (B_t)_{t \in \mathcal{T}})$ its tree decomposition.

- The node labels of $\mathcal{T}$ contain information on the structure of the subgraph contained in the bag and which vertex also occurs at the parent node.
- Bounded tree-width $\Rightarrow \mathcal{O}(2^k)$ many labels suffice.
Decidable emptiness (proof sketch)

Let $G$ be an input graph and $(\mathcal{T}, (B_t)_{t \in T})$ its tree decomposition.

- The node labels of $\mathcal{T}$ contain information on the structure of the subgraph contained in the bag and which vertex also occurs at the parent node.
- Bounded tree-width $\Rightarrow O(2^k)$ many labels suffice.
- Transform the MSO formula $\varphi_C$ defining the class of graphs $\mathcal{C}$ into a MSO formula $\widehat{\varphi}_C$ about trees.
Decidable emptiness (proof sketch)

Let $G$ be an input graph and $(T, (B_t)_{t \in T})$ its tree decomposition.

- The node labels of $T$ contain information on the the structure of the subgraph contained in the bag and which vertex also occurs at the parent node.
- Bounded tree-width $\Rightarrow \mathcal{O}(2^k)$ many labels suffice.
- Transform the MSO formula $\varphi_C$ defining the class of graphs $C$ into a MSO formula $\widehat{\varphi}_C$ about trees.
- Transform $\widehat{\varphi}_C$ into a tree automaton $TA_C$. 
Decidable emptiness (proof sketch)

Let $G$ be an input graph and $(\mathcal{T}, (B_t)_{t \in \mathcal{T}})$ its tree decomposition.

- The node labels of $\mathcal{T}$ contain information on the structure of the subgraph contained in the bag and which vertex also occurs at the parent node.
- Bounded tree-width $\Rightarrow \mathcal{O}(2^k)$ many labels suffice.
- Transform the MSO formula $\varphi_C$ defining the class of graphs $\mathcal{C}$ into a MSO formula $\widehat{\varphi}_C$ about trees.
- Transform $\widehat{\varphi}_C$ into a tree automaton $TA_C$.
- For the graph automaton $GA$, define a tree automaton $TA$ running over $\mathcal{T}$. 
Decidable emptiness (proof sketch)

Let $G$ be an input graph and $(\mathcal{T}, (B_t)_{t \in \mathcal{T}})$ its tree decomposition.

- The node labels of $\mathcal{T}$ contain information on the the structure of the subgraph contained in the bag and which vertex also occurs at the parent node.
- Bounded tree-width $\Rightarrow \mathcal{O}(2^k)$ many labels suffice.
- Transform the MSO formula $\varphi_C$ defining the class of graphs $\mathcal{C}$ into a MSO formula $\widehat{\varphi}_C$ about trees.
- Transform $\widehat{\varphi}_C$ into a tree automaton $TA_C$.
- For the graph automaton $GA$, define a tree automaton $TA$ running over $\mathcal{T}$.
- There is a graph in $\mathcal{C}$ that is accepted by $GA$ iff the intersection of $TA$ and $TA_C$ is not empty.
Labeling of the tree decomposition

The labeling of tree decomposition captures the isomorphism type of the graph.
For a node \( v \in T \) and its parent \( u \in T \), let the bag 
\( B_v = \{ v_1, \ldots, v_k \} \) and \( B_u = \{ u_1, \ldots, u_k \} \).
The label for \( v \) will be \( ((L_a)_{a \in \Sigma}, P, W) \) where
\[
\begin{align*}
\&L_a = \{(i, j) \mid (v_i, v_j) \in E_a\} \\
\&P = \{(i, j) \mid v_i = u_j\} \text{ and} \\
\&W = \{(i, j) \mid v_i = v_j\}.
\end{align*}
\]
Note: Using more careful encoding, this can be achieved using \( \mathcal{O}(2^k) \) instead of \( \mathcal{O}(2^{k^2}) \) many labels [2].
Graph automaton as tree automaton

For a graph automaton $GA = (Q, (T_a)_{a \in \Sigma}, type)$ define a bottom-up tree automaton $B = (Labels, Q', Q', \Delta)$ where $Q' = (Q \times 2^\Sigma \times 2^\Sigma)^{k+1}$ and the transition rules

- check that the state at the node respects the tiling requirements $T_a$ and
- accumulate the $In$ and $Out$ sets for every vertex and cross-references them with the constraints in $type$. 
Applications
Nested word automata

- Nested words have a tree-width of at most 2.
- Therefore NWAs have decidable emptiness.

![Diagram of a nested word automaton with nested words and tree decomposition]
Nested word automata

- Nested words have a tree-width of at most 2.
- Therefore NWAs have decidable emptiness.

![Nested word automata diagram]

a nested word

its tree decomposition
Generalize NWAs to have $n$ instead of just one nesting relation.

Corresponds to a PDA with $n$ stacks.

$n$-NWAs have undecidable emptiness.

Therefore $n$-nested words have unbounded treewidth.
Bounded context switching NWAs

Bounded context switching NWA is an $n$-NWA where each word is partitioned into at most $k + 1$ “contexts”. Each context utilizes at most one of the $n$ stacks.

- Tree-width of $k + 1$.
- Decidable emptiness.
Other modifications to NWAs

▶ $k$-phase $n$-NWAs: in each phase any stack can be pushed, but only one stack can be popped. Tree-width: $3 \cdot 2^{k-1} + 1$. 
Other modifications to NWAs

- \textit{k-phase $n$-NWAs:} in each phase any stack can be pushed, but only one stack can be popped. Tree-width: $3 \cdot 2^{k-1} + 1$.  
- \textit{Ordered $n$-NWAs:} any stack can be pushed, but a stack can be popped only if all stacks with with lower index are empty. Tree-width: $(n + 1) \cdot 2^{n-1} + 1$. 
Further topics

- Formal definition of the canonical tree decomposition
- Efficient coding of tree labels
- Courcelle’ theorem
- Simulation of tree automata
- Recognizing connected graphs
Summary

- Graph automata are a powerful automata model.
- Restriction to an MSO-definable class $C$ of graphs with bounded tree-width yields decidable emptiness.
- Graph automata naturally generalize nested word automata and various modifications thereof.
- However, our definition of graph automata is not particularly useful for problems on graphs.
References


3-color the Petersen graph
3-colored the Petersen graph
Canonical tree decomposition for nested words

For any $n$-nested word $N = (V, \text{Init}, \text{Final}, L, (E_j)_{0 \leq j < n})$, the canonical tree-decomposition of $N$, $\text{can-td}(N) = (\mathcal{T}, (B_t)_{t \in \mathcal{T}})$ is defined as follows.

- The set of nodes of the tree $\mathcal{T}$ are the vertices $V$ of $N$.
- If $(u, v) \in E_j$, then $v$ is the right-child of $u$ in $\mathcal{T}$.
- If $(u, v) \in L$ and for all $0 \leq j < n$ and $z \in V$, $(z, v) \not\in E_j$, then $v$ is the left-child of $u$.

The bags $B_v$ associate the minimum set of vertices to the nodes $v \in \mathcal{T}$ that satisfy the following.

- For all $v \in V$, $v \in B_v$.
- For every $u, v \in V$, if $u$ is the parent of $v$ in $\mathcal{T}$, then $u \in B_v$.
- For $u, v \in V$, if $(u, v) \in L$ then $u \in B_z$ for all vertices $z$ that are on the unique path from $u$ to $v$ in $\mathcal{T}$.
Labeling of the tree decomposition using $\mathcal{O}(2^k)$ labels

For a node $v \in T$ and its parent $u \in T$, let the bag $B_v \subseteq \{v_1, \ldots, v_k\}$ and $B_u \subseteq \{u_1, \ldots, u_k\}$ where $v_i \neq v_j$ and $u_i \neq u_j$ for $i \neq j$.

Without loss of generality one can assume

- that the vertex $v_i \in B_v$ is equal to a vertex in the parent bag $u_j \in B_u$ iff $i = j$ and
- that every edge node in the tree captures at most one edge in the graph.

The label for $v$ will be $((L_a)_{a \in \Sigma}, P, W)$ where

- $L_a = (i, j)$, where $(v_i, v_j) \in E_a$ and $v_i, v_j \in B_v$,
- $P = \{i \mid v_i = u_i, v_i \in B_v, u_j \in B_u\}$ and
- $W = \{i \mid v_i \in B_v\}$.

This encoding uses $(k^2)^\#\Sigma \cdot 2^k \cdot 2^k = \mathcal{O}(2^k)$ many labels [2].
Simulating tree automata

Simulating tree automata induces the following difficulties:

1. Tree automata ignore vertex types.
2. Tree automata have labeled nodes, graph automata labeled edges.
3. Every edge has to specify its position in the predicate.
Simulating tree automata: Example

Consider the tree language of valid propositional logic formulae.

Example tree from the tree automata presentation

Corresponding input for the graph automaton

→ Go back
Courcelle’s theorem

Theorem (Courcelle [4])
Every graph property definable in monadic second-order logic can be decided in linear time on graphs of bounded tree-width.