Series-Parallel Graphs

Mahesh Viswanathan

Fall 2018

Series-Parallel graphs are a family of *multi-graphs* that arise as models of certain kinds of electrical networks and enjoy good algorithmic properties. In this lecture, we will introduce this class of graphs, and provide some evidence of why they amenable to efficient computation. Before defining this class of graphs, let us recall that by *multi-graphs*, we mean a generalization of graphs, where there can be multiple edges between a pair of vertices, and the possibility of vertices to self loops. In particular, we will be interested in multi-graphs that have a special *source* and *sink* vertex. We call such graphs *two-terminal graphs*.

Definition 1. Formally, a two-terminal graph is $G = (V, E, s, t, I)$, where V is a finite set of *vertices*, E is a finite set of *edges*, $s \in V$ is the *source*, and $t \in V$ is the *sink*, and $I \subseteq V \times E \times V$ is the *incidence relation* that identifies the source and target of each edge.

Example 2. Consider the two-terminal graph shown in Figure 1. We can describe the graph $G = (V, E, s, t, I)$, formally, as follows.

- $V = \{1, 2, 3, 4, 5\}$
- $E = \{A, B, C, D, E, F\}$
- $s = 1$ and $t = 5$
- I is given by

$$\{(1, A, 2), (2, A, 1), (1, B, 2), (2, B, 1), (2, C, 3), (3, C, 2)$$
$$(2, D, 4), (4, D, 2), (3, E, 5), (5, E, 3), (4, F, 5), (5, F, 4)\}$$

Two-terminal series-parallel graphs are two-terminal multi-graphs built from edges using two operations — *series* and *parallel* compositions. We define will define these two operations before giving an inductive definition of series-parallel graphs.

Definition 3 (Series Composition)**.** Series composition of two graphs is shown in Figure 2. The formal definition is as follows. Let $G_1 = (V_1, E_2, s_1, t_1, I_1)$ and $G_2 = (V_2, E_2, s_2, t_2, I_2)$ be two two-terminal graphs. Their series composition, denoted $G_1 \cdot G_2$ is (V, E, s, t, I) where

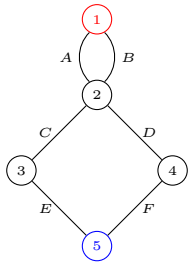

Figure 1: An example of a two terminal (multi) graph.

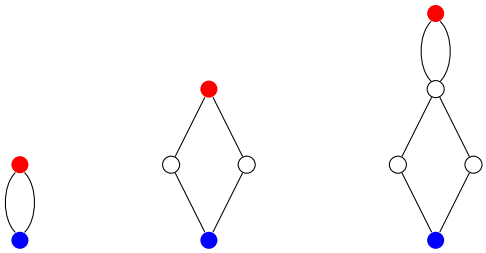

Figure 2: The graph on the left is G_1, and the graph in the middle is G_2. The graph on the right is $G_1 \cdot G_2$, the series composition of G_1 and G_2.

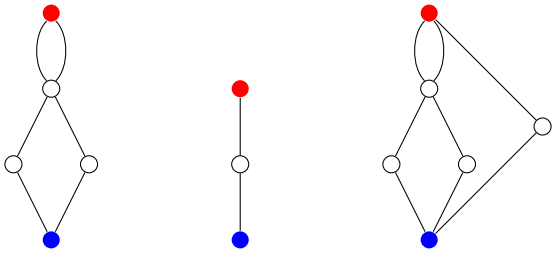

Figure 3: Graph G_1 (left) and G_2 (middle) and their parallel composition $G_1||G_2$ (right).

- $V = (V_1 \cup V_2) \setminus \{s_2\}$
- $E = E_1 \cup E_2$
- $s = s_1$ and $t = t_2$
- I is the set
$$((I_1 \cup I_2) \cap (V \times E \times V)) \cup$$
$$\{(t_1, e, u), (u, e, t_1) \mid u \neq s_2 \text{ and } (s_2, e, u), (u, e, s_2) \in I_2\} \cup$$
$$\{(t_1, e, t_1) \mid (s_2, e, s_2) \in I_2\}$$

Definition 4 (Parallel Composition)**.** Parallel graph of two graphs is shown in Figure 3. Formally, let $G_1 = (V_1, E_2, s_1, t_1, I_1)$ and $G_2 = (V_2, E_2, s_2, t_2, I_2)$ be two two-terminal graphs. Their parallel composition, denoted $G_1||G_2$ is (V, E, s, t, I) where

- $V = (V_1 \cup V_2) \setminus \{s_2, t_2\}$
- $E = E_1 \cup E_2$
- $s = s_1$ and $t = t_1$
- I is the set
$$((I_1 \cup I_2) \cap (V \times E \times V)) \cup$$
$$\{(s_1, e, u), (u, e, s_1) \mid u \notin \{s_2, t_2\} \text{ and } (s_2, e, u), (u, e, s_2) \in I_2\} \cup$$
$$\{(t_1, e, u), (u, e, t_1) \mid u \notin \{s_2, t_2\} \text{ and } (t_2, e, u), (u, e, t_2) \in I_2\} \cup$$
$$\{(s_1, e, s_1) \mid (s_2, e, s_2) \in I_2\} \cup$$
$$\{(t_1, e, t_1) \mid (t_2, e, t_2) \in I_2\}$$

A series-parallel graphs are built inductively from edges using the series and parallel composition opertaors.

Definition 5 (Series-Parallel)**.** *Two-terminal Series-Parallel graphs (TTSPG)* are built up using an edge, and series and parallel compositions. Formally, they are defined inductively as follows:

- $G = (\{s,t\}, \{e\}, s, t, \{(s,e,t),(t,e,s)\})$ is a TTSPG.
- If G_1 and G_2 are TTSPGs then $G_1 \cdot G_2$ and $G_1 || G_2$ are TTSPGs.

A (multi-)graph $G = (V, E, I)$ (without source and sink) is a *series-parallel graph* if there are $s, t \in V$ such that $G' = (V, E, s, t, I)$ is a TTSPG.

1 Algorithmic Properties of Series-Parallel Graphs

Series-Parallel graphs enjoy nice algorithmic properties. In this section, we state a couple of results; the proofs of these results are beyond the scope of this course. We begin by observing that there is a fast algorihtm to determine if a graph is a series-parallel graphs.

Theorem 6 (Valdes-Tarjan-Lawler)**.** *There is a linear time algorithm that determines if G is a series-parallel graph.*

Proof Sketch. Relies on observing that series-parallel graphs can be alternately defined as: G is a series-parallel graph if it can be turned into a single edge by sequence of the following operations

- Replacing a pair of parallel edges by a single edge between the common endpoints.
- Replacing the two edges incident upon a vertex (not equal to source and sink) of degree two by a single edge connecting the other endpoints of the removed edges.

□

Many computational problems, whose decision versions are NP-complete, turned out to solvable in linear time. Unfortunately, proving these observations is beyond the scope of the course. However, when we introduce tree-width of graphs, and study dynamic programming algorithms on graphs with bounded tree-width, we will be able to demonstrate efficient algorithms for these problems on series-parallel graphs.

Theorem 7. *The following problems are solvable in linear time on series-parallel graphs: maximum matching, maximum independent set, minimum dominating set, and Hamiltonian completion.*

2 Monadic Second Order Logic on Series-Parallel Graphs

To demonstrate the nice algorithmic properties of series-parallel graphs, we show that MSO logic on series-parallel graphs is decidable. Multi-graphs (and hence series-parallel graphs) can be viewed as first order structures over the following signature $\tau_{MG} = \{V, E, I\}$, where V is a unary predicate identifying elements on the universe that are vertices, E is a unary predicate identifying those that are edges, and I is a 3-ary predicate for the incidence relation. Let us look at an example of such a structure.

Example 8. Consider the graph shown in Figure 1. Ignoring the special source and sink vertices, the graph can be viewed as a structure over signature τ_{MG} as follows. $\mathcal{G} = (G, V^{\mathcal{G}}, E^{\mathcal{G}}, I^{\mathcal{G}})$, where $G = \{1,2,3,4,5,A,B,C,D,E,F\}$, $V^{\mathcal{G}} = \{1,2,3,4,5\}$, $E^{\mathcal{G}} = \{A,B,C,D,E,F\}$, and $I^{\mathcal{G}}$ is given as

$$\{(1,A,2),(2,A,1),(1,B,2),(2,B,1),(2,C,3),(3,C,2)$$
$$(2,D,4),(4,D,2),(3,E,5),(5,E,3),(4,F,5),(5,F,4)\}$$

Our main result of this section is the following: Given a series-parallel graph $\mathcal{G}$ (as a structure over τ_{MG}) and a MSO sentence φ (over τ_{MG}), it is decidable to determine if $\mathcal{G} \models \varphi$. Our proof of this observation is based on the general principle of *interpretations*, examples of which we have seen in the past. For example, to prove the decidability of $\mathrm{Th}((\mathbb{N}, 0, 1, +))$, we "reduced" it to WS1S where

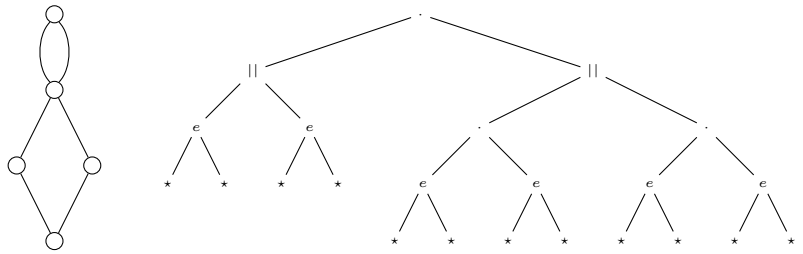

Figure 4: Graph G (on left) can be seen as $((G_1||G_1) \cdot ((G_1 \cdot G_1)||(G_1 \cdot G_1)))$, where G_1 is the graph with one edge.

- Each natural number was mapped to a *set* of natural numbers
- Each basic predicate in Pressburger arithmetic was translated to a MSO formula over $(0, S)$

This gave a translation of first order formula $\varphi(x_1, \ldots x_n)$ to an MSO formula $\varphi'(X_1, \ldots X_n)$ such that

$$(\mathbb{N}, 0, 1, +) \models \varphi[\alpha] \qquad \text{iff} \qquad (\mathbb{N}, 0, S) \models \varphi'[[X_i \mapsto \text{Set}(\alpha(x_i))]_{i=1}^n]$$

To prove the decidability of MSO over series-parallel graphs, we will follow a similar idea. We will "reduce" MSO on series-parallel graphs to MSO on binary trees. More precisely, with any series-parallel graph G, we will associate a tree $T(G)$, and translate any MSO formula $\varphi(x_1, \ldots x_n, Y_1, \ldots Y_m)$ over τ_{MG} to an MSO formula $T(\varphi)(X_1, \ldots X_n, Y_1, \ldots Y_m)$ over $(<, S_1, S_2, \{Q_a\}_{a \in \Sigma})$ such that

$$G \models \varphi[\alpha] \qquad \text{iff} \qquad T(G) \models T(\varphi)[\alpha']$$

where α' is the "corresponding" assignment.

To carry out this plan, we need to first identify the "tree interpretation" of any series-parallel graph. The idea is to take $T(G)$ to be the "parse tree" of the graph G. For example, consider the graph G and its corresponding "parse tree" $T(G)$ shown in Figure 4. We can define this correspondence precisely, as follows. Recall that if t_1 and t_2 are labeled binary trees, then $A(t_1, t_2)$ denotes the labeled binary tree whose root is labeled A, and the left and right subtrees of the root are t_1 and t_2, respectively.

Definition 9. Let $\Gamma = \{\star, e, \cdot, ||\}$. For a series-parallel graph G, $T(G)$ will be a Γ-labeled tree defined inductively as follows.

- If $G = (\{s, t\}, \{e\}, \{(s, e, t), (t, e, s)\})$ then $T(G) = e(\star, \star)$.
- If $G = G_1 \cdot G_2$ then $T(G) = \cdot(T(G_1), T(G_2))$.
- If $G = G_1||G_2$ then $T(G) = ||(T(G_1), T(G_2))$.

The tree $T(G)$ corresponding to a series-parallel graph G has representatives of each edge and vertex — every edge of G corresponds to a unique vertex labeled e in $T(G)$, while every vertex of G corresponds to a unique set of leaves labeled $\star$. This correspondence is illustrated in the example series-parallel graph G and its tree representation $T(G)$ in Figure 5.

This correspondence between vertices and edges of G to sets of vertices in $T(G)$, helps us formulate the precise inductive invariant to maintain when we reduce MSO on series-parallel graphs to MSO on labeled trees. Recall that $T(G)$ is a Γ-labeled graph, where $\Gamma = \{\star, e, \cdot, ||\}$. Formally, our inductive invariant is as follows. Given a formula $\varphi(x_1, \ldots x_n, Y_1, \ldots Y_m)$ over τ_{MG}, we will construct an MSO formula $T(\varphi)(X_1, \ldots X_n, Y_1, \ldots Y_m)$ over $(<, S_1, S_2, \{Q_a\}_{a \in \Gamma})$ such that

$$G \models \varphi[\alpha] \qquad \text{iff} \qquad T(G) \models T(\varphi)[\alpha']$$

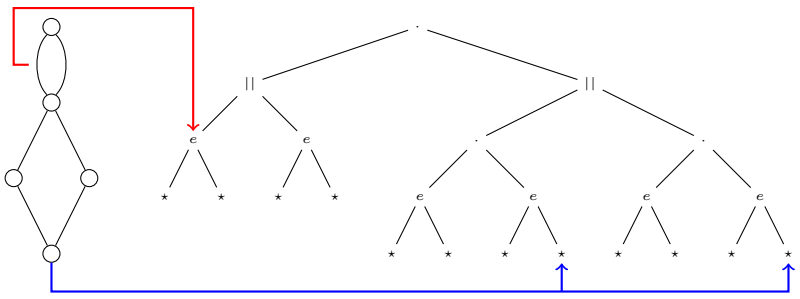

Figure 5: Red edge showing edge correspondence, and blue edges showing vertex correspondence.

where α' is the assignment that maps X_i to the set corresponding to the vertex/edge $\alpha(x_i)$ and Y_i to the set corresponding to the set $\alpha(Y_i)$.

We devote the rest of the section to describing the translation $T(\varphi)$ for a formula φ. The formula $T(\varphi)$ (as always) will be given inductively based on the structure of φ. Let us start with the base cases.

- $\varphi := x_i = x_j$: $T(\varphi) := X_i = X_j$
- $\varphi := Vx_i$: $T(\varphi) := \forall x(X_i x \to Q_\star x)$
- $\varphi := Ex_i$: $T(\varphi) := \forall x(X_i x \to Q_e x)$
- $\varphi := Ix_i x_j x_k$: $T(\varphi) := \exists x \exists y \exists z(X_i x \wedge X_j y \wedge X_k z \wedge ((S_1 yx \wedge S_2 yz) \vee (S_1 yz \wedge S_2 yx))$
- $\varphi := Y_i x_j$: $T(\varphi) := \forall x(X_j x \to Y_i x)$

Next, let us consider the Boolean connectives. These cases are straightforward. When $\varphi := \bot$, $T(\varphi) := \bot$ and when $\varphi := \varphi_1 \to \varphi_2$, $T(\varphi) := T(\varphi_1) \to T(\varphi_2)$.

Let us now consider the interesting case of handling quantifiers. Consider for example, the formula $\exists x\psi$ with a first order existential quantifier. Intuitively, $\exists x\psi$ says that there is an element of the universe (either a vertex or edge) such that ψ holds when x is interpreted as this element. Since vertices and edges of the graph correspond to sets of vertices in the tree, the natural translation would take the form $\exists X T(\psi)$, where the set X either encodes a vertex or edge of the graph. So we need to be able to ensure that X is not an arbitrary set of vertices of the tree, but rather those that correspond to vertices or edges of the graph. We will need to define these conditions inside MSO, and for that we will need auxiliary sentences.

Let us first try to write down when a set X (of vertices in the tree) corresponds to an edge of the graph. From Figure 5 we see that X corresponds to an edge iff X is a set containing a single vertex whose label is e. This can be easily written down in MSO.

$$(\exists x(Q_e x \wedge Xx) \wedge \forall x \forall y((Xx \wedge Xy) \to (x = y)))$$

Next, let us try to identify when X corresponds to a vertex of the graph. Notice, from Figure 5, a single vertex of G corresponds to a set of leaves of the tree. To say that tree vertices X correspond to a single vertex of the graph, we need to make sure that X contains all leaves that correspond to a vertex, and *only those*. So we need to identify when a set of leaves X, contains all copies of graph vertex and only those. We can make some observations about which leaves correspond to the "same" vertex of the graph.

- For a vertex labeled $\cdot$, the rightmost child of the left subtree is the same vertex as the leftmost child of the right subtree.
- On the other hand, for a vertex labeled $||$, the leftmost leaves of a left and right child are same and the rightmost leaves of left and right child are same.

Based on the above observations, we can write down when a set of tree vertices X, satisfies the property that it is "valid" set of vertices of the graph, i.e., X either contains all copies of a given vertex or none. Let $\mathsf{dot}(x, y)$ be a formula that captures when x and y are the same graph vertex because of an ancestor labeled $\cdot$, and let $\mathsf{par}(x, y)$ be the formula that describes when x and y are the same vertex because of an ancestor labeled $||$. Using these auxiliary formulas (which we will also write down explicitly soon), we can write the formula $\mathsf{vert}(X)$ which characterizes when X is a valid set of vertices as follows.

$$\mathsf{vert}(X) = \forall x(Xx \to Q_\star x) \wedge \forall x \forall y((\mathsf{dot}(x, y) \vee \mathsf{par}(x, y)) \to (Xx \leftrightarrow Xy)).$$

Now dot and par can be defined using the conditions identified above. However to do that, we need to write auxiliary formulas $\mathsf{lm}(x, y)$ that says that y is a leftmost leaf of x, and $\mathsf{rm}(x, y)$ which says that y is the rightmost leaf of x. We have the following set of formulas to complete the description of $\mathsf{vert}(X)$.

$$\begin{aligned}
&\mathsf{lm}(x, y) = Q_\star y \wedge \forall P((Px \wedge \forall z_1 \forall z_2((Pz_1 \wedge S_1 z_1 z_2) \to Pz_2)) \to Py)\\
&\mathsf{rm}(x, y) = Q_\star y \wedge \forall P((Px \wedge \forall z_1 \forall z_2((Pz_1 \wedge S_2 z_1 z_2) \to Pz_2)) \to Py)\\
&\mathsf{dot}(x, y) = \exists z \exists z_1 \exists z_2 (Q_\cdot z \wedge S_1 z z_1 \wedge S_2 z z_2 \wedge \mathsf{rm}(z_1, x) \wedge \mathsf{lm}(z_2, y))\\
&\mathsf{par}(x, y) = \exists z \exists z_1 \exists z_2 (Q_{||} z \wedge S_1 z z_1 \wedge S_2 z z_2 \wedge ((\mathsf{lm}(z_1, x) \wedge \mathsf{lm}(z_2, y)) \vee (\mathsf{rm}(z_1, x) \wedge \mathsf{rm}(z_2, y)))
\end{aligned}$$

The formula $\mathsf{vert}(X)$ captures conditions when X corresponds to a valid set of graph vertices. When describing the translation of $\exists x\psi$, we need to characterize when a set X corresponds to a *single* edge or vertex. We could say that this holds if either X corresponds to a single edge (which we wrote before as a formula) or X is a valid set of vertices that is "minimal". Formally,

$$\begin{aligned}
\mathsf{sing}(X) = \quad & (\exists x(Q_e x \wedge Xx) \wedge \forall x \forall y((Xx \wedge Xy) \to (x = y))) \vee\\
& (\exists x Xx \wedge \mathsf{vert}(X) \wedge \forall P((\mathsf{vert}(P) \wedge \exists x(Xx \wedge Px)) \to \forall x(Xx \to Px))
\end{aligned}$$

We now have all the auxiliary formulas to describe the translation of quantifiers.

- $\varphi := \exists x\psi$: $T(\varphi) := \exists X \mathsf{sing}(X) \wedge T(\psi)$.
- $\varphi := \exists X\psi$: $T(\varphi)$ should say Y is a valid set of vertices and edges such that $T(\psi)$ holds.

$$\begin{aligned}
T(\varphi) := \exists Y (\exists V \exists E(\forall x(Yx \to (Vx \vee Ex))) \wedge \mathrm{vert}(V) \wedge\\
\forall x(Ex \to Q_e x) \wedge T(\psi))
\end{aligned}$$

Decidability of MSO on series-parallel graphs then follows from the decidability of MSO on labeled binary trees.