Introduction to Program Synthesis (WS 2024/25)

Chapter 2.2 - Foundations (Program Representation: Graph)

Dr. rer. nat. Roman Kalkreuth

Chair for AI Methodology (**AIM**), Department of Computer Science,
RWTH Aachen University, Germany

Computer Programs: Representations
Graph

- **Graph** → Highly versatile data structure that can be used to represent all types of algorithms
  - ⇝ Numerous application options: Modelling of networks, logic &m quantum circuits, algorithms, molecules, ...
  - ⇝ Representation of cyclic structures
  - ⇝ Naturally more complex than trees
- **Tree** → special type of graph that is **connected** and **acyclic**
  - ⇝ Commonly used to represents hierarchical structures

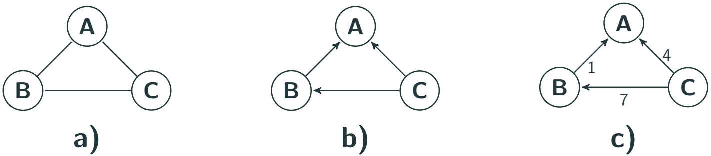
Definition (Graph)

A graph $G = (V, E)$ is a set of vertices and a set of edges E with $E \subseteq V \times V$.

- **Vertices (Nodes)** $\rightarrow$ Individual entities or points in the graph.
- **Edges (Links)** $\rightarrow$ Connections between the nodes

Computer Programs: Representations
Types of Graphs

- **Undirected graph (a)** → Edges do not have a direction
  - ⇝ Connections between nodes are bidirectional
- **Directed graph (Digraph) (b)** → Edges have a direction
  - ⇝ Go from one node to another → represented as an arrow
- **Weighted graph (c)** → Each edge has an associated weight or cost
  - ⇝ Represents the strength or distance between nodes

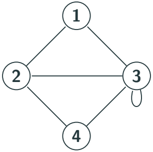


Computer Programs: Representations
Graph Theory

- An edge (u, v) is a **loop** if $u = v$
- For $v_i \in V$ the set of edges $(v_0,\ v_1,\ v_2,\ \dots\ ,\ v_k)$ is a **path** if each pair $(v_i,\ v_{i+1}) \in E\ \forall i = 0, 1, \dots, , k - 1$
  - $\rightsquigarrow$ A path $p = (v_0,\ v_1,\ v_2,\ \dots\ ,\ v_k)$ is called a circle if $v_0 = v_k$
  - $\rightsquigarrow$ The **length** of a path is the number of its edges $\rightarrow |p|$

Computer Programs: Representations
Graph Theory

- The **degree** of a vertex $v \in V$ is the number of edges that are incident to v
  - ⇝ Loops count twice
  - ⇝ $\deg(\mathrm{v}) = |\{(a, b) \in E : a = v \quad or \quad b = v\}|$
  - ⇝ The **max degree** of G is $\Delta(G) = \max\{\deg(\mathrm{v}) : v \in V\}$
  - ⇝ The **min degree** of G is $\delta(G) = \min\{\deg(\mathrm{v}) : v \in V\}$
  - ⇝ The **average degree** of G is denoted as $d(G)$



Computer Programs: Representations
Graph Theory

- G is **connected** if $\forall u, v \in V$ with $u \neq v$ builds a path
  - $\leadsto$ Requires that every pair of vertices in the graph is connected
- **Distance** between of two nodes u, v is the length of the shortest path from u to v

Definition (Tree)

A graph T is a tree if it is connected and free of loops. T becomes disconnected if any edge is removed. A tree with n vertices has $n-1$ edges and a vertex with degree 1 is a leaf.

Computer Programs: Representations

Representations of Graphs

- **Adjacency matrix** $\rightarrow$ with $a_{ij} = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & \text{else} \end{cases}$
  - $\leadsto$ Inefficient data structure
  - $\leadsto$ If E is small the matrix has lots of zeros
- **Adjacency list** $\rightarrow$ Each node v has a (neighborhood) list $L(v)$ with $L(v) = \{u \in V : (v, u) \in E\}$
  - $\leadsto$ Overhead for handling the list data structure

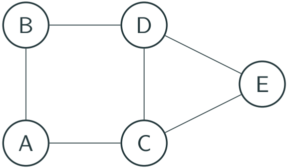
Computer Programs: Representations
Representations of Graphs

Adjacency Matrix

$$
\begin{array}{c c}
\begin{matrix} a & b & c & d & e \end{matrix} & \\
\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} & \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix}
\end{array}
$$

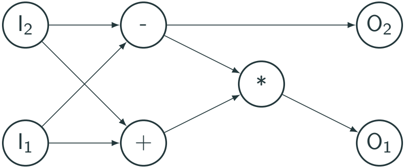
Adjacency List

```
L(a) = ( b, c )
L(b) = ( a, d )
L(c) = ( a, d, e )
L(d) = ( b, c, e )
L(e) = ( d, c )
```



Computer Programs: Representations
Computational Graph

- Can be considered a **generalization** of **expression trees**
- Significantly **better connectivity** required for the representation of many programs and algorithms
- Representation of **multiple outputs** is feasible



Definition (Computational Graph)

A computational graph $C = (V, E, F)$ is a **directed acyclic graph** consisting of a set of functions F, set of vertices V with and set of edges with $E \subseteq V \times V$ where the nodes correspond to operation. Each **computational node** is an element of $V \times F$.

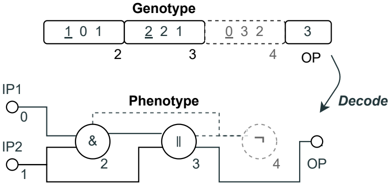
Computer Programs: Representations
Computational Graph

- Computational graphs be used in many for many ML and PS applications:
  - $\rightsquigarrow$ **Back-propagation** $\rightarrow$ automatic differentiation[1]
  - $\rightsquigarrow$ **Symbolic regression** $\rightarrow$ Discovery of mathematical expressions
  - $\rightsquigarrow$ **Logic synthesis** $\rightarrow$ Design of logic circuits with multiple outputs
  - $\rightsquigarrow$ **Neural architecture search** $\rightarrow$ Synthesis of deep neural networks

[1] https://pytorch.org/blog/computational-graphs-constructed-in-pytorch/

Computer Programs: Representations
Computational Graph

- **Cartesian Genetic Programming** [MT00] → Nature-inspired search heuristic for PS
  - ⇝ Adapts **genotype - phenotype** mapping to represent a computational graph
  - ⇝ Vector of integers is decoded to an acyclic directed graph
  - ⇝ Genetic programming will be addressed later in the lecture



Function Lookup Table

Index	Symbol	Function
0	¬	Negation
1	&	Logical and
2	‖	Logical or
3	⊕	Exclusive or

Computer Programs: Representations

Graphs and tree trade off

- **Graphs** → More flexible and versatile than trees
  - ⇝ More complex to handle due to the high degree connectivity
  - ⇝ Use of graphs leads to more complex search spaces
- **Trees** → Natural hierarchical limitation by definition
  - ⇝ Ideal for spanning up less complex search spaces that represent hierarchical candidate programs

References

[MT00] Julian F. Miller and Peter Thomson. “Cartesian Genetic Programming”. In: *Genetic Programming, European Conference, Edinburgh, Scotland, UK, April 15-16, 2000, Proceedings*. Ed. by Riccardo Poli et al. Vol. 1802. Lecture Notes in Computer Science. Springer, 2000, pp. 121–132. DOI: 10.1007/978-3-540-46239-2_9. URL: https://doi.org/10.1007/978-3-540-46239-2%5C_9.