Relations and Graphs

Graphs are Pictures of (Binary) Relations

E. Wenderholm

Department of Computer Science
SUNY Oswego

© 2016 Elaine Wenderholm All rights Reserved
Outline

1 Relation
   - A Function that returns a boolean
   - Special Properties of Relations

2 Graphs
   - A Picture of a Binary Relation
   - Types of Graphs
   - Properties of Graphs

3 Trees
   - Trees are Restricted Graphs
   - Binary Trees
A predicate (also called a property) is function $f : X_1 \times \ldots \times X_n \rightarrow boolean, n \geq 1$ that returns a boolean value.
A predicate (also called a property) is a function $f : X_1 \times \ldots \times X_n \rightarrow boolean, n \geq 1$ that returns a boolean value.

The sets can be different sets, or all the same sets.
A **predicate** (also called a **property**) is function $f : X_1 \times \ldots \times X_n \rightarrow \text{boolean}, n \geq 1$ that returns a boolean value.

- The sets can be different sets, or all the same sets.
- A predicate whose domain is a set of $k$-tuples (of the same set) is called a
  - $k$-ary relation
  - $k$-ary predicate
  - $k$-ary relation on $A$. 
A predicate (also called a property) is function \( f : X_1 \times \ldots \times X_n \rightarrow boolean, n \geq 1 \) that returns a boolean value.

The sets can be different sets, or all the same sets.

A predicate whose domain is a set of \( k \)-tuples (of the same set) is called a

- \( k \)-ary relation
- \( k \)-ary predicate
- \( k \)-ary relation on \( A \).

A predicate defines a partition of the domain.
Predicates

Functions that return a boolean

- A **predicate** (also called a **property**) is function $f : X_1 \times \ldots \times X_n \rightarrow boolean, n \geq 1$ that returns a boolean value.

- The sets can be different sets, or all the same sets.

- A predicate whose domain is a set of $k$-tuples (of the same set) is called a
  - $k$-ary relation
  - $k$-ary predicate
  - $k$-ary relation on $A$.

- A predicate defines a partition of the domain. Some $k$-tuples are mapped to $true$; the rest are mapped to $false$. 
Relations
Definitions

- \( R \subseteq A = (A_1 \times \ldots A_k) = \{(a_1, \ldots, a_k) | R(a_1, \ldots, a_k) = true\} \)
Relations

Definitions

- \( R \subseteq A = (A_1 \times \ldots A_k) = \{(a_1, \ldots, a_k)|R(a_1, \ldots, a_k) = true\} \)
  
  A \( k \)-ary relation \( R \) is a subset of \( A \) in which \( R \) evaluates to \textit{true}.

- If all the \( A_i \) are the same sets, we say that \( R \) is a relation \textit{on} \( A \).

- A 2-\textit{ary} relation is called a \textit{binary relation}.

- A binary relation is often written as an \textit{infix} operator.
$R$ is some binary relation on $A$. We say:

- $R$ is **reflexive** iff $\forall a \in A$, $aRa$. 

An equivalence relation forms a partition of the domain.
$R$ is some binary relation on $A$. We say:

- $R$ is **reflexive** iff $\forall a \in A, aRa$.
- $R$ is **symmetric** iff $\forall a, b \in A, aRb \rightarrow bRa$. 


Relations

Relations with Special Properties

- \( R \) is some binary relation on \( A \). We say:
  - \( R \) is **reflexive** iff \( \forall a \in A, aRa \).
  - \( R \) is **symmetric** iff \( \forall a, b \in A, aRb \rightarrow bRa \).
  - \( R \) is **transitive** iff \( \forall a, b, c \in A, aRb \land bRc \rightarrow aRc \)
• $R$ is some binary relation on $A$. We say:
  • $R$ is reflexive iff $\forall a \in A$, $aRa$.
  • $R$ is symmetric iff $\forall a, b \in A$, $aRb \rightarrow bRa$.
  • $R$ is transitive iff $\forall a, b, c \in A$, $aRb \land bRc \rightarrow aRc$

• $R$ is an equivalence relation if it is: reflexive, symmetric, and transitive.

• An equivalence relation forms a partition of the domain.
Take some binary relation $R$ on $A$.

$R \subseteq A \times A = \{(a_1, a_2) \mid aRb \text{ is true}\}$

A Graph $G = (V, E)$ is:
Directed Graphs
A Picture of a Binary Relation

- Take some binary relation \( R \) on \( A \).
  \[ R \subset A \times A = \{(a_1, a_2)|a R b \text{ is true}\} \]
- A Graph \( G = (V, E) \) is:
  - \( V \) is the set of nodes (Vertices) of the graph.
Directed Graphs
A Picture of a Binary Relation

- Take some binary relation $R$ on $A$.
  \[ R \subseteq A \times A = \{(a_1, a_2) | aRb \text{ is true} \} \]
- A Graph $G = (V, E)$ is:
- $V$ is the set of nodes (Vertices) of the graph. Each node is drawn, perhaps with a dot, with it’s name.
Take some binary relation $R$ on $A$.

$R \subseteq A \times A = \{(a_1, a_2) | aRb \text{ is true } \}$

A Graph $G = (V, E)$ is:

- $V$ is the set of nodes (Vertices) of the graph. Each node is drawn, perhaps with a dot, with its name.
- $E$ is the set of ordered pairs (Edges) for which the relation $R$ is true.
Directed Graphs
A Picture of a Binary Relation

- Take some binary relation $R$ on $A$. 
  $R \subset A \times A = \{(a_1, a_2)|aRb \text{ is true}\}$
- A Graph $G = (V, E)$ is:
  - $V$ is the set of nodes (Vertices) of the graph. Each node is drawn, perhaps with a dot, with it’s name.
  - $E$ is the set of ordered pairs (Edges) for which the relation $R$ is true.
  - The elements in the ordered pair are in $V$. Given the ordered par $(a, b)$, it is drawn as $a \rightarrow b$.
  - A graph is **directed** if it’s edges are drawn with arrows.
Types of Graphs
Properties in Pictures

- All the edges in an **undirected graph** are drawn with straight lines, not arrows.
- An undirected graph is a picture of a symmetric relation.
- A **directed graph** has all directed edges. All edges are drawn with arrows.
- A **labeled** graph has data associated with each edge. This is typically thought of as a cost.
The **degree** of a node is the number of edges that are incident on the node.
The **degree** of a node is the number of edges that are incident on the node. The **in-degree** is the number of incoming edges to the node; the **out-degree** is the number of outgoing edges from the node.

- A **path** is a sequence of nodes connected by edges.
- A **simple path** does not visit any node more than once.
- A **cycle** is a path that starts and ends at the same node.
- A graph that contains no cyclics is called **acyclic**
All Trees are Graphs
All Graphs are not Trees

- A Tree contains no cycles.
- There is a distinguished node, called the **root** of the tree, which has no incoming edges.
- A tree is acyclic.
- There is a unique path from the root to every other node in the tree.
- A **leaf** is a node which has no outgoing edges.
- The **frontier** of a tree a sequence of the leaf nodes of the tree, written in left-to-right order.
- Trees are drawn with the root as the topmost node. Edges are directed, and drawn downward. Edges appear to be symmetric but arrow are directed downward by default.
Binary Trees
A maximum of 2 outgoing edges

All the nodes in Binary Tree have at most two outgoing edges. A Binary Tree is usually defined recursively, as follows.
All the nodes in Binary Tree have at most two outgoing edges. A Binary Tree is usually defined recursively, as follows. Basis: A *Binary Tree* is the Empty Tree.
Binary Trees
A maximum of 2 outgoing edges

All the nodes in Binary Tree have at most two outgoing edges. A Binary Tree is usually defined recursively, as follows. Basis: A *Binary Tree* is the Empty Tree. Recursion: A *Binary Tree* consists of a set of nodes V. A distinguished node R ∈ V, called the root of the tree, has no incoming nodes. The rest of the nodes, V-R, are partitioned into two subsets, called the Left Subtree of R, and the Right Subtree of R, each of which is a *Binary Tree*. 