Constraint Satisfaction Problems

Mathematical Background: Sets, Relations, and Graphs

Malte Helmert and Stefan Wölfl

Albert-Ludwigs-Universität Freiburg

April 17, 19, and 24, 2007

Sets

Sets:

Naive understanding:

a set is a "well-defined" collection of objects.

Sets

Principles (ZF):

- **Extensionality:** Two sets are equal if and only if they contain the same elements.
- **Empty set:** There is a set, $\emptyset$, with no elements.
- **Pairs:** For any pair of sets x, y, $\{x, y\}$ is a set.
- **Union:** For any set x, there exists a set, $\bigcup x$, whose elements are precisely the elements of at least one of the elements of x.
- **Separation:** For any set x and any property $F(y)$, there is a subset of x, $\{y \in x : F(y)\}$, containing precisely the elements y of x for which $F(y)$ holds.
- **Foundation:** Each non-empty set x contains some element y such that x and y are disjoint sets.
- **Power set:** For any set x there exists a set 2^x such that the elements of 2^x are precisely the subsets of x.
- ...(axiom of replacement, infinite set axiom, axiom of choice)

Definitions

Definition

Binary set operations:

$$A \cup B := \{x \,:\, x \in A \text{ or } x \in B\}$$
$$A \cap B := \{x \in A \,:\, x \in B\}$$
$$A \setminus B := \{x \in A \,:\, x \notin B\}$$

$A \subseteq B$, $A \subsetneq B$, etc., are defined as usual.

(Ordered) pairs:

$$(x, y) := \{\{x\}, \{x, y\}\}$$
$$(x_1, \ldots, x_n) := ((x_1, \ldots, x_{n-1}), x_n)$$
$$A \times B := \{(a, b) \,:\, a \in A \text{ and } b \in B\}$$

Boolean Algebra

Definition

A Boolean algebra (with complements) is a set A with

- two binary operations $\cap$, $\cup$,
- a unary operation $-$, and
- two distinct elements 0 and 1

such that for all elements a, b and c of A:

$a \cup (b \cup c) = (a \cup b) \cup c$	$a \cap (b \cap c) = (a \cap b) \cap c$	Ass
$a \cup b = b \cup a$	$a \cap b = b \cap a$	Com
$a \cup (a \cap b) = a$	$a \cap (a \cup b) = a$	Abs
$a \cup (b \cap c) = (a \cup b) \cap (a \cup c)$	$a \cap (b \cup c) = (a \cap b) \cup (a \cap c)$	Dis
$a \cup -a = 1$	$a \cap -a = 0$	Compl

Sets and Boolean Algebras

Definition

A set algebra on a set A is a non-empty subset $B \subseteq 2^A$ that is closed under unions, intersections, and complements.

Note: a set algebra on A contains A and $\emptyset$ as elements.

Lemma

Each set algebra defines a Boolean algebra. Each finite Boolean algebra "can be written as" (is isomorphic to) the full set algebra on some finite set.

Theorem (Tarski)

Each Boolean algebra can be represented as a set algebra.

Relations

Definition

A relation over sets $X_1, \ldots, X_n$ is a subset

$$R \subseteq X_1 \times \cdots \times X_n.$$

The number n is referred to as arity of R.
An n-ary relation on a set X is a subset

$$R \subseteq X^n := X \times \cdots \times X \quad (n \text{ times}).$$

Since relations are sets, set-theoretical operations (union, intersection, complement) can be applied to relations as well.

Binary Relations

For binary relations on a set X we have some special operations:

Definition

Let R, S be binary relations on X.
The converse of relation R is defined by:

$$R^{-1} := \{(x, y) \in X^2 \,:\, (y, x) \in R\}.$$

The composition of relations R and S is defined by:

$$R \circ S := \{(x, z) \in X^2 \,:\, \exists y \in X \text{ s.t. } (x, y) \in R \text{ and } (y, z) \in S\}.$$

The identity relation is:

$$\Delta_X := \{(x, y) \in X^2 \,:\, x = y\}.$$

Relation Algebra

Definition (Tarski)

A relation algebra is a set A with

- binary operations $\cap$, $\cup$, and $\circ$
- unary operations $-$ and $^{-1}$, and
- distinct elements 0, 1, and δ such that

(a) $(A, \cap, \cup, -, 0, 1)$ is a Boolean algebra.

(b) For all elements a, b and c of A:

$$a \circ (b \circ c) = (a \circ b) \circ c$$

$$a \circ (b \cup c) = (a \circ b) \cup (a \circ c)$$

$$\delta \circ a = a \circ \delta = a$$

$$(a^{-1})^{-1} = a \text{ and } (-a)^{-1} = -(a^{-1})$$

$$(a \cup b)^{-1} = a^{-1} \cup b^{-1}$$

$$(a \circ b)^{-1} = b^{-1} \circ a^{-1}$$

$$(a \circ b) \cap c^{-1} = 0 \text{ if and only if } (b \circ c) \cap a^{-1} = 0$$

Relations and Relation Algebras

Definition

An algebra of relations (or: concrete relation algebra) on a set A is a non-empty subset $B \subseteq 2^{A \times A}$ that is closed under unions, intersections, compositions, complements, and converses, and contains Δ_A as an element.

Lemma

Each concrete relation algebra defines a relation algebra.

The converse of the lemma is not true, even if we restrict to finite relation algebras.

Example: Point Algebra

Consider a Boolean algebra A with (exactly) three atoms δ, a, b, i. e., $x \cap y = 0$ for $x, y \in \{\delta, a, b\}$ and $x \neq y$, and $1 = \delta \cup a \cup b$. Define converses of atoms by:

$$^{-1} : \mathrm{Atom}(A) \to \mathrm{Atom}(A), \quad \delta \mapsto \delta,\ a \mapsto b,\ b \mapsto a$$

Furthermore, define composition of atoms

$$\circ : \mathrm{Atom}(A) \times \mathrm{Atom}(A) \to A$$

by a composition table:

$\circ$	δ	a	b
δ	δ	a	b
a	a	a	1
b	b	1	b

Obtain a relation algebra (check it!) by extending these functions to functions $^{-1} : A \to A$ and $\circ : A \times A \to A$ as follows:

$$(x \cup y)^{-1} = x^{-1} \cup y^{-1}$$

$$(x_1 \cup y_1) \circ (x_2 \cup y_2) = (x_1 \circ x_2) \cup (x_1 \circ y_2) \cup (x_2 \cup y_1) \cup (x_2 \cup y_2)$$

Example: Point Algebra

Consider a Boolean algebra A with (exactly) three atoms δ, a, b, i. e., $x \cap y = 0$ for $x, y \in \{\delta, a, b\}$ and $x \neq y$, and $1 = \delta \cup a \cup b$.
Define converses of atoms by:

$$^{-1} : \mathrm{Atom}(A) \to \mathrm{Atom}(A), \quad \delta \mapsto \delta,\ a \mapsto b,\ b \mapsto a$$

Furthermore, define composition of atoms

$$\circ : \mathrm{Atom}(A) \times \mathrm{Atom}(A) \to A$$

by a composition table:

$\circ$	δ	a	b
δ	δ	a	b
a	a	a	1
b	b	1	b

Obtain a relation algebra (check it!) by extending these functions to functions $^{-1} : A \to A$ and $\circ : A \times A \to A$ as follows:

$$(x \cup y)^{-1} = x^{-1} \cup y^{-1}$$

$$(x_1 \cup y_1) \circ (x_2 \cup y_2) = (x_1 \circ x_2) \cup (x_1 \circ y_2) \cup (x_2 \cup y_1) \cup (x_2 \cup y_2)$$

Example: Representing the Point Algebra

Task: Find a concrete relation algebra B (with 8 elements) on some set X and a (bijective) map $\phi : A \to B$ such that for all $x, y \in A$

$$\begin{aligned}
\phi(x * y) &= \phi(x) * \phi(y), \quad \text{for } * \in \{\cap, \cup, \circ\} \\
\phi(-x) &= (X \times X) \setminus \phi(x) \\
\phi(x^{-1}) &= \phi(x)^{-1} \\
\phi(0) &= \emptyset \\
\phi(1) &= X \times X \\
\phi(\delta) &= \Delta_X
\end{aligned}$$

Solution: Consider a dense linear order $(X, <_X)$ without endpoints (e. g., the linear order on $\mathbb{Q}$). Define ϕ by

$$a \mapsto <_X \quad \text{and} \quad b \mapsto >_X .$$

The crucial point to prove is that $\phi(x \circ y) = \phi(x) \circ \phi(y)$.

Example: Representing the Point Algebra

Task: Find a concrete relation algebra B (with 8 elements) on some set X and a (bijective) map $\phi : A \to B$ such that for all $x, y \in A$

$$\begin{aligned}
\phi(x * y) &= \phi(x) * \phi(y), \quad \text{for } * \in \{\cap, \cup, \circ\} \\
\phi(-x) &= (X \times X) \setminus \phi(x) \\
\phi(x^{-1}) &= \phi(x)^{-1} \\
\phi(0) &= \emptyset \\
\phi(1) &= X \times X \\
\phi(\delta) &= \Delta_X
\end{aligned}$$

Solution: Consider a dense linear order $(X, <_X)$ without endpoints (e. g., the linear order on $\mathbb{Q}$). Define ϕ by

$$a \mapsto <_X \quad \text{and} \quad b \mapsto >_X.$$

The crucial point to prove is that $\phi(x \circ y) = \phi(x) \circ \phi(y)$.

Example: The Pentagraph Algebra

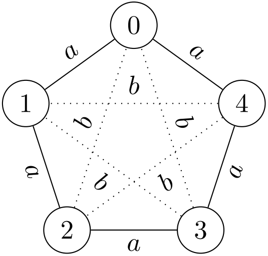
Consider the same Boolean algebra as in the case of the point algebra.
Define converses of atoms by:

$$\delta \mapsto \delta,\ a \mapsto a,\ b \mapsto b.$$

Define composition by:

$\circ$	δ	a	b
δ	δ	a	b
a	a	$\delta \cup b$	$a \cup b$
b	b	$a \cup b$	$\delta \cup a$

The resulting algebra can be represented by a pentagraph:



Relations over Variables

Let V be a set of variables. For each $v \in V$, let $\mathrm{dom}(v)$ (the domain of v) be a non-empty set (of values).

Definition

A relation over (pairwise distinct) variables $v_1, \dots, v_n \in V$ is an $n+1$-tuple

$$R_{v_1,\dots,v_n} := (v_1, \dots, v_n, R)$$

where R is a relation over $\mathrm{dom}(v_1), \dots, \mathrm{dom}(v_n)$.

The sequence $v_1, \dots, v_n$ is referred to as range of $R_{v_1,\dots,v_n}$. R is referred to as graph of $R_{v_1,\dots,v_n}$.

We will not always distinguish between the relation and its graph, e. g., we write

$$R_{v_1,\dots,v_n} \subseteq \mathrm{dom}(v_1) \times \cdots \times \mathrm{dom}(v_n).$$

Constraint Networks

Definition

A constraint network is a triple

$$\mathcal{C} = \langle V, \mathrm{dom}, C \rangle$$

where:

- V is a non-empty and finite set of variables.
- dom is a function that assigns a non-empty (value) set (domain) to each variable $v \in V$.
- C is a set of relations over variables of V (constraints), i. e., each constraint is a relation $R_{v_1,\ldots,v_n}$ over some variables $v_1, \ldots, v_n$ in V.

Solvability of Networks

Definition

A constraint network is solvable (or: satisfiable) if there exists an assignment

$$a : V \to \bigcup_{v \in V} \mathrm{dom}(v)$$

such that

(a) $a(v) \in \mathrm{dom}(v)$, for each $v \in V$,

(b) $(a(v_1), \ldots, a(v_n)) \in R_{v_1,\ldots,v_n}$ for all constraints $R_{v_1,\ldots,v_n}$.

A solution of a constraint network is an assignment that solves the network.

Selections, . . .

Definition

Let $\overline{v} := (v_1, \ldots, v_n)$ and let $R_{\overline{v}}$ be a relation over $\overline{v}$.
Let $a_1 \in \mathrm{dom}(v_{i_1}), \ldots, a_k \in \mathrm{dom}(v_{i_k})$ be fixed values.
Then

$$\sigma_{v_{i_1}=a_1,\ldots,v_{i_k}=a_k}(R_{\overline{v}}) := \Big\{(x_1, \ldots, x_n) \in R_{\overline{v}} \,:\, x_{i_j} = a_j, 1 \leq j \leq k\Big\}$$

is a relation over $\overline{v}$.

The (unary) operation $\sigma_{v_{i_1}=a_1,\ldots,v_{i_k}=a_k}$ is called selection or restriction.

. . . Projections, . . .

Let $(i_1, \ldots, i_k)$ be a k-tuple of pairwise distinct elements of $\{1, \ldots, n\}$ $(k \leq n)$. For an n-tuple $\overline{x} = (x_1, \ldots, x_n)$, define $\overline{x}_{i_1,\ldots,i_k} := (x_{i_1}, \ldots, x_{i_k})$.

Definition

Let $\overline{v} := (v_1, \ldots, v_n)$ and let $R_{\overline{v}}$ be a relation over $\overline{v}$. Then

$$\pi_{v_{i_1},\ldots,v_{i_k}}(R_{\overline{v}}) :=$$
$$\left\{\overline{y} \in \prod_{1 \leq j \leq k} \mathrm{dom}(v_{i_j}) \;:\; \overline{y} = \overline{x}_{i_1,\ldots,i_k}, \text{ for some } \overline{x} \in R_{\overline{v}}\right\}$$

is a relation over $\overline{v}_{i_1,\ldots,i_k}$.

The (unary) operation $\pi_{v_{i_1},\ldots,v_{i_k}}$ is called projection.

. . . Joins

Let $R_{\overline{v}}$ and $S_{\overline{w}}$ be relations over variables $\overline{v}$ and $\overline{w}$, respectively.

For tuples $\overline{x}$ and $\overline{y}$ define:

- $\overline{x} - \overline{y}$: the subsequence of elements in $\overline{x}$ that do not occur in $\overline{y}$.
- $\overline{x} \cap \overline{y}$: the subsequence of $\overline{x}$ with elements that occur in $\overline{y}$.
- $\overline{x} \cup \overline{y}$: the sequence resulting from $\overline{x}$ by adding $\overline{y} - \overline{x}$.

Definition

$$R_{\overline{v}} \bowtie S_{\overline{w}} := \left\{ \overline{x} \cup \overline{y} \;:\; \overline{x} \in R_{\overline{v}},\ \overline{y} \in R_{\overline{w}},\ \text{and } \overline{x}_{\overline{v} \cap \overline{w}} = \overline{y}_{\overline{v} \cap \overline{w}} \right\}$$

is a relation over $\overline{v} \cup \overline{w}$, the join of $R_{\overline{v}}$ and $S_{\overline{w}}$.

Note: For binary relations R and S:

$$R_{x,y} \circ R_{y,z} = \pi_{x,z}(R_{x,y} \bowtie R_{y,z}).$$

Examples

Consider relations $R := R_{x_1,x_2,x_3}$ and $R' := R'_{x_2,x_3,x_4}$ defined by:

x_1	x_2	x_3
b	b	c
c	b	c
c	n	n

x_2	x_3	x_4
a	a	1
b	c	2
b	c	3

Then $\sigma_{x_3=c}(R)$, $\pi_{x_2,x_3}(R)$, $\pi_{x_2,x_1}(R)$, and $R \bowtie R'$ are:

x_1	x_2	x_3
b	b	c
c	b	c

x_2	x_3
b	c
b	c
n	n

x_2	x_1
b	b
b	c
n	c

x_1	x_2	x_3	x_4
b	b	c	2
b	b	c	3
c	b	c	2
c	b	c	3

Normalized Constraint Networks

Let $\mathcal{C} = \langle V, \mathrm{dom}, C \rangle$ be a constraint network.
According to our definition it is possible that C contains constraints

$$R_{v_{i_1},\ldots,v_{i_k}} \quad \text{and} \quad S_{v_{j_1},\ldots,v_{j_k}}$$

where $(j_1, \ldots, j_k)$ is just a permutation of $(i_1, \ldots, i_k)$.

In this case, we can simplify the network by deleting $S_{v_{j_1},\ldots,v_{j_k}}$ from C and rewriting $R_{v_{i_1},\ldots,v_{i_k}}$ as follows:

$$R_{v_{i_1},\ldots,v_{i_k}} \leftarrow R_{v_{i_1},\ldots,v_{i_k}} \cap \pi_{v_{i_1},\ldots,v_{i_k}}(S_{v_{j_1},\ldots,v_{j_k}}).$$

Given an arbitrary order on the set of variables V, we can systematically delete-and-refine constraints. The result is a constraint network that contains *exactly one constraint for each subset of variables*. This network is referred to as a normalized constraint network.

Normalized Constraint Networks

Let $\mathcal{C} = \langle V, \mathrm{dom}, C \rangle$ be a constraint network.
According to our definition it is possible that C contains constraints

$$R_{v_{i_1},\dots,v_{i_k}} \quad \text{and} \quad S_{v_{j_1},\dots,v_{j_k}}$$

where $(j_1, \dots, j_k)$ is just a permutation of $(i_1, \dots, i_k)$.

In this case, we can simplify the network by deleting $S_{v_{j_1},\dots,v_{j_k}}$ from C and rewriting $R_{v_{i_1},\dots,v_{i_k}}$ as follows:

$$R_{v_{i_1},\dots,v_{i_k}} \leftarrow R_{v_{i_1},\dots,v_{i_k}} \cap \pi_{v_{i_1},\dots,v_{i_k}}(S_{v_{j_1},\dots,v_{j_k}}).$$

Given an arbitrary order on the set of variables V, we can systematically delete-and-refine constraints. The result is a constraint network that contains *exactly one constraint for each subset of variables*. This network is referred to as a normalized constraint network.

Normalized Constraint Networks

Let $\mathcal{C} = \langle V, \mathrm{dom}, C \rangle$ be a constraint network.
According to our definition it is possible that C contains constraints

$$R_{v_{i_1},\ldots,v_{i_k}} \quad \text{and} \quad S_{v_{j_1},\ldots,v_{j_k}}$$

where $(j_1, \ldots, j_k)$ is just a permutation of $(i_1, \ldots, i_k)$.

In this case, we can simplify the network by deleting $S_{v_{j_1},\ldots,v_{j_k}}$ from C and rewriting $R_{v_{i_1},\ldots,v_{i_k}}$ as follows:

$$R_{v_{i_1},\ldots,v_{i_k}} \leftarrow R_{v_{i_1},\ldots,v_{i_k}} \cap \pi_{v_{i_1},\ldots,v_{i_k}}(S_{v_{j_1},\ldots,v_{j_k}}).$$

Given an arbitrary order on the set of variables V, we can systematically delete-and-refine constraints. The result is a constraint network that contains *exactly one constraint for each subset of variables*. This network is referred to as a normalized constraint network.

Undirected Graph

Definition

An (undirected) graph is an ordered pair

$$G := \langle V, E \rangle$$

where:

- V is a finite set (of vertices, nodes);
- E is a set of two-element subsets of (not necessarily distinct) nodes (called edges).

The order of a graph is the number of vertices $|V|$. The size of a graph is the number of edges $|E|$. The degree of a vertex is the number of vertices to which it is connected by an edge.

Undirected Graph

Definition

An (undirected) graph is an ordered pair

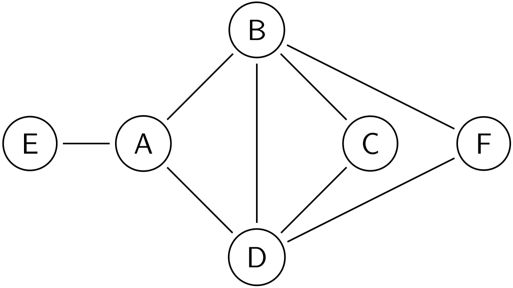
$$G := \langle V, E \rangle$$

where:

- V is a finite set (of vertices, nodes);
- E is a set of two-element subsets of (not necessarily distinct) nodes (called edges).

The order of a graph is the number of vertices $|V|$. The size of a graph is the number of edges $|E|$. The degree of a vertex is the number of vertices to which it is connected by an edge.

Graph: Example



Graph: Definitions

Definition

Let $G = \langle V, E \rangle$ be an undirected graph.

(a) If $e = \{u, v\} \in E$, then u and v are called adjacent (connected by e).

(b) A path in G is a sequence of edges $e_1, \dots, e_k$ such that $e_i \cap e_{i+1} \neq \emptyset$.
Sometimes, paths are defined via vertices:
A path in G is a sequence of vertices $v_0, \dots, v_k$ such that $\{v_{i-1}, v_i\} \in E$ $(1 \leq i \leq k)$. k is the length, v_0 is the start vertex, and v_k is the end vertex of the path.

(c) A cycle is a path $v_0, \dots, v_k$ with $v_0 = v_k$.

(d) A path $v_0, \dots, v_k$ is simple if $v_i \neq v_j$ for all $i \neq j$.

(e) A cycle $v_0, \dots, v_k$ is simple if $v_i \neq v_j$ for all $i, j \geq 1, i \neq j$.

Graph: Definitions

Let $G = \langle V, E \rangle$ be an undirected graph.

Definition

(a) G is connected if, for each pair of vertices u and v, there exists a path from u to v.

(b) G is a tree if G is cycle-free.

(c) G is complete if any pair of vertices is connected.

Definition

Let $G = \langle V, E \rangle$ be an undirected graph. Let S be a subset of V. Then $G_S := \langle S, E_S \rangle$ is called the subgraph relative to S, where

$$E_S := \{\{u, v\} \in E : u, v \in S\}.$$

Definition

A clique in a graph G is a complete subgraph of G.

Graph: Definitions

Let $G = \langle V, E \rangle$ be an undirected graph.

Definition

(a) G is connected if, for each pair of vertices u and v, there exists a path from u to v.

(b) G is a tree if G is cycle-free.

(c) G is complete if any pair of vertices is connected.

Definition

Let $G = \langle V, E \rangle$ be an undirected graph. Let S be a subset of V. Then $G_S := \langle S, E_S \rangle$ is called the subgraph relative to S, where

$$E_S := \big\{\{u, v\} \in E \ : \ u, v \in S\big\}.$$

Definition

A clique in a graph G is a complete subgraph of G.

Graph: Definitions

Let $G = \langle V, E \rangle$ be an undirected graph.

Definition

(a) G is connected if, for each pair of vertices u and v, there exists a path from u to v.

(b) G is a tree if G is cycle-free.

(c) G is complete if any pair of vertices is connected.

Definition

Let $G = \langle V, E \rangle$ be an undirected graph. Let S be a subset of V. Then $G_S := \langle S, E_S \rangle$ is called the subgraph relative to S, where

$$E_S := \big\{\{u, v\} \in E \;:\; u, v \in S\big\}.$$

Definition

A clique in a graph G is a complete subgraph of G.

Examples

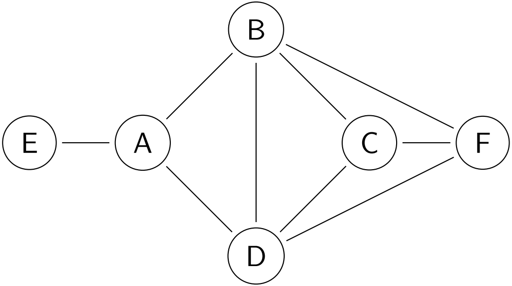


Figure: Example

Examples

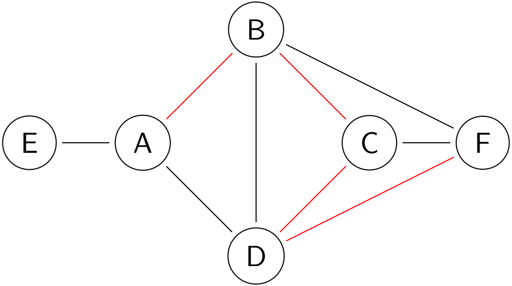


Figure: A path A,B,C,D,F

Examples

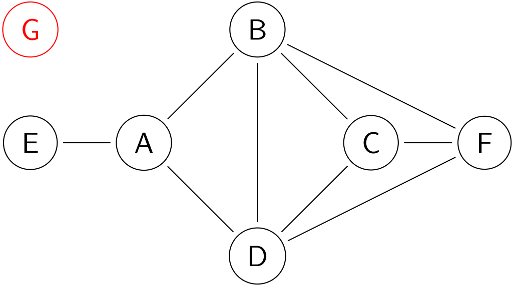


Figure: A non-connected and incomplete graph

Examples

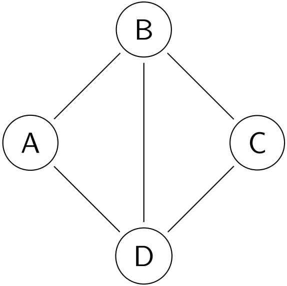


Figure: A subgraph

Examples

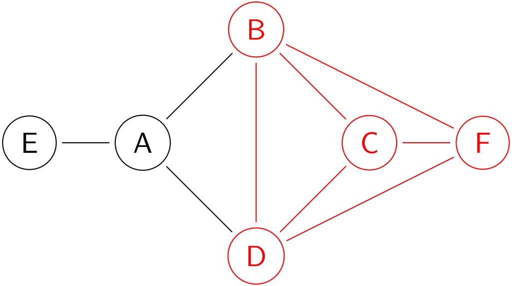


Figure: A clique

Directed Graph

Definition

A directed graph (or: digraph) is an ordered pair

$$G := \langle V, A \rangle$$

where:

- V is a set (of vertices or nodes),
- A is a set of (ordered) pairs of vertices (called arcs, edges, or arrows).

The number of edges with a vertex v as start vertex is called the outdegree of v; the number of vertices with v as end vertex is the indegree of v. Nodes that point to v are called parents, nodes to which an edge from v points are called child nodes.

Directed Graph

Definition

A directed graph (or: digraph) is an ordered pair

$$G := \langle V, A \rangle$$

where:

- V is a set (of vertices or nodes),
- A is a set of (ordered) pairs of vertices (called arcs, edges, or arrows).

The number of edges with a vertex v as start vertex is called the outdegree of v; the number of vertices with v as end vertex is the indegree of v. Nodes that point to v are called parents, nodes to which an edge from v points are called child nodes.

Directed Graph: Definitions

Definition

Let $G = \langle V, A \rangle$ be a directed graph.

(a) A (directed) path is a sequence of arcs $e_1, \ldots, e_k$ such that the end vertex of e_i is the start vertex of e_{i+1} (analogously, (directed) cycle).

(b) A digraph is strongly connected if each pair of nodes u, v is connected by a directed graph from u to v.

(c) A digraph is acyclic if it has no directed cycles.

Digraph: Example

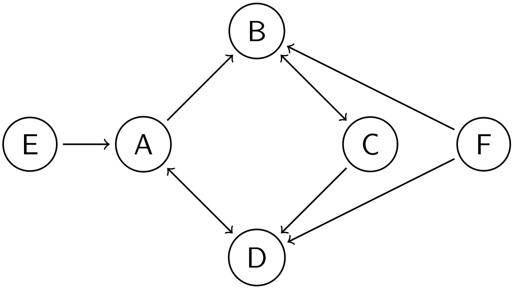


Figure: A directed graph with a strongly connected subgraph

Primal Constraint Graphs

Let $\mathcal{C} = \langle V, \mathrm{dom}, C \rangle$ be a (normalized) constraint network. For a constraint $R_{x_1,\dots,x_k}$, the set $\{x_1, \dots, x_k\}$ is called the scope $R_{x_1,\dots,x_k}$.

Definition

The primal constraint graph of a network $\mathcal{C} = \langle V, \mathrm{dom}, C \rangle$ is the undirected graph

$$G_{\mathcal{C}} := \langle V, E_{\mathcal{C}} \rangle$$

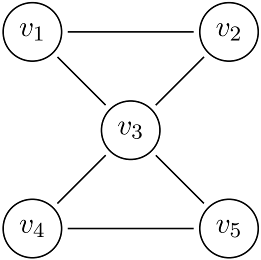
where

$$\{u, v\} \in E_{\mathcal{C}} \iff \{u, v\} \text{ is a subset of the scope of some constraint in } \mathcal{C}.$$

Primal Constraint Graph: Example

Consider a constraint network with variables $v_1, \dots, v_5$ and two ternary constraints R_{v_1,v_2,v_3} and S_{v_3,v_4,v_5}.

Then the primal constraint graph of the network has the form:



Absence of an edge between two variables/nodes means that there is no *direct* constraint between these variables.

Hypergraph

Definition

A hypergraph is a pair

$$H := \langle V, E \rangle$$

where

- V is a set (of nodes, vertices),
- E is a set of non-empty subsets of V (called hyperedges), i. e., $E \subseteq 2^V \setminus \{\emptyset\}$.

Note: Hyperedges can contain an arbitrarily many nodes.

Constraint Hypergraph

Definition

The constraint hypergraph of a constraint network $\mathcal{C} = \langle V, \mathrm{dom}, C \rangle$ is the hypergraph

$$H_{\mathcal{C}} := \langle V, E_{\mathcal{C}} \rangle$$

with

$$X \in E_{\mathcal{C}} \iff X \text{ is the scope of some constraint in } \mathcal{C}.$$

In the example above (constraint network with variables $v_1, \ldots, v_5$ and two ternary constraints R_{v_1,v_2,v_3} and S_{v_3,v_4,v_5}) the hyperedges of the constraint hypergraph are:

$$E_{\mathcal{C}} = \{\{v_1, v_2, v_3\}, \{v_3, v_4, v_5\}\}.$$

Dual Constraint Graphs

Definition

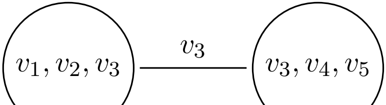
The dual constraint graph of a constraint network $\mathcal{C} = \langle V, \mathrm{dom}, C\rangle$ is the labeled graph

$$D_{\mathcal{C}} := \langle V', E_{\mathcal{C}}, l\rangle$$

with

$$\begin{aligned} X \in V' &\iff X \text{ is the scope of some constraint in } \mathcal{C} \\ \{X, Y\} \in E_{\mathcal{C}} &\iff X \cap Y \neq \emptyset \\ l : E_{\mathcal{C}} \to 2^V, &\quad \{X, Y\} \mapsto X \cap Y \end{aligned}$$

In the example above, the dual constraint graph is



Literature

Rina Dechter.
Constraint Processing,
Chapter 1 and 2, Morgan Kaufmann, 2003

Roger D. Maddux.
Introductory course on relation algebras, finite-dimensional cylindric algebras, and their interconnections,
in: H. Andrka, J. Monk, I. Nmeti (eds.), Algebraic Logic, North-Holland, Amsterdam, 1991, pp. 361-392.

Wikipedia contributors,
Graph theory, Graph (mathematics), Boolean Algebra, Relational Algebra, (2007, April),
In Wikipedia, The Free Encyclopedia. Wikipedia.