

# Constraint Satisfaction Problems

## Constraint Networks

**Bernhard Nebel** and **Stefan Wöfl**

based on a slideset by  
Malte Helmert and Stefan Wöfl  
(summer term 2007)

Albert-Ludwigs-Universität Freiburg

October 26, 2009

Constraint  
Satisfaction  
Problems

Nebel and  
Wöfl

Constraint  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks

# Constraint Networks

## Definition

A **constraint network** is a triple

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

where:

- $V$  is a non-empty and finite set of **variables**.
- $\text{dom}$  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  $R_{v_1, \dots, v_n}$  over some variables  $v_1, \dots, v_n$  in  $V$ .

The set of scopes  $\{S_1, \dots, S_t\}$  is called **network scheme**.

Constraint  
Satisfaction  
Problems

Nebel and  
Wöfl

Constraint  
Networks

Solution  
Normalized  
Constraint  
Networks  
Deduction  
Minimal  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks

# Example: 4-Queens Problem

Consider variables  $v_1, \dots, v_4$  (associated to the columns of a  $4 \times 4$ -chess board).

Each of these variables  $v_i$  has as its domain  $\{1, \dots, 4\}$  (conceived of as the row positions of a queen in column  $i$ ).

	$v_1$	$v_2$	$v_3$	$v_4$
1				
2				
3				
4				

Define then binary constraints (thus encoding possible queen movements):

$$R_{v_1, v_2} := \{(1, 3), (1, 4), (2, 4), (3, 1), (4, 1), (4, 2)\}$$

$$R_{v_1, v_3} := \{(1, 2), (1, 4), (2, 1), (2, 3), (3, 2), (3, 4), (4, 1), (4, 3)\}$$

...

# Graph Representation of Binary Constraint Networks

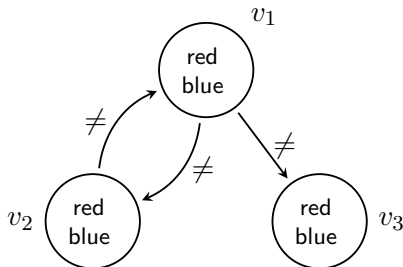
Constraint networks with binary constraints only can be represented by a **directed labelled graph** (even: **an undirected graph** if all constraints are symmetric).

Example: The constraint network defined by:

$$V = \{v_1, v_2, v_3\},$$

$$\text{dom}(v_i) = \{\text{red}, \text{blue}\} \quad (i = 1..3),$$

$$C = \{((v_1, v_2), \neq), ((v_2, v_1), \neq), ((v_1, v_3), \neq)\}$$



Constraint  
Satisfaction  
Problems

Nebel and  
Wölfel

Constraint  
Networks

Solution  
Normalized  
Constraint  
Networks  
Deduction  
Minimal  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks

# Graph Representation of Binary Constraint Networks

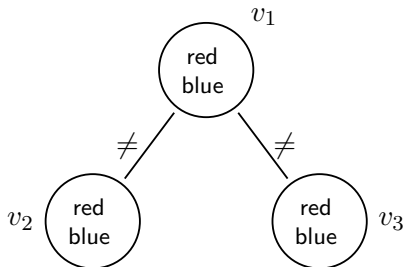
Constraint networks with binary constraints only can be represented by a **directed labelled graph** (even: **an undirected graph** if all constraints are symmetric).

Example: The constraint network defined by:

$$V = \{v_1, v_2, v_3\},$$

$$\text{dom}(v_i) = \{\text{red}, \text{blue}\} \quad (i = 1..3),$$

$$C = \{((v_1, v_2), \neq), ((v_2, v_1), \neq), ((v_1, v_3), \neq)\}$$



Constraint  
Satisfaction  
Problems

Nebel and  
Wölfel

Constraint  
Networks

Solution  
Normalized  
Constraint  
Networks  
Deduction  
Minimal  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks

# Solvability of Networks

## Definition

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

$$a: V \rightarrow \bigcup_{v \in V} \text{dom}(v)$$

such that

- (a)  $a(v) \in \text{dom}(v)$ , for each  $v \in V$ ,
- (b)  $(a(v_1), \dots, a(v_n)) \in R_{v_1, \dots, v_n}$  for all constraints  $R_{v_1, \dots, v_n}$ .

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

# Instantiation, Partial Solution

Let  $\mathcal{C} = \langle V, \text{dom}, C \rangle$  be a constraint network.

## Definition

- (a) An **instantiation** of a subset  $V'$  of  $V$  is an assignment  $a : V' \rightarrow \bigcup_{v \in V'} \text{dom}(v_i)$  with  $a(v_i) \in \text{dom}(v_i)$ .
- (b) An instantiation  $a$  is a **partial solution** if  $a$  satisfies each constraint with scope  $S \subseteq V'$ .  
We also say:  $a$  is **consistent** relative to  $\mathcal{C}$ .
- (c) For an instantiation  $a$  of a subset  $V' = \{v_1, \dots, v_n\}$  and a constraint  $R$  with scope  $S \subseteq V'$ , let

$$\bar{a}[S] := (a(v_1), \dots, a(v_n)).$$

Hence a solution is an instantiation of all variables in  $V$  that is consistent relative to  $\mathcal{C}$ .

# Instantiation, Solution

Note:

- (a) An instantiation of variables in  $V' \subseteq V$ ,  $a$ , is a partial solution (consistent relative to  $\mathcal{C}$ ) iff

$$\bar{a}[S] \in R, \quad \text{for each constraint } R \text{ with scope } S \subseteq V'.$$

- (b) Not every partial solution is part of a (full) solution, i.e., there may be partial solutions of a constraint network that cannot be extended to a solution. For the 4-queens problem, for example,

	$v_1$	$v_2$	$v_3$	$v_4$
1	q			
2			q	
3				
4		q		

Constraint  
Satisfaction  
Problems

Nebel and  
Wölfel

Constraint  
Networks

Solution  
Normalized  
Constraint  
Networks  
Deduction  
Minimal  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks



# Normalized Constraint Networks

Let  $\mathcal{C} = \langle V, \text{dom}, C \rangle$  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  $(i_1, \dots, i_k)$ .

Without changing the set of solutions, 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 a fixed order on the set of variables  $V$ , we can systematically delete-and-refine constraints. The result is a constraint network that contains *at most one constraint for each subset of variables*. This network is referred to as a **normalized constraint network**.

Constraint  
Satisfaction  
Problems

Nebel and  
Wöfl

Constraint  
Networks

Solution

Normalized  
Constraint  
Networks

Deduction  
Minimal  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks

# Normalized Constraint Networks

Let  $\mathcal{C} = \langle V, \text{dom}, C \rangle$  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  $(i_1, \dots, i_k)$ .

Without changing the set of solutions, 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 a fixed order on the set of variables  $V$ , we can systematically delete-and-refine constraints. The result is a constraint network that contains *at most one constraint for each subset of variables*. This network is referred to as a **normalized constraint network**.

Constraint  
Satisfaction  
Problems

Nebel and  
Wöfl

Constraint  
Networks

Solution

Normalized  
Constraint  
Networks

Deduction  
Minimal  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks

# Normalized Constraint Networks

Let  $\mathcal{C} = \langle V, \text{dom}, C \rangle$  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  $(i_1, \dots, i_k)$ .

Without changing the set of solutions, 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 a fixed order on the set of variables  $V$ , we can systematically delete-and-refine constraints. The result is a constraint network that contains *at most one constraint for each subset of variables*. This network is referred to as a **normalized constraint network**.

Constraint  
Satisfaction  
Problems

Nebel and  
Wöfl

Constraint  
Networks

Solution

Normalized  
Constraint  
Networks

Deduction  
Minimal  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks

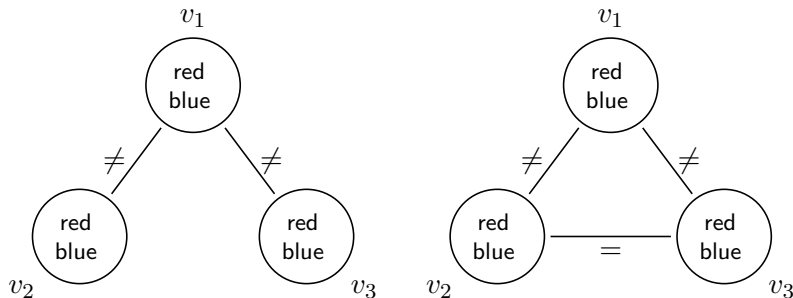
# Equivalence

Let  $\mathcal{C}$  and  $\mathcal{C}'$  be constraint networks on the same set of variables and on the same domains for each variable.

## Definition

$\mathcal{C}$  and  $\mathcal{C}'$  are **equivalent** if each solution of  $\mathcal{C}$  is a solution of  $\mathcal{C}'$ , and vice versa.

Example:



# Tightness

Let  $\mathcal{C}$  and  $\mathcal{C}'$  be (normalized) constraint networks on the same set of variables and on the same domains for each variable.

## Definition

$\mathcal{C}$  is **as tight as**  $\mathcal{C}'$  if for each constraint  $R$  of  $\mathcal{C}$  with scope  $S$ ,

- (a)  $\mathcal{C}'$  has no constraint with scope  $S$ , or
- (b)  $R \subseteq R'$ , where  $R'$  is the constraint of  $\mathcal{C}'$  with scope  $S$ .

Notes:

- Constraint tightness has a large influence on the solubility of constraint networks.
- Be warned: different concepts of tightness can be found in the literature
- Here: Tightness does not account for comparing constraints with different arities

Constraint  
Satisfaction  
Problems

Nebel and  
Wöfl

Constraint  
Networks

Solution  
Normalized  
Constraint  
Networks

Deduction  
Minimal  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks

# Intersection of Networks

Let  $\mathcal{C}$  and  $\mathcal{C}'$  be constraint networks as above.

## Definition

The **intersection** of  $\mathcal{C}$  and  $\mathcal{C}'$ ,  $\mathcal{C} \cap \mathcal{C}'$ , is the network defined by intersecting for each scope  $S$  of constraints  $R_S \in \mathcal{C}$  and  $R'_S \in \mathcal{C}'$  the respective relations, i.e.,

$$R''_S := R_S \cap R'_S.$$

If for a scope  $S$  only one of the networks contains a constraint, then we set:

$$R''_S := R_S \quad (\text{or } := R'_S, \text{ resp.})$$

## Lemma

*If  $\mathcal{C}$  and  $\mathcal{C}'$  are equivalent networks, then  $\mathcal{C} \cap \mathcal{C}'$  is equivalent to both networks and as tight as both networks.*

Constraint  
Satisfaction  
Problems

Nebel and  
Wölfel

Constraint  
Networks

Solution  
Normalized  
Constraint  
Networks

Deduction  
Minimal  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks

# Minimal Network

## Definition

Let  $\mathcal{C}_0$  be a constraint network and let  $\mathcal{C}_1 \dots, \mathcal{C}_k$  be the set of *all* constraint networks (defined on the same set of variables and the same domains) that are equivalent to  $\mathcal{C}_0$ .

$$\bigcap_{1 \leq i \leq k} \mathcal{C}_i$$

is the **minimal network** of  $\mathcal{C}_0$ .

## Lemma

*The minimal network is equivalent to and as tight as all the constraint networks  $\mathcal{C}_i$ . There is no network equivalent to  $\mathcal{C}_0$  that is tighter than the minimal network.*

Constraint  
Satisfaction  
Problems

Nebel and  
Wölfel

Constraint  
Networks

Solution

Normalized  
Constraint  
Networks

Deduction  
Minimal  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks

# Projecting Relations

Let  $R_S$  be a relation with scope  $S = \{v_1, \dots, v_k\}$  (we can think of  $R_S$  as a constraint network ...).

## Definition

The **projection network** of  $R_S$ ,  $\text{Proj}(R_S)$ , is the constraint network defined by:

$$\begin{aligned}V &:= S \\ \text{dom}(v_i) &:= \pi_{v_i}(R_S) \\ R_{v_i, v_j} &:= \pi_{v_i, v_j}(R_S)\end{aligned}$$

Note: The projection network is an upper approximation by **binary networks** in the following sense:

## Lemma

*Any solution of  $R_S$  (as a network) defines a solution of  $\text{Proj}(R_S)$ .*

Constraint  
Satisfaction  
Problems

Nebel and  
Wöfl

Constraint  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks



# Binary Representation

## Definition

A relation  $R_S$  with scope  $S$  has a **binary representation** if the relation (conceived of as a network) is equivalent to  $\text{Proj}(R_S)$ .

From the fact that a relation has a binary representation, it does not follow that all its projections have binary representations as well (Exercise!).

## Definition

A relation  $R_S$  with scope  $S$  is **binary decomposable** if the relation itself and all its projections to subsets of  $S$  (with at least 3 elements) have a binary representation.

Constraint  
Satisfaction  
Problems

Nebel and  
Wöfl

Constraint  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks

# Primal Constraint Graphs

Let  $\mathcal{C} = \langle V, \text{dom}, C \rangle$  be a (normalized) constraint network.

## Definition

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

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

where

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

Constraint  
Satisfaction  
Problems

Nebel and  
Wölfel

Constraint  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Primal  
Constraint  
Graphs

Dual Constraint  
Graph

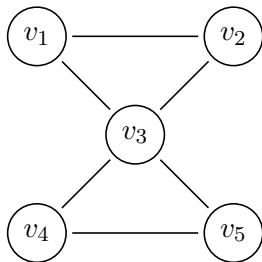
Constraint  
Hypergraph

Solving  
Constraint  
Networks

# 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.

# Dual Constraint Graphs

## Definition

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

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

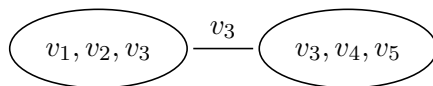
with

$X \in V' \iff X$  is the scope of some constraint in  $\mathcal{C}$

$\{X, Y\} \in E_{\mathcal{C}} \iff X \cap Y \neq \emptyset$

$l : E_{\mathcal{C}} \rightarrow 2^V, \quad \{X, Y\} \mapsto X \cap Y$

In the example above, the dual constraint graph is:



# Constraint Hypergraph

## Definition

The **constraint hypergraph** of a constraint network  $\mathcal{C} = \langle V, \text{dom}, C \rangle$  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  $v_1, \dots, v_5$  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:

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

Constraint  
Satisfaction  
Problems

Nebel and  
Wöfl

Constraint  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Primal  
Constraint  
Graphs

Dual Constraint  
Graph

Constraint  
Hypergraph

Solving  
Constraint  
Networks

# Simple Solution Strategy: Guess and Check

**Backtracking:** search systematically for consistent partial instantiations in a depth-first manner:

- **forward phase:** extend the current partial solution by assigning a consistent value to some new variable (if possible)
- **backward phase:** if no consistent instantiation for the current variable exists, we return to the previous variable.

# Backtracking Algorithm

**Backtracking**( $\mathcal{C}, a$ ):

---

*Input:* a constraint network  $\mathcal{C} = \langle V, D, C \rangle$  and  
a partial assignment  $a$  of  $\mathcal{C}$   
(e.g., the empty instantiation  $a = \{ \}$ )

*Output:* a solution of  $\mathcal{C}$  or “inconsistent”

**if**  $a$  is not consistent with  $\mathcal{C}$ :

**return** “inconsistent”

**if**  $a$  is defined for all variables in  $V$ :

**return**  $a$

select **some variable**  $v_i$  for which  $a$  is not defined

**for** **each value**  $x$  from  $D_i$ :

$a' := a \cup \{v_i \mapsto x\}$

$a'' \leftarrow \text{Backtracking}(\mathcal{C}, a')$

**if**  $a''$  is not “inconsistent”:

**return**  $a''$

**return** “inconsistent”

Constraint  
Satisfaction  
Problems

Nebel and  
Wölfel

Constraint  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks



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

Constraint  
Satisfaction  
Problems

Nebel and  
Wölfel

Constraint  
Networks

Projection  
Networks

Constraint  
Networks and  
Graphs

Solving  
Constraint  
Networks