

Foundations of Artificial Intelligence

23. Constraint Satisfaction Problems: Constraint Networks

Thomas Keller and Florian Pommerening

University of Basel

April 5, 2023

Constraint Satisfaction Problems: Overview

Chapter overview: constraint satisfaction problems

- 22.–23. Introduction
 - 22. Introduction and Examples
 - 23. Constraint Networks
- 24.–26. Basic Algorithms
- 27.–28. Problem Structure

Constraint Networks

Constraint Networks: Informally

Constraint Networks: Informal Definition

A **constraint network** is defined by

- a finite set of **variables**
- a finite **domain** for each variable
- a set of **constraints** (here: **binary relations**)

The objective is to find a **solution** for the constraint network, i.e., an assignment of the variables that complies with all constraints.

Informally, people often just speak of **constraint satisfaction problems (CSP)** instead of constraint networks.

More formally, a “CSP” is the algorithmic problem of finding a solution for a constraint network.

Constraint Networks: Formally

Definition (binary constraint network)

A **(binary) constraint network**

is a 3-tuple $\mathcal{C} = \langle V, \text{dom}, (R_{uv}) \rangle$ such that:

- V is a non-empty and finite set of **variables**,
- dom is a function that assigns a non-empty and finite **domain** to each variable $v \in V$, and
- $(R_{uv})_{u,v \in V, u \neq v}$ is a family of binary relations (**constraints**) over V where for all $u \neq v$: $R_{uv} \subseteq \text{dom}(u) \times \text{dom}(v)$

possible generalizations:

- infinite domains (e.g., $\text{dom}(v) = \mathbb{Z}$)
- constraints of higher arity
(e.g., satisfiability in propositional logic)

Variables and Domains

Running Example (informal)

- assign a value from $\{1, 2, 3, 4\}$ to the variables w and y
- and from $\{1, 2, 3\}$ to x and z
- such that ...

Running Example (formal)

$\mathcal{C} = \langle V, \text{dom}, (R_{uv}) \rangle$ with

- $V = \{w, x, y, z\}$
- $\text{dom}(w) = \text{dom}(y) = \{1, 2, 3, 4\}$
- $\text{dom}(x) = \text{dom}(z) = \{1, 2, 3\}$
- ...

Binary Constraints (1)

binary constraints:

- For variables u, v , the constraint R_{uv} expresses which **joint assignments** to u and v are allowed in a solution.

Running Example (informal)

- ... such that
 - ..., $w < z$, ...

Running Example (formal)

..., $R_{wz} = \{\langle 1, 2 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}, \dots$

Binary Constraints (2)

binary constraints:

- If $R_{uv} = \text{dom}(u) \times \text{dom}(v)$, the constraint is **trivial**: there is no restriction, and the constraint is typically not given explicitly in the constraint network description (although it formally always exists!).

Running Example

$$\begin{aligned} \dots, R_{xz} = \{ & \langle 1, 1 \rangle, \langle 1, 2 \rangle, \langle 1, 3 \rangle, \\ & \langle 2, 1 \rangle, \langle 2, 2 \rangle, \langle 2, 3 \rangle, \\ & \langle 3, 1 \rangle, \langle 3, 2 \rangle, \langle 3, 3 \rangle \}, \dots \end{aligned}$$

Binary Constraints (3)

binary constraints:

- Constraints R_{uv} and R_{vu} refer to the same variables.
Hence, usually only one of them is given in the description.

Running Example (informal)

- ... such that
 - ..., $w < z$, ...

Running Example (formal)

..., $R_{wz} = \{\langle 1, 2 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}, \dots$
 ..., $R_{zw} = \{\langle 2, 1 \rangle, \langle 3, 1 \rangle, \langle 3, 2 \rangle\}, \dots$

Unary Constraints

unary constraints:

- It is often useful to have additional restrictions on **single** variables as constraints.
- Such constraints are called **unary** constraints.
- A unary constraint R_v for $v \in V$ corresponds to a restriction of $\text{dom}(v)$ to the values allowed by R_v .
- Formally, unary constraints are not necessary, but they often allow us to describe constraint networks more clearly.

Running Example

$\text{dom}(z) = \{1, 2, 3\}$ could be described as
 $\text{dom}(z) = \{1, 2, 3, 4\}, R_z = \{1, 2, 3\}$

Example

Full Formal Model of Running Example

$\mathcal{C} = \langle V, \text{dom}, (R_{uv}) \rangle$ with

- variables:

$$V = \{w, x, y, z\}$$

- domains:

$$\text{dom}(w) = \text{dom}(y) = \{1, 2, 3, 4\}$$

$$\text{dom}(x) = \text{dom}(z) = \{1, 2, 3\}$$

- constraints:

$$R_{wx} = \{\langle 2, 1 \rangle, \langle 4, 2 \rangle\}$$

$$R_{wz} = \{\langle 1, 2 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}$$

$$R_{yz} = \{\langle 2, 1 \rangle, \langle 3, 1 \rangle, \langle 3, 2 \rangle, \\ \langle 4, 1 \rangle, \langle 4, 2 \rangle, \langle 4, 3 \rangle\}$$

Compact Encodings and General Constraint Solvers

Constraint networks allow for **compact encodings** of large sets of assignments:

- Consider a network with n variables with domains of size k .

↪ k^n assignments

Compact Encodings and General Constraint Solvers

Constraint networks allow for **compact encodings** of large sets of assignments:

- Consider a network with n variables with domains of size k .

~> k^n assignments

- For the **description** as constraint network, at most $\binom{n}{2}$, i.e., $O(n^2)$ constraints have to be provided.

Every constraint in turn consists of at most $O(k^2)$ pairs.

~> encoding size $O(n^2 k^2)$

- We observe: The number of assignments is **exponentially larger** than the description of the constraint network.

Compact Encodings and General Constraint Solvers

Constraint networks allow for **compact encodings** of large sets of assignments:

- Consider a network with n variables with domains of size k .

~> k^n assignments

- For the **description** as constraint network, at most $\binom{n}{2}$, i.e., $O(n^2)$ constraints have to be provided.

Every constraint in turn consists of at most $O(k^2)$ pairs.

~> encoding size $O(n^2 k^2)$

- We observe: The number of assignments is **exponentially larger** than the description of the constraint network.
- As a consequence, such descriptions can be used as inputs of **general** constraint solvers.

Examples

Example: 4 Queens Problem

4 Queens Problem as Constraint Network

- **variables:** $V = \{v_1, v_2, v_3, v_4\}$
 v_i encodes the rank of the queen in the i -th file
- **domains:**
 $\text{dom}(v_1) = \text{dom}(v_2) = \text{dom}(v_3) = \text{dom}(v_4) = \{1, 2, 3, 4\}$
- **constraints:** for all $1 \leq i < j \leq 4$, we set: $R_{v_i, v_j} = \{\langle k, l \rangle \in \{1, 2, 3, 4\} \times \{1, 2, 3, 4\} \mid k \neq l \wedge |k - l| \neq |i - j|\}$
e.g. $R_{v_1, v_3} = \{\langle 1, 2 \rangle, \langle 1, 4 \rangle, \langle 2, 1 \rangle, \langle 2, 3 \rangle, \langle 3, 2 \rangle, \langle 3, 4 \rangle, \langle 4, 1 \rangle, \langle 4, 3 \rangle\}$

	v_1	v_2	v_3	v_4
1				
2				
3				
4				

Example: Sudoku

Sudoku as Constraint Network

- **variables:** $V = \{v_{ij} \mid 1 \leq i, j \leq 9\}$; v_{ij} : Value row i , column j
- **domains:** $\text{dom}(v) = \{1, \dots, 9\}$ for all $v \in V$
- **unary constraints:** $R_{v_{ij}} = \{k\}$,
if $\langle i, j \rangle$ is a cell with predefined value k
- **binary constraints:** for all $v_{ij}, v_{i'j'} \in V$, we set
 $R_{v_{ij}v_{i'j'}} = \{\langle a, b \rangle \in \{1, \dots, 9\} \times \{1, \dots, 9\} \mid a \neq b\}$,
if $i = i'$ (same row), or $j = j'$ (same column),
or $\langle \lceil \frac{i}{3} \rceil, \lceil \frac{j}{3} \rceil \rangle = \langle \lceil \frac{i'}{3} \rceil, \lceil \frac{j'}{3} \rceil \rangle$ (same block)

2	5			3		9		1
	1					4		
4		7				2		8
		5	2					
				9	8	1		
	4				3			
			3	6			7	2
	7							3
9		3				6		4

Assignments and Consistency

Assignments

Definition (assignment, partial assignment)

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

A **partial assignment** of \mathcal{C} (or of V) is a function

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

with $V' \subseteq V$ and $\alpha(v) \in \text{dom}(v)$ for all $v \in V'$.

If $V' = V$, then α is also called **total assignment** (or **assignment**).

~> **partial assignments** assign values to some or to all variables

~> (total) **assignments** are defined on all variables

Example

Partial Assignments of Running Example

$$\alpha_1 = \{w \mapsto 1, z \mapsto 2\}$$

$$\alpha_2 = \{w \mapsto 3, x \mapsto 1\}$$

Total Assignments of Running Example

$$\alpha_3 = \{w \mapsto 1, x \mapsto 1, y \mapsto 2, z \mapsto 2\}$$

$$\alpha_4 = \{w \mapsto 2, x \mapsto 1, y \mapsto 4, z \mapsto 3\}$$

Consistency

Definition (inconsistent, consistent, violated)

A partial assignment α of a constraint network \mathcal{C} is called **inconsistent** if there are variables u, v such that α is defined for both u and v , and $\langle \alpha(u), \alpha(v) \rangle \notin R_{uv}$.

In this case, we say α **violates** the constraint R_{uv} .

A partial assignment is called **consistent** if it is not inconsistent.

trivial example: The empty assignment is always consistent.

Example

Consistent Partial Assignment

$$\alpha_1 = \{w \mapsto 1, z \mapsto 2\}$$

Inconsistent Partial Assignment

$$\alpha_2 = \{w \mapsto 2, x \mapsto 2\}$$

violates $R_{wx} = \{\langle 2, 1 \rangle, \langle 4, 2 \rangle\}$

Inconsistent Assignment

$$\alpha_3 = \{w \mapsto 2, x \mapsto 1, y \mapsto 2, z \mapsto 2\}$$

violates $R_{wz} = \{\langle 1, 2 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}$ and

$$R_{yz} = \{\langle 2, 1 \rangle, \langle 3, 1 \rangle, \langle 3, 2 \rangle, \langle 4, 1 \rangle, \langle 4, 2 \rangle, \langle 4, 3 \rangle\}$$

Solution

Definition (solution, solvable)

Let \mathcal{C} be a constraint network.

A consistent and total assignment of \mathcal{C} is called a **solution** of \mathcal{C} .

If a solution of \mathcal{C} exists, \mathcal{C} is called **solvable**.

If no solution exists, \mathcal{C} is called **inconsistent**.

Solution of the Running Example

$$\alpha = \{w \mapsto 2, x \mapsto 1, y \mapsto 4, z \mapsto 3\}$$

Consistency vs. Solvability

Note: Consistent partial assignments α **cannot necessarily** be extended to a solution.

It only means that **so far** (i.e., on the variables where α is defined) no constraint is violated.

Trivial Example: empty assignment on unsolvable problem

Example (4 queens problem): $\alpha = \{v_1 \mapsto 1, v_2 \mapsto 4, v_3 \mapsto 2\}$

	v_1	v_2	v_3	v_4
1	q			
2			q	
3				
4		q		

Complexity of Constraint Satisfaction Problems

Proposition (CSPs are NP-complete)

It is an NP-complete problem to decide whether a given constraint network is solvable.

Proof

Membership in NP:

Guess and check: guess a solution and check it for validity.
This can be done in polynomial time in the size of the input.

NP-hardness:

The graph coloring problem is a special case of CSPs and is already known to be NP-complete.

Tightness of Constraint Networks

Definition (tighter, strictly tighter)

Let $\mathcal{C} = \langle V, \text{dom}, R_{uv} \rangle$ and $\mathcal{C}' = \langle V, \text{dom}', R'_{uv} \rangle$ be constraint networks with equal variable sets V .

\mathcal{C} is called **tighter** than \mathcal{C}' , in symbols $\mathcal{C} \sqsubseteq \mathcal{C}'$, if

- $\text{dom}(v) \subseteq \text{dom}'(v)$ for all $v \in V$
- $R_{uv} \subseteq R'_{uv}$ for all $u, v \in V$ (including trivial constraints).

If at least one of these subset equations is strict, then \mathcal{C} is called **strictly tighter** than \mathcal{C}' , in symbols $\mathcal{C} \sqsubset \mathcal{C}'$.

Equivalence of Constraint Networks

Definition (equivalent)

Let \mathcal{C} and \mathcal{C}' be constraint networks with equal variable sets.

\mathcal{C} and \mathcal{C}' are called **equivalent**, in symbols $\mathcal{C} \equiv \mathcal{C}'$, if they have the same solutions.

Outline and Summary

CSP Algorithms

In the following chapters, we will consider **solution algorithms** for constraint networks.

basic concepts:

- **search**: check partial assignments systematically
- **backtracking**: discard inconsistent partial assignments
- **inference**: derive equivalent, but tighter constraints to reduce the size of the search space

Summary

- formal definition of **constraint networks**:
variables, domains, constraints
- **compact encodings** of exponentially many configurations
- **unary** and **binary** constraints
- **assignments**: partial and total
- **consistency** of assignments; **solutions**
- deciding solvability is **NP-complete**
- **tightness** of constraints
- **equivalence** of constraints