

# Foundations of Artificial Intelligence

## D4. Constraint Satisfaction Problems: Inference and Forward Checking

Malte Helmert

University of Basel

April 8, 2026

# Foundations of Artificial Intelligence

April 8, 2026 — D4. Constraint Satisfaction Problems: Inference and Forward Checking

D4.1 Inference

D4.2 Forward Checking

D4.3 Summary

## Constraint Satisfaction Problems: Overview

Chapter overview: constraint satisfaction problems

- ▶ D1–D2. Introduction
- ▶ D3–D6. Basic Algorithms
  - ▶ D3. Backtracking
  - ▶ D4. Inference and Forward Checking
  - ▶ D5. Arc Consistency
  - ▶ D6. Path Consistency
- ▶ D7–D8. Problem Structure

## D4.1 Inference

## Inference

## Inference

Derive additional constraints (here: unary or binary) that are implied by the given constraints, i.e., that are satisfied in all solutions.

## Inference: Example

## Running Example

binary constraints:

- ▶  $R_{wx} = \{\langle 2, 1 \rangle, \langle 4, 2 \rangle\}$
- ▶  $R_{wz} = \{\langle 1, 2 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}$
- ▶  $R_{yz} = \{\langle 2, 1 \rangle, \langle 3, 1 \rangle, \langle 3, 2 \rangle, \langle 4, 1 \rangle, \langle 4, 2 \rangle, \langle 4, 3 \rangle\}$

domains (unary constraints):

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

Can we use the constraint  $R_{wz}$  ( $w < z$ ) to come up with a unary constraint  $R_w$ ?

## Inference: Example

## Running Example

binary constraints:

- ▶  $R_{wx} = \{\langle 2, 1 \rangle, \langle 4, 2 \rangle\}$
- ▶  $R_{wz} = \{\langle 1, 2 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}$
- ▶  $R_{yz} = \{\langle 2, 1 \rangle, \langle 3, 1 \rangle, \langle 3, 2 \rangle, \langle 4, 1 \rangle, \langle 4, 2 \rangle, \langle 4, 3 \rangle\}$

domains (unary constraints):

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

Can we use the constraint  $R_{wz}$  ( $w < z$ ) to come up with a unary constraint  $R_w$ ?

↪ tighten domain with unary constraint (sometimes called node consistency)

## Inference: Example

## Running Example

binary constraints:

- ▶  $R_{wx} = \{\langle 2, 1 \rangle\}$
- ▶  $R_{wz} = \{\langle 1, 2 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}$
- ▶  $R_{yz} = \{\langle 2, 1 \rangle, \langle 3, 1 \rangle, \langle 3, 2 \rangle, \langle 4, 1 \rangle, \langle 4, 2 \rangle, \langle 4, 3 \rangle\}$

domains (unary constraints):

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

How does this affect the binary constraint  $R_{wz}$ ?

## Inference: Example

### Running Example

binary constraints:

- ▶  $R_{wx} = \{\langle 2, 1 \rangle\}$
- ▶  $R_{wz} = \{\langle 1, 2 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}$
- ▶  $R_{yz} = \{\langle 2, 1 \rangle, \langle 3, 1 \rangle, \langle 3, 2 \rangle, \langle 4, 1 \rangle, \langle 4, 2 \rangle, \langle 4, 3 \rangle\}$
- ▶  $R_{yy} = \{\langle 1, 3 \rangle, \langle 1, 4 \rangle, \langle 2, 4 \rangle\}$

domains (unary constraints):

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

Can we generate a “new” binary constraint from  $w < z$  and  $z < y$ ?  
(i.e., tighten a trivial constraint)

## Trade-Off Search vs. Inference

Inference formally

For a given constraint network  $\mathcal{C}$ , replace  $\mathcal{C}$  with an **equivalent**, but **tighter** constraint network.

Trade-off:

- ▶ the **more complex** the inference, and
- ▶ the **more often** inference is applied,
- ▶ the **smaller** the resulting state space, but
- ▶ the **higher** the complexity **per search node**.

## When to Apply Inference?

different possibilities to apply inference:

- ▶ once as **preprocessing** before search
- ▶ **combined with search**: before recursive calls during backtracking procedure
  - ▶ already assigned variable  $v \mapsto d$  corresponds to  $\text{dom}(v) = \{d\}$ 
    - ↪ more inferences possible
  - ▶ during backtracking, derived constraints have to be **retracted** because they were based on the given assignment
    - ↪ powerful, but possibly expensive

## Backtracking with Inference

**function** BacktrackingWithInference( $\mathcal{C}, \alpha$ ):

**if**  $\alpha$  is inconsistent with  $\mathcal{C}$ :

**return inconsistent**

**if**  $\alpha$  is a total assignment:

**return**  $\alpha$

$\mathcal{C}' := \langle V, \text{dom}', (R'_{uv}) \rangle := \text{copy of } \mathcal{C}$

apply inference to  $\mathcal{C}'$

**if**  $\text{dom}'(v) \neq \emptyset$  for all variables  $v$ :

select **some variable**  $v$  for which  $\alpha$  is not defined

**for each**  $d \in \text{copy of } \text{dom}'(v)$  **in some order**:

$\alpha' := \alpha \cup \{v \mapsto d\}$

$\text{dom}'(v) := \{d\}$

$\alpha'' := \text{BacktrackingWithInference}(\mathcal{C}', \alpha')$

**if**  $\alpha'' \neq \text{inconsistent}$ :

**return**  $\alpha''$

**return inconsistent**

## Backtracking with Inference: Discussion

- ▶ **Inference** is a placeholder: different inference methods can be applied.
- ▶ Inference methods can recognize unsolvability (given  $\alpha$ ) and indicate this by clearing the domain of a variable.
- ▶ Efficient implementations of inference are often **incremental**: the last assigned variable/value pair  $v \mapsto d$  is taken into account to speed up the inference computation.

## D4.2 Forward Checking

## Forward Checking

We start with a simple inference method:

### Forward Checking

Let  $\alpha$  be a partial assignment.

**Inference:** For all unassigned variables  $v$  in  $\alpha$ , remove all values from the domain of  $v$  that are in conflict with already assigned variable/value pairs in  $\alpha$ .

$\rightsquigarrow$  definition of **conflict** as in the previous chapter

### Incremental computation:

- ▶ When adding  $v \mapsto d$  to the assignment, delete all pairs that conflict with  $v \mapsto d$ .

## Forward Checking: Example

### Running Example

Removing values in conflict with  $\alpha = \{w \mapsto 2\}$ :

binary constraints:

- ▶  $R_{wx} = \{\langle 2, 1 \rangle, \langle 4, 2 \rangle\}$
- ▶  $R_{wz} = \{\langle 1, 2 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}$
- ▶  $R_{yz} = \{\langle 2, 1 \rangle, \langle 3, 1 \rangle, \langle 3, 2 \rangle, \langle 4, 1 \rangle, \langle 4, 2 \rangle, \langle 4, 3 \rangle\}$

domains:

- ▶  $w$  is already assigned
- ▶  $\text{dom}(x) = \{1, 2, 3\}$
- ▶  $\text{dom}(y) = \{1, 2, 3, 4\}$
- ▶  $\text{dom}(z) = \{1, 2, 3\}$

## Forward Checking: Example

### Running Example

Removing values in conflict with  $\alpha = \{w \mapsto 2\}$ :

binary constraints:

- ▶  $R_{wx} = \{\langle 2, 1 \rangle, \langle 4, 2 \rangle\}$
- ▶  $R_{wz} = \{\langle 1, 2 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}$
- ▶  $R_{yz} = \{\langle 2, 1 \rangle, \langle 3, 1 \rangle, \langle 3, 2 \rangle, \langle 4, 1 \rangle, \langle 4, 2 \rangle, \langle 4, 3 \rangle\}$

domains:

- ▶  $w$  is already assigned
- ▶  $\text{dom}(x) = \{1\}$
- ▶  $\text{dom}(y) = \{1, 2, 3, 4\}$
- ▶  $\text{dom}(z) = \{3\}$

## Forward Checking: Discussion

properties of forward checking:

- ▶ correct inference method (retains equivalence)
- ▶ affects domains (= unary constraints), but not binary constraints
- ▶ consistency check at the beginning of the backtracking procedure no longer needed (**Why?**)
- ▶ cheap, but often still useful inference method
- ↔ apply at least forward checking in the backtracking procedure

In the next chapters, we consider more powerful inference methods.

## D4.3 Summary

## Summary

- ▶ **inference**: derivation of additional constraints that are implied by the known constraints
- ↔ **tighter equivalent** constraint network
- ▶ **trade-off** search vs. inference
- ▶ inference as **preprocessing** or **integrated** into backtracking
- ▶ cheap and easy inference: **forward checking**
  - ▶ remove values that conflict with already assigned values